**Государственное образовательное учреждение высшего профессионального образования
Поволжский государственный университет телекоммуникаций и информатики**

А.Ю. Гребешков

Микропроцессорные системы и программное обеспечение в средствах связи

Рекомендовано
в качестве учебного пособия по специальностям
210406 – «Сети связи и системы коммутации»
210401 – «Физика и техника оптической связи»
направления 210400 «Телекоммуникации»

**ПГУТИ
Самара**

2009

Рецензия Московского технического университета связи и информатики, зарегистрирована в Московском государственном университете печати (МГУП) 10.10.2008, рег. №155.

УДК 621.395

Гребешков А.Ю. Микропроцессорные системы и программное обеспечение в средствах связи: Учеб. пособие.– Самара, ПГУТИ, 2009 г. – 298 с.:илл.

ISBN

В настоящем учебном пособии рассматриваются вопросы применения микропроцессорной техники и программного обеспечения в средствах связи. Приводится базовая информация по технике микропроцессорных средств, архитектуре и способам построения современных микропроцессоров. Рассматриваются вопросы классификации, построения и особенностей применения телекоммуникационного программного обеспечения различного назначения. Подробно рассматриваются операционные системы реального времени. Даются общие сведения по архитектуре, характеристикам и способам применения специализированных микропроцессоров – сетевых процессоров, процессоров ввода/вывода, процессоров цифровой обработки сигналов. На примере коммутационной системы EWSD рассматриваются вопросы техники микропроцессорных систем и особенности программного обеспечения реального времени. Дополнительно приводятся сведения по развитию архитектур микропроцессоров, в частности по многопоточным и многоядерным процессорам.

Учебное пособие подготовлено согласно государственного образовательного стандарта высшего профессионального образования, направление подготовки дипломированного специалиста 210400 «Телекоммуникации» для студентов специальностей 210406 «Сети связи и системы коммутации» и 210401 «Физика и техника оптической связи», аспирантов, работников отрасли «Связь», интересующихся вопросами применения микропроцессорных и программных систем.

Таб. 7. Ил. 74, Библиогр.: 41 назв.

Содержание

Введение

Настоящее учебное пособие разработано согласно требований государственного образовательного стандарта высшего профессионального образования, направление подготовки дипломированного специалиста 210400 «Телекоммуникации» по специальным дисциплинам 210406 – «Сети связи и системы коммутации», 210401 – «Физика и техника оптической связи». Основной целью данного учебного пособия является систематизация имеющейся учебной и научной информации в части, касающейся архитектуры микропроцессоров и программного обеспечения, особенностей применения микропроцессорной техники и программного обеспечения в средствах связи.

Данное учебное пособие охватывает следующие темы :

- архитектура и основные технические характеристики микропроцессоров различных типов;
- организация ввода-вывода: программное управление вводом-выводом, каналы прямого доступа в память;
- назначение и виды прерываний;
- многопроцессорные системы: архитектура, способы связи, комплексирование;
- программное обеспечение, операционные системы реального времени;
- управляющие комплексы узлов коммутации – основные требования, тенденции развития, архитектура и комплексирование.

Дополнительно рассматриваются современные тенденции развития микропроцессорной техники и программного обеспечения.

1. Микропроцессорные системы

1.1 Базовые понятия и определения

Под ***микропроцессорной системой*** в рамках настоящего учебного пособия понимается совокупность микропроцессоров и выполняемых ими программ, использующих для обработки данных определённые методы, процедуры, алгоритмы и информационные технологии.

Алгоритм работы, согласно А.А. Маркову – точное предписание, определяющее вычислительный процесс, идущий от варьируемых (изменяемых) исходных данных к искомому результату.

Информационная технология (ИТ) – процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов.

Центральным компонентом микропроцессорной системы является микропроцессор. Под ***компонентом*** здесь и далее понимается часть какого то объекта или системы, выделяемая по определенному признаку или совокупности признаков и рассматриваемая как единое целое.

Микропроцессор (МПр) представляет собой синхронное цифровое вычислительное устройство обработки данных, функционирующее на основе загружаемой программы для электронно-вычислительных машин (ЭВМ). Микропроцессор является функциональной частью вычислительной машины или системы обработки информации, предназначенной для интерпретации программ. Физически микропроцессор выполняется в виде одной или нескольких больших интегральных схем. Под ***обработкой данных*** здесь и далее понимается последовательность систематических операций, производимых над данными, представляющими предназначенную для обработки информацию. ***Информация*** – сведения (сообщения, данные) независимо от формы их представления. ***Данные*** – информация, представленная в виде, пригодном

для обработки автоматическими средствами при возможном участии человека (согласно ГОСТ 15971 – 90). ***Вычислительная машина*** – совокупность технических средств, создающая возможность проведения обработки информации и получения результата в необходимой форме. ***Интерпретация*** – процесс, при котором специальная программа осуществляет последовательное преобразование и выполнение каждого оператора языка программирования, на котором написана загружаемая в МПр программа.

Под ***программным обеспечением (программой для ЭВМ)*** понимается объективная форма представления совокупности данных и команд, предназначенных для функционирования средства связи и иных компьютерных устройств с целью получения определенного результата. Программа реализует определенный алгоритм, использует информационную технологию обработки данных. В ходе выполнения программы используются различные форматы и способы обмена управляющими командами и сигналами. ***Программирование*** – раздел прикладной математики, разрабатывающий методы использования вычислительных машин для реализации алгоритмов. ***Сигнал*** – это физический процесс, распространяющийся в канале связи и несущий сообщение о некотором событии, состоянии объекта наблюдения или контроля, код команды управления.

Язык программирования – формальная знаковая система, предназначенная для записи программ для ЭВМ, также используемая для связи человека с цифровым вычислительным устройством. Язык программирования предназначен для описания данных (информации) и алгоритмов (программ) их обработки на вычислительном устройстве.

В современных средствах связи микропроцессорные системы находят самое широкое применение, что обусловлено повсеместным применением цифровых средств связи с программным управлением. Для функционирования таких средств связи применение микропроцес-

сорной техники является единственно возможным способом реализовать необходимые функции [6].

На физическом уровне компоненты процессора могут изготовляться отдельно, а затем устанавливаться на основание микросхемы в процессе ее изготовления. Основой современной элементной базы для изготовления МПр является твердотельный планарный транзистор на кремниевой подложке, изготавливаемый по диффузионной технологии. Полупроводниковая структура формируется на поверхности кристалла кремния в 15..25 слоях из поликремния, металла, диэлектрика. Технологический процесс изготовления такого транзистора представляет собой поэтапную (по областям транзистора) диффузию примесей в кристаллическую структуру кремниевой подложки при температурах порядка +800°С с очень жесткими ограничениями на градиент температуры. Для получения приемлемых характеристик градиент должен иметь порядок ± 0,1°/час [25]. В настоящее время существуют МПр, выполненные в виде единичных кристаллов большой площади (до 300… 400 мм^2) или в виде сборки из нескольких больших (сверхбольших) интегральных схем (СБИС); общая тенденция – увеличение площади кристаллов СБИС в среднем на 10%..15% в год. Физический размер кристалла микропроцессора (чипа) определенным образом связан с технологической или проектной нормой производства т.е. со значением максимального смещения границы топологического элемента при изготовлении транзистора. Например, площадь кристалла равна 20-140 мм^2 при технологической норме 90 нм, а при технологической норме 65 нм площадь кристалла составляет 80-100 мм^2. Расстояние между транзисторами на кристалле МПр сейчас составляет одну десятитысячную толщины человеческого волоса. По точности изготовления современные транзисторы равносильны тому, чтобы провести автомобиль по прямой длиной в 650 км с отклонением от оси менее 2,5 см. (все данные компании Intel).

Тенденция такова, что количество элементов на единицу площади и сама площадь кремниевой пластины МПр увеличиваются. Это связано с тем что конструкция, функции и возможности МПр постоянно совершенствуются и развиваются. Характерно в этой связи наличие эмпирического «закона Мура». Гордон Мур, один из основателей компании Intel (США) предположил, что число транзисторов на кристалле будет удваиваться каждые 24 месяца (скорректированное утверждение, сделанное в 1979-1980 г.г.). Этот «закон» с определенными оговорками, поправками и уточнениями в целом оказался справедливым, хотя существует и критика некоторых утверждений Мура (см. *А. Скробов,* «Закон Мура», 2005 г. Режим доступа [http://cs.usu.edu.ru/study/moore/]). Кроме того, рост производительности МПР не всегда напрямую зависит от количества и сложности транзисторов на кристалле МПр.

Усложнение производства сказывается на стоимости оборудования для производства новых МПр. Существует менее известный «второй закон Мура», введённый в 1998 году Юджином Мейераном, который гласит, что стоимость фабрик по производству микросхем экспоненциально возрастает с усложнением производимых микросхем. Так, стоимость фабрики, на которой корпорация Intel производила микропроцессоры Pentium по 0,6-микронной технологии с 5,5 млн транзисторов на кристалле, составляла 2 млрд. долл. США; стоимость завода Fab32 по производству процессоров на базе 45-нм техпроцесса, составила уже 3 млрд долларов США.

Микропроцессор включает набор различных аппаратных компонентов, которые функционально объединены в виде ***центрального вычислительного (процессорного) устройства ЦПУ*** (central processing unit, CPU). Согласно ГОСТ 15971–90, ЦПУ, которое также называется центральным процессором, выполняет в данной вычислительной машине или системе обработки информации основные функ-

ции по обработке данных и управлению работой других частей вычислительной машины или системы.

К обработки данных относится последовательность операций объединения, проверки, арифметические операции, в большинстве случаев определенные/ограниченные во времени. ***Операция*** – однозначно определенное действие, составляющее выполнение команды или реакцию на определенные условия. В программировании операция – действие, производимое над данными, переменными, константами, функциями. В вычислительных машинах различают операции обработки данных или вычислительные операции, операции управления и операции над командами программы (операции переадресации). В группе вычислительных операций можно выделить:

Арифметические операции – сложение, вычитание, умножение, деление), выполняемые в соответствии с правилами арифметики; результатами арифметических операций, как правило, являются числа с фиксированной или плавающей запятой, поля переменной длины или двоичные, шестнадцатеричные числа.

Логические поразрядные операции – логические сложение, умножение, равнозначность, отрицание равнозначности — сравнение), выполняемые в соответствии с правилами алгебры логики; результатами являются отдельные разряды исходных величин, представленные в двоичной форме;

Логические операции – поиск, выборка, упорядочивание, группировка, выполняемые над отдельными разрядами данных или совокупностями разрядов.

К операциям управления, обеспечивающим выполнение программы и работу устройств вычислительной машины, относят передачу управления, организацию циклов, обращение к внешним устройствам, пересылку данных, прерывание основной программы, изменение режима работы устройств (пуск, останов, чтение, запись и т.п.).

Центральное процессорное устройство выполняет машинные команды, считываемые из физической памяти. ***Машинная команда*** – оператор языка программирования, опознаваемый и выполняемый аппаратными средствами микропроцессора. Машинная команда описывает элементарную операцию, которую должен выполнить МПр. ***Физическая память –*** аппаратная часть микропроцессорной системы, в которую могут записываться и храниться данные и команды, и при необходимости – считываться. ***Физическая оперативная память*** – память, в которой размещаются данные, обрабатываемые командами, собственно команды в ходе непосредственного выполнения (интерпретации) программ.

На аппаратном уровне центральное процессорное устройство выполняет обработку данных с помощью арифметико-логического устройства (АЛУ) и встроенных устройств кратковременного хранения данных – регистров. В состав ЦПУ также входит блок декодирования команд или устройство управления, которое преобразует машинные команды, загруженные в процессор из физической памяти, во внутренние инструкции (микрокоманды) и далее – в функциональные/физические сигналы управления отдельными компонентами процессора – логическими схемами. Также ЦПУ поддерживает встроенную систему прерываний выполнения последовательности операций (инструкций), что позволяют изменять порядок выполнения машинных команд. При создании микропроцессорной системы, ЦПУ конструктивно дополняется компонентами управления физической памятью, устройствами ввода–вывода данных.

Ввод – передача данных от внешнего по отношению к физической оперативной памяти источника информации в физическую оперативную память. ***Вывод*** – процесс передачи данных от физической оперативной памяти к внешним запоминающим устройствам или к внешней, по отношению к МПр, аппаратуре.

Рассмотрим некоторые из описанных функций ЦПУ подробнее.

Центральное процессорное устройство поддерживает встроенную систему команд обработки данных в виде набора доступных микроопераций над данными, выполняемых аппаратными средствами. Микрооперация выполняется за 1 такт т.е. за одну регистровую передачу. Также ***такт*** – минимальный рабочий интервал, в течение которого совершается одно элементарное действие [8]. Например, при выполнении команды сложения с плавающей запятой можно выделить следующие микрооперации: вычитание порядков, выравнивание мантисс, сложение, нормализация. С помощью микроопераций выполняются передача, запоминание и преобразование кодов команд с помощью пересылки между регистрами МПр через логические схемы ЦПУ. ***Код команды*** – способ отображения информации, задаваемый соответствием между необходимым действием и сигналом, с помощью которых это действие фиксируется. Микрооперации также соответствует информационный код (микрокод).

При выполнении микрооперации для организации необходимого срабатывания логических схем ЦПУ формируется набор управляющих сигналов; код набора таких сигналов называется микрокомандой. ***Микрокоманда*** – это команда для управления логическими схемами, которая вызывает и инициализирует выполнение микрооперации для исполнения операции, предписанной машинной командой. Можно выделить следующие основные микрокоманды: выборка команды из памяти или регистра, расшифровка полей команды, выборка (чтение) необходимых операндов, выполнение команды, сохранение результатов в регистр или в память. ***Операнд*** – машинный код (данные, номер регистра), над которым при исполнении микрокоманды производится операция, указанная в машинной команде.

Микрокоманда, также называемая инструкцией МПр, определяет наименование (код) операции над данными и место нахождения данных. Место нахождения данных может быть указано в явном виде, на-

пример как номер регистра. Логическая последовательность микрокоманд, соответствующая исполнению машинной команды для осуществления требуемой операции, называется ***микропрограммой*** [17].

Микрокоманды и микропрограммы хранятся в физическом запоминающем устройстве, ЗУ или в оперативной памяти, ПЗУ. ***Запоминающее устройство (ЗУ)*** – компонент микропроцессорной системы или самостоятельное устройство, предназначенное для записи, хранения и предоставления доступа (считывания) информации. ***Запись информации*** – занесение информации в память или ЗУ для хранения.

Хранение информации – процесс существования/передачи информации во времени без изменения её вида и содержания.

Считывание – процесс преобразования физического состояния запоминающей среды, отображающей хранимую информацию, в информационные сигналы стандартной формы. Эти информационные сигналы позволяют восстановить исходную информацию/сообщение, существовавшую на момент записи.

Считывание микрокоманды из адресованной ячейки запоминающего устройства означает появление на выходах ЗУ определенного уровня сигнала (высокий или низкий уровень). Соответственно, формируется двоичный код команды, который поступает на входы МПр и рассматривается уже как функциональный сигнал управления. В результате срабатывают соответствующие логические схемы внутри МПр, которые согласно поступившей микрокоманде выполняют требуемую микрооперацию. Описанная схема соответствует микропрограммному управлению, также именуемому управлением с хранимой/гибкой логикой управления. Если последовательность исполнения операций задаётся набором микросхем, вырабатывающих определенные функциональнее сигналы для выполнения микроопераций, то это управление с жёсткой логикой.

Арифметико-логическое устройство позволяет выполнять вычислительные и логические операции на аппаратном уровне (на уровне схемной логики), для чего имеет в своем составе сумматор, схемы базовых логических операций, а также схемную логику, обеспечивающую перестройку с одной операции на другую. ***Регистры*** представляют собой функционально–ориентированные ячейки памяти ёмкостью, например 16..32 бита; данные регистра могут быть обработаны за 1…3 такта работы процессора; содержимое регистра записывается или считывается последовательно, параллельно или циклически со сдвигом через специализированный регистр – бит (флаг) переноса. ***Бит*** – двоичная единица представления данных.

В случае создания МПр в виде сборки из нескольких интегральных схем, в едином корпусе можно размещать не только несколько АЛУ, но и контроллеры управления физической памятью; буферную память небольшой ёмкости, недоступную для пользователя, автоматически используемую МПр для ускорения операций с информацией (кэш-память), блоки предсказания ветвления, служебные регистры различного назначения.

Буферная память – память, предназначенная для промежуточного хранения данных, при обмене ими между устройствами вычислительной машины, работающих с разными скоростями.

Некоторые типы современных МПр могут конструктивно объединять на одном кристалле не только устройства управления внешней физической памятью, но и саму физическую память, а также устройства ввода/вывода. В результате появляется ***однокристальная микро–ЭВМ.***

В ГОСТ Р 51840–2001 особо выделяют ***программируемый контроллер*** – цифровую электронную систему, предназначенную для применения в промышленных условиях. Программируемый контроллер использует программируемое запоминающее устройство для внутреннего

хранения ориентированных на пользователя инструкций, для выполнения логических операций, операций упорядочивания, отсчёта времени, математических действий, управления через аналоговые или цифровые входы и выходы различными устройствами или процессами.

Как уже отмечалось, ЦПУ с помощью АЛУ выполняет основные вычислительные и логические операции. Таким образом, с точки зрения программы для ЭВМ, микропроцессор представляет собой арифметико-логическое вычислительное устройство [4,5].

Далее в главе 1 настоящего учебного пособия более подробно рассматриваются основные функциональные компоненты микропроцессора, система команд и форматы данных.

В главе 2 рассматривается программное обеспечение средств связи и коммутации, в основном операционные системы реального времени.

В главе 3 рассматривается применение микропроцессорных систем и программного обеспечения на примере системы коммутации EWSD.

В главе 4 рассматриваются особенности конструкции и обработки данных в специализированных микропроцессорах.

В главе 5 рассматриваются тенденции развития микропроцессоров.

1.2 Архитектура микропроцессора

Под ***архитектурой микропроцессора*** понимается совокупность принципов и подходов, структурных, технических, технологических решений, определяющих концепцию взаимосвязи элементов процессора. Архитектура включает компоненты логической, физической, программной организации МПр. Архитектура охватывает набор команд, поддерживаемый процессором, регистры процессора, способы хранение и обработки данных в памяти процессора. В последнее время часто можно

встретить термин ***микроархитектура процессора*** – это реализация архитектуры на уровне аппаратных (полупроводниковых) компонентов. Как правило, микроархитектура описывается в виде функциональной блок–схемы. Архитектура микропроцессора прямо или косвенно определяет основные принципы функционирования процессора, его наблюдаемые характеристики и области практического применения [14,15,18,26]. В 1946 году американским учёным фон Нейманом (von Neumann) была предложена архитектура вычислительной машины и процессора, ныне относящаяся к числу классических. Эта архитектура была разработана на основе следующих базовых принципов:

Принцип программного управления – программа для ЭВМ, которую обрабатывает ЦПУ, состоит из последовательности машинных команд, выбираемых из памяти с помощью счётчика команд. Счётчик – регистр, значение которого либо автоматически увеличивается на единицу, либо его состояние меняется принудительно при выполнении команд условного или безусловного перехода.

Принцип однородности памяти – программы и данные хранятся в одной и той же памяти. Над кодами команд можно выполнять те же действия, что и над кодами данных. В процессе выполнения можно изменять последовательность выполнения отдельных частей программы, например организовывать циклы, переход к подпрограммам, возврат из подпрограмм.

Подпрограмма – часть программы для ЭВМ, имеющая самостоятельное значение и применяемая при решении различных задач одного класса. Подпрограмма, как правило, описывает самостоятельный этап вычислительного процесса и может быть использована неоднократно в одной или нескольких различных программах для ЭВМ. Типичные подпрограммы используются для вычисления элементарных функций, управление вводом-выводом.

Принцип адресности – основная физическая память процессора состоит из пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка. Отсюда следует возможность давать имена областям памяти, так, чтобы к хранимым в них данным можно было бы впоследствии обращаться. Каждая команда загружаемой в микропроцессор программы для ЭВМ хранится в ячейке физической памяти с уникальным адресом.

Адрес – цифровое обозначение ячейки памяти вычислительной машины.

Архитектура фон Неймана предусматривает, что программа для ЭВМ и обрабатываемые данные загружаются в физическую оперативную память с помощью устройств ввода/вывода. При этом вся информация, поступающая в процессор, кодируется с помощью двоичных сигналов. В результате аппаратная часть процессора является неизменной величиной, а вот программа – в силу возможности модификации – величина переменная.

В соответствии с изложенными принципами и подходами, фон Нейман предложил включить в состав вычислительной машины АЛУ, физическую память, устройства ввода-вывода. Кроме того, в состав ЦПУ включается :

Устройство управления (УУ) [8] – считывает команды из ОЗУ и организует их выполнение в соответствии с порядком выполнения команд, заданных программой. Таким образом, УУ в строгой последовательности в рамках тактовых и цикловых временных интервалов работы микропроцессора осуществляет:

- выборку команды;
- интерпретацию команды с целью анализа формата, служебных признаков и вычисления адреса операнда (операндов);
- установление номенклатуры и временной последовательности всех функциональных управляющих сигналов;

- генерацию управляющих импульсов и передачу их на управляющие шины функциональных частей МПр и вентили между ними;
- анализ результата операции и изменение своего состояния так, чтобы определить месторасположение (адрес) следующей команды.

Регистры процессора (регистры общего назначения) – программно-доступные ячейки памяти в составе ЦПУ для сверхоперативного хранения данных длиной 8, 16, 32, 64 или 128 бит. Регистры могут быть универсальными или специализированными (регистры операции ввода-вывода, регистр таймера, регистр счётчика, использование регистров только в командах умножения-деления). В состав ЦПУ также может включаться быстродействующее постоянное запоминающее устройство (ПЗУ). В результат вычислительная система, т.н. «машина фон Неймана», имеет видна рис. 1.1.

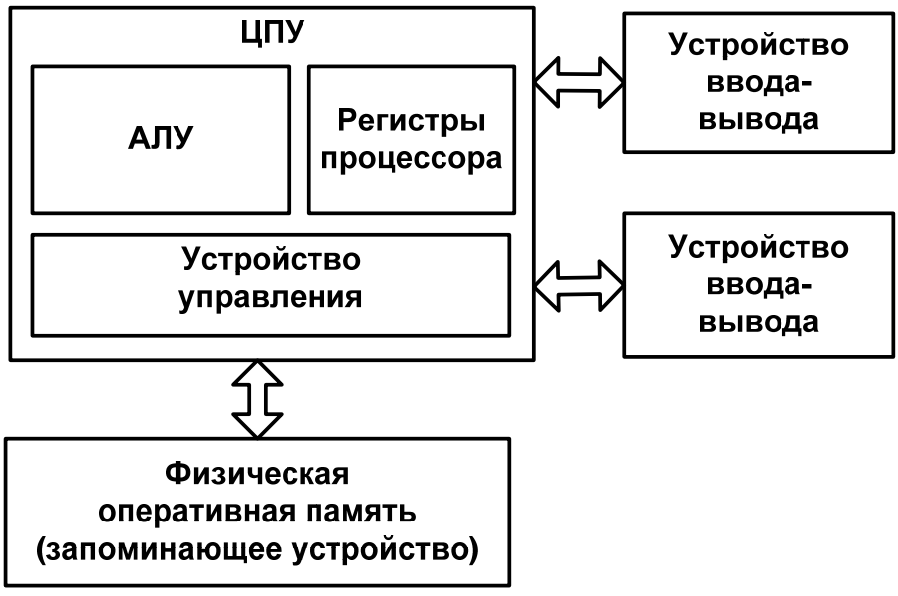


Рис. 1.1 – Состав вычислительной машины фон Неймана

Физическая оперативная память является матрицей, состоящей из m строк с n двоичными разрядами в каждой строке. Строка имеет

уникальный идентификатор – адрес. Один разряд на пересечении строки и столбца способен хранить информацию размером (объёмом) в 1 бит и называется ***запоминающим элементом***. Совокупность из 8, 16 запоминающих элементов образуют ячейку памяти, способную хранить 1,2 байта информации.

По физической природе запоминающие элементы могут быть полупроводниковыми, оптическими, ферромагнитными.

Для организации обмена внешние устройства, память и микропроцессор физически соединяются электрическими цепями (проводниками), формирующими физические связи устройств. Эти проводники называются линиями, линии формируют интерфейсы между устройствами. ***Интерфейс*** есть граница и способ адаптации между двумя взаимодействующими системами (устройствами), определенный общими функциональными, конструктивными характеристиками, требованиями к протоколу обмена. Также интерфейс – совокупность средств и правил, обеспечивающих взаимодействие устройств вычислительной машины или системы обработки информации и (или) программ.

Часть линий, сгруппированных по функциональному назначению, называется ***шиной.*** Функционально различают внутренние шины и внешние шины.

К внутренним шинам относятся :

- локальная шина (VL-bus, первичная PCI-шина) – подключена к контактам МПр и работает на его частоте, может соединять МПр с ОЗУ или соединять МПр с контроллером памяти/контроллером общей системной шины;
- общая системная шина (вторичная PCI-шина, PCI-Express) – соединяет МПр с ограниченным числом высокоскоростных внешних устройств через мост/шлюз (в случае PCI-Express – через коммутатор), а также соединяет высокоскоростные устройства с оперативной памятью ОЗУ;

- шина расширения (ISA, EISA) – при наличии шины PCI соединяет общую системную шину с относительно низкоскоростными внешними устройствами; ранее ISA и EISA рассматривались как стандарты для общей системной шины.

К внешним шинам относятся шины ввода/вывода – к ним подключаются внешние устройства с различными интерфейсами, такими как SCSI, Serial ATA, Serial Attached SCSI (SAS), USB, FireWire/IEEE 1394. В свою очередь шины ввода-вывода через специальное объединяющее устройство – мост – соединяются с общей системной шиной. Допускается, что при наличии специальных адаптеров PCI, устройства могут подключаться к общей системной шине непосредственно.

Наличие тех или иных видов шин, их объединение зависит от конструкции микропроцессорной системы. Различают одномагистральную и многомагистральную структуру. В одномагистральной структуре МПр, ОЗУ, ВУ подключаются к одной шине. В многомагистральной структуре МПр, ОЗУ и мост соединяются локальной шиной, высокоскоростные ВУ с МПр и ОЗУ соединяются через мост и общую системную шину , а прочие внешние устройства сначала подключаются к шине ввода-вывода, а шина ввода-вывода подключается к шине расширения или к общей системной шине.

Шины отличаются друг от друга проводностью, скоростью передачи. В шинах выделяют линии данных, линии адресов, линии управления, линии прерывания, используемые для передачи соответственно данных, адресов, сигналов управления и информации, включая прерывания.

Шина данных (data bus) — двунаправленная (дуплексная) шина, предназначена для передачи данных, закодированных двоичным кодом, между компонентами МПр, устройствами памяти и внешними устройствами. Разрядность шины (количество линий связи) определяет скорость и эффективность информационного обмена, а также макси-

мально возможное количество команд. Обычно шина данных имеет 8, 16, 20, 32 или 64 линии/разряда. За один цикл обмена по 64-разрядной шине может передаваться 8 байт информации, а по 8-разрядной — только один байт. Под ***машинным циклом*** здесь и далее понимается время, в течении которого производится выборка двух операндов из регистров, выполнение операции в АЛУ и запоминание результатов в регистре. Машинный цикл выполняется в течении нескольких тактовых импульсов (тактов), поступающих от генератора тактовой частоты МПр.

Шина адреса (address bus) — устройство, предназначенное для передачи кода адреса ячейки физической памяти. Разрядность шины адреса определяет максимально возможное количество адресов физических ячеек и, следовательно, максимально возможный размер хранимой программы и объем запоминаемых данных. Количество адресов, обеспечиваемых шиной адреса, определяется как 2^N, где N — количество разрядов шины. Например, 16-разрядная шина адреса обеспечивает адресацию (обращение) к 65 536 уникальным адресам ячеек физической памяти. Разрядность шины адреса обычно кратна 4 и может достигать 32 и даже 64. Шина адреса может быть однонаправленной (тогда шиной всегда управляет только микропроцессор) или двунаправленной (тогда процессор может временно передавать управление шиной другому устройству, например контроллеру внешнего устройства).

Вся совокупность линий интерфейса между компонентами микропроцессорной системы называется ***магистралью.*** Условно можно выделить две магистрали: магистраль информационного канала и магистраль управления информационным каналом (см. Писарев А.П. Интерфейсы АСОИУ: Курс лекций. – Пенза: Пензенский гос. ун-т, 2007. - 68 с. Режим доступа [http://window.edu.ru/window/library?p_rid=53991]).

По ***магистрали информационного канала*** передаются коды данных, коды адресов, коды команд и коды состояний устройств. Ана-

логичные наименования присваиваются соответствующим шинам, реализующим интерфейс.

Коды данных представляют информацию, относящуюся к содержанию процессов в программной системе; здесь используется двоичное кодирование в формате машинного кода. Коды данных передаются по шине данных.

Коды адресов предназначены для выбора, с помощью магистрали, заданных устройств или ячеек памяти для записи или считывания данных. Обычно для адресации используется двоичный номер объекта; есть варианты, когда каждому устройству выделяется отдельная линия адреса. Коды адресов передаются по шине адреса.

Коды команд используются для управления функционированием устройств и обеспечения сопряжения между ними. Существует минимально необходимый набор команд, который может быть расширен пользователем за счет резервных полей в кодах команд. Различают команды управления обменом информации между устройствами, команды изменения состояния и изменения режимов работы. К наиболее распространенным командам относятся: «Чтение», «Запись», «Конец передачи», «Запуск».

Коды состояния представляют собой сообщения, описывающие состояния устройств сопряжения. Коды формируются в ответ на действия команд или являются отображением состояний функционирования устройства, таких как «Занятость устройства», «Наличие ошибки», «Готовность устройства» к приему или передаче информации и т. п. В большинстве случаев коды данных, адресов, команд и состояний передаются по шинам с разделением времени за счет мультиплексирования шин. Это достигается введением дополнительных линий для обозначения типа передаваемой информации, называемых ***линиями идентификации.*** Их применение позволяет существенно сократить общее

число линий информационной магистрали, однако при этом происходит снижение быстродействия передачи информации.

Магистраль управления информационным каналом по своему функциональному назначению делится на ряд шин:

- шина управления обменом;
- шина передачи управления;
- шина прерывания;
- шина специальных управляющих сигналов.

Шина управления обменом включает в себя линии синхронизации передачи информации. В зависимости от принятого принципа обмена (асинхронного, синхронного) число линий может изменяться от одной до трех. Асинхронная передача происходит при условии подтверждения приемником готовности к приему и завершается подтверждением о приеме данных. При синхронной передаче темп выдачи и приема данных задается регулярной последовательностью сигналов.

Шина передачи управления выполняет операции приоритетного занятия магистрали информационного канала. Наличие этой шины определяется тем, что взаимодействие в большинстве интерфейсов выполняется по принципу «ведущий-ведомый», при котором «ведущее» устройство может брать управление шиной на себя в определенные моменты времени. При наличии в системе нескольких устройств, способных выполнять функции «ведущего», возникает проблема приоритетного распределения ресурсов шины (арбитража).

Шина прерывания применяется в основном в машинных интерфейсах ЭВМ и программно-модульных системах. Основная ее функция — идентификация устройства, запрашивающего сеанс обмена информацией. Идентификация состоит в определении контроллером (процессором ввода-вывода) исходной информации о запрашиваемом устройстве. В качестве информации об устройстве используется адрес источ-

ника прерывания либо адрес программы обслуживания прерывания (вектор прерывания).

Шина специальных управляющих сигналов включает в себя линии, предназначенные для обеспечения работоспособности и повышения надежности устройств интерфейса. К этим линиям относятся: линии питания, контроля источника питания, тактирующих импульсов, защиты памяти, общего сброса, контроля информации и т. п.

Среди перечисленных сигналов магистрали управления наиболее часто используются следующие :

RD (read) – сигнал чтения из памяти;

WR (write) – сигнал записи в память;

MREQ (memory request) – сигнал инициализации памяти МПр;

IORQ (input – output request) – сигнал инициализации портов ввода-вывода;

READY – сигнал готовности;

RESET – сигнал сброса.

Способ организации магистрали управления информационным каналом не оказывает существенного влияния на архитектуру процессора.

Одним из главных признаков для формирования различных архитектур МПр является организации памяти микропроцессора. Здесь выделяют две основные архитектуры МПр :

- архитектура фон Неймана с общей памятью;
- гарвардская архитектура, предложенная Говардом Айкеном (Howard Aiken) с разделяемой памятью.

Архитектура фон Неймана с общей памятью представлена на рис. 1.2.

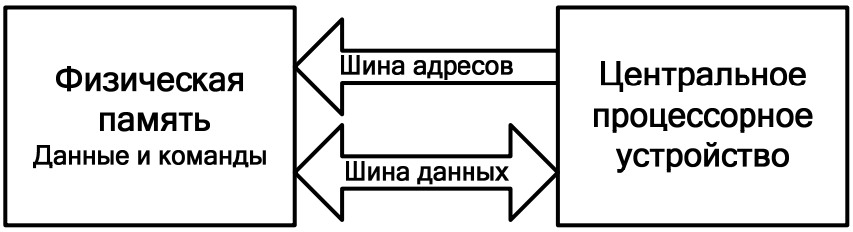


Рис. 1.2 – Архитектура процессора с общей памятью

Машина фон Неймана является наиболее универсальной по способу применения и отличается гибкостью при использовании различных программных средств. Недостатком машины фон Неймана можно считать некоторое снижение быстродействия для поиска нужной ячейки в общей памяти данных и команд. Это обусловлено тем, что быстродействие ЦПУ в несколько раз больше быстродействия физической памяти.

Гарвардская архитектура представлена на рис. 1.3 :

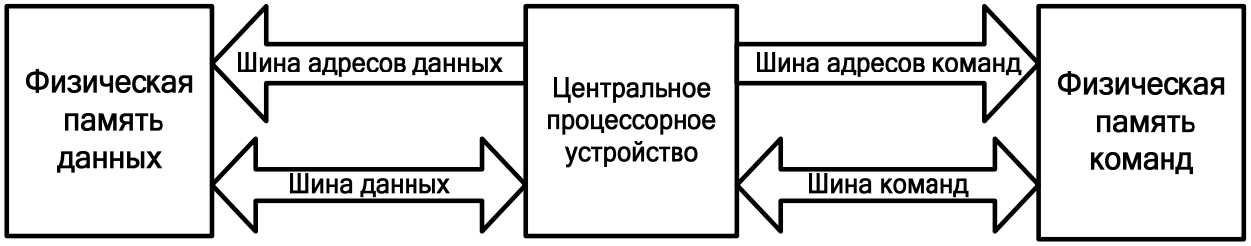


Рис. 1.3 – Гарвардская архитектура процессора с разделяемой памятью

Основной особенностью гарвардской архитектуры является использование раздельных адресных пространств и раздельных физических областей памяти для хранения команд и хранения данных. ***Память данных*** представляет собой функциональную часть вычислительной машины или системы обработки информации, предназначенную для приема, хранения и выдачи данных. Соответственно ***память команд*** предназначена для приема, хранения и выдачи команд, объединенных в программу для ЭВМ.

Как уже говорилось, информация для МПр в двоичном коде соответствует определенным уровням напряжения на внешних контактах

его корпуса. Каждый контакт корпуса МПр передаёт сигнал, соответствующий значению одного бита, и процессору нужно различать только две градации напряжения: есть сигнал – нет сигнала, высокий уровень – низкий уровень. По шине адреса сигналы (уровни напряжений) синхронно передаются/поступают на контакты МПр. Однако при появлении различных шин для обмена адресами и данными на кристалле процессора неизбежно увеличивается количество выводов/контактов корпуса, т.е. происходит усложнение конструкции процессора. Этой проблема решается использованием общей шины данных и шины адреса для всех внешних данных. Внутри процессора проблема решается использованием шины данных, шины команд и двух шин адреса. Такую концепцию называют ***модифицированной Гарвардской архитектурой.***

Гарвардская архитектура начала интенсивно использоваться только в конце 1970-х годов, когда началось интенсивное применение цифровых сигнальных процессоров. Причиной появления интереса к гарвардской архитектуре было то, что в цифровых сигнальных процессорах необходимый объем памяти данных МПр, используемый для хранения промежуточных результатов, как правило, на порядок меньше требуемого объема памяти программ. Следует отметить, что при цифрой обработке сигналов зачастую требуется выбрать три составляющие — два операнда и команду, например в БПФ–фильтрах. Для этого используется т.н. ***кэш-память.*** В кэш-памяти может храниться команда — при этом обе шины остаются свободными, и появляется возможность передать два операнда одновременно. Использование кэш-памяти вместе с разделёнными шинами получило название «Super Harvard Architecture» (SHARC), ***расширенная Гарвардская архитектура***. Особенности организации кэш-памяти рассматриваются в разделе 1.7.

Достоинства гарвардской архитектуры следующие :

1. Применение небольшой по объему памяти данных способствует ускорению поиска информации в памяти, что увеличивает быстродействие МПр.
2. Гарвардская архитектура позволяет организовать параллельное выполнение программ – выборка следующей команды может происходить одновременно с выполнением предыдущей, в результате чего сокращается время выборки команды.

Недостатком гарвардской архитектуры является усложнение архитектуры МПр. Также необходимо генерировать дополнительные управляющие сигналы для памяти команд и памяти данных. В системах коммутации и, в более широком смысле – в средствах связи, применяются как процессоры с архитектурой фон Неймана, так и процессоры с гарвардской архитектурой.

В целом все типы МПр характеризуются тремя основными техническими характеристиками :

Тактовая частота (clock rate) – частота синхронизирующих работу МПр («тактовых») импульсов, которые задаются генератором тактовой частоты, которые регулируют выполнение циклов выборки и исполнения команд. Измеряется тактовая частота в Гц.

Быстродействие, производительность МПр (performance) **– п**оказатель качества МПр, который выражается в миллионах элементарных операций, выполняемых в одну секунду (операций/с). Различают производительность для обработки данных с фиксированной точкой (целые числа) и производительность для обработки данных с плавающей точкой (повышенная точность).

Разрядность МПр – количество бит информации, которое ЦПУ может обработать с помощью одной команды за 1 такт (см. *Денисов К.М.,* режим доступа [http://ets.ifmo.ru:8101/denisov/lec/lec5.htm]). Разрядность микропроцессора определяется разрядностью арифметико-логического устройства, внутренних регистров данных и шины данных. На сегодняшний

день существуют 8-, 16-, 24-, 32- и 64-разрядные микропроцессоры. Для обработки данных с разрядностью большей, чем разрядность микропроцессора, необходимо реализовывать специальные алгоритмы вычислений с повышенной разрядностью. Это может снизить быстродействие МПр.

1.3 Состав и функциональная архитектура управляющих комплексов

Исторически управление средствами связи развивалось следующим образом. На механо-электрических средствах связи с индивидуальными коммутационными приборами (приборами искания), таких как АТС декадно-шагового типа, применялось индивидуальное управление. Здесь сигналы управления – цифры набора номера – поступали непосредственно от абонента на приборы искания. В механо–электрических средствах связи с групповыми приборами искания, такими как многократные координатные соединители, появились групповые управляющие устройства, выполненные по аппаратной логике, в виде последовательно соединенных реле («пирамиды реле») – т.н. схемное управление. Управляющие устройства, такие как групповой искатель, абонентский регистр могли не только принимать и обрабатывать цифры набора номера абонентов, но и обмениваться сигналами занятия, разъединения с помощью передачи электрических сигналов по специальным проводам управления. Такое управление можно охарактеризовать как «жёсткое», реализующее только один алгоритм управления установлением соединения. Впрочем, решение по схемному управлению соответствовало уровню применяемой коммутационной техники. Решительный переворот в развитии управляющих комплексов средств связи произошёл с началом внедрения электронных и квазиэлектронных коммутационных элементов, программного управления, что потребова-

ло применения вычислительных машин для управления процессами коммутации и передачи.

Под ***управлением*** в настоящем учебном пособии понимается любое изменение состояния средства связи, его компонента, процесса, ведущие к достижению поставленной цели. Целью, здесь является поддержание средства связи в состоянии, соответствующему штатному режиму эксплуатации с показателями, соответствующими паспортным показателям.

Программное управление работой некоторого объекта осуществляется системой автоматического управления (на основе микропроцессора или электронной вычислительной машины) по заданной программе для ЭВМ, в результате чего вырабатываются сигналы, воздействующие на исполнительные органы управляемого объекта для целенаправленного изменения его режима работы или состояния. В средствах связи для реализации системы автоматического управления используется управляющий комплекс средств связи.

Управляющий комплекс (УК) средства связи – совокупность технологически сопряженных управляющих устройств, предназначенных для автоматического управления процессами преобразования сигналов электросвязи, коммутацией и передачей каналов и пакетов, а также для реализации автоматизированных функций технической эксплуатации средств связи, учёта и контроля трафика.

Управляющее устройство *(УУ)* – функционально и конструктивно законченное изделие, вырабатывающее из потока поступающей информации последовательность управляющих сигналов или машинных команд для целенаправленного воздействия на аппаратуру и приборы средства связи. Под ***аппаратурой*** понимаются технические средства определенного класса.

Управляющий комплекс средства связи включает следующие виды устройств управления :

- индивидуальные управляющие устройства (контроллеры);
- групповые управляющие устройства;
- центральное управляющее устройство (см. рис. 1.4).

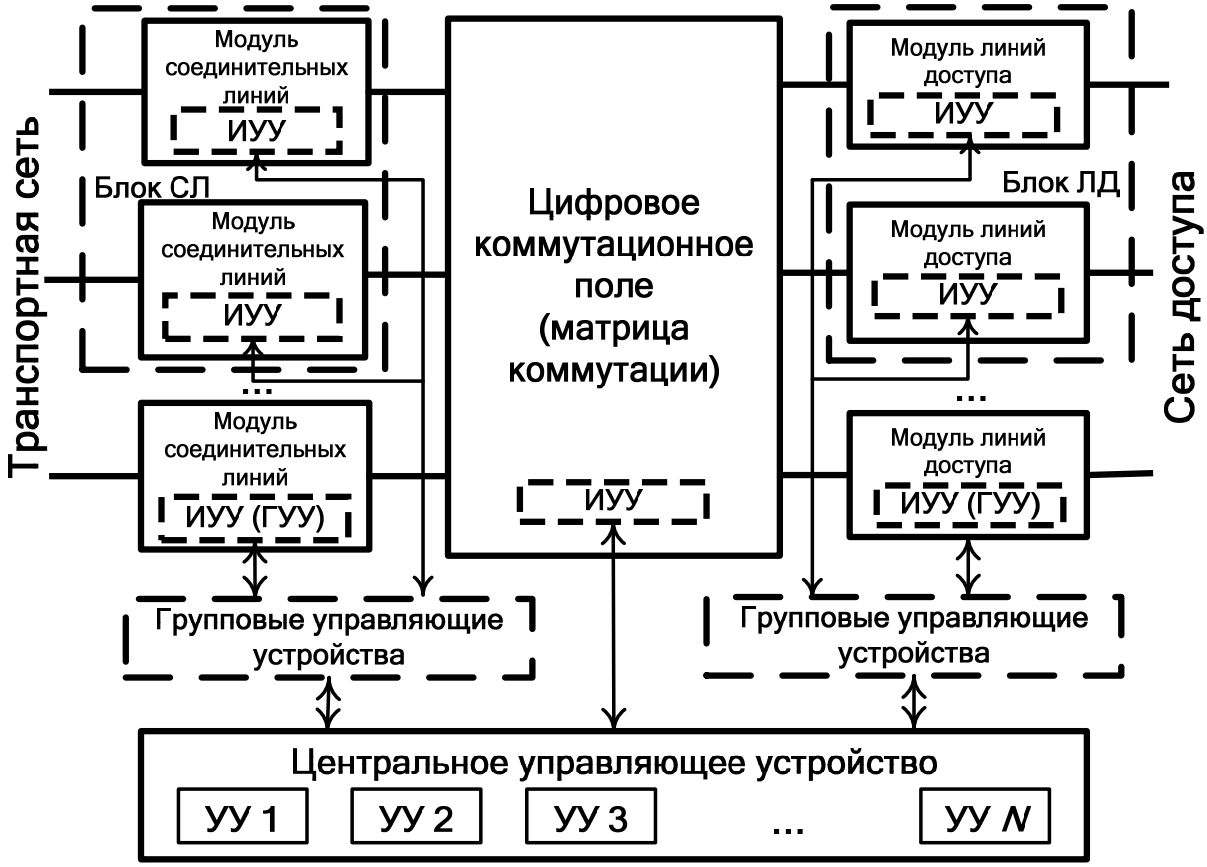


Рис. 1.4 – Общая функциональная схема управляющих устройств средства связи

Под ***модулем*** на рис. 1.4 понимается сборочная единица средства связи, состоящая из одной или нескольких деталей, соединенных на предприятии изготовителе, установленная на монтажном месте (позиции) и предназначенная для обеспечения паспортных функций средства связи, включая функции технической эксплуатации. Модули могут объединяться в блоки с общим функциональным назначением – блок соединительных линий (блок СЛ), блок линий доступа (блок ЛД). На рисунке 1.4 модуль соединительных линий, модуль линий доступа и цифровое коммутационное поле (матрица коммутации) выполняют свои стандартные функции для создания сквозного канала связи между

пользователями услуг электросвязи или организации маршрута коммутации кадров/пакетов в процессе сеанса связи. Одной из основных функций модуля является реализация интерфейса между средством связи и внешней средой.

Интерфейс с внешней средой предоставляет следующие услуги :

- интерфейс человек–компьютер (человек–машина);
- информационный обмен с внешней средой;
- приём/передача сигнала электросвязи.

Для системы передачи, в случае использования схемы рис. 1.4, модули линий доступа могут отсутствовать. Пунктиром на рисунке показаны управляющие устройства, которые могут как присутствовать, так и отсутствовать в конструкции конкретного средства связи.

В современных средствах связи микропроцессоры используются практически во всех аппаратных модулях и блоках систем коммутации и передачи. Это также обусловлено повсеместным применением программного управления [11,25]. Программы управления средствами связи могут загружаться в физическую память микропроцессора управления с внешнего источника (накопитель на жёстком магнитном диске, оптический накопитель).

Программы управления могут применяться в виде замонтированного программного обеспечения firmware, т.е. постоянно храниться в постоянных запоминающих устройствах с возможностью стирания и перезаписи или без такой возможности.

Рассмотрим функции управляющих устройств более подробно.

Индивидуальные управляющие устройства (ИУУ) – предназначены для управления данным модулем. Выполняют ограниченный набор функций, как правило относящихся к управлению физическим и канальным уровнем модели взаимосвязи открытых систем. В частности ИУУ осуществляет :

- отслеживание момента изменения состояния линии или канала/тракта для определения момента занятия, разъединения, поступления новой информации;
- обмен данными по управлению и взаимодействие с другими ИУУ;
- обмен данными по управлению с ГУУ и/или ЦУУ;
- запуск и приём результатов стандартных тестов технического состояния линий, каналов и трактов;
- мультиплексирование и демультиплексирование;
- мониторинг оборудования, самотестирование и самопроверка (в рамках технической эксплуатации).

Часть указанных функций может осуществляться с помощью специальных микросхем/микросхемных наборов, т.е. на основе управления с жёсткой логикой. ИУУ может принимать программную команду управления от ГУУ и на её основе формировать функциональный сигнал управления для аппаратуры связи.

Сигналы управления – это изменения показателей, признаков, переменных величин некоторых физических объектов, используемые в системе управления для передачи информации или для воздействия на объект управления. ***Сигнал управления в средствах связи*** – электромагнитный сигнал, передаваемый между управляющими устройствами средств связи, для целенаправленного воздействия на объект управления. ***Функциональный сигнал управления*** – сигнал, передаваемый управляющим устройством средства связи или МПр, для реализации требуемой функции на объекте управления. ***Функция*** – совокупность действий средства связи или его компонента, направленная на достижение определенной цели.

Программная команда управления – кодированное описание операции или обозначение программы (процедуры), которая должна быть исполнена управляющим устройством в процессе осуществления

требуемой функции средства связи. Программными командами управления обмениваются не МПр, а программные средства управления.

В случае отсутствия в конструкции ИУУ, часть его функций может быть реализована аппаратно, а функции мониторинга, запуска и приёма результатов тестов может быть передана ИУУ или ЦУУ. ИУУ относятся к 1-му уровню управления средством связи.

Групповые управляющие устройства (ГУУ) предназначены для управления одним или несколькими блоками (модулями). Также ГУУ осуществляют координацию и взаимодействие с другими ГУУ. Благодаря развитию микропроцессорной техники и программного обеспечения, современные ГУУ могут самостоятельно вырабатывать управляющие программные команды или функциональные сигналы управления. ГУУ выполняют следующие функции:

- поддержка процедур сетевых и коммуникационных протоколов (запрос–ответ, разбиение и сборка пакетов, анализ заголовков и цифр набора номера);
- обработка данных систем сигнализации;
- анализ ошибок приёма-передачи;
- управление и контроль ИУУ;
- взаимодействие с другими ГУУ при занятии свободных путей/трактов между заданными блоками дл установления соединения или сеанса связи;
- обмен данными по управлению с ЦУУ.

В части функций технического управления и эксплуатации ГУУ выполняют функции тестирование и самопроверки, сбор данных по технической эксплуатации от ИУУ и передача этих данных в ЦУУ. Групповое управляющее устройство не является обязательным для средств связи малой мощности/ёмкости. Групповые управляющие устройства, если они реализованы на достаточно мощных МПр, могут одновременно выполнять функции ИУУ.

В случае отсутствия ИУУ, ГУУ может осуществлять преобразование команд центральных управляющих устройств в функциональные сигналы управления, поступающие на оборудование блоков, на цифровое коммутационное поле (матрицу коммутации). К примеру, на выходе центрального управляющего устройства может быть программная команда управления, а ГУУ преобразует эту команду в электрические импульсы определённой формы и уровня – функциональный сигнал управления – вызывающий срабатывание конкретного прибора в блоке соединительных линий или в блоке линий доступа. ГУУ относятся ко 2-му уровню управления средством связи.

Центральное управляющее устройство (ЦУУ), как правило, включает в себя комплекс управляющих устройств (УУ). Эти УУ могут быть одинаковыми по конструкции, но за счёт различия загружаемых программ для ЭВМ выполняют разные функции. Это ещё одно доказательство в пользу преимуществ программного управления средствами связи.

Центральное управляющее устройство выполняет наиболее сложные, «интеллектуальные», функции управления средством связи и процессами, осуществляемыми средством связи. Например, центральное управляющее устройство выполняет функции маршрутизации сообщений и пакетов, функции технического обслуживания и эксплуатации, функции администрирования доступом абонентом или пользователей, управляет данными о трафике, управляет процессами ввода-вывода с внешних устройств и управляет обменом с персоналом по эксплуатации. В случае отсутствия в конструкции средства связи ГУУ или ИУУ, ЦУУ также выполняет и функции нижестоящих управляющих устройств. ЦУУ относятся к 3-му уровню управления средством связи.

Сначала микропроцессоры применялись только в центральном управляющем устройстве. Впоследствии микропроцессоры, однокристальные микро-ЭВМ (микроконтроллеры) стали применяться в группо-

вых и индивидуальных управляющих устройствах. В настоящее время в составе современного средства связи МПр или однокристальной микро-ЭВМ оборудован практически каждый функциональный блок (модуль). Большое распространение получают управляющие комплексы, в которых МПр в составе ГУУ и ЦУУ обладают сходными техническими характеристиками, хотя функционально выполняют различные задачи. В этом заключается очевидное преимущество программного управления, когда для реализации новой функции управления достаточно изменить загружаемую программу управления.

С точки зрения функций и организации связей между УУ в составе управляющего комплекса, различают централизованную, иерархическую, квази-распределенную и распределенную функциональную архитектуру управляющих комплексов средств связи.

Под ***архитектурой управляющего комплекса*** понимается совокупность принципов и подходов, структурных, функциональных, технических решений, определяющих концепцию взаимосвязи управляющих устройств. Архитектура включает компоненты функциональной, физической, программной организации УК.

В случае ***централизованной функциональной архитектуры УК*** существует чётко выделенное ЦУУ, в состав которого входит основной управляющий процессор или группа процессоров, которые выполняют все функции управления средством связи, как показано на рис. 1.5.

Здесь ЦУУ имеют прямые интерфейсы с коммутационным полем и функциональными блоками, вплоть до абонентского или линейного интерфейса/стыка. Как только от пользователя поступает запрос на установление соединения, сигнал об изменении состояния абонентской линии сразу передаётся в центральный процессор. Далее начинается процесс установления соединения или начинается процесс организация сеанса связи, где все этапы контролируются ЦУУ.

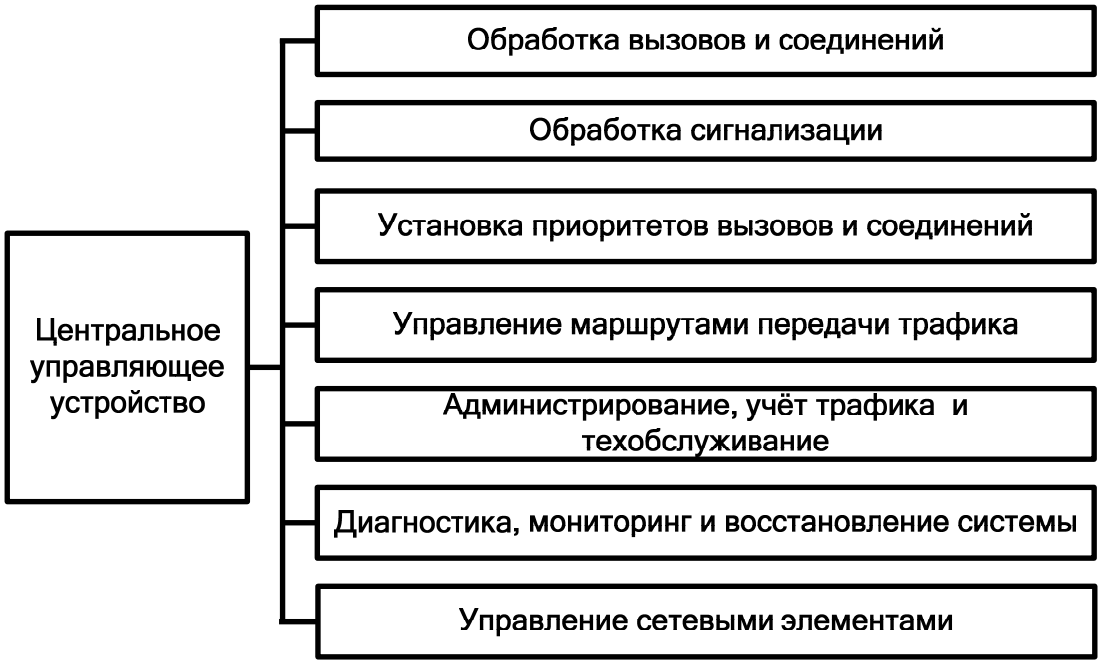


Рис. 1.5 – Централизованная функциональная архитектура УК средства связи

ЦУУ выполняет анализ цифр набора номера, выбор исходящего тракта или направления связи, анализ абонентских данных на предмет разрешения или запрещения оказания тех или иных услуг связи. Данные по каждому этапу установления соединения, состоянию тракта и разъединению записываются в оперативную память ЦУУ. В процессе соединения ЦУУ обрабатывает сигналы внутри- или межстанционной сигнализации в соответствии с установленным стандартным протоколом, например ОКС№7, R 1.5, SIGTRAN.

Установка приоритетов вызовов и соединений означает, что ЦУУ контролирует наличие приоритетов той или иной группы пользователей на основе абонентских данных. Заявки от пользователей с высшим приоритетом обслуживаются ЦУУ в первую очередь.

Управление маршрутами передачи трафика означает возможность выбора ЦУУ прямого или транзитного (альтернативного) маршрута переноса сигнала или трафика по сети связи.

Администрирование и техобслуживание предусматривает, что ЦУУ осуществляет контроль и управление станционными данными,

хранение и обработку информации о правах и паролях персонала эксплуатации, выполняет функции техобслуживания. В частности, в процессе устранения последствий отказов, осуществляется ввод оборудования в рабочий режим, вывод оборудования из рабочего режима, конфигурация оборудования.

Диагностика, мониторинг и восстановление системы предусматривают, что ЦУУ постоянно проводит тестирование аппаратного и программного комплекса средства связи. В случае обнаружения сбоя или отказа ЦУУ последовательно запускает процедуры тестирования и устранения неисправностей. В наиболее критичных случаях, например при длительном отсутствии электропитания, производится автоматический останов с последующим перезапуском программного обеспечения управления. Также автоматически создаётся резервная копия баз данных и программ управления на внешних запоминающих устройствах ЦУУ.

Управление сетевыми элементами предусматривает, что данное средство связи, и соответственно ЦУУ, может выполнять функции узла управления и контроля по отношению к другим средствам связи. Здесь возможны процедуры удаленного запуска, перезагрузки и блокировки аппаратных и программных компонент управляемых средств связи.

Достоинством централизованной архитектуры управления является простота реализации управляющего комплекса. Недостатком является невозможность масштабирования и, следовательно, ограничения «сверху» по обслуживанию поступающей нагрузки примерно до 6…8 тысяч абонентских номеров или портов. Другой проблемой централизованной схемы управления является надёжность и живучесть. Для обеспечения надёжности и живучести централизованной архитектуры управления нередко приходится дублировать ЦУУ на 100%, т.е. создавать второй управляющий комплекс, который работает параллельно основному, что увеличивает стоимость системы. Указанные не-

достатки частично преодолеваются в более сложной системе с распределённым управлением.

При ***иерархической функциональной архитектуре управления*** ЦУУ предоставляет возможность реализации некоторых функций управления групповым управляющим устройствам, как это показано на рис. 1.6.

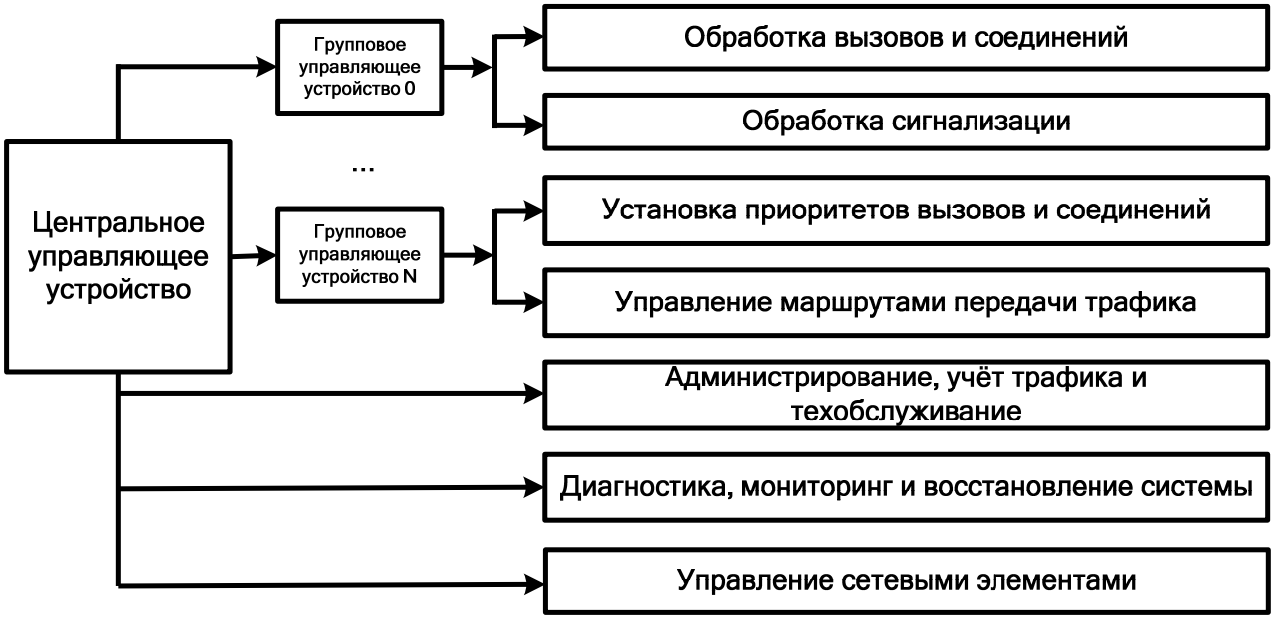


Рис. 1.6 – Иерархическая функциональная архитектура УК средства связи

Каждое ГУУ достаточно «интеллектуально», и в состоянии самостоятельно выполнять функции, переданные ему ЦУУ. В итоге ЦУУ не нагружается рутинными, стандартными задачами, что в целом обеспечивает увеличение числа обслуживаемых вызовов в единицу времени с помощью распределения нагрузки по ГУУ. В результате абонентская ёмкость средства связи увеличивается до 10 тысяч и более номеров или портов. Также существенно улучшаются возможности по наращиванию мощности управляющего комплекса при увеличении поступающего трафика за счёт масштабируемости технического решения. При появлении новых абонентов увеличивается число ГУУ, причём блоки соединительных линий и блоки линий доступа находятся под непосредственным управлением ГУУ, выполняют все функции обслуживания

вызовов и организации сеансов связи. Линейные и абонентские интерфейсы находятся под контролем ИУУ. В иерархической архитектуре ЦУУ постоянно диагностирует и контролирует техническое состояние ГУУ, выводит из эксплуатации отказавшие ГУУ, в случае необходимости осуществляет перезапуск ГУУ.

Иерархическая архитектура обладает еще одним достоинством – при выходе из строя какого-либо из ГУУ производится автоматическое перераспределение выполняемых заданий и система управления в целом продолжает функционирование. Недостатком средства связи с иерархической архитектурой является необходимость координации работы многих ГУУ, например при работе с общими станционными данными, с абонентскими данными, которые хранятся в ЦУУ. Выход из строя ЦУУ может привести к останову средства связи в целом.

В **квази-распределенной функциональной архитектуре управления** на рис. 1.7, ЦУУ по прежнему контролирует ГУУ, а ГУУ в свою очередь осуществляют практически все функции управления средством связи. В квази-распределенной архитектуре ГУУ могут быть назначены для управления отдельными группами блоков соединительных линий, блоков линий доступа. Иными словами, ГУУ могут быть сходными по функциональности, но отличаться друг от друга номерами обслуживаемых периферийных блоков. ГУУ поддерживают большую часть данных, необходимых для функционирования управляемой группы блоков, например сведения об абонентах.

В квази-распределенной функциональной архитектуре ГУУ отвечает за обработку цифр набора номера, определение маршрутов соединений, поддержание и обработку общестанционных данных, включая сведения обо всех абонентах, диагностику. За мониторинг и восстановление всей системы в целом и управление сетевыми элементами по-прежнему отвечает ЦУУ. Существует вариант, при котором одно из определенных заранее ГУУ, передавая сигналы управления по

сквозной внутренней шине управления или через цифровое коммутационное поле, может управлять и перезагружать другие ГУУ.

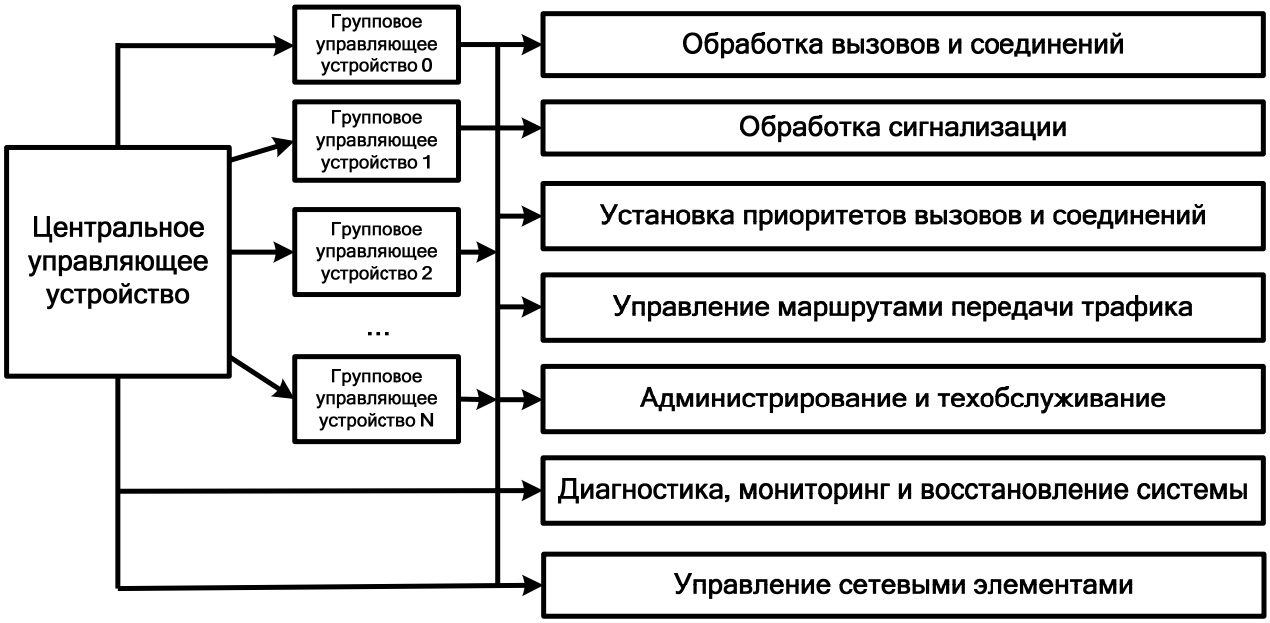


Рис. 1.7 – Квази-распределенная функциональная архитектура УК средства связи

В результате абонентская ёмкость средства связи увеличивается до 300 тысяч номеров.

Квази-распределенная функциональная архитектура, отличается повышенной надежностью и живучестью. Имеется достаточно высокая степень масштабируемости. Сложность с координацией и управлением ГУУ решается за счёт ЦУУ. Недостатком является наличие в архитектуре ЦУУ, отказ которого приводит к частичному или полному отказу всего средства связи в целом. Даже если функции ЦУУ при этом будет исполнять одно из ГУУ, его вычислительной мощности всё равно не хватит на полноценную замену ЦУУ. С другой стороны, делать одно из ГУУ равноценным ЦУУ означает увеличение стоимости системы. Тем не менее, квази-распределенную функциональную архитектуру можно считать компромиссом между сложностью исполнения, стоимостью решения и надежностью.

Наконец, рассмотрим предельный случай – распределенную функциональную архитектуру системы управления средством связи на рис. 1.8.

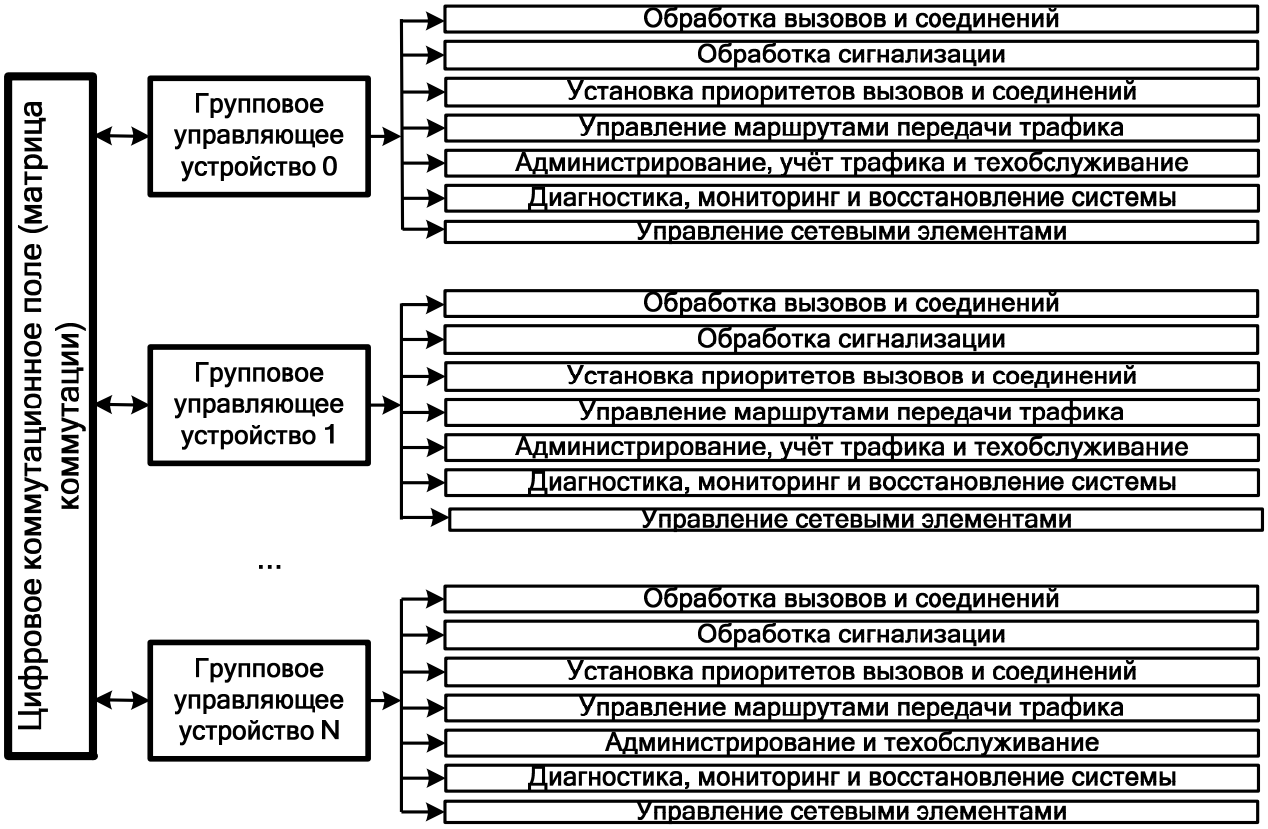


Рис. 1.8 – Распределенная (полностью децентрализованная) функциональная архитектура УК средства связи

В этой функциональной архитектуре полностью отсутствует ЦУУ. Все базовые функции средства связи выполняются независимыми ГУУ. При этом может сохраняться «закрепление» ГУУ за функциональными блоками.

