**Содержание**

[Глава 1 История создания больших ЭВМ 5](#_Toc237598725)

[1. Россия и компания IBM 5](#_Toc237598726)

[2. История развития: от S/360 до zArchitecture 7](#_Toc237598727)

[2.1. Ключевые вехи эволюции мейнфреймов 10](#_Toc237598728)

[2.2. Современный сервер - это мейнфрейм 11](#_Toc237598729)

[3. Основные достоинства мейнфреймов 12](#_Toc237598730)

[4. Виды работ, выполняемые на мейнфреймах 14](#_Toc237598731)

[Глава 2. Конструкция серверов zSeries 15](#_Toc237598732)

[5. Модульный принцип конструктивного исполнения 15](#_Toc237598733)

[6. Двухядерный процессорный узел 17](#_Toc237598734)

[7. Плата центрального электронного узла CEC 18](#_Toc237598735)

[7.2. Шкафы для размещения большой ЭВМ и контроллеры каркасов 21](#_Toc237598736)

[7.3. Типы процессорных узлов 23](#_Toc237598737)

[8. Конфигурация и реконфигурация серверов 23](#_Toc237598738)

[8.1. Виды плановой реконфигурации 24](#_Toc237598739)

[8.2. Неплановая реконфигурация 24](#_Toc237598740)

[8.3. Режимы выполнения изменений 24](#_Toc237598741)

[9. Система управления сервером (мейнфреймом) 25](#_Toc237598742)

[9.1. Процессоры гибкой поддержки FSP 25](#_Toc237598743)

[9.2. Элемент поддержки SE 25](#_Toc237598744)

[9.3. Консоль управления НМС 26](#_Toc237598745)

[9.4. Основные функции консоли управления HMC 27](#_Toc237598746)

[6. Вопросы для контроля усвоения материала 28](#_Toc237598747)

[7. Краткие выводы по главе 2 29](#_Toc237598748)

[Глава 3. Архитектура центральных процессоров мейнфреймов zSeries 30](#_Toc237598749)

[3. Ключевые особенности архитектуры zArchitecture 30](#_Toc237598750)

[4. Базовая архитектура большой ЭВМ класса мейнфрейм 32](#_Toc237598751)

[2.1. Основная память 33](#_Toc237598752)

[2.2. Расширенная память 35](#_Toc237598753)

[2.3. Центральный процессорный узел 36](#_Toc237598754)

[5. Регистровая модель процессора 37](#_Toc237598755)

[3.1. Регистры общего назначения 38](#_Toc237598756)

[3.2. Регистры для выполнения операций с плавающей точкой 38](#_Toc237598757)

[3.3. Регистр управления операциями с плавающей точкой 39](#_Toc237598758)

[3.4. Регистры управления 39](#_Toc237598759)

[3.5. Регистры доступа 39](#_Toc237598760)

[3.6. Регистр префикса 40](#_Toc237598761)

[3.7. Слово состояния процессора PSW 42](#_Toc237598762)

[4. Два режима выполнения команд процессором 44](#_Toc237598763)

[4.1. Целочисленные границы команд 45](#_Toc237598764)

[4.2. Форматы адресов и команд 45](#_Toc237598765)

[8. Краткие выводы по главе «Архитектура процессора zSeries» 47](#_Toc237598766)

[9. Термины и определения 50](#_Toc237598767)

[10. Вопросы для контроля усвоения материала главы 3 50](#_Toc237598768)

[Глава 4. Микроархитектура процессоров zSeries 52](#_Toc237598769)

[6. Типовая архитектура мультипроцессорной системы с общей памятью 52](#_Toc237598770)

[1.1. Способы размещения строк в кэш-памяти 54](#_Toc237598771)

[1.2 Два условия обеспечения когерентности 55](#_Toc237598772)

[1.3. Протоколы когерентности кэш-памяти и способы размещения строк 57](#_Toc237598773)

[1.3.1. Протоколы записи с аннулированием 57](#_Toc237598774)

[1.3.2. Протокол записи с обновлением 57](#_Toc237598775)