Групповые управляющие устройства в рассматриваемой схеме являются практически автономными и поддерживают все функции, необходимые для штатного функционирования средства связи. При обслуживании вызовов и установлении сеансов связи ГУУ обмениваются друг с другом программными командами управления через цифровое коммутационное поле. Абонентская ёмкость средства связи составляет до 500 тысяч номеров. Данная архитектура, как и все распределенные архитектуры, отличается высокой надежностью и живучестью. Также

архитектура отличается высокой степенью масштабируемости. Недостатком является достаточно высокая сложность координации действий равноправных ГУУ.

В заключении следует отметить, что в настоящее время ёмкость средств связи определяется количеством эквивалентных портов. **Порт эквивалентный –** условная единица, соответствующая физическому порту с нормализованной скоростью передачи (200 бит/с, 64 кбит/с, 1 Мбит/с). **Порт, физический** – аппаратное средство для реализации интерфейса, в том числе с внешней средой, на физическом уровне. Физический порт также реализует интерфейс со средой распространение сигнала электросвязи.

1.4 Использование микропроцессоров в средствах связи

В целом МПр, которые используются в средствах связи, можно разделить на две группы с точки зрения общих технических возможностей обработки данных .

К первой группе относятся ***универсальные МПр*** (процессоры общего назначения). Эти процессоры предназначены для выполнения самого широкого набора вычислительных операций. В дополнение к стандартным операциям, современные МПр могут выполнять на аппаратном уровне криптографические алгоритмы RSA и DSA с 2048-разрядным ключами для защиты информации (например, МПр типа UltraSPARC T1).

Ко второй группе относятся ***специальные МПр***, например процессоры цифровой обработки сигналов, сетевые процессоры. Эти процессоры предназначены для выполнения ограниченного набора вычислительных операций, но за счёт, того, что эти операции поддерживаются аппаратными средствами, выполнение идёт с высокой скоростью и минимальной задержкой по времени. Необходимость в таком решении возникает при цифровой обработки акустических и видео-сигналов, при

обработке радиосигналов, в ходе кодирования/декодирования информации и т.п.

Следует отметить, что специальные МПр зачастую непосредственно подключены к физическим интерфейсам, к аппаратуре преобразования и кодирования, к различным датчикам, антеннам излучения, коммутационным приборам. Это позволяет уменьшить время реакции МПр на событие. В результате специальные МПр монтируются непосредственно в модули.

Специальные МПр могут реализовываться в виде ***микроконтроллеров*** [17] – программируемых однокристальных вычислительных устройств с встроенным набором средств для ввода-вывода данных, применяемое для решения задач управления и первичной обработки данных.

Иерархия процессоров согласно их использованию в современных средствах связи представлена на рис. 1.9.

Различают следующие специализированные МПр :

Сетевые (коммуникационные) процессоры реализуют аппаратную поддержку и управление интерфейсами и коммуникационными протоколами такими как Ethernet, HDLC, X.25. E1 (G.703), USB, ATM, а также аппаратную поддержку протоколов шифрования (IPSec). Примерами таких МПр являются Моторола MC683xx, MPC8xx, AMD Am186CC, Intel IXA IXP 2XXX, Intel IXP 4XXX. Кроме того, существуют специальные модемные процессоры для поддержки стандартов V.3x, V.9x.

Процессоры цифровой обработки сигналов, ПЦОС или цифровые сигнальные процессоры ЦСП (digital signal processor, DSP) предназначены для реализации методов цифровой обработки сигналов: фильтрация, спектральный анализ, смешение сигналов, масштабирование [24,32]. Отличительная особенностью ПЦОС является обработка больших объёмов данных в реальном времени, но с ограниченным набором операций.

Рис. 1.9 – Функциональная иерархия процессоров в современных средствах связи [С учётом *Wolf Tilman*, курс ECE 697J, Univ. of Massachusetts Amherst, USA]

Процессор цифровой обработки сигналов выполняют четко алгоритмизированные задачи кодирования, декодирования, ЦАП/АЦП. Примерами ПЦОС являются МПр типа TI TMS 320, Analog Devices ADSP 21xxx, Motorola DSP56xxx, Motorola DSP96xxx. Процессоры ПЦОС выполняют вычислительные операции (как правило, сложение и умножение) в системах анализа сигналов, при реализации кодеков или кодеров различного назначения. Эти процессоры широко применяются

в системах сотовой связи стандартов GSM, CDMA для осуществления кодирования и сжатия исходного аналогового речевого сигнала. Аппаратная часть ПЦОС оптимизирована для выполнения операций с плавающей точкой или с целыми числами. Это позволяет за счет точности и скорости вычислений максимально точно кодировать, а затем воспроизводить переданный сигнала, особенно в условия зашумления.

Процессоры встраиваемых (управляющих) приложений [15] предназначены для автоматической реализации функций управления и обладают следующими основными свойствами :

- работа без взаимодействия с человеком;
- непосредственное подключение к объекту управления;
- работа в реальном времени;
- несложные алгоритмы вычислений;
- наличие автоматических алгоритмов регулирования и адаптации при изменении характеристик входных данных.

Различают следующие виды встроенных процессоров:

- универсальные встроенные процессоры;
- процессоры с расширенными коммуникационными возможностями;
- процессоры с расширенными возможностями дискретного ввода/вывода.

Рассматривая в целом схему на рис. 1.9 можно отметить, что производительность, сложность архитектуры, стоимость увеличиваются снизу вверх. В тоже время при движении сверху вниз разрядность, тактовая частота, производительность снижаются. В составе ЦУУ, как правило, применяются универсальные МПр. В составе ГУУ могут применяться универсальные или сетевые процессоры. В составе ИУУ могут применяться процессоры цифровой обработки сигналов, процессоры ввода-вывода и процессоры сетевых или внутренних интерфейсов (процессоры встраиваемых приложений).

Необходимо добавить, что нередко специализированные процессоры (в том числе сетевые) выполняются по технологии программируемых логических интегральных схем (ПЛИС). Устройство ПЛИС представляет собой матрицу логических вентилей (цифровых устройств, реализующих логические функции И–НЕ, ИЛИ–НЕ и т.п.), логика работы и переключения которых может быть задана программным образом. Отличительной чертой ПЛИС является наличие архитектуры, допускающей реконфигурирование цифровых цепей и базирующейся на массиве конфигурируемых логических блоков (configurable logic blocks, CLBs), окруженных блоками ввода/вывода. При этом программируемые переключатели и маршрутизаторы, входящие в состав микросхемы, позволяют осуществлять маршрутизацию сигналов в ПЛИС любым, наперед заданным образом. Разработчики оборудования используют ПЛИС в основном из-за их малых размеров, высокой скорости работы, малого энергопотребления и возможности обновления логики работы. ПЛИС может быть реконфигурирована и перепрограммирована в соответствии с нуждами конкретной задачи, не затрачивая ресурсы на дорогостоящее производство заказных микросхем. Микросхема ПЛИС может включать от 1до 3 миллионов вентилей, время затрачиваемое на переключение вентилей – по крайней мере 25 нс.

В дальнейшем в рамках учебного пособия будут изучаться универсальные процессоры, специализированные процессоры – сетевые процессоры, процессоры цифровой обработки сигналов, процессоры ввода-вывода.

1.5 Алгоритм работы процессора

Понятие «процессор» было введено в 1946 г. Под ***процессором,*** в общем случае, понимается схемно-реализованный алгоритм. Элементами этой реализации является цифровые вычислительные устройства, которые реализуют логические функции. Как уже отмечалось,

команды, которые выполняются процессором, хранятся в памяти (оперативное запоминающее устройство, ОЗУ или постоянное запоминающее устройство, ПЗУ) в виде программ. Программа, которая выполняется МПр, записывается в машинном коде, т.е. в виде последовательности «0» и «1». Программы представлены в виде упорядоченной последовательности команд, причем упорядочивание последовательности команд соответствует алгоритму программы. В целом, любой МПр выполняет единый алгоритм обработки данных. Описание данного алгоритма в виде последовательности процедур см. рис. 1.10. Рассмотрим основные процедуры на рис. 1.10 более подробно.

На рис. 1.10 процедуры «Начало» и «Останов» не указаны.

Выборка или чтение очередной команды осуществляется из ячейки памяти (ЯП), адрес которой находится в счетчике команд (СчК). Разрядность счётчика команд определяет размер адресного пространства МПр. Следует отметить, что при включении МПр (подача электропитания на МПр) самый первый адрес, по которому производится считывание команды из физической памяти, задается по умолчанию разработчиком. Например, это может быть нулевой адрес.

Рис. 1.10 – Общая последовательность процедур функционирования процессора

В частности, при включении электропитания микропроцессор 8086 начинает работу с передачи управления по адресу FFFF:0000; этот адрес заносится в регистры, что является аппаратной особенностью данного типа микропроцессора. Адрес FFFF:0000 принадлежит постоянному запоминающему устройству ПЗУ BIOS; при обращении по этому адресу осуществляется переход на начало программы самопроверки при включении электропитания POST (power on self test), которая хранится в ПЗУ.

Если счетчик n-разрядный, то максимальный номер ячейки памяти, ЯП (фактически, размер адресного пространства), которая доступна для хранения данных и кодов операций (инструкций) будет равен $M_{адр}= 2^n-1$. Микропроцессор обращается по адресу, который формируется с помощью СчК и извлекает из памяти требуемую команду и загружает в регистр команд. Минимально адресуемая ЯП является восьмиразрядной (1 байт). Считанная из памяти команда записывается в регистр команд, при этом содержимое счетчика увеличивается на 1, в случае если это не команда перехода.

На диапазон доступных адресов влияет разрядность шины адреса МПр. Допускается расширение адресного пространства МПр с помощью специальных методов, в частности с помощью параллельного порта [19].

При ***декодировании команд*** осуществляется преобразование программной команды в последовательность функциональных сигналов управления (электрических импульсов определенного уровня и длительности), которые поступают на входы/выходы всех компонентов МПр и внешних устройств. В результате срабатывают микросхемы АЛУ и регистров.

Исполнение команды предусматривает осуществление требуемой операции с помощью микрокоманд и микропрограмм. В результате исполнения, происходит запись в регистр результата.

Регистром результата может быть:

- регистр - аккумулятор;
- регистр общего назначения;
- постоянное запоминающее устройство (ПЗУ);
- оперативное запоминающее устройство (ОЗУ);
- другие запоминающие устройства.

В целом можно выделить три группы из числа наиболее часто выполняемых МПр:

1. операции чтения (RD, Чт) из физической памяти и записи (WR, Зп) в физическую память;
2. операции преобразования в АЛУ;
3. операции условного или безусловного перехода внутри исполняемой программы.

При выполнении описанных групп операций адресное пространство памяти МПр распределяется между различными внешними запоминающими устройствам. Кроме того, регистры общего назначения, объединенные в блок или файл (блок регистров общего назначения, БРОН) могут иметь индивидуальные назначенные адреса.

В итоге, физическая память (адресное пространство физической памяти) может быть распределено между данными и командами, необходимыми для функционирования различных функциональных компонент и загружаемых программ микропроцессора:

- группа (пространство) адресов для хранения информации ПЗУ, таблицы прерываний в т.ч. хранение информации запросов на прерывания (interrupt request, IRQ);
- группа (пространство) адресов для операционной системы;
- группа (пространство) адресов для программ пользователя;
- группа (пространство) адресов для хранения информации, поступающей от портов ввода/вывода;

- группа (пространство) адресов для хранения информации от каналов прямого доступа к памяти (direct memory access, DMA).

Режим DMA и прерывания подробно будут рассмотрены ниже.

В МПр существует несколько способов определения очередности запуска программ на исполнение. В рассмотренном выше алгоритме команда запускалась на исполнение, как только ей передавалось управление от только что выполненной команды. Существует также возможность, когда команда получает возможность запуска при готовности данных (операндов) соответствующей команды. И наконец, команда может быть запущена, когда потребуется её результат. В средствах связи последний способ применяется едва ли не чаще, чем первые два.

Как видно из приведённого перечня, управление распределением физической памяти между различными аппаратными и программными средствами представляет собой сложную динамическую задачу управления ресурсами МПр. Рассмотрим подробнее средства и методы решения задачи управления памятью микропроцессора с помощью виртуальной памяти.

1.6 Организация виртуальной памяти

Существует существенный разрыв между быстродействием элементов МПр и элементов оперативной памяти. Этот разрыв в зависимости от конструкции процессора и памяти составляет 10..100 раз. Кроме того, в архитектуре фон Неймана для обмена процессор–память используется шина адреса и шина данных; шина имеет конечную пропускную способность, что позволяет называть наличие шины «бутылочным горлышком» (узким местом) архитектуры фон Неймана.

Идеальная физическая память микропроцессора – это та, которая не приводит к простою МПр. На практике памяти, быстродействие элементов которой соответствует быстродействию элементов МПр, не су-

ществует. Производители стремятся увеличить быстродействие памяти, в том числе за счет уменьшения времени считывания информации, как в памяти типа RAMBUS. В памяти стандарта DDR2 (double data rate, удвоенная частота обмена данными) рабочая частота шины данных и адреса к памяти в два раза выше, чем тактовая частота, используемая в ячейках памяти. Таким образом, увеличение задержек в отдельных ячейках памяти компенсируется общим увеличением пропускной способности шины. В памяти типа DDR3 рабочая частота шины уже в 4 раза выше, чем частота отдельной ячейки памяти.

Для увеличения производительности при обмене процессор–оперативная память без существенных изменений в конструкции памяти применяются следующие способы:

- логическое и физическое упорядочивание данных в памяти;
- применение многоуровневой организации памяти (использование кэш-памяти).

Логическое и физическое упорядочивание предусматривает организацию виртуальной памяти. Здесь и далее термин ***«виртуальный»*** характеризует процесс или устройство в системе обработки информации, кажущихся реально существующими, поскольку все их функции реализуются какими-либо другими средствами.

Виртуальная память – это общее адресное пространство, с которым работает программное обеспечение (ПО), безотносительно к тому, где физически находится программа или данные – в ОЗУ, на диске или на накопителе на магнитном диске (НМЛ). Виртуальная память моделирует память существенно большего размера, чем оперативная память, которая фактически доступна МПр.

Современный МПр работает в многозадачном режиме, т.е. в оперативной памяти находятся программы и данные, предназначенные для одновременного выполнения нескольких задач (процессов). Поэто-

му необходим механизм разделения физической памяти МПр между различными процессами или программными задачами.

Под ***задачей*** здесь и далее понимается совокупность процедур, машинного или внемашинного вида, обеспечивающих разовую подготовку и выдачу информации для оформления одного или нескольких управляющих воздействий независимо от числа объектов, подвергающихся этому воздействию.

Для решения проблемы упорядочивания, виртуальная память разбита на несколько непересекающихся групп адресов, которые могут объединяться в логическую страницу, каждой из которых присваивается номер страницы (логический адрес). На основе логических адресов и логических страниц виртуальная память позволяет разделить физическую память на логические блоки адресов, при этом каждый логический блок адресов закреплён за конкретной программной задачей или за группой задач и отображается на физический блок (сегмент) физической памяти.

Виртуальная память разбивается на отдельные блоки (страницы), логические страницы разбиваются на ячейки. Размер страницы составляет 2^m байт, $6 \le m \le 12$, при этом данные к которым следует обратиться через виртуальную страницу могут находиться в различных физических блоках или сегментах памяти.

Физический блок (сегмент) – это элемент разбиения физической памяти (ОЗУ, ПЗУ, НЖМД, НМЛ), при этом размер сегмента, как правило, равен размеру виртуальной страницы (виртуального блока).

В результате применения виртуальной памяти появляются следующие возможности :

- существует механизм разделения физической памяти между различными программами;
- поддерживается механизм защиты на основе поддержки различных прав доступа (только чтение, чтение и запись) к об-

ластям физической памяти, в первую очередь – выделение системной области памяти, где хранится ОС;

- появляется возможность управлять размещением программных кодов в физической памяти в рамках отведённых блоков;
- данные могут перемещаться в пределах физической памяти, не нарушая при этом логической целостности программ;
- обеспечивается возможность передачи данных непосредственно между программами, не применяя при этом возможности ОС (действует только для общего виртуального пространства адресов).

В связи с наличием виртуальной и физической памяти необходимо осуществлять операцию отображения или пересчета виртуального адреса в физический. Для этого используется специальная таблица страниц (сегментов). Пример специальной таблицы страниц показан на рис. 1.11.

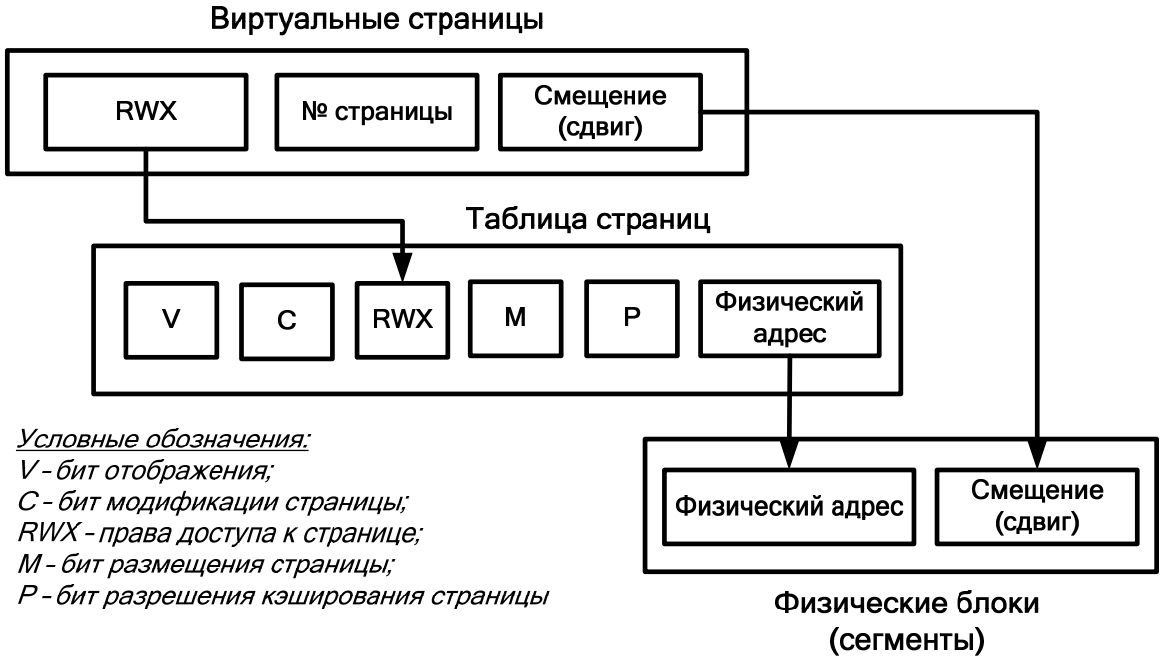


Рис. 1.11 – Организация отображения виртуального адреса в физический адрес [14]

Таблица страниц формируется и поддерживается операционной системой для каждой программы и хранится в основной памяти (ОЗУ). Каждой виртуальной странице (блоку) ставится в соответствие элемент таблицы страниц, который включает :

- Номер физического блока.
- Индикатор активности (1 – страница в ОЗУ, доступна для модификации данных т.е. активна; 0 – страница не находится в ОЗУ, недоступна для модификации данных и. следовательно неактивна).

Рассмотрим значение бит в таблице страниц:

- Бит V – определяет, возможен ли пересчет из виртуального адреса в физический.
- Бит C – указывает на изменения данных на странице:
  C=0, если данные не менялись операциями считывания или копирования;
  C=1, если данные были модифицированы.
- Бит RWX – определяет, доступна страница только для чтения, или для чтения и записи.
- Бит M – указывает на то, размещена ли данная страница в ОЗУ, в этом случае M=1. Если страница размещена на прочих устройствах памяти (НЖМД, НМЛ), то бит M=0.
- Бит P – разрешает или запрещает кэширование страницы. Например, если P=1, то данные из ОЗУ могут быть перенесены в кэш.

В целом существует две системы виртуальной памяти:

Системы с фиксированным размером блоков (страничная организация памяти). Достоинством данного решения является относительная простота организации. К недостаткам относятся :

- необходимо выделения специальных страниц для общих программ;

- возможность появления большого числа незанятых страниц, т.к. размер программы не всегда кратен размеру страниц.

Адрес ячейки виртуальной страницы формируется из двух полей: адрес страницы (старший разряд логического адреса) + номер слова в странице (младший разряд логического адреса).

Система с переменным размером блоков (сегментная организация памяти) – здесь сегмент формируется как отдельная логическая единица информации со своей нумерацией слов в пределах сегмента.

Существуют следующие сегменты :

- программные сегменты, которые хранят общие программы с возможностью выборки команд и чтение констант, при этом операция записи в сегмент запрещена;
- сегменты данных, которые хранят только данные с разрешением либо на чтение, либо на запись.

Достоинством данного технического решения является адаптация к размерам программ; недостатком данного технического решения является сложность организации, особенно в части защиты и наделения правами доступа к данным. Для преодоления данной сложности иногда используется аппаратное решение.

С помощью таблицы страниц появляется возможность оптимизировать загрузку процессора, увеличив тем самым производительность МПр. Другим способом увеличения производительности оперативной памяти при обмена с МПр является организация буферной или кэш–памяти.

1.7 Назначение и организация кэш-памяти

Кэш (cache) память является буферной памятью МПр, расположена непосредственно на кристалле МПр и предназначенной для временного хранения данных, необходимых для текущих операций про-

цессора. Кэш-память использует свойство программного обеспечения, согласно которому 90% обращений в память производится по ограниченной области адресов. Эта область адресов называется ***рабочим множеством***, которое изменяется по мере выполнения загруженной программы. Рабочее множество данных включает содержимое физических ячеек оперативной памяти с последовательными адресами. Текущее рабочее множество данных (блок данных) можно временно копировать и сохранять в буферной памяти с минимальным временем доступа со стороны МПр для чтения/записи. Эта буферная память сохраняет данные и команды всё время, необходимое для их исполнения. После окончания обработки данных содержимое кэш-памяти копируется содержимое в ОЗУ, откуда вновь считывается в кэш-память новое рабочее множество.

Разрядность кэша составляет от 4 до 128 байт, ёмкость кэш памяти составляет от 4 кбайт до 16 Мбайт в зависимости от уровня кэш-памяти и типа микропроцессора. Управляет кэш-памятью контроллер кэш-памяти. При необходимости считывания в МПр операнда, контроллер кэш-памяти ищет указанный операнд сначала в кэш-памяти. Если операнд не найден, то формируется т.н. «кэш-промах» (missing cache) при операции чтения кэш-памяти. Тогда контроллер кэш-памяти считывает в кэш-память из оперативной памяти физический блок данных т.е. содержимое нескольких физических ячеек памяти с последовательными адресами.

При записи в кэш-память, каждое слово данных (про слово данных см. раздел 1.7), т.е. содержимое ячеек памяти ОЗУ, сопровождается адресным тэгом (tag, метка), указывающим, какой блок данных оперативной памяти представляет данная запись. В качестве тэга может использоваться смещение/сдвиг относительно номера страницы памяти (младшие биты адреса). Таким образом кэш является непосредственно адресуемой памятью. Размер/разрядность тэга ограничена поэтому в

данный момент времени для кэширования доступна не вся оперативная память, а например 256 Мбайт.

За счёт подключения непосредственно к ЦПУ, кэш–память имеет малое время обращения, около десятков наносекунд. Если физическая память объёмом M разделена на N сегментов, обычно объемом 1024 байт, то кэш содержит n страниц такого же объема, причем $N \gg n$. Любая страница в кэш может заполняться информацией из M всего за несколько тактов. В каждой странице кэш-памяти расположены данные со смежными адресами, разные страницы кэш-памяти могут не стыковаться по адресам. Выделяют два типа кэш памяти:

- Кэш-память с запоминанием новой информации одновременно в кэше и оперативной памяти (сквозное запоминание). В оперативной памяти всегда есть последняя копия информации, хранящейся в кэше. Однако в этом случае длинный цикл доступа к данным в ОЗУ снижает общую производительность вычислительной системы.
- Кэш-память с вытеснением, когда запоминание результатов обработки данных МПр производится только в кэш-памяти. Эти результаты копируются в оперативную память только при передаче во внешние устройства или при вытеснении информации из кэша при загрузке новых данных и(или) программ .

Существует несколько способов организации кэш-памяти.

- ***Кэш с прямым отображением*** – каждый сегмент (блок) основной памяти имеет фиксированное место в кэше. В итоге сегменты основной памяти с одинаковыми младшими разрядами хранятся в одинаковых блоках кэша. Этот способ часто используется в современных МПр.
- ***Кэш полностью ассоциативный*** – сегмент (блок) основной памяти может находиться в любом месте кэша.

- ***Множественно-ассоциативный кэш*** – сегмент (блок) основной памяти размещается на ограниченном числе мест в кэше. Создаётся группа блоков; этот способ часто используется в современных МПр.

Конструкция кэш обуславливает его ограниченный размер и ёмкость хранимой информации. Ограничение объема кэша позволяет сократить время доступа к данным. При работе с кэшем преобладают операции чтения из кэша (до 90% операций) и только 10% операций приходятся на процедуру записи в кэш. В частности, запись позволяет обеспечить синхронизацию данных между кэш и ОЗУ, для этого используется специальные алгоритмы. Имеется следующая цепочка операций: чтение оригинала блока данных из кэша → модификация части блока с помощью ЦПУ → запись нового значения блока в кэш. Модифицированный блок кэш-памяти записывается в ОЗУ только после полного замещения информации в кэш. Для того, чтобы отметить модифицировался блок или нет, используют специальный бит состояния, аналогичный биту С. Если бит не изменил своего значения, то копирование в ОЗУ отменяется, что уменьшает время обращения МПр к данным и повышает производительность вычислительного устройства. Физически размер одной ячейки запоминающего устройства кэш–памяти достаточно большой и составляет 6…8 транзисторов и занимает площадь 0,57…0,7 мкм^2. В результате размер кэш–памяти ограничивается физическими размерами кристалла МПр.

Уровень кэш–памяти определяется конструктивно с помощью физического расстояния до центрального процессорного устройства. Кэш 1-го уровня конструктивно ближе всех расположен к ЦПУ и имеет физическую ширину (разрядность), равную разрядности шине данных. Кэш 1-го уровня находится на кристалле процессора и может рассматриваться как регистр большой ёмкости. Кэш 2-го уровня находится на кристалле процессора и мультиплексирует данные системной шины и кэша

первого уровня. Кэш 3-го уровня непосредственно подключается к процессору, но на кристалле не располагается. С учетом вышесказанного, наличие кэша и регистров МПр обеспечивает следующие технические характеристики для памяти различного уровня :

- регистр общего назначения (РОН) имеет размер 64…256 байт, время доступа к данным составляет $t_{дост}$ = 1 такт (5 нс);
- кэш 1-го уровня (L1) имеет размер 16…32 кбайт, время доступа к данным составляет $t_{дост}$ = 1…2 такта (менее 10 нс);
- кэш 2-го уровня (L2) имеет размер от 256 кбайт до 512 Кбайт, с изменением конструкции МПр – до 6 Мбайт, время доступа к данным составляет $t_{дост}$ = 2..3 такта (менее 10 нс);
- кэш 3-го уровня (L3) имеет ёмкость до 9 Мбайт, это максимальное значения например для МПр типа Itanium2 Madison; время доступа к данным составляет $t_{дост}$ = 3…5 тактов (менее 10 нс);
- ёмкость ОЗУ составляет до 4 Гбайт на 1 микросхему памяти, $t_{дост}$ = 10…55 тактов (20…60 нс).

В целом иерархия памяти МПр с архитектурой фон Неймана и кэш-памятью L1 и L2 может быть представлена на рис. 1.12 :

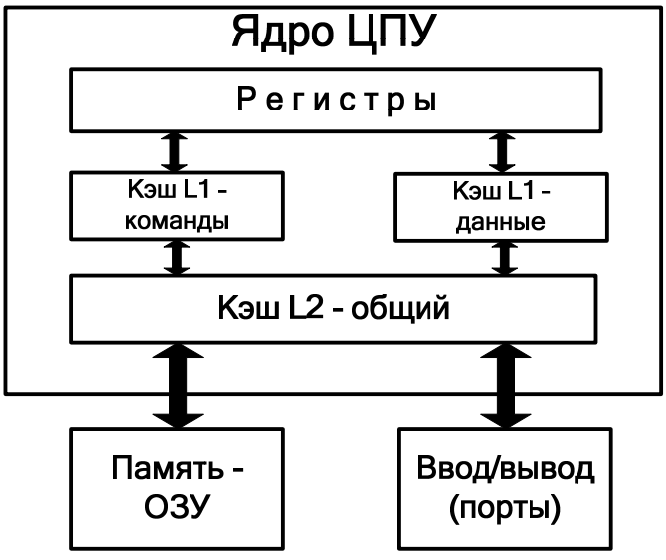


Рис. 1.12 – Организация памяти процессора с архитектурой фон Неймана и кэш 1-го и 2-го уровня

Обычно содержимое кэш-памяти L1 целиком находится в кэш-памяти L2; все содержимое L2 является частью содержимого кэш-памяти L3.

Микропроцессор на рис. 1.12 согласно классификации Флинна, может быть условно отнесён к классу SISD (Single Instruction Single Data) одиночный поток команд–одиночный поток данных. Под ***потоком команд*** можно понимать последовательность команд одной программы. ***Поток данных*** — это последовательность данных, обрабатываемых одной программой. Таким образом, на рис. 1.12 изображена последовательная ЭВМ, которая выполняет единственную программу. В этом простейшем варианте МПр имеет только один счетчик команд и одно АЛУ. На уровне программы, загружаемой в МПр, можно говорить просто о потоке (thread), где каждый поток есть реализация алгоритма. Таким образом, многопоточность на практике означает одновременное выполнение нескольких, как правило связанных между собой, алгоритмов на одном МПр.

Увеличение производительности данной микропроцессорной системы за счёт развития архитектуры, связанной с обработкой данных, детально рассматривается в главе 5.

1.8 Управление вводом-выводом, прямой доступ к памяти

Вводом/выводом (ВВ) [13,17] называется процесс обмена (переноса) данных между МПр, основной памятью ОЗУ и внешними устройствами ввода-вывода, к которым относятся НЖМД, НМЛ, накопитель на оптическом диске (НОД), магнитооптический накопитель, клавиатура, манипулятор типа «мышь», прочие внешние устройства, в т.ч. процессоры других управляющих устройств.

Устройства ввода-вывода – устройства, специализированные на на ввод программ и данных в управляющий комплекс средтсва свя-

зи, вывод результатов обработки данных УК, а также преобразование данных из одной формы в другую.

Для связи с УК или микропроцессорной системой устройства ввода-вывода подключаются к параллельным или последовательным шинам ввода-вывода или к общесистемным шинам. Подключение к шинам осуществляется непосредственно, или с помощью контроллеров (адаптеров) ввода/вывода.

Контроллер ввода-вывода (контроллер ВВ) есть микропроцессорное устройство, предназначенное для управления процессами ввода-вывода. Контроллер ВВ осуществляет приём/передачу сигналов от внешних устройств, их схемную или программную обработку с приёмом/передачей управляющих или информационных сигналов на шину для МПр или ОЗУ. Контроллеры ВВ согласуют уровни электрических сигналов; преобразуют запросы/команды внешних устройств в формат, необходимый микропроцессору и наоборот; управляют обменом данными. Контроллер ввода-вывода может быть главным контроллером (формирует прямой канал ввода-вывода к МПр) или контроллером внешнего или периферийного устройства (имеет доступ к главному контроллеру с одной стороны и подключен к внешнему устройству с другой стороны).

В процессе программного управления вводом/выводом передается информация двух видов: управляющие данные (командные слова) и собственно данные.

Управляющие данные от МПр – инициируют действия, не связанные непосредственно с передачей данных, например запуск внешнего устройства, запрещение прерываний. ***Управляющие данные от внешних устройств (слова состояния)*** – содержат информацию об определенных признаках, например о готовности внешнего устройства к передаче данных, сведения о наличии ошибок при обмене и т.п. Состояние обычно представляется в кодированной форме – один бит для

каждого признака или параметра устройства. Перечисленные данные хранятся в специализированных программных или аппаратных регистрах контроллера ВВ или МПр.

Регистр, содержащий группу бит, к которому процессор обращается в операциях ВВ**,** образует ***порт ввода-вывода*****.** Это описание порта в рамках программной модели. На физическом уровне порт представляет собой аппаратное средство для реализации интерфейса, в том числе с внешней средой. Физический порт также реализует интерфейс со средой распространение сигнала электросвязи.

Наиболее общая программная модель внешнего устройства, которое может выполнять ввод и вывод, содержит четыре регистра ВВ: регистр выходных данных (выходной порт), регистр входных данных (входной порт), регистр управления и регистр состояния (рис. 1.13).

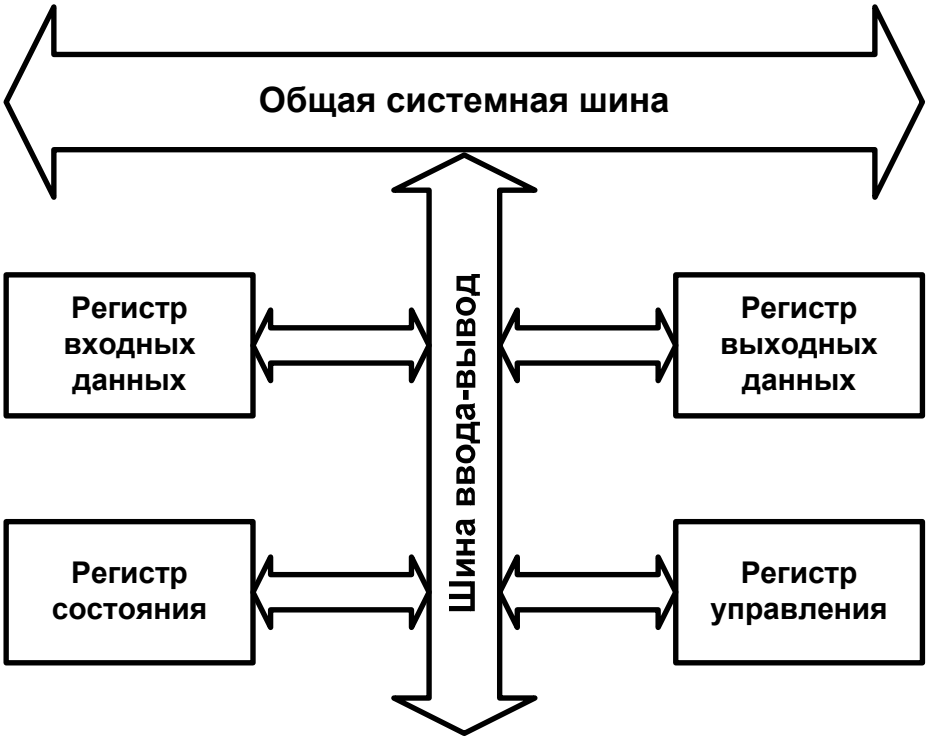


Рис. 1.13 – Программное представление внешнего устройства для организации ввода-вывода [8]

Каждый из регистров на рис. 1.13 имеет уникальный физический адрес, который идентифицируется дешифратором адреса.

В зависимости от особенностей устройства ввода/вывода функциональная схема изменяется :

- отдельные регистры состояния и управления могут объединяться в один регистр;
- в устройстве ввода (вывода) может использоваться только регистр входных (выходных) данных;
- для ввода и вывода может использоваться двунаправленный (дуплексный) порт.

Обращаться к портам ввода/вывода можно с помощью специального адресного пространства для портов ввода/вывода в оперативной памяти или с использованием общего адресного пространства, когда порт ввода/вывода рассматривается процессором как ячейка памяти. Эти различия в адресации позволяют реализовать два способа ввода/вывода. В результате существует интерфейс ВВ с изолированными шинами и интерфейс ВВ с общими шинами.

Интерфейс ВВ с изолированными шинами характеризуется раздельной адресацией памяти и портов (внешних устройств) при обмене информацией. Здесь ввод/вывод предполагает наличие специальных команд ввода/вывода, общий формат которых показан на рис. 1.14.

При выполнении команды ввода IN содержимое прямо или косвенно адресуемого входного регистра PORT передается во внутренний регистр REG микропроцессора.

При выполнении команды OUT содержимое регистра REG передается в выходной порт PORT. В МПр могут быть и другие команды управления, относящиеся к ВВ, например команды управления связанные с проверкой и модификацией содержимого регистра управления и состояния.

Интерфейс с общими шинами (ввод/вывод с отображением на

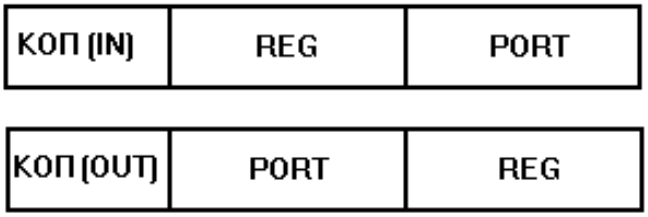


Рис. 1.14 – Формат команд ввода-вывода

память) имеет организацию, при которой часть общего адресного пространства памяти МПр отводится для внешних устройств, регистры ввода/вывода которых адресуются так же, как и ячейки памяти. Если, к примеру, адресное пространство памяти составляет 64 Кбайт, а для программного обеспечения достаточно 32 Кбайт, то область адресов от 0 до 32К-1 (К=1024) используется для пространства памяти, а область адресов от 32К до 64К–1 используется для операций ввода/вывода. При этом признаком, дифференцирующим обращения к памяти и портам ВВ, может быть старший бит адреса. В рассмотренном случае для адресации портов ВВ используются сигналы READ (чтение) и WRITE (запись) по указанному физическому адресу памяти, закреплённому за регистром (портом) ввода/вывода.

В операционных системах имеется набор подпрограмм (драйверов ВВ), управляющих операциями ВВ для стандартных внешних устройств. Благодаря этим драйверам пользователь может не знать многих особенностей конструкции внешних устройств и интерфейсов ВВ, а просто использовать имеющиеся программные протоколы.

При синхронной последовательной передаче в процессе ввода/вывода каждый передаваемый бит данных сопровождается импульсом синхронизации, информирующим приемник о наличии на линии информационного бита. Асинхронная последовательная передача при вводе/выводе означает, что у передатчика и приемника нет общего ге-

нератора синхроимпульсов и что синхронизирующий сигнал не посылается вместе с данными.

В управляющих комплексах применяются три режима ввода/вывода:

- программно-управляемый ВВ (называемый также программным или нефорсированным ВВ);
- ВВ по прерываниям (форсированный ВВ);
- прямой доступ к памяти.

Программно-управляемый ВВ характеризуется тем, что инициирование и управление ВВ осуществляется программой, выполняемой микропроцессором, а внешние устройства играют сравнительно пассивную роль и сигнализируют только о своем состоянии, в частности, о готовности к операциям ввода/вывода.

Ввод–вывод по прерываниям инициируется не микропроцессором, а внешним устройством, генерирующим специальный сигнал прерывания. Реагируя на этот сигнал готовности устройства к передаче данных, МПр передает управление ВВ подпрограмме обслуживания устройства, вызвавшего прерывание. Действия, выполняемые этой подпрограммой ВВ, определяются устройством, вызвавшем прерывание.

Режим прямого доступа к памяти, DMA (direct memory access) – метод обращения внешнего устройства к оперативной памяти компьютера или управляющего комплекса без участия процессора. Используется, когда мощности процессора для обработки запросов на прерывания недостаточно, в результате чего скорость обработки данных существенно замедляется. Для предотвращение такого варианта, микропроцессор исключается из цепочки управления передачей данных между основной памятью и внешним устройством (устройствами).

Режим прямого доступа к памяти DMA позволяет освободить центральное процессорное устройство от задачи чтения данных из устрой-

ства ввода-вывода и записи этих данных в память для выполнения вычислительных задач. В результате применения DMA операция чтения – записи в память производится самим внешним устройством (которое должно быть достаточно «интеллектуально») или специальным контроллером DMA, как это показано на рис. 1.15.

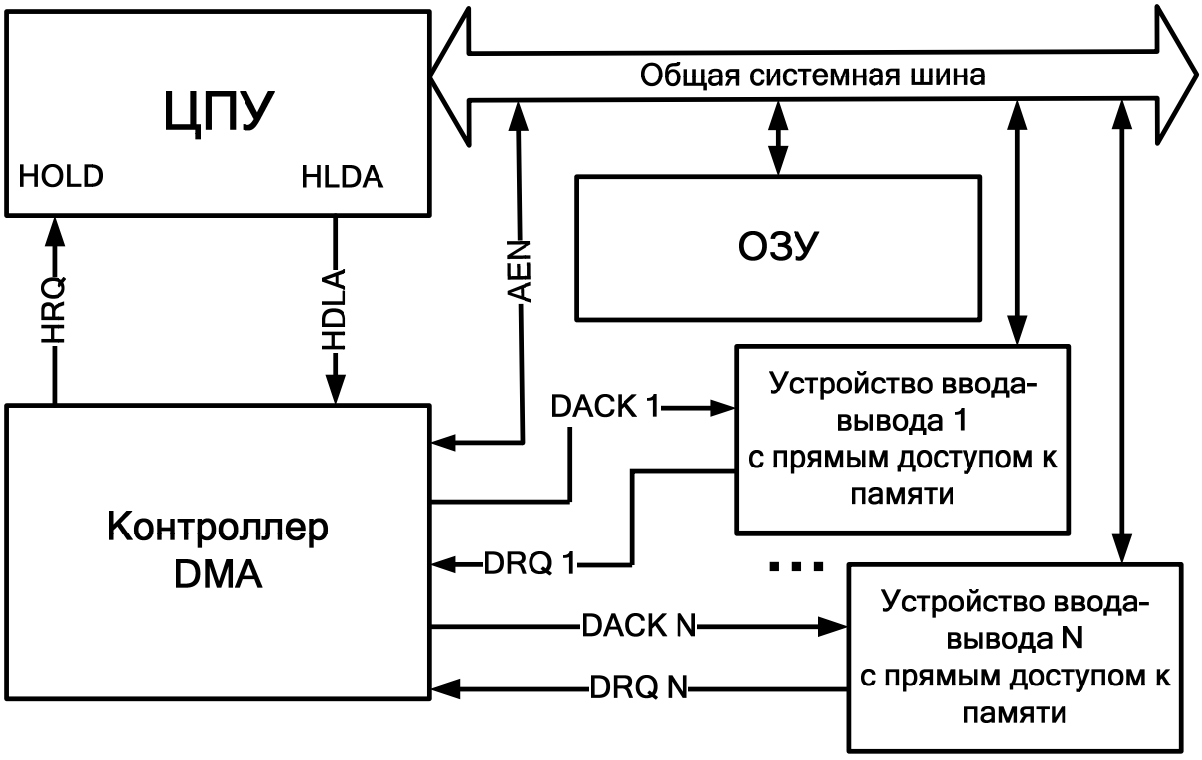


Рис. 1.15 – Взаимодействие ЦПУ и контроллера DMA

Для выполнения прямого доступа в память ОЗУ, устройство ввода-вывода посылает сигналы запроса прерывания для DMA, DRQ (DMA Request), в сторону контроллера DMA. Контроллер DMA формирует сигнал запроса шины для DMA, HRQ (Hold DMA Request) в сторону ЦПУ. ЦПУ, получив сигнал HRQ, завершает текущий обмен и временно отключается от общей системной шины, при этом генерируется сигнал HDLA в сторону контроллера DMA и управление обменом передаётся этому контроллеру. Котроллер DMA выставляет сигнал подтверждения прямого доступа к памяти DACK 1…DACK N (DMA Acknowledge) в сторону внешнего устройства, что может рассматриваться как разрешение

начала обмена данными. Управление доступом к общей системной шине для операции ВВ передаётся внешнему устройству; контроллер DMA формируют сигнал AEN (address enable), который получают по общей системной шине все остальные устройства, которые извещаются о начале операции прямого доступа к памяти. Контроллер DMA с этого момента управляет обменом внешнее устройство – оперативная память. Как только обмен заканчивается, контроллер DMA снимает запрос DRQ и AEN; управление доступом к шине возвращается к ЦПУ. За время DMA ЦПУ может выполнять другие задачи, что безусловно повышает производительность системы в целом.

1.9 Языки программирования

Языки программирования, используемые в том числе для разработки программного обеспечения управления средствами связи, делятся на две группы. Первая группа включает языками программирования низкого уровня. К первой группе относится ***машинно-ориентированный или машинный язык (машинный код)*** – это способ записи команд и данных, которые непосредственно реализуются аппаратными средствами вычислительного устройства (процессора).

Машинный язык включает систему команд и метод кодирования информации. В команде всегда указывается тип выполняемой операции и местонахождение операндов. Типовыми символами машинного языка являются двоичные символы «0» и «1», которые задают адреса в командах, коды операций и признаки команд. При исполнении программ, «0» и «1» в виде физических сигналов поступают на входы микросхем и приводят к исполнению аппаратными средствами МПр заданных операций. Из последовательности команд составляются программы, реализующие алгоритмы задач управления системой коммутации. Эффективность решения различных задач с помощью МПр зависит от того, насколько машинный язык приспособлен для реализации задан-

ных алгоритмов управления средством связи. Кроме того, немаловажную роль играет используемые способы программирования.

Программирование на машинном языке ведётся в системе команд, поддерживаемой данным типом процессора. В результате аппаратные и логические ресурсы процессора используются максимальным образом; поэтому машинный язык рекомендуется использовать для создания операционных систем, библиотек стандартных программ в т.ч. ввода/вывода, которые расширяют возможности процессора. Также машинный язык применяется для создания ПО, на которое наложены ограничения по времени выполнения и занимаемой ёмкости памяти МПр.

Достоинства машинного языка следующие :

- компактность и высокая скорость выполнения программ;
- возможность непосредственного обращения и использования требуемых аппаратных ресурсов МПр;
- предсказуемость объектного кода и распределение памяти.

Недостатками машинного языка являются :

- привязка к системе микрокоманд и особенностям микроархитектуры данного МПр;
- трудоемкость процесса составления программ;
- низкая скорость программирования;
- невозможность непосредственного использования программ, составленных на определенном языке на МПр другого типа.

Для упрощения программирования в машинных языках часто используются языки символического кодирования, в которых коды операций и адреса в командах (см. раздел 1.11) вместо двоичного или шестнадцатеричного кода заменяются на символы (идентификаторы) или текстовые мнемонические коды, форма написания которых позволяет программисту лучше запомнить смысл выполняемой операции. Дополнительно здесь могут использоваться макрокоманды, которые рассмат-

ривались в разделе 1.1. Примером такого языка программирования является ***язык ассемблера*** – представляет собой символьную форму машинного языка с рядом возможностей, характерных для языка высокого уровня, включая макрокоманды (см. ГОСТ 19781-90). Ассемблер облегчает процесс программирования по сравнению с программированием в машинных кодах, потому что позволяет присваивать символические имена регистрам компьютера и памяти, а также позволяет задавать удобные способы адресации. Кроме того, он позволяет использовать различные системы счисления (например, десятичную или шестнадцатеричную) для представления числовых констант, использовать в программе комментарии и др. В результате существенно упрощается процедура, при которой для выполнения на МПр, программы на ассемблере транслируются в машинные коды с помощью трансляторов или компиляторов.

Транслятор – программа для перевода программ одного языка программирования на другой. ***Компилятор*** – транслятор, выполняющий перевод текста программы с проблемно-ориентированного или универсального языка на машинно-ориентированный язык, в машинный код. После этого машинный код может непосредственно запускаться на исполнения на МПр. ***Ассемблирование*** – процесс трансляции программы с языка ассемблера в машинный код.

К недостаткам ассемблера можно отнести привязку этого языка к конкретным типам процессоров. В частности, ассемблер в лабораторной работе [11] отличается от ассемблера процессора Intel Pentium.

В программе, составленной на машинном языке, используются определённые операторы для выполнения каждой вычислительной или логической операции. ***Оператор языка программирования*** – конструкция языка программирования, задающая одну или несколько операций, производимыми над операндами. При этом оператор точно указывает, где должны храниться числа (адрес ячейки запоминающего уст-

ройства, адрес операнда), как пересылать и обрабатывать числа и где хранить результаты вычислений. Срока программного кода с оператором на языке ассемблера транслируется, как правило, в одну машинную команду.

Вторая группа включает языки программирования высокого уровня, машинно-независимые, в котором команды и данные записываются в языковой форме, привычной для восприятия человека. «Высокий уровень» применительно к языку программирования означает, что многие операции выполняются в нем автоматически, поэтому программистам при решении той же проблемы приходится писать меньше программного кода. Например автоматически выполняются следующие операции:

- назначение регистров выполняется компилятором, не требуется писать программу для пересылки информации между регистрами и оперативной памятью;
- для организации циклов в программе можно использовать простые ключевые слова, такие как WHILE и IF - компилятор в процессе преобразования исходного текста в машинный код сам генерирует все необходимые для их реализации машинные команды.

В результате строка программного кода с оператором языка программирования высокого уровня транслируется примерно в три – семь машинных команд. Ко этой группе относятся ***проблемно-ориентированные*** языки программирования, отражающий особенности класса задач, для записи которых они предназначен. К проблемно-ориентированным языкам относятся Фортран, Алгол, Лисп. К языкам высокого уровня относятся ***универсальные языки*** программирования, не являющиеся машинно-ориентированными, но которые могут быть транслированы на различные языки. К универсальным языкам может быть отнесён язык программирования Си, часто используемый в средствах связи. Си – язык программирования общего назначения, который

с одной стороны по своим возможностям иногда превосходит ассемблер, с другой стороны – программы на Си могут запускаться на различных типах МПр. Язык Си поддерживает процедуры т.е. аппарат подпрограмм, используемый для решения той или иной задачи.

Подпрограмма – часть программы для ЭВМ, реализующая определенный алгоритм и оформленная таким образом, что допускает гибкую настройку на входные и выходные данные, называемые параметрами подпрограммы.

Отличительной особенностью языка программирования Си является большой набор операций, многие из которых соответствуют машинным командам, и поэтому допускают прямую трансляцию в машинный код. В результате программы, написанные на Си, сравнимы по скорости исполнения с программами, написанными на языке ассемблера. Язык Си позволяет однозначно описать алгоритм.

С учётом вышеизложенного, достоинства языков высокого уровня следующие :

- близость к человеческому языку;
- развитые средства автоматизации программирования и отладки программ, отсюда – высокая скорость разработки ПО;
- возможность непосредственного использования программ, составленных на определенном языке, на МПр разных типов.

Недостатками языков программирования высокого уровня являются:

- недостаточная компактность и меньшая скорость выполнения программ, чем у языков программирования первой группы;
- для запуска на МПр требуется трансляция или компиляция в язык низкого уровня (в машинный код);
- невозможность непосредственного обращения и использования требуемых аппаратных ресурсов МПр;
- необходимость распределения памяти;

Любой язык программирования имеет:

- лексику, т.е. определенный состав символов языка (A,B,C,…,Z; 0,…9; @,$,#,…);
- синтаксис – правило сочетания символов в слова и предложения (GO TO, MOV, ADD);
- семантика – правило предписывающие смысловое значение сочетания символов (GO TO <идти к …> или <переход к …>; MOV <перенос данных>, ADD <сложить>).

Лексика определяется системой счисления; синтаксис определяется форматом команд и данных; семантика определяется смысловым содержанием команд. Рассмотрим требования к системе команд машинного языка программирования для разработки программного обеспечения управления средства связи, имея в виду прежде всего язык ассемблер. Здесь возможно предъявить следующие общие требования:

Функциональная полнота системы команд означает, что система команд языка низкого уровня в средстве связи с программным управлением, должна полностью соответствовать паспортным функциям средства связи.

Обеспечение максимальной производительности УУ, что достигается путем оптимизации системы команд языка программирования низкого уровня, т.е. команда машинного языка должна иметь сравнительно небольшую длину (в битах), что обеспечивает выполнение требуемых операций МПр за минимальное число тактов.

Минимизация емкости ЗУ, требуемой для хранения программ управления – достигается применением совершенных алгоритмов, оптимальной длиной команд и рациональным размещением данных на запоминающих устройствах в процессе исполнения программы.

Рассмотрим используемые типы и форматы данных в языках программирования низкого уровня.

1.10 Типы и форматы данных

В процессе работы МПр используют данные и команды языка программирования низкого уровня различных форматов.

Под ***форматом*** понимается способ представления информации в двоичной форме для хранения и обработки в памяти УУ, где значение и номер позиции бита в ряду последовательно расположенных бит имеет содержательное значение. В первую очередь формат характеризуется длиной – количеством разрядов или бит в слове. Длина слова данных и команд должна соответствовать возможностям микропроцессора с одной стороны и возможностям операционной системы с другой стороны. Как правило, разрядность микропроцессора, превосходит или равна соответствующим возможностям операционной системы и программного обеспечения управления средства связи. С точки зрения разрядности, МПр может поддерживать следующие основные типы данных [16]:

байт — восемь последовательно расположенных бит;

слово — два байта, имеющих последовательные адреса. Слово делится на младший байт и старший. Младший байт всегда хранится по меньшему адресу, который является ***адресом слова.***

Существует также ***двойное слово*** — четыре байта, расположенных по последовательным адресам памяти. Двойное слово состоит из младшего слова и старшего слова. Младшее слово хранится по меньшему адресу памяти, который является адресом двойного слова. Иногда используется ***учетверенное слово данных*** — восемь байт, расположенных по последовательным адресам. Учетверенное слово делится на младшее двойное слово и старшее двойное слово. Младшее двойное слово хранится по меньшему адресу, который является адресом учетверенного слова.

В некоторых случаях применяется ***128-битный упакованный тип данных*** — формат, который впервые появился в МПр Pentium III.

Для работы с упакованным типом данных были введены специальные команды. Общее описание форматов перечисленных типов данных см. рис. 1.16.

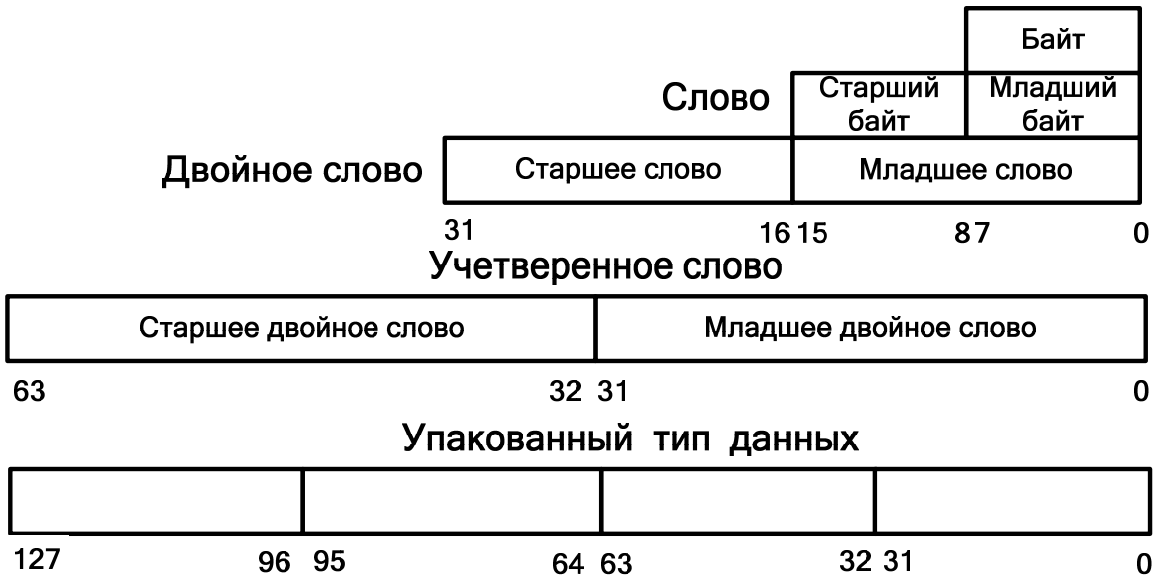


Рис. 1.16 – Примеры форматов данных МПр

Приведенные форматы данных позволяют поддерживать следующие логические данные:

Целое число со знаком (см. рис. 1.17). Максимальное значение такого числа равно 2^{n-1}- 1.

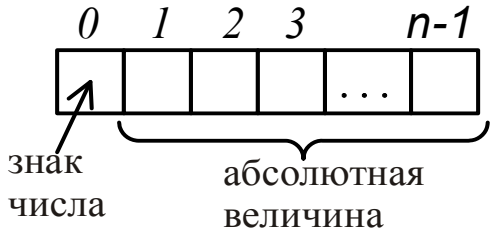


Рис. 1.17 – Формат целого числа со знаком

Основные значения целых чисел со знаком приведены в таблице 1.1.

Таблица 1.1. – Диапазоны целых чисел со знаком [16]

Размерность, бит	Без знака	Со знаком
8	0 … 255	− 128… +128
16	0 … 65 535	− 32 768 ... +32 767
32	0 … 4 294 967 295	− 2 147 483 648 ... + 2 147 483 648

Логический «0» в начале формата на рис. 1.17 означает положительное число; логическая «1» в начале формата на рис. 1.17 означает отрицательное число. Для данных размерности 64 бита диапазон возможных значений составляет от -9×10^{18} до $+9\times10^{18}$. Логический полноразрядный код см. на рис. 1.18:

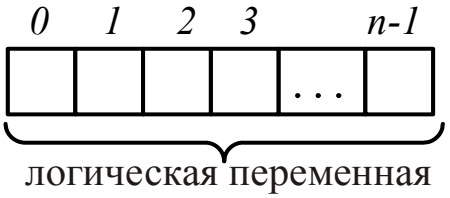


Рис. 1.18 – Формат логического кода

К логическому полноразрядному коду относится т.н. ***битовое поле*** — непрерывная последовательность бит. Каждый бит является независимым и может рассматриваться как отдельная переменная. Логический код также можно использовать при представлении чисел в виде:

неупакованный двоично-десятичный тип — байтовое представление десятичной цифры от 0 до 9. Числа хранятся как байтовые значения без знака по одной цифре в каждом байте (в младшей тетраде);

упакованный двоично-десятичный тип — представление двух десятичных цифр от 0 до 9. Каждая цифра хранится в своей тетраде (цифра старшего разряда — в старшей тетраде, цифра младшего разряда — в младшей тетраде).

Типы данных с плавающей точкой — специальные типы данных для обработки чисел с плавающей точкой в математическом сопроцессоре. ***Математический сопроцессор*** – сопроцессор, выполняющий операции над числами, представленными в форме с плавающей точкой. Сопроцессор – вспомогательный процессор, предназначенный для выполнения математических и логических действий, не входящих в стандартный набор команд ЦПУ.

Формат числа с плавающей точкой (вещественные числа) приведён на рис. 1.19:

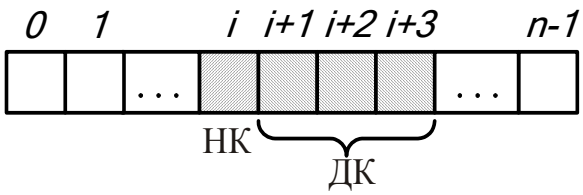
Рис. 1.19 – Формат чисел с плавающей точкой

Диапазон возможных значений вещественных чисел приведён в таблице 1.2 :

Таблица 1.2. – Диапазон значений вещественных чисел [16]

Формат	Короткий	Длинный	Расширенный
Разрядность (бит)	32	64	80
Разрядность мантиссы	24	53	64
Разрядность характеристики	8	11	15
Диапазон значений	$10^{-38}…10^{+38}$	$10^{-308}…10^{+308}$	$10^{-4932}…10^{+4932}$

Кроме приведенных двух основных форматов, данные могут записываться в виде формата «кусок» данных или «фрагмента данных» (см. рис. 1.20) :



Условные обозначения :
НК – начало куска
ДК – длина куска

Рис. 1.20 – Формат «куска» данных

Для работы с «куском» или фрагментом необходимо указать начало «куска». «Куском» или фрагментом могут являться целые числа или логически полноразрядный код.

Еще одним способ предоставления данных является массив данных. Под ***массивом данных*** понимается упорядоченная совокупность слов данных или команд, которые располагаются в смежных ячейках ЗУ. Массивы данных содержат тематически близкую информацию, например, данные о состоянии абонентских комплектов, данные о состоянии межстанционных линий, при этом схема массива данных имеет вид (см. рис. 1.21):

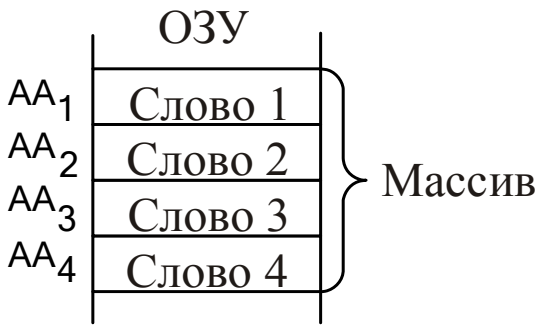


Рис. 1.21 – Формат массива данных

Применительно к массиву может использоваться относительная адресация, при которой абсолютный адрес (АА) ячейки памяти рассчитывается как начальный адрес массива плюс сдвиг: АА = НАМ + «сдвиг», где «сдвиг» – это целое число. В дальнейшем начальный адрес массива A_1 также именуется «базовым адресом». Средства связи чаще всего работают с массивами данных и обрабатывают числа с фиксированной точкой. Исключение составляют цифровые сигнальные процессоры в составе кодеров, систем сжатия изображений и т.п. В частности, для повышения качества обработки данных ПЦОС могут работать с плавающей точкой. Рассмотрим далее основные форматы и системы команд современного микропроцессора.

1.11 Системы и форматы команд

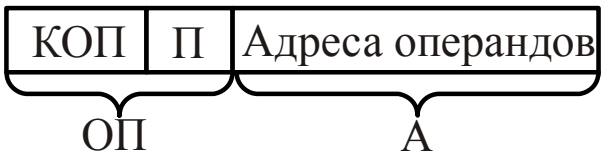
Набор команд машинного языка, которые может выполнять процессор, точно соответствует тем операциям над данными, которые может выполнять аппаратное обеспечение процессора с помощью микропрограмм. Другими словами, загружаемая в процессор программа для ЭВМ, написанная или транслированная на языке программирования низкого уровня, использует микрокоманды процессора для выполнения машинных команд, заданных программным обеспечением. Значение каждой микрокоманды (инструкции) имеет для данного типа процессора строго определенное значение. Например, инструкция 4 предписывает микропроцессору 8088 (или 8086), что нужно сложить значение, хранящееся в ячейке памяти, с содержимым регистра AL. Микрокоманды процессора не могут быть изменены загружаемым программным обеспечением.

Как отмечалось в разделе 1.9, загружаемое программное обеспечение МПр (программа для ЭВМ) для исполнения аппаратными средствами процессора должна быть преобразована в специальную форму, учитывающую только существующие микрокоманды процессора. Это преобразование осуществляется специальной программой – компилятором, на выходе которой получается машинный код в виде «0» и «1», который непосредственно считывается из оперативной памяти и интерпретируется микропроцессором.

Команды языка ассемблера могут предусматривать пересылку данных (между регистрами, между регистрами и внешними устройствами); могут инициировать операции ввода-вывода; могут требовать выполнения арифметических и логических операций. Также команды МПр могут предусматривать передачу управления, например при ветвлении в программе, передачу управления подпрограмме и возврат из подпрограммы в основную программу. Существуют также команды управления МПр, команды поддержки языков высокого уровня, команды под-

держки функций операционных систем по управлению памятью, команды для поддержки однотипных операций над упакованным форматом данных.

Структура любой команды языка низкого уровня (ассемблера), как правило, делится на две части (два поля), длина поля в битах или разрядах определяет формат команды [18]. Операционная часть указывает, что надо делать; для этого в ней содержится код операции; адресная часть указывает над каким операндами следует проводить операцию. С учетом сказанного формат машинной команды имеет вид (см. рис 1.22.) :



Условные обозначения :
КОП – код операции, задает тип операции, которая должна быть выполнена;
П – указывает на модификацию операции;
Адреса операндов – указание местонахождения операндов в памяти.

Рис. 1.22 – Общая структура машинной команды

В качестве содержания поле «П» может указываться размер и тип ожидаемого операнда.

Адрес в команде – часть машинной команды, которая указывает на устройство или элемент данных.

В адресной части могут указываться следующие значения :

- A1, A2, A3 – абсолютные адреса ячеек физической памяти, причём A1 и A2 – адреса ячеек операндов, A3 – адрес ячейки для записи результата.
- A4 – абсолютный адрес ячейки физической памяти с машинным кодом следующей команды
- R1,R2 – номера регистров общего назначения, где находятся операнды.

- B – значение базового адреса.
- D – смещение или сдвиг.

В зависимости от числа адресов, которые указаны в адресной части команды могут быть четырех-, трех-, двух- или одноадресные.

Четырехадресная машинная команда имеет формат (см. рис. 1.23) :

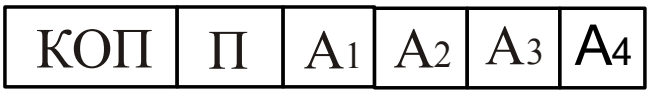


Рис. 1.23 – Формат четырехадресной машинной команды

Результат операции записывается по адресу A_3, при этом $A_{1...}A_3$ – номер ячейки памяти или номер регистра, A_4 – только адреса ячеек памяти. Такая машинная команда обеспечивает максимальную производительность МПр за счет того, что адреса операнда и адрес результата указывается явным образом. Однако в большинстве МПр поле A_4 не используется. Использование адреса A_3 в некоторых случаях также является избыточным. Поэтому большинство МПр серии x86 Intel или Motorola поддерживают двухадресные команды, которые имеют следующий формат (см. рис. 1.24):

Рис. 1.24 – Формат двухадресной машинной команды

Результат выполнения машинной команды записывается на место одного из операндов. Такая команда является наиболее часто используемой. Одноадресная команда имеет формат (см. рис. 1.25) :

Рис. 1.25 – Формат одноадресной машинной команды

Адрес первого операнда в одноадресной команде задается явно; второй операнд хранится в специальном неадресуемом регистре (аккумуляторы). Результат может записываться по адресу A1 либо в аккумулятор. Для задач управления, где большая часть времени уходит на пересылки данных и логические операции целесообразнее использовать двухадресные команды, а для многошаговых вычислительных процедур целесообразно использовать одноадресные команды.

Команды ПУСК, ОСТАНОВ процессора относятся к безадресным.

В некоторых случаях в адресную часть записываются не адреса данных, а сами данные. Этот приём позволяет увеличить быстродействие системы, но снижает гибкость программы. Кроме того, поле имеет ограниченную размерность, поэтому в «тело» команды могут быть записаны данные ограниченной размерности и, следовательно, величины. Рассмотрим основные систем команд, применявшихся в вычислительных устройствах [8,29].

Исторически наибольшее распространение получил МПр (вычислитель) со **сложной системой команд CISC** (complex instruction set computer), с достаточно большим перечнем команд. Основным для этой системы команд являлся формат на рис. 1.26.

Рис. 1.26 – Общая схема формата команды в системе CISC

На рис. 1.26 поле КОП содержит код арифметико-логической операции осуществляемой АЛУ, первое поле адресной части содержит номер регистра R1, где размещается первый операнд, а второе поле адресной части содержит обращение в память с базовым (начальным) адресом B2 и смещением (сдвигом) D2. Такой команде для выполнения требуется несколько тактов, что вызвано в первую очередь необходимостью работы с ОЗУ или кэш–памятью. Для системы CISC характер-

ны, прежде всего, большое количество способов адресации, включая специализированные способы индексации с помощью массивов; достаточно существенно число команд обмена типа <память–регистр>,<регистр–регистр>,< регистр–память>. Также здесь используется рассмотренная выше микропрограммная реализация выполнения команд. Различным командам для выполнения требуется различное число тактов. Сложный формат команд требует достаточно сложной конструкции управляющих устройств на кристалле МПр, в результате УУ физически может занимать до 50% площади кристалла МПр.

К примеру, общий формат CSIC–команды для процессора Intel 80x86 (согласно перевода статьи «Значение размеров инструкций Intel процессоров») показан на рис. 1.27 :

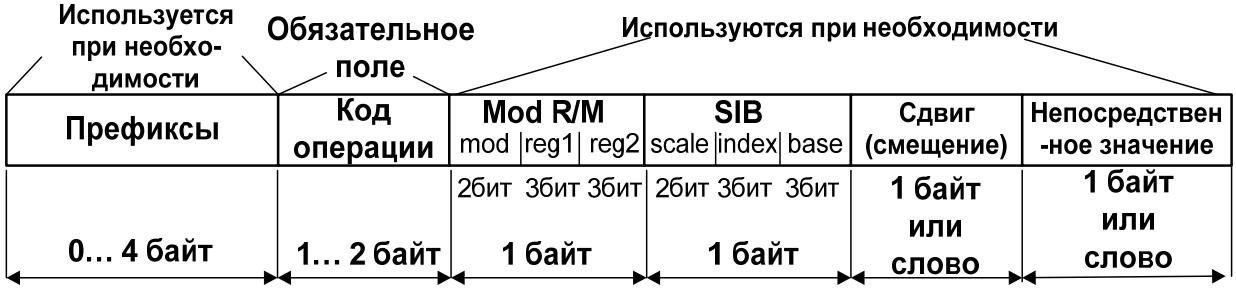


Рис. 1.27 – Формат инструкции системы CISC для процессора Intel 80x86
[Источник http://www.democoder.ru/dcdn_article_view.php?dcid=15]

Рассмотрим формат команды процессора Intel 80x86 по полям в порядке их следования слева направо.

«Префиксы» предназначены для изменения выполнения инструкций, например префиксы способны изменить установленный по умолчанию сегмент инструкции в физической памяти, изменить назначенный по умолчанию размер машинного слова, управлять загрузкой локальной шины процессора. Префиксы разделены на 4 группы, причём в поле «Префиксы» может быть только по одному префиксу из каждой группы.

Поле **«Код операции»** содержит данные об операционном коде, указывает процессору, какая инструкция должна быть выполнена. Кроме того, в состав данного поля могут входить биты, описывающие размер и тип ожидаемого операнда. Например, у инструкции NOT (логическая функция или операция «НЕ»,), соответствующий КОП = 1111011w, где бит **w** определяет, является операнд словом или байтом. Код операции «OR» = 000010dw, где бит **d** определяет, какой операнд является источником, а какой – адресатом, бит **w** определяет является операнд словом или байтом. В некоторых случаях размер т.е. разрядность поля кода операции может достигать 3 байт.

Поле **«Mod R/M»** указывает, какие регистры или сегменты физической памяти используются в качестве источников операндов. Поле mod указывает на значение операнда, например, если mod = 00, то reg1 содержит абсолютный адрес ячейки памяти операнда; если mod=11, то reg1 содержит собственно операнд. В свою очередь, поля reg1 и reg2 имеют формат трехбитных кодов регистра, и указывают коды регистров, которые содержат операнды для выполнения операции в поле **КОП**. Это соответствует непосредственной адресации. В некоторых операциях с плавающей точкой reg2 может содержать дополнительные биты КОП, а не код регистра.

Поле **«SIB»** (сокращённая форма от выражения Scale x Index + Base) используется только в случае 32-х разрядного режима и является расширением адресного формата. Это поле состоит из комбинации двух регистров (Index, Base) и масштабного коэффициента (Scale). Для составляющих данного поля используется расчётная формула $(Index \times 2^{\wedge Scale}) + Base$, результат вычислений по которой заменяет значение в поле reg1.

Поле **«Смещение»** используется при значении mod = 01 или mod = 10, значение данного поля представляет собой часть адреса операн-

да (см. ниже в разделе 1.12 схему относительной адресации). Поле **«Непосредственное значение»** содержит значение операнда.

Компьютер с **сокращенной системой команд RISC** (redused instruction set computer) более перспективен в случае использования процессоров с параллельной обработкой данных, поскольку число команд в такой системе меньше и, вследствие их простоты, большинство команд можно реализовать аппаратно и выполнить за один такт. Традиционно в RISC-системах повышение производительности достигается за счёт аппаратных решений, снижающих число обращений к оперативной памяти. Сокращение числа команд в RISC-системе достигается путем использования простых способов адресации и применением специальных команд загрузки и запоминания при обмене между регистрами и физической памятью. Обработка данных, в отличие от CISC, никогда по времени не совмещается с операциями чтения–записи в память. В системе RISC форматы команд разбиты на два простых формата одинаковой длины (32 бита), как это показано на рис. 1.28:

Рис. 1.28 а,б – Общая схема формата команд в системе RISC

Формат на рис. 1.28 а) предназначен только для работы с памятью, формат на рис. 1.28 б) предназначен только для работы с АЛУ, регистр общего назначения Ri – является источником второго операнда, результат вычислений можно поместить в поле Ri. Арифметические и логические команды являются трехадресными – указываются регистры

результата, регистр–источник (приемник) операнда и регистр–источник операнда.

Если предположить, что имеется k инструкций (команд) для работы с памятью и l инструкций (команд) для работы с АЛУ, то система команд типа CISC должна содержать всего $k \times l$ команд с форматом, приведённым на рис. 1.27. При тех же условиях система RISC с форматом на рис. 1.28 будет содержать $k+l$ команд. Следовательно, отказ от системы команд CISC значительно сокращает список системы команд, хотя при выполнении сложных операций над данными выигрыш может быть незначительным. В итоге, при понижении уровня системы команд увеличивается число служебных операций (команд), которые можно выполнять параллельно.

Итак, архитектура RISC за счёт упрощений команд, позволяет упростить аппаратные средства микропроцессора и благодаря этому повысить быстродействие МПр. Выполнение более сложных, но редко встречающихся команд обеспечивается подпрограммами. Большинство команд являются быстрыми пересылками типа <регистр – регистр> и выполняются без обращения к оперативной памяти; обращения к ОЗУ сохраняются лишь в командах загрузки регистров из памяти и записи в память. В связи с этим RISC - микропроцессоры должны иметь достаточно большое число регистров общего назначения (до нескольких десятков и даже сотен).

Современные процессоры стремятся гибко сочетать как CISC- так и RISC-решения.

Далее рассмотрим различные способы адресации, применяемые в рамках системы команд микропроцессора.

1.12 Способы адресации

Под ***адресацией*** здесь понимается способ указания операндов в системе команд.

Пример ***непосредственной адресации*** приведён на рис. 1.29. При непосредственной адресации операнд размещается непосредственно в «теле» команды.

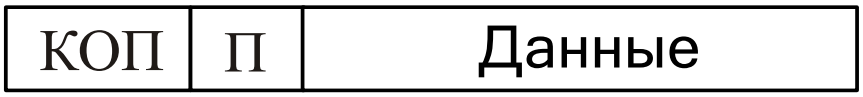


Рис. 1.29 – Схема непосредственной адресации

Достоинством непосредственной адресации является быстрота выполнения команды за счёт сокращения времени поиска операнда. Недостатком непосредственной адресации является фиксированная длина операнда и, следовательно, ограничение значения операнда. Непосредственная адресация оптимальна при работе с константами.

Пример ***прямой адресации*** приведён на рис. 1.30. В поле A_2 содержится номер регистра общего назначения или абсолютный адрес ячейки ОЗУ, где находится второй операнд.

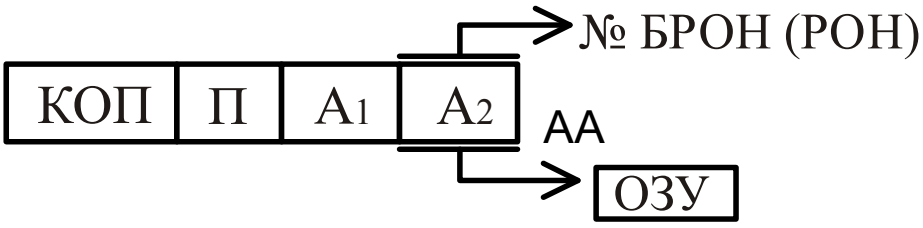


Рис. 1.30 – Схема прямой адресации

Достоинством прямой адресации является более гибкий, по сравнению с непосредственной, способ адресации. Недостатком является увеличение времени выполнения команды за счёт затрат времени на считывание (запись) операнда из БРОН (РОН) или ОЗУ. В случае, если

в поле A_2 указан № БРОН (РОН) говорят о прямой регистровой или регистровой адресации.

Пример **косвенной адресация** приведён на рис. 1.31. При косвенной адресации адрес А2 указывает на БРОН или на абсолютный адрес ОЗУ, где хранится абсолютный адрес требуемой ячейки физической памяти (в данном случае – ОЗУ).

Косвенная адресация широко используется в RISC-процессорах, её удобно использовать при обработке списков, при создании циклических программ. Достоинством косвенной адресации является увеличение гибкости программы – абсолютные адреса можно формировать в процессе выполнения программы.

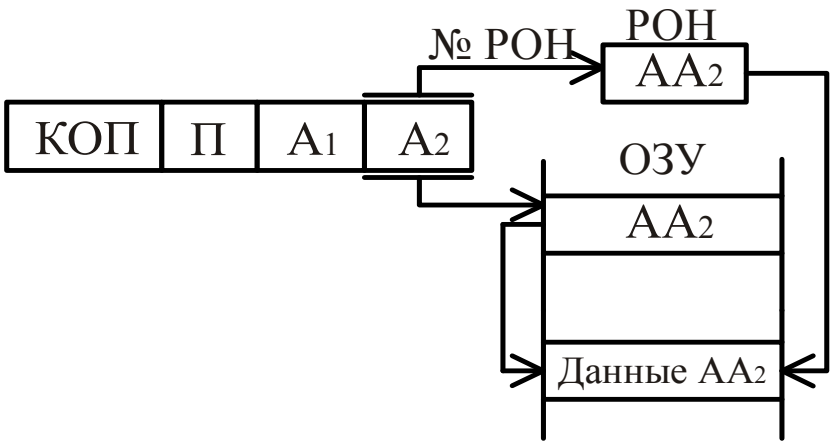


Рис. 1.31 – Схема косвенной адресации

Недостатком косвенной адресации является увеличение, по сравнению с прямой адресацией, времени выполнения команды за счёт затрат времени на считывание (запись) операнда. Кроме того, при использовании РОН (регистровая косвенная адресация) может потребоваться время на загрузку в РОН значения AA_2.

Пример **относительной адресации** или базирование приведён на рис. 1.32.

При относительной адресации абсолютный адрес, по которому находится 2-ой операнд, формируется как сумма начального адреса

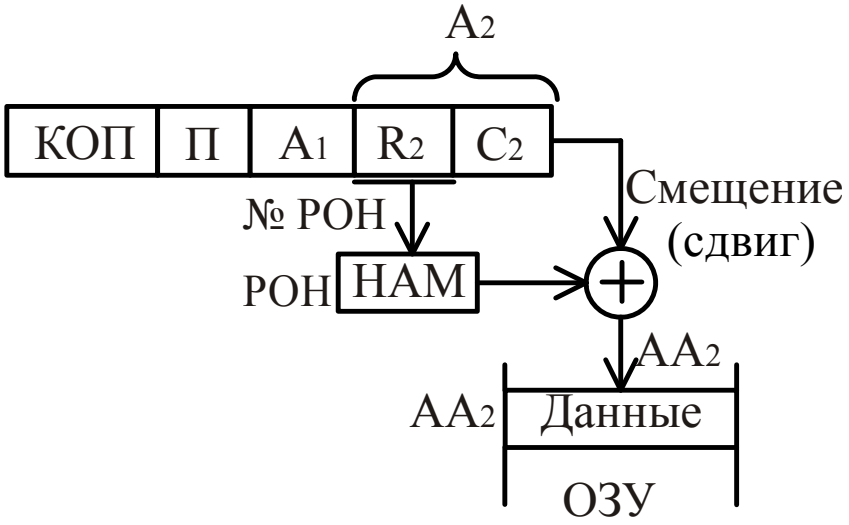


Рис. 1.32 – Схема относительной адресации

массива (НАМ) или базового адреса (не применяется для операций с массивами) в поле R2 и некоторого сдвига (конкатенации, приписывания разрядов) в поле C2; при этом НАМ хранится в БРОН или в другом прямоадресуемом регистре.

Данная адресация используется не только при работе с массивами однотипных элементов, но при организации виртуальной памяти, особенно при перемещении программ внутри адресного пространства МПр. Для этого требуется только определить базовый адрес, который записывается в счётчик команд, а смещение определяет адрес перехода относительно базового адреса.

Достоинством относительной адресации является оптимизация работы с массивами данных, особенно при обращении к последовательным ячейкам памяти, когда не нужно заново формировать НАМ. Недостатком относительно адресации является очередное увеличение времени выполнения команды, а также усложнение формата команды.

Техническое совершенство МПр не в последнюю очередь характеризуется доступными способами адресации и форматом поддерживаемых команд. Чем больше МПр поддерживает способов адресации, тем совершеннее могут быть программы, написанные на машинном языке и следовательно, микропроцессор может выполнять более слож-

ные операции по обработке данных различного назначения. Также не последнюю роль играет состав и возможности микрокоманд, микропрограмм, которые исполняются МПр непосредственно на аппаратном уровне.

1.13 Контрольные вопросы к главе 1

1. Дайте определение понятию «архитектура процессора».
2. В чём особенность архитектуры фон Неймана ?
3. В чём достоинства и недостатки гарвардской архитектуры?
4. Где применяется гарвардская архитектура ?
5. Какие процессоры (общего назначения или специализированные) используются в составе центрального управляющего устройства?
6. Для выполнения каких функций используются процессоры сетевой интерфейсной карты?
7. Где хранится адрес/номер следующей команды для выполнения процессором ?
8. С какой целью используется виртуальная память?
9. С помощью какого компонента виртуальные адреса пересчитываются в физические?
10. Какие существуют способы организации кэш-памяти?
11. Какой способ адресации применяется для максимального быстрого времени поиска операнда?
12. В чём достоинства и недостатки прямой адресации?
13. Какой способ адресации целесообразно применить в массивах данных?

2. Программное обеспечение средств связи

2.1 Состав и функции программного обеспечения средств связи

Как уже отмечалось, программное обеспечение в средствах связи, включая системы коммутации, используется прежде всего для управления средством связи. Управление средством связи осуществляется с помощью системы управления. Под системой программной понимается набор компонентов, объединенных для выполнения определенной функции или набора функций средства связи. Термин «система» здесь охватывает отдельные программные приложения, программные системы в традиционном смысле, подсистемы, линейки программных продуктов, семейства готовых программных продуктов, имеющие отношение к управлению. ***Компонентом*** называется модульная часть системы, которая инкапсулирует (содержит) часть содержимого системы, необходимую для функционирования данного компонента в составе системы. По определенному признаку, например функциональному, компоненты могут объединяться в подсистемы. В рассматриваемом случае речь идёт о программных т.е. логических компонентах.

Система управления современными средствами связи представляет собой интегрированный комплекс, состоящий из одного или более процессов, аппаратных устройств, программ, средств и людей, предоставляющий возможность удовлетворить определенную потребность или условие (согласно стандарта IEEE 12207). Применительно к средствам связи речь идёт о потребностях, связанных с оказанием услуг связи, передачей трафика или переносом сигналов электросвязи. Условием является удовлетворение перечисленных потребностей с качеством, соответствующим действующим нормам и правилам, и в объ-

ёме, требуемом для обслуживания подключенных к системе максимально допустимого числа пользователей. Для современных средств связи характерно автоматическое управление основными функциями и оборудованием с помощью загружаемого и хранимого, в т.ч. замонтированного, программного обеспечения. Поэтому можно говорить о том, что современная система управления средствами связи представляет собой ***преимущественно-программную систему*** т.е. систему, в которой программное обеспечение оказывает значительное влияние на проект, конструкцию, развертывание и развитие всей системы (согласно стандарта IEEE 1471).

Под ***архитектурой программного обеспечения системы управления средствами связи*** понимается набор значимых решений по поводу организации системы программного обеспечения, набор структурных элементов и их интерфейсов, при помощи которых компонуется (комплексируется) система управления, вместе с их поведением, определяемым во взаимодействии между этими структурными элементами, компоновка элементов в постепенно укрупняющиеся подсистемы. Примером структурного элемента является отдельная подсистема, процесс, база данных, аппаратное обеспечение, готовый программный продукт.

Программное обеспечение современных средств связи представляет собой сложный программный комплекс. Современные средства связи большой ёмкости (от 20 тысяч портов и более) имеют квазираспределенную функциональную архитектуру управляющего комплекса. Соотношение программного обеспечения и управляющих устройств различного назначения показано на рис. 2.1. В целом можно говорить о трёх уровня управления и, соответственно, о трёх уровня распределения программного обеспечения. Как видно на рис. 2.1, существуют общие компоненты программной системы, присутствующие на всех трёх уровнях. Рассмотрим эти компоненты более подробно.

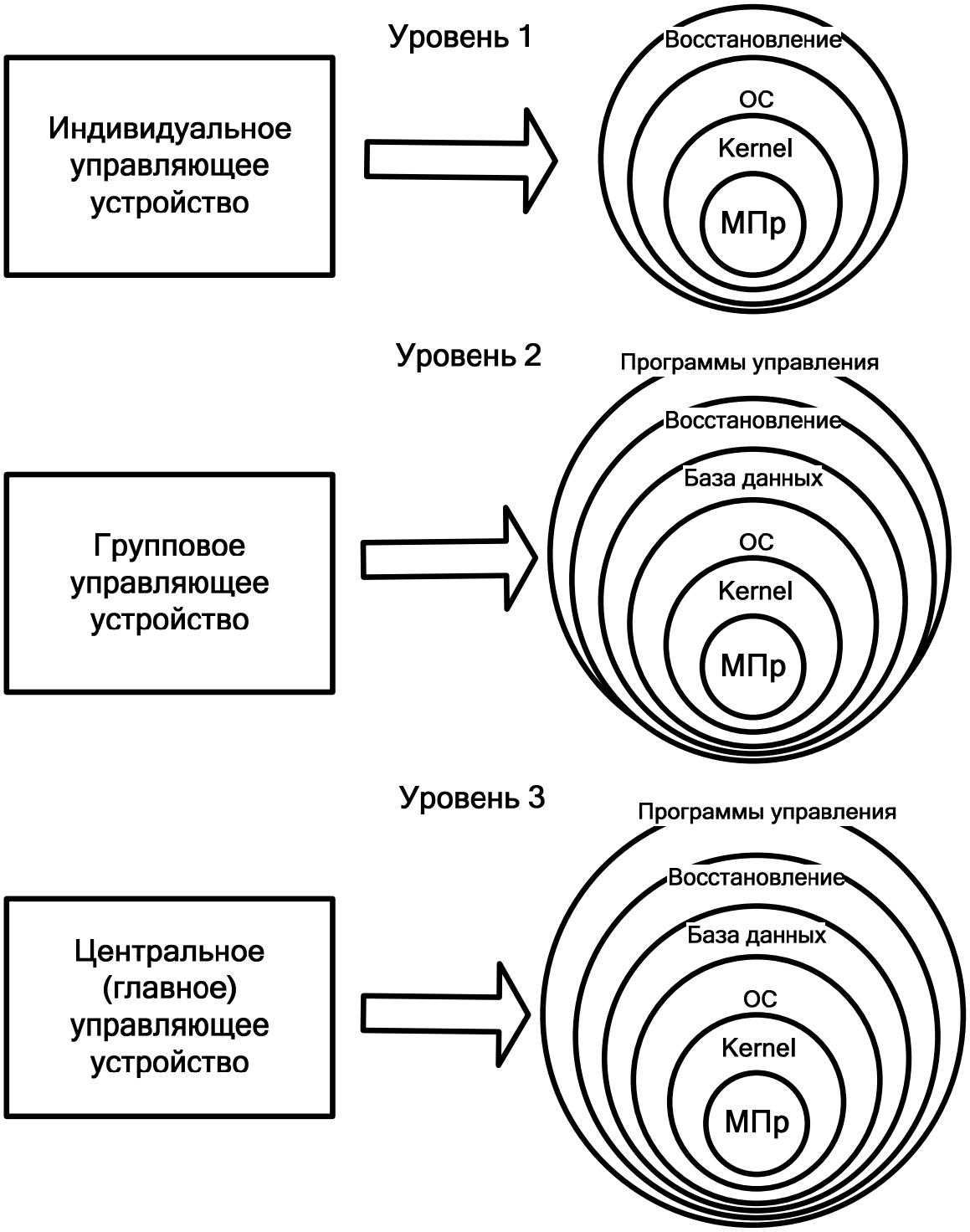


Рис. 2.1 – Управляющие устройства средств связи и их программное обеспечение [с учётом данных Syed R. Ali, Bellcore]

Основой, ядром программного обеспечения управления современным средством связи является операционная система. Подробнее операционные системы рассматриваются в разделе 2.2. Здесь же не-

обходимо сказать о том, что в средствах связи применяются операционные системы реального времени.

Операционная система – совокупность системных программ, предназначенная для обеспечения определенного уровня эффективности системы обработки информации за счёт автоматизированного управления её работой и предоставления пользователю определенного набора услуг. Операционная система, а также программы начальной загрузки МПр управляющих устройств, относятся к системному программному обеспечению. С учётом ГОСТ Р 51840–2001, ***системное программное обеспечение*** – программное обеспечение, определяющее функционирование средства связи как с прикладными программами так и без них. Системное программное обеспечение представляет собой множество подпрограмм, действующих как интерпретатор при преобразовании инструкций прикладных программ, загруженных в память МПр в виде машинных кодов, и требуемых устройствами аппаратного обеспечения.

Ядро операционной системы, kernel, представляет собой часть программного обеспечения операционной системы, которая хранится в ОЗУ все МПр управляющих устройств. Это не случайно, т.к. kernel обеспечивает выполнение самых элементарных функций операционной системы, таких как:

- оперативное управление и планирование процессов;
- управление основной памятью (регистры, кэш-память, ОЗУ);
- управление вводом/выводом запросов для внешних устройств и буферов;
- защита областей оперативной памяти в части запрета/разрешения операций записи/чтения;
- обслуживание прерываний с учётом приоритетов.

Управление ***базой данных*** предусматривает, что в современных средствах связи для станционных данных, статистических данных, дан-

ных об абонентах используются такая форма представления, организация, систематизация, чтобы эти данные могли быть найдены и обработаны с помощью управляющих устройств на основе МПр. Базы данных современных средств связи большой ёмкости, как правило, распределенные и реляционные. Распределенность означает, что УУ могут поддерживать свои базы данных, которые необходимо периодически синхронизировать с базами данных других УУ, прежде всего с базой данных ЦУУ. В реляционной базе все данные организованы в виде двумерных таблиц, причём каждый столбец в таблице включает данные одинакового типа (число, символ), столбец имеет уникальное имя, в таблице нет одинаковых данных. Между таблицами поддерживается система логических ссылок, что позволяет обратившись к одной таблице далее получить доступ к другим таблицам.

Программы восстановления работоспособности средств связи распределены между всеми управляющими устройствами. Восстановление предусматривает реализацию перехода средства связи в штатный режим эксплуатации после запуска/перезапуска программного обеспечения управления, ввода в эксплуатацию вновь установленной, замененной или восстановленной после сбоя аппаратуры. Восстановление может потребоваться, например, при полном пропадании электропитания, после отказа физического элемента. Повторный запуск программных и аппаратных средств в штатном режиме после завершения процесса восстановления или ремонта называется ***инициализацией*** (inizialization, INIT). Существует несколько уровней инициализации.

На первом уровне инициализации (INIT 1) осуществляется запуск в штатном режиме индивидуальных управляющих устройств ИУУ, модулей соединительных линий, модулей линий доступа, отдельного периферийного оборудования. Это так называемая локальная инициализация.

На втором уровне инициализации (INIT 2) осуществляется запуск в штатном режиме групповых управляющих устройств ГУУ, а также запуск блоков соединительных линий, блоков линий доступа, соответствующих управляющих устройств и периферийного оборудования, если INIT 1 не дал положительных результатов.

На третьем уровне инициализации (INIT 3) осуществляется запуск в штатном режиме центральных управляющих устройств, соответствующего им аппаратного и программного обеспечения. В результате может потребоваться перезапуск ГУУ (INIT 2), например для синхронизации баз данных устройств с последующим переходом к INIT 1. Уровень INIT 3 следует рассматривать как критический, перезапуск на таком уровне может привести к потере работоспособности средства связи вплоть до завершения INIT 1, на что может уйти несколько часов.

В случае полного или частичного неуспеха инициализации INIT 3, INIT 2 или INIT 1 средство связи переходит в режим ручного восстановления. В этом случае инженер по техобслуживанию и эксплуатации проводит детальное тестирование отказавших элементов, перезапускает программное обеспечение управления с внешнего носителя, заменяет отказавшее или сбойное оборудование на стативе. После выполнения процедур ремонта, перезагрузки вновь запускаются уровни INIT 1….INIT 3. При этом инженер имеет возможность определять направление и уровень процессов восстановления и инициализации.

Многие средства связи поддерживают концепцию построения программной системы управления как основной программы управления (generic program) или прикладной программной системы (application program system, APS). Основная программа управления включает все необходимые программные компоненты и структурные единицы для обеспечения функционирования средств связи. Можно выделить следующие шесть базовых компонент прикладного программного обеспечения управления средством связи :

Коммутационные программы – управляют процессами установления соединений и разъединения каналов/трактов, процессами коммутации в цифровом коммутационном поле, процессами маршрутизации и передачи кадров/пакетов, включая транзитные и тестовые соединения и сообщения. Также включают программы оперативного управления периферийным оборудованием и сетевыми элементами в процессе коммутации и передачи.

Программы технической эксплуатации (ТЭ) предназначены для автоматизации процессов контроля, диагностики, тестирования, восстановления оборудования и ПО средства связи, включают программы установок обновлений ПО и патчей (частных исправлений ПО без перекомпиляции всего программного кода ПО). Программы ТЭ так же используется при запуске оборудования и программного обеспечения в эксплуатацию после устранения отказа.

Административные программы предназначены для авторизации и идентификации пользователей программного обеспечения, для разграничения доступа пользователей к функциям программного управления средством связи с помощью паролирования и присвоения соответствующих полномочий. Административные программы используются для запуска задач по сбору статистических данных о трафике, сбору и предоставлению данных о состоявшихся соединениях (CDR-записи), сведений о работе оборудования системы коммутации.

Программы функциональных возможностей средства связи предназначены для реализации различных функций средства связи, которые могут реализовываться как дополнительные возможности в процессе функционирования коммутационных программ. Это относится к созданию узлов коммутации услуг интеллектуальной сети SSP (Service Switching Point), поддержке агента протокола SNMP, услуги Centrex и т.п.

База станционных данных предназначена для хранения сведений о конфигурации и составе аппаратуры средства связи, составе программного обеспечения. В базе станционных данных может указываться как фактически установленное оборудование (модули соединительных линий, модули линий доступа, модули пространственно-временной коммутации, ИУУ, ГУУ, ЦУУ) так и максимально допустимое число аппаратуры, которая может быть установлена в данной конфигурации. В базе станционных данных так же могут находиться таблицы со сведениями статистических счётчиков и регистров для записи и хранения данных по трафику, по количеству записей о состоявшихся разговорах, счётчики с указанием попыток вызовов/занятий в направлении связи, по данному тракту и т.п. База станционных данных позволяет формировать таблицы с описанием маршрутов пропуска нагрузки с привязкой к данным транспортной сети и сети доступа.

База абонентских данных предназначена для хранения информации об абонентах. При этом перечень информации об абонентах, список параметров и доступных характеристик, данные о назначаемых абоненту основных и дополнительных услугах связи предоставляет производитель средства связи. Оператор связи осуществляет информационное наполнение базы данных, осуществляет ввод актуальной информации по каждому абоненту, например номер абонента, тип номеронабирателя, IP-адрес порта, количество и порядок анализа цифр набора номера, код зоны семизначной нумерации, приоритет абонента или группы абонентов и т.п.

В заключение рассмотрим особенности функций программного обеспечения по всем трём уровням управления на рис. 2.1.

На уровне 1 осуществляется управление отдельными модулями с помощью МПр малой мощности. Каждый МПр поддерживает функционирование либо собственной малой ОС/kernel, либо часть общестанционной многопроцессорной/распределенной ОС. Часть функций управ-

ления здесь реализуется специализированными МПр со схемной реализацией алгоритма обработки данных либо с помощью замонтированного ПО. Это объясняется тем, что на уровне 1 вся обработка сигналов электросвязи осуществляется в реальном масштабе времени. Управление уровнем 1 осуществляется с уровня 2. Также на уровне 1, как описано выше, реализуются процессы восстановления и инициализации оборудования и программного обеспечения.

На уровне 2 осуществляется групповое управление отдельными блоками или группами блоков с помощью МПр средней мощности. Каждый МПр поддерживает функционирование либо собственной малой ОС/kernel, либо часть общестанционной многопроцессорной/распределенной ОС. Большинство функций управления здесь реализуется специализированными МПр с загружаемым ПО, например сетевыми процессорами. Допускается использование МПр общего назначения. На уровне 2 осуществляется обработка информационной части сигналов – заголовков пакетов/кадров, анализ цифр набора номера, сбор, обработка и анализ результатов тестирования и диагностики аппаратуры и программного обеспечения уровня 1 и уровня 2. На уровне 2 принимается решение о начале процедуры обслуживания пользователя, о завершении процедуры, формируется CDR, осуществляется фильтрация пакетов на основании анализа информационной части, обрабатываются сообщения поддерживаемых систем сигнализации. Управление уровнем 2 осуществляется с уровня 3. На уровне 2, как описано выше, реализуются процессы восстановления и инициализации оборудования и программного обеспечения этого уровня и управление процессами восстановления уровня 1. Используются компоненты программ управления и базы данных в части, касающейся работы данного ГУУ; не используются административные программы и программы функциональных возможностей.

На уровне 3 осуществляется единое управление всеми блоками или группами блоков, цифровым коммутационным полем, периферийным оборудованием с помощью МПр средней или большой мощности. Каждый МПр поддерживает функционирование либо собственной малой ОС/kernel, либо часть общестанционной многопроцессорной/распределенной ОС. Большинство функций управления здесь реализуется МПр общего назначения (универсальные МПр) с загружаемым ПО. На уровне 3 осуществляется маршрутизация вызовов или пакетов, общестанционный сбор, обработка и анализ данных тестирования и диагностики аппаратуры и программного обеспечения уровня 1, 2 и 3. На уровне 3 принимается решение о сценарии обслуживания пользователя, решаются проблемы маршрутизации трафика, осуществляется обновление и модернизация программного обеспечения управления, поддерживается система диалога «человек–машина», осуществляется управление уровнем 2. Используются все компоненты программ управления и базы данных в части, касающейся работы данного ЦУУ; могут не использоваться программы, уже функционирующие на уровне 2.

По способу хранения и обработки данных программное обеспечение управления средств связи можно условно разделить на внутреннее и внешнее. Внутреннее программное обеспечение постоянно (резидентно) находится в оперативной памяти средств связи или хранится в состоянии постоянной готовности к применению на накопителе на жёстком диске. Внешнее программное обеспечение физически постоянно хранится на внешних запоминающих устройствах (накопители на магнитных лентах, оптические и магнитооптические накопители), а также на отдельных персональных ЭВМ, иных вычислительных устройствах, не входящих в состав управляющего комплекса средства связи [1,17].

Внутреннее ПО включает ядро (kernel), операционную систему, программы восстановления и инициализации, базы данных и программы управления. Как правило, состав, структура и способы организации

внутреннего ПО неразрывно связаны со структурой и способами построения управляющего комплекса, типами используемых процессоров.

Внешнее ПО – это программное обеспечение, которое непосредственно не используется при штатной эксплуатации средства связи. Внешнее ПО включает в себя программы автоматизированного проектирования ПО средств связи, отладочные программы, испытательно-наладочные программы.

Программы автоматизированного проектирования применяются на этапе разработки и кодирования программного обеспечения для создания программ на языках программирования высшего уровня и в машинных кодах. ***Отладочные программы*** используются для контроля и тестирования целостности, корректности и устойчивости к сбоям оборудования разработанного ПО управления. Отладка может осуществляться на программных эмуляторах средств связи или на макетах оборудования. Как правило, в качестве макета оборудования используется серийный промышленный образец. Возможна автономная отладка отдельных модулей ПО с последующей комплексной проверкой ПО в целом. ***Испытательно-наладочные программы*** предназначены для обнаружения и локализации неисправностей в УК в процессе пусконаладки и эксплуатации.

Внешнее ПО не работает в реальном времени, т.к. оно не решает непосредственно задачи поддержки функционирования средств связи. В этой связи рассмотрим далее функции операционных систем, которые обеспечивают функционирование средств связи в реальном времени.

2.2 Функции, назначение, классификация операционных систем

Применительно к задачам средства связи, операционная система может рассматриваться как совокупность системных программ, предна-

значенная для обеспечения определенного уровня производительности управляющего комплекса за счет автоматизированного управления работой средства связи и предоставляемого пользователю определенного набора услуг. Операционная система обеспечивает управление процессами, использующими программно-аппаратные ресурсы.

Под ***ресурсом*** понимается логический (программный) или физический компонент управляющего комплекса в совокупности с режимом обработки данных, используемых данным ресурсом. В более общем плане ресурс является физическим или логическим средством, необходимым для функционирования процесса. Ресурсом может быть как аппаратура (процессоры, оперативная память, каналы ввода-вывода, периферийные устройства), так и общие программы, файлы, адресное пространство, машинное время В мультипроцессорных системах наиболее употребительными общими программными ресурсами являются системные таблицы, описывающие состояние системы и процессов, например, таблицы очередей к процессорам, таблицы очередей каналов ввода/вывода, таблицы распределения памяти.

Управление ресурсами со стороны операционной системы означает выполнение следующих действий:

- управление доступа к ресурсам со стороны внешних устройств и программ;
- распределение ресурсов между процессами (вычислительными задачами).

Основной единицей планирования для распределения ресурсов в вычислительной системе, в том числе в УК средства связи, является задача. ***Задача*** — это независимая единица (блок) вычислений, которая обладает собственными ресурсами и представляет собой реализацию процессов пользователя программного обеспечения средства связи. Типовой задачей является, например, реализация процесса обработки вызовов. Описание задачи, содержащее состав требуемых ре-

сурсов, время работы задачи и другие условия выполнения, составляются пользователем/разработчиком и входят в состав задачи. Задача состоит из одного или нескольких процессов. В результате выполнения задачи реализуется требуемая функция средства связи.

Под ***процессом*** в общем понимается последовательность действий или процедур, необходимых для реализации той или иной задачи средства связи, предписанных к исполнению соответствующим алгоритмом. В более общем плане, процесс есть работа, производимая SISD - процессором при выполнении программы с данными. Это означает, что процесс является результатом взаимодействия процессор – загружаемая программа. Одна программа или система программ может порождать несколько процессов, каждый из которых работает над своим набором данных. К примеру, коммутационная программа, решающая задачу коммутации входа и выхода, может порождать процесс поиска свободного соединительного пути в коммутационном поле, процесс маршрутизации вызова, процесс занятия требуемых канальных временных интервалов между входом и выходом для соединения.

Итак, процесс использует ресурсы и поэтому является одной из единиц для планирования распределения вычислительных заданий в многопроцессорной системе. В составе процесса на уровне операционной системы выделяют один или несколько потоков управления данными или командами. Поток можно рассматривать как минимальную «единицу выполнения» или «единицу планирования» (scheduling). Потоки могут выполняться независимо друг от друга, что встречается достаточно редко. Гораздо чаще, для одновременного выполнения нескольких потоков, требуется взаимодействие и синхронизация между ними, для чего в составе ОС применяются специальные службы и механизмы. Потоки могут создаваться и уничтожаться динамически, их количество внутри процесса может изменяться в зависимости от интен-

сивности и характера вычислительных задач. Детальнее эта проблема будет рассмотрена в главе 3 и 5.

Следует отметить, что более «узким» трактованием понятия «процесс» является выполняемое МПр программное приложение с собственным виртуальным адресным пространством, программным кодом, используемыми данными и другими ресурсами операционной системы, такими как файлы. В дальнейшем в рамках настоящего учебного пособия используется «общее» определение процесса, введённое выше.

Процессы могут находиться в последовательно сменяющих друг друга состояниях:

- Выполнение (активное состояние) – процесс непосредственно исполняется МПр.
- Ожидание (пассивное состояние) – процесс заблокирован по внутренним, по отношению к процессу, причинам; для его запуска на исполнение должно наступить некоторое событие, например таймер выдержки времени устанавливается в 0.
- Готовность (пассивное состояние) – процесс заблокирован по внешним по отношению к процессу причинам; например МПр временно занят выполнением других процессов.

Каждый процесс характеризуется своим описанием, которое называется дескриптором. ***Дескриптор*** – служебное машинное слово, которое создается специальной программной процедурой перед выполнением процесса на основе описания задачи и информации, имеющейся в программе, загружаемой в МПр. Дескрипторы хранятся в системной области памяти МПр и используются во время работы процесса. Уничтожение процесса вызывает также и уничтожение дескриптора. Дескриптор содержит описание идентификатора процесса, сведения о состоянии процесса, данные о приоритете процесса, место нахождения в физической памяти машинного кода, реализующего процесс, информация о ресурсах, которыми процесс может пользоваться (например,

открытые файлы, устройства ввода/вывода). Процессы могут объединяться в очереди с определённой дисциплиной облуживания, например «первый пришёл – первыё ушёл» FIFO (first input, first out).

Очереди процессов представляют собой дескрипторы отдельных процессов, объединенные в списки. Таким образом, каждый дескриптор, кроме перечисленной выше информации, содержит по крайней мере один указатель на другой дескриптор, соседствующий с ним в очереди. Также дескрипторы формируются для ресурсов и классов ресурсов. В класс объединяются однотипные ресурсы. Дескриптор класса ресурсов создается операционной системой и обычно содержит следующие компоненты:

1. Имя класса ресурса.

2. Состав ресурса, например: число процессоров в системе; число сегментов, страниц основной памяти.

3. Таблицу занятости единиц ресурса, которая указывает имена процессов, занимающих каждый ресурс.

4. Описание очереди ждущих процессов, которое указывает число элементов очереди, адреса начального и конечного элементов очереди.

5. Адрес или имя распределителя, ответственного за порядок занятия единиц ресурса процессами. Распределитель процессора обычно называется планировщиком или диспетчером. Работа диспетчера будет рассмотрена ниже.

Дескриптор может применять для описания массивов данных и команд. В этом случае дескриптор содержит сведения о размере массива данных, его местоположении в адресном пространстве, адресе начала массива, типе данных, режиме защиты данных и некоторые другие параметры данных. В частности, задание в дескрипторе размера массива данных позволяет контролировать выход за границу массива данных при индексации его элементов для ускорения поиска информа-

ции. При использовании дескрипторов обращение к информации в памяти производится только через них, т.е. дескрипторы можно рассматривать как дальнейшее развитие способа косвенной адресации.

В рамках управления доступом к ресурсам возможно создание т.н. ***виртуальной машины*** – когда операционная система с помощью специальной промежуточной программы или программно-аппаратного средства (виртуальная машина) скрывает от конечного пользователя (программист, внешнее устройство) внутренние особенности организации программного управления средства связи. Пример структурной схемы виртуальной машины представлен на рис. 2.2. Виртуальная машина позволяет представить один физический МПр как несколько независимых процессоров для каждой вычислительной задачи.

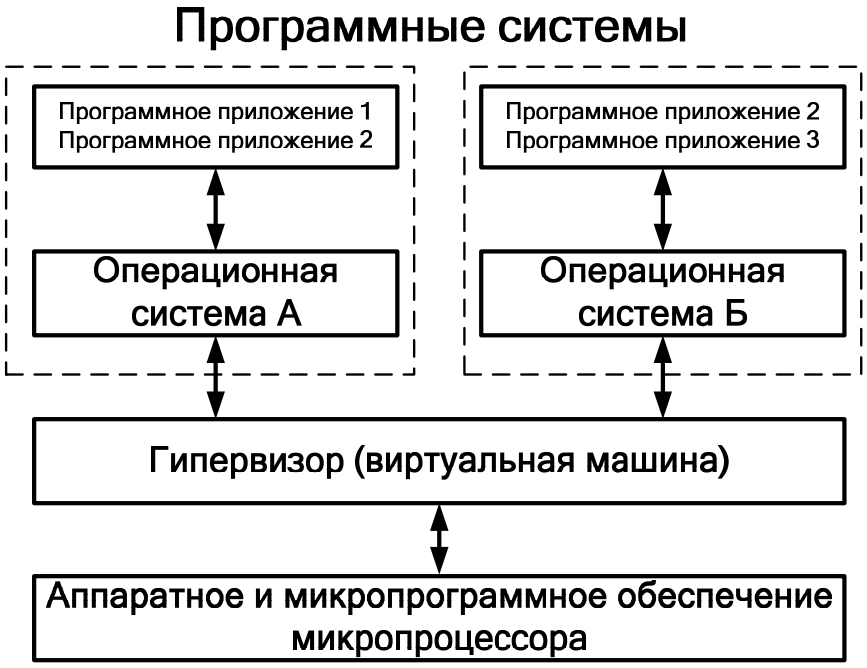


Рис. 2.2 – Структурная схема виртуальной машины

Таким образом, «виртуальная машина» позволяет оптимизировать использование ресурсов МПр.

Распределение ресурсов – важнейшая из функций, которые реализует ОС. Именно особенности выполнения данной функции являют-

ся определяющими для классификации ОС по признакам, приведённым на рис. 2.3.

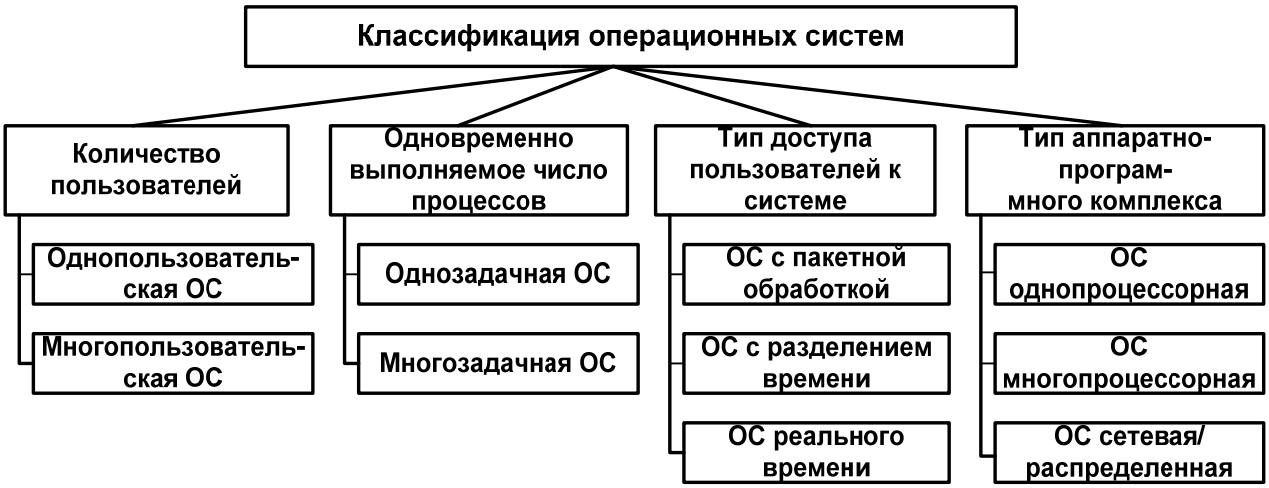


Рис. 2.3 – Классификация операционных систем [13]

По количеству пользователей однопользовательская операционная система может обслуживать только одного пользователя. Многопользовательская ОС позволяет создавать одноранговую или иерархическую вычислительную сеть с обслуживанием множества пользователей. Далее подробнее рассмотрим классификацию «Тип доступа пользователей к системе».

Пакетная обработка – режим выполнения совокупности задач, при котором все задачи выполняются автоматически, без синхронизации с событиями вне данной системы обработки информации, в частности без связи с лицами, представившими задание для выполнения. Операционные системы с пакетной обработкой предусматривают, что из программ, подлежащих исполнению формируется пакет, который загружается в управляющий комплекс и далее обрабатывается ОС и МПр. В этом случае пользователи непосредственно с ОС в процессе выполнения пакета, не взаимодействуют.

Разделение времени – процесс обработки информации, при котором ресурсы системы обработки информации (микропроцессорная

система) предоставляются каждому процессу из группы процессов обработки информации на интервалы времени длительность и очередность предоставления которых определяются управляющей программой системы обработки информации с целью обеспечения одновременной работы процессов данной группы в интерактивном режиме.

Интерактивный режим – режим взаимодействия в процессе обработки информации с человеком, выражающийся в разного рода воздействиях на этот процесс, предусмотренных механизмом управления конкретной системы и вызывающих ответную реакцию процесса.

Операционные системы с разделением времени обеспечивают одновременный интерактивный доступ нескольких пользователей через ОС к ресурсам управляющего комплекса. Ресурсы выделяются каждому пользователю «по очереди» на определенное время, согласно установленной дисциплине обслуживания, по приоритету.

Для многопользовательских и многозадачных ОС различают две основные дисциплины обслуживания процессов с точки зрения выделения процессорного времени на ту или иную задачу. Это вытесняющий и согласующий режим многозадачной работы.

При ***вытесняющем режиме*** распределением процессорного времени занимается только ОС, в результате чего для выполнения каждой задачи МПр занимается на строго определённый интервал времени с учётом приоритета задачи. Достоинством вытесняющего режима является отсутствие монопольного доступа к МПр для любой программы или задачи. Здесь же возможен запуск программ по предварительно составленному расписанию. Недостатком вытесняющего режима является негибкое и малоэффективное использование ресурсов процессора, например в том случае, если задача требует больше времени на исполнение, чем отведено расписанием.

При ***согласующем режиме*** каждая задача, получив управление ресурсами и доступ к МПр, сама определяет, когда МПр следует «от-

дать» другой задаче. Достоинством согласующего режима является возможность эффективного использования ресурсов процессора в случае перегрузок и критических повреждений, когда процессор должен обслуживать только программы самовосстановления, тестирования, эксплуатации или испытательно-наладочных программ.

Недостатком согласующего режима является возможность монопольного доступа к процессору со стороны одной программы, что влечёт за собой остановку выполнения других программ и повышает вероятность перезагрузки/перезапуска в случае, если произошёл сбой «многопольной» программы.

Под ***сетевой операционной системой*** понимается операционная система, обеспечивающая обработку, хранение и передачу данных в информационно-вычислительной сети. Сетевая операционная система определяет взаимосвязанную группу протоколов верхних уровней модели взаимосвязи открытых систем, обеспечивающих основные функции информационно-вычислительной сети: адресацию объектов, функционирование служб, обеспечение безопасности данных, управление сетью.

Примером современной сетевой многозадачной ОС является, например, операционная систем MS Windows Server 2003 Standard / Enterprise edition и MS Windows Server 2003 DataCenter Edition, которые поддерживают как 32-х так и 64-х разрядные вычисления. Для одного экземпляра ОС DataCenter Edition компанией Microsoft Corp., США заявлена поддержка 64 МПр и до 512 Гбайт ОЗУ, при наличии 8 вычислительных узлов в кластере, где кластер – полносвязная схема из 8 вычислительных комплексов, выполняющая параллельную обработку данных или работающую в режиме «горячего» резерва.

Также к сетевым многозадачным операционным системам относится ОС HP-UX.11i v.2, компания Hewlett-Packard, которая в рамках одного сервера поддерживает до ста двадцати восьми 64-х разрядных

МПр типа Itanium2 (конструкция с двухпроцессорными модулями), до шестидесяти четырех единичных 64-х разрядных МПр типа Itanium2, до 1 Тбайт ОЗУ и до тридцати двух узлов в кластере. Современными сетевыми многозадачными ОС также являются операционная система Solaris производства компании SUN Microsystems, операционная система AIX производства компании IBM, операционная система Red Hat Enterprise Linux производства компании Red Hat, США. Для современных ОС общего назначения производитель, как правило, указывает качественное назначение операционной системы – уровень отдела компании, небольшая фирма, предприятие (малое, большое), сервер приложений, сервер баз данных, сервер для центра обработки и хранения данных.

В средствах связи, в связи с чрезвычайно жёсткими требованиями к гарантированному времени обработки данных, применяются операционные системы реального времени (ОС РВ). Поэтому далее рассмотрим ОС РВ более подробно, но сначала рассмотрим понятие «прерывание», как одно из ключевых для понимания работы микропроцессорных систем.

2.3 Назначение и виды прерываний

Согласно ГОСТ Р 50304–92 под **прерыванием** понимается операция процессора, состоящая в регистрации предшествующего прерыванию состояния процессора и установление нового состояния. С учётом ГОСТ 15971–90, прерывание является реакцией процессора на некоторые условия, возникающие как в самом процессоре так и вне его.

Прерывания могут генерироваться внешними, по отношению к МПр, устройствами. Таким прерывания называют внешними или аппаратными. Также прерывания могут генерироваться при выполнении специальных команд в процессе исполнения программы.

При формировании сигнала прерывания сначала проводится идентификация устройства, которое сгенерировало запрос на данное прерывание. Текущее состояние регистров МПр, значение счётчика команд запоминается, чтобы после обработки прерывания вернуться к выполнению прерванной программы. Далее происходит обработка прерывания, например из памяти загружается и исполняется соответствующая программа обработки поступившего запроса на прерывание. Далее восстанавливается исходное, до-прерывания, состояние МПр и продолжает исполняться прерванная программа.

В результате обработки запроса на прерывание вырабатывается сигнал, передаваемый по специальным линиям прерывания. Линии прерывания служат для того, чтобы сигнализировать процессору через контроллер общей системной шины, который транслирует прерывания шины во внутренние прерывания процессора, что шине произошло некоторое событие. Если приоритет вновь поступившего прерывания больше, чем приоритет текущей задачи, МПр переключается с выполнения текущей задачи на обработку события, вызвавшего прерывание и запускает на исполнение соответствующую задачу (процесс).

Различают следующие виды прерываний [22]:

- ***Внутрипроцессорные прерывания*** – возникают при попытке МПр выполнить операцию с ошибочным кодом или в результате аппаратного сбоя.
- ***Внутрисистемные прерывания*** – возникают в случае тех или иных событий на внешних устройствах, не входящих в состав МПр, например прерывания от таймера, от устройств ввода-вывода, нарушение электропитания, ошибки обращения к общей системной шине.
- ***Прерывания, намеренно заложенные*** в программу, которую выполняет МПр; эти прерывания также называются планируе-

мыми программными прерываниями и часто применяются программистами для отладки ПО.

- ***Межпроцессорные прерывания*** – возникают при обмене данными между различными МПр.

К перечню, очевидно, надо добавить уже упомянутые прерывания от внешних устройств.

В микропроцессорных системах обычно используется одноуровневая система прерываний, т. е. сигналы «Запрос на прерывание» от всех внешних устройств поступают на один вход (порт) процессора. Поэтому возникает проблема идентификации внешнего устройства, запросившего обслуживание по прерывания. Возникает также задача реализации заданной очередности (приоритета) обслуживания внешних устройств при одновременном поступлении нескольких сигналов прерывания.

Существуют два основных способа идентификации внешних устройств, ВУ, запросивших обслуживания по прерыванию :

- программный опрос регистров состояния, в частности проверка состояния разряда «Готовность ВУ» контроллеров всех внешних устройств (ВУ);
- использование векторов прерывания.

При программном опросе в конце машинного цикла выполнения очередной команды процессор проверяет наличие требования прерывания от ВУ. Если сигнал прерывания есть, в процессоре прерывание разрешено, то процессор переключается на выполнение подпрограммы обработки прерываний. Начинается опрос регистров состояния контроллеров всех ВУ, работающих в режиме прерывания. Как только подпрограмма обнаружит готовое к обмену ВУ, сразу выполняются действия по его обслуживанию. Программный опрос используется только в тех случаях, когда отсутствуют жесткие требования на время обработки

сигналов прерывания внешних устройств. Приоритет ВУ определяется порядком их опроса.

При использовании векторов прерывания в некоторой области памяти ОЗУ существует таблица, где хранятся адреса процедур обработки прерывания. Эта таблица называется таблицей векторов прерываний. Как только сигнал прерывания получен, МПр выполняет команду перехода (безусловного или условного) на строку таблицы, соответствующей данному прерыванию. Затем по адресу, полученному из таблицы, производится переход в ту область памяти, где хранится подпрограмма обработки прерывания.

При использовании векторов прерывания для обработки прерываний может использоваться специальное устройство – программируемый контроллер прерываний (Programmable Interrupt Controller, PIC). Это устройство назначает приоритеты поступающим запросам на прерывание, выявляет запросы с наивысшим приоритетом. В случае поступления запроса на прерывание от внешнего устройства, программируемый контроллер прерываний формирует сигнал запроса прерывания INT (Interrupt) в сторону МПр. Если запрос обозначает прерывание, допустимое для данного типа МПр, то процессор генерирует в сторону программируемого контроллера прерываний сигнал подтверждения прерывания INTA (Interrupt Acknowledgement). Программируемый контроллер прерываний передаёт на шину данных МПр вектор (код) прерывания, который считывается МПр. МПР определяет физический адрес ячейки памяти, начиная с которого следует считать программу, обслуживающую прерывание.

При необходимости обслуживания большого числа источников запросов прерываний существует возможность каскадной схемы включения программируемых контроллеров прерываний, причём один из контроллеров будет ведомым, а остальные – ведомыми.

Для средств связи характерны программные прерывания, запускаемые по таймерам. Например, если абонент снял трубку, но не приступил к набору номера в течении 3 или 5 минут, соответствующий таймер отслеживает это событие. Как только таймер становится равным нулю, генерируется соответствующее прерывание. Получив это прерывание, ЦУУ или ГУУ запускает подпрограмму обработки такого «безотбойного» абонента. В результате в сторону абонента включается зуммер «занято», а сам абонент выводится из обслуживания, ресурсы управляющего комплекса на него затрачиваются минимально.

2.4 Операционные системы реального времени

Режим реального времени (real time processing) – режим обработки информации, при котором обеспечивается взаимодействие системы обработки информации (микропроцессорной системы) с внешними по отношению к ней процессами в темпе, соизмеримом со скоростью протекания этих процессов (см. ГОСТ 15971–90). В системах реального времени существенную роль играет время генерации выходного сигнала. Здесь сигнал на входе соответствует каким-то изменениям на управляемом объекте (физическом процессе). Выходной сигнал должен быть связан с этими изменениями. Поэтому временная задержка от получения входного сигнала до выдачи выходного сигнала должна быть небольшой, чтобы обеспечить приемлемое время реакции, например миллисекунды.

Программное обеспечение считается работающим в реальном времени, если его быстродействие адекватно скорости протекания физических процессов в системах связи [2,9]. Например, быстродействие ОС РВ системы коммутации должно быть таково, чтобы в ЧНН обеспечивать время установления межстанционного соединения 3..5 секунд при использовании системы сигнализации ОКС№7.

Операционная система реального времени обеспечивают гарантированное, заранее установленное время отклика ОС на внешние события. Таким образом, правильность функционирования системы реального времени зависит не только от логической корректности вычислений, предсказуемости поведения, но и от времени, за которое вычисления физически производятся. Предсказуемость поведения означает, что ОС РВ являются детерминированными системами, каждое действие которых в ответ на внешнее или внутренние событие является строго документированным и доступным пользователю или разработчику.

Следует отметить, что максимальное время отклика программы на внешнее событие или воздействие достигается максимальным использованием машинным кодом особенностей архитектуры и внутренних инструкций процессора. Например, в случае использования RISC–процессоров целесообразно применять одно- или двухадресные команды, чтобы в идеале они выполнялись за одни такт работы МПр.

Типовая ОС РВ должна соответствовать стандартам переносимых интерфейсов операционных систем POSIX (Portable Operating System Interface) [10]. Это необходимо, в первую очередь, для выполнения прикладных программ, в том числе для операционной системы UNIX. На практике эта задача решается с большой сложностью. Стандарты POSIX разрабатываются совместными усилиями исследовательских групп IEEE (общественная, некоммерческая организация), Американского национального института стандартов ANSI (American National Standard Institute, частная некоммерческая организация), Международной организации по стандартизации ISO (International Standard Organization, международная организация по стандартизации в рамках ООН, 146 стран), IEC (международная организация по стандартизации, 64 страны) а также Open Group (неправительственная международная организация по стандартизации программного обеспечения, 200 произ-

водителей). При реализации ОС РВ должны учитываться следующие стандарты POSIX для стандартизации программных интерфейсов :

- Стандарт IEEE 1003.1a OS definitions (определения ОС) – определяет основные интерфейсы ОС, управление заданиями, сигналы, функции файловой системы, работа с устройствами пользователей, конвейеры, буферы с дисциплиной обслуживания очереди FIFO.
- Стандарт IEEE 1003.1b Realtime Extensions (расширения реального времени) описывает сигналы реального времени, диспетчеризация по приоритетам, таймеры, синхронный/асинхронный ввод/вывод, разделяемая память, сообщения.
- Стандарт IEEE 1003.1c Threads (потоки) – определяет правила управления потоками, атрибутами потоков, диспетчеризацию потоков и процессов.
- Стандарт IEEE 1003.1d-1999 содержит требования к дополнительным расширенным возможностям ОС РВ.

В частности, стандарт POSIX 1003.1a даёт следующее определение: «Реальное время в операционных системах — это способность операционной системы обеспечить требуемый уровень сервиса в определённый промежуток времени». Под ***диспетчеризацией*** понимается применение методов оперативного управления, характеризующихся централизацией функции управления и контроля.

В связи с использованием в средствах связи к ОС РВ могут быть предъявлены следующие требования:

- операционная система должна быть многозадачной и допускающей режим вытеснения;
- операционная система должна поддерживать приоритеты для потоков команд и потоков данных;

- операционная система должна поддерживать предсказуемые механизмы синхронизации нескольких потоков управления вычислениями;
- операционная система должна поддерживать предсказуемость поведения т.е. давать совершенно определённый, заранее определенный отклик на конкретное внешнее воздействие или на внутренний сигнал при всех возможных рабочих нагрузках;
- операционная система должна поддерживать максимальное, известное наперёд, время отклика на внешнее событие при всех возможных рабочих нагрузках;
- операционная система должна обеспечивать безотказную работу т.е. обеспечивать существенные промежутки времени между сбоями и перезагрузками, а при необходимости – игнорировать сбой.

Здесь под сбоем понимается самоустраняющийся или перемежающийся отказ программных или аппаратных средств УК.

Состав, структура, машинный код ОС РВ в средствах связи как правило, жестко привязаны к используемом типы процессоров или иной аппаратуры. Жёсткая привязка ОС РВ к аппаратному обеспечению обусловлена необходимостью минимизировать задержки, для чего в максимальной степени учитываются аппаратные ресурсы и особенности архитектуры микропроцессора. Фундаментальное требование к использованию оперативной памяти ОС РВ заключается в том, что время доступа к данным в ОЗУ должно быть ограничено (или, другими словами, предсказуемо). ОС РВ, обеспечивающие механизм виртуальной памяти, должны уметь блокировать процесс в оперативной памяти, не допуская подкачки данных, временно выгруженных на НЖМД.

Классификация ОС РВ приведена на рис. 2.4.

Рис. 2.4 – Классификация ОС РВ [9]

Особенностью ОС РВ ***жёсткого реального времени*** (системы с детерминированным временем) является появление сообщения об отказе, если система неспособна обеспечить реакцию на событие в установленное заранее время. Отказ приводит к невозможности решить поставленную задачу и вызывает переход к блоку аварийных программ. Иногда к ОС жёсткого реального времени относят операционные системы, которые способны поддерживать необходимые временные требования к задачам реального времени даже при наиболее неблагоприятных нагрузках на МПр. Системы жёсткого реального времени чаще всего используются в системах контроля и управления. Эти системы сложны в реализации, что обусловлено высокими требованиями по безопасности. В системах жесткого реального времени обычно применяется статическое распределение физической оперативной памяти МПр с чётким закреплением адресов за данными и загружаемыми программами.

В ОС РВ ***мягкого реального времени*** (квази-реального времени) в случае, если система неспособна обеспечить реакцию на событие в установленное заранее время, сообщение от отказе не генерируется и ситуация не рассматривается как критическая. Операционная систе-

ма считается работающей в мягком реальном времени, если ОС РВ способна обеспечивать временные требования в среднем. В системах мягкого реального времени возможно динамическое распределение памяти с перераспределением адресов за данными и загружаемыми программами в зависимости от загрузки процессора и приоритетов выполняемых задач и процессов, но без использования механизма виртуальной памяти.

Специализированная ОС РВ может выполняться только на данном типе МПр, универсальная ОС РВ обладает свойством переносимости и может выполняться на нескольких типах МПр.

В настоящее время проводятся работы по стандартизации способов и протоколов взаимодействия ОС РВ. Основной целью стандартизации является выполнение нескольких микроядер (отдельных экземпляров) ОС РВ на одном МПр. Структура микроядра ОС РВ на примере ОС РВ QNX будет рассмотрена в разделе 2.5

Следует отметить, что режим мягкого реального времени может обеспечит и стандартное ядро операционной системы, например ядро ОС Linux 2.6. Согласно данным М. Тим Джонс (см. Тим Джонс, М Анатомия Linux-архитектур реального времени. Режим доступа [http://www.ibm.com/developerworks/ru/library/l-real-time-linux/]), ядро Linux 2.6 можно получить поддержку мягкого режима реального времени на основе простой конфигурации, которая обеспечивает для ядра полную вытесняемость (рис. 2.5).

Полная вытесняемость означает следующее. В стандартном ядре Linux 2.6, когда процесс с низким приоритетом выполняет обращение к ядру ОС, то процесс с высоким приоритетом должен ждать пока обработка запроса процесса с низким приоритетом не будет завершена. Только после этого процесс с высоким приоритетом сможет получить доступ процессору.

Полная вытесняемость означает следующее: в стандартном ядре 2.6 Linux, когда процесс с низким приоритетом выполняет обращение к ядру ОС, то процесс с высоким приоритетом должен ждать, пока

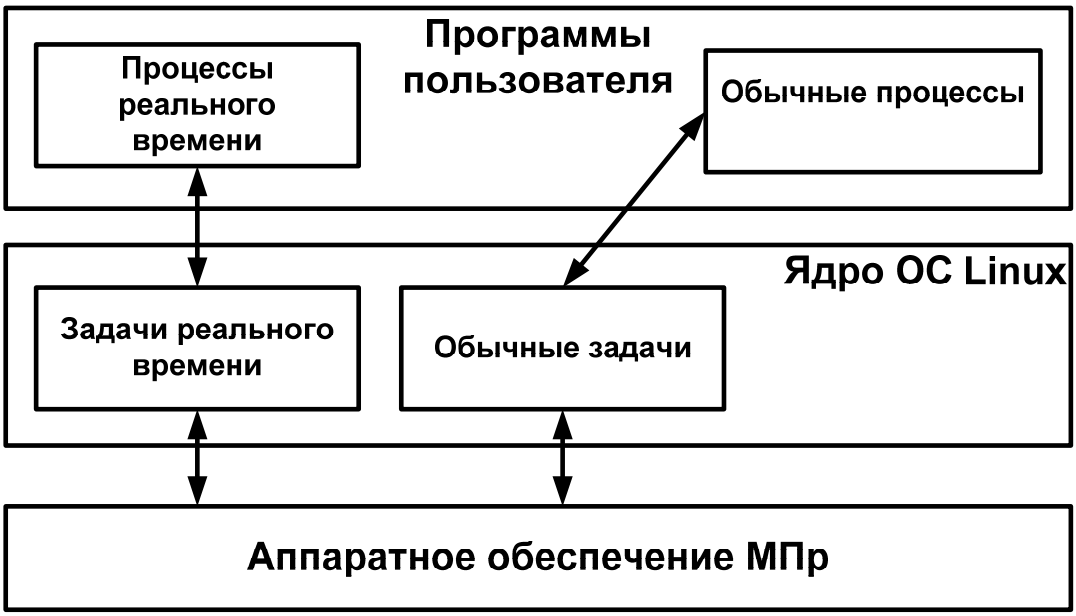


Рис. 2.5 – Использование ядра ОС Linux 2.6 в системах мягкого реального времени

обработка запроса процесса с низким приоритетом не будет завершена. Только после этого процесс с высоким приоритетом сможет получить доступ процессору. Это обеспечивает поддержку мягкого реального времени, хотя приводит к небольшому снижению пропускной способности и уменьшению производительности ядра ОС. В средствах связи средней и большой ёмкости этот метод неприменим.

Для перехода к режиму жёсткого реального времени, к обычному ядру Linux можно добавить т.н. тонкое ядро ОС, которое будет выполнять задачи реального времени. Другим назначением тонкого ядра является управление прерываниями. Тонкое ядро перехватывает прерывания, благодаря этому гарантируется, что работа тонкого ядра не будет прервана другим ядром Linux, которое не выполняет задачи реального времени. Это позволяет тонкому ядру обеспечить жесткую поддержку режима реального времени. Все обращения к аппаратной части

МПр идут только через тонкое ядро. Примерами реализации такого подхода являются RTLinux (принадлежит Wind River Systems), Real-Time Application Interface (RTAI) и Xenomai. Недостатком здесь является сложность отладки и совместного функционирования задач реального времени и задач отложенного времени.

К наиболее распространённым типам ОС РВ относятся QNX, LynxOS, OS-9, VxWorks/Tornado, Windows CE, UNIX-RTR, Virtuoso (для процессоров цифровой обработки сигналов), ОС2000 (разработана Научно-исследовательским институтом системных исследований Российской академии наук по заказу Министерства обороны России для МПр Intel и RISC–процессоров). ОС2000 выполнена в соответствии со стандартами POSIX, прошла государственные испытания, рекомендована для применения в частях и подразделениях Министерства обороны России.

2.5 Состав, структура и функционирование ОС РВ QNX

В настоящем учебном пособии в качестве примера ОС РВ рассмотрим ОС РВ QNX Software Systems (QSSL, www.qnx.com). Последняя версия этой ОС РВ QNX Neutrino 6.3 [22], подробно описанная в руководстве Операционная система реального времени QNX Neutrino 6.3. Системная архитектура: Пер. с англ. – СПб.: БХВ-Петербург, 2006, имеет микроядерную архитектуру и обладает следующими особенностями :

- разрабатывалась как сетевая операционная система;
- полная поддержка симметричных микропроцессорных систем;
- поддержка трёхмерных графических спецификаций OpenGL;
- поддержка разнообразных файловых систем на перепрограммируемых постоянных запоминающих устройствах (ППЗУ), НЖМД, CD/DVD;

- улучшенная поддержка сетевых протоколов TCP/IP v4, TCP/IP v6, IPsec;
- поддержка спецификаций Java J9 (компания IBM), Webshere Embedded Environment, совместимую с Java2 Mobile Edition (производство Sun Microsystems, США);
- поддержка асинхронного обмена сообщениями;
- поддержка интерфейса USB 2.0, поддержка памяти ОЗУ ёмкостью до 4 Гбайт;
- ориентация на языки программирования высокого уровня Java, Си.