[1.3.3. Сквозная запись 58](#_Toc237598776)

[7. Алгоритм MESI 58](#_Toc237598777)

[8. Структура дублированного процессора zSeries 61](#_Toc237598778)

[9. Функциональная схема процессора 66](#_Toc237598779)

[9.1. Блок команд 66](#_Toc237598780)

[2.2. Блок операций 66](#_Toc237598781)

[3.3. Блок управления BCE 67](#_Toc237598782)

[3.4. Сопроцессор 68](#_Toc237598783)

[3.4.1. Блок сжатия 68](#_Toc237598784)

[3.4.1. Блок транслятора 69](#_Toc237598785)

[3.4. Блок восстановления 69](#_Toc237598786)

[11. Краткие выводы по главе «Микроархитектура процессоров zSeries» 71](#_Toc237598787)

[12. Вопросы для контроля усвоения материала 72](#_Toc237598788)

[Глава 5. Логическое разделение ресурсов 73](#_Toc237598789)

[10. Уровни конфигурирования системы 73](#_Toc237598790)

[11. Логический раздел LPAR 76](#_Toc237598791)

[12. Параметры логического раздела LPAR 77](#_Toc237598792)

[13. Процессоры логического раздела LPAR 77](#_Toc237598793)

[14. Распределение памяти в логических разделах LPAR 77](#_Toc237598794)

[15. Взаимодействие логического раздела с подсистемой ввода-вывода 79](#_Toc237598795)

[6.1. Принципы адресации периферийных устройств 79](#_Toc237598796)

[6.1. Способы выделения каналов ввода-вывода для LPAR 80](#_Toc237598797)

[16. Кластеры логических разделов (LPAR-claster) 81](#_Toc237598798)

[16.1. Кластер совмещенных дисков прямого доступа DASD 81](#_Toc237598799)

[16.2. Кольцо канал-канал (CTC/GRS rings) 82](#_Toc237598800)

[7.3. Параллельный системный комплекс Parallel SysPlex 83](#_Toc237598801)

[17. Использование логических разделов в мейнфрейме в университете 85](#_Toc237598802)

[18. Система распределения ресурсов Intelligent Resource Director (IRD) 87](#_Toc237598803)

[13. Вопросы для контроля усвоения материала главы 5 87](#_Toc237598804)

[14. Краткие выводы по главе «LPAR» 88](#_Toc237598805)

[Глава 6. Основы мультипроцессирования 89](#_Toc237598806)

[19. Управление работой процессора 89](#_Toc237598807)

[1.1. Состояния процессора 89](#_Toc237598808)

[1.2. Средства управления работой процессора 91](#_Toc237598809)

[1.3. Пять вариантов сброса процессора 92](#_Toc237598810)

[1.3.1. Сброс процессора 92](#_Toc237598811)

[1.3.2. Начальный сброс процессора 93](#_Toc237598812)

[1.3.3. Сброс подсистемы ввода-вывода 93](#_Toc237598813)

[1.3.4. Сброс с очисткой 93](#_Toc237598814)

[1.3.5. Сброс по питанию 94](#_Toc237598815)

[2. Основные принципы мультипроцессирования 94](#_Toc237598816)

[2.1. Взаимодействие процессоров через общую память 95](#_Toc237598817)

[2.2. Средства межпроцессорных обменов 96](#_Toc237598818)

[2.2.1. Команда SIGNAL PROCESSOR (SIGP) 96](#_Toc237598819)

[2.2.2. Механизм обслуживания прерываний 96](#_Toc237598820)

[2.2.2.1. Код прерываний 99](#_Toc237598821)

[2.2.2.2. Процесс прерывания 99](#_Toc237598822)

[3.2.2.3. Маскирование 100](#_Toc237598823)

[2.2.2.4. Плавающие прерывания 100](#_Toc237598824)

[2.2.2.5. Приоритеты прерываний 101](#_Toc237598825)

[2.3. Средства временной синхронизации 102](#_Toc237598826)

[2.3.1. Часы реального времени TOD 102](#_Toc237598827)