В частности, маршрутизатором нового поколения Cisco CRS-1 управляет ОС РВ Cisco IOS XR, основанная на QNX Neutrino. В результате появляется возможность построить систему маршрутизации на сетях связи со скоростью передачи до 92 Тбит/сек с поддержкой оптического интерфейса (Optical Carrier) OC-768c/STM-256c. Здесь же обеспечивается поддержка функционирования 1152 физических разъёмов (посадочных гнёзд) для линейных модулей, работающих со скоростью 40 Гбит/с. Такая высокая скорость передачи данных обеспечивается с помощью сетевого процессора Cisco Silicon Packet Processor (SPP) на базе специализированной микросхемы (ASIC). ОС РВ QNX Neutrino построена на принципе организации микроядра (microkernel) и обмена сообщениями между процессами. ОС РВ реализована в виде множества независимых процессов различного уровня (менеджеры и драйверы). Особенностью архитектуры ОС РВ является то, что драйверы внешних устройств, файловые системы, стеки сетевых протоколов, приложения пользователей запускаются вне микроядра, и рассматриваются ОС РВ как отдельные задачи, использующие различные области памяти с помощью механизма защиты памяти. Поэтому размер микроядра составляет 10 Кбайт. Обмен сообщениями между процессами осуществляется с помощью механизма IPC (см. рис. 2.6).

Рис. 2.6 – Микроядро ОС РВ на примере QNX версии 4.X
[Источник www.aswl.ru/articles/rtos/gpo]

Микроядро осуществляет :

- управление потоками и сигналами (согласно POSIX);
- синхронизацию потоков;
- управление таймерами;
- планирование процессов (потоков);
- поддержку обмена сообщениями между всеми процессами (потоками) в системе.

Рассмотрим часть этих функций более подробно.

Специальная программа – планировщик – является частью микроядра и запускается всякий раз, когда в результате сообщения или прерывания изменяется состояние процесса. В QNX каждому процессу назначен приоритет. Приоритет также связан с прерыванием. Как уже говорилось, прерывание – это операция процессора, состоящая в реги-

страции предшествующего прерыванию состояния процессора и установление нового состояния (см. также раздел 2.3). Прерывания можно рассматривать как служебный сигнал, поступающий от аппаратного или программного обеспечения, и предназначенный в том числе для изменения порядка выполнения программ. Микроядро ОС РВ QNX в первую очередь выполняет обработку аппаратных прерываний; в результате все аппаратные прерывания и ошибки сначала направляются в микроядро, затем соответствующему драйверу или менеджеру системы. Подробнее прерывания будут рассмотрены ниже.

С помощью анализа приоритетов готовых к выполнению процессов, планировщик выбирает процесс, который будет выполняться следующим по порядку. Эти процессы в терминах QNX обозначаются как готовые к исполнению (READY) – т.е. эти процессы потенциально готовы использовать МПр. Для выполнения выбирается процесс с самым высоким приоритетом. Согласно приоритету формируется очередь готовности на исполнение. Для выполнения выбирается первый поток с наивысшим приоритетом. Фактически очередь готовности ОС РВ QNX Neutrino 6.3 состоит из 256 очередей – по одной очереди на каждое из двухсот пятидесяти шести имеющихся прерываний.

Учитывая, что процессы состоят из потоков, планирование процессов сводится к планированию потоков. READY-поток помимо собственно исполнения, может находиться в следующих состояниях :

- блокироваться, если он ожидает внешнего события, например ответа на запрос IPC. Блокированный поток удаляется из очереди готовности на обработку МПр, после чего запускается поток с наивысшим приоритетом.
- вытесняется и прерывается потоком с наивысшим приоритетом из очереди готовности на обслуживание. Вытесненный поток сохраняет свой приоритет и становится в начале очереди на исполнение.

- поток отдаёт управление МПр другому потоку и становится в конец очереди на исполнение, сохраняя свой приоритет.

Способы (дисциплины) планирования запуска процессов (потоков) в ОС РВ QNX следующие:

- Планирование в порядке поступления (FIFO scheduling) – поток выполняется пока не будет блокирован или вытеснен потоком с более высоким приоритетом.
- Циклическое планирование (round-robin scheduling) – поток выполняется, пока не закончится отведённый ему квант времени (time slice*)*, или не появится более высокоприоритетный поток.

Примечание. Под квантом времени понимается единица времени, выделяемая каждому процессу. В QNX эта единица равна 4-х тактовому периоду Clock Period(), где ClockPeriod() – функция, позволяющая установить значение системного таймера, кратное наносекундам.

- Адаптивное планирование (adaptive scheduling) – если поток не закончен, а квант времени истёк, то приоритет процесса уменьшается на 1 и выполняется следующий готовый к выполнению процесс.
- Спорадическое планирование (sporadic scheduling) – потоку отводится верхний лимит (бюджет) времени на время исполнения в пределах данного периода времени. Потоку выделяется бюджет времени на исполнение с нормальным или пониженным приоритетом. Бюджет процесса может периодически пополняться. Спорадическое планирование позволяет обеспечить более точное управление потоком. При этом потоки могут быть как периодическим так и апериодическим (случайным) и выполняться не препятствуя друг другу.

Пользователь может изменять приоритет с помощью программных настроек, например с помощью функции setprio().

Внутри микроядра передача сообщений между процессами (а фактически – между потоками) осуществляется синхронно и напрямую между передатчиком и приёмником, здесь же обеспечиваются средства

синхронизации выполнения нескольких процессов. В результате обеспечивается взаимодействие программных приложений, использующих 32-х битовые и 16-ти битовые коды. Микроядро обрабатывает не только собственные сообщения но и пакеты стека протоколов TCP/IP. Для обмена используются стандарты POSIX, в частности процедуры *open(), read(), write().*

Например, при обмене сообщениями микроядра с драйверами внешних устройств процедура ***open()*** работает следующим образом :

1. Каждый драйвер внешнего устройства регистрируется в каталоге и получает область допустимых имён и значений.

2. Программное приложение обращается через open() к дереву имён т.е. к программному каталогу, где зарегистрирован драйвер.

3. В ответ программное приложение получает дескриптор файла, с помощью которого может формировать запросы к файлу драйвера.

При формировании запросов используются стандартные библиотеки языка программирования Си. Все сообщения представлены с помощью языка программирования Си. В рамках QNX для обмена сообщениями между процессами используются запросы типа *send(), receive(), reply().* В частности, формат сообщения ***send ()*** отправленного процессом А процессу Б для версии QNX 4.3 выглядит следующим образом:

send (pid, smsg, rmsg, smsg_len, rmsg_len),

где

pid – идентификатор процесса Б, который должен принять сообщение (то есть процесса Б);

smsg –*б*уфер сообщения (для посылаемого сообщения)

rmsg – *б*уфер ответа (для ответа, полученного от процесса Б)

smsg_len – *д*лина сообщения в байтах;

rmsg_len – *м*аксимальная длина ответа в байтах, который будет принят процессом А

Также поддерживается асинхронный обмен сообщениям с помощью сигналов. Примером такого сигнала (для ОС РВ QNX версия 4.x) является сигнал SIGPWR – перезагрузка программного обеспечения, вызываемая одновременным нажатием на клавиатуре компьютера клавиш Ctrl-Alt-Del или запуском прикладной программы shutdown.

Операционная система QNX Neutrino использует технологию отказоустойчивости FLEET (Fault tolerance), регулирования нагрузки (Load-balancing), эффективность (Efficient). Компания QNX Neutrino самостоятельно заявила о соответствии ОС РВ QNX стандартам IEEE 1003.1a, IEEE 1003.1b, IEEE 1003.1c, IEEE 1003.1d образца 2001 г.

В настоящее время в QNX Neutrino 6.3 используется мультиядерная структура, представленная на рис. 2.7. Обмен сообщениями между микроядрами здесь осуществляется с помощью усовершенствованного механизма IPC в виде виртуальной шины обмена сообщениями. Виртуальная шина для обмена сообщениями может быть реализована с помощью Ethernet, стека протокола TCP/IP.

Достоинством технического решения на рис. 2.7 является возможность подключения к шине достаточно широкого набора устройств. При этом подключенные устройства становятся доступными всем процессам и микроядрам.

Менеджеры процессов осуществляют дополнительные функции, в частности управления памятью и управление каталогами (деревом каталогов), а также дескриптором файла (file descriptor). Существуют следующие виды менеджеров ОС РВ QNX, осуществляющих управление на уровне процессов :

- Менеджер процессов (Proc) – отвечает за создание новых процессов в системе и управление наиболее фундаментальными ресурсами, связанными с процессами (создание, загрузка, выполнение, завершение процессов). Процессы могут иметь сим-

волические имена. Также менеджеры процессов осуществляют поддержку работы таймеров и обработку прерываний.

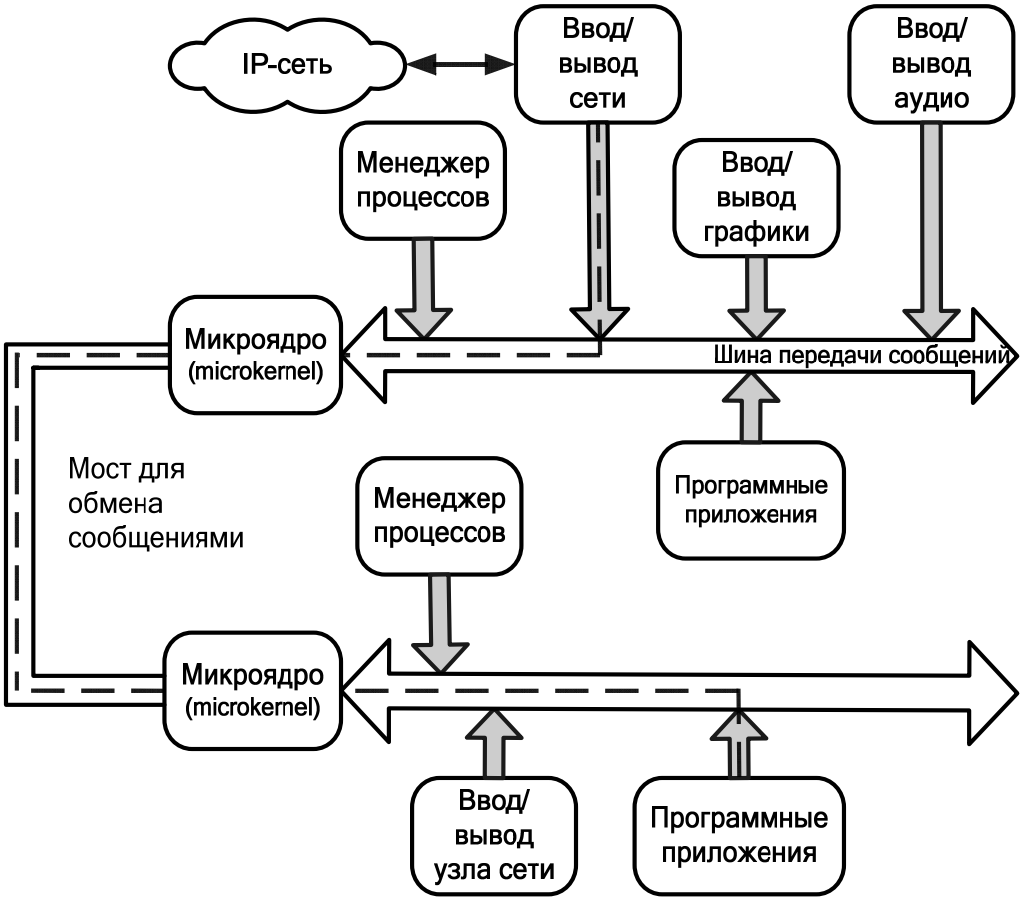


Рис. 2.7 – Мультиядерная структура ОС РВ QNX Neutrino 6.2
[Источник: http//www.qnx.org]

- Менеджер файловой системы (Fsys) – обеспечивает стандартизированные средства хранения данных на дисковых подсистемах и доступа к ним. Fsys отвечает за обработку всех запросов на открытие, закрытие, чтение и запись файлов в различных стандартах FAT, FAT32, NTFS, UNIX. В результате один и тот же НЖМД может использоваться различными ОС. Менеджер файловой системы поддерживает каталоги, каналы ввода/вывода (в части специального файла для обмена между процессами), структуру ОС (например блок загрузчика ОС РВ, корневой каталог, битовую карту накопителя на жёстком диске).

- Менеджер устройств (Dev) – управляет взаимодействием с устройствами с помощью специальных драйверов. Услуги ввода-вывода обеспечиваются процессами, которые могут быть вызваны динамически во время работы системы. Все драйверы выполняются как пользовательские потоки и не относятся к процессам микроядра.
- Менеджер сети (Net) – ответственен за распространение сообщений QNX на локальную сеть и в сеть Интернет (IP-сеть).

На рис. 2.7 показано, как программный процесс, поддерживаемый одним микроядром, обрабатывает данные, полученные через порт ввода/вывода, поддерживаемый другим микроядром. Каждое микроядро может функционировать на отдельном процессоре. В итоге реализуется унифицированный доступ ко всем аппаратным средствам и программным ресурсам, причём без проверки прав доступа. Для рассматриваемой схемы характерен быстрый обмен сообщениями между микроядрами как встроенная функция операционной системы. Предлагаемая ОС РВ может использовать в качестве аппаратного обеспечения двух– и более ядерный процессор, причём каждое микроядро может запускаться на отдельном аппаратном ядре процессора. Подробнее о многоядерных процессорах см. главу 5.

Схема на рис. 2.7 позволяет реализовать распределенные вычисления со следующими функциональными и технологическими возможностями:

- поддержка стандартов POSIX;
- динамическое взаимодействие между оборудованием и программным обеспечением на разнесённых системах коммутации;
- поддержка службы глобальных имен для обнаружения нового оборудования и программных приложений;

- останов исполнения программного приложений на одной системе коммутации и перезапуск программного приложений на другом управляющем комплексе без необходимости перезагрузки программного обеспечения.

Дополнительно QNX Neutrino поддерживает возможность работы симметричных мультипроцессорных систем по схеме на рис. 2.8.

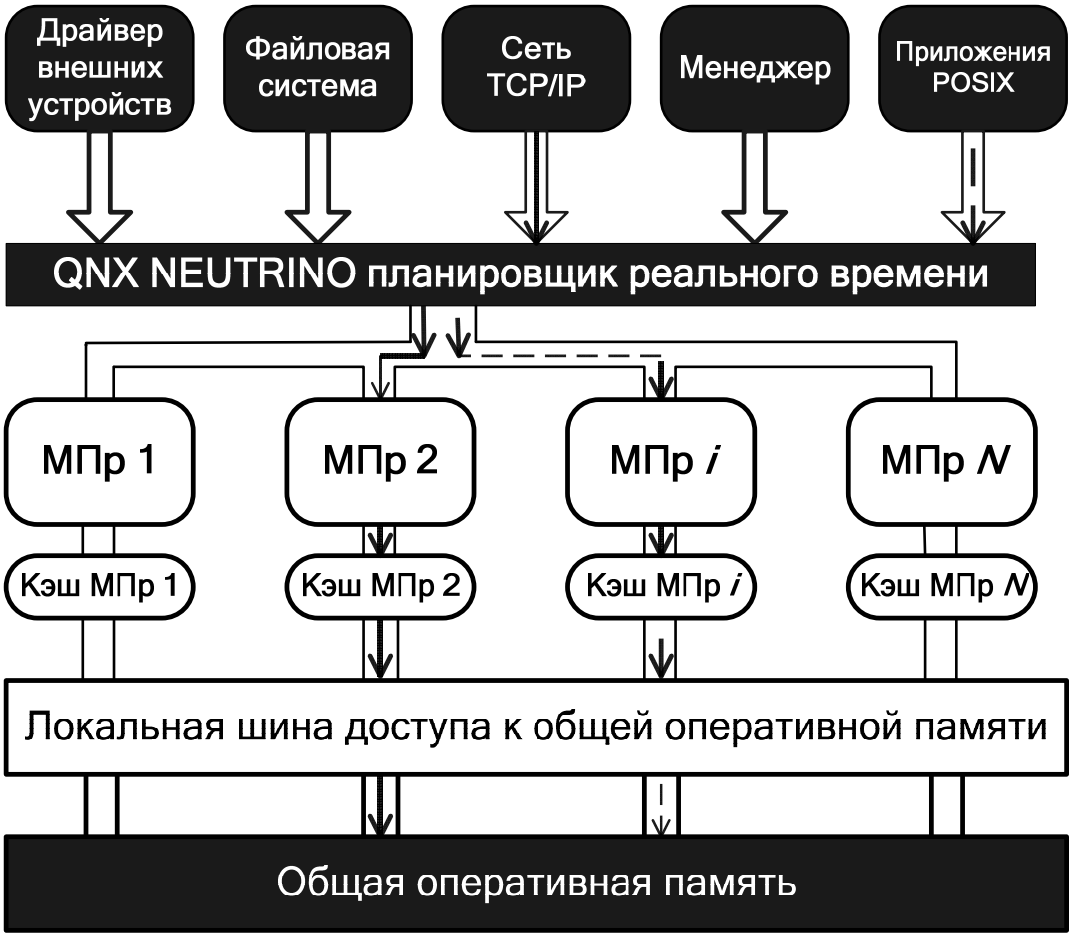


Рис. 2.8 – Поддержка симметричной микропроцессорной структуры ОС РВ QNX 6.2, 6.3 [Источник http://www.qnx.org]

Симметричная мультипроцессорная система SMP (symmetrical multiprocessing) состоит из двух или более тесно взаимодействующих МПр, которым доступны такие общие ресурсы, как оперативная память общие системные шины, локальные шины доступа к общей оперативной памяти. В SMP каждый МПр может самостоятельно выполнять многопоточную обработку данных, включая машинный код микроядра, ма-

шинный код приложений, обработку прерываний. При использовании SMP нет необходимости в том, чтобы каждый процессор был жёстко запрограммирован на выполнение одной задачи, например только на обработку приложения, драйвера или стека протоколов. При многопоточной обработке каждый процессор может выполнять обработку отдельного потока заданий, причём есть возможность переключаться ч выполнения одного потока заданий на другое.

Особенностью архитектуры SMP является наличие общей оперативной памяти, разделяемой всеми процессорами. Оперативная память является также средой для передачи сообщений между процессорами. Все МПр при обращении к оперативной памяти имеют равные права и одну и ту же адресацию для всех ячеек памяти. Это позволяет эффективно обмениваться данными с другими МПр. Вся система работает под управлением единой ОС или ОС РВ, которая в реальном времени автоматически распределяет процессы и потоки заданий по процессорам. В некоторых случаях возможна и явная привязка процессора к выполнению конкретного процесса.

Основным преимуществом SMP-систем является простота и универсальность, прежде всего для программирования. Здесь нет явных ограничений на модель программирования, используемую при создании приложения. Как правило, применяется модель параллельных потоков, когда все процессоры работают абсолютно независимо друг от друга. Допускается реализация модели, использующей межпроцессорный обмен. Использование общей памяти увеличивает скорость такого обмена, причём пользователь имеет доступ ко всему объему оперативной памяти. Для SMP-систем существуют сравнительно эффективные средства автоматического распараллеливания выполнения вычислений. Для ускорения доступа к общей памяти используется кэш-память процессоров.

Недостатком систем с SMP является ограниченная производительность, связанная в первую очередь с использованием локальной шины доступа к общей памяти. Как правило, эта шина имеет ограниченную (хотя и высокую) пропускную способность. Это затрудняет увеличение производительности управляющего комплекса при увеличении числа процессоров и количества подключаемых пользователей. Для решения этой задачи либо увеличивают разрядность шины (до 256 бит), либо применяют коммутатор (матричный коммутатор данных) для неблокирующего подключения многих МПр к общей физической памяти. Последнее решение соответствует шине PCI-Express. Коммутатор представляет собой специальную электронную схему, соединяющую любой вход с любым выходом, в результате любое устройство может быть неблокирующим образом подключено к оперативной памяти. Получила также распространение архитектура с распределенной разделяемой памятью, когда часть модулей физической памяти закреплена за группой процессоров.

Следует отметить, что SMP-система, как правило, может использовать, только единственный тип процессоров, а программное обеспечение, разработанное для однопроцессорной управляющей системы может не запуститься при переходе к SMP–системе.

Ещё одной проблемой SMP является обеспечение когерентности данных. Это означает, что в данный момент времени во всей системе SMP для любого элемента данных, например бита состояния порта ввода-вывода, существует только одно значение. Иными словами, невозможно, чтобы для одного МПр порт был в состоянии «0» а для другого – уже в состоянии «1». Это особенно важно соблюдать при организации процедуры обслуживания вызовов. Для решения данной проблемы используется специальная дополнительная шина слежения, которая объединяет кэш-память всех МПр. По шине слежения МПр отслеживают действия друг друга на предмет влияния на собственную кэш-

память. Если один из процессоров, например МПр 2 на рис. 2.8, обращается к данным, которые обрабатывает в кэш-памяти другой МПр *i*, то МПр *i* перехватывает по шине слежения запрос МПр 2, далее передаёт в сторону МПр 2 содержимое своей кэш-памяти. Как только МПр *i* записывает в общую память результат своей работы, то по шине слежения этот момент отслеживают другие МПр и, соответственно, обновляют содержимое своей кэш-памяти.

В мультипроцессорных структурах возможны и другие режимы функционирования ОС РВ. В частности, может применяться асимметричная процессорная обработка (asymmetric multiprocessing, AMP), когда на каждом ядре процессора работает отдельная операционная система. Ещё одним вариантом является исключительная многопроцессорная обработка, при которой единая операционная система одновременно управляет всеми ядрами, но прикладное программное обеспечение «закреплено» за отдельным ядром. Например, программы обработки вызовов и система административных программ, могут быть закреплены за отдельными ядрами операционных систем или за отдельными процессорами. Этот вопрос будет рассмотрен на примере коммутационной системы EWSD в главе 3.

2.6 Применение ОС РВ в системе управления сетями связи

Рассмотрим в качестве примера применение ОС РВ для управления сетями связи. Для этого рассмотрим особенности применения операционного ядра реального времени для мультипроцессорных систем RTEMS (Real-Time Executive for Multiprocessor Systems). Это некоммерческая ОС РВ, ориентированная на использование в относительно небольших и средних встраиваемых системах управления, контроля и диагностики. Система ОС РВ RTEMS реализована на языке программирования Си, разработчиком является компания OAR (On-line Applications Research Corporation), США. Система была создана по заказу мини-

стерства обороны США для использования в системах управления ракетными комплексами. На данную ОС РВ отсутствуют какие-либо экспортные ограничения, она свободно распространяется в исходных кодах через Интернет (www.oarcorp.com, www.rtems.com). Компания OAR обеспечивает поддержку, обучение специалистов и разработку программного обеспечения по заказам.

ОС РВ RTEMS находится под защитой модифицированной версии публичной лицензии GNU (GNU General Public License). В качестве стека TCP/IP в RTEMS используется версия стандартного стека сетевых протоколов FreeBSD, которая не накладывает каких-либо ограничений на используемые программные приложения.

Базовой средой разработчика при работе с RTEMS является ОС Linux (UNIX). Возможна настройка рабочего места для использования Windows 9x, Windows NT, Windows 2000. ОС РВ RTEMS обеспечивает высокоэффективную среду исполнения для «глубоко встраиваемых» систем (deeply embedded segment), в которых не предполагается частая перенастройка или смена алгоритмов функционирования. Это могут быть приложения для портативных устройств (например, сотовые телефоны), в которых используются аппаратные средства с ограниченными ресурсами (микроконтроллеры). Соответствуя специфике разработки и применения систем данного класса, RTEMS отличается модульностью, масштабируемостью и предсказуемостью поведения, а также поддерживает мультипроцессорные конфигурации.

ОС РВ RTEMS соответствует стандартам POSIX 1003.1b. ОС РВ RTEMS поддерживает стандартный стек сетевых протоколов TCP/IP, который основан на реализации аналогичного стека операционной системы FreeBSD, который включает в себя следующие протоколы: UDP, TCP, FTP, HTTP.

Операционная система RTEMS обеспечивает возможность использования большого набора средств и методов отладки. Средства

отладки свободно распространяются и могут быть получены с сайта www.gnu.org. ОС RTEMS написана на языке высокого уровня, поэтому ее перенос на различные процессорные платформы теоретически производится с минимальными трудозатратами.

Характерные особенности микроядра RTEMS следующие:

- поддержка различных мультипроцессорных систем;
- распределение машинного времени на основе управления событиями в соответствии с приоритетами (динамический алгоритм диспетчеризации);
- наличие менеджера задач, который позволяет изменять минимальный размер кванта времени, выделенного данной задаче;
- управление прерываниями гарантирует своевременное обнаружение запросов внешних устройств и оперативный вызов программ обработки. При этом гарантируется максимальное время вызова анализатора причины прерываний;
- имеется несколько механизмов взаимодействия и синхронизации задач;
- используется динамическое выделение памяти для приоритетных задач.

Микроядро реального времени RTEMS поддерживает 255 уровней приоритетов. Чем больше значение приоритета задачи, тем более привилегированной она является. Количество задач, имеющих одинаковый приоритет, не ограничено. Каждая задача всегда имеет какой-либо уровень приоритета, начальное значение которого присваивается при создании задачи и в дальнейшем может быть динамически изменено.

Операционная система RTEMS применяется в рамках системы управления сетевыми IP–коммутаторами. Для реализации такой системы управления используются коммуникационные контроллеры (сетевые процессоры) MC68EN360 компании Motorola.

Разработанное программное обеспечение управления, функционирующее на основе RTEMS, представляет собой SNMP-агента, который реализует следующие функции:

- поддерживает управляющие базы SNMP (MIB II), Ethernet MIB;
- обеспечивает обработку запросов SNMP-менеджера;
- производит трансляцию команд SNMP и поддерживает обмен данными между консолью управления на базе персонального компьютера и коммуникационным модулем с использованием интерфейса RS-232;
- обеспечивает прохождение команд и ответов с консоли на коммуникационный модуль и обратно;
- хранит необходимые данные (пароли, адреса менеджеров и т.п.) в энергонезависимой памяти ПЗУ.

Управление осуществляется с использованием протокола SNMP, который обеспечивает возможность управления объектами с помощью стандартных средств мониторинга и контроля [7].

Программное обеспечение SNMP-агента (рис. 2.9) представляет собой набор задач и служб, каждая из которых выполняется независимо и одновременно с остальными задачами.

В этот набор входят:

Init – главная программная задача, которая производит инициализацию коммуникационного модуля и запускает остальные задачи;

TimerSrv – служба таймера, посылает периодические запросы и совершает другие действия, связанные с временем;

SNMPSrv – осуществляет функции протокола SNMP. Данная служба реализуется на основе сетевого протокола UDP;

ConsoleSrv – служба работы с консолью для доступа к программному обеспечению IP-маршрутизатора;

TerminalSrv – служба работы с терминальным оборудованием маршрутизатора.

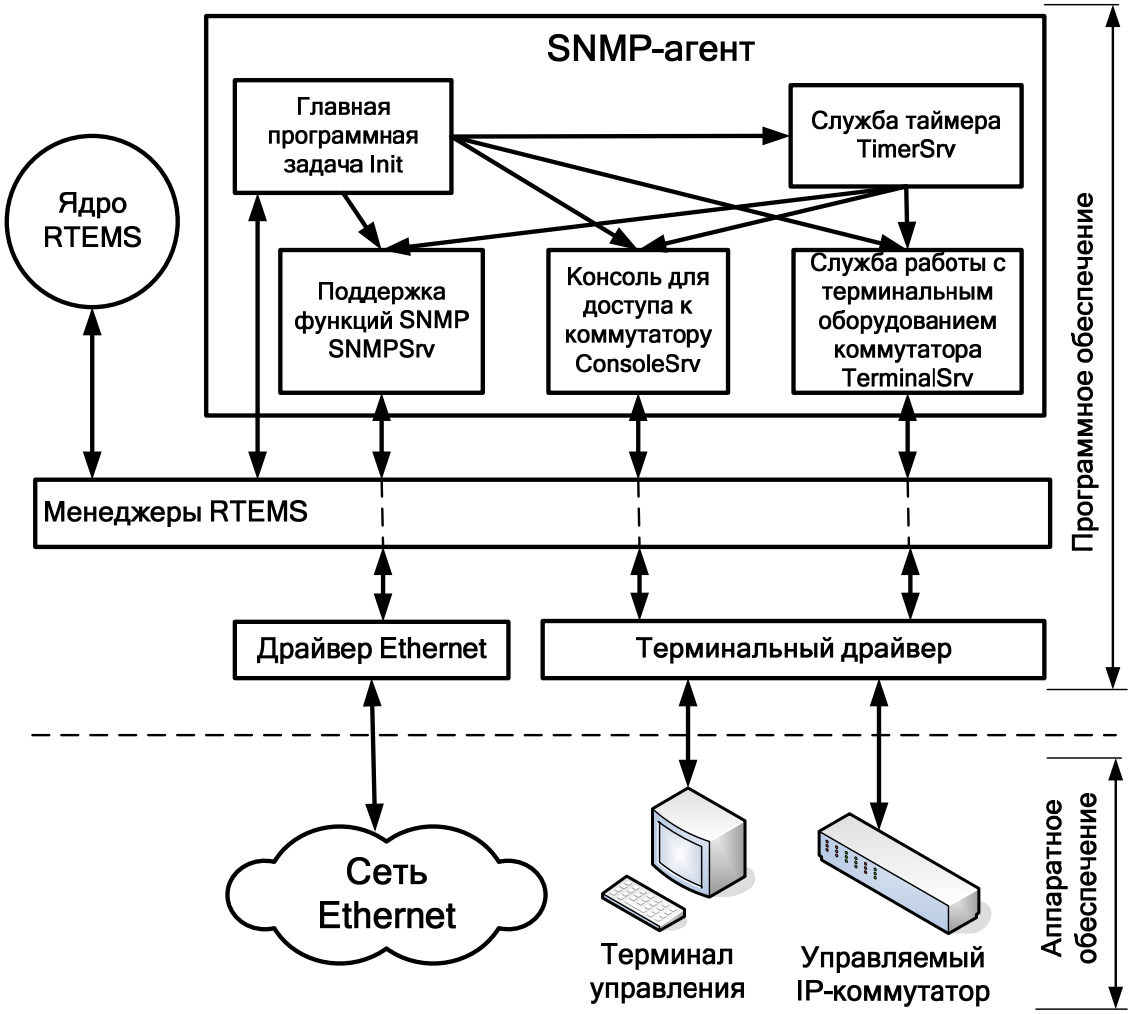


Рис. 2.9 – Реализация агента SNMP с помощью RTEMS

В качестве аппаратного обеспечения в системе управления использован отдельный коммуникационный модуль, который включает 32-разрядный микроконтроллер MC68EN360, оперативную память, блок последовательного обмена данными, разъем расширения и порт отладки. Этот модуль имеет также два выхода по стандарту RS-232, к одному из которых подключается IP–коммутатор, а к другому консоль управления на базе персонального компьютера. Для обмена с внешними устройствами используется Ethernet с помощью которого передаются сообщения и команды SNMP для системы управления сетью связи NMS (network management system).

Выбор операционной системы для описанной реализации агента и менеджера протокола SNMP производился в соответствии со следующими критериями: компактность кода, надежность функционирования, наличие эффективных средств работы с внешними устройствами, а также возможность полного контроля над программным продуктом, что предполагало необходимость применения систем с открытыми исходными текстами. Анализ операционных систем, доступных в исходных кодах, показал, что RTEMS удовлетворяет поставленным условиям. Кроме того, RTEMS имеет встроенный стек TCP/IP, что значительно сокращает время разработки и отладки программного обеспечения.

В рассмотренной ОС РВ используются следующие основные менеджеры RTEMS (понятие «менеджер» в данном случае соответствует менеджеру ОС РВ QNX, а не менеджеру протокола SNMP) :

Менеджер ввода/вывода обеспечивает работу драйверов внешних устройств, не накладывая ограничений на внутреннюю структуру внешних устройств. ***Менеджер доступа к физической памяти*** включает менеджера разделов и регионов.

Раздел – это область памяти, состоящая из буферов (разделов) фиксированной длины. Каждый из этих буферов может быть выделен для использования задачей или процессом с помощью команд менеджера разделов. При запросе на выделение буфера он выделяется из начала последовательности свободных буферов. Когда буфер освобождается, то он циклически помещается в конец последовательности.

Регион (в данном случае) – область памяти переменной длины, кратной размеру физического сегмента. Регион состоит из сегментов различного размера. При поступлении запроса на выделение сегмента, размер запрошенного сегмента округляется до целого количества физических страниц и при наличии свободного сегмента соответствующего размера этот сегмент выделяется операционной системой под требуемую задачу или процесс.

Менеджер доступа к памяти реализует следующий набор функций: создание, удаление, установка значений переменных; освобождение, занятие областей регионов/разделов и буферов, содержащихся в них. Для регионов реализуется возможность добавления необходимого объёма памяти.

Менеджер таймеров обеспечивает следующие функции работы с таймерами: создание и удаление таймеров, доступ к таймерам, запуск подпрограмм по событию/сигналу от таймера.

Менеджер часов реального времени применяется для информирования пользователя о текущей дате. Этот менеджер обеспечивает также формирование и обработку сигналов об истечении минимальных промежутков времени, которые задаются на этапе конфигурирования системы и равны целому числу микросекунд.

Менеджер инициализации отвечает за запуск и остановку работы ОС РВ RTEMS. Запуск ОС РВ RTEMS производится путем создания и запуска всех инициализирующих задач и инициализирующих процедур для каждого драйвера внешнего устройства. В случае мультипроцессорной системы происходит также инициализация механизмов межпроцессорного взаимодействия.

Менеджер прерываний позволяет оперативно реагировать на прерывания, обеспечивая возможность «вытеснения» (временного останова или завершения) задачи сразу после выхода из процедуры обработки прерывания. Менеджер прерываний дает также возможность внешней программе пользователя подключить процедуру обработки прерывания к соответствующему вектору прерывания. При выполнении определенных команд ОС РВ RTEMS может возникнуть необходимость отключения обработки прерываний, чтобы обеспечить непрерывное выполнение критических задач. Максимальное время отключения прерываний различно для разных процессоров и указывается в документации ОС РВ RTEMS для соответствующего процессора.

Достоинством RTEMS является возможность ее конфигурирования с учетом реальных требований программного приложения. Например, для увеличения надежности и повышения компактности результирующего программного кода в системе управления IP-коммутатором были исключены все неиспользуемые менеджеры и библиотеки, из 16 менеджеров оставлены только 10 (исключены менеджеры двухпортовой памяти, разделов и регионов, мультипроцессорности, сообщений и событий). В результате суммарный объем загружаемого в оперативную память программного кода вместе с приложением, стеком сетевых протоколов и областями данных составил около 270 Кбайт. Система имела возможность использовать всю оперативную память, имеющуюся на модуле. В итоге, полученный программный продукт для поддержки управления IP-маршрутизатором содержит пять независимых пользовательских процессов и набор необходимых менеджеров ОС РВ. Блокировка одного из процессов не приводит к остановке остальных, что повышает надёжность системы управления на основе ОС РВ.

2.7 Контрольные вопросы к главе 2

1. Для решения каких задач применяется внутреннее программное обеспечение?
2. Почему программа обработки вызовов работает в реальном времени?
3. Какие функции выполняет ядро (kernel) операционной системы?
4. Чем отличаются уровни инициализации управляющих устройств при восстановлении работоспособности средств связи?
5. Для решения каких задач применяются операционные системы «жёсткого реального времени»?
6. Как осуществляется обмен сообщениями между внешними процессами ОС РВ по отношению к ядру ОС?

7. За счёт чего достигает относительно малый размер (в байтах) ядра ОС РВ?

8. Какова функция менеджера процессов?

9. Что такое драйвер внешних устройств?

10. Что такое «симметричная многопроцессорная структура», для чего она используется?

11. Для чего используется система прерываний?

12. Что такое операция «ввод–вывод»?

13. Для чего используется регистр выходных данных?

14. Можно ли регистр выходных данных объединить с регистром входных данных?

15. Какие существуют режимы ввода–вывода ?

16. Для чего применяется режим прямого доступа к памяти DMA?

3. Управляющий комплекс АТСЭ EWSD

3.1 Функции и структура управляющего комплекса EWSD

Рассмотрим в качестве примера использования микропроцессорных и программных систем в средствах связи многопроцессорный управляющий комплекс цифровой системы коммутации EWSD производства компании Siemens AG, Германия. Учитывая, что ранее система EWSD описывалась в литературе [27,41], рассмотрим более детально только центральное управляющее устройство EWSD – компактный координационный процессор CP113c (coordination processor 113, compact). Координационный процессор CP113c предназначен для выполнения следующих функций управления системой коммутации:

1. Обработка вызова, в том числе трансляция цифр набора номера, маршрутизация вызова, определение зоны обслуживания, учет продолжительности соединения, учет данных об оконечном и транзитном трафике.
2. Функции технической эксплуатации, в том числе организация обмена с внешним ЗУ и связь с системой управления сетью, в первую очередь с системой Net Manager производства Siemens.
3. Обеспечение надежности – самоконтроль и самодиагностика технического состояния, обнаружение ошибок обработка ошибок.

Функциональные блоки многопроцессорной системы CP113 C/CR и их взаимосвязи представлены на рис. 3.1. Следует отметить, что каждый блок, называемый «процессором» является совокупностью цифровых вычислительных устройств, объединенных в микросхемный набор (chipset).

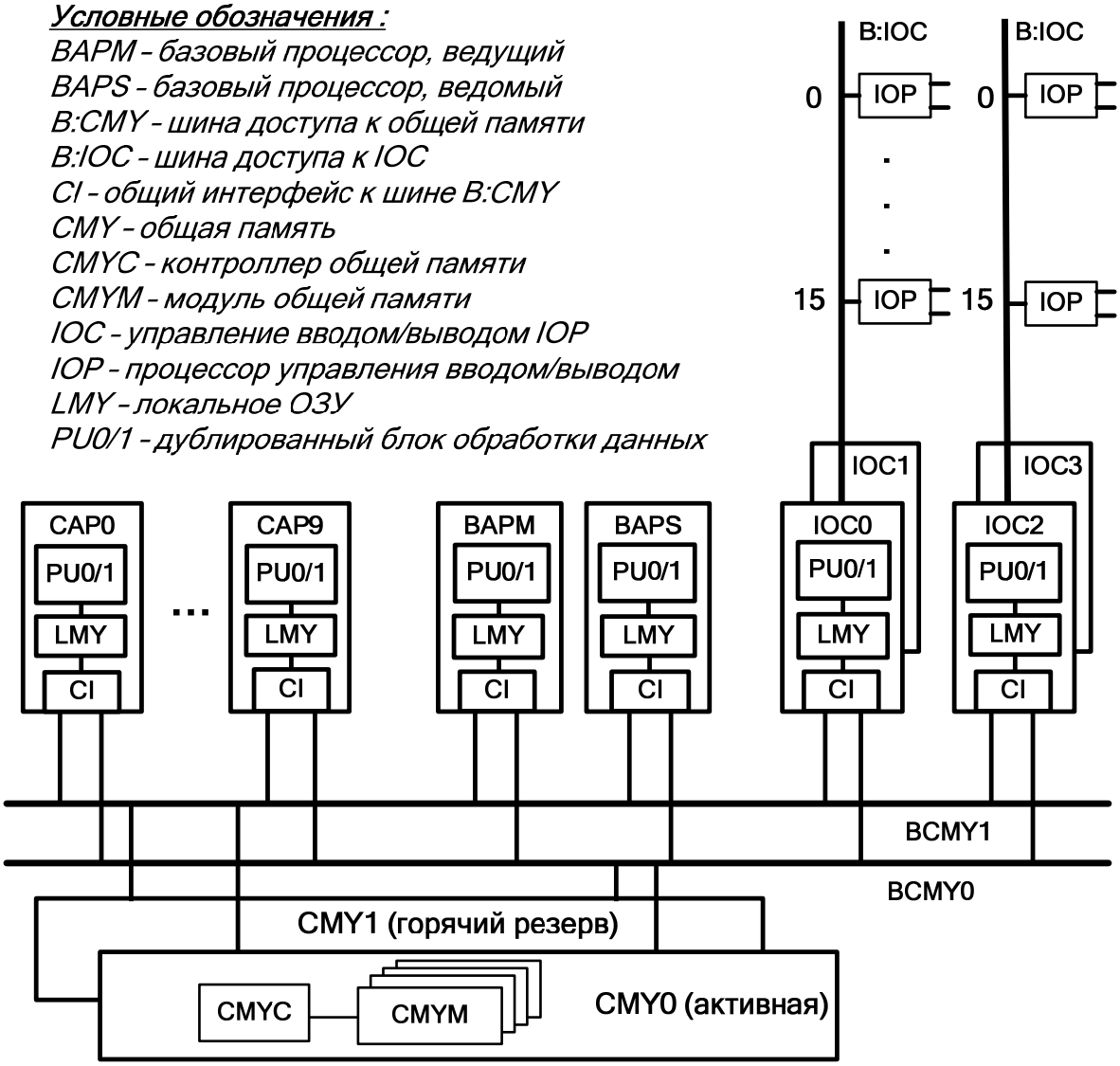


Рис. 3.1 – Функциональная схема процессора CP113C (EWSD v10) [32]

Рассмотрим отдельные функциональные блоки в составе CP113c на рис. 3.1. ***Базовый (ведущий) процессор, BAPM*** (base processor master) в нормальном режиме работы обрабатывает вызовы, автоматизирует функции эксплуатации и осуществляют функции обеспечения надежности и администрирования. ***Ведомый процессор BAPS*** (base processor slave) в штатном режиме работы только обрабатывает вызовы. Процессоры BAPM и BAPS работают в режиме разделения нагрузки – каждый процессор обрабатывает 50% поступающих запросов, процессов и задач. Загрузка процессоров распределяется таким образом, чтобы в данный момент времени она составляла не более 80% от мак-

симальной вычислительной мощности. Уровень максимальной допустимой нагрузки составляет 95%, примерно 5% от мощности процессора остаются незагруженной даже в режиме перегрузки. Это необходимо для запуска аварийных программ восстановления. В случае выхода из строя процессора BAPM его функции безобрывно начинает выполнять BAPS. Установка признака «ведущий – ведомый» осуществляется программно и может быть мгновенно изменена в случае ремонта одного из BAP или при перезагрузке программного обеспечения управления. На процессорах BAP имеется ПЗУ типа EPROM ёмкостью 4 Мбайта.

Процессоры обработки вызовов CAP (call access processor) осуществляют функции обработки поступающих вызовов. К минимальной конфигурации из двух BAP и двух IOC процессоры CAP добавляются с учетом роста номерной емкости АТС.

Процессор управления вводом-выводом IOC (input/output control) – управляет доступом периферийных устройств (DLU, LTG, MB, SN, OMT, MDD) к шине доступа к общей памяти CP113. К одному IOC подключается до 12 процессоров IOP. ***Процессор ввода/вывода IOP*** (input/output processor) подключает периферийное оборудование – НЖМД, терминал технического обслуживания и эксплуатации, дисковод – к IOC. Шина B:IOC является мультиплексированной, 32-х разрядной, предназначена для подключения процессоров IOP к IOC.

Шина B:CMY (bus to common memory) предназначена для обмена данными между CAP, BAP, IOC, CMY. Шина доступа к общей памяти B:CMY имеет скорость передачи 32 Мбайт/сек, тактовая частота работы шины составляет 16 МГц. Шина использует механизм временного мультиплексирования, информация передаётся и принимается в четырех канальных временных интервалах, причём каждый временной интервал позволяет обращаться к одному из четырёх банков общей оперативной памяти. Линии шины распределены следующим образом :

- для передачи адресов выделено 32 разряда;

- для передачи проверочных бит адресов ECC (error correction code) выделено 8 разрядов;
- 2 разряда – биты занятия для адресации одного из четырех банков памяти
- для передачи данных выделено 32 разряда;
- для передачи проверочных бит данных ECC выделено 8 разрядов.

На шине имеется 16 портов для подключения к BAP, CAP, IOC и два порта для подключения к общей памяти CMY.

Общая память CMY (common memory) является общей оперативной физической памятью, предназначенной для хранения коммутационных программ, административных программ, программ технической эксплуатации, базы станционных данных, базы абонентских данных, данных по межстанционной соединительно сети, статистических учетных данных по маршрутизации, используемых всеми процессорами в составе СР113. Емкость CMY составляет от 64 Мбайт до 512 Мбайта. Память CMY конструктивно состоит из четырёх физических банков памяти, нумеруемых от 0 до 3; ёмкость каждого банка составляет от 16 Мбайт до 64 (128) Мбайт. В каждом банке находятся ячейки с уникальными (неповторяющимися) физическими адресами. Общая память CMY работает с тактовой частотой 16 МГц, подключена к двум портам ввода-ввода для шины общей памяти.

В целях обеспечения надежности, общая память CMY дублирована, при этом содержимое 0 ветви памяти (CMY0) должно быть до бита идентично содержимому памяти ветви 1 (CMY1). С этой целью производится регулярная синхронизация физического содержимого ветви памяти, являющейся активной, с содержимым ветви памяти, находящейся в «горячем» резерве (stand by). Эта процедура обеспечивает постоянную оперативную готовность, надёжность и безотказность программного обеспечения управления.

В качестве МПР в процессоре СР113с используется семейство МПр типа Motorola MC68020, MC68040. Эти МПр имеют RISC архитектуру т.е. используют короткие инструкции (команды). В результате за один такт может быть выполнено 1…4 инструкции. Основной микропроцессор MC68040 работает с тактовой частотой 25 МГц; разрядность данных 32 бита; разрядность адреса 32 бита; максимальная тактовая частота составляет 66 МГц; размер кэш-памяти второго уровня L.2 составляет до 8 Кбайт. Данный МПр преимущественно выполняет операции фиксированной точкой, что позволяет при невысокой тактовой частоте добиваться требуемой производительности управляющего комплекса. В процессорах IOP может использоваться МПр MC 68040 с тактовой частотой 16 МГц, разрядность данных IOP составляет 32 разряда. Процессор IOP имеет локальное ОЗУ ёмкостью 128 Кбайт. Напряжение питания перечисленных МПр составляет 5 В, производительность составляет до 1 170 000 операций/сек.

Рассмотрим детальнее характеристики семейства МПр MC 68XXX используемых в координационном процессоре СР113с (v.10).

3.2 Особенности микропроцессоров Motorola MC68XX

Микропроцессор MC68020 [3] – первый полностью 32-разрядный процессор третьего поколения, относящийся к семейству MC68XXX фирмы Motorola, применяется в процессоре IOP. В MC68020 введены новые режимы адресации для поддержки языков высокого уровня и новые инструкции. MC68020 имеет внутренний кэш инструкций размером 256 байт. Для ускорения взаимодействия с сопроцессорами MC68020 поддерживает сопроцессорный интерфейс. Микропроцессор MC68EC020 является разновидностью MC68020 для встроенных приложений и программно совместим с MC68020. Имеет уменьшенную, по сравнению с базовым процессором, шину адреса (A0…A23).

Основные характеристики MC68020, MC68EC020:

- процессор MC68020 полностью 32-х битный (32-х битная шина адреса и 32-х битная шина данных, шины не мультиплексированы), ёмкость адресуемого пространства физической памяти 4 Гбайт;
- имеет шестнадцать 32-битных регистров общего назначения и пять управляющих регистров специального назначения;
- поддержка двухуровневой системы защиты информации;
- совместимость по объектным кодам с предыдущими микропроцессорами семейства MC68XXX;
- конвейерная архитектура с высоким уровнем параллелизма;
- внутренний кэш инструкций;
- интерфейс для сопроцессоров;
- поддержка виртуальной памяти и виртуальной машины;
- расширенные режимы адресации для поддержки языков высокого уровня (18 режимов адресации);
- поддержка шести основных типов данных – биты, битовые поля, байты (длина 8 бит), слово (длина 16 бит), длинное слово (длина 32 бита), двоично-десятичные числа (4 бита).

Микропроцессор MC68040 – 32-разрядный процессор, применяется в составе процессоре BAPM/BAPS, CAP, IOP. Микропроцессор MC68040 включает следующие основные компоненты:

- устройство обработки целочисленных данных (integer unit, IU), совместимого с целочисленным устройством МПр MC68030;
- устройство обработки данных с плавающей точкой (floating point unit, FPU);
- внутреннее устройство управления памятью (memory management unit, MMU).
- устройство памяти инструкций/команд (integer memory unit, IMU);
- устройство памяти данных (data memory unit, DMU);

- контроллер внутрипроцессорной магистрали (шины).

Устройство обработки целочисленных данных IU обеспечивает выработку исполнительного (виртуального или физического) адреса памяти, обработку инструкций и, при необходимости, передачу инструкции на исполнение в FPU и получение из него результата, передачу результата обработки инструкции в блок памяти данных и в контроллер магистрали.

Устройство обработки данных с плавающей точкой FPU осуществляет необходимое конвертирование формата информации, полученной из IU, обработку данных IU, обратное конвертирование формата результата обработки данных и передачу данных в IU.

Оба устройства памяти (IMU, DMU) идентичны по структуре. Они осуществляют трансляцию логических/виртуальных адресов, выработанных IU, (IMU – трансляцию адресов инструкций, а DMU – трансляцию адресов данных) в физические, имеют в своем составе: устройство MMU, 128-байтные кэш адресных трансляций и 4-х килобайтный кэш операндов.

Контроллер внутрипроцессорной магистрали поддерживает внешние циклы магистрали и обеспечивает передачу операндов между магистралью IMU, DMU, и IU по раздельным внутренним магистралям процессора. Микропроцессор MC68EC040 не имеет встроенных FPU и MMU, но в остальном функционально и конструктивно совместим с MC68040. Основные технические характеристики MC68040, MC68LC040, MC68EC040 следующие:

- совместимость «снизу-вверх» по объектным кодам с предыдущими процессорами семейства M68XXX;
- 32-разрядные немультиплексируемые шины адреса и данных;
- физическое адресное пространство, измеряемое 4 Гбайт;

- двухуровневая система защиты информации, поддерживаемая возможностью работы в режиме пользователя и в режиме супервизора;
- устройство обработки целочисленных данных, совместимое с устройством обработки MC68030;
- устройство обработки данных с плавающей точкой FPU (только MC68040);
- два независимых устройства памяти инструкций и памяти данных (IMU, DMU) для МПр MC68040;
- поддержка виртуальной памяти и виртуальной машины;
- конвейерная архитектура с высоким уровнем распараллеливания операций;
- поддержка многопроцессорных систем;
- шестнадцать 32-битных регистров общего назначения;
- два указателя стека (главный и указатель прерываний) и десять управляющих регистров специального назначения в режиме супервизора;
- внутренние кэши инструкций и данных по 4 Кбайт каждый;
- 18 режимов адресации и 7 типов данных.

Перечисленные особенности МПр позволяют обеспечить выполнение МПр операций, соответствующих обработке данных на уровнях 1…3 модели взаимосвязи открытых систем. Рассмотрим далее функциональные блоки СР113с более подробно.

3.3 Архитектура, комплексирование и способы связи координационного процессора СР113с

3.3.1 Функциональные блоки ВАР и САР

В основе процессоров ВАР, САР или IOC [31] лежит общий функциональный компонент – подсистема выполнения программ (program

executor, PEX). Аппаратно PEX (см. рис. 3.2) выполнен в виде отдельной монтируемой платы (физического модуля), который имеет собственную физическую позицию на стативе.

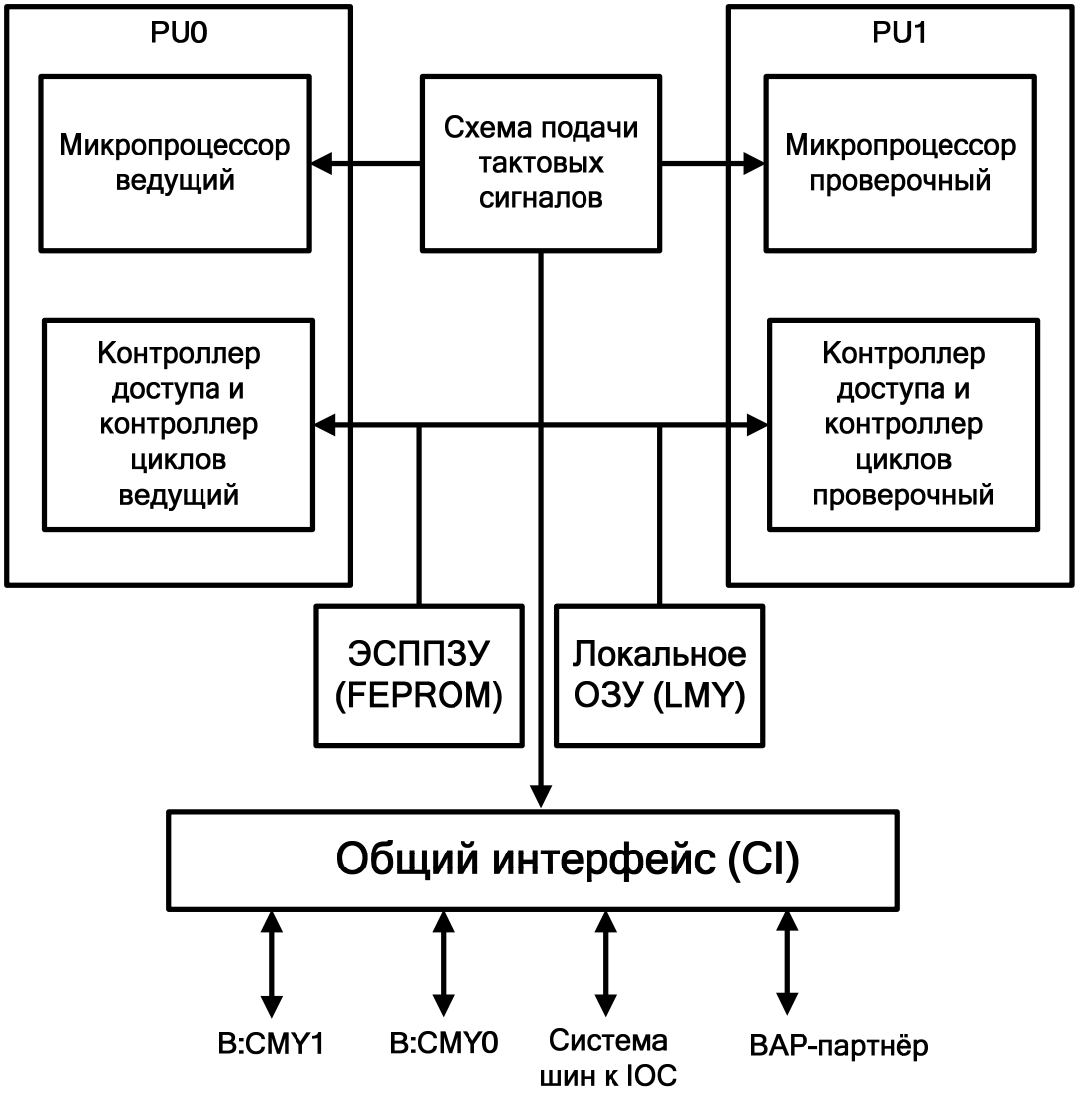


Рис. 3.2 – Функциональная блок-схема модуля PEX [32]

В зависимости от того, в качестве какого устройства используется модуль PEX, т.е. BAP, CAP или IOC и в соответствии с монтажной позицией на стативе, активизируются требуемые функции аппаратных средств в составе процессора.

Процессор блока PU включает МПр MC68040, контроллер доступа, контроллер циклов. Контроллер доступа и контроллер циклов выполняют следующие функции:

- обмен информацией или данными с локальным ОЗУ;
- обмен информации или данными с общим интерфейсом CI;
- постоянное сравнение результатов операции PU0 и PU1.

Контролер доступа и контролер циклов разделяются на ведущий и проверочный. Они работают синхронно и обеспечивают сравнение результатов работы микропроцессоров ведущего блока PU0 и ведомого блока PU1 по принципу «такт-в-такт». В случае, если результаты обработки данных, выполненных PU0 и PU1 не совпадают, то генерируется аварийный сигнал и соответствующий процессор (CAP, BAP или IOC) отключается от шины B:CMY.

Схема подачи тактового сигнала формирует тактовые импульсы с частотой 25 (16) МГц. Локальное ОЗУ (local memory, LMY) имеет емкость от 32 до 64 Мбайт, и строится на базе микросхем динамической памяти с произвольным доступом (dynamic access memory, DRAM). В оперативной памяти LMY хранится информация, необходимая для текущего функционирования данного блока PU, в частности промежуточные результаты вычислительных операций, файлы операционной системы, программа обработки вызова.

Быстро стираемое электрически программируемое постоянное запоминающее устройство с возможностью чтения FEPROM (flash erasable programmable read only memory) хранит программы начальной загрузки PEX и программы диагностики аппаратной части PEX, что аналогично функциям BIOS персонального компьютера.

Общий интерфейс CI (common interface) используется для организации обмена данных с шинами B:CMY, а так же обмена с IOC. Процессор PEX для нормальной работы требует постоянного отвода тепла, поэтому статив с PEX оборудован системой принудительной вентиляции.

Как уже отмечалось, по соображениям надежности хранения данных, общая память процессора CP113 разделена на четыре банка. Ем-

кость каждого банка составляет 64, 128, 256 Мбайт. Рассмотрим подробнее работу CP113 в части записи и считывания данных в общую память CMY.

3.3.2 Функциональные блоки, управляющие обменом с CMY

Шина доступа к общей памяти B:CMY передаёт адреса и данные в CMY в режиме дуплекс с мультиплексированием (разделением) по времени. Для доступа к каждому из четырёх банков физической памяти выделяется отдельный канальный временной интервал продолжительностью 125 нсек, общая длина цикла доступа к общей оперативной памяти составляет 500 нсек. В течении цикла доступа к памяти доступны все четыре банка; такой доступ можно охарактеризовать как детерминированный синхронный доступ с временным разделением или квазимгновенный доступ.

Запись слова данных в память осуществляется только с шины B:CMY, которая в данный момент времени является активной; «активность» шины устанавливается средствами системного программного обеспечения EWSD. При этом слово данных для записи одновременно появляется на обеих ветвях шины BCMY0 и BCMY1.

Процесс обработки данных при записи в физическую память осуществляется синхронно следующим образом :

- контроллер памяти 0 отвечает за обработку записываемого слова данных с 0 бита по 15-й бит и за биты ECC с 4 по 7-й;
- контроллер памяти 1 отвечает за обработку записываемого слова данных с 16 бита по 31-й бит и за биты ECC с 0 по 3-й (см. схему общего вида на рис. 3.3)

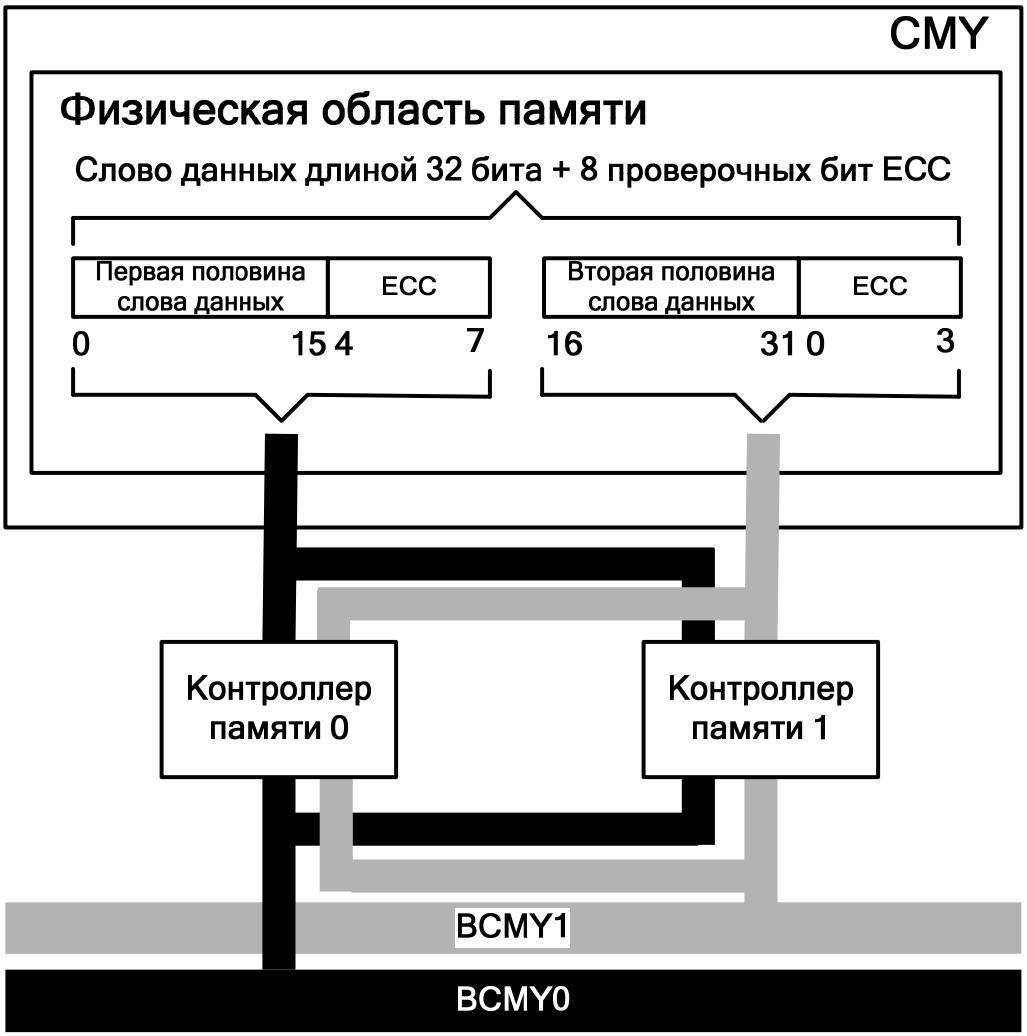


Рис. 3.3 – Запись и считывание данных из памяти CMY [32]

При этом оба контроллера памяти работают независимо. При чтении данных из общей памяти контроллер памяти 0 и контроллер памяти 1 производят объединение слова данных и бит ECC; в результате полное слово данных передаётся одновременно как на шину B:CMY0 так и на шину BCMY1.

Контроллер памяти, представленный на рис. 3.3, имеет сложную конструкцию и состоит из двух аппаратных модулей: контроллер общей памяти CMYC (common memory control) и модуль физической среды запоминания общей памяти CMYM (common memory medium). Модуль CMYC управляет собственно обменом данными, модуль CMYM управляет размещением данных непосредственно в физической среде хранения. Рассмотрим эти два модуля более подробно.

Управление обменом адресами и данными между B:CMY и CMY осуществляет ***модуль CMYC*** – контроллер общей памяти (см. рис. 3.4).

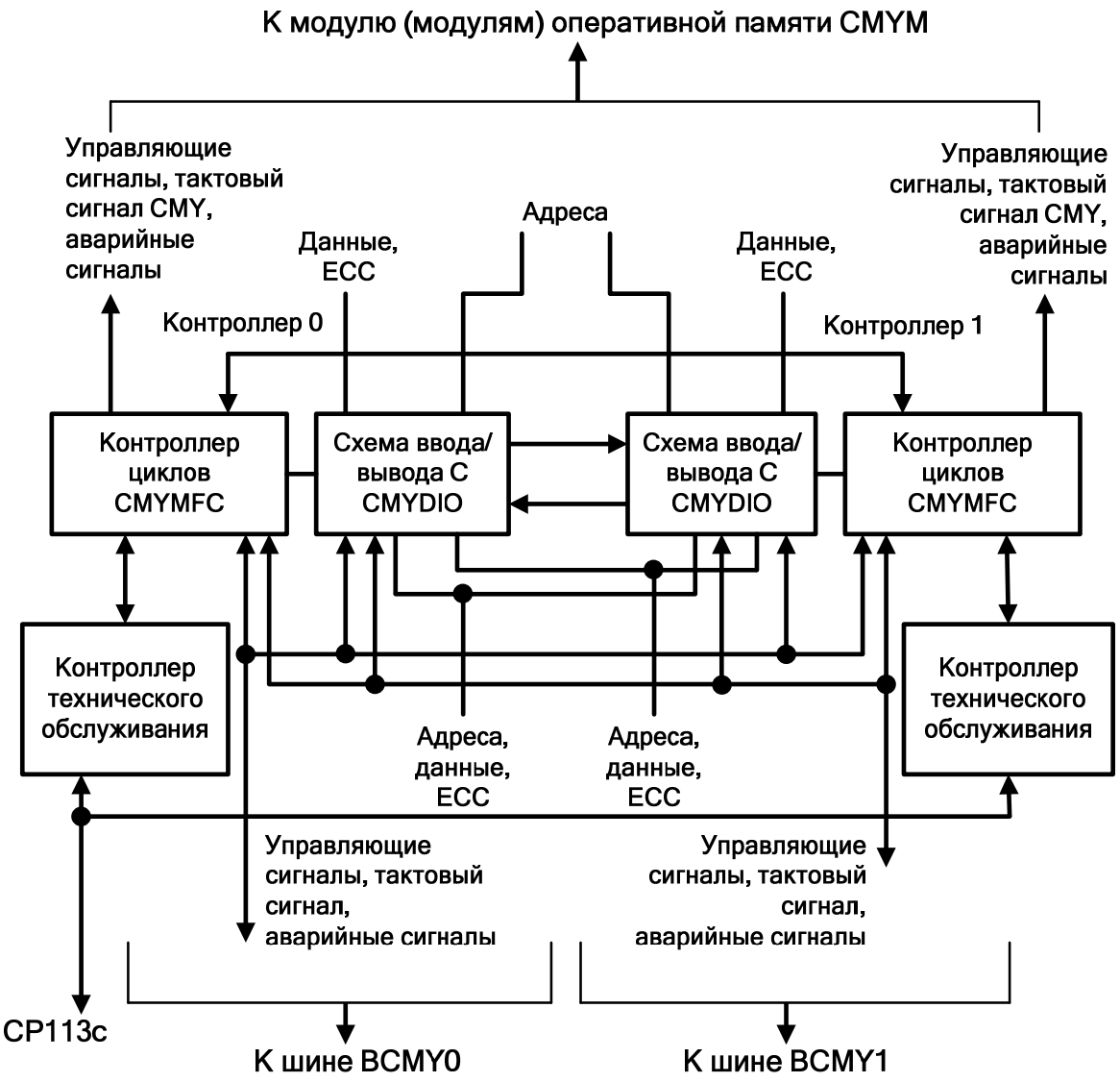


Рис. 3.4 – Функциональная схема модуля CMYC [32]

Модуль CMYC предназначен для управления обменом между шиной доступа к общей памяти и модулями общей памяти CMYM. Модуль CMYC выполнен в виде отдельного аппаратного модуля, который на стативе монтируется в фиксированной позиции. В состав CMYC входят следующие функциональные блоки :

Схема ввода/вывода по сети данных при доступе к общей памяти CMYDIO (common memory, data net an input/output stage) – реа-

лизована в виде заказной микросхемы, именуемой также специальной интегральной схемой, ориентированной на приложение ASIC (application-specific integrated circuit). Эта схема реализует следующие функции:

- прием из обеих шин B:CMY данных и адресов для записи в физическую оперативную память;
- передача в BCMY0 и BCMY1 данных, считанных из физической оперативной памяти;
- проверка корректности адресов и данных с помощью кода исправления ошибок ECC.

При записи с помощью кода ECC существует возможность исправить однобитовые ошибки; при чтении данные корректируются непосредственно в памяти CMY с помощью модуля CMYM. Многобитовые ошибки обнаруживаются, но не исправляются. В случае обнаружения ошибки, процедура записи в память или чтения из памяти повторяется до двух раз, а после этого, при наличии ошибки, запускается программа диагностики и восстановления. Ограниченное количество повторов обусловлено необходимостью функционирования системы коммутации в реальном масштабе времени.

Поддержка сети для передачи данных и поддержка передачи адресов, контроль ECC и управление выбором шин для записи/считывания данных также реализовано с помощью ASIC в целях обеспечения работы в реальном времени.

Контроллер циклов при обращении к общей памяти CMYMFC (common memory, maintenance facilities an cycle control) генерирует все внутренние управляющие сигналы для CMY, которые необходимы для поддержки синхронности циклов записи/считывания.

Контролер технического обслуживания позволяет выполнять анализ обнаруженных ошибок и выводить данные на панель технического обслуживания процессора CP113c.

Итак, в процессе ввода/вывода перечисленное аппаратное обеспечение выполняет следующие функции :

- организация сети для передачи данных и адресов;
- контроль коррекции ошибок (error correction code, ECC) в адресах и данных;
- управление выбором шин B:CMY.

Сеть для передачи данных и сеть для передачи адресов при обмене с общей памятью CMY распределена между модулями CMYC и модулем CMYM. Сеть для передачи данных и сеть для передачи адресов в составе модуля CMYC поддерживают следующие внутренние интерфейсы:

- через интерфейсы к шинам B:CMY передаются адреса и данные для записи и данные для чтения;
- через интерфейс к модулю CMYM передаются адреса и данные для записи в физическую область CMY или для чтения из физической памяти.

Модуль общей памяти CMYM является непосредственной физической средой для хранения данных и адресов. Модуль имеет сложную конструкцию и включает :

- сеть для передачи адресов, контроллер циклов, которые реализованы на одной микросхеме ASIC;
- сеть для передачи данных CMYDIO, которая с помощью специализированной микросхемы ASIC реализует возможность обмена данными между модулями CMYC и банками физической памяти CMY.

Отличием конструкции модуля CMYM является наличие в его составе банков памяти; в составе CMYC банки памяти отсутствуют.

Модуль CMYM имеет следующие встроенные функции по коррекции ошибок :

- проверка при чтении данных из памяти с помощью ECC бит;

- коррекция однобитовых ошибок при чтении данных и запись скорректированных данных обратно в физическую память;
- генерация ECC бит для записываемых слов данных;
- сравнение ECC бит, созданных контроллером памяти 0 и контроллером памяти 1, в целях мониторинга корректности обмена данными между модулями CMYC и CMYM.

Для организации работы с памятью в процессоре CP113c важна система прерываний. Рассмотрим эту систему подробнее.

3.3.3 Прерывания в процессоре CP 113

Система коммутации EWSD работает в режиме реального времени. Поэтому все запросы, посылаемые в CP113c, должны обрабатываться за минимальное время после их получения. С точки зрения процессора CP 113, поступающие запросы представляют собой задачи, подлежащие исполнению. Запросы могут инициироваться в самом процессоре CP113 или за его пределами. Примером ***внешнего запроса*** является реакция аппаратного и программного обеспечения на снятие абонентом микротелефонной трубки. Управляющие устройства – контроллер модуля абонентских линий (ИУУ), контроллер цифрового абонентского блока DLU (ГУУ), процессор GP блока LTG (ГУУ) – последовательно обнаруживают и обрабатывают запрос на установление соединения. В результате процессор GP посылает в CP113 сообщение для запуска коммутационной программы обработки поступающего вызовов. В процессе обработки вызова идентифицируется вызывающий абонент, происходит приём и обработка цифр набора номера согласно процедурам [27], инициируются следующие этапы процесса установления соединения. Внешние запросы также представляют собой аппаратные прерывания, инициируемые внешними аппаратными блоками (устройствами).

Примером ***внутреннего запроса*** является, например, прикладной процесс, инициирующий вывод данных на внешний носитель для целей сохранения и обеспечения надёжности. В результате операционная система выполняет задачу вывода данных. Внутренние запросы также представлены программными прерываниями, которые инициируются процессами с помощью вызовов супервизора/планировщика.

Проблема состоит в том, что в управляемой в реальном времени системе коммутации одновременно могут существовать несколько запросов (задач), подлежащих обработке. Последовательность и приоритет обработки задач определяется процессором СР 113 на основе уровней прерывания, задаваемых для отдельных задач. Здесь каждой задаче назначается уровень прерывания, указывающий относительную важность задачи (весовой коэффициент задачи) в рамках главной программы управления EWSD. В системе EWSD определено 8 уровней прерывания (пронумерованных от 0 до 7), причем уровень прерывания 7 является самым высоким, а 0 – самым низким. При поступлении новой задачи программа, выполняющаяся в текущий момент, может прерваться, а новая задача начинает обрабатываться, если её уровень прерывания выше чем у текущей задачи. Запрос на выполнение задачи процессором СР113с называется «запросом на прерывание». Процесс обработки прерываний происходит в три этапа, которые можно определить следующим образом:

1 этап. Запрос на прерывание – включает обнаружение новой ожидающей обработки задачи. Здесь процессор СР113с обнаруживает, что внутренний или внешний запрос на прерывание (и соответственно, новая задача) ожидает обработки. Процессор прерывает программу, выполняющуюся в текущий момент времени, выдает запрос на прерывание и запускает обработчик прерываний.

2 этап. Анализ прерывания – анализ ждущей обработки задачи и назначение уровня прерывания для обработки задач. Здесь обра-

ботчик прерываний принимает ожидающую обработки задачу и выполняет ее предварительный анализ для определения следующей информации:

- определение уровня прерывания, связанный с запросом на прерывание;
- характер рассматриваемой задачи;
- действия, которые необходимо инициировать;
- уровень прерывания, на котором будет происходить последующая обработка этой задачи, и программа, которая будет выполнять эту обработку (см. таблицу. 3.1).

Таблица 3.1 Таблица прерываний процессора СР113

Прерывания		**Функции программного и аппаратного обеспечения СР113**
Уровень	**Номер**	
7		Процедура перезагрузки
6	**15**	Тест шины B:CMY при начальной загрузке
	14	Обработка ошибки аппаратной части МПр.
	13	Периодические программные прерывания
5	**12**	Прерывания системы отладки ПО (зарезервировано разработчиками ПО)
4	**11**	Объединённый тест синхронизации компонентов
	10	Остановка системы для обработки ошибки ПО
	9	Обработка ошибки ПО центральной части СР113
3	**8**	Запрос IOC из BIOS
	7	Слежение за аппаратной частью
2	**x**	Обработка ошибок локального ПО
	x	Выполнение запросов управления
	6	Межпроцессорная связь на уровне операционной системы
	5	Сканирование генератора системного времени для контроля ввода/вывода периферийных устройств СР
1	**4**	Ошибка ввода/вывода
	3	Сообщения от IOP:UNI
	2	Сообщения от IOP:MB
	1	Управление вводом/выводом данных на физическом уровне
	0	Задание по техобслуживания для диагностики IOP
0		Штатная работа СР113, выполнение программных задач/процессов согласно из приоритетов.

3 этап. Обработка прерываний – выполнение задачи на уровнях прерываний от 0 до 7.

Способ обработки прерываний определяется индивидуально для каждой задачи и зависит от ее относительной важности для системы управления в целом и уровня прерывания, на котором выполняется программа обработки задачи.

Классификация программ по уровням прерывания – то есть определение, на каком уровне прерывания выполняется каждая конкретная программа – является одним из основных вопросов, который решается на уже стадии проектирования программной системы управления.

3.3.4 Межпроцессорный обмен

Как отмечалось в подразделе 3.1, процессоры BAP и CAP физически присоединяются к обеим шинам B:CMУ через общий интерфейс (common interface, CI). Общий интерфейс используется как для физического доступа к общей памяти СМУ так и для межпроцессорной связи. В обмене данными может участвовать аппаратный трейсер (отладчик) для реализаций специальных операций по контролю и устранению ошибок. Общий интерфейс CI, по сути, является развитой системой ввода–вывода и содержит следующие компоненты :

- регистры сигнализации блока интерфейса CI процессора PU для B:CMУ;
- средства обнаружения ошибок блока интерфейса CI процессора PU;
- буферы приёма/передачи для временного хранения адресов и данных;
- средства управления (логика) обмена данными.

В многопроцессорной системе каждый процессор (источник сообщения) должен уметь соединиться с любым другим процессором (получателем сообщения). Это достигается благодаря:

- наличию аппаратных средств, через которые в процессоре – получателе сообщения может быть задействовано требуемое прерывание (логика прерывания);
- специальная зона (область) связи (communication area, CA) между процессорами в общей оперативной памяти СМУ.

Для каждого процессора в СМУ имеется таблица указателей с 16-ю адресными зонами; каждая адресная зона связана с 16-ю прерываниями, используемыми для обмена сообщениями между процессорами Число 16 обусловлено максимальным количеством процессоров на рис. 3.1. Связь между процессорами происходит следующим образом(см. рис. 3.5).

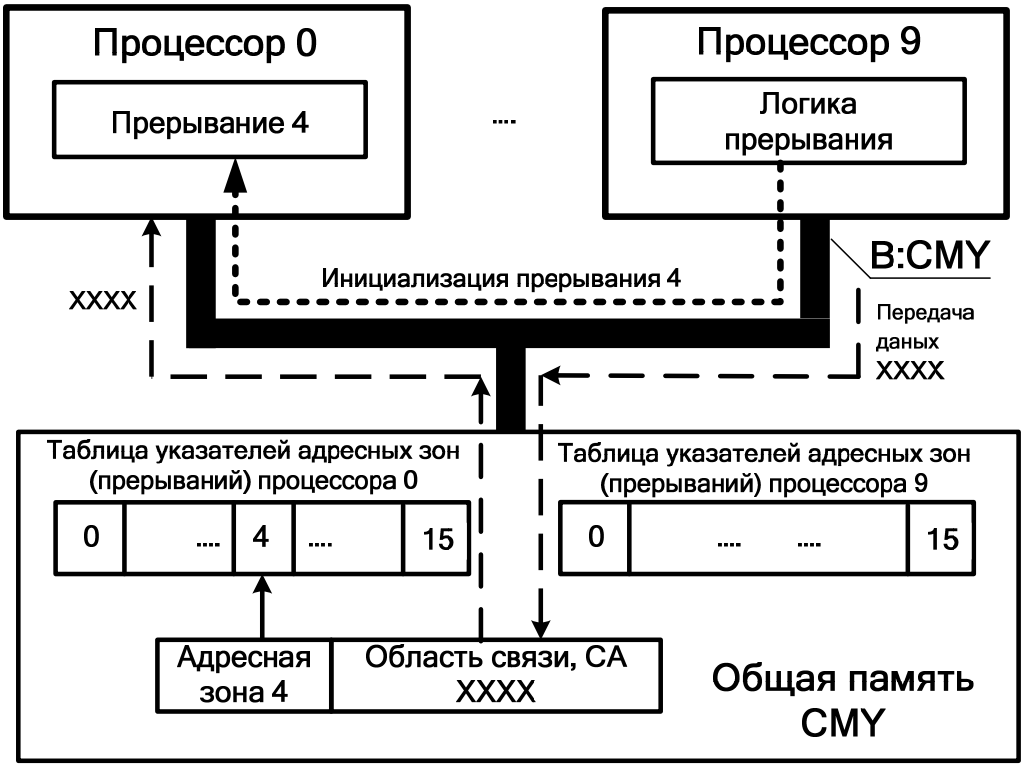


Рис. 3.5 – Схема межпроцессорного обмена [12]

На рис. 3.5 процессор 9 посылает сообщение процессору 0. Для этого процессор 9 из числа доступных в общей памяти CMY резервирует область связи CA. Область связи представляет собой совокупность ячеек памяти CMY, куда записываются сообщения от одного процессора к другому. В выбранную область связи CA процессор 9 через CMY записывает некоторое сообщение XXXX для процессора 0. Адресная зона, т.е. физический адрес начальной ячейки области связи CA заносится в таблицу указателей адресных зон процессора 0 (в рассматриваемом примере зоной связи назначается адресная зона 4). Далее процессор 9 через B:CMY инициирует прерывание в процессоре 0, причём номер прерывания соответствует зоне связи 4. Прерывание 4 инициируется средствами межпроцессорной коммуникации с помощью посылки сообщения. В результате этого прерывания процессор 0 читает начальный адрес области связи CA в ячейке 4 таблицы указателей адресных и считывает сообщение XXXX из соответствующей адресной зоны. Следует отметить, что межпроцессорная связь может иметь место только через общую память CMY. Итак, процессоры в составе CP113c, не могут напрямую получать доступ к локальной оперативной памяти LMY других процессоров.

3.4 Надёжность многопроцессорных управляющих комплексов

Многопроцессорные управляющие комплексы, наряду с высоким быстродействием, должны обеспечивать высокую надежность и отказоустойчивость. Это достигается за счет способности к реконфигурации и многократному (как правило, двукратному) дублированию наиболее критичных компонент управляющего комплекса. ***Надежность*** в целом определяется как свойство объекта (управляющего комплекса) сохранять во времени в установленных пределах значения всех параметров, характеризующих способность выполнять требуемые функции в задан-

ных режимах и условиях применения, технического обслуживания, хранения и транспортирования (ГОСТ 27.002–89). Под ***отказом*** понимается случайное событие, нарушающее работоспособность управляющего комплекса. Различают ***самоустраняющийся отказ*** или однократный отказ, устраняемый незначительным вмешательством оператора (инженера по эксплуатации), ***перемежающийся отказ*** – многократно возникающий самоустраняющийся отказ одного и того же характера, а также ***явный отказ*** – отказ, обнаруживаемый визуально или штатными методами и автоматизированными средствами контроля и диагностирования в процессе применения управляющего комплекса по назначению. Существует также ***критический отказ*** – отказ управляющего комплекса в целом или его отдельного компонента, тяжесть последствий которого в пределах данного анализа признана недопустимой и требует принятия специальных мер по снижению вероятности данного отказа и/или возможного ущерба, связанного с его возникновением (согласно ГОСТ 27.310–95).

Применительно к управляющим устройствам в составе управляющих комплексов различают:

- критический отказ – по сути, аварийная ситуация;
- явный отказ (полный выход из строя одного из элементов УК);
- сбой т.е. самоустраняющийся или перемежающийся отказ элементов УК.

Сбой возникает вследствие одновременного неблагоприятного изменения нескольких параметров устройства и существует сравнительно кратковременно. Перемежающиеся отказы могут возникать, например, при плохом контакте во врубной колодке, посредством которой модуль подключается к монтажной (материнской) плате.

Для оценки надежности аппаратуры управляющего комплекса рассмотрим случай невосстанавливаемых изделий, когда отсутствует возможность ремонта или замены элементов управляющего комплекса

в разумные сроки. Примером этой ситуации является управляющий комплекс космического спутника связи или отсутствие запасных частей на складе с временем доставки менее 24 часов. Очевидно, что такой подход даст нижнюю предельную оценку показателей надёжности. Все остальные оценки для изделий с возможностью восстановления в разумное время, будут, по крайней мере, не хуже полученной для невосстанавливаемых изделий.

Основными параметрами надежности для невосстанавливаемых изделий являются ***интенсивность отказов*** λ, в**ероятность безотказной работы** за время t, $P(t)$, **среднее время работы до отказа** T:

$$\lambda \approx \frac{m}{N \cdot t} \tag{3.1}$$

$$P(t) = e^{-\lambda t} \tag{3.2}$$

$$T = \int_0^{\infty} P(t)\,dt = \frac{1}{\lambda}, \quad \text{где} \tag{3.3}$$

m – число единиц оборудования, отказавших за время t,

N – число исправных единиц оборудования на начало промежутка времени t;

λ – параметр экспоненциального распределения времени работы до отказа.

Параметр λ определяет долю (а не количество) изделий или единиц оборудования, отказавших в течении заданного интервала времени, например за один час. На этапе промышленной эксплуатации управляющих комплексов в штатном температурно-влажностном режиме интенсивность отказов изделий можно оценивать по математическому ожиданию этой величины. Для интегральных полупроводниковых микросхем значение интенсивности отказов составляет $\lambda=10^{-6}...10^{-8}$

1/час, поэтому среднее время работы до отказа отдельной микросхемы достаточно большое, хотя и не бесконечное: $T = 10^6 \ldots 10^8$ часов. В состав управляющего комплекса входят сотни МПр и микросхем, отказ одной из них необязательно приводит к полному отказу модуля или управляющего комплекса. Кроме того, если микросхемы сами по себе и не восстанавливаемы, то модульная структура оборудования всегда позволяет заменить плату(модуль) с вышедшей из строя микросхемой. Возможна и замена МПр путём перемонтажа. Поэтому на практике критический отказ – событие, состоящее в выходе из строя 50% и более управляющего комплекса современного средства связи – достаточно маловероятно и происходит редко. Тем более маловероятен полный отказ координационного процессора СР113 по причине выхода из строя аппаратных средств; фирма Siemens оценивает частоту такого события как 1 раз в 300 лет.

Рассмотрим расчёт вероятности безотказной работы управляющих комплексов с учётом сделанных допущений и предположений.

Одним из основных способов повышения надёжности таких систем является ***резервирование –*** способ обеспечения надежности объекта за счет использования дополнительных средств и (или) возможностей, избыточных по отношению к минимально необходимым для выполнения требуемых функции. Различают два вида резервирования: общее и раздельное (поэлементное, параллельное).

Пусть компоненты (единицы) управляющего комплекса объединены в последовательную систему, где отказ любой компоненты приводит к отказу системы в целом. Компоненты управляющего комплекса включаются последовательно, одна за другой и формируют общий контур управления. При общем резервировании для повышения надежности резервируется весь контур управления, т.е. все взаимодействующие компоненты. В итоге, в схеме появляется избыточность, вызванная по-

явлением резерва в виде второго, третьего, m-го контуров управления (см. рис. 3.6).

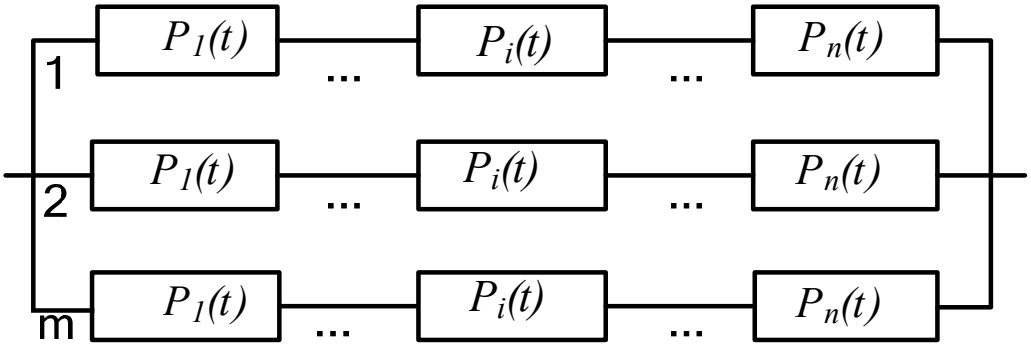


Рис. 3.6 – Схема общего резервирования

Как уже отмечалось, вероятность безотказной работы схемы на рис. 3.6 оценим для предельного случая, когда полностью отсутствует возможность замены вышедшего из строя элемента из ЗИП или ремонт элемента в разумные сроки (24 часа) невозможен. Пусть отказ одного элемента не зависит от отказа другого элемента; при этом отказавший элемент рассматривается как полностью неработоспособный, т.е. не имеет место перемежающийся отказ, когда элемент периодически выдаёт сигнал сбоя, а потом временно переходит в работоспособное положение. Тогда вероятность безотказной работы $P_{общее}(t)$ схемы на рис. 3.6 оценивается по формуле [38]:

$$P_{общее}(t) = 1 - (1 - \prod_{i=1}^{n} p_i(t))^m, \qquad (3.4)$$

где

n – число компонентов (управляющих устройств, процессоров);

m – число контуров (параллельных компонентов) резервирования;

$p_i(t)$ – вероятность безотказной работы отдельного i-го компонента в m-ном контуре за время t.

В случае общего резервирования наиболее часто используется режим работы, при котором контур (или компонент), выступающий в качестве резерва, включен с самого начала и обрабатывает ту же нагрузку, что и основной контур (компонент). Это случай т.н. нагруженного или «горячего» резерва. Контуры управления здесь автоматически делятся на активный и резервный. В случае выхода из строя одного контура производится автоматическое безобрывное переключение на резервный контур с выдачей аварийного сообщения оператору. Одновременно производится автоматический запуск процедур реконфигурации управляющего комплекса с целью повторного запуска/изоляции отказавшего контура. В дальнейшем отказавший контур или его часть ремонтируется, как правило, путём замены отказавших компонент на исправные. По указанной схеме резервируются, например, управляющие устройства АТСКЭ типа «Квант» – двухмашинный управляющий комплекс.

Недостатком общего резервирования является его неэкономичность, так как объем оборудования управления возрастает в общем случае в *m* раз (для АТСКЭ «Квант» *m=2*), а производительность остается на уровне производительности одного управляющего комплекса. Поэтому такое резервирование управляющего комплекса применяется в системах коммутации с малой ёмкостью (до 8 000 номеров) и централизованным управлением. Также эта схема может применяться в групповых управляющих устройствах, где дополнительная стоимость резервного маломощного управляющего устройства существенно не увеличивает стоимость средства связи в целом. Следует отметить, что для невосстанавливаемых систем при m = 3…4 общее резервирование позволяет достичь характеристик, близких к идеальным, т.е. существует почти стопроцентная гарантия работоспособности управляющего комплекса. Однако этот вариант технического решения является самым дорогостоящим и применяется в исключительных случаях.

Для повышения эффективности общего резервирования возможен вариант разделения обрабатываемой нагрузки в соотношении 50/50 т.е. в нормальном режиме 50% нагрузки обрабатывает основной контур, 50% нагрузки обрабатывает резервный контур. При отказе основного контура, резервный контур без задержки подключается как основной. Кроме того, может применяться режим «холодного» резервирования, при котором резервный управляющий комплекс подключается только после полного отказа основного управляющего комплекса. Этот способ позволяет продлить срок эксплуатации резервного комплекса. Однако на время запуска резервного управляющего комплекса средство связи не сможет выполнять свои основные функции т.е. соединения абонентов будут потеряны, а время восстановления увеличивается.

Отсутствие возможностей восстановления отказавшего устройства характерно для систем космической связи, которые находятся на околоземной орбите, а также для систем связи, недоступных для технического обслуживания (например, специальная или разведывательная аппаратура). Кроме того, может отсутствовать обслуживающий персонал, готовый к использованию ЗИП. Для остальных случаев значительно эффективнее выглядит раздельное (параллельное, поэлементное) резервирование наиболее критически важных компонентов. При раздельном резервировании резервируются отдельные компоненты управляющих устройства, например отдельные процессоры, каналы ввода/вывода, элементы памяти и общесистемные шины (см. рис. 3.7).

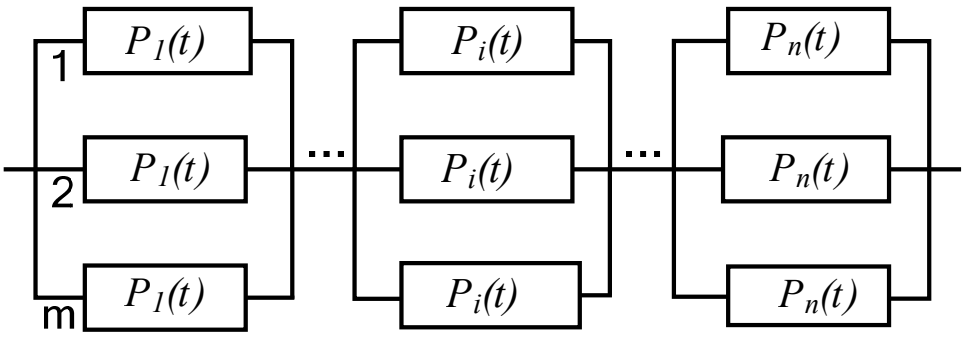


Рис. 3.7 – Блок-схема раздельного резервирования

В случае отказа одного компонента его функции выполняет дублирующий компонент без существенной потери качества связи или производительности управляющего комплекса. В этом случае вероятность безотказной работы комплекса на рис. 3.7., $P_{разд}(t)$, определяется по формуле :

$$P_{разд}(t) = \prod_{i=1}^{n}[1-(1-p_i(t))^m]. \quad (3.5)$$

В целом для одних и тех же исходных данных раздельное резервирование более эффективно чем общее.

Рассмотрим оценку надежности процессора СР113с. С учётом введённых понятий об общем и раздельном (параллельном) резервировании построим блок-схему для оценки надежности СР113с (см. рис. 3.8).

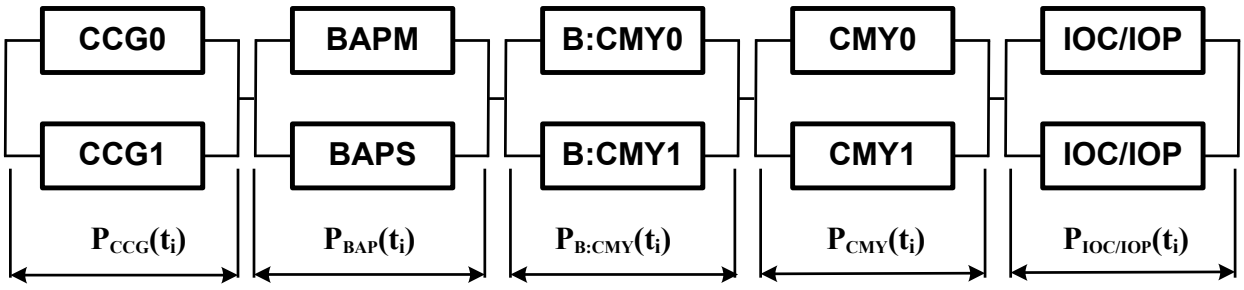


Рис. 3.8 – Упрощенная блок-схема обеспечения надежности СР113 для случая полного простоя системы

Очевидно, что здесь имеет место случай раздельного резервирования. Вероятность безотказной работы схемы можно оценить по формуле :

$$P_{CP113}(t) = P_{CCG}(t_i) \times P_{BAP}(t_i) \times P_{B:CMY}(t_i) \times P_{CMY}(t_i) \times P_{IOC/IOP}(t_i) \quad (3.6)$$

где

$P_{CP113}(t)-$ вероятность безотказной работы процессора СР113 в целом;

$P_{CCG}(t_i), P_{BAP}(t_i), P_{B:CMY}(t_i), P_{CMY}(t_i), P_{IOC/IOP}(t_i)$ – вероятность безотказной работы отдельных раздельно резервированных компонентов процессора CP113;

t_i – время работы i-го компонента;

В свою очередь, с учётом формулы (3.5) :

$$P_{CCG,BAP,B:CMY,CMY,IOC/IOP}(t_i) = 1-(1-e^{-\lambda_i t_i})^2, \qquad (3.7)$$

где

λ_i – интенсивность отказов i-го компонент (CCG, BAP, B:CMY, CMY, IOC/IOP);

$m=2$ – для случая на рис. 3.8.

Рассмотрим ещё один случай резервирования. Пусть многопроцессорный управляющий комплекс содержит l одинаковых процессоров, h из которых обладают мощностью, достаточной для обеспечения штатной производительности управляющего комплекса. Тогда получается, что число $l–h$ процессоров являются избыточными. В нормальном режиме работы избыточные процессоры, как правило, работают в режиме разделения нагрузки с основными процессорами. Это так называемый случай нагруженного резерва. При этом резервные процессоры в любой момент времени могут на 100% заменить отказавший основной (базовый) процессор. Другими словами, если бы процессоры загружались на 100%, их число можно было бы уменьшить, но при этом ни о каком резервировании говорить не приходилось бы.

Если «избыточный» процессор полноценно заменяет любой из основных в случае отказа последнего, то реализуется т.н. «плавающее» резервирование. Примером плавающего резервирования является случай «n+1», где n – количество рабочих компонент, 1 – количество резерва. Разновидностью такого резервирования является кратное резервирование с постоянно включенным резервом (в случае, если

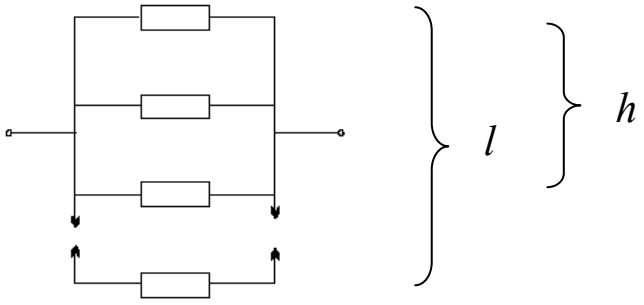


Рис. 3.9 – Блок-схема обеспечения надежности для случая резервирования с дробной кратностью и постоянно включенным (нагруженным) резервом

оказавший процессор не ремонтируется или не заменяется из ЗИПа за приемлемое время, например за 8 часов). Пусть резервированная система управления состоит из l отдельных равнонадежных процессоров с вероятностью безотказной работы p_0 каждый (см. рис. 3.9). Для нормальной работы схемы на рис. 3.9, например, для обеспечения заданной производительности управляющего комплекса в ЧНН, необходимо, чтобы исправными были не менее чем h процессоров. Кратность резервирования такой системы m равна $m = (l– h)/h$. Вероятность безотказной работы системы $P_{сист}\ (t)$ на рис. 3.0 составит величину [38]:

$$P_{сист}(t) = \sum_{i=0}^{l-h} C_l^i p_0^{l-i}(t) \sum_{j=0}^{i} (-1)^j C_i^j p_0^j(t) \qquad (3.8)$$

где

$$C_l^i = \frac{l!}{i!(l-i)!} \qquad (3.9)$$

Для рассматриваемого случая также можно определить среднее время (математическое ожидание) работы системы до отказа T, которое составляет величину :

$$T = \int_0^{\infty} P_{сист}(t)dt \approx \frac{1}{\lambda_0}\sum_{i=0}^{l-h}\frac{1}{h+i}, \qquad (3.10)$$

где

λ_0 – интенсивность отказов отдельного компонента (процессора).

Отметим, что при $l–h = 2..3$ наработка на отказ управляющего комплекса будет приближаться к границе физического срока службы. С другой стороны, постоянное наличие явно избыточных процессоров (резервирующих компонентов) приводит к увеличению стоимостных показателей системы. В этом случае вариант $n+1$ является разумным компромиссом по соотношению «стоимость–эффективность». Примером резервирования «$n+1$» является процессоры BAPS–CAP; примером избыточности с разделением нагрузки являются процессоры IOP, IOC, DLUC.

Существует вариант с ненагруженным резервом, который учитывает необходимость затрат времени на включение резерва. В рассматриваемом случае кратность резервирования такой системы $m = m_0 / n,$ где m_0 – число резервных компонентов, находящихся в ненагруженном резерве, n – число компонентов основной системы, которые в случае выхода из строя заменяются на резервные. Тогда вероятность безотказной работы системы с ненагруженным резервом определяется по формуле:

$$P_{сист}(t) = e^{-\lambda_0 t}\sum_{i=0}^{m}\frac{(\lambda_0 t)^i}{i!} \qquad (3.11)$$

Среднее время (математическое ожидание) работы до отказа системы T для ненагруженного резерва составляет величину :

$$T = \frac{m+1}{\lambda_0} \qquad (3.12)$$

После восстановления отказавших компонентов средство связи будет выполнять функции в полном объеме. Восстановление производится в процессе функционирования системы.

На практике формулы оценки надёжности управляющих комплексов являются более сложными, т.к. они учитывают ремонтопригодность устройств, наличие и сроки замены вышедших из строя модулей из ЗИПа, воздействие на управляющих комплекс процедур контроля и диагностики, уровень и следствие отказов [39,40]. Для описания этих случаев используются модели Марковских процессов, а в качестве модели потоков отказов оборудования для предельных оценок целесообразно использовать пуассоновские потоки.

Для достижения требуемых показателей надёжности в современных средствах связи, как правило, реализован механизм автоматической реконфигурации. Под ***конфигурацией*** здесь понимается работоспособный набор управляющих устройств средства связи, обеспечивающий выполнение основных функций. ***Реконфигурация*** – автоматическое или ручное изменение состава набора управляющих устройств. Реконфигурация осуществляется, если в многопроцессорной управляющей системе имеются аппаратные и программные средства контроля и диагностики/теста, реконфигурации и повторного запуска системы после отказа (например, рестарт управляющей программы системы коммутации).

Средства контроля могут быть реализованы аппаратным, программным или программно-аппаратным способом. Аппаратные средства контроля предназначены для контроля передачи информации и контроля правильности выполнения арифметико-логических операций. К примеру, контроль путей пересылки информации между ОЗУ и ЦПУ, ОЗУ и внешними устройствами производится на основе избыточного кодирования (коды Хэмминга, Грея и др.). Особенно распространен простой код с проверкой четности, стандартно требующий только одно-

го дополнительного проверочного разряда на блок восьми двоичных разрядов. Эти же методы, в совокупности с дублированием основных аппаратных компонент, используются для контроля операций в АЛУ. Аппаратный контроль выполняется с минимальной задержкой по времени, практически одновременно/параллельно с основными вычислениями, что и является основным достоинством аппаратного контроля. Автоматическая реконфигурация в системе коммутации возможна в случае развитых аппаратных средств переключения, например переключение на резерв. Простейшим случаем является безобрывное подключение резервного контура или «избыточного» управляющего устройства, как это сделано в процессорах BAPM и BAPS.

Программный контроль не требует дополнительного оборудования, однако такой контроль выполняется с задержкой во времени. Задержка обусловлена необходимостью произвести программные вычисления. К программным методам контроля относятся тестовый и программно-логический контроль. Тестовый контроль выполняется периодически, либо при наличии свободного времени в управляющем устройстве. Также тестовый контроль выполняется после обнаружения отказа другими средствами, при этом частично могут быть выполнены функции диагностики. Примером программно-логического контроля является одно- или двукратное повторение вычислительной операции, проверка предельных значений переменных, вычисление и сравнение контрольных сумм файлов.

Рассмотрим основные показатели надёжности системы коммутации EWSD. Общие характеристики надежности при обслуживании приведены в таблице 3.2.

Для расчета требуемого количества запасных модулей используется пуассоновское распределение вероятностей выхода из строя. Существенными параметрами для расчета надежности является требуемая вероятность непрерывности обслуживания и время

Таблица 3.2 – Общие данные по надёжности EWSD v15
[данные компании Siemens]

Характеристика надёжности	Значение характеристика при обслуживании (все причины отказа)
Общее время простоя системы	3 минуты в год
Время простоя одиночного станционного окончания (порта)	30 минут в год
Время простоя звена ОКС№7	82 минуты в год
Вероятность преждевременного освобождения	2×10^{-5}
Ремонтные операции (аппаратные средства)	5 операций на 1000 портов и в год

оборота, которое определяется как интервал между временем заказа модуля замены и временем его получения. Требуемое количество запасных частей рассчитывается индивидуально для каждого проекта. После расчета требуемого количества модулей замены для конкретной системы коммутации можно рассчитать объем требуемых работ по техническому обслуживанию и ремонтных работ. Для системы коммутации EWSD этот показатель составляет в среднем приблизительно 2 отказа модуля в год на 1000 станционных окончаний (по данным Siemens).

3.5 Надёжность программного обеспечения

К системному и прикладному программному обеспечению управляющих комплексов средств связи предъявляются следующие требования:

- Вся адресация к памяти для внешних устройств должна выполняться на логическом уровне. В этом случае отказ компонента УК приводит только к изменению таблиц связи логических и физических адресов.

- Операционная система должна носить распределенный характер, то есть в оперативной памяти процессоров должны находиться копии машинных кодов ОС или фрагменты кодов ОС для ускорения программных процессов реконфигурации или восстановления.

В наибольшей степени перечисленным условиям удовлетворяют управляющие комплексы с возможностью параллельной обработки данных и с квази-распределенной или распределенной функциональной архитектурой (см. выше рис. 1.7 – 1.8).

Надежность программного обеспечения – способность программы для ЭВМ с достаточно большой вероятностью безотказно выполнять паспортные функции течение заданного периода времени. ***Степень надежности*** характеризуется вероятностью работы программы для ЭВМ без отказа в течение определенного периода времени. Надёжность является одним из основных показателей ПО управления средства связи. Надёжность ПО для пользователя или персонала по эксплуатации заключается в вероятности реализации следующих событий:

- появление сбоев, как результат воздействия регулярных или единичных ошибок ПО;
- выбор режима эксплуатации ОС, который способствует проявлению скрытых ошибок.

Некоторые данные [T. Ostrand, E. Weyuker, 2002] говорят о том, что большие программы для ЭВМ содержат от 6 до 16 ошибок на 1000 строк программного кода; по другим данным, существует от 2 до 75 ошибок на 1000 строк кода, при этом уровень ошибок в драйверах внешних устройств в 3..7 раз больше чем в обычном коде [A. Chou et all, 2001]. Следовательно, к примеру ОС Linux, содержит по крайней мере 15 тысяч ошибок, а Windows XP – в 2 раза больше. Это означает, что любое используемое ПО нельзя считать абсолютно надёжным. При

этом собственно количество ошибок в ПО не всегда может рассматриваться как единственная оценка надёжности ПО. Во-первых, пользователь сталкивается не собственно с ошибкой, а с результатом её проявления; во вторых – воздействие одной ошибки может компенсировать другую ошибку; в третьих – ошибки могут иметь разный «вес», одна ошибка сразу вызывает критический отказ, другая приводит к малозаметным последствиям. Следует иметь в виду, что устранение ошибок ПО, как правило, предусматривает модификацию программного кода, что, в свою очередь, приводит к изменению показателей надёжности ПО в целом. Поэтому любые изменения ПО должны проходить организованно, осуществляться только подготовленным персоналом, сделанные изменения должны обязательно проверяться и тестироваться в периоды снижения абонентской нагрузки на систему коммутации.

Этим частично объясняется тот факт, что «…пользователи порой предпочитают обновленным версиям программ старые, проверенные, эксплуатировавшиеся длительное время, даже если в них обнаружены погрешности: опыт эксплуатации стоит очень дорого, и даже если в программе выявлены ошибки, гораздо дешевле внести исправления и дополнения в инструкции к программе (если это, конечно, возможно), чем пожертвовать накопленным опытом». [цит. по Романюк С.Г. «Оценка надёжности программного обеспечения». – Открытые системы. – №4.–1994.]

Надежность ПО повышается прежде всего за счёт применения передовых методов проектирования, разработки и тестирования ПО, строгим соблюдением технологии эксплуатации и резервного копирования станционных и абонентских баз данных. Существуют и другие методы повышения надёжности [см. Таненбаум Э. и др. Надежные и защищенные операционные системы – Открытые системы.– №6.–2006.]. Например, для существующих операционных систем предлагается защищать ядро операционной системы (kernel) от драйверов внешних устройств. Драйверы окружаются специальной «тонкой» программной оболочкой, кото-

рая отслеживает взаимодействие драйвера с ядром операционной системы. Оболочка анализирует все запросы между ядром ОС и драйвером на предмет корректности. При этом для драйверов все доступные страницы виртуальной памяти, относящиеся к программам ядра операционной памяти, переводятся в режим «только для чтения» (read only). Также используются дублирующие драйверы, для того, чтобы внешние программы могли выполняться корректно после сбоя драйвера. Дублирующий драйвер в штатном режиме регистрирует всю информация между работающим драйвером и ядром ОС. После перезапуска дублирующий драйвер передаёт основному все ранее зарегистрированные данные для ускорения процесса восстановления.

Ещё одним средством, обеспечивающим изоляцию ошибок операционной системы, является виртуализация. Если драйверы устройств работают в одной или в нескольких виртуальных машинах, изолированных от основной виртуальной машины, где работает остальная операционная система и прикладные программы, то в случае сбоя в драйвере выходит из строя только его виртуальная машина, а не основная. Однако в случае перезапуска, драйвер будет запущен со значениями по умолчанию, а не со значениями на момент сбоя.

Надёжность ПО можно повысить ранее рассмотренным способом в ОС РВ QNX – применением одного или нескольких микроядер, которые защищены от ошибок в драйверах внешних устройств. Микроядро (microkernel) обрабатывает аппаратные прерывания, предоставляет базовые механизмы для управления процессами, реализует взаимодействия между процессами IPC и выполняет планирование процессов. Микроядро предоставляет небольшой набор возможностей по вызову функций микроядра для авторизованных драйверов, например чтение избранной части адресного пространства конкретного пользователя или запись в авторизованные порты ввода/вывода. «Над» или «вокруг» микроядра находится уровень драйверов устройств. Каждое устройство

ввода/вывода имеет свой собственный драйвер, который функционирует как отдельный процесс в своем собственном частном адресном пространстве, защищенном с помощью аппаратного модуля управления памятью (MMU). Уровень драйверов устройств включает в себя процессы драйверов для НЖМД, терминала (клавиатуры и дисплея), модуля абонентских блоков или блоков соединительных линий. Эти драйверы работают в режиме пользователя и не могут выполнять привилегированные команды, операции чтения и записи на портах ввода/вывода управляющего комплекса. Для того, чтобы получить эти сервисы для обмена данными, драйверы внешних устройств должны обратиться к ядру операционной системы. Хотя такая архитектура увеличивает накладные расходы на реализацию решения, она значительно повышает надежность – ведь здесь микроядро, в отличие от традиционных «монолитных» операционных систем, полностью контролирует и ограничивает действия драйверов.

Далее в рассматриваемом решении, «над» уровнем драйверов внешних устройств находится уровень менеджера процессов, который принимает запросы от процессов операционной системы на вызовы функций микроядра, таких как, read, write, и выполняет их; здесь же реализуется функции управления памятью. На этом же уровне находится сетевой интерфейс (сервер сети), который поддерживает стек протоколов TCP/IP, поддержка доменных имен. И только «над» уровнем менеджеров находятся собственно программы пользователя – база данных, восстановление, программы управления (см. выше рис. 2.1).

Указанная конструкция, которая уже применяется в рамках ОС РВ QNX, обеспечивает высокую надежность за счёт распределения функций, ошибки драйверов не могут нанести непоправимого ущерба ядру операционной системы, поскольку все процессы и полномочия строго разделены. Драйверы и процессы могут пользоваться только собствен-

ными адресными пространствами, причём для команд и данных используются различные пространства адресов.

Некоторым недостатком данной мультиядерной структуры является снижение производительности до 10% за счёт того, что драйверы выполняются как внешние процессы, а не встроенные компоненты операционной системы.

Ещё одним решением задачи повышения надёжности ПО является применение специальных языков программирования, которые предусматривают строгую формализацию описания процессов и их взаимодействия при сохранении возможности использования единого адресного пространства. Ни один процесс не сможет изменять данные другого процесса.

Опыт эксплуатации показывает, что чем больше запусков ПО или чем дольше цикл работы ПО без сбоев – тем надёжнее программное обеспечение. Чем дольше период эксплуатации ПО – тем вероятнее обнаружение и устранение ошибок ПО, этим программное обеспечение выгодно отличается от аппаратных средств.

На этапе проектирования и разработки программного обеспечения для управления средством связи следует применять разнообразные методы, которые можно разбить на следующие группы :

- Предупреждение ошибок – методы, позволяющие минимизировать или исключить появление ошибки.
- Обнаружение ошибок – методы, направленные на разработку дополнительных функций программного обеспечения, помогающих выявить ошибки.
- Устойчивость к ошибкам – реализация дополнительных функций программного обеспечения, предназначенных для исправления ошибок и их последствий, обеспечивающих штатное функционирование программной системы при наличии ошибок.

При разработке программного обеспечения управляющих комплексов средств связи разработчику следует последовательно применять все три группы методов с расчётом на то, большая часть ПО управления функционирует автоматически в реальном масштабе времени. С учётом вышесказанного, в качестве практического примера, рассмотрим программное обеспечение управления EWSD.

3.6 Программное обеспечение управления EWSD

3.6.1 Общие сведения о программном обеспечении EWSD

Принципы построения и структура программного обеспечения управления EWSD соответствуют квази-распределенной функциональной архитектуре управления на рис. 1.7. В системе EWSD управление соединениями осуществляют координационный процессор CP113c, в обработке вызовов участвуют процессор GP в составе LTG, процессоры DLUC в составе DLU. Администрирование, дополнительные функциональные возможности и автоматическую техэксплуатцию осуществляет координационный процессор CP113c. Функции обработки данных звена общеканальной сигнализации осуществляет процессор CCNC в составе устройства управления ОКС №7 (EWSD v15). Подробно эти вопросы рассматриваются в учебном пособии [27] и в книге [41].

Программное обеспечение управления EWSD организовано в соответствии с функциями, выполняемыми отдельными программными подсистемами управления EWSD. Программное обеспечение управления EWSD разработано на языках программирования CHILL [1], Си, Ассемблер.

Программное обеспечение устанавливается в системе EWSD одним из двух способов:

- в виде готовой системы прикладных программ (application program system, APS), загружаемой в координационный про-

цессор CP113 или в главные процессоры MP сетевого контроллера системы сигнализации (SSNC, применяется начиная с EWSD v15) с магнитной ленты или магнитооптического диска;

- в виде замонтированного микропрограммного обеспечения, хранящегося в постоянных запоминающих устройствах FEPROM подсистем EWSD.

Версия APS загружается в оперативную память CMY с внешнего носителя (НМЛ, НОД), и далее записывается на НЖМД. Часть программных модулей APS постоянно (резидентно) находятся в оперативной памяти; некоторые программы APS записаны на НЖМД и при необходимости загружаются в оперативную память. Программное обеспечение EWSD имеет модульную структуру. ***Программный модуль*** представляет собой фрагмент машинного кода, он является наименьшим блоком, полученным в результате компиляции. Модуль может содержать, в соответствии с используемым языком CHILL, следующие операторы и процедуры языка программирования:

- определение используемого кода;
- используемые константы;
- объявления данных;
- определения процедур;
- определение типа процесса.

Средний размер программного модуля в программном обеспечении EWSD, как правило, не превышает 1000 операторов языка программирования.

Программное обеспечение начальной загрузки и ПО диагностики аппаратной части выполнено в виде замонтированного микропрограммного обеспечения, которое хранится FEPROM. Микросхемы FEPROM установлены практически на каждом модуле EWSD, который оборудован процессором любого назначения. Это СППЗУ допускает

коррекцию хранимого программного обеспечения с помощью специальных программ перепрошивки или обновления.

Показатели качества программного обеспечения EWSD формируются на основании группы стандартов ISO 9000-2000. Среди этих показателей следует выделить следующие :

Надежность программного обеспечения характеризуется следующими показателями:

- техническая правильность реализации функций системы коммутации;
- законченность ПО;
- возможность предотвращения отказов ПО;
- защита от отказов;
- устойчивость к перегрузкам (например, ограничение числа входящих вызовов).

Обеспечение надёжности программного обеспечения достигается комплексом мер. В частности, непротиворечивость, целостность и предотвращение ошибок программного обеспечения достигаются следующими мерами:

- защита файлов от несанкционированного доступа;
- периодические проверки на непротиворечивость данных;
- специальные меры безопасности, предотвращающие множественный доступ к данным;
- проверки на достоверность и непротиворечивость данных, передаваемых по интерфейсу с внешними устройствами;
- процедуры проверки контрольной суммы файлов для контроля данных и программы.

Защита от отказов и минимизация распространения ошибок достигается следующими мерами:

- разделение программы и данных на отдельные, динамически подключаемые модули, которые хранятся в раздельных областях физической памяти;
- защита памяти для программ и для полупостоянных данных;
- резервное копирование и дублирование системных файлов и файлов пользователя;
- текущий контроль реакции в реальном времени для программ;
- текущий контроль производительности системы.

В результате реализации перечисленных мероприятий время полного простоя системы коммутации EWSD, связанного с отказом программного обеспечения, составляет в общем времени простоя системы в среднем приблизительно 1 минуту в год (данные компании Siemens).

Удобство технического обслуживания ПО обусловлено следующими показателями:

- модульность системы;
- модифицируемость ПО – возможность изменение программного кода без перекомпиляции всего ПО;
- тестируемость, т.е. возможность проверки.

Удобный способ взаимодействия с пользователем обусловлен простотой использования, легкостью изучения интерфейса пользователя; устойчивостью к ошибкам оператора (т.н. «защита от дурака»).

Эффективность ПО обусловлена предсказуемым поведением ПО в процессе эксплуатации.

Программное обеспечение EWSD может рассматриваться в рамках концепции разработки ПО. Эта концепция включает:

- коммуникационную модель ПО;
- управление выполнением программ в реальном времени;
- принципы программирования.

Длительность разработки базовой версии ПО составляет 5…7 лет и включает десятки миллионов человеко-часов. В процессе разработки программное обеспечение EWSD приобретает достаточно сложную структуру, оптимизированную для выполнения программ в реальном времени. Рассмотрим процессы разработки ПО и структурные блоки программного обеспечения управления EWSD более подробно.

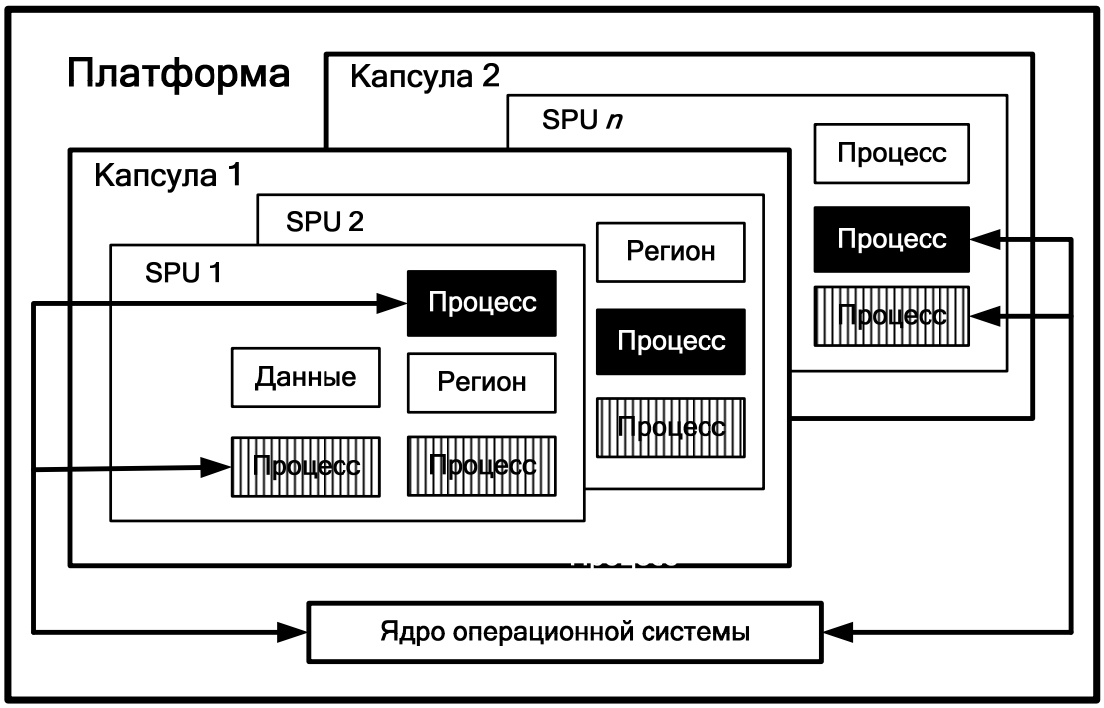
Разработка ПО EWSD включает следующие этапы : планирование будущего ПО; проектирование программного обеспечение (разработка SDL-диаграмм, структурных описаний, алгоритмов); производство программного обеспечения, включая разработку программ с использованием соответствующих языков программирования; установка ПО на систему EWSD; тестирование и отладку ПО; эксплуатацию ПО; снятие ПО с эксплуатации путём деинсталляция или замены/обновления версий ПО.

При разработке программного обеспечения EWSD используется термин «платформа обработки» или процессинговая платформа (processing platform), под которой подразумевается совокупность аппаратных средств процессора CP113c, операционной системы и средств разработки ПО. Это понятие используется на этапе планирования будущей разработки программного обеспечения (см. рис. 3.10).

Процессинговая платформа содержит одну или несколько капсул (capsule). Этот структурный блок программного обеспечения формируется на этапе производства ПО.

Капсула представляет собой блок распределения машинных/аппаратных ресурсов с точки зрения защищенного диапазона виртуальных адресов. Под ***защитой*** здесь понимается резервирование группы виртуальных адресов для реализации того или иного блока предоставления услуг (service provision unit, SPU). Блок SPU формируется на этапе проектирования программного обеспечения. В свою очередь SPU содержит процессы, данные и процедуры, а также регионы и

документированные интерфейсы для обмен с другими SPU. Распределение блоков SPU по капсулам и распределение капсул по платформам происходит на этапе производства программного обеспечения. Во время исполнения капсула целиком находится на заданной платформе.



Условные обозначения :

- виртуальный CPU 1

- виртуальный CPU 2

- распределение ядом ОС временного бюджета между CPU

SPU - блок предоставления услуг;
OS - операционная система;
CPU - виртуальный процессор;
SPU - способ структуризации прикладного ПО (на уровне алгоритма).

Рис. 3.10 – Структура программного обеспечения EWSD с двумя виртуальными CPU

Регион в данном случае является программным средством синхронизации, позволяющее осуществлять монопольный доступ к программно-аппаратным ресурсам СР113. Регион содержит данные и процедуры, однако, не включает в себя каких-либо процессов; регион распределяет доступ к ресурсам между различными SPU. Ре-

гионы используются для административного управления совместно используемыми данными. Виртуальный процессор на рис. 3.10 аналогичен виртуальной машине, ранее рассмотренной на рис. 2.2.

Рассмотрим более подробно функции операционной системы, позволяющие поддерживать схему на рис. 3.10.

3.6.2 Функции операционной системы реального времени EWSD

Операционная система EWSD выполняет общие организационные функции в отношении управления аппаратно-программными ресурсами средства связи. Операционная система в виде отдельных ядер ОС (kernel OS) загружается в локальные ОЗУ (LMY) процессоров. Ядра операционных систем процессоров BAP, CAP практически идентичны. Операционная система EWSD должна выполнять свои задачи в режиме реального времени, поэтому она поддерживает систему прерываний и систему приоритетов прерываний и процессов. Процессоры IOC и IOP содержат операционную систему, адаптированную к их конкретным функциям. Операционная система данных процессоров представляет собой «подмножество» той операционной системы, которая используется в BAP/CAP.

Центральным компонентом операционной системы EWSD является управляющая программа. К функциям управляющей программы ОС относятся следующие:

Инициализация – этот процесс выполняется во время запуска системы коммутации EWSD в эксплуатацию и в течение восстановления системы после сбоя или отказа. Задачей инициализации является подготовка процессоров к работе. Каждый процессор в процессе инициализации выполняет свою собственную инициализацию.

Планирование процессов – это назначение процессоров для выполнения отдельных процессов на определенные интервалы времени.

Обработка прерываний выполняется операционной системой для пользовательских (прикладных) программ, активизируется с помощью прерываний. Уровень 7 имеют прерывания с самым высоким приоритетом, а уровень 0 – прерывания с самый низким приоритетом. Процессы обычно выполняются на уровне прерываний 0. Входящим событиям назначаются уровни прерывания выше 0. Наступление какого-либо события приводит к приостановке выполняющегося в текущий момент времени процесса и активизации функции обработки прерываний. Функция обработки прерываний сохраняет текущий статус выполняемого процесса и вызывает подпрограмму прерываний для запрашиваемого уровня прерывания.

Управление процессами – здесь осуществляется запуск и завершение процессов, выполняемых в различных процессорах, и управляют ресурсами (например, буферами).

Связь между программными процессами – обеспечение взаимодействие между различными процессами, выполняемыми в CP113c. В качестве системы связи используются буферы в CMY, к которым имеется доступ из всех процессоров BAP и CAP.

Межпроцессорная связь – управление узлом связи на основе EWSD посредством обмена сообщениями между операционными системами в различных процессорах.

Управление таймированием – установка и обновление системной даты и времени с точностью до секунды, обработка задач таймирования и вычисление нагрузки процессоров.

Управление памятью – предусматривает хранение и обновление информации о расположении, размерах и именах загруженных программных модулей; преобразование логических адресов в физические (управление памятью); временное выделение области памяти для прикладных программ, например для вывода данных; контроль всех операций чтения и записи в CMY и LMY.

Ещё одним важным компонентом операционной системы являются программы–загрузчики. При первоначальном вводе системы коммутации в эксплуатацию и после каждого полного перезапуска программного обеспечения необходимо выполнять загрузку программного обеспечения в СР113с. Эта задача выполняется загрузчиками, которые содержат следующие программы:

- системный загрузчик, который включает базовую систему ввода-вывода;
- загрузчик образа оперативной памяти;
- загрузчик локальной памяти, включая библиотеку программ модуля LIB113.

Все перечисленные функции практически реализованы, загружаются и функционируют в системе в виде отдельных файлов. Примерный список файлов операционной системы EWSD и прикладных программ APS приведён в таблице 3.3. Таблица 3.3 соответствует программному обеспечению EWSD v10.

Рассмотрим далее подробнее особенности программы-загрузчика. Базовая система ввода-вывода EWSD используется конкретными пользовательскими процессами, в частности, программой начальной загрузки, для посылки запросов ввода-вывода на логическом или физическом уровне (например, для считывания или записи из/в файлы на НЖМД или НОД).

Загрузчик образа оперативной памяти используется для загрузки программного обеспечения в физическую память CMY и LMY в процессоре BAPM.

Загрузчик локальной памяти используется для загрузки ПО в LMY в процессорах BAP, CAP и контроллерах IOC. Библиотека модуля LIB113 содержит файлы, передаваемые загрузчиком образа оперативной памяти из НЖМД/НОД в CMY и LMY.

Таблица 3.3. – Файлы операционной системы и прикладных программ пользователей СР113с

Наименование файла (файлов) операционной системы	Назначение файла (файлов)
SY.GENLIST	Список генераций (экземпляров) программного обеспечения
SY.INSTALL	Файл, содержащий данные загрузки, выполняемой с внешнего носителя (НОД, НМЛ) для хранения данных
SY.TASKLIST	Файл, содержащий резидентное (постоянно хранящееся в ОЗУ) программное обеспече-
SY.SEMILIB	Файл, содержащий полупостоянные данные СР113 (генерируемые из SY.TASKLIB)
SY.LOADLIB.LA	Файл, содержащий нерезидентное (постоянно не хранящееся в ОЗУ) программное обеспечение СР113
SY.LOADLIB.MA	Файл, содержащий нерезидентные маски т.е. шаблоны вывода данных для инженерного персонала EWSD
SY.LOADLIB.CA	Файл, содержащий описание нерезидентных команд, доступных операторам EWSD.
SY.PSW.Txxx	Файл, содержащий программное обеспечение для процессоров периферийных УУ (зависит от состава проекта EWSD)
SY.PSW.T098	Файл, содержащий программное обеспечение для устройства управления ОКС№7 CCNC
SY.SIMP	Файл, содержащий полупостоянные данные для CCNC
CA.SU.UCHA	Абонентские тарифные счётчики
CA.ST.UCHA	Статистические счётчики
CA.TR.UCHA	Счётчики соединительных линий
TCA.CA.xxx	Копия текущего содержимого счетчиков (показывается только при обращении к НЖМД)
LG(LQ).LOGx	Файл журналирования с регистрацией действий инженерного персонала обслуживания системы EWSD и ответами системы на введённые команды
PW.LOG	Файл журналирования с регистрацией паролей доступа к программной системе EWSD
AM.ALARM	Файл с сообщениями о критических аварийных ситуациях системы EWSD

Наименование файла (файлов) операционной системы	Назначение файла (файлов)
HF.ARCHIVE	Файл с данными об аварийных сигналах, которые инициированы средствами технического обслуживания и регламентными тестами. Здесь содержатся шаблоны вывода данных с результатами восстановления системы после сбоев или отказов.

Ещё одним компонентом загрузчика является подпрограмма управления библиотеками, которая отвечает за административное управление библиотекой модуля LIB113. Под ***библиотекой*** здесь понимается совокупность подпрограмм (модулей), составленных на одном из языков программирования и удовлетворяющих определенным единым требованиям к структуре, организации их входов и выходов, описаниям подпрограмм. В свою очередь, библиотеки модулей, управляемые подпрограммой управления библиотеками, содержат системные файлы TASKLIB, SEMILIB, LOADLIB и другие файлы, перечисленные в таблице 3.3.

3.6.3 Процессы и их приоритеты в программной системе EWSD

Как было указано выше, процесс является составной частью программной задачи. Процесс формируется на этапе проектирования программного обеспечения EWSD. Важнейшими характеристиками процесса являются ***приоритет и класс процесса***, которые определяются на этапе разработки с учётом требований выполнения задач в реальном времени. Приоритеты позволяют управляющей программе (планировщику) операционной системы управлять попеременным выполнением процессов путем выбора процесса с наивысшим приоритетом из набора готовых к выполнению процессов. Приоритет процесса определяется уровнем прерывания, на котором он выполняется.

Всего в системе EWSD существует 16 приоритетов процессов, от нулевого до пятнадцатого. Нулевой приоритет соответствует самому низкому приоритету, а пятнадцатый приоритет соответствует самому высокому приоритету. Приоритеты с 1 до 6 и с 8 до 13 могут назначаться процессам произвольно. Приоритеты 0, 14 и 15 зарезервированы для системных процессов. Приоритет 7 зарезервирован целиком для процесса обработки вызовов.

Приоритеты и соответствующие им процессы сведены в таблицу 3.4. Значение приоритета, выбираемое для данного процесса в рамках приведённой шкалы приоритетов выбирается группой управления проектом разработки программного обеспечения с участием специалистов по операционной системе и инженеров проекта, занимающихся разработкой соответствующего процесса. За точку отсчета принимается приоритетное выполнение процесса обработки вызовов, как основной задачи системы коммутации.

Таблица 3.4. – Процессы и их приоритеты в ПО СР 113

Приоритет	Процессы/группы процессов	Доступность приоритета для изменения
15	Процесс операционной системы	Зарезервирован
14	Высокоприоритетный временной процесс для административного управления таймерами операционной системы, Высокоприоритетный процесс мониторов защиты (область защищённых адресов).	Зарезервирован
13	Прочие временные процессы для административного управления таймерами операционной системы. Высокоприоритетный процесс технического обслуживания.	Доступен
8 – 12	Процессы ввода/вывода. Высокоприоритетные процессы администрирования, технического обслуживания и эксплуатации (OA&M-процессы)	Доступен
7	Обработка вызовов (CALLP)	Зарезервирован
2 – 6	Низкоприоритетные OA&M-процессы	Доступен
1	Процессы ревизии	Доступен

При назначении приоритетов всем другим процессам необходимо ориентироваться прежде всего на обеспечение наивысшего приоритета процессу обработки вызовов. Чем выше приоритет процесса, тем меньше время ожидания процессом начала выполнения. С другой стороны, время, в течение которого процесс может заменяться (прерываться) другими высокоприоритетными процессами, также уменьшается.

По отношению к времени работы программного обеспечения между двумя последовательными полными перезапусками процессы могут существовать постоянно (циклические процессы) или создаваться динамически (запускаться), а затем снова уничтожаться средствами операционной системы

Циклические процессы загружаются в память резидентно/постоянно и запускаются один раз в ходе начальной загрузки программного обеспечения, в том числе и после полного перезапуска (перезагрузки) ПО. Циклические процессы запускаются операционной системой в определённом порядке, который определяет разработчик ПО. К циклическим процессам относятся высокоприоритетные системные процессы; процессы обработки вызовов также включаются в эту группу запуска.

После запуска циклические процессы сначала выполняют однократные предварительные задачи (инициализация), в ходе которых высокоприоритетные процессы создают условия для запуска и выполнения следующих по порядку процессов. Это продолжается до тех пор, пока, наконец, не будут созданы условия для запуска прикладных (пользовательских) процессов. В этот момент циклические процессы приостанавливают свое выполнение. Приостановка означает, что с момента готовности к запуску пользовательских процессов, прикладные процессы готовы выполнять текущие системные задачи и ожидают поступления

новых запросов. Циклические процессы обычно выполняют срочные задачи, которые требуется выполнять достаточно быстро.

Отметим, что технически инициализация или начальная загрузка ПО EWSD является критической операцией. Она осуществляется в момент запуска системы коммутации в эксплуатацию или в момент полной перезагрузки ПО EWSD. В случае полной перезагрузки ПО система коммутации с программным управлением временно (от 30 минут до 2 часов) теряет работоспособность. Полная перезагрузка может осуществляться, к примеру, после полного выключения электропитания на ЭАТС, что в производственной обстановке случается достаточно редко.

Динамические процессы запускаются, когда имеется соответствующая задача, например :

- отдельные задачи, нечасто выполняемые во время функционирования системы, например обработка ввода/вывода на внешние запоминающие устройства – НЖМД, НМЛ, НОД;
- задачи, некритичные к времени выполнения процесса, например, запрос персоналом состояния функциональных блоков CP113.

Результат запроса о состоянии функциональных блоков CP113 будет рассмотрен в подразделе 3.6.5.

Динамически запускаемым процессам назначают более низкие приоритеты, чем циклическим процессам. Они не находятся постоянно в оперативной памяти ОЗУ. Операционная система временно загружает динамические процессы для выполнения соответствующих задач и затем их запускает на исполнение. Для начала процедуры запуска прикладная программа (программа пользователя), которая инициировала процесс, сначала посылает операционной системе запрос на запуск. Как это было описано в подразделе 2.5 для ОС РВ QNX, по получении запроса на запуск, операционная система EWSD сначала переводит

процесс в состояние ready-to-start (готов к запуску), а затем в состояние ready-to-execute (готов к выполнению). В результате процесс ставится в очередь выполняемых процессов в соответствии с его приоритетом. Если в очереди нет выполняемых процессов более высокого приоритета, то выполняется данный процесс.

Продолжим рассмотрение структуры программного обеспечения СР113 на примере сетевого контроллера (устройства управления) системы сигнализации ОКС №7.

3.6.4 Распределение ресурсов главного процессора управления ОКС№7

Административное управление (планирование времени) выполнением процессов на основе приоритетов позволяет своевременно обрабатывать срочные задачи. Однако при этом снижается вероятность обработки в заданное время низкоприоритетных задач. Если операционная система постоянно занята выполнением высокоприоритетных процессов, то процессам с более низким приоритетом время на выполнение будет выделяться достаточно редко. Тем не менее, операционная система должна также выделять время и для выполнения низкоприоритетных процессов, даже в том случае, если количество высокоприоритетных процессов велико. Особенно важным является поставленная задача в системе управления ОКС №7 на примере системы EWSD v15, а именно в главном процессоре (main processor, MP). Для решения этой задачи в структуре программного обеспечения проектируются виртуальные CPU.

Виртуальный CPU определяет наименьшую величину вычислительной мощности/пропускной способности физического процессора MP. Этот показатель называется временным бюджетом CPU. В целом временной бюджет централизованно выделяется всем процессам, закрепленным за этим CPU. В процессе выполнения программы вре-

менной бюджет определяется планировщиком операционной системы. Виртуальный CPU может использоваться для нескольких блоков SPU и капсул, а также для нескольких программных оболочек. Отметим, что SPU может содержать процессы, которые принадлежат различным виртуальным CPU.

Для того чтобы обрабатывать процессы, имеющие одинаковый приоритет в рамках одного и того же виртуального CPU, используется процедура квантования времени. Если время выполнения процесса составило определенное количество миллисекунд, и при этом в системе имеются другие, готовые к выполнению процессы с тем же самым приоритетом, то выделение времени первому процессу приостанавливается. По этой причине процессы разрабатываются таким образом, чтобы имелась возможность прерывать их выполнение другими процессами в любом месте исполняемого машинного кода. Это верно даже в том случае, если процессы имеют одинаковый приоритет и принадлежат тому же самому виртуальному CPU.

Каждый виртуальный CPU имеет одну очередь процессов для каждого приоритета процесса. Все очереди обрабатываются в соответствии с принципом «первым пришел – первым обслужен» (first input – first out, FIFO).

Планировщик операционной системы в сетевом контроллере системы сигнализации ОКС№7 (signaling system network control, SSNC) работает в соответствии с описанным принципом квантования времени. В начале каждого интервала времени планировщик ОС вычисляет объем ресурсов, уже использованных каждым виртуальным CPU в процессе выполнения задач. Это значение сравнивается с запланированным на этапе проектирования ПО бюджетом этапа выполнения каждого виртуального CPU. В результате для каждого виртуального CPU рассчитывается индивидуальный временной кредит. Виртуальному CPU с самым высоким временным кредитом выделяется время на

выполнение. Если выбранному виртуальному CPU не требуется весь интервал времени, то остаток времени выделяется виртуальному CPU со следующим по величине временным кредитом. Этот механизм повторяется в начале следующего кванта времени. Для того, чтобы не снижать эффективность функционирования системы коммутации и поддерживать реальный режим времени, операционная система позволяет иметь максимум 16 виртуальных CPU.

3.6.5 Базы данных системы EWSD

В основу организации базы данных положены требования по эксплуатации коммутационной системы. С точки зрения содержания, все данные EWSD можно разделить на данные по обработке вызовов и административные данные.