[3.3.1. Компаратор времени 104](#_Toc237598828)

[2.3.2. Процессорный таймер 105](#_Toc237598829)

[2.3.2. Расхождения в отсчетах времени (Timer Stepping) 105](#_Toc237598830)

[3. Запуск мультипроцессорной операционной системы 106](#_Toc237598831)

[4. Краткие выводы по главе 6 «Основы мультипроцессирования»: 107](#_Toc237598832)

[5. Вопросы для контроля усвоения материала главы 6 107](#_Toc237598833)

[Глава 7. Работа с основной памятью. Адресация памяти 110](#_Toc237598834)

[20. Три типа физической памяти 110](#_Toc237598835)

[1.1. Информационные форматы 112](#_Toc237598836)

[1.2. Целочисленные границы в памяти 113](#_Toc237598837)

[21. Типы адресов и адресных пространств основной памяти 114](#_Toc237598838)

[2.1. Абсолютный адрес и абсолютное адресное пространство 115](#_Toc237598839)

[2.2. Реальный адрес и реальные адресные пространства 117](#_Toc237598840)

[2.3. Виртуальный адрес и виртуальные адресные пространства 118](#_Toc237598841)

[2.4. Режимы виртуальной адресации и типы виртуальных адресов 120](#_Toc237598842)

[2.3.1. Главный виртуальный адрес (Primary Virtual Address) 121](#_Toc237598843)

[2.3.2. Вторичный виртуальный адрес (Secondary Virtual Address) 121](#_Toc237598844)

[2.3.3. Виртуальный адрес, определяемый регистром доступа (AR-Specified Virtual Address) 122](#_Toc237598845)

[2.3.4. Базовый виртуальный адрес (Home Virtual Address) 122](#_Toc237598846)

[2.5. Эффективный адрес 122](#_Toc237598847)

[2.5.1. Логический адрес 123](#_Toc237598848)

[2.5.2. Адрес команды 123](#_Toc237598849)

[2.6. Размеры адресов 124](#_Toc237598850)

[3. Механизм префиксации 125](#_Toc237598851)

[3.1. Формат регистра префикса 126](#_Toc237598852)

[3.2. Правила префиксации 130](#_Toc237598853)

[15. Вопросы для контроля усвоения материала 130](#_Toc237598854)

[16. Краткие выводы по главе 7 131](#_Toc237598855)

[Глава 8. Работа с основной памятью. Динамическое преобразование адресов. 132](#_Toc237598856)

[1.1. Этапы динамического преобразования адресов 132](#_Toc237598857)

[2.2. Два варианта задания адресного пространства в процессоре 133](#_Toc237598858)

[3. Номер адресного пространства ASN 136](#_Toc237598859)

[4. Номера таблицы ASTE 137](#_Toc237598860)

[4.1. Текущий номер в таблице ASTE (ASTESN) 137](#_Toc237598861)

[4.2. Начальный номер в таблице ASTE (ASTEIN) и повторное использование номера ASN 138](#_Toc237598862)

[5. Трансляция ASN 139](#_Toc237598863)

[5.1. Управление ASN-трансляцией 140](#_Toc237598864)

[5.2. Таблицы ASN-трансляции 141](#_Toc237598865)

[5.2.1.1. Формат записи в таблице AFT 141](#_Toc237598866)

[5.2.1.2. Формат записи в таблице AST 142](#_Toc237598867)

[5.3. Трансляция кодов ALET 144](#_Toc237598868)

[5.4. Формат ASCE 146](#_Toc237598869)

[5.5. Буфер быстрой адресации ALB (Access Lookaside Buffer) 147](#_Toc237598870)

[5.6. Авторизация адресных пространств 147](#_Toc237598871)

[9.1. Авторизация с помощью кода ASN 148](#_Toc237598872)

[9.2. Расширенная ASN - авторизация 148](#_Toc237598873)

[10.1. Форматы виртуальных адресов 149](#_Toc237598874)

[10.1.1. Формат 31-разрядного виртуального адреса 149](#_Toc237598875)