Данные по обработке вызовов используются для выполнения процесса обработки вызовов, например для установления соединения. В этом случае должны быть доступны самые различные типы данных:

- данные о станционном оборудовании;
- данные о трансляции цифр набора номера;
- данные о телефонных номерах абонентов;
- абонентские данные и данные о подключенных УПАТС;
- информация о маршрутизации и данные о соединительных линиях
- данные о тарифах и данные о стоимости вызовов.

Также могут потребоваться следующие данные:

- данные управления сетью сигнализацией ОКС №7;
- данные по управлению сетью связи.

К ***административным данным*** относятся следующие типы данных:

- тестовые данные;
- данные о трафике;
- переменные станционные данные;
- данные об оборудовании системы;
- данные об аварийных сигналах.

Данные EWSD, содержащиеся в базе, делятся на три категории:

***Постоянные данные**,* которые описывают неизменные характеристики системы коммутации. Эти данные не изменяют своего значения в течение рассматриваемого интервала времени, например время функционирования текущей версии ПО. Постоянными данными являются данные начального запуска программного обеспечения управления. К этим данным относятся, например, максимальная длина таблиц данных. Доступ к постоянным данным осуществляется в режиме «только для чтения».

К ***полупостоянным данным*** относятся, например, абонентские данные или данные о портах, которые сравнительно редко изменяют своё значение в течение рассматриваемого интервала времени. Они, как правило, защищены от записи. Прикладные программы (например, программы обработки вызовов) могут обращаться к этим данным только в режиме доступа «только для чтения». При необходимости полупостоянные данные могут быть модифицированы оператором путем выполнения соответствующих заданий с помощью команд языка MML.

Переменные данные постоянно изменяют своё значение в течение рассматриваемого интервала времени; они описывают текущее состояние системы коммутации и действия по обработке вызовов. К переменным данным относятся, например, информация о рабочих состояниях аппаратных средств, сведения о статусе соединений и наличии свободных ресурсов. Кроме того, к переменным данным относятся результаты действий по обработке вызовов – данные о

стоимости вызовов, результаты измерения трафика. Переменные данные не защищены от записи. Чтение и модификация этих данных в основном осуществляется программами обработки вызовов.

Структура данных в базах данных процессора CP113c и контроллера системы сигнализации ОКС №7 реализована на основе индивидуально-определяемых модулей данных. Модуль данных обычно состоит из таблицы с данными, относящимися к какой- либо задаче или источнику, а также включает процедуру доступа для обращения к этим данным.

Процессор CP113c хранит в оперативной памяти CMY все полупостоянные данные и, в зависимости от выполняемых им задач, переменные данные базы данных. Процессор CP113 осуществляет управление перечисленными данными. Для обеспечения защиты и сохранности данных на НЖМД хранится актуальная копия всех полупостоянных данных.

В локальной памяти процессоров GP каждой LTG содержатся полупостоянные и переменные данные, относящиеся к задачам обработки вызовов, рабочим задачам и задачам обеспечения защиты данных. Полупостоянные данные LTG формируются из таблиц данных CP113c. Блок LTG получает полупостоянные данные во время загрузки из CP113c. В цифровых абонентских блоках (DLU) прежде всего содержатся полупостоянные данные об абонентских портах. В DLU эти данные загружаются процессорами линейных групп LTG. Небольшая часть данных DLU хранится в виде микропрограммного обеспечения. В контроллере MP (EWSD v15) или в CCNC ОКС №7 (EWSD v10) содержатся полупостоянные и переменные данные, необходимые для функционирования ОКС №7.

Размер и содержимое базы данных системы коммутации определяются целями проекта, режимом использования EWSD (око-

нечная станция, узловая станция, зоновый транзитный узел), требуемым набором услуг.

Таблицы базы данных в основной APS представлены в виде пустой структуры с минимальным размером. Для того, чтобы специфичные для АТСЭ данные могли быть введены в таблицы, эти таблицы должны быть расширены до требуемого размера. Для расширения таблиц данных используется специальный программный инструмент разработки – генератор станционных данных (ODAGEN). Эта специальная программа представляет собой часть программного обеспечения сетевого узла, позволяющая расширить функциональные возможности ПО EWSD. Программа ODAGEN расширяет размер таблиц (длину, количество строк) данных, хранящихся на НЖМД. В основу значений расширения отдельных таблиц данных положена проектируемая конфигурация системы. Проектируемые значения размеров таблиц вводятся в ODAGEN с помощью MML-команд. После этого расширенная основная APS загружается в CMY процессора CP113c. Теперь таблицы данных в общей памяти CMY имеют размер, требуемый для загрузки реальных станционных и абонентских данных. Для передачи данных в таблицы данных используются файлы MML-команд.

Рассмотрим пример информации, которую пользователь получает из базы данных системы EWSD в части станционных данных.

С помощью терминала технического обслуживания и эксплуатации инженером с терминала техобслуживания и эксплуатации вводится команда STAT SSP (языка MML). В ответ ПО управления генерирует ответ в виде шаблона на рис. 3.11. Эти данные касаются текущего технического состояния процессора CP113c. Рассмотрим расшифровку значений шаблона на рис. 3.11.

В целом рабочее состояние аппаратных компонентов CP113c определяется тремя атрибутами:

```
LXUNI/A39075D0290/RUSCPZ1V10340757/013          06-03-19  03:07:22
3563      OMT-01/SAMUNI      3080/02056
STATSSP;                                          EXEC'D
UNIT      OST      UNIT      OST      UNIT      OST
----------------   ----------------   ----------------
BAP-0     SPR      BAP-1     MAS      CAP-0     PLA
CAP-1     PLA      CAP-2     PLA      CAP-3     PLA
CAP-4     PLA      CAP-5     PLA      CAP-6     PLA
CAP-7     PLA      CAP-8     PLA      CAP-9     PLA
IOC-0     ACT      IOC-1     ACT      IOC-2     PLA
IOC-3     PLA      CMY-0     ACT      CMY-1     ACT
BCMY-0    ACT      BCMY-1    ACT
IOPMB-32  ACT      IOPMB-33  ACT      IOPMB-40  ACT
IOPMB-41  ACT      IOPMB-42  ACT      IOPMB-43  ACT
IOPUNI-0  ACT      IOPUNI-1  ACT      IOPTA-0   ACT
MDD-0     ACT      MDD-1     ACT      MTD-0     PLA
MOD-0     *MBL     OMT-0     *UNA     OMT-1     ACT
END JOB 3563
```

Рис. 3.11 – Переменные данные о рабочих состояниях аппаратных средств процессора CP113

Административное состояние, которое может изменяться персоналом станции. Административные состояния записываются в память CMY (полупостоянное хранение). На рис. 3.11 используется следующие обозначения технического состояние компонентов УУ: MBL (maintenance blocked) – заблокирован для технического обслуживания, т.е. в данном случае магнитооптический накопитель MOD готов только для запуска программы проверки и поиска неисправностей.

Состояние ошибки – явно указывает на то, что аппаратный модуль является неисправным. При этом отказы, при которых аппаратный компонент (модуль) находится в состоянии полного отказа, отличаются от отказов, при которых аппаратный компонент (модуль) все еще может функционировать. На рис. 3.11 терминал OMT-0 считается не готовым, UNA (unavailable) т.к. в момент ввода команды MML терминал OMT-0 был выключен.

Состояние системы – определяется функцией конфигурирования в соответствии с внутренними и внешними событиями. При необходимости, персонал с помощью команд языка MML может изменять состояние системы или отдельных аппаратных компонент. Состояние системы представляет собой критически важные переменные данные. На рис. 3.11 процессоры IOC-0, IOC-1, шины BCMY-0, BCMY-1, область памяти CMY-0, CMY-1, процессоры ввода/вывода буфера сообщений IOP:MB, IOP:UNI, IOP:TA, терминал OMT-1, НЖМД MDD-0, MDD-1 технически исправны и находятся в рабочем т.е. активном состоянии ACT (active).

Операционная система может штатно изменять рабочее состояние процессора BAP. Штатное (регламентное) переключение подразумевает изменение рабочего состояния двух процессоров BAP ведущий (master, MAS) – BAP резервный (spare, SPR) без какого-либо ухудшения обслуживания. При этом MAS меняется на SPR, а SPR меняется на MAS. Аварийное переключение происходит в случае отказа одного из BAP (автоматически переходит в состояние UNA) и приводит к присвоению резервному BAP статуса ведущего MAS, без нарушения в целом операций коммутации.

Следует отметить, что процессоры CAP0…CAP9 имеют состояние «запланирован» (planninig, PLA,). Это состояние означает физическую готовность CP113c к дооборудованию дополнительными процессорами, хотя в настоящее время эти процессоры физически не установлены и программно не активизированы.

3.6.6 Межпроцессный обмен

Рассмотрим далее связь между программными процессами. Коммуникация между процессами ПО EWSD осуществляется с помощью операционной системы, которая реализует асинхронную связь между различными элементами ПО. В процессе связи учитывается приоритет

того или иного процесса. Два программных процесса могут взаимодействовать между собой используя следующие способы обмена :

- межпроцессная коммуникация;
- удалённый вызов процедуры
- использование разделённых данных.

Рассмотрим каждый из этих способов подробнее. При межпроцессной коммуникации для посылки и приёма сообщений операционная система используются буферы приёма и буферы передачи. Буферы представляют собой области физической памяти, которые средствами операционной системы назначаются в качестве буферов записи (буфер передачи) и буферов чтения (буфер приёма). При межпроцессном обмена используются следующие примитивы [7] или элементарные процедуры операционной системы:

- Примитив SEND – активизирует асинхронный обмен сообщениями, при котором сообщение помещается в буфер передачи и далее контроль за передачей сообщения передаётся операционной системе. С помощью служебных программ передающий процесс может быть информирован о том, насколько доступен (заполнен) буфер приёма.
- Примитив CAST – активизирует асинхронный обмен сообщениями при котором передающий процесс не информирован о том, насколько доступен (заполнен) буфер приёма. Особенностью CAST является использование как в пределах одной системы так и при межсистемном обмене.
- Примитив RECEIVE – используется для выборки сообщения из заданного буфера.
- Примитив RECEIVE_CASE – используется для выборки сообщения из заданного буфера или из группы буферов.

Общая схема межпроцессной связи показана на рис. 3.12.

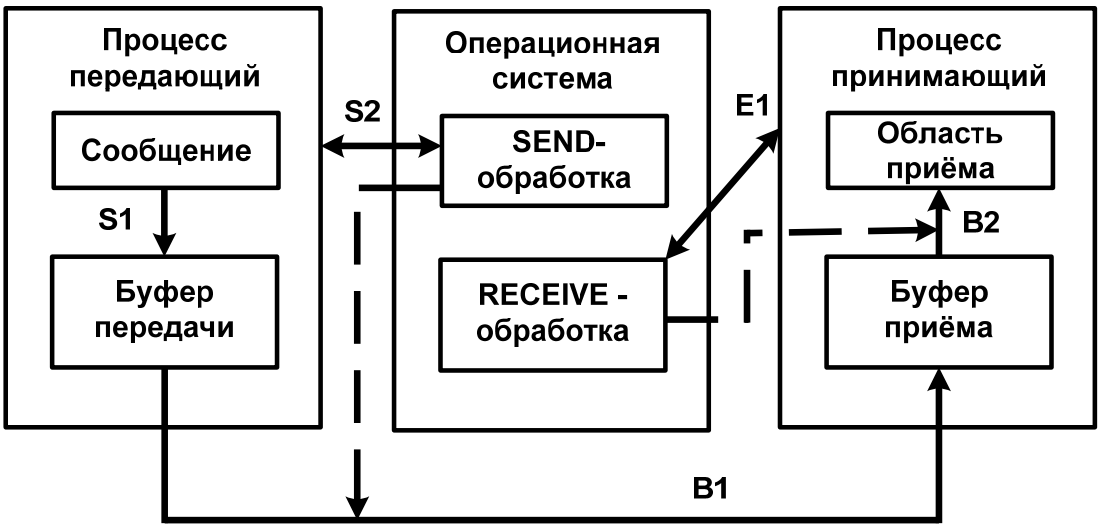


Рис. 3.12 – Принцип межпроцессной коммуникации

Рассмотрим процессы на рис. 3.12 :

Процесс S1: передающий процесс помещает передаваемое сообщение в буфер передачи.

Процесс S2: передающий процесс вызывает процедуру операционной системы SEND и далее передаёт управление обменом данной процедуре операционной системы.

Процесс B1: операционная система передаёт сообщение от буфера передачи к буферу приёма.

Процесс E1: принимающий процесс для управления приёмом вызывает процедуру операционной системы RECEIVE и тем самым передаёт управление приёмом операционной системе.

Процесс B2: операционная система передаёт сообщение из буфера приёма в область приёма, после чего сообщение становится доступным для принимающего процесса.

Рассматриваемая схема является упрощённой и не учитывает приоритета процессов, а также переключения между процессами в случае вовлечения в обмена более чем двух процессов.

Удалённый вызов процедуры предполагает синхронный обмен сообщениями без ограничений на место расположения вызываемой процедуры. Вызываемая процедура может находиться или в той же са-

мой системе коммутации, что и вызывающий (передающий) процесс или вызываемая процедура может находиться в другой системе коммутации.

Использование разделённых данных предполагает, что обмен сообщениями между параллельно запушенными процессами осуществляется с помощью физического разделения данных между этими процессами. Это предотвращает конфликты при обращении нескольких процессов к одним и тем же данным, особенно когда процессы используют единое адресное пространство.

3.7 Контрольные вопросы к главе 3

1. В чём состоит функциональное назначение процессора BAP?
2. Чему равна разрядность процессора BAP?
3. Для чего в составе процессора BAP имеются два блока обработки PU?
4. С какой целью при обработке данных генерируются биты ECC?
5. Чему равна разрядность шины доступа к общей памяти, как эта разрядность распределена между данными, адресами и сигналами управления ?
6. Каково функциональное назначение контроллера общей памяти CMYC?
7. Почему в составе общей памяти применяется несколько банков памяти ?
8. Какова процедура обработки однобитовой ошибки при вводе/выводе в общую память?
9. Как обрабатывается многобитовая ошибка при вводе/выводе в общую память?
10. Какие варианты (схемы) обеспечения надёжности управляющих комплексов систем коммутации существуют?
11. Назовите достоинства и недостатки общего резервирования.

12. Укажите достоинства схемы поэлементного резервирования.
13. Что такое схема резервирования «n+1»?
14. Какими методами и средствами обеспечивается надёжность программного обеспечения управления?
15. Почему процессы операционной системы имеют более высокий приоритет чем процессы обработки вызовов?
16. Что такое полупостоянные данные?

4. Специализированные процессоры в средствах связи

4.1 Сетевые процессоры в средствах связи

Сетевой процессор (network processor) – специализированное программируемое вычислительное средство, которое применяется для выполнения функций обработки потоков данных, пакетов и кадров, относящихся к различным телекоммуникационным протоколам в реальном режиме времени или с минимальной задержкой по времени. Особенности обработки пакетов или кадров согласно тому или иному сетевому протоколу определяются программным обеспечением, которое загружается в сетевой процессор. Для вычислений сетевой процессор использует ограниченное число инструкций (микрокоманд), достаточных для обработки данных с высокой скоростью

Основное назначение сетевого процессора – выполнять функции устройства управления сетевым трафиком при коммутации пакетов или кадров. Сетевой процессор является промежуточным звеном между физическим сетевым интерфейсом и матрицей коммутации в коммутаторе, многопротокольном маршрутизаторе, в межсетевом экране. Создание сетевого процессора является логическим продолжением работ по созданию аппаратных решение на основе интегральных схем, ориентированных на приложения ASIC.

Сетевой процессор состоит из одного или нескольких ЦПУ или машин обработки данных (engines, microengines), которые представляют собой самостоятельные вычислительные средства с памятью команд, счетчиками и регистрами. Эти вычислительные средства могут выполнять специализированные функции обработки пакетов данных. Поэтому машины обработки данных иногда называются «процессорами обработки пакетов», «дополнительными процессорами» [9] или протокольными процессорами (protocol processors) (см. рис. 4.1).

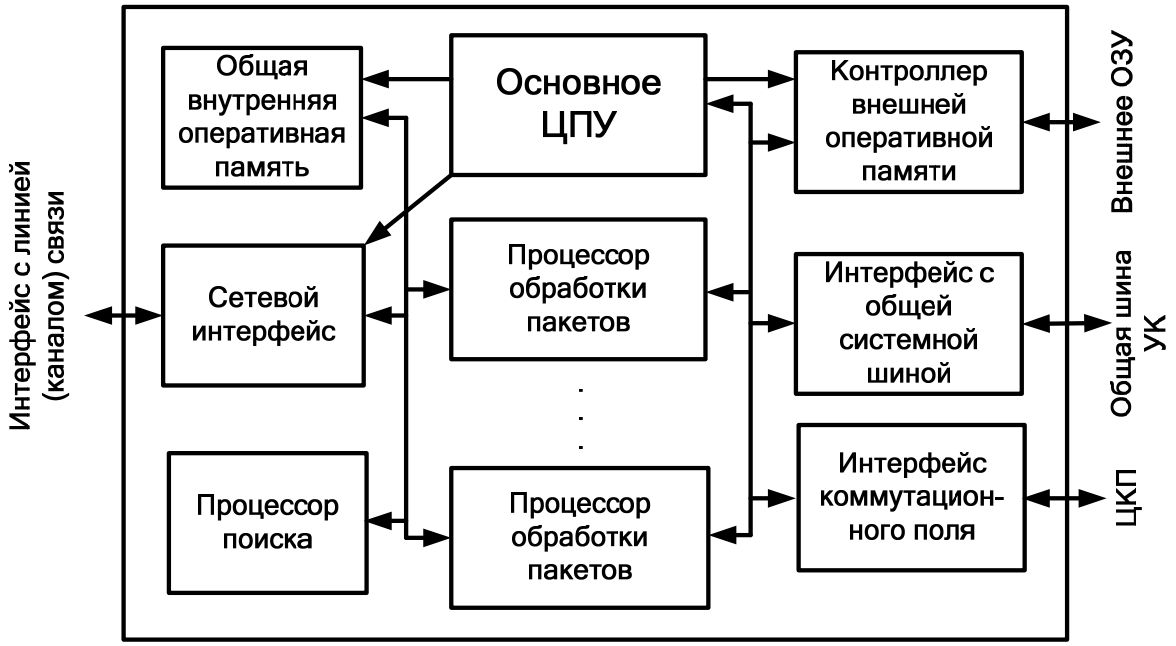


Рис. 4.1 – Функциональная блок-схема сетевого процессора

В дополнение к процессорам (контроллерам) обработки пакетов в состав сетевого процессора могут включаться :

- Основное ЦПУ, которое координирует работу всех остальных блоков сетевого процессора, поддерживает маршрутные таблицы и сведения о качестве обслуживания (QoS), обрабатывает пакеты, связанные с запросами по управлению сетью и обновлением маршрутных таблиц. Это ЦПУ делается на базе процессора общего назначения, на нём может запускаться такие ОС РВ как VxWorks, embedded Linux, а также средства разработки ПО для сетевого процессора.
- Аппаратные ускорители/сопроцессоры, разгружающие ЦПУ или процессоры обработки пакетов от таких функций, как вычисление контрольных сумм, поддержки режима DMA и прочее.
- Процессор поиска, который является самостоятельным ЦПУ и осуществляет поиск в таблицах маршрутизации; например по заголовку полученной IP-дейтаграммы осуществляет поиск в

таблице IP-адресов назначения следующего IP-узла, куда данная дейтаграмма будет передана.

Сетевой процессор может выполнять такие операции обработки данных, как уменьшение значения содержимого поля «время жизни» (Time-to-Live) для IP-дейтаграммы или повторное вычисление значения контрольной суммы CRC с помощью циклического избыточного кода.

В состав комплекта микросхем (chip set) для сетевого процессора входят блоки интерфейса с ОЗУ и интерфейсы с высокоскоростной общей системной шиной. Также в состав сетевого процессора может входить процессор оперативного управления (control processor). Этот процессор выполняет функции обработки пакетов, которые имеют более жесткие требования к времени и достоверности обработки, например контрольные пакеты, пакеты управления. Также процессор оперативного управления выполняет функции сбора статистики о работе сетевого процессора в целом. Как правило, встроенный процессор оперативного управления имеют сетевые процессоры с низкими требованиями к производительности. Для высокоскоростных сетевых процессоров характерен внешний процессор оперативного управления, который создаётся на базе процессора общего назначения и подключается к сетевому процессору по общей системной шине.

Основное ЦПУ сетевого процессора, а также процессоры обработки пакетов могут строиться как на базе архитектуры RISC-процессоров, так и на базе процессоров с длинным командным словом VLIW (very long instruction word), аналогичной формату команд в архитектуре с явным параллелизмом команд EPIC. Командное слово VLIW представляет собой блок (связку) машинных слов длиной 64 или 128 бит. Формат 128 бит включает в себя три машинных слова по 41 биту, поле шаблона длиной 5 бит для управления суперскалярной обработкой. В случае архитектуры с VLIW, каждый процессор обработки пакетов может выполнять отдельную функциональную задачу с набором

отдельных микрокоманд (инструкций). В случае использования RISC-архитектуры для процессоров обработки пакетов, можно поддерживать режим параллельной обработки информации.

В режиме обработки пакетов или кадров сетевой процессор может выполнять следующую последовательность процедур:

1. Получение через сетевой интерфейс пакетов, ячеек или кадров.
2. Полная или частичная запись полученных данных в общую внутреннюю оперативную память.
3. Определение порядка обработки пакетов/ячеек или кадров.
4. Собственно обработка, что включает в себя определение типа пакета, обработку данных заголовка пакета/кадра, определение данных маршрутизации, модификацию заголовка и присвоение требуемого класса обслуживания QoS.
5. На основании QoS пакет может быть временно задержан при передаче/обработке.
6. Перенаправление пакета или кадра на требуемое устройство ввода/вывода, например на интерфейс коммутационного поля или в буфер приёма-передачи.

В большинстве случаев для минимизации времени обработки и обеспечения высокой скорости обмена только п.п. 3,4,5 реализуются программно, остальные пункты – реализуются аппаратно.

Создание сетевых процессоров вызвано повышением сложности и ускорением процессов обработки информации в современных средствах связи. Например, в качестве примера требований к сетевому процессору, функционирующему на интерфейсе OC-192/10 Гбит/с, можно указать, что длительность временного интервала, достаточная для глубокой проверки пакета на скорости работы интерфейса 10 Гбит/сек, составляет всего 35 наносекунд. За это время сетевой процессор должен выполнить функции обработки входящих ячеек/пакетов, соответствую-

щие сетевому уровню (3-й уровень) согласно модели взаимосвязи открытых систем. Далее обработанные пакеты должны быть переданы устройствам управления в правильной последовательности, с требуемой скоростью и с нормативным качеством.

Одним из способов способом повышения производительности сетевых процессоров является физическое размещение сетевого процессора по принципу «ближе к каналу связи». Сетевые процессоры могут физически располагаться в разнообразных модулях соединительных линий или линий доступа, реализующих интерфейс средств связи с внешней средой. На этих модулях находится физическое окончание линии связи/тракта передачи и происходит обработка данных при вводе/выводе. Также здесь осуществляется добавление и удаление контрольных бит для защиты и контроля целостности данных. На модуле или блоке соединительных линий могут быть установлены ИУУ или ГУУ, реализующие функции управления соединением, маршрутизации пакетов, обработку сигнализации. Модуль может также содержать цифровой сигнальный процессор, чтобы поддерживать, к примеру, передачу речи через IP-сеть.

Сетевой процессор через общие системные шины или шины ввода-вывода может соединяться с процессором ввода/вывода, с процессором сетевой интерфейсной карты. Некоторые сетевые процессоры включают в свой состав контроллеры (микроконтроллеры) доступа, реализующие функции интерфейсов с внешней средой, при условии, что сетевой процессор компонуется непосредственно с ними на одном модуле (плате). В результате образуется устройство, которое называют «сетевым адаптером» (network adapter) или «портом» (port), однако в этих устройствах сетевые процессоры обрабатываются именно кадры, а обработку IP-пакетов выполняет сетевой процессор или центральное процессорное устройство. В настоящее время большинство сетевых

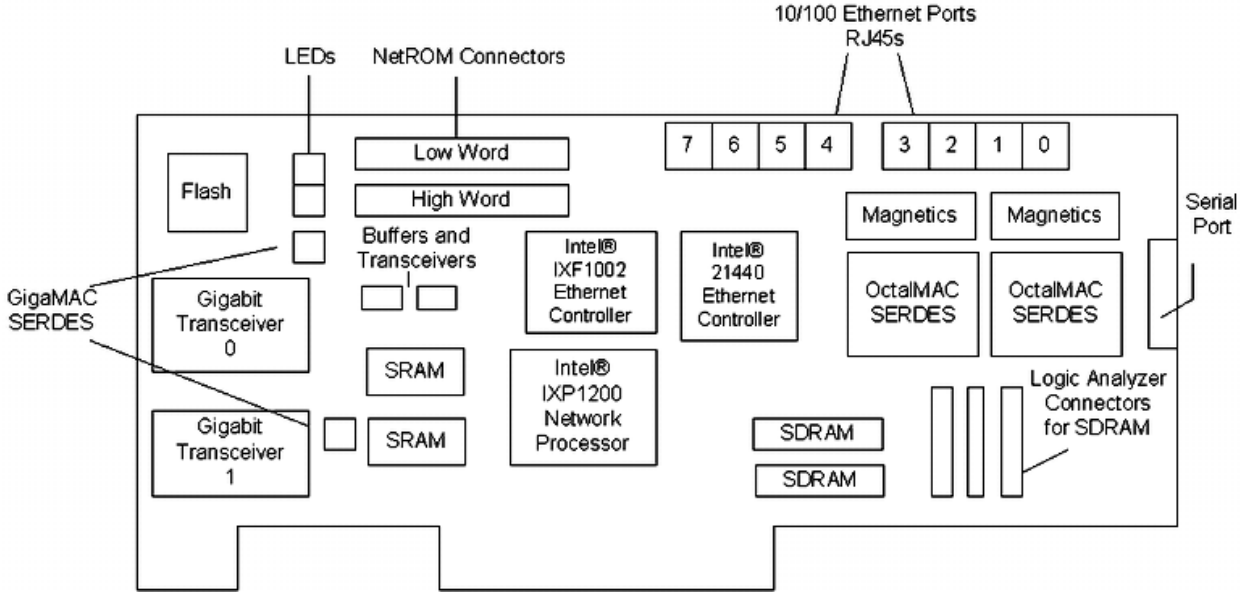
процессоров используют стандартные интерфейсы для выполнения функций обработки кадров на уровне 2 и 3 модели ВОС.

Процедура поиска, которую реализует процессор поиска, это одна из важнейших процедур, которую нужно учитывать при выборе типа сетевого процессора для применения в средствах связи. В первую очередь, необходимо рассмотреть число обращений к пакету, который будет обрабатываться. Тогда можно будет вычислить количество запросов в секунду, которое будет необходимо для корректной обработки пакетов минимального размера. Большинство сетевых процессоров обеспечивают поддержку таблиц поиска, а в характеристиках указывается число запросов в секунду, которые они могут выполнить. Далее рассматривается тип запроса и размер таблицы поиска, на основании чего делается вывод о глубине и продолжительности поиска и, соответственно, о времени необходимом для этих процедур.

В составе управляющего комплекса сетевые процессоры могут выполнять функции ГУУ, включая распределение ресурсов, диагностику и управление последствиями отказов. В частности, сетевой процессор Intel IXP1200, выполняет функции обработки, преобразования и маршрутизации пакетов данных, передаваемых по сетям связи. Для этого IXP1200 объединил в себе два вычислительных компонента: встроенный микропроцессор, выполняющий функции основного ЦПУ; дополнительные микропроцессорные ядра, выполняющие функции обработки пакетов. Тактовая частота основного ЦПУ и процессоров обработки пакетов составляет 232 МГц, поддерживается передача учетверённых слов данных. Встроенный 32-х разрядный микропроцессор в составе IXP1200 служит для выполнения задач по управлению IP-сетью. Одновременно шесть программируемых микроконтроллеров (дополнительные микропроцессорные ядра) ведут многопоточную обработку данных, передаваемых по сети [9]. В течение одного машинного цикла одновременно могут выполняться семь различных сетевых за-

дач, а 18 других задач ставятся в очередь на выполнение. При этом микроконтроллеры допускают возможность перепрограммирования с целью оптимизации обработки данных для различных приложений.

Каждый процессор IXP1200 способен обрабатывать 3 миллиона пакетов в секунду, что соответствует скорости 1,5 Гбит/сек. Физическая компоновка (размещение) сетевого процессора IXP 1200 на модуле (плате) показано на рис. 4.2. Компоновка на рис. 4.2 входит в состав архитектуры XP1200 Ethernet Evaluation Board.



Условные обозначения :

Buffers – буферы приёма-передачи,
Connectors – переключатели
Ethernet Controller – контроллер Ethernet
Flash – постоянная флэш-память для данных и команд
Gigabit Transceiver – передатчик со скорость 1 Гбит/сек
LED – светоиндикаторы
Magnetics – ферромагнетики
SDRAM – синхронная динамическая память с произвольным доступом
Serial Port – последовательный порт
SRAM – статическая память с произвольным доступом

Рис. 4.2 – Компоновка процессора IXP 1200 на модуле [9]

Эта архитектура применяется для целей тестирования и разработки программного обеспечения, проверки и контроля электромагнитных и механических характеристик, определения состава компонентов микросхемного набора (chipset) сетевого процессора IXP 1200.

Следует обратить внимание на наличие в составе микросхемного набора таких компонентов, как :

- SerDes – блок преобразования байта данных или слова данных для передачи по последовательному интерфейсу данных (Serial/Deserializer) в обоих направлениях т.е. последовательно-параллельное и параллельно-последовательное преобразование
- Octal MAC – микросхема для реализации функций контроллера доступа к среде передачи по стандарту IEEE 802.3 (порт Ethernet в режиме дуплекс или полудуплекс) с размером кадра до 1536 байт, с расширением до 64 кбайт. Также поддерживает статистические счётчики, протоколы управления RMON, SNMP.

Другой сетевой процессор, IBM PowerNP, представляет собой многопроцессорную систему, включающая 16 процессоров обработки пакетов, 7 специализированный сопроцессоров, обеспечивающих аппаратную поддержку ускорения вычислений; процессор основного ЦПУ на базе PowerPC. Эта система поддерживает обработку и передачу пакетов пассивных оптических сетей и локальных вычислительных сетей по технологии Gigabit Ethernet со скоростью передачи до 2,5 Гбит на уровнях 2–5 модели ВОС. Также в состав IBM PowerNP входят периферийные интерфейсы. Каждая пара процессоров обработки пакетов совместно использует специализированный аппаратный сопроцессор. Процессор обработки пакетов имеет трёхстадийный конвейер. Один из них выполняет обработку таблиц поиска, два других – функцию обработки кадров Ethernet и осуществление связи с другими сетевыми устройствами обработки данных. Семь сопроцессоров выполняют следующие функции:

- Обеспечение буферизации кадров для поддержки режима DMA при работе с оперативной памятью.
- Расчет контрольных сумм заголовков пакетов.

- Предоставление всем процессорам обработки пакетов доступа к внутренним регистрам, счётчикам и оперативной памяти.
- Обеспечение высокоскоростной пересылки данных между процессорами обработки пакетов.
- Управление обновлением значения счетчиков для процессорами обработки пакетов.
- Контроль информации по управлению потоками данных на соответствие предварительно назначенной скорости обработки и передачи.

Тактовая частота работы сетевого процессора IBM PowerNP составляет 133 МГц, потребляемая мощность 20 Вт.

Ещё одним примером сетевого процессора является цифровой терминал тракта сигнализации (signaling link terminal digital, SILTD) сигнализации ОКС №7 в системе EWSD v10. SILTD является аппаратным модулем в состав устройства управления ОКС №7 CCNC и в состав EWSD v10 может входить до 254 устройств SILTD. SILTD обеспечивает управление функциями уровня 2 системы сигнализации ОКС №7.

Устройство SILTD на рис. 4.2. выполняет следующие задачи :

- последовательная передача сигнальных сообщений в подчиненный мультиплексор MUXS и прием сообщений из него по цифровым каналам;
- установка скорости передачи 54 Кбит/сек (по стандарту ANSI) или 64 Кбит/сек (по стандарту ETSI) при цифровом режиме работы;
- последовательная передача сигнальных сообщений в модем по аналоговому каналу со скоростью передачи 4,8 Кбит/сек;
- последовательная передача и прием тестовых последовательностей из/в CCNP и CP113;
- ввод тракта сигнализации ОКС №7 в работу после устранения неисправностей.

Рис. 4.3 – Схема функционального блока SILTD

Блок памяти SILTD состоит из ССПЗУ и ЗУПВ. В ССПЗУ содержатся программы восстановления и начальной загрузки; в ЗУПВ содержатся переменные данные.

В составе SILTD применяется специальное передаточное оперативное запоминающее устройство (память передачи) ЗУПВ, которая передает сообщения между SILTD и контроллером терминалов звена сигнализации (signaling link terminal control, SILTC). Эта память позволяет осуществить независимый от времени доступ к ЦПУ SILTD со стороны контроллеров SILTC. Контроллер SILTC контролирует и обслуживает до восьми единиц SILTD. Контроллер SILTC осуществляет распределение сигнальных сообщений, поступающих из периферийного адаптера сигнализации (signaling periphery adapter, SIPA) к соответствующим терминалам SILTD. Контроллер SILTC выбирает сообщения из SILTD и передает их для дальнейшей обработки в ГУУ CCNC. Для реализации описанных функций, средства SILTC содержат следующие функциональные компоненты:

- ИУУ (процессор управления SILTC) – работает в режиме с минимальной нагрузкой, поскольку доступ к памяти осуществляется с помощью DMA.
- Контроллер прерываний – принимает запрос на прерывание от SILTD и вызывает соответствующую сервисную программу обработки данных.
- Программируемый контроллер DMA – выбирает данные из блоков ЗУПВ контроллеров SILTC и распределяет их одному из двух последовательных интерфейсов с процессором сети сигнализации по общему каналу (common channel signaling network processor, CCNP). В противоположном направлении контроллер DMA считывает данные из контроллера HDLC и записывает их в ЗУПВ. В SILTC используются два контроллера DMA.
- Таймер – вырабатывает тактовый сигнал для вызова программы синхронизации. Частота и тип выходных импульсов определяется программным обеспечением.

- Блок памяти SILTC состоит из ЭСППЗУ и ЗУПВ. ЭСППЗУ хранит программу восстановления и текущего контроля и все программы обработки. В ЗУПВ содержатся переменные данные.
- Для обмена сообщениями между SILTC и подключенными SILTD используется шинный интерфейс с системой шин для шины терминала звена сигнализации (B:SILT).

Система программ управления SILTD состоит из следующих комплексов программ:

Комплекс программ «Reception Part» осуществляет проверку очередности принятых сигнальных сообщений, а также очередность подтверждений на сообщения, которые были переданы SILTD – партнеру. Правильно принятые сообщения направляются далее к устройству CCNP, для обработки на уровне 3 (функции управления сетью сигнализации ОКС №7). Кроме того, данный комплекс программ информирует уровень 3 подсистемы обработки сообщений ОКС №7 об изменениях состояния каналов передачи сигнальной информации, обновляет счетчики повторной передачи, контролирует период ожидания для подтверждения (положительного или отрицательного), запускает повторение передачи сигнальных единиц (при необходимости), делает запросы на повторение передачи правильно принятых сигнальных единиц.

Комплекс программ «Transmission Part» подготавливает сигнальное сообщение для передачи. С этой целью выбирается адрес запоминающего устройства для передачи сообщений, номер очередности сообщения, значение бит-индикаторов.

Комплекс программ «Link State Part изменяет состояние тракта сигнализации и запускает пакет программ «Retrieval Part».

Комплекс программ «Retrieval Part» возвращение на уровень 3 все еще не переданные или еще не подтвержденные сигнальные сообщения.

Комплекс программ «Congestion Part» применяется при перегрузке приемника сигнальных сообщений в SILTD; этот комплекс программ периодически осуществляет передачу сигнальной единицы состояния звена «Signaling Information Busy» – «Переполнение сигнальной информацией» в противоположную/корреспондирующую систему коммутации (сигнальный пункт). Эта сигнальная единица передается до тех пор, пока состояние занятости приемника сигнальных сообщений в SILTD сохраняется. На входящей станции (сигнальном пункте) EWSD в момент принятия сигнальной единицы занятости комплекс программ «Congestion Part» деактивизирует текущий контроль времени ожидания подтверждения до максимальной продолжительности последнего. Таким образом, передача сигнальной единицы не повторится.

Комплекс программ подсистемы техобслуживания «Maintenance» осуществляет анализ ошибок, восстановление, тест сигнального тракта, рутинное блока SILTD.

Комплекс программ подсистемы администрирования «Administration» осуществляет запись измеренных значений сигнальной нагрузки и текущий контроль пороговых значений указанной нагрузки.

Как видно из примера, комплекс программ управления SILTD в общем соответствует составу главной программы управления средства связи в целом.

4.2 Процессоры цифровой обработки сигналов

Процессоры цифровой обработки сигналов, ПЦОС (DSP, digital signal processing) или цифровые сигнальные процессоры, ЦСП находят самое широкое применение в телекоммуникациях при решении большого круга задач, в частности для подавления помех, адаптивной импульсно-кодовой модуляции и иных видов кодирования сигналов, для мультиплексирования каналов, DTMF кодирования/декодирования, шифрования данных. При этом ПЦОС решают математические задачи

свертки, корреляционного анализа, преобразование Гильберта, быстрого преобразования Фурье, решения задачи адаптивной фильтрации, взвешивание сигналов, синтез сигналов.

При использовании цифровой обработки сигналов в телекоммуникациях аналоговая звуковая или видео–информация на входе при помощи аналого-цифрового преобразователя переводится в цифровую форму, затем полученный цифровой сигнал передается по цифровой линии связи, а на выходе – восстанавливается исходный сигнал. Типовая структурная схема устройства цифровой обработки сигналов (без канала связи для передачи) приведена на рисунке 4.4.

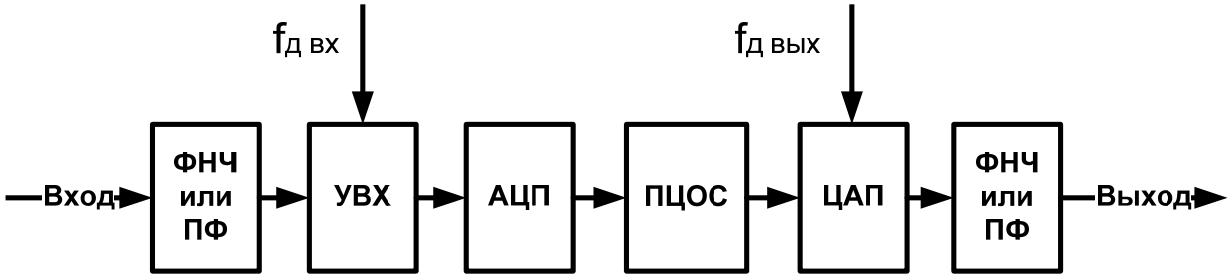


Рисунок 4.4. – Структурная схема устройства цифровой обработки сигналов.

Представленное на рисунке 4.4 цифровое устройство должно работать в реальном масштабе времени. В составе выделяется устройство выборки и хранения (УВХ), которое также называют дискретизатором по времени, оно непрерывно стробирует аналоговый сигнал с частотой, равной частоте дискретизации входного сигнала $f_{д\ вх}$, Гц. Аналого-цифровой преобразователь (АЦП), который также называют квантователем или дискретизатором по уровню, выдает новый цифровой отсчет сигнала для дальнейшей обработки ПЦОС. В качестве ПЦОС может служить программируемая логическая схема, собственно сигнальный процессор или микроконтроллер. Для обеспечения работы системы в реальном масштабе времени, ПЦОС должен закончить все вычисления в пределах интервала дискретизации $1/f_{д\ вх}$ и передать вы-

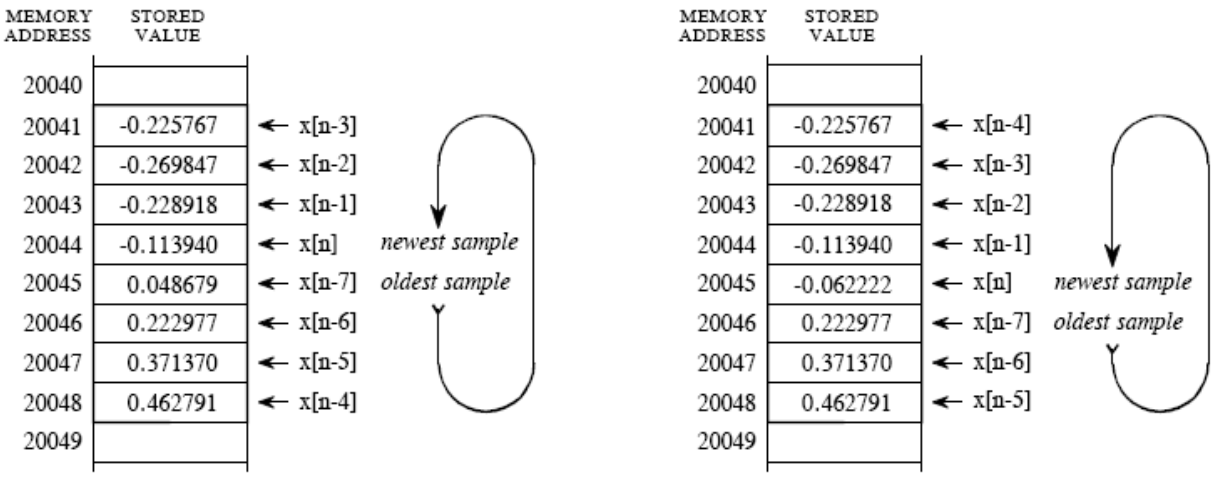
ходной отсчет сигнала на выходной цифро-аналоговый преобразователь (ЦАП) до момента поступления следующего отсчета с аналого-цифрового преобразователя. На выходе схемы выполняется обратное преобразование ЦАП с частотой дискретизации $f_{д\ вых}$,Гц. В качестве примера устройства цифровой обработки сигнала, работающего по такому принципу, может выступать цифровой фильтр. Возможен и другой принцип построения цифровых устройств реального времени. Например, при реализации на сигнальном процессоре алгоритма быстрого преобразования Фурье (БПФ), накопленный блок входных данных целиком загружается в оперативную память ПЦОС. В этом случае для обеспечения работы системы цифровой обработки сигнала в реальном масштабе времени необходимо, чтобы пока ПЦОС выполняет алгоритм БПФ над ранее полученным блоком данных, в его внутренней памяти накапливался новый блок данных. Таким образом ПЦОС должен успеть вычислить спектр входного сигнала в течение интервала накопления следующего блока данных, чтобы быть готовым обработать новый блок данных.

Итак, основное назначение ПЦОС – выполнение вычислительных операций (сложение и умножение) при обработки цифровой информации. На практике ПЦОС применяются в системах анализа сигналов, при реализации кодеков или кодеров различного назначения. Например, ПЦОС применяются в системах сотовой связи стандартов GSM, CDMA для осуществления сжатия на приёме и восстановления на передаче исходного аналогового речевого сигнала. Поэтому ПЦОС работают в реальном масштабе времени и имеют предсказуемое время исполнения четко алгоритмизированных задач.

Обработка сигналов ПЦОС выполняется непрерывно, пока имеется входной сигнал. В случае, если выполняется обработка ограниченного числа отсчётов входного сигнала, например восьми отсчётов, то значения этих восьми отсчётов постоянно хранятся в памяти ПЦОС и

непрерывно обновляются, поскольку на входе с частотой $f_{д\ вх}$, появляются новые отсчёты. Лучшим способом управлять этими хранящимися и постоянно обновляемыми отсчётами является круговая буферизация.

Рисунок 4.5 иллюстрирует восьмиотсчётный круговой буфер. Этот круговой буфер размещён в восьми последовательных ячейках оперативной памяти, с 20041 по 20048.



а *б*

Условные обозначения:

Memory address – адрес физической памяти
Stored value – хранимая величина
Newest sample – новое значение величины
Oldest sample – предыдущее значение величины

Рис. 4.5 а,б – Пример применения кругового буфера [34]

Рисунок 4.5 а) показывает восемь входных отсчётов x[n], сохраненных в данный момент времени, тогда как рис. 4.5 б) показывает изменения после появления нового отсчёта на входе. Идея круговой буферизации состоит в том, что конец массива данных с отсчётами соединён с его началом. Ячейка ЗУ (ЯЗУ) с адресом 20041 рассматривается как следующая за ячейкой ЗУ с адресом 20048, также, как ЯЗУ с адресом 20044 – рассматривается как следующая за 20045. За массивом следят с помощью ***указателя*** – переменной, значение которой является адресом ячейки памяти, указывающим, где находится самый

последний обрабатываемый отсчёт. Например, на рис. 4.5, а) указатель содержит адрес 20044, в то время как на рис. 4.5, б) он содержит адрес 20045. Когда появляется новый отсчёт, он заменяет самый «старый» отсчёт в массиве, и указатель перемещается на один адрес вперед. Круговые буферы эффективны, потому что когда появляется новый отсчёт, должно быть изменено только одно значение – значение указателя.

Чтобы управлять круговым буфером, необходимы четыре параметра. Первый – указатель, который указывает начало кругового буфера в памяти (в этом примере, 20041). Второй – указатель на конец массива (например, 20048), или переменная, которая содержит его длину (например, 8). Третий параметр – шаг адресации памяти. На рис. 4.5 размер шага равен единице, например: ячейка с адресом 20043 содержит один отсчёт, ячейка с адресом 20044 содержит следующий отсчёт, и так далее. Часто значение шага исключительно важно. Например, адресация может относиться к байтам, и каждый отсчёт может требовать двух или четырех байт, чтобы вместить его значение. Соответственно, размер шага должен быть равен двум или четырём. Эти три параметра определяют размер и конфигурацию кругового буфера, и не будут меняться в течение выполнения программы. Четвертый параметр – указатель на самый последний отсчёт, должен изменяться с каждым появлением нового отсчёта. Другими словами, должна использоваться программируемая логика, которая контролирует, как обновляется значение четвертого параметра, основываясь на значении первых трех параметров. Так как эта логика весьма проста, она должна быть очень быстродействующей. ПЦОС должны быть оптимизированы для применения управляющих круговых буферов так, чтобы достичь самой высокой из возможных скоростей работы.

Пусть для каждого нового отсчёта в схеме на рис. 4.5, ПЦОС должен выполнить следующую операцию :

$$y[n] = a_0 x[n] + a_1 x[n-1] + a_2 x[n-2] + a_3 x[n-3] + a_4 x[n-4] + \cdots$$

где

$y\ (n)$ – значение n-го сигнала на выходе;

a_0, a_1, a_2 – коэффициенты преобразования;

$x\ [n]$ – значение n-го отсчёта входного сигнала.

Коэффициенты преобразования, по аналогии с входными отсчётами, записываются в круговой буфер коэффициентов.

Для выполнения описываемой операции, ПЦОС выполняет следующую последовательность действий:

1. Получает значение отсчёта с аналогово-цифрового преобразователя и создает прерывание.
2. Загружает из оперативной памяти программ и запускает на исполнение требуемую (под)программу по прерыванию.
3. Записывает значение отсчёта в круговой буфер входного сигнала.
4. Обновляет указатель для кругового буфера входного сигнала.
5. Устанавливает в ноль регистр–аккумулятор (при наличии регистра–аккумулятора) или рабочий регистр.
6. Управляет программным циклом обработки отсчёта для каждого из коэффициентов.
7. Выбирает коэффициент из кругового буфера коэффициентов.
8. Обновляет указатель для кругового буфера коэффициентов.
9. Выбирает отсчёт из кругового буфера входного сигнала.
10. Обновляет указатель для кругового буфера входного сигнала.
11. Умножает коэффициент на отсчёт.
12. Добавляет произведение в регистр–аккумулятор или в рабочий регистр.
13. Перемещает результат – выходной отсчёт из аккумулятора или регистр в имеющийся буфер приёма/передачи.

14. Перемещает выходной отсчёт из буфера приёма/передачи в цифро-аналоговый преобразователь.

Перечисленные действия надо выполнять максимально быстро. Так как операции 6–12 будут повторяются многократно (один раз для каждого коэффициента), этим операциям нужно уделить особое внимание. Традиционные микропроцессоры выполняют эти 14 действий *последовательно* (один за другим), тогда как ПЦОС для ускорения вычислений должны выполнять их *параллельно*. В некоторых случаях, все действия в цикле (операции 6-12) могут быть завершены в отдельном такте. Для обеспечения возможности ускоренной обработки данных ПЦОС имеют гарвардскую архитектуру, как это показано на рис. 1.3. Следует отметить, что для повышения производительности некоторые ЦСП используют модифицированную или супегарвардскую архитектуру (SHARC, super harvad architecture computer), которая допускает обмен содержимым между памятью программ и памятью данных, что расширяет возможности устройства. Рисунок 4.6 иллюстрирует особенности супергарвардской архитектуры.

Термин «супергарвардская архитектура» был введён фирмой Analog Devices, чтобы описать особенности ПЦОС типа ADSP-2106X и типа ADSP-211XX. Идея состоит в том, чтобы усовершенствовать гарвардскую архитектуру, добавляя некоторые конструктивные особенности для улучшения производительности. Для этого в составе ПЦОС с архитектурой SHARC появились два достаточно важных компонента: кэш команд и контроллер ввода – вывода. Важность этой «добавки» обусловлена тем, что алгоритмы цифровой обработки сигналов вообще тратят большую часть времени на циклы, например такие, как операции 6-12, рассмотренные выше. Это значит, что один и тот же набор команд будет постоянно передаваться из памяти команд в ЦПУ. В этом случае супергарвардская архитектура обладает преимуществом, а именно – наличием кэша команд.

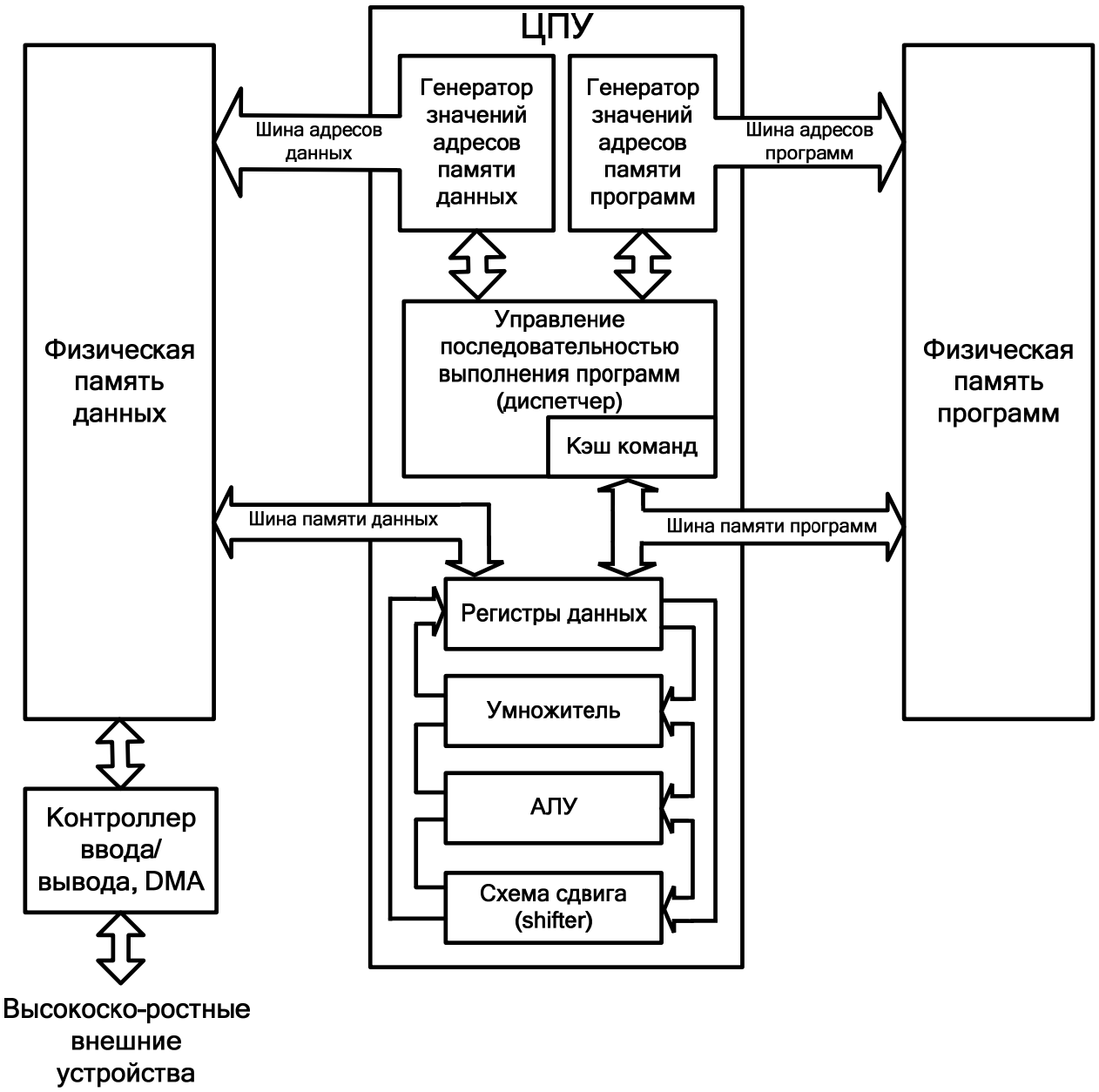


Рис. 4.6 – Структурная блок-схема SHARC ПЦОС

Этот кэш содержит, например, 32 самые последние по времени исполнения команды. Исполняясь впервые в рамках цикла, команды должны передаваться по шине команд. Это приводит к замедлению скорости обмена на участке ЦПУ–память команд из-за конфликта с коэффициентами, которые также должны передаваться по этому пути. Однако, при дальнейшем выполнении цикла, команды можно считывать из кэша команд, не загружая при этом шину памяти команд. Это значит, что все передачи информации из памяти в ЦПУ могут быть выполнены

в единственном цикле: отсчёт входного сигнала поступает по шине данных, коэффициент передаётся по шине команд, а команда поступает из кэша команд.

Контроллер ввода – вывода на рис. 4.6 связан с памятью данных. Здесь предусмотрены высокоскоростные последовательные и параллельные порты связи. Например, при тактовой частоте 40 МГц имеется два последовательных порта, которые действуют каждый со скоростью в 40 Мбит/с, в то время как шесть параллельных портов, каждый из которых обеспечивает передачу данных со скоростью 40 Мбит/с, совокупно на скорости 240 Мбит/с.

Специальное аппаратное обеспечение позволяет передавать эти потоки данных непосредственно в память (прямой доступ к памяти, или DMA), без необходимости передавать их в регистры ЦПУ. Другими словами, операции 1 и 14 рассмотренные выше, происходят независимо и одновременно с другими действиями, при этом циклы ЦПУ не используются. Основные шины (шина команд и шина данных) доступны также снаружи микросхемы ПЦОС через контакты на корпусе, что обеспечивет дополнительный интерфейс к внешней памяти и внешним устройствам. Для повышения производительности также используют конвейерную организацию вычислений, которая подробно рассматривается в главе 5.

Для ПЦОС, чья упрощенная схема приведена на рис. 4.6, характерным является наличие аппаратного умножителя (multiplier), позволяющего выполнять умножение двух чисел за один командный такт. В процессорах общего назначения умножение обычно реализуется за несколько тактов, как последовательность операций сдвига и сложения. Также в ПЦОС используются специальные команды как, например, умножение с накоплением (MAC): C = A x B + C, с указанным в команде числом выполнений в цикле и с правилом изменения индексов используемых элементов массивов A и B. Применяются инверсия битов адре-

са, разнообразные битовые операции и сдвиги (shifter). В ПЦОС реализуется аппаратная поддержка программных циклов и кольцевых буферов, когда один или несколько операндов извлекаются из памяти в цикле исполнения команды.

Генераторы значений адресов управляют адресами, посылаемыми в память команд и данных, определяя, где информация должна считываться, а где записываться. Это избавляет от необходимости использовать такты ЦПУ, чтобы следить за тем, как хранятся данные. Например, в DSP SHARC, каждый из двух генераторов может контролировать по восемь круговых буферов.

При математической обработке данных используются :

- умножитель (устройство умножения);
- арифметико-логическое устройство (АЛУ),
- схема сдвига (многорегистровая схема циклического сдвига).

Устройство умножения считывает значения операндов из двух регистров данных, перемножает их и помещает результат в другой регистр. АЛУ выполняет сложение, вычитание, взятие по модулю, логические операции (И, ИЛИ, НЕ), преобразование между форматами с фиксированной и плавающей точкой и аналогичные функции. Элементарные двоичные операции выполняются схемой циклического сдвига, такие как сдвиг, циклический сдвиг, извлечение и добавление и т.д. Важной особенностью процессоров семейства SHARC является то, что можно обращаться параллельно к устройству умножения и к АЛУ. За один такт данные от одних регистров данных могут передаваться к устройству умножения, а данные от других регистров передаются в АЛУ, а результаты результата можно записать в любой из оставшихся свободных регистров.

В ПЦОС широко используются методы сокращения длительности выполнения команд за счёт использования в ЦПУ с RISC-архитектурой.

В целом ПЦОС можно разделить на ПЦОС с обработкой данных в формате с фиксированной точкой и более дорогие ПЦОС, аппаратно поддерживающие операции над данными в формате с плавающей точкой. Использование данных в формате с плавающей точкой обусловлено необходимостью обеспечить повышенную точность вычислений при интегральных и дифференциальных преобразованиях. Здесь точность обеспечивается экспоненциальным форматом представления данных. ПЦОС с плавающей точкой обычно используют минимум 32 бита для хранения каждой величины. Это даёт намного больше битовых комбинаций, чем для фиксированной точки т.к. 2^{32} = 4 294 967 296. Главная особенность обозначения с плавающей точкой – это то, что представляемые числа распределены *не* равномерно. В обычном формате стандарта ANSI/IEEE Std. 754-1985 самое большое и самое маленькое числа соответственно $\pm 3.4\times10^{38}$ и $\pm 1.2\times10^{-38}$, соответственно. Здесь значения чисел с плавающей точкой достаточно неравномерно распределены между этими двумя крайними значениями. В результате разница между любыми двумя числами примерно в десять миллионов раз меньше, чем абсолютное значение чисел. Как следствие, получаются большие промежутки между большими числами, но малые промежутки между малыми числами.

Процедуры компрессии, декомпрессии, адаптивной фильтрации связаны с определением логарифмических зависимостей, результаты которой существенно зависят от точности представления данных в широком динамическом диапазоне. Работа с данными в формате с плавающей точкой существенно упрощает и ускоряет обработку, повышает точность и достоверность, поскольку не требует выполнения операций округления и нормализации данных, отслеживания ситуаций потери значимости и переполнения. Однако для операций с плавающей точкой необходимы функционально более сложные устройства для чего не-

обходимо использовать более сложные технологии производства микросхем, что приводит у удорожанию изделия в целом.

Особенностью конструктивного исполнения ПЦОС является :

1. Наличие энергонезависимой выделенной оперативной памяти для хранения программ с возможностью перепрограммирования.
2. Наличие энергозависимой выделенной оперативной памяти для хранения данных.
3. Наличие функциональных блоков, которые выполняют только операцию умножения и только операцию сдвига.

Особенностью ПЦОС является и наличие аккумуляторов повышенной емкости, в частности для 32-разрядного ПЦОС разрядность аккумулятора может составлять до 80 бит. Это обусловлено необходимостью уменьшить ошибку округления, связанную с многократными математическими операциями с фиксированной точкой. Кроме того, в ПЦОС применяются теневые регистры для хранения результатов вычисления. ***Теневой регистр*** – это сдвоенный регистр, содержимое одного из них может быть переписано в другой регистр в процессе дублирования или синхронизации содержимого за один такт. В результате, в случае обработки прерываний можно быстро сохранить текущие значения внутренних регистров в один из теневых регистров, а потом практически мгновенно восстановить текущее состояние всех внутренних регистров за минимальное число тактов. Напротив, в МПр общего назначения содержимое каждого регистра в случае прерывания по очереди переписывается в стек, за каждый такт – по одному регистру, что снижает быстродействие микропроцессорной системы.

Итак, благодаря особенностям конструкции, всего за один машинный цикл SHARC ПЦОС на рис. 4.6 может выполнить умножение (действие 11), сложение (действие 12), два перемещения данных (действия 7 и 9), обновление двух указателей круговых буферов (действия 8 и 10),

управление циклом (действие 6). Потребуются дополнительные такты, связанные с началом и окончанием цикла (действия 3, 4, 5 и 13, плюс перемещение начальных значений), однако, это существенно не снижает эффективности ПЦОС.

Программы для ПЦОС разрабатываются на двух языках программирования: Ассемблер и Си. Соотношение «Ассемблер к Си» по количеству разработчиков примерно как 1 к 10. Язык Си по сравнению с Ассемблером требует больше памяти для хранения команд, что несколько увеличивает размеры и стоимость ПЦОС, однако программы, написанные на Си, могут быть более переносимыми, чем программы, написанные на Ассемблере.

В числе наиболее распространенных ПЦОС можно назвать изделия следующих компаний – Motorola (серия 56002, 96002), Intel (серия i960), Texas Instruments Inc. (серия TMS320) и Analog Devices (серия 21xx, 210xx). В частности ПЦОС TMS320C2x обладает следующими особенностями :

- выполнение умножения и сохранения результатов за один командный цикл;
- наличие четырехстадийного (четырехкаскадного) конвейера;
- наличие команд, поддерживающих вычисления с плавающей точкой;
- наличие внутреннего ПЗУ программ (ROM) размером 4Кслов для TMS320C25 или ПЗУ с ультрафиолетовым стиранием (EPROM) 4Кслов для TMS320E25;
- выполнение программ осуществляется чтением из памяти программ RAM, расположенной на кристалле процессора;
- объем памяти программ RAM — 544 слова, из которых 256 слов могут быть использованы как память данных;
- расширяемая внешняя память может иметь объем 128Кслов (64К слов на память программ, 64К слов на память данных);

- реализована возможность перемещения содержимого памяти данных и памяти программ блоками;
- реализована возможность организации циклов ожидания при доступе к «медленной» внешней памяти ОЗУ или внешним устройствам;
- процессор TMS320C20 содержит на кристалле таймер и последовательный порт ввода/вывода;
- микросхема ПЦОС включает пять (TMS320C20) или восемь (TMS320C25) вспомогательных регистров и специальное арифметическое устройство для них;
- существуют команды обработки битовых данных;
- наличие режима прямого доступа к памяти DMA (только для МПр TMS320C25).

Простейший ЦСП TMS320C20x имеет производительность до 40 миллионов операций в секунду, время машинного цикла составляет 200 нс, ЦПУ процессора 16-ти разрядное, АЛУ – 32-х разрядное, аккумулятор – 32-х разрядный. Имеются схемы сдвига (shifters), умножитель, встроенная энергонезависимая Flash-память ёмкостью 32К слов, где длина слова равна 16 разрядам. Скорость обмена с внешними устройствами (ЦАП, АЦП) через синхронный последовательный порт составляет до 20 Мбит/сек, буфер ввода/вывода имеет 4 места в очереди, которая обслуживается по дисциплине FIFO (первый пришёл – первый ушёл) с 8-ю и 16-ю разрядными данными. Имеется внутренний генератор тактовой частоты, который может использовать внешние опорные частоты для синхронизации.

Выбор того или иного ПЦОС для средства связи – многокритериальная задача. Процессоры ЦОС могут показывать разную производительность, например, для приложений, требующих выполнения больших объёмов математических вычислений (таких как цифровая фильтрация сигнала, вычисление корреляционных функций и т.п.) в сравне-

нии с задачами, требующими интенсивного обмена с внешними устройствами (многопроцессорные системы, различного рода контроллеры). Расширенные коммуникационные возможности (например высокоскоростные интерфейсы), наличие достаточных объёмов внутрикристальной памяти для данных и программ, возможность защиты программ от несанкционированного доступа, поддержка режима энергосбережения являются очевидными признаками технического совершенства ПЦОС.

4.3 Процессоры сетевой интерфейсной карты

В качестве процессора сетевой интерфейсной карты можно рассматривать ***сетевой адаптер*** или ***сетевую интерфейсную карту*** (network interface card, NIC) – устройство ввода-вывода в составе линейного модуля (карты) средства связи или сервера телекоммуникационных служб, которое создаёт интерфейс с физической средой передачи. Сетевой адаптер решает задачи надежного обмена электрическими или оптическими сигналами по внешним линиям связи. Как и любой процессор, сетевой адаптер работает под управлением драйвера устройств ввода-вывода операционной системы. Сетевой адаптер обычно выполняет следующие функции:

Оформление передаваемой информации в виде кадра определенного формата. Кадр включает несколько служебных полей, среди которых имеется адрес средства связи/компьютера назначения, контрольная сумма кадра. Контрольная сумма на приёме кадра вычисляется заново, сравнивается с полученной и сетевой адаптер компьютера/средства связи назначения делает вывод о корректности доставленной по сети информации.

Получение доступа к среде передачи данных. В локальных вычислительных сетях между группой компьютеров применяются разделяемый доступ к каналу связи (общая шина, кольцо) по специальному алгоритму. Наиболее часто применяются метод случайного доступа

или метод с передачей маркера доступа по кольцу. В последних версиях стандартов и технологий локальных сетей наметился переход от использования разделяемой среды передачи данных к использованию виртуальных каналов связи. Технологиями, использующими виртуальные каналы, являются ATM и коммутирующие модификации традиционных технологий, например коммутируемый Ethernet (Switching Ethernet). При использовании виртуальных каналов связи в функции сетевого адаптера часто входит установление виртуального соединения с центральным коммутатором сети.

Кодирование последовательности бит кадра последовательностью электрических/оптических сигналов при передаче данных и декодирование при их приеме. Кодирование должно обеспечить помехозащищённую передачу исходной информацию по линиям связи с определенной полосой пропускания и определенным уровнем помех таким образом, чтобы принимающая сторона смогла правильно распознать посланную информацию с высокой степенью вероятности.

Преобразование данных из параллельной формы в последовательную форму и обратно. Эта операция связана с тем, что для упрощения проблемы синхронизации сигналов, удешевления линий связи и в связи с физическими проблемами затухания в сетях связи информация передается в последовательной форме, бит за битом, а не побайтно, как внутри средства связи.

Синхронизация битов, байтов и кадров. Для устойчивого приема передаваемой информации необходимо поддержание постоянного синхронизма приемника и передатчика информации. Сетевой адаптер использует для решения этой задачи специальные методы кодирования, не использующие дополнительной шины с тактовыми синхросигналами. Эти методы обеспечивают периодическое изменение состояния передаваемого сигнала, которое используется тактовым генератором приемника для подстройки синхронизма. Кроме синхронизации на уровне

битов, сетевой адаптер решает задачу синхронизации и на уровне байтов, и на уровне кадров.

Сетевые адаптеры различаются по типу и разрядности используемой в компьютере внутренней шины данных – ISA, EISA, PCI. Сетевые адаптеры различаются также по типу принятой в сети сетевой технологии – Ethernet, Token Ring и т.п. Как правило, конкретная модель сетевого адаптера работает по определенной сетевой технологии (например, Ethernet). Для каждой информационной технологии сейчас имеется возможность использования различных сред передачи данных. Например, стандарт Ethernet поддерживает передачу по коаксиальному кабелю, по неэкранированной витой паре и по оптоволоконному кабелю. В результате сетевой адаптер может поддерживать как одну, так и одновременно несколько физических сред. В случае, когда сетевой адаптер поддерживает только одну среду передачи данных, а необходимо использовать другую, применяются трансиверы и конверторы.

В частности, компания Intel, США предлагает решения NIC Ethernet со скоростью передачи от 10 Мбит/с до 10 Гбит/с, с широким диапазоном MAC–адресов, поддержкой различных физических сред на основе интегрированных однокристальных решений. Та же компания Intel производит интегральные схемы процессоров уровня 2 модели BOC и процессоров для коммутируемых решений уровней 2/3/4 модели BOC для модульных и стековых коммутаторов, разработанных для уменьшения стоимости порта при предоставлении максимальной производительности. Рассмотрим устройства для различных уровней BOC более подробно.