[10.1.2. Формат 64-разрядного виртуального адреса 150](#_Toc237598876)

[10.2. Динамическое преобразование адресов 152](#_Toc237598877)

[10.3. Управление числом уровней преобразования 155](#_Toc237598878)

[10.4. Буфер быстрой переадресации TLB 156](#_Toc237598879)

[4.2. Краткие выводы по главе 157](#_Toc237598880)

[4.3. Вопросы для контроля усвоения материала 158](#_Toc237598881)

[Глава 9 Подсистема ввода-вывода мейнфреймов zSeries 159](#_Toc237598882)

[22. Основные принципы работы подсистемы ввода-вывода в большой ЭВМ 159](#_Toc237598883)

[2. Компоненты подсистемы ввода-вывода в большой ЭВМ 160](#_Toc237598884)

[2. Механизм создания множественных образов канальной подсистемы 162](#_Toc237598885)

[4. Адресация в подсистеме ввода-вывода 165](#_Toc237598886)

[4.1. Идентификация объектов канальной подсистемы 169](#_Toc237598887)

[4.1.1. Идентификация канальных путей 169](#_Toc237598888)

[4.1.2. Нумерация подканалов 170](#_Toc237598889)

[4.1.3. Нумерация периферийных устройств 170](#_Toc237598890)

[4.1.4. Идентификация периферийных устройств 172](#_Toc237598891)

[4.1.5. Идентификация подсистемы ввода-вывода 173](#_Toc237598892)

[4.2. Управление назначением канальных путей 173](#_Toc237598893)

[4.2.1. Слово состояния канала CCW 173](#_Toc237598894)

[4.2.2. Формат команды ввода-вывода 175](#_Toc237598895)

[4.2.3. Особенности выполнения операций ввода-вывода 175](#_Toc237598896)

[4.2.4. Назначение канальных путей 175](#_Toc237598897)

[5. Три режима обмена информацией: frame-multiplex mode, burst mode, or byte-multiplex 176](#_Toc237598898)

[17. Вопросы для контроля усвоения материала 178](#_Toc237598899)

[18. Краткие выводы по главе 9 «Подсистема ввода-вывода мейнфрейма zSeries» 179](#_Toc237598900)

[19. Литература 180](#_Toc237598901)

## Глава 1 История создания больших ЭВМ

## Россия и компания IBM

Что нас ждет? BD04972_

После прочтения этой главы Вы сможете:

1. сказать, как давно компанию IBM и Россию связывают деловые отношения;
2. назвать ключевые вехи развития архитектуры большой ЭВМ класса мейнфрейм;
3. обосновать четыре принципа построения современной ЭВМ;
4. перечислить особенности архитектуры мейнфреймов;
5. назвать два вида работ, выполняемых на мейнфреймах.

История отношений между Россией и компанией IBM (International Business Machines) начинается c того, что один из основателей компании IBM – признанного лидера – производителя вычислительных машин, Герман Холлерит в 1890-х годах изобрел статистический табулятор, который использовался при всемирной переписи населения, в том числе при переписи населения России. Холлерит использовал перфокарты как эффективное средство для хранения информации, в перфокартах дырки означали определенные параметры: пол, возраст, профессию человека. Надо сказать, что форма перфокарты, предложенной Холлеритом, почти без изменений сохранилась до наших дней (см. рисунки 1). Внизу для сравнения размеров приведена 80-колонная карта IBM (рисунок 2).

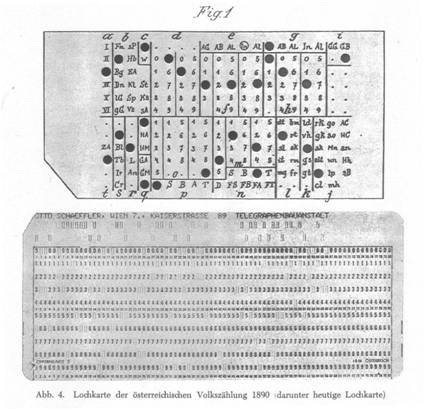


Рисунок 1 - Пробитая карта австрийской переписи 1890 г.