При использовании в сетевом окончании, устройства, реализующие функции уровня 1 и уровня 2 BOC, дополняются процессорами сетевой инфраструктуры Intel, обеспечивая полное решение управления пакетным трафиком, необходимое для преодоления критических моментов в точках доступа линейной платы с разделением каналов. Ре-

шение с двумя кристаллами состоит из блока агрегации для создания и отображения кадров и устройства высокого уровня контроля данных (HDLC), чтобы обеспечить обработку задач уровня 2 модели ВОС, анализ входящих данных и поддержку множественных протоколов, включая мультиплексирование с разделением времени для различных телекоммуникационных протоколов. В дополнение к оптимизации работы за счёт разгрузки сетевого процессора от обработки данных уровня 2, решение NIC на базе дополнительного процессора уровня 1/уровня 2 может устранить потребность в нескольких линейных платах, чтобы осуществлять преобразования T1/E1, DS3/E3, OC-3/STM-1, OC-12/STM-4 разделением каналов до DS-0.

Устройства уровня 1 и уровня 2 Intel обеспечивают универсальную архитектуру интерфейсной платы, чтобы поддерживать разнообразные типы трафика и способы его передачи с помощью программного обеспечения управления NIC. В некоторых случаях, при фиксированном алгоритме обработки данных в NIC модно использовать специализированные микросхемы. В частности, для построения мультиплексоров PDH [37], применяются сдвоенные трансиверы на основе микросхемы Intel LXT-332 (см. рис. 4.7).

Микросхема Intel LXT-332 представляет собой полностью интегрированный блок линейного интерфейса для работы на скорости 1544 кбит/сек или 2048 кбит/сек. В состав данной микросхемы входят :

- кодер-декодер HDB-3, B8ZS;
- линейный интерфейс согласно Рек. МСЭ-Т G.703 с эквалайзером, управляющим амплитудой выходных импульсов передатчика;
- аттенюатор джиттера, коммутируемый в так передачи и в так приёма;
- встроенный кварцевый генератор;

- генератор псевдослучайной последовательности и детектор ошибок для контроля тактов;

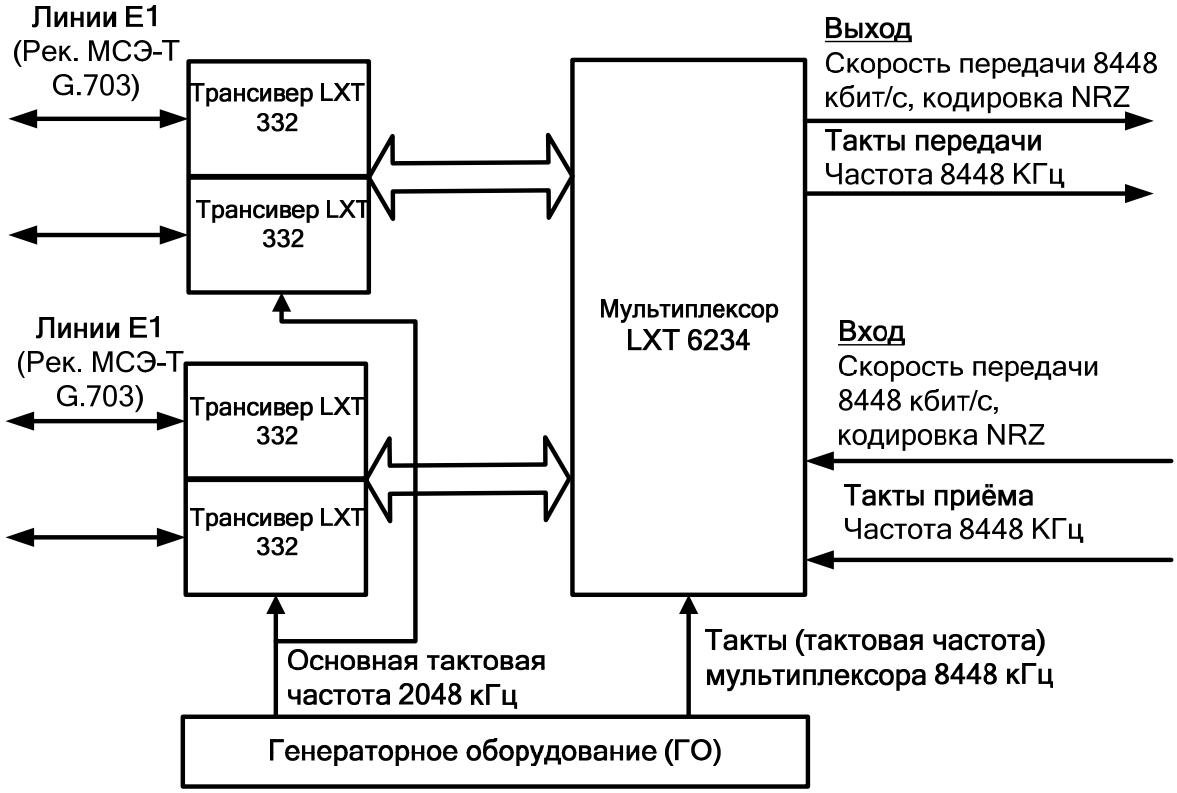


Рис. 4.7 – Структурная схема мультиплексора E2 на основе микросхем компании Intel [37]

- устройства замыкания шлейфов как в сторону станции так и в сторону линии связи;
- последовательный интерфейс для работы с МПр;
- схема диагностики и контроля.

Микросхема LXT-332 выполнена по К-МОП технологии и имеет два основных режима работы :

- Аппаратный режим, не требующий подключения внешнего управляющего МПр.
- Микропроцессорный режим, предусматривающий работу с внешним управляющим МПр. МПр здесь используется для

конфигурирования микросхемы LXT-332, диагностики, сканирования аварийных состояний.

Аппаратный режим позволяет строить самые простые устройства, не используя программное обеспечение управление. Такое решение позволяет поддерживать все функциональные режимы, за исключением контроля тактов с помощью встроенного генератора псевдослучайной последовательности и детектора ошибок.

Микропроцессорный режим позволяет получать более гибкое решение, использовать микросхему трансивера LXT-332 во взаимодействии с системой управления мультиплексором, которая обеспечивает дистанционный контроль аварийных состояний, конфигурирование и мониторинг состояний всех трактов системы передачи.

Трансиверы, аналогичные рис. 4.7, могут работать в микропроцессорном режиме под управлением восьмиразрядных МПр Intel или Motorola с использованием последовательного или параллельного интерфейса. В режиме микропроцессорного управления обеспечивается немедленный контроль (по процедуре прерывания) работы драйверов приема и передачи данных системы передачи. Для проверки работоспособности трансиверов имеются различные диагностические режимы: организация местного, удаленного и аналогового шлейфов, а также включение сигнала индикации аварийного сигнала, СИАС) В микросхему LXT332 встроен генератор и детектор псевдослучайной последовательности, ПСП. Для операций с трактом Е1 используется 2^{15}-1 (32768) ПСП с инвертированием на выходе. Имеются средства для введения логических ошибок в ПСП и нарушений чередования полярности в линейный код. Использование ПСП в ИС LXT332 возможно только в режиме микропроцессорного управления.

Как видно на рис. 4.7, трансиверы связаны с мультиплексором по внутреннему интерфейсу на основе NRZ-кодированного сигнала. Набор

тактовых сигналов, необходимых для работы мультиплексора, формируется набором генераторного оборудования.

Дальнейшее описание работы схемы на рис. 4.7 можно найти на стр. 138 – 146 [37].

4.4 Процессоры ввода-вывода

Для увеличения скорости обмена между внешним устройством и памятью управляющего комплекса используют специализированные процессоры (сопроцессоры) ввода/вывода. Это обусловлено тем, что в связи с интенсивным ростом использования телекоммуникационных сетей для межмашинного или межпроцессорного обмена, а также в связи с увеличением объёма и скорости передачи данных, скорость передачи при вводе-выводе стала одним из главных критериев для оценки эффективности телекоммуникационного устройства. Серверы информационно-вычислительных систем, серверы телематических служб, сетевые устройства памяти, сети хранения данных требуют высокоскоростного ввода/вывода для обеспечения высокой общей производительности. Процессоры ввода/вывода обеспечивают повышение скорости передачи данных между аппаратными компонентами средств связи, позволяют устранить задержки при обмене информацией в коммуникационных системах и повысить общую производительность за счет того, что функции управления вводом/выводом с центрального процессора перекладываются на специализированный процессор ввода/вывода. В некоторых случаях процессор ввода-вывода берет на себя функции обработки прерываний ввода/вывода и контроля четности. Это позволяет ускорить выполнение программ на процессоре ЦУУ или ГУУ, более эффективно использовать такие ресурсы как общую системную шину и оперативную память.

Например, процессор ввода/вывода Intel IOP 321 имеет тактовую частоту 400 МГц или 600 МГц, применяется для работы в телекоммуни-

кационных системах для аппаратной поддержки ввода/вывода данных с использованием стека протоколов TCP/IP, а также при использовании технологии RAID и контроллеров НЖМД типа iSCSI. Данный процессор с RISC–системой команд обеспечивает скорость ввода/вывода 1,6 Гбит/сек при работе с шиной расширения, конструктивно имеет кэш данных и кэш команд ёмкостью по 32 Кбайт каждый. Данный процессор допускает применение встроенного контроллера DMA, а также контроллера доступа к оперативной памяти. Процессор работает с внешней шиной с тактовой частотой 200 МГц и разрядностью 64 бита. Размер корпуса процессора составляет 35 х 35 мм. Уменьшенное потребление мощности данного процессора делает ненужным использование радиатора для рассеивания избыточного тепла, выделяемого данным процессором. Процессор поддерживает шину PCI-X и оперативную синхронную динамическую оперативную память PC200 DDR SDRAM, где 200 – тактовая частота работы памяти, измеренная в МГц.

Рассмотрим в качестве процессора ввода-вывода специализированный микропроцессор Intel 80321 с RISC-архитектурой производства компании. Функциональная блок-схема данного МПр приведена на рис. 4.8.

Ядро (core, ЦПУ) МПр 80321 изготовлено по технологии Intel XScale, тактовая частота работы ядра составляет 600 МГц, имеется кэш-память для инструкций ёмкостью 32 Кбайт и кэш данных ёмкостью 32 Кбайт, а также дополнительный мини-кэш данных ёмкостью 2 Кбайт. Кэш-памяти команд и кэш-памяти малой ёмкости традиционно предназначена для хранения постоянно меняющихся данных.

Технология XScale является фирменной технологией компании Intel. Эта технология предусматривает наличие семи стадийного конвейера для выполнения операций с целыми числами, восьми стадийного конвейера для чтения-записи данных в оперативную память.

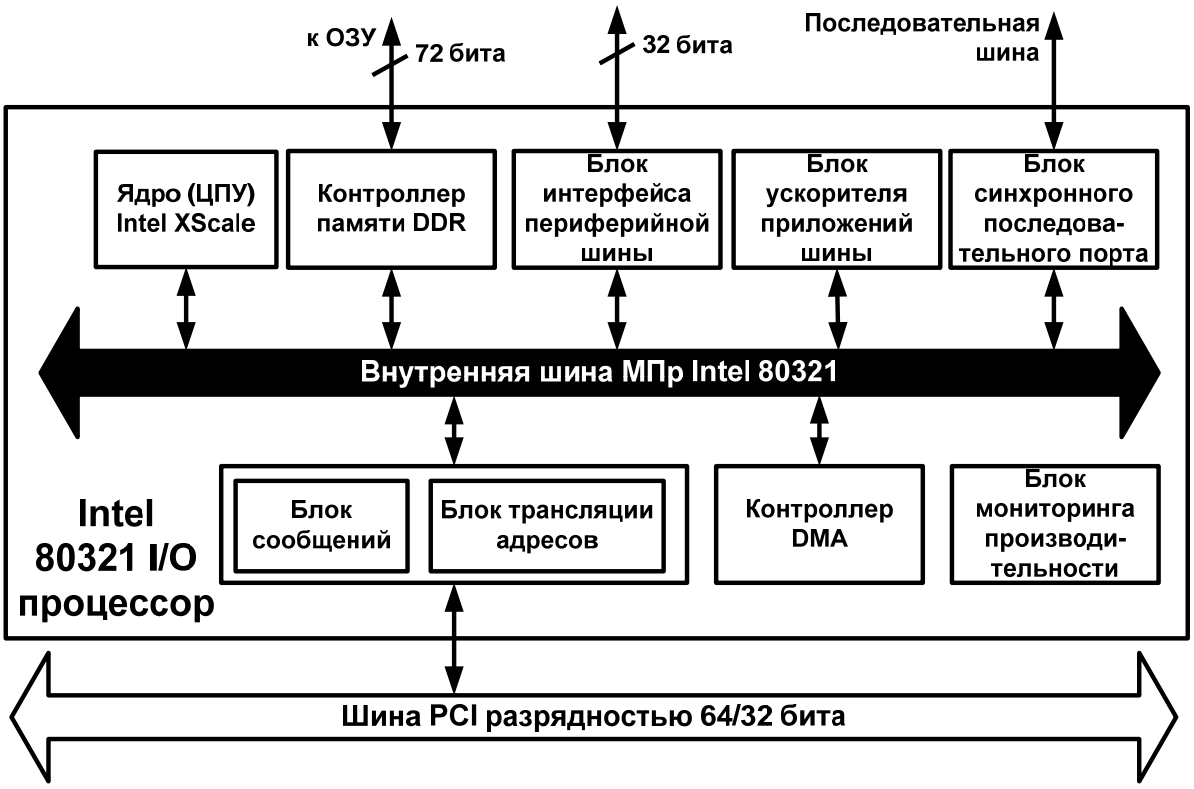


Рис. 4.8 – Функциональная блок-схема МПр ввода-вывода Intel 80321

В процессоре Intel 80321 имеется 128-строчный буфер предсказания переходов (см. А. Фокин Процессоры ввода/вывода Intel на базе технологии Xscale//Компоненты и технологии.–2006.–№3. Режим доступа [http://www.kit-e.ru/articles/cpu/2006_3_94.php]). Этот буфер хранит историю предыдущих ветвлений в исполняемой процессором ввода-вывода программе. Ветвление предусматривает смену потока исполняемых команд, что соответствует блоку решений «да–нет» в алгоритме. ЦПУ процессора ввода-вывода постоянно просматривает выполняемый фрагмент машинного кода и предсказывает следующий программный переход. История ветвлений хранится в буфере вместе с адресами перехода. При обнаружении в потоке команд операции перехода, вычисленный адрес перехода сравнивается с адресами, хранящимися в буфере предсказания переходов. В случае совпадения, адрес из буфера переходов используется в качестве адреса команды, которая посылается в кэш-память команд. В итоге, в случае правильного предсказания перехода, команда из другой программной ветви загружается в конвейер без за-

держки. Наличие буферов адресов перехода позволяет повысить вероятность правильного предсказания перехода, увеличивая производительность процессора ввода-вывода на 15%. Данная технология будет также подробно рассматриваться в главе 5.

Рассмотрим теперь назначение и характеристики прочих компонентов процессора ввода-вывода.

Внутренняя шина (Internal Bus) представляет собой высокоскоростную магистраль разрядностью 64 бита и с тактовой частотой 200 МГц, соединяющую все внутренние компоненты процессора ввода-вывода между собой.

Контроллер DMA (DMA Controller) обеспечивает высокоскоростную передачу данных с минимальной задержкой по времени между шиной PCI и оперативной памятью. Контроллер DMA позволяет организовать передачу сцепленных данных (цепочек данных), а также случайных, разрозненных, несцепленных данных. Контроллер DMA программируется через ЦПУ МПр (core) 80321; поддерживает адресное пространство размером 2^{32} адресов при работе с внутренней шиной Internal Bus и адресное пространство размером 2^{64} адресов при работе с общесистемной шиной PCI. Скорость обмена с внутренней шиной Internal Bus составляет до 1600 Мбайт/сек; скорость обмена с шиной PCI режиме PCI-X составляет до 1064 Мбайт/сек.

Блок трансляции адресов (Address Translation Unit, ATU) позволяет прямой доступ к локальной памяти МПр 80321 для обращений с шины PCI. Блок ATU поддерживает отображение между адресами шины PCI и внутренним адресным пространством МПр 80321. Трансляция адресов контролируется через программируемые регистры, доступные как через интерфейс с шиной PCI так и ядру Intel XScale, что обеспечивает гибкость при отображении одного адресного пространства на другое. Поддерживает очереди на чтение/запись ёмкостью до 4 Кбайт.

Блок сообщений (Messaging Unit, MU) обеспечивает обмен данными между шиной PCI и МПр 80321. Этот блок использует систему прерываний для уведомления МПр о поступлении новых данных. Здесь применяются специальные внутренние регистры для организации промежуточного хранения и обмена данными.

Контроллер оперативной памяти DDR позволяет реализовать прямое управление подсистемой памяти PC200 DDR SDRAM. Возможности контроллера позволяют программно поддерживать выбор микросхемы памяти и коды коррекции ошибок (error correction codes, ECC).

Блок интерфейса периферийной шины (peripheral bus interface unit, PBI) представляет собой тракт для обмена данными для тех компонентов аппаратного обеспечения МПр 80321, которые не имеют интерфейса с шиной PCI и/или размещение которых неоптимально на шине PCI. Примером таких компонентов является флэш-память (flash memory) и интерфейсные порты к ПЦОС. Блок PBI позволяет МПр 80321 обрабатывать данные и взаимодействовать с указанными выше компонентами при организации ввода/вывода. Блок PBI поддерживает 32-х разрядную передачу данных с рабочей тактовой частотой 33, 66 и 100 МГц.

Блок ускорителя приложений (Application Accelerator Unit, AAU) выполняет операцию переноса блоков данных в локальную память МПр 80321 или из локальной памяти а также выполняет булевы операции с данными, такие как «исключающее ИЛИ» (XOR).

Блок мониторинга производительности (performance monitoring unit, PMON) позволяет организовать мониторинг событий, происходящих на МПр 80321. Для этого могут использоваться 14 счётчиков событий, запрограммированных для наблюдения за событиями. Множество событий, за которым осуществляется наблюдение должны быть определены заранее.

Блок синхронного последовательного порта (synchronous serial port, SSP) реализует дуплексный синхронный последовательный интерфейс с тактовой частотой от 7,4 КГц до 1,84 МГц. Этот интерфейс позволяет подключать широкий набор внешних аналогово-цифровых преобразователей (конвертеров), аудио кодеков а также иные устройства, использующие последовательный интерфейс передачи данных.

Предлагаемый процессор ввода-вывода позволяет подключать различные внешние устройства, такие как NIC, Ethernet-адаптеры, ПЦОС к оперативной памяти средтсва связи.

4.5 Мультиплексоры и трансиверы в оптических средствах связи

Рассмотрим использование специализированных МПр и микросхем в оптических системах связи на примере мультиплексоров SDH. Общая структурная схема мультиплексора добавления-выделения SDH представлена на рис. 4.9.

Мультиплексор имеет два двухволоконных оптических интерфейса «Запад» - «Восток» и поддерживает добавление-выделение в тракт STM-1 до 42 потоков E1.

Для построения рассматриваемого мультиплексора можно использовать специализированный микросхемный набор (чипсет) компании Intel, в состав которого входят два базовых типа СБИС-микросхем :

- микросхема терминатора секций STM – 1/0 LXT 6051;
- микросхема отображения (SDH-мэппер) 21 E1 типа LXT6251.

Микросхема LXT 6051 имеет более 100 управляющих регистров, доступных для обмена данными внешней микропроцессорной системе управления по специальному последовательному двунаправленном интерфейсу. Микросхема LXT 6051 выполняет преобразование трактов VC-4 (VC-3) в тракт STM-1 (STM-0) соответственно через AU-4 и AUG (AU-3 для STM-0).

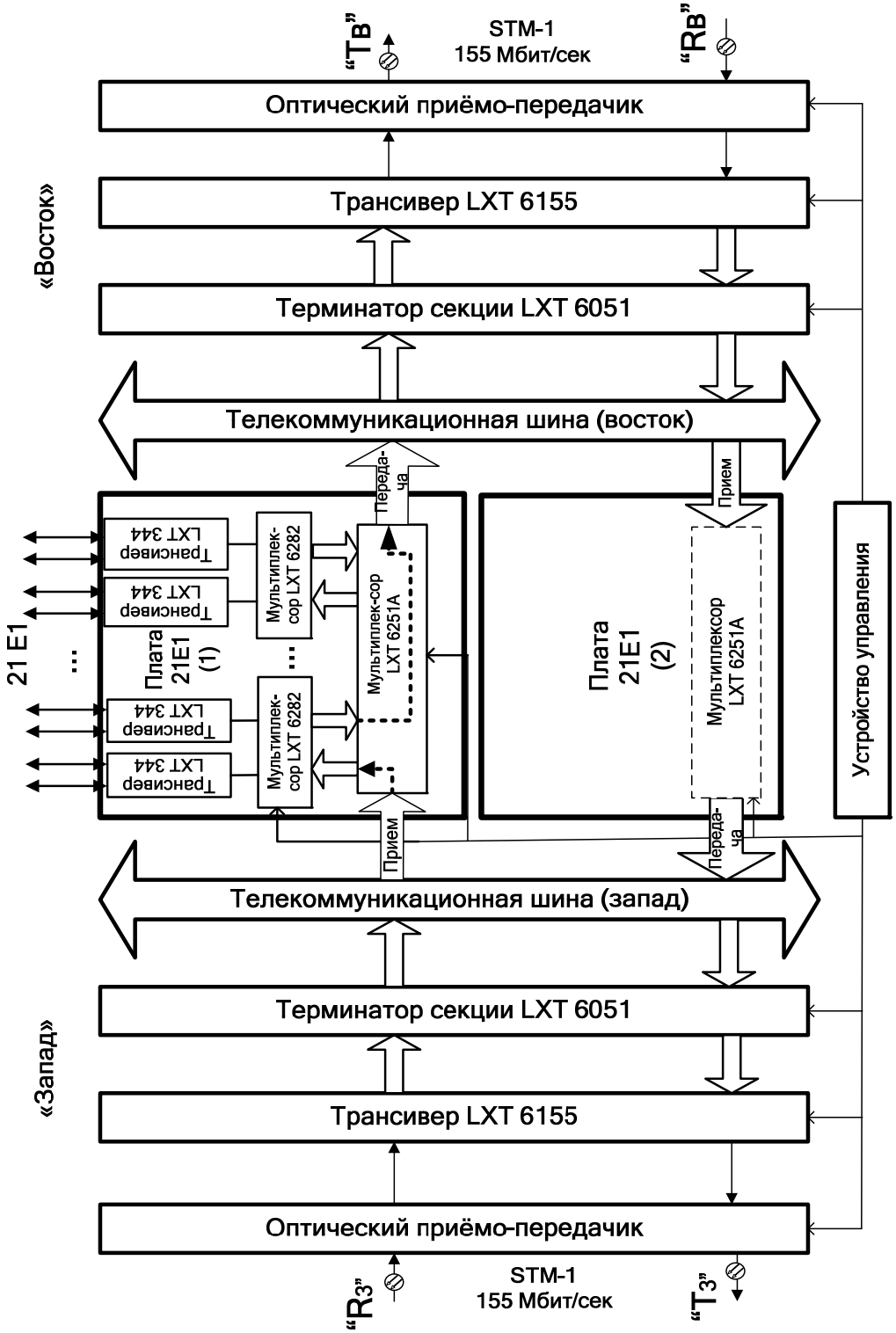


Рис. 4.9 – Структурная схема мультиплексора SDH на основе микросхем компании Intel [37]

Микросхема СБИС LXT 6251 может поддерживать до восьми интерфейсов E1 и выполняет преобразование E1 → C-12 → VC12 → TUG-2 → TUG3. Микросхема SDH-мэппера формирует маршруты передачи информационных структур, соответствующих добавляемым (вводимым) и выводимым с их помощью потокам E1, а также потокам, пропускаемым через мультиплексор транзитом и передаваемым в оба направления SDH-кольца. Указанная микросхема выполняет функции размещения потоков E1 в виртуальных контейнерах VC-12, извлечение потоков из этих контейнеров (мэппинг, mapping), мультиплексирование 21 VC-12 в TUG-3, гибкое конфигурирование доступом. Микросхема LXT 6251 имеет более 20 управляющих регистров, доступных внешней микропроцессорной системе управления. Через эти регистры осуществляется конфигурирование данной СБИС и мониторинг её состояния.

Кроме перечисленных, используются дополнительные микросхемы, обеспечивающие работу интерфейсов в сторону линий связи E1 и в сторону оптических линий :

- трансивер оптической линии LXT6155, сопрягающий параллельный интерфейс STM-1 (последовательный интерфейс STM-0) терминатора секции с последовательным интерфейсом оптического приёмопередатчика;
- интерфейсные устройства E1 LXT6282 и трансиверы линий E1 LXT344 для сопряжения LXT6251 с трактами E1.

Интерфейсные устройства осуществляют выравнивание джиттера, мониторинг состояний трактов E1. Четырехканальные трансиверы LXT344 поддерживают электрические параметры интерфейсов E1 в соответствии с рек. МСЭ-Т G.703. Для этой же цели используются восьмиканальные трансиверы LXT6282.

Рассмотрим порядок работы рассмотренных микросхем в составе мультиплексора SDH.

NRZ-кодированные сигналы со скоростью 155 Мбит/сек с выходов приёмных частей оптических приёмопередатчиков поступают через последовательную эмиттерно-связанную логику интерфейса на входы трансиверов на основе микросхем LXT 6155 восточного и западного направления. Приёмные части трансиверов осуществляют выделение приемного тактового сигнала в каждом из направлений и преобразуют принимаемый сигнал в сигналы восьмиразрядного параллельного интерфейса STM-1. С выходов трансиверов сигналы подаются на входы терминаторов секции LXT6051 западного и восточного направлений. Терминаторы анализируют заголовки STM-1, AU3, AU4, VC-3, VC-4 и выполняют функции окончания секции регенератора, секции мультиплексора, окончания маршрутов высших порядков, а также функции мониторинга и обнаружения аварийных состояний, конфигурирования маршрутов приёма и передачи под управлением внешнего микроконтроллера устройства управления.

Терминаторы секций соединяются с платами мэпперов, где каждая плата мэппера поддерживает ввод/вывод до 21 потока E1. На каждой из плат мэппера установлены рассмотренные выше микросхемы LXT6251, интерфейсные устройства LXT6282 и трансиверы LXT344, LXT6282. Указанные платы мэпперов соединены с терминаторами секций с помощью двунаправленных высокоскоростных шин по стандарту IEEE P1396.

В обратном направлении, с выходов передающей части мэпперов сигналы через телекоммуникационные шины восточного и западного направлений поступают в передающие части терминаторов секций и далее через параллельные 8-ми разрядные интерфейсы – в трансиверы, откуда по последовательным интерфейсам 155 Мбит/сек на передающие части оптических приёмопередатчиков и после преобразования в оптические сигналы – в волоконно-оптическую линию связи через оптические порты.

Следует отметить, что оптические интерфейсы STM-1 могут входить в состав функциональных блоков современных систем коммутации, например 5ESS производства компании Lucent Technologies, США. Данные интерфейсы управляются централизованно ЦУУ и/или ИУУ АТСЭ. На данных интерфейсах могут выполняться операции конфигурирования заголовков всех информационных структур для обмена с внешними мультиплексорами и кросс-коннекторами (аппаратура оперативного переключения) SDH, конфигурирование заголовков для транспортировки структур на соответствующие порты ввода/вывода E1 удалённых устройств доступа, для мониторинга и ведения базы данных аварийных состояний. В некоторых случаях для связи устройства управления мультиплексорами могут быть связаны с ЦУУ/ИУУ системы коммутации через физический интерфейс.

Мультиплексоры могут соединяться с коммутационным полем АТСЭ как через интерфейсы E1 так и через специальные высокоскоростные внутристанционные интерфейсы. Управляющая информация передаётся в специальных канальных временных интервалах. При этом требуется применять дополнительную аппаратуру, осуществляющую разборку сигнала такого интерфейса и сопряжение с интерфейсами мультиплексора, например через интерфейсы NRZ-кодированных сигналов между трансиверами E1 и устройствами интерфейса E1. Трансиверы E1 в этом случае могут использоваться для подключения только внешних устройств, а для внутристанционных связей не используются.

4.6 Контрольные вопросы к главе 4

1. Какие функции реализуют сетевые процессоры?
2. В чём особенность архитектуры сетевых процессоров?
3. Требуется ли для сетевых процессоров внешнее управляющее устройство?

4. В чём особенность супергарвардской архитектуры процессоров цифровой обработки сигналов?
5. Для чего в ПЦОС используются регистры–аккумуляторы повышенной разрядности?
6. В чём заключается эффективность применения теневых регистров в ПЦОС?
7. Почему в ПЦОС некоторые вычисления поддерживаются аппаратно?
8. Чем обусловлено наличие в ПЦОС кругового буфера?
9. Перечислите функции сетевого адаптера в средствах связи и ПЭВМ.
10. Для чего процессор ввода-вывода поддерживает режим DMA?

5. Тенденции развития микропроцессоров

5.1 Развитие технологий производства микропроцессоров

Развитие современных МПр определяется новыми технологиями конструирования и производства аппаратной части процессора, совершенствованием процесса обработки и хранения данных МПр. Первая интегральная микросхема на кремниевой пластине была продемонстрирована американским исследователем Джеком Килби руководству компании Texas Instruments, США 12 сентября 1958 года. Чуть позже тоже самое сделал Роббер Нойс из компании Fairchild Semiconductor. В настоящее время появились признаки того, что использование традиционных технологий производства МПр на кремневой пластине подходит к физическому пределу (см. рис. 5.1).

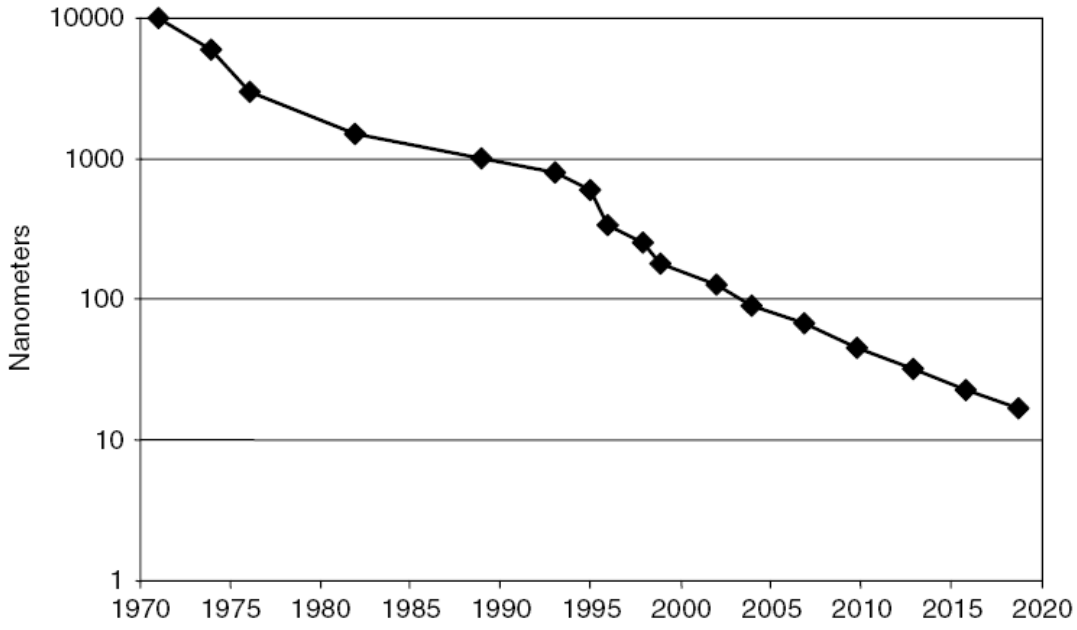


Рис. 5.1 – Уменьшение технологической нормы при производстве МПр по годам
[www.books.elsevier.com/bookscat/samples/0123724988/Sample_Chapters/02~chapter_one.pdf]

Ещё в 2003 г. Компания IBM получила транзистор размером в 8 молекул. Считается, что с точки зрения экономической эффективности производства технологической нормой является 22…32 нм, которая будет задействована не ранее 2010 года. В частности, компания IBM заключила соглашение о производстве микросхем с топологическим размером элемента в 32 нм с компанией BASF (Германия) с выпуском на рынок в 2010 г. Также к 2010 году к технологической норме 32 нм планирует перейти компании AMD. В 2007– 2008 годах к выпуску процессоров по 45–нм технологии перешли соответственно компания Intel и компания AMD. Технологическим пределом изготовления транзисторных микросхем считается размер транзистора в 10 нм. Сейчас в транзисторах на базе 65-нм технологии, компания Intel довела размер затвора транзистора до 35 нм (это приблизительно на 30% меньше, чем при производстве по 90-нм технологии), а толщина оксидного слоя затвора уменьшена до 1,2 нм. Технологическая норма 65 нм соответствует размерам шести атомов, а при норме 45 нм микроэлектронные компоненты уменьшаются настолько, что становятся меньше длины волны лазерного луча, с помощью которого эти элементы вытравливаются на кремниевой пластине. Для преодоления указанного недостатка используются различные методы. В перспективе планируется использовать новые методы литографии, например иммерсионную литографию. Уже сейчас вместо воздуха между линзами и кремнием при изготовлении микросхем используется жидкость, что уменьшает длину волны и повышает разрешение на 40%. Применение оптической литографии на основе глубокого ультрафиолетового излучения (deep-ultraviolet, DUV) позволяет довести норму проектирования до 29,9 нм. В качестве альтернативы диоксиду кремния с перспективой на 20…25 лет рассматривается гафний. Имеются варианты построения биочипов, в которых для реализации вычислительных устройств применяются белковые элементы. На конец 2006 года рекорд скорости переключения транзистора

составляет 845 ГГц. Это «супертранзистор» изготовлен из фосфида индия и арсенида индия-галлия; этот состав позволил повысить скорость инжекции электронов, снизить плотность тока и время накопления заряда. Уменьшены и размеры самого транзистора – ширина его базы составляет всего 12,5 нм. Работая при комнатной температуре, такой транзистор переключается на скорости 765 ГГц, а частота 845 ГГц достигнута при охлаждении до –55 °С. Конечная цель исследований – довести частоту переключения транзистора до 1 ТГц.

С точки зрения увеличения вычислительной мощности МПр, начиная с 1980-х годов производительность МПр наращивалась за счёт постоянного увеличения тактовой частоты, уменьшения размера и, соответственно, увеличения количества транзисторов на единицу площади кристалла МПр. На переключение транзисторных компонентов, схемно реализующих логические элементы, затрачивается определённая мощность. При увеличении тактовой частоты электроимпульсов постоянного тока в полупроводниковых и металлических компонентах процессора возникает избыточное тепловыделение, в первую очередь за счёт законов физики. Тепловыделение элементной базы процессоров принято измерять в пикоджоулях на переключение одного бита (1 ПкДж/Бит = 10^{-12} Дж/Бит). Это энергия, выделяемая при переключении одного вентиля. При современных тактовых частотах и плотностях интеграции на кристалле суммарные тепловыделения имеют порядок нескольких ватт на площади в 1 квадратный сантиметр. В связи с этим достаточно остро стоит проблема отвода тепла от процессора для обеспечения необходимого температурного режима.

Эта ситуация усугубляется ещё и тем, что при уменьшении физических размеров полупроводниковых компонентов, прежде всего затворов транзисторов, неизбежно возникают сильные токи утечки; причём чем выше тактовая частота и энергопотребление, тем больше токи утечки. В итоге опять возникает избыточное тепловыделение и без

принятия мер по охлаждению процессор может перегреться и отказать. Теоретическим пределом роста тактовой частоты современных МПр для применения в промышленных условиях считается величина 10 ГГц (см. рис. 5.2).

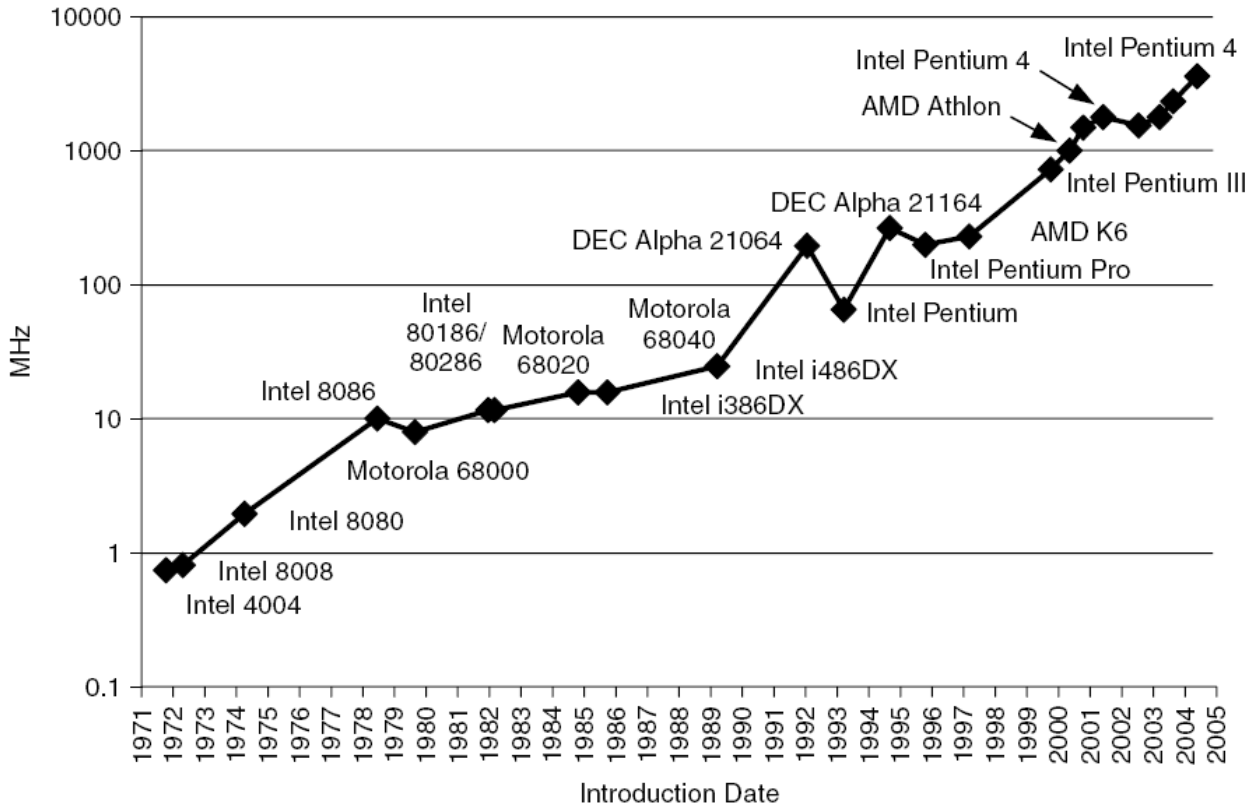


Рис. 5.2 – Увеличение тактовой частоты МПр по годам

Следует отметить, что нагрев кристалла МПр происходит неравномерно. Результаты исследований лаборатории изучения цепей корпорации Intel под руководством Р. Кришнамурти (Ram Krishnamurty) показывают, что до +125°С нагревается лишь небольшой участок процессора – АЛУ. Остальная часть МПр, включая кэш-память, нормально функционирует при вполне приемлемой для кристалла МПр температуре не выше +65°С. Если МПр выполняет простую пересылку данных из одного участка памяти в другой, то это не приводит к нагреву. Когда производятся сложные математические операции, процессор нагревается в основном за счёт нагрева АЛУ; усугубляет ситуацию то, что современные МПр используют не одно, а от четырех до двадцати четы-

рех АЛУ. Поскольку АЛУ постоянно обмениваются данными и располагаются поблизости друг от друга, то плотность выделения тепла увеличивается.

С учётом роста затрат на реализацию технически сложной системы охлаждения процессоров в рамках микропроцессорных систем, в сентябре 2004 г. ведущие производители МПр заявили об отсутствии планов выпуска процессоров с тактовой частотой свыше 4 ГГц. В частности, одним из последних «быстрых» процессоров стал Pentium 4 560 на ядре Prescott (3,6 и 3,8 ГГц) и мощностью 100 Вт на максимальной частоте. Переход к мощности процессора в 200 Вт стал невозможен в связи с нежеланием производителей ЭВМ обеспечивать новые условия охлаждения. Для AMD максимально производительным стал процессор Athlon 64 с тактовой частотой 2,4 ГГц. Эксперимент, чьи результаты приводились на сайте www.thg.ru показывает, что при разгоне Intel Pentium 4 до 5,2 ГГц необходимо охлаждение из жидкого азота с температурой на кристалле до -190°С. В мае 2008 г. компания IBM объявила о выпуске на рынок нового высокопроизводительного суперкомпьютера IBM Power 575 Hydro-Cluster с фирменным водяным охлаждением на базе МПр IBM POWER6. Решение с водяным охлаждением скорее носит исключительный характер и применяется пока в больших центрах обработки данных, например в физическом Институте по изучению плазмы Макса Планка, Германия. Несмотря на водяное охлаждение, МПр IBM POWER6 оснащен несколькими энергосберегающими технологиями, среди которых режим «nap» (спящий режим), который снижает потребление энергии ЦПУ на 30-35% при бездействии операционной системы и динамическое регулирование частоты и напряжения питания процессора. Ядро процессора IBM POWER6 работает на частотах 3,5…4,7 ГГц, ядро способно выполнять одну инструкцию за 3 нс, на кристалле с десятиуровневой медной металлизацией площадью 341 мм^2 находится почти 790 миллионов транзисторов, МПр выполнен с по-

мощью комбинации 80-нм и 65нм технологий производства процессоров.

Описанные в настоящем подразделе проблемы тепловыделения, охлаждения, энергоэффективности заставляют производителей МПр искать иные пути повышения вычислительной мощности, нежели увеличение тактовой частоты. Например, изменения в конструктивном исполнении транзисторов, в частности, снижение токов утечки также способствуют решению проблемы избыточного тепловыделения.

Кроме того, соотношение энергопотребления оперативной памяти и МПр составляет по крайней мере 2 к 1 (см *Р. Митчелл* Новый прожорливый монстр//Computer World Россия. – 13.06.2007. – №22, с.27). Поэтому применение упоминавшейся в главе 1 оперативной памяти стандарта DDR2 с мощностью потребления 2 Вт при простое и 4,6 Вт под нагрузкой при напряжении питания 2,5 В тоже способствует решению проблемы снижения энергопотребления и тепловыделения. Память стандарта DDR3 предусматривает напряжение питания 1,5 В.

В целом, начиная с 1990-х годов были проведены успешные разработки, позволяющие повысить производительность МПр не увеличивая тактовую частоту с применением встроенных технологий энергосбережения. Рассмотрим далее соответствующие способы.

5.2 Конвейерная обработка данных

Исторически первым способом повышения производительности процессора без увеличения тактовой частоты является применение суперскалярной архитектуры т.е. использование развитых механизмов конвейерной обработки данных с поддержкой параллельного режима вычислений.

В учебном пособии [29] приводятся следующие характеристики эффективности внедрения перечисленных методов вычислений :

1. *Ускорение* r параллельной системы, которое используется на начальных этапах проектирования или в научных исследованиях для оценки предельных возможностей архитектуры.

2. *Быстродействие* V, которое является главной характеристикой при конкретном проектировании или выборе существующей параллельной ЭВМ под класс пользовательских задач.

Ускорение определяется выражением:

$$r = T_n / T_1, \qquad (5.1)$$

где

T_1 — время решения задачи на однопроцессорной системе,

T_n — время решения той же задачи на n- процессорной системе.

Пусть $W = W_{ск} + W_{пр}$,

где

W — общее число операций в задаче;

$W_{пр}$ — число операций, которые можно выполнять параллельно;

$W_{ск}$ — число скалярных (нераспараллеливаемых) операций.

Обозначим через *t* время выполнения одной операции. Тогда согласно ***закона Амдала*** [4] получаем значение r :

$$r = \frac{W \cdot t}{(W_{ck} + \frac{W_n}{n}) \cdot t} = \frac{1}{a + \frac{1-a}{n}} \xrightarrow[n\to\infty]{} \frac{1}{a} \qquad (5.2)$$

Здесь $a = W_{ck} / W$ — удельный вес скалярных операций в общем числе операций. Закон Амдала определяет принципиально важные для параллельных вычислений положения:

1. Ускорение вычислений зависит как от потенциального параллелизма программной задачи (величина $1-a$), так и от параметров средства вычислительной техники (числа процессоров n).

2. Предельное ускорение вычислений определяется свойствами программной задачи.

Пусть $a = 0{,}2$ (что является реальным значением), тогда ускорение не может превосходить 5 при любом числе процессоров, то есть максимальное ускорение вычислений определяется прежде всего потенциальным параллелизмом задачи. Очевидной является чрезвычайно высокая чувствительность ускорения r к изменению величины a.

Выражение (5.2) определяет ускорение только одного уровня вычислительной системы. Однако реальные системы являются многоуровневыми как с точки зрения программных конструкций, так и по аппаратной реализации. Реальные параллельные ЭВМ обычно используют параллелизм нескольких уровней и полное ускорение такой ЭВМ *R* можно в первом приближении описать выражением:

$$R = \prod_{i=1}^{M} r_i \qquad (5.3)$$

где

M — число вложенных уровней вычислений, используемых для распараллеливания;

r_i — собственное ускорение уровня i, определяемое параллелизмом соответствующих данному уровню объектов: независимых задач, программ, ветвей, итераций цикла, операторов.

В развитии конвейерных вычислений основополагающим явился метод, названный «принципом водопровода» (позже он стал называться *конвейером*), предложенный советским академиком С.А.Лебедевым в 1956 г. Конвейерная организация вычисления [29,30] предусматривает, что цикл выполнения машинной команды разбивается на несколько элементарных ступеней, стадий или блоков. Команда передвигается по конвейеру, освобождая стадию для следующей команды. Для хранения данных, передаваемых с одной ступени на другую, используются про-

межуточные буферы, находящиеся между стадиями. При этом продолжительность каждой стадии в идеале составляет 1 такт работы МПр, что существенно меньше времени выполнения всей команды (см. рис. 5.3):

Рис. 5.3 – Пример разбиения команды «Считывание» данных из ОЗУ в регистр при конвейерной обработке данных

Организация конвейера позволяет совмещать во времени выполнения разных стадий, например, в один и тот же момент t_i:

Команда №1 – находится на стадии 3;

Команда №2 – находится на стадии 1.

В итоге, время выполнения команд можно существенно сократить. Это позволяет увеличить производительность процессора при одной и той же тактовой частоте.

Принцип организации конвейера команд впервые был использован в конструкции советской ЭВМ М–20, БЭСМ-6 (1957…1966 гг., разработка Института точной механики и вычислительной техники АН СССР) и английской ЭВМ ATLAS (1957-1963 гг.). Конвейер команд в то время предполагал наличие многоблочной памяти и секционированного процессора, в котором на разных этапах обработки находилось несколько команд. Также конвейеры применяли в ЭВМ серии IBM/360 (США) и ЕС ЭВМ (СССР).

Конвейер может быть ***синхронным***, если работает в принудительном темпе и для выполнения каждой стадии выделяется одно и тоже время. Конвейер может быть ***асинхронным***, единое время для каждой стадии отсутствует. Информация с предыдущей стадии передаётся на следующую при условии, что следующая стадия полностью

завершила обработку предыдущей команды. Асинхронный конвейер применяется в старших моделях МПр Pentium.

В современных МПр конвейер может иметь длину до 20 стадий, что характерно для микропроцессоров Pentium IV Northwood; для микропроцессора модификации Prescott количество стадий составляет 31 ступень, в МПр IBM Power количество стадий равно 15. При количестве стадий больше 5..6 МПр может называться «суперконвейерным». В марте 2006 г. компанией Intel было заявлено о том, что в связи с тенденцией к снижению тактовой частоты МПр в перспективных моделях МПр компании Intel (модификации Conroe и Merom) количество стадий будет сокращено до 14, однако количество операций на каждой стадии будет увеличено. Это достигается за счёт возможности выполнений 4 инструкций за 1 такт (в МПр Pentium 4 было только 3 инструкции), в том числе за счёт применения параллельных вычислений на многоядерных процессорах.

Многоядерные процессоры подробно рассматриваются в разделе 5.5 настоящего учебного пособия.

В общем конвейеры можно разделить на две группы:

Векторные конвейеры – выполняют одну операцию над группами разных данных, называемых векторами (например, строка в двумерном массиве). Такие конвейеры, как правило, являются арифметическими, то есть их ступени выполняют части арифметико-логических операций. По ***вектором*** понимается, например, одномерный массив, который образуется из многомерного массива, если зафиксирован только один из номеров строки или столбца. Области применения векторных операций над массивами обширны: цифровая обработка сигналов (цифровые фильтры); механика, моделирование сплошных сред; метеорология; оптимизация; задачи движения; расчеты электрических характеристик БИС.

Наличие векторных конвейеров позволяет реализовать ЭВМ в которых выполняется единственная программа, но каждая ее команда обрабатывает много чисел. МПр с поддержкой векторных конвейеров относятся к классу ***SIMD*** (Single Instruction Multiple Data) — один поток команд, много потоков данных.

Скалярные конвейеры, в которых на разных ступенях обработки одновременно находятся команды с разными кодами операций, но обрабатывающие одни и те же данные. Скалярные конвейеры могут содержать только конвейер команд, но в процессорах с плавающей запятой часто скалярный конвейер включает и арифметические ступени. Таким образом, скалярный конвейер может выполнять векторные операции, для чего необходимо на вход последней в каждом такте стадии подавать один и тот же код операции.

Наличие скалярных конвейеров позволяет реализовать ЭВМ, в которых выполняется несколько программ, причём каждая команда обрабатывает одни и те же данные (единое пространство данных). Такие процессоры относятся к классу **MISD** (Multiple Instruction Single Data) — много потоков команд, один поток данных. Не всегда для выполнения данной команды нужны все без исключения стадии. Возникает ситуация, при которой стадии может быть не нужна и поэтому пропущена. Такая стадия называется ненагруженной. Для снижения ненагруженных стадий, оптимизации вычислений путём их распараллеливания, в современных МПр используются два и более конвейеров (см. рис. 5.4) :

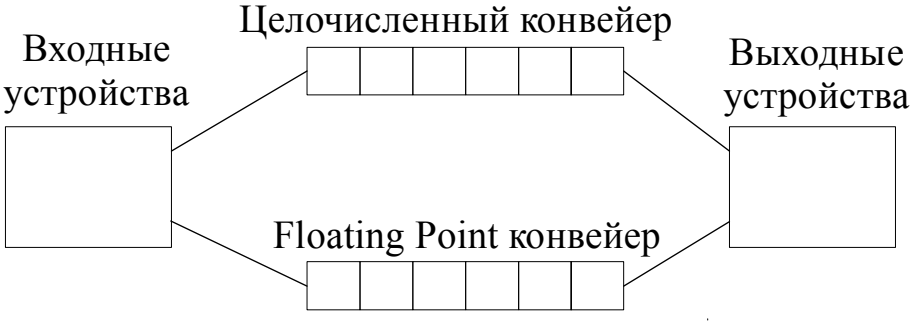


Рис. 5.4 – Архитектура с двумя конвейерами

В частности, целочисленный конвейер имеет следующие стадии (для архитектуры 80x86 Pentium) – см. рис. 5.5 [36]:

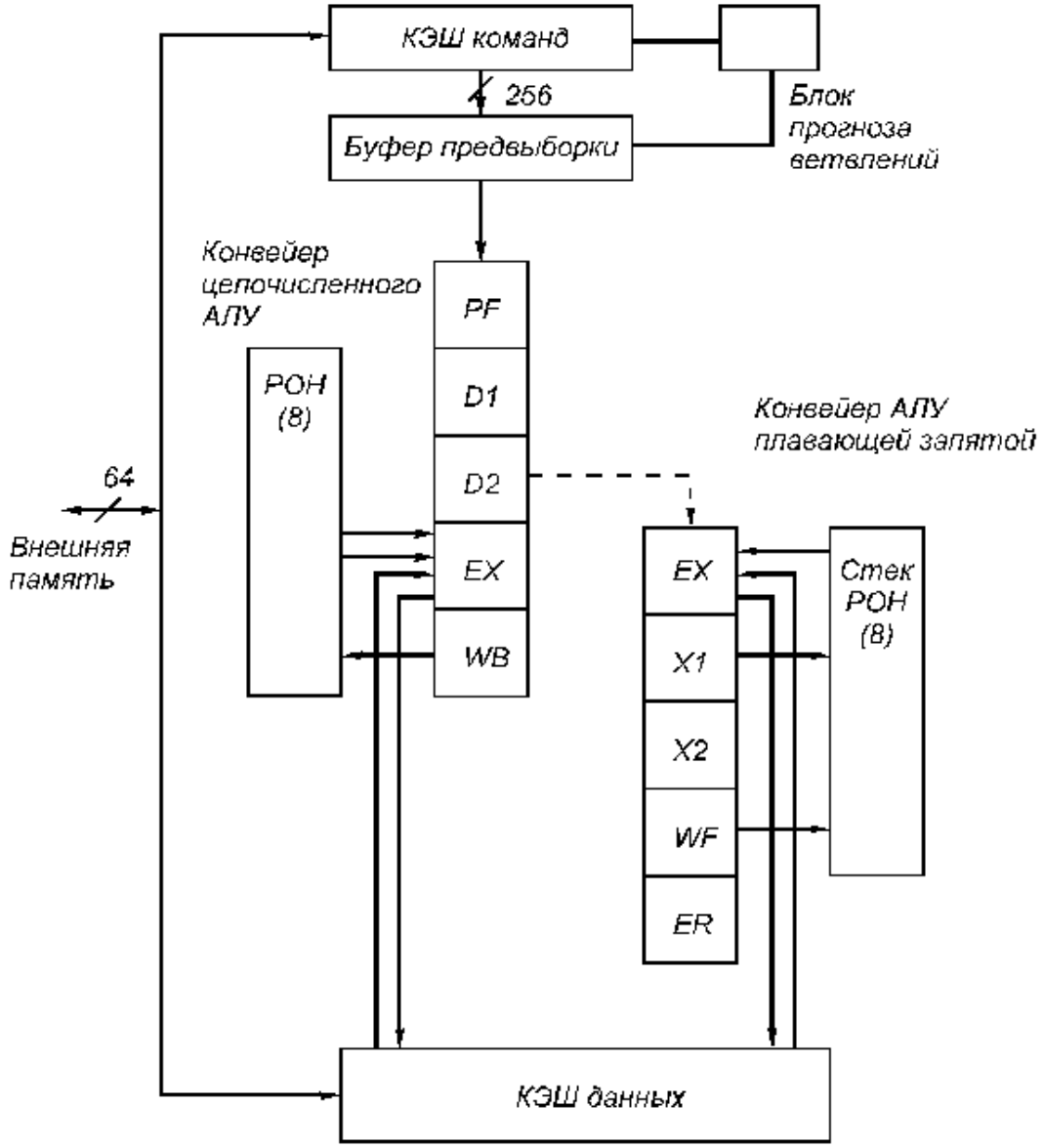


Рис. 5.5 – Целочисленный конвейер процессора Pentium

- стадия (ступень) предвыборки PF (Prefetch), которая осуществляет предварительную упреждающую выборку группы команд в буфер команд;
- ступень декодирования полей команды D1 (Decoder 1);

- ступень декодирования D2 (Decoder 2), на которой производится вычисление абсолютного адреса ячейки памяти операнда, если операнд расположен в физической памяти;
- на ступени исполнения EX (Execution) производится выборка операндов из РОН или памяти, выполнение операции АЛУ;
- на ступени записи результата WR (Write Back) производится передача полученного результата в блок регистров общего назначения.

Для конвейера АЛУ с плавающей точкой на стадии (ступени) EX производится чтение операндов из РОН или чтение памяти; на стадии X1 – выполняется часть операции АЛУ с плавающей точкой или запись в регистр РОН; X2 – продолжение выполнение команд АЛУ с плавающей точкой; WF – округление и запись результат в регистр РОН; на ступени ER (Error Reporting) выводится сообщение о наличии ошибок.

Следует обратить внимание на блок прогноза ветвлений на рис. 5.5. Он предназначен для формирования/предсказания адреса перехода в исполняемой программе на основании анализа ранее выполненных команд. В результате предсказанные команды сначала считываются из памяти в буфер предвыборки, а потом из буфера конвейер загружается предсказанными командами. При неверном предсказании содержимое конвейера сбрасывается и происходит возврат к тому адресу программы, начиная с которого был неверно предсказан переход.

В последнее время наблюдается переход к микропрограммной архитектуре EPIC, подразумевающей явно-параллельное выполнение программ. EPIC реализована на МПр Intel Itanium и Itanium 2. Распараллеливание потока команд здесь реализуется с помощью сложных, но эффективных компиляторов, встроенных в МПр в качестве хранимого ПО.

Указанные МПр имеют следующие особенности [33]:

- большое количество регистров (в случае Itanium2 имеем до 128 РОН, 128 регистров для операций с плавающей точкой, 8 регистров для хранения данных о переходах);
- поддержка параллельной обработки на уровне машинного кода;
- предсказание ветвлений (предикация);
- спекулятивное выполнение команд (загрузка команд в кэш по предположению о будущем направлении вычислений).

Конвейерная организация вычислений имеет следующие недостатки:

- возможны простои конвейера из-за наличия команд, которые требуют исполнения в АЛУ или на других ступенях конвейера за несколько тактов;
- возможны простои конвейера, если команды на разных стадиях используют одни и те же данные;
- возможны простои конвейера из-за аннулирования содержимого конвейера и повторной загрузки конвейера в случае ошибки при предварительном (спекулятивном) выборе направления условного перехода;
- ограниченная пропускная способность аппаратных средств РОН, памяти различных видов и шин.

Компания AMD, в отличие от компании Intel для решения задач распараллеливания использует в основном аппаратные средства. Рассмотрим особенности суперскалярной архитектуры более подробно.

5.3 Суперскалярная архитектура микропроцессора

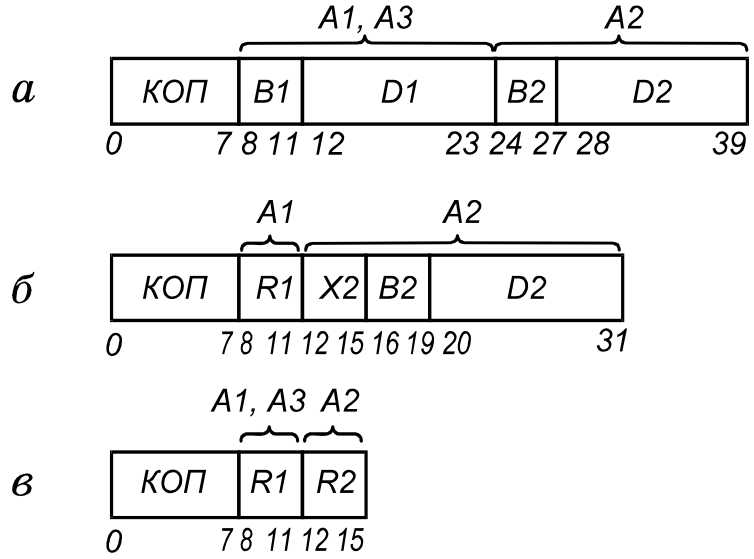
Суть ***суперскалярной архитектуры*** – наличие параллельной обработки данных с помощью двух или более конвейеров, как правило скалярных. Это позволяет оптимизировать загрузку АЛУ, уменьшить

потерю производительности, в результате появления пустых/ненагруженных стадий («пузырьков»).

Применение суперскалярной архитектуры обусловлено тем, что исходные данные для *i*-й операции вырабатываются заранее, например, при выполнении (*i* – 2)-й или (*i* – 3)-й операции. Тогда при соответствующем построении вычислительной системы можно совместить во времени выполнение *i*-й операции с выполнением (*i* – 1)-й, (*i* – 2)-й, … операций. В современных МПр данные каждого конвейера могут обрабатываться собственным АЛУ.

Суперскалярная архитектура может реализовываться аппаратными средствами, когда из кэш-памяти выбираются несвязанные между собой команды и запускаются параллельно. Параллелизм обнаруживается на стадии исполнения программы. Это характерно для МПр Pentium. В другом случае существенно меняется порядок подготовки программ к исполнению. С помощью специального распараллеливающего компилятора (встроенного программного обеспечения МПр) выполняется анализ программы на предмет команд, которые могут исполняться параллельно. Такие команды далее объединяются в пакеты команд – длинные командные слова (Very Long Instruction Word, VLIW). Каждая команда такого пакета запускается на исполнение на «свой» конвейер и соответствующе АЛУ, параллельно с другими командами. Использование встроенных в МПр компиляторов гарантирует отсутствие конфликтов при параллельной обработке. Здесь параллелизм обнаруживается на стадии компиляции программы. Существуют дополнительные форматы данных, которые используются в случае использования суперскалярной архитектуры процессора (см. рис 5.6) .

Преимущества суперскалярной архитектуры могут быть существенно улучшены с помощью изменения последовательности выполнения команд в МПр.



Условные обозначения :

КОП — код операции;
A1, A2, A3 — абсолютные адреса первого, второго операндов и результата;
R1, R2 — поля для указания номеров РОН, где размещены операнды;
Xi, *Bi* — поля для указания номеров РОН, где хранятся индексный и базовый адреса; *Di* — смещение

5.6, а — оба операнда размещены в памяти;
5.6, б — один операнд размещен в памяти, другой в РОН;
5.6, в — оба операнда размещены в РОН.

Рис. 5.6 – Структура команды конвейерной ЭВМ

Это достигается не только путём применения современных компиляторов но и управлением вычислениями в зависимости от последовательности команд или по мере готовности данных для вычислений. В последнем случае имеет место управление вычислениями на основе данных. Например, получая на входе операцию сложения, умножения и деления МПр может сначала выполнить наиболее сложную операцию деления, а потом операцию сложения и умножения. Результаты, тем не менее, выдаются в порядке предписанной исходной исполняемой программой. По данным Высшей компьютерной школы МГУ, Россия можно привести следующий пример повышения эффективности с помощью переупорядочивая команд внутри процессора.

Пусть в программе для ЭВМ встретилась некоторая последовательность команд :

A = B x 6

C = E x 12

A = A x D

C = C + 1,

где переменные A и C хранятся в оперативной памяти, переменные B, D и E – хранятся в регистрах процессора.

Последовательность действий процессора при выполнении данной последовательности команд следующая :

Шаг 1. Вычисляется значение Bx6 и результат записывается в регистр (условная длительность операции – 3 такта).

Шаг 2. Результат шага 1 считывается из регистра в оперативную память (условная длительность операции – 10 тактов).

Шаг 3. Вычисляется значение Ex12 и результат записывается в регистр (условная длительность операции – 3 такта).

Шаг 4. Результат шага 3 считывается из регистра в оперативную память (условная длительность операции – 10 тактов).

Шаг 5. Значение переменной A записывается из оперативной памяти в регистр (условная длительность операции – 10 тактов).

Шаг 6. Вычисляется значение AxD и результат записывается в регистр (условная длительность операции – 3 такта).

Шаг 7. Результат шага 6 считывается из регистра в оперативную память (условная длительность операции – 10 тактов).

Шаг 8. Значение переменной C записывается из оперативной памяти в регистр (условная длительность операции – 10 тактов).

Шаг 9. Инкрементируется регистр с результатом шага 8 (условная длительность операции – 1 такт).

Шаг 10. Результат шага 9 из регистра считывается в оперативную память (условная длительность операции – 10 тактов).

Итого всего 70 тактов и 10 шагов.

Теперь внутри процессора порядок следования данных команд изменяется следующим образом:

A = B x 6

A = A x D

C = E x 12

C = C + 1

Последовательность действий процессора при выполнении данной последовательности команд следующая :

Шаг 1. Вычисляется значение B x 6 и результат записывается в регистр (условная длительность операции – 3 такта).

Шаг 2. Вычисляется значение A x D и результат записывается в регистр (условная длительность операции – 3 такта).

Шаг 3. Результат шага 2 считывается из регистра в оперативную память (условная длительность операции – 10 тактов).

Шаг 4. Вычисляется значение E x 12 и результат записывается в регистр (условная длительность операции – 3 такта).

Шаг 5. Инкрементируется содержимое регистра с результатом операции на шаге 4 (условная длительность операции – 1 такт).

Шаг 6. Результат операции на шаге 5 считывается из регистра в оперативную память (условная длительность операции –10 тактов).

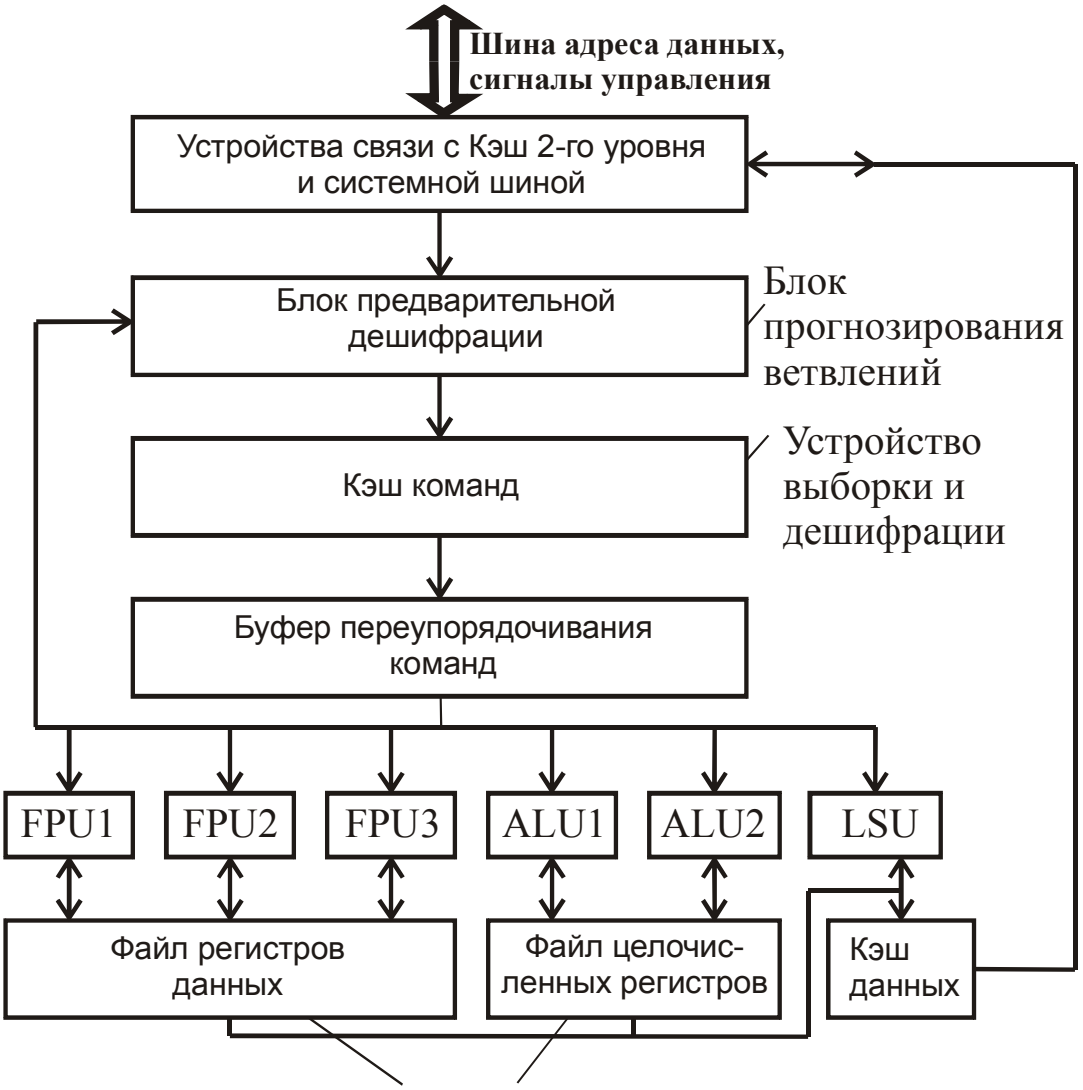
Итого всего 30 тактов и 6 шагов.

Как видно из примера, теоретически имеется возможность практически двукратного увеличения производительности МПр за счёт снижения числа тактов и количества шагов обработки данных.

Следует отметить, что структура команд в МПр с архитектурой EPIC отличается от команд архитектуры RISC и CISC. В МПр Itanium и Itanium2 в качестве формата используется связка команд, длиной 128 бит. Связка содержит три команды и т.н. шаблон, который указывает на существующие зависимости между командами. Компилятор, проанализировав шаблон устанавливает, можно ли запустить на исполнение

первую команду параллельно со второй или вторая команда должна выполняться сразу после первой.

Остальные свойства Itanium и Itanium2 в части оптимизации вычислений аналогичны перечисленным выше. В итоге структурная схема суперскалярного МПр на примере Intel Itanium имеет вид (см. рис. 5.7).



Условные обозначения :
ALU – арифметико-логическое устройство для операций с целыми числами
FPU – арифметико-логическое устройство для операций с плавающей точкой
LSU – блок предсказания перехода

Рис. 5.7 – Обобщённая архитектура суперскалярного процессора на примере Itanium [33]

Как видно из рис. 5.7, при использовании суперскалярной архитектуры МПр содержит несколько конвейеров и несколько АЛУ. Это позволяет одновременно исполнять смежные арифметико-логические операции, что соответствует реализации явного параллелизма при выполнении вычислений. Для разных операций – например с целыми числами и обработки мультимедиа АЛУ – конвейер имеет различную длину.

Для оптимальной загрузки конвейеров можно разработать методы, позволяющие с вероятностью до 95% (на примере Intel Pentium 4) предсказать направление условного перехода в программе. Эта вероятность повышается, если имеется информация о предыдущих переходах. Для реализации данной процедуры применяется блок прогнозирования ветвлений, который использует блок предварительной дешифрации команд. В результате команды загрузки данных из ОЗУ и/или кэш памяти выполняются задолго до команды (инструкции), использующей эти данные. Особенно это относится к командам условного перехода.

В процедурах предсказания переходов применяется т.н. «спекулятивное» исполнение команд. Суть ***спекулятивного исполнения*** состоит в том, что после сделанного предположения об адресе перехода МПр выполняет операции по предсказанному направлению перехода. Иными словами, МПр выполняет действия, не предписанные в данный момент загружаемой программой. При этом неизвестно, будет ли действительно, по мере исполнения программы, передано управление в выбранном направлении или нет. Тем не менее, обработка данных продолжается. Если управление на самом деле будет передано на предсказанное направление, то результаты выполнения команд будут уже готовы. Если управление будет передано в другое место программы, то полученные ранее результаты уничтожаются. Последствием ошибки предсказания перехода может быть образование нескольких пустых стадий «пузырьков» в конвейере, что снижает производительность МПр.

Также различают статическое и динамическое предсказание ветвления в программе. При ***статическом предсказании*** направление перехода задаётся разработчиком МПр, например все условные переходы «вперед» будут выполняться, а переходы «назад» – не будут. При ***динамическом предсказании*** направление ветвления обусловлено результатами предшествующего выполнения команд и может меняться в процессе исполнения программы. Динамическое предсказание более точно и эффективно, хотя и достаточно сложно для реализации.

В динамике управление переходами осуществляется следующими способами:

1. Самый простой способ состоит в том, чтобы остановить прием команд на вход конвейера до тех пор, пока команда перехода не достигнет ступени исполнения и не будет вычислено реальное направление перехода. Этот метод, упрощая управление, тем не менее приводит к простою конвейера.

2. Наиболее сложный, второй способ, заключается в том, чтобы при появлении команды перехода выбирать и условно выполнять обе ветви возможных переходов. Условное выполнение обозначает, что до момента определения адреса перехода запрещается запись результатов вычислений в регистры или в кэш/ОЗУ.

3. Третьим способом динамической обработки ветвлений является предсказание. Самым простым способом предсказания является выбор для выполнения той ветви, которая следует непосредственно за командой перехода (короткий переход). Но в 50% случаев этот выбор будет ошибочным.

Наиболее достоверные предсказания выдаются на основе истории процесса вычислений. Поэтому на рис. 5.7 присутствует устройство прогнозирования ветвлений. Оно содержит таблицу достаточно большого размера, например, на 256 строк. В каждой строке таблицы записаны для выполнения части программы:

1. Адрес команды перехода.

2. Адрес дальнего перехода.

3. Бит «истории», который указывает, по какому направлению произошел переход при последнем использовании команды перехода.

Буфер упреждающей выборки обычно содержит от нескольких до нескольких десятков предварительно выбранных команд, чтобы сгладить задержки, связанные с обращениями в память. Одновременно с подачей команд на вход конвейера устройство управления производит в буфере предварительной выборки просмотр вперед выбранных команд. Если при просмотре обнаружена команда перехода, то по таблице «истории» определяется направление перехода. Буфер упреждающей выборки содержит две зоны: для текущей и альтернативной ветви; поэтому переключение с зоны на зону не вызывает простоев. Описанный механизм ветвления позволяет выбирать правильные пути ветвления с вероятностью более 80%.

Устройство управления функционирует на основе предположения, что при повторном выполнении одной и той же команды перехода переход будет осуществлен по одному и тому же адресу. В соответствии с этим в буфер упреждающей выборки выбирается ветвь, предписанная битом «истории», если этой ветви в буфере упреждающей выборки еще нет.

Перечисленные способы в совокупности позволяют уменьшить простои МПр, связанные с ожиданием загрузки команд и данных из относительно медленной памяти. Компилятор перемещает команды загрузки из ОЗУ так, чтобы они выполнялись как можно раньше.

Реализацию описанных технологий оптимизации обработки данных в современных МПр в увязке с проблемами энергосбережения рассмотрим ниже.

5.4 Технологии оптимизации вычислений и встроенного энергосбережения

В настоящее время широко используются возможности изменения тактовой частоты и энергопотребления МПр в зависимости от характера вычислительных задач (например, технология Speed Step, предложенная Intel). Эти технологии дополняются возможностями, связанными с декодированием инструкций и предсказанием переходов. Результат хорошо виден на примере МПр типа Pentium M (см. рис. 5.8) [19].

Процесс обработки данных в МПр типа Pentium M состоит из следующих этапов :

- Выборка команд (инструкций) и данных из кэша L2.
- Декодирование инструкций в примитивы (микрокоманды).
- Передача декодированных микрокоманд в блок исполнения, выполнения вычислений.
- Запись результатов в ОЗУ.

С учётом повторяемости некоторых вычислительных процедур, некоторые типовые, последовательно выполняемые микроинструкции хранятся в специальном кэш L1, где из них формируются микропрограммы – отслеживания (traces, трассы). Под ***трассой*** понимается последовательность микрокоманд, в которые декодированы ассемблерные команды МПр, принадлежащие одной или нескольким ветвям исходной программы. Если в кэш L1 попадают инструкции (микрокоманды), совпадающие с трассой, то такие инструкции выполняются, даже если порядок их следования не совпадает с трассой. Для обеспечения высокого процента попаданий используется специальный блок предсказания ветвлений Branch Unit. Этот блок позволяет на основе ветвлений в программе предсказывать переходы. При этом модифицируется порядок выполнения команд в исходной программе. Для предсказания в МПр Pentium M используется технология Advanced Branch Prediction, которая увеличивает точность предсказания на основе

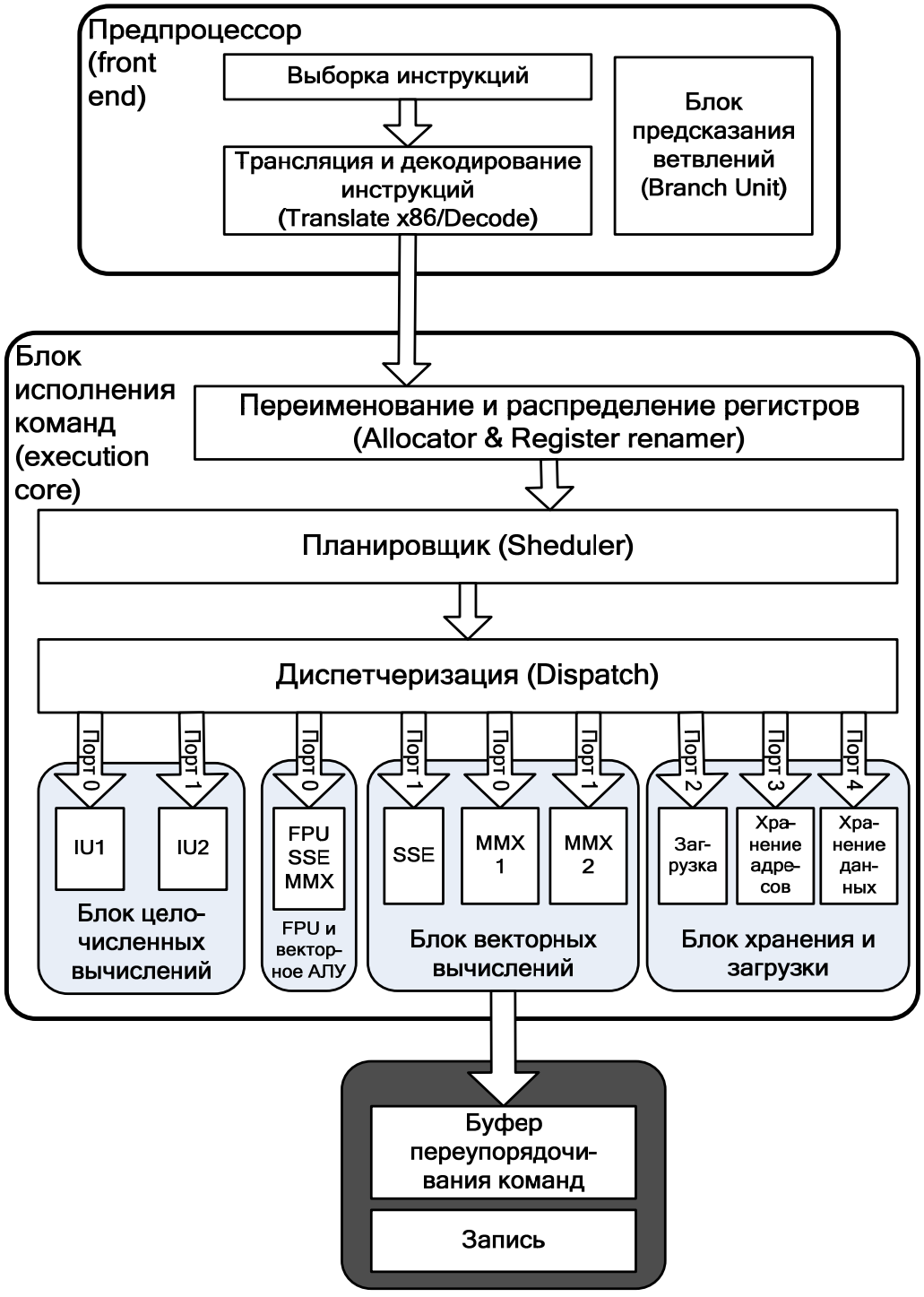


Рис. 5.8 – Структурная блок-схема процессора Pentium M [21]

анализа поведения программ в прошлом и на этой основе предсказывает будущее. В частности, используются :

- Бимодальное прогнозирование – прогноз условных и безусловных переходов.

- Локальное прогнозирование – прогнозирование циклов, чьё завершение связано со счётчиком.
- Глобальное прогнозирование – общее прогнозирование ветвлений путём отслеживания выполнения программ.

В результате наличия этих трёх блоков вероятность ошибки прогнозирования снижается на 20%. Используемая технология предвыборки данных требует анализа L1 и поиска в потоке запросов тех данных, к которым будет происходить обращение. Pentium M отслеживает одновременно 8 операций «потоки вверх» (переход от низших адресов к высшим), так и 4 операции «поток вниз» (переход от высших адресов к низшим).

В блоке «Переименование и распределение регистров МПр» происходит переименование и распределение регистров МПР для бесконфликтного доступа.

Планировщик переупорядочивает инструкции (микрокоманды) и распределяет их по функциональным устройствам. Суть переупорядочивания заключается в том, что планировщик определяет степень готовности команд к исполнению и меняет порядок следования команд в зависимости от готовности данных для выполнения команд.

Диспетчер перераспределяет микрокоманд по портам, которые выполняют функции шлюзов к блокам вычислений. Для окончательного выполнения инструкции загружаются в блок регистров.

Микропроцессор типа Pentium M использует технологию объединения микроопераций (micro-op fusion), которая позволяет осуществлять слияние нескольких микроопераций в одну, где микрооперации – декодированные инструкции. Инструкции объединяются с операндами. В целом микрокоманда выполняется за время выполнения самой длительной инструкции. Используется выделенный диспетчер стеков (dedicated stack manager), который отслеживает загрузку системных ресурсов с помощью аппаратных средств.

МПр Pentium M поддерживает технологию энергосбережения Intel Speed Step. Эта технология предусматривает использование нескольких возможных напряжений питания и частот (рабочих точек). Крайний рабочие точки задаются аппаратно а промежуточные – устанавливаются программно. Управление переходом между точками осуществляется как самим МПр так и регулятором напряжения VRM (Voltage Regular Module). МПР посылает в VRM специальные последовательности.

Для увеличения тактовой частоты сначала увеличивается напряжение электропитания. Период изменения напряжения длится около 100 мкс. После изменения напряжения электропитания скачкообразно увеличивается частота за время 10 мкс. При уменьшении тактовой частоты сначала уменьшается тактовая частота, и только потом снижается напряжение электропитания.

Некоторые данные о МПр Pentium M и более поздних моделях энергоэффективных процессоров Intel приведены в таблице 5.1.

Таблица 5.1 – Характеристики МПр Intel с эффективным энергопотреблением

№№ п/п	Наименование МПр и технологическая норма производства	Макс. тактовая частота, МГц	Мин. тактовая частота, МГц	Частота системной шины, МГц	Потребляемая мощность, Вт
1	Intel Pentium M 770, 90 нм	2130	800	533	27
2	Intel Pentium M 725, 90 нм	1600	600	400	21
3	Low voltage (пониженное энергопотребление) Intel Pentium M 758, 90 нм	1500	600	400	10
4	Ultra low voltage (сверхнизкое энергопотребление) Intel Pentium M 753, 90 нм	1200	600	400	5
5	Intel Core Duo U2500 (сверхнизкое энергопотребление) двухъядерный, 65 нм	1200	600	533	9
6	Intel Core Solo U1400 (сверхнизкое энергопотребление),65 нм	1200	600	533	5,5

Как видно из таблицы 5.1, развитие конструкций процессоров для мобильных устройств привело к появлению низковольтных процессоров для информационно-вычислительных систем с потреблением от

20% до 70% электроэнергии меньше, чем однотипные процессоры для стационарных вычислительных систем.

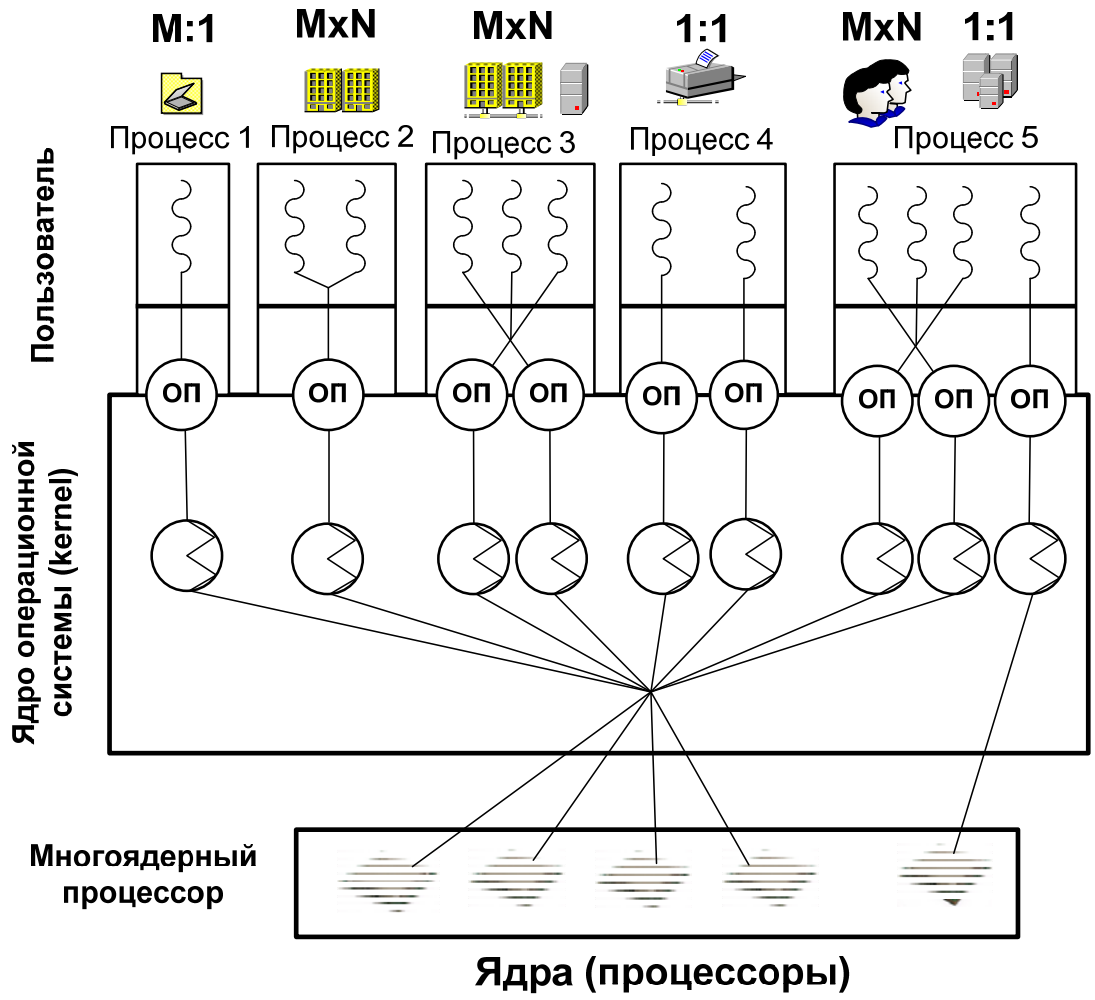
Компания Intel работает над двумя новыми методами проектирования и производства микропроцессорных систем, направленными на повышение энергетической эффективности транзисторов и уменьшение потребляемой энергии системных плат. Один из методов предусматривает использование отдельного напряжения питания для центрального процессора и кэш-памяти. У центрального процессорного устройства будет свое напряжение питания, а у кэш-памяти — свое напряжение питания. Это позволит конструкторам систем исключить из микропроцессорного набора интегральные микросхемы, регулирующие подачу напряжения для этих компонентов в современных процессорных системах и высвободить физическое пространство для размещения новых компонент на системной плате, исключив лишнюю нагрузку на источник питания.

Другой способ снижения общего энергопотребления заключается во введении стабилизаторов напряжения в микросхемы для повышения их эффективности; в результате уменьшается энергия, потребляемая микросхемным набором и другими компонентами, расположенными на системной плате. Это достигается путем введения стабилизаторов напряжения непосредственно в микросхемы, вместо использования отдельных аналоговых компонентов.

5.5 Многопоточная обработка данных и многоядерные процессоры

Повысить производительность современных МПр можно за счёт выполнения в параллельном режиме нескольких потоков вычислительных задач. Можно одновременно принимать информацию сети Интернет, обеспечивать приоретизацию и антивирусную проверку принятых сообщений, поддерживать виртуальные машины.

Многоядерные процессоры на аппаратном уровне поддерживают архитектуру **MIMD** *(multiple instruction multiple data)* — много потоков команд, много потоков данных. Наиболее эффективен физический параллелизм, при котором каждый из потоков команд и/или данных обрабатывается собственным ядром (core), где ядро – это самостоятельное ЦПУ (CPU) с АЛУ, регистрами и кэш-памятью L1. Каждое ядро поддерживает конвейерные вычисления, в первую очередь – целочисленные конвейеры. Проиллюстрируем сказанное т.н. «диаграммой Рута» на рис. 5.9.



Условные обозначения :

- потоки процессов пользователей,M

- внутриядерные потоки, N

ОП - облегчённый процесс

Рис. 5.9 – Диаграмма Рута [28]

Как видно из диаграммы на рис. 5.9, исполнение программных приложений пользователей порождает выполнение прикладных процессов. Прикладные процессы, в свою очередь, порождают потоки команд и потоки данных. Эти пользовательские потоки с помощью специальной библиотеки потоков (thread libraries), входящих в состав системного программного обеспечения, преобразуются в облегчённые процессы, а потом – во внутриядерные процессы операционной системы (kernel level treads). Именно эти внутриядерные потоки и выполняются собственно процессорами (ядрами). Такое многоэтапное разделение исходных задач на элементарные потоки обусловлено стремлением разработчиков таких систем упростить задачу согласованного и синхронного выполнение многих потоков. Кроме того, дробление позволяет более эффективно распределять потоки между ресурсами процессоров.

Потоки пользователей находятся внутри процессов. На рис. 5.9 показаны различные способы преобразования потоков во внутриядерные процессы:

M:1 – все потоки пользователей преобразуются в один ядерный (процесс 1);

1:1 – каждому потоку пользователей соответствует один ядерный (процесс 4);

M:N – M потокам пользователей соответствует N ядерных потоков (процесс 2,3).

Процесс 5 относится к гибридным.

Потоки и процессы имеют различный способ представления в операционной системе. Процессы имеют идентификаторы процессов, идентификаторы пользователей процессов. Процессы используют каталоги, дескрипторы файлов, средства межпроцессной коммуникации.

Обработка потоков проще и основана на приоритетах. Потоки могут создаваться и уничтожаться. Они делятся на два типа :

- ядерные потоки – используются для обработки недоступных пользователю прерываний;
- пользовательские потоки – применяются для организации управления выполнением задач.

Доступ к потокам осуществляется с помощью прикладного программного интерфейса API, который реализуется с помощью библиотек потоков. Библиотеки потоков опираются на рассмотренные выше стандарты POSIX или на стандарты разработчиков, которые предлагает к примеру, компания SUN Microsystems. Эти библиотеки позволяют автоматизировать функции создания и уничтожения потоков, управление расписанием потоков, передача данных и сообщение между потоками.

Возможная архитектура многопотокового сетевого процессора (по данным P. Crowley, M. Fiuczynski, J.-L. Baer) представлена на рис. 5.10.

Данная архитектура содержит множество идентичных многопотоковых процессоров, имеющих собственную кэш-память инструкций и кэш-память данных. Каждая область кэш-памяти разделена между потоками вычислений, которые могут обрабатываться аппаратными средствами каждого процессора. Переключение контекста вычислений предусмотрено в аппаратных средствах с нулевым циклом издержек. Это означает, что, если один поток терпит неудачу при обращении к области кэш-памяти, то другой поток может немедленно начать обрабатываться без задержки машинного цикла обработки.

Для того, чтобы оперативная память вне кристалла сетевого процессора соответствовала требованиям предъявляемым к скорости обработки данных, процессоры объединяются в группы (кластеры, clusters) и используют общий интерфейс доступа к памяти вне кристалла сетевого процессора. Планировщик задач направляет пакеты с независимых потоков на различные процессоры, для того, чтобы увеличить скорость и в максимальной степени использовать технологию параллельных вычислений. Таким образом, после перенаправления пото-

ка вычислений, к примеру, на процессор 1, все пакеты данного потока вычислений обрабатываются тем же процессором 1.

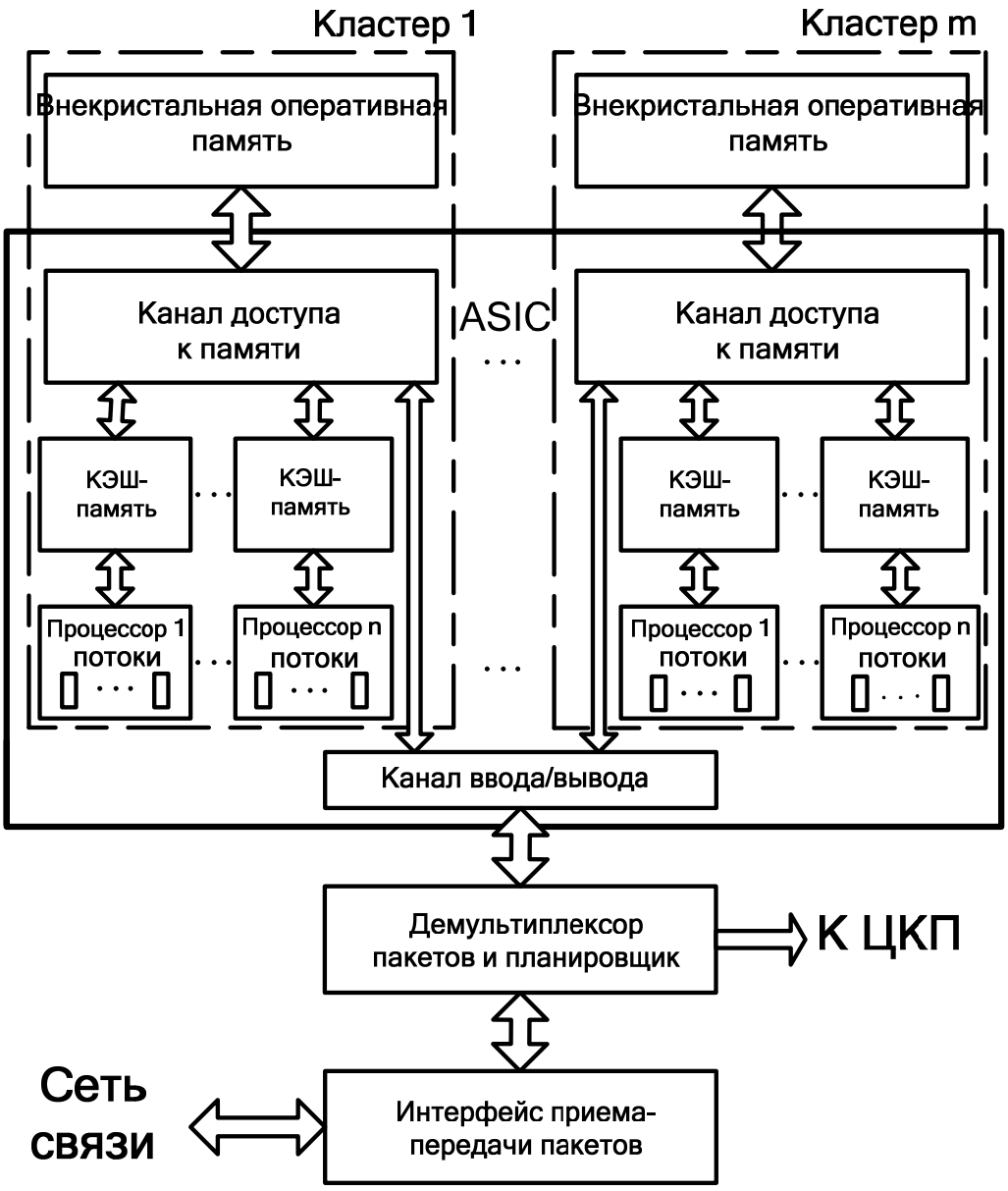


Рис. 5.10 – Архитектура многопотокового сетевого процессора

Другой МПр, относящийся к MIMD, тип Niagara (реализован в виде универсального RISC МПр типа UltraSPARC T1, производство компании SUN Microsystems, США) обеспечивает аппаратную поддержку выполнения 32 потоков, которые разделены на восемь групп (по четыре потока в каждой группе). МПр Niagara в перспективе будет поддерживать на уровне ядра 8-ми стадийный конвейер для целочисленных вычислений

и 12-ти стадийный конвейер для вычислений с плавающей запятой. Как уже говорилось, многопотоковые вычисления наиболее целесообразно выполнять параллельно на физическом уровне. Физический параллелизм обеспечивается тем, что каждый из потоков команд и/или данных обрабатывается собственным ядром. Такой подход описывается в рамках архитектуры ***CMP*** (chip multiprocessors). Общая схема возможной эволюции многоядерных МПр представлена на рис. 5.11.

В настоящее время на одной кремниевой пластине реализуется 2..4 процессорного ядра. В перспективе на кремневой пластине будет реализовано до 8…16 процессорных ядер. Каждый из потоков команд и/или данных в данный момент времени может обрабатываться собственным ядром.

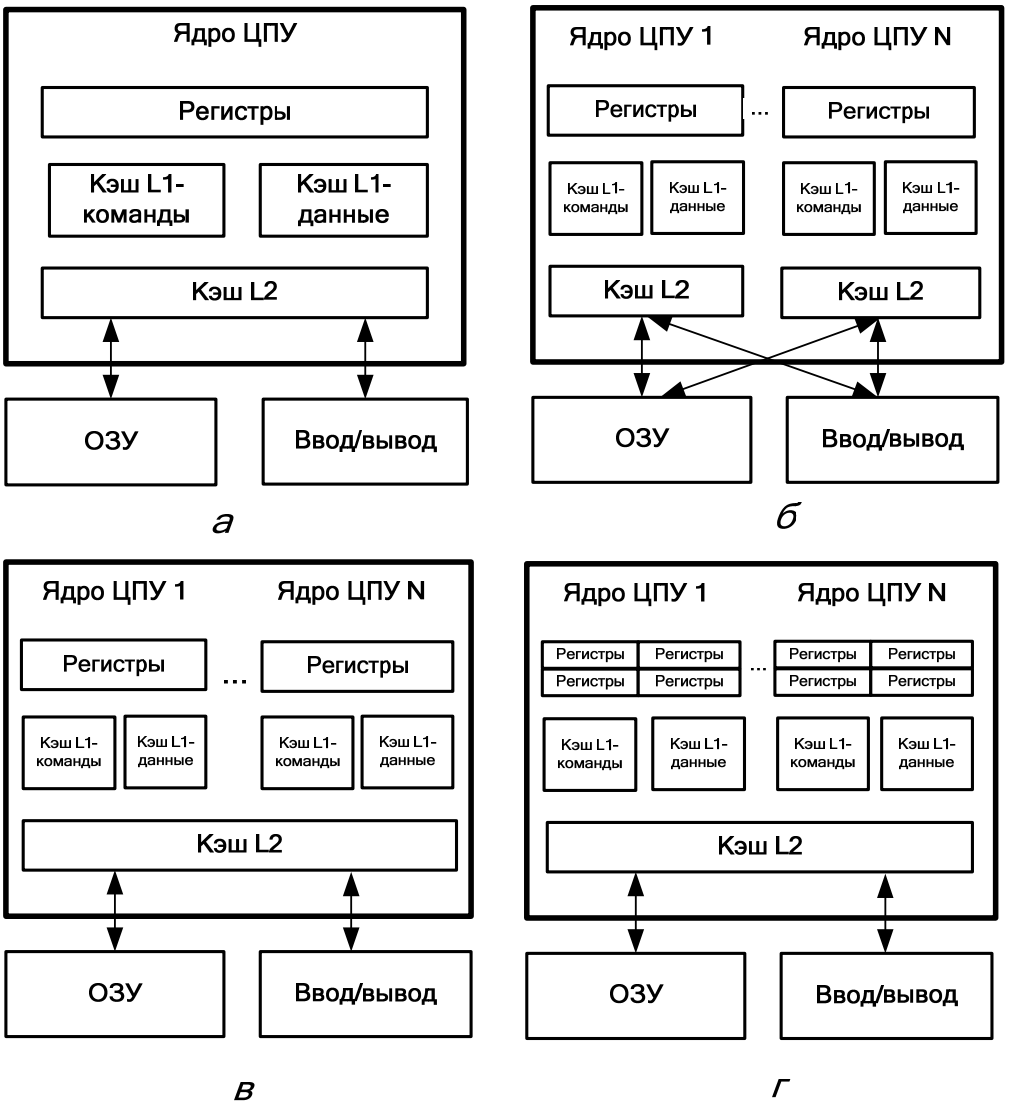


Рис. 5.11 а, б, в, г – Эволюция архитектур многоядерных CMP- процессоров

К примеру, восьмиядерный процессор на основе Niagara может одновременно обрабатывать 8 потоков, а всего восьмиядерный МПр может обрабатывать 64 потока. Каждому потоку назначается собственный набор регистров, благодаря чему нет необходимости тратить время на обращение к ОЗУ для сохранения текущего состояния вычислений. С помощью специального алгоритма реализуется переключение между потоками, причём на исполнение запускается тот поток, который дольше всех ожидал в очереди. Следует отметить, что согласно [35], ресурсы МПр выделяются потокам динамически. Если ядро МПр работает на частоте 1,2 ГГц, то в случае обработки 4 потоков каждый из них будет выполняться на частоте 300 МГц, в случае обработки 2 потоков каждый будет выполняться на частоте 600 МГц.

Многоядерный процессор, по сути, это многопроцессорная система, реализованная на кристалле, обеспечивающая повышение эффективности работы вычислительной системы в целом. Из закона Амдала следует, что прирост производительности (*R*) вычислительной системы зависит от количества процессоров (*N*) и доли последовательных операций в программе ($c=a$, где a определено в формуле 5.2):

$$R \le \frac{1}{(c + \frac{1-c}{N})}.$$

Граничные значения переменной c соответствуют полностью параллельным $(c=0)$ и полностью последовательным $(c=1)$ программам. Если лишь 1/10 часть программы выполняется последовательно, то в принципе невозможно ускорение в десять раз — вне зависимости от числа используемых процессоров (ядер). Важное следствие закона Амдала состоит в том, что максимальный рост производительности в N раз при N ядрах недостижим. В противном случае последовательно ис-

полняемая часть программы должна быть равна нулю, что практически невозможно. Прирост вычислительной производительности системы при использовании многоядерных процессоров показан на рис. 5.12.

Многопоточная обработка данных и команд многоядерными процессорами является альтернативой рассмотренной ранее системе переупорядочивания команд для ускорения вычислений и предсказания направлений вычислений как средства борьбы с задержками при реализации вычислений. Переупорядочивание команд и предсказание вычислений требуют применения достаточно сложных алгоритмов, требует дополнительных вычислительных ресурсов и в итоге приводит к росту энергопотребления и выделяемого тепла.

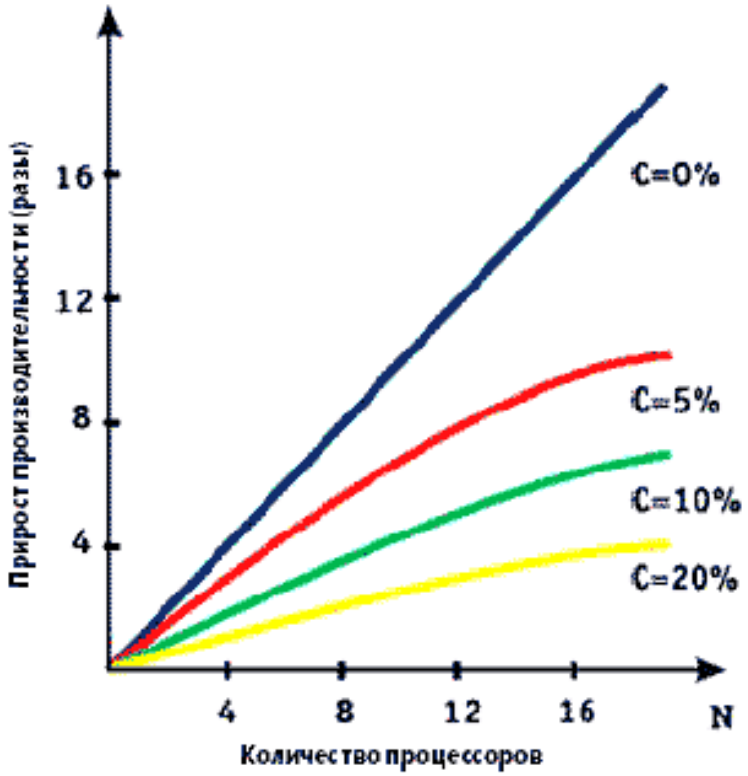


Рис. 5.12 – Изменение производительности многоядерных процессоров
[Источник http://www.osp.ru/text/302/1156508.html]

В случае многоядерных процессоров, например на прототипе Niagara, проблема задержек вычислений решается иначе. Если один из потоков задерживается на выполнении инструкции, то ядро может переключиться на выполнении другого потока. Тоже самое относится и к проблеме ветвлений – если достигнута команда перехода, то ядро не

вычисляет потенциальный переход а происходит переключение на другой поток до тех пор, пока условие перехода не будет выполнено. При это потоки являются независимыми и ядро не анализирует взаимозависимости между инструкциями различных потоков.

На середину 2006 года двухъядерные МПр производства компаний Intel и AMD использовали в основном отдельную кэш–память; компания IBM выпускает двухъядерные МПр с объединенной кэш-памятью.

В большинстве случаев частота процессора примерно пропорциональна напряжению питания, а активная выделяемая мощность – квадрату этого напряжения. В результате при переходе от одноядерной архитектуры процессора к двухъядерной можно сохранить тот же уровень производительности, снизив частоту каждого из процессоров и напряжение электропитания почти вдвое. При этом суммарное тепловыделение процессора снизится в четыре раза.

Достоинства конструкций многоядерных МПр состоят в следующем:

- малые размеры «ядра» МПр уменьшаются так, что в одном конструктиве можно «упаковывать» больше «процессоров», повышая тем самым удельную вычислительную мощность на единицу площади МПр;
- при сохранении той же производительности МПр можно вдвое уменьшить тактовую частоту;
- поскольку процессоры, находящиеся на одном кристалле, используют общие системные ресурсы, возникает дополнительная экономия физического пространства;
- при уменьшении тактовой частоты и сложности одного ядра существенно сокращается потребление электроэнергии.

Недостатком многоядерных МПр является усложнение проектирования и изготовления микропроцессора. Однако, если есть проработанное ядро (ЦПУ), то оно может тиражироваться в нужных количест-

вах, а проектирование ограничивается созданием внутренней инфраструктуры кристалла.

Многоядерные процессоры могут использоваться в коммутаторах и многопротокольных маршрутизаторах т.к. за счёт распараллеливания вычислений они могут быть оптимизированы для управления трафиком сетей при использовании таких технологий как VoIP, IPTV, Web.2.0.

Особое место в многоядерной архитектуре занимает работа с памятью. В среднесрочной перспективе компания Intel предлагает каждое ядро соединить непосредственно с микросхемой памяти емкостью 256 Мбайт при помощи специальной технологии Through Silicon Vias (TSV). Эта технология представляет собой специальные электрические проводники, соединяющие МПр с памятью. Сейчас память и МПр обмениваются данными через контроллер памяти и соответствующую шину (см. рис. 1.2 и 1.3), которые работают значительно медленнее, чем МПр. Это одно из основных узких мест архитектуры фон Неймана. Технология TSV, которая начиная с 2006 г. Проходит стадию лабораторного тестирования, возможно, сможет заменить контроллер памяти, что приведёт к увеличению скорости обмена «процессор–память». Пока соответствующие испытания проведены для статической памяти SRAM, в перспективе – испытания для динамической памяти DRAM. Кроме того, предстоит решить проблему объединения МПр и памяти в единый корпус, а также решить традиционную проблему теплоотвода. Компания SUN также предполагает использовать усовершенствованную конструкцию ОЗУ, что увеличит скорость обмена МПр – память.

Рассмотрим некоторые практические примеры реализации многоядерных процессоров.

Одна из первых двухъядерных платформ производства компании Intel включает в себя процессор Intel Pentium Extreme Edition 840 с тактовой частотой 3,2 ГГц и набор микросхем Intel 955X Express. Частота системной шины МПр, построенном на основе 90-нанометровой техно-

логии, составляет 800 МГц. Среди других характеристик этого решения — 2 МБ кэш-памяти 2-го уровня (по 1 МБ на каждое ядро МПр), поддержка технологии Intel Extended Memory 64, Hyper-Threading. Размер кристалла МПр составляет около 206 кв. мм, количество транзисторов — около 230 млн. В другом двухъядерной процессоре типа Xeon Tulsa, делается упор на производительность. Этот МПр будет работать с тактовой частотой 3,4 ГГц. МПр Tulsa оснащен 16-Мбайт объединенной кэш-памятью — каждое ядро может обращаться к данным из общего кэша. Конструктивная тепловая мощность (TDP, или тепловой потолок) МПр Tulsa составляет 150 Вт. Это будет первый процессор Intel с технологией виртуализации Pellston, которая позволит процессору исполнять сразу несколько операционных систем. В МПр Tulsa будет применяться технология экономии энергии. Кэш-память можно вводить в состояние «сна» и «глубокого сна», экономя таким образом до 6 Вт потребляемой энергии на один МПр.

Следует отметить, что в современных МПр кэш-память большую часть времени включена и неизбежно дает утечку тока. Поэтому, например, Компания IBM в свою очередь предполагает повысить эффективность кэш-памяти МПр, применяя вместо технологии SRAM решения, основанные на DRAM. При этом площадь кристалла МПр, отводимая под временное хранение обрабатываемых данных, возможно, уменьшится в 3 раза, а энергопотребление в пассивном (ненагруженном) режиме сократится до 5 раз. Норма проектирования такого МПр составляет 45 нм.

В конце 2006 г. Intel анонсировала выпуск четырехядерного процессора Intel Core2 Quad Processor. Этот процессор ориентирован на применение в персональных компьютерах, имеет тактовую частоту 2,4 ГГц, частота общесистемной шины 1,066 ГГц, размер кэш-памяти L.2 составляет 8 Мбайт, технологическая норма производства 65 нм, рассеиваемая тепловая мощность 105 Вт, напряжение питания 1,1….1,372

В, рабочая температура процессора 62,2 градуса по Цельсию. Процессор поддерживает 64-х разрядную архитектуру.

В августе 2007 г. компания Sun Microsystems сообщила о выпуске процессора UltraSPARC T2 с тактовой частотой 1,4 ГГц., который основан на архитектуре UltraSPARC T1, но вдвое больше по производительности. Новый МПр основан на решениях Niagara. В производстве МПр использовалась технология 65 нм, процессор имеет 8 ядер, каждое из которых поддерживает до 8 потоков. Таким образом, общее количество поддерживаемых потоков доходит до 64; каждое ядро обладает собственным модулем операций с плавающей точкой. Кэш-память второго уровня составляет 4 МБ. МПр имеет два встроенных Ethernet-контроллера поддерживающих скорость обмена данными 10 Гбит/с. МПр потребляет UltraSPARC T2 около 2 Вт на один поток а максимальное энергопотребление всего устройства составляет 120…130 Вт.

Как уже говорилось, компания IBM,США разработала двухъядерный процессор общего назначения IBM POWER6. Микропроцессор POWER 6 содержит два обрабатывающих ядра, у каждого из которых своя высокоскоростная кэш-память второго уровня L2 емкостью 4 Мбайт. Два ядра также могут совместно использовать кэш L3 ёмкостью 32 Мбайта; при этом кэш L3 конструктивно находится вне кристалла микропроцессора, хотя контроллер управления L3 находится на кристалле МПр. Также на кристалле МПр находятся два контроллера оперативной памяти ОЗУ. Каждое ядро может одновременно управлять двумя потоками инструкций. В целях виртуализации POWER6 можно разделить от 2 до 1024 отдельных виртуальных сегментов, каждый из которых сможет работать со своей операционной системой. Конструкторы реализовали в МПр POWER 6 механизм обнаружения и исправления максимального количества ошибок. Этот механизм направлен на то, чтобы ошибки были выявлены раньше, чем они приведут к сбою в работе программного обеспечения. Для этого на каждом цикле МПр ре-

гистрирует состояние всех хранящихся в регистрах данных; в случае ошибки МПр возвращается к предыдущему состоянию и повторяет последний шаг вычислений. При более серьезных ошибках все текущее состояние процессора (команды, данные) могут быть переданы в другое ядро — так называемое «горячее резервирование центрального процессора». Аналогичный механизм ранее рассматривался в процессоре CP113 в процессорах BAP и CAP.

5.6 Контрольные вопросы к главе 5

1. Опишите основные особенности конвейерной обработки данных.
2. Может ли у процессора быть несколько конвейеров ?
3. Почему суперскалярная архитектура процессора в последнее время получает широкое распространение и применение?
4. В чём заключается сущность закона Амдала?
5. Почему в современных процессорах повышенное внимание уделяется вопросам энергоэффективности ?
6. Что увеличивается в первую очередь – напряжение электропитания процессора или тактовая частота ?
7. Для чего в современных процессорах применяется функция предсказания переходов?
8. Что такое «спекулятивное» предсказание переходов?
9. В чём особенность архитектуры многоядерных процессоров?
10. Укажите достоинства и недостатки многоядерных процессоров.

Рекомендуемая литература

1. *Артемьев М.Ю. Самоделов В.П.* Программное обеспечение управляющих систем электросвязи: Учебник для техникумов.– М.: Радио и связь – 1990 – 272 с., ил.

2. АТС как система реального времени. Табличные методы разработки программного обеспечения АТС: Метод. Указания к лаб. Работам (спец. 200900)/ В.А. Манохин и др. – СПб.– СПбГУТ.–2002. Режим доступа [http://dvo.sut.ru/libr/skiri/i126manh/index.htm]

3. Архитектура микропроцессоров Motorola M68K: Уч. Пособие// Губарев С.И., Росинский Д.Н., Руденко О.Г., Михалев А.И.- Днепропетровск: Национальная металлургическая академия Украины, кафедра ИТС, 2002. Режим доступа [http://dmeti.dp.ua/kaf/k-its/books/]

4. *Басалин П.Д.* Архитектура вычислительных систем.: Учебник.– Нижний Новгород.– Издательство Нижегородского госуниверситета.–2003.– 243 с.

5. *Бойко В.И.* и др. Схемотехника электронных систем. Микропроцессоры и микроконтроллеры: Учебник.– БХВ–Петербург, 2004.– 464 с.

6. *Гольдштейн Б.*С. Системы коммутации: Учебник для ВУЗов.–БХВ–Санкт–Петербург, 2003.–318 с.

7. *Гребешков А.*Ю. Управление сетями электросвязи по стандарту TMN:Учеб. пособие с грифом УМО.– М.: Радио и связь, 2004. – 155с.

8. *Ершова Н.Ю., Ивашенков О.Н., Курсков С.Ю*. Микропроцессоры.: Учебное пособие в электронном виде. – Петрозаводский государственный университет. Режим доступа [http://dfe3300.karelia.ru/koi/posob/microcpu/vved.htm]

9. *Зарубин А.А.* Микропроцессорное программное управление. Архитектура IXA: методические рекомендации к практическим занятиям (спец. 200900)/ СПбГУТ.– СПб, 2004.

10. *Золотарев С., Горбунов Н.* POSIX и ОС РВ : попытка систематизации.// PC Week RE. – 2005. – №10(472). – С.39– 41.

11. Использование микропроцессоров в системах управления узлов коммутации, часть 1.: Метод. указ. к лабор. работам// Сост. Т.Б. Мурашова; под ред. проф. Карташевского В.Г. – Самара.: ПГАТИ, 1997 г.–27 с.

12. Координационный процессор CP113 системы EWSD: Метод. указ. к лабор. работам на ПЭВМ.// Гребешков А.Ю. по материалам компании Siemens. – Самара: Изд-во ПГАТИ, 2003.– 67 с.

13. *Копейкин М.В, Спиридонов В.В, Шумова Е.О.* Организация ЭВМ и систем. Память ЭВМ. – Уч. пособие в электронной версии. – Сев. Зап. Гос. Заочный тех. Университет, ООО «ВЕДИ» – 2003. Режим доступа [http://ord.com.ru/files/book3/]

14. *Корнеев В.В, Киселёв А.В*. Современные микропроцессоры. – 3-е изд., перераб. И доп. – СПб.: БХВ–Петербург, 2003. – 448 с.

15. *Кустарев П.В.* Специализированные процессоры. Процессоры для встраиваемых приложений: Конспект лекций.– С-Пб.: СПбГИ – МО(ТУ).– 2002.– 30 с.

16. *Лепешинский В.Н.*Лабораторный практикум по курсу «Микропроцессоры и микрокомпьютеры» для студентов специальности 53 01 02 «Автоматизированные системы обработки информации» дневной формы обучения. //Белорусский государственный университет информатики и радиоэлектроники, Минск, 2002.

17. *Мелехин В.Ф.* Вычислительные машины, системы и сети : учебник для студ. высш. учеб. заведений / В.Ф. Мелехин, Е.Г. Павловский. – 2-е изд. , стер. – М.: Издательский центр «Академия», 2007.– 560 с.

18. Микропроцессорная техника: Учебник для сред. проф. образования/ А.В. Кузин, М.А. Жаворонков. – М.: Издательский центр «Академия», 2004. – 304 с.

19. *Микушин А.В.* Электронный конспект «Микропроцессоры в устройствах и системах», «Микропроцессоры и цифровая обработка сигналов», СибГУТИ. Режим доступа [http://www.sibsutis.ru/~mavr/index.html]

20. *Новиков Ю.В., Скоробогатов П.К.* Основы микропроцессорной техники. – Интернет-университет электронных технологий.– М.: 440с.

21. *Пахомов С.* Sonoma — новое пришествие. //Компьютер–Пресс.– 2005.–№2.– с. 148–156.

22. Программирование микропроцессорных систем: Учеб. Пособие для вузов по спец. «Автоматиз. сист. Обр.информ. и упр.»/В.Ф.Шаньгин, А.Е.Костин, В.М.Илюшечкин, П.А.Тимофеев; под ред. В.Ф. Шаньгина.– М.: Высшая школа, 1990.

23. *Саватеев И.* QNX– что нового?// PC Week/RE.– 2005 – №1.

24. *Солонина А.И., Улахович Д.А., Яковлев Л.А.* Алгоритмы и процессоры цифровой обработки сигналов./Уч. пособие для студ. По направлению 654400 «Телекоммуникации» – СПб.: БХВ-Петербург, 2002. – 464 с.

25. Управляющие системы электросвязи и их программное обеспечение: Учебник для вузов/ *Р.А. Аваков, В.О. Игнатьев и др.*– М.: Радио и связь, 1991. – 256 с.

26. *Ульянов М.В.* Архитектуры процессоров./Учебное пособие для студентов по спец. 2201.– М.: МГАПИ, 2002. – 68 с. Режим доступа [http://window.edu.ru/window_catalog/files/r24114/ulianov.pdf]

27. Цифровая коммутационная система EWSD./Учебное пособие. – Самара: Самарская междугородная телефонная станция, 1997.

28. *Черняк Л.* Ядра и потоки современных микропроцессоров. //Открытые системы. – 2005. – №12.

29. *Шпаковский Г.И.* Организация параллельных ЭВМ и суперскалярных процессоров: Учеб. пособие. — Минск: Белгосуниверситет, 1996. — 296 с.: ил. ISBN 985-6144-50-6.

30. *Шпаковский Г.И.* Параллельные микропроцессоры для цифровой обработки сигналов и медиа данных. – Минск: БГУ, 2000. – 196 с. ISBN 985-445-306-5.

31. EWSD версия 15. Документация по эксплуатации. – Siemens.– Январь, 2002.

32. EWSD version 10. Operations and maintenance documentation. – Siemens.– 1996.

33. Intel Itanium Processor. Hardware Developers Manual. – August, 2001. Document Number 248701– 002.

34. *Smith Steven W.* The Scientist and Engineer's Guide to Digital Signal Processing. Режим доступа [http://www.DSPguide.com].

35. *Боровский А.* От Niagara к Rock. //Открытые системы. – 2007. – №5.

36. *Костров Б.В., Ручкин В.Н.* Микропроцессорные системы и микроконтроллеры .- М.: «ТехБук», 2007. – 320 с.

37. *Никульский И.* Оптические интерфейсы цифровых коммутационных станций и сетей доступа.- М.: Техносфера, 2006. – 256 с.

38. *Гаспер Б.С., Липатов И.Н.* Решение задач по курсу «Прикладная теория надежности»: учебное пособие. – Пермь.: Пермский государственный технический университет.–1998. – 88с.

39. Надёжность и техническое обслуживание АМТС с программным управлением: справ. пособие/ Р.Р. Вегенер и др.; Под ред. В.Г. Дедоборща и Н.Б. Суторихина. – М.: Радио и связь, 1989. – 320 с.: ил.

40. *Ушаков И.А.* Курс теории надежности систем: учеб. пособие для вузов. – М.: Дрофа, 2008. – 239с.: ил.

41. Цифровые системы коммутации для ГТС/Под ред. В.Г Карташевского и А.В. Рослякова. – М.: Эко-Трендз, 2008. – 352с.: ил.

Список использованных сокращений

AAU – Application Accelerator Unit – Блок ускорителя приложений
ACT – Active – активное состояние устройства, полная работоспособность
ANSI – American National Standard Institute – Американский национальный институт стандартов
API – Application Programming Interface – Прикладной программный интерфейс
APS – Application Program System – система прикладных программ
ASIC – Application-Specific Integrated Circuit – Интегральная схема, ориентированная на приложение, заказная микросхема
ATM – Asynchronous Transfer Mode – Асинхронный режим переноса
ATU – Address Translation Unit – Блок трансляции адресов
AU –Administrative Unit – административный блок SDH
BAPM – Base Processor Master – Базовый процессор ведущий
BAPS – Base Processor Slave – Базовый процессор ведомый
B:CMY – Bus to Common Memory – Шина доступа к общей памяти
BER – Basic Encoding Rules – Базовые правила кодирования (данных)
BIOS – Base Input/Output System – Базовая система ввода/вывода
CA – Communication Area – Зона (область) связи в СР113с
CAP – Call Access Processor– Процессор обработки вызовов
CD – Compact Disk – Компакт–диск (оптический носитель информации)
CDMA – Code division multiplexing access –Технология мультидоступа с кодовым разделением каналов
CDR – Call Detail Record – Подробная запись о состоявшихся соединениях
CI – Common Interface – Общий интерфейс
CISC – Complex Instruction Set Computer – компьютер (вычислитель) со сложной системой команд
CCS №7 – Common Channel Signaling no 7 – Общеканальная система сигнализации №7
CCNC – Common Channel Network Control – Контроллер сети сигнализации по общему каналу
CCNP – Common Channel Network Processor – Процессор контроллера сети сигнализации по общему каналу
CMP – Chip Multiprocessors – Много процессоров на одном кристалле
CMY – Common Memory – Общая память
CMYC – Common Memory Control – Контроллер общей памяти
CMYDIO – Common Memory, Data net an Input/Output Stage – Схема ввода/вывода по сети данных при доступе к общей памяти
CMYMFC – Common Memory, Maintenance Facilities an Cycle Control – Контроллер циклов при обращении к общей памяти
CMYM – Common Memory Medium – Запоминающая среда общей памяти
CRC – Cyclical redundancy check – Контроль с помощью циклического избыточного кода
CP – Coordination Processor – Координационный процессор
CPU – Central Processing Unit – Центральное вычислительное (процессор-

ное) устройство

DACK – DMA Acknowledge – Подтверждение доступа DMA

DDR – Double Data Rate – Удвоенная скорость передачи данных

DLU – Digital Line Unit – Блок цифровых линий (абонентских)

DLUC – Digital Line Unit Controller – Контроллер цифрового блока

DMA – Direct Memory Access – Прямой доступ к памяти

DMU – Data Memory Unit – Устройство памяти данных

DRAM – Dynamic Access Memory – Динамическая память с произвольным доступом (к данным)

DRQ – DMA Request – Запрос доступа DMA

DSA – Digital Signature Algorithm – Алгоритм с использованием открытого ключа для создания электронной подписи.

DSP – Digital signal processor – Цифровой сигнальный процессор (процессор цифровой обработки сигналов)

DVD – Digital versatile disk – Универсальный цифровой диск

ECC – Error Correction Code – Код проверки/коррекции ошибки, проверочный код

EISA – Extended (enhanced) industry standard architecture – Расширенная стандартная архитектура для промышленного применения (стандартная общесистемная шина 32-х разрядных процессоров. Введено с 1988 г.)

EPIC – Explicitly Parallel Instruction Computing – Технология обработки команд с явным параллелизмом (явно–параллельные вычисления)

EPROM – Erasable Programmable Read-Only Memory – Стираемое программируемое постоянное запоминающее устройство с возможностью чтения

ETSI – European Telecommunication Standard Institute – Европейский институт стандартов в области связи (телекоммуникаций)

FAT – File Allocation Table – Таблица размещения файлов

FAT32 – File Allocation Table 32 – Таблица размещения файлов в 32-х разрядно операционной системе

FEPROM – Flash Erasable Programmable Read only Memory – Быстро стираемое электрически программируемое постоянное запоминающее устройство с возможностью чтения

FIFO – First Input, First Out – Первый пришёл, первый ушёл (дисциплина обслуживания заявок в очереди)

FPU – Floating Point Unit – устройства обработки данных с плавающей точкой

FTAM – File Transfer Access Methode – Управление доступом передачи файлов

FTP – File Transfer Path – Протокол передачи файлов

GNU – GNU is Not Unix – Неправительственный международный проект по свободному распространению программного обеспечения

GP – Group Processor – Групповой процессор

GPRS – General Packet Radio Service – Общая радиослужба передачи данных

GSM – Global System for Mobile Communications – Глобальная система подвижной радиосвязи (система сотовой связи)

HDB3 – High-Density Bipolal - биполярный код высокой плотности.

HDLC – High-level data link control protocol – Протокол высокого уровня для управления каналом передачи данных

HTTP – Hypertext Transfer Protocol – Протокол передачи гипертекстовой ин-

формации

HTML – HyperText Markup Language – Язык гипертекстовой разметки (в сети Интернет)

IEEE – Institute of Electrical and Electronics Engineers – Институт инженеров по электротехнике и электронике

IEC – International Electrotechnical Commission – Международная электротехническая комиссия

IETF – Internet Engineering Task Force – Рабочая группа по инженерным проблемам Интернета

IOC – Input/Output Control – Управление вводом/выводом (процессор)

IOP – Input/Output Processor – Процессор управления вводом/выводом

IOP:MB – Input/Output Processor for Message Buffer – Процессор управления вводом/выводом для буфера сообщений

IOP:UNI – Input/Output Processor Universal – Процессор управления вводом/выводом универсальный

iSCSI – internet Small Computer Systems Interface – Интерфейс малых компьютерных систем при подключении к сети Интернет (стандарт высокоскоростного параллельного интерфейса, разработанный ANSI, используется для подключения к компьютеру периферийных устройств, других компьютеров или ЛВС)

IP – Internet Protocol –Протокол межсетевого взаимодействия

IPC – Interprocessing Communication – Межпроцессоное взаимодействие

IRQ – Interrupt Request – Запрос на прерывание

ISDN – Integrated Service Digital Network– Цифровая сеть с интеграцией служб, ЦСИС.

ISO – International Standard Organization – Международная организация по стандартизации

I/O – Input/output – ввод/вывод

ISA – Industry-Standard Architecture – в настоящем пособии – название системной шины IBM PC/XT, в настоящее время не поддерживается.

ITU – International Telecommunication Unit – Международный союз электросвязи

ITU-T – International Telecommunication Unit – Standardization Sector – Международный союз электросвязи – сектор стандартизации

IU – Integer Unit – Целочисленное устройство, блок вычислений целых чисел

IXA – Internet Exchange Architecture – архитектура поддерживающая обмен информацией через сеть интернет

LMY – Local Memory – Локальная память (локальное оперативное запоминающее устройство)

LSU – Блок предсказания перехода

LTG – Line Trunk Group – Линейная группа

MAS – Master – Ведущий

MB – Message Buffer –буфер сообщений

MBL – Maintenance Blocked – заблокирован для технического обслуживания

MDD – Mediation Disk Device – Накопитель на жёстком магнитном диске

MIB – Management Information Base – База информации управления

MIMD – Multiple Instruction Multiple Data – Много потоков команд, много оттоков данных

MIS – Management Information Service – Услуга информации по управлению

MISD – Multiple Instruction Single Data – Много потоков команд, один поток данных

MML – Man-machine language – Язык общения «человек–машина»

MMU – Memory Management Unit – Устройство управления памятью

MP – Main Processor – Главный процессор

MPLS – Multi-Protocol Label Switch – Многопротокольная коммутация на основе меток

MU – Messaging Unit – Блок сообщений

MUXS – Multiplexer slave – Подчинённый мультиплексор

NIC – Network Interface Card – Сетевая интерфейсная карта

NRZ – Non-Return to Zero – «Без возврата к нулю» – метод бинарного кодирования информации, при котором единичные биты представляются положительным значением, а нулевые отрицательным.

NTFS – New Technology File System – файловая система, применяемая начиная с операционной системы Windows NT. Поддерживает объектно-ориентированные приложения и технологию самовосстановления.

OA&M – Operation, Administration and Maintenance – Техническое обслуживание, администрирование и техническая эксплуатация

OMG – Object Management Group – Группа по управлению объектами, неправительственная организация

OMT – Operations and Maintenance Terminal – Терминал (ПЭВМ) технического обслуживания и эксплуатации

OS – Operation System – Управляющая система (операционная система)

OSF – Operation System Function – Функция управляющей системы

OSI – Open System Interaction – Взаимосвязь открытых систем

PBI – Peripheral bus interface unit – Блок интерфейса периферийной шины

PCI – Peripheral Component Interconnect – Межкомпонентное соединения с периферийными элементами. Промышленный стандарт 32-х разрядной системной шины с возможностью расширения до 64 разрядов.

PMON – Performance Monitoring Unit – Блок мониторинга производительности

PU – Processing Unit – Блок (процессор) обработки данных

PEX – Program Executor – Подсистема выполнения программ, модуль

POSIX – Portable Operating System Interface – переносимый интерфейс операционной системы (под переносимостью понимается свойство открытых систем)

QoS – Quality of Service – Качество обслуживания

RAID – Redundant Array of Independent Disks – Избыточный (резервированный) дисковый массив, состоящий из независимых накопителей на жёстких магнитных дисках.

RAM – Random Access Memory – Устройство памяти с произвольной выборкой данных, ЗУПВ. Часто используется как ОЗУ.

RFC – Request For Comments – « необходим комментарий» (обозначение документа IETF

RISC – Redused Instruction Set Computer – компьютер (вычислитель) с сокращенной системой команд

RMON – Remote Monitoring – Дистанционный мониторинг

RPC – Remote Procedure Call – Удалённый (дистанционный) запрос процедуры.

RSA – Rivest, Shamir, and Adelman (algorithm) – Алгоритм шифрования

(криптографический алгоритм) по схеме открытого ключа

SDH – Synchronous Digital Hierarchy – Синхронная цифровая иерархия

SDL – Specification and Description Language – Язык спецификаций и описаний

SDRAM – Synchronous dynamic RAM – Синхронное динамическое запоминающее устройство с произвольной выборкой данных, ЗУПВ.

SILTC – Signaling Link Terminal Controller – Управляющее устройство оконечного оборудования тракта сигнализации

SILTD – Signaling Link Terminal Digital – Цифровой терминал тракта сигнализации ОКС№7

SIMD – Single Instruction Multiply Data – Один поток команд, много потоков данных.

SIPA – Signaling Peripherical Adapter – Сигнальный периферийный адаптер контроллера управления ОКС№7 в системе EWSD

SISD – Single Instruction Single Data – Одиночный поток команд–одиночный поток данных

SLA – Service Level Agreement – Соглашение об уровне обслуживания

SN – Switching Network – Цифровой коммутационное поле

SNMP – Simple Network Management Protocol – Простой протокол сетевого управления

SPARC – Scalalable Processor Architecture – Наращиваемая/масштабируемая архитектура процессора (архитектура процессоров с изменяемой производительностью)

SPR – Spare – Резервный

SPU – Service Provision Unit – Блок предоставления услуг

SRAM – Static Random Access Memory – Статическая память с произвольным доступом (к данным)

SSNC – Signal System Network Controller – Сетевой контроллер системы сигнализации

SSP – Synchronous Serial Port – Блок синхронного последовательного порта

STM –Synchronous Transport Module – Синхронный транспортный модуль.

SQL – Server-Client Language – Язык запросов Сервер-клиент (архитектура «клиент-сервер»)

TCP – Transmission Control Protocol – Протокол контроля передачи (входит в стек протоколов TCP/IP)

TMF – Telecommunication Management Forum, TeleManagement Forum – Форум по управлению телекоммуникациями, неправительственная организация

TMN – Telecommunications Management Network – Сеть управления электросвязью

TUG – Tributary Unit Group – Группа трибутарных (компонентных) блоков

UDP – User Datagram Protocol –Протокол передачи дейтаграмм пользователя

UMTS – Universal Mobile Telecommunications System – Универсальная система мобильной связи

UNA – Unavailable – Устройство не готово (отсутствует операционная готовность)

UNI – User-Network Interface – Интерфейс «пользователь – сеть».

USB – Universal Series Bus – Универсальная шина с последовательной передачей бит

VLIW – Very Long Instruction Word – Длинное командное слово

VoIP – Voice over IP – передача голоса по протоколу IP
VPN – Virtual Privet Network – виртуальная частная (выделенная) сеть
VC – Virtual Container – Виртуальный контейнер
VC-n – Virtual Container of level n – Виртуальный контейнер уровня n (n=1, 2,3,4,12)
X.25 – Сеть передачи данных по протоколу X.25
WAN – Wide Area Network – глобальная вычислительная сеть (или сеть связи)
А – Адресная часть (поле) формата команды
АА – Абсолютный адрес
АК – Абонентский комплект
АЛУ – Арифметико-логическое устройство
АМТСЭ – Автоматическая междугородная телефонная станция электронная
АСУ – Автоматизированная система управления (сетью связи)
АТС – Автоматическая телефонная станция
АТСЭ – Автоматическая телефонная станция электронная
АЦП – Аналогово-цифровое преобразование
БД – База данных
БИС – Большая интегральная схема
БРОН – Блок регистров общего назначения
ВВ – Ввод/вывод
ВУ – Внешние устройства
ЗИП – Запасные инструменты и приборы
ИСО – Международная организация по стандартизации
ГОСТ – Государственный стандарт
ГОСТ Р – Государственный стандарт России
ГУУ – Групповое управляющее устройство
Зп – Операция записи в физическую память
ЗУ – Запоминающее устройство
ЗУПВ – Запоминающее устройство с произвольной выборкой
ИУУ – Индивидуальное управляющее устройство
К-МОП – Кремний – металл – окисел – полупроводник (технология производства микросхемы)
КОП – Код операции
ЛВС – Локальная вычислительная сеть (LAN)
МСЭ – Международный союз электросвязи
МСЭ-Т – Сектор стандартизации электросвязи МСЭ
МПр – Микропроцессор (микропроцессоры)
МЭК – Международная электротехническая комиссия, IEC
НАМ – Начальный адрес массива
НСД – Несанкционированный доступ (к данным, к оборудованию)
НЖМД – Накопитель на жёстком магнитном диске
НМЛ – Накопитель на магнитной ленте
НОД – Накопитель на оптическом диске
ОЗУ – Оперативное запоминающее устройство
ОП – Операционная часть (поле) формата команды; также – операнд

ОКС№7 – Система сигнализации ОКС номер 7
ОС – Операционная система
ОС РВ – Операционная система реального времени
П – Признак модификации операции
ПЗУ – Постоянное запоминающее устройство
ППЗУ – Программируемое (перепрограммируемое) постоянное запоминающее устройство
ПО – Программное обеспечение
ПУУ – Периферийное управляющее устройство
ПЦОС – Процессор цифровой обработки сигналов
ПЭВМ – Персональная электронно-вычислительная машина
РД – Руководящий документ
РОН – Регистр общего назначения
САДМП – Система административных программ
САП – Система автоматизированного проектирования
СБИС – Сверхбольшая интегральная схема
СИАС – Сигнал индикации аварийного сигнала
СИНП – Система испытательно-наладочных программ
СКП – Система коммутационных команд
СОЗУ – Сверхоперативное запоминающее устройство
СОП – Система отладки программ
СППЗУ – Стираемое перепрограммируемое постоянное запоминающее устройство
СПТО – Система программ технического обслуживания
СУБД – Система управления базами данных
СУЭС – Сеть управления электросвязью
СчК – Счётчик команд
ТФОП –Телефонная сеть связи общего пользования (также ТфСОП).
ТЭЗ - Типовой элемент замены
УПАТС – Учрежденческо-производственная АТС
УУ – Устройство управления
ЦАП – Цифро-аналоговое преобразование
ЦСИС – Цифровая сеть с интеграцией служб
ЦКП – Цифровое коммутационное поле
ЦПУ – Центральное вычислительное (процессорное) устройство
ЦСП – Цифровой сигнальный процессор (процессор цифровой обработки сигналов)
ЦУУ – Центральное управляющее устройство
ЧНН – Час наибольшей нагрузки
Чт – Операции чтения из физической памяти
ЭВМ – Электронно-вычислительная машина
ЭСППЗУ – Электрически стираемое (пере)программируемое постоянное запоминающее устройство
ЯП – Ячейка памяти