Рисунок 2. Внизу для сравнения размеров приведена 80-колонная карта IBM [Zemanek 1973,c.548]

В России Герман Холлерит побывал четыре раза, в одной из таких поездок его и застиг фотограф (см. рисунок 3).



Рисунок 3 – Герман Холлерит в России, 1897 год.

В 1896 году Г. Холлерит, изобретатель бумажных перфокарт, зарегистрировал фирму «Tabulating Machine», которая в 1911 году, объединившись еще с двумя американскими компаниями, составила основу появившейся в 1924 году компании IBM, ныне ведущей компании мира.

Биография Г. Холлерита приведена в разделе Персоналии.

## История развития: от S/360 до zArchitecture

История развития больших электронных вычислительных машин (далее – ЭВМ) берет свое начало в 1960-х годах. До этого времени каждая вычислительная машина разрабатывалась под заказ для конкретных бизнес-целей, поэтому была уникальна. К 1964 году ситуация изменилась, производители вычислительной техники начали создавать стандарты аппаратного и программного обеспечения.

Президент компании IBM Дж. Уотсон-младший объявил 7 апреля 1964 г. о выпуске семейства больших ЭВМ с названием System/360. В этом названии по-русски «Система S/360» цифра 360 обозначает полную окружность в градусах и символизирует удовлетворение всех возможных потребностей пользователей ЭВМ. То есть, целью проектаS/360было разработать семейство различных по производительности и стоимости компьютеров, позволяющих удовлетворить любые запросы пользователей, и это был один из самых дорогостоящих проектов в истории вычислительной техники.

Компьютеры этого семейства получили название *"мейнфреймы" (mainframe),* по названию типовых стоек IBM, в которых размещалось оборудование центрального процессора.

Каждый из компьютеров семейства создавался из набора унифицированных аппаратных и программных модулей, совместимых на информационном (представление и кодирование информации), программном (единая система команд и программистская модель) и аппаратном (схемотехническая база, электрические сигналы, кабели, разъемы и другие конструктивные решения) уровнях.

Первые машины были громоздки, занимали значительную площадь, на рисунке 4 показана одна вычислительная машина 1960х годов, она занимала большую комнату.



Рисунок 4 – Мейнфрейм 1960-х годов

Для демонстрации компьютера того времени Вы можете использовать видео-ролик на 3 минуты (файл IBM360.mp4), прилагаемом к этой книге.

Концепция семейства программно-совместимых компьютеров стала стандартом для всей компьютерной промышленности, недаром на слуху у всех популярное высказывание «IBM-совместимые компьютеры».

Компьютеры семейства System/360 породили новое явление в компьютерной индустрии, создав стандарт на аппаратно-программное обеспечение, так называемые "платформу" и «архитектуру». Надо отметить, что все последующие поколения этого семейства поддерживают программную совместимость "снизу-вверх", что означает, что программы, написанные в 1960-х годах для первых компьютеров этой серии, работают и на самых последних современных моделях этой серии.

Такой подход обеспечил эволюционное развитие программных продуктов, существенно снижая затраты пользователей при переходе на новые аппаратные средства. Сейчас пользователи могут запускать свои программы на стандартном мейнфрейме, не задумываясь о том, какое аппаратно-программное обеспечение при этом требуется. Более того, пользователям нет необходимости обновлять машину, заботиться о совместимости старого и нового программного обеспечения.



Рисунок 5 – Современная большая ЭВМ ( мейнфрейм)

Первые бизнес-приложения, написанные на языках COBOL, FORTRAN, или PL/1, выполнялись на мейнфреймах, и многие из этих старых программ до сих пор работают.

За прошедшие десятилетия мейнфреймы достигли выдающихся процессорных возможностей (до 54-х процессоров в одной машине), сегодня они обслуживают сотни тысяч пользователей, управляются с экзабайтами (264) данных, реконфигурируются «на ходу» без остановки рабочего режима, имеют наработку на отказ более 14-ти лет, имеют более 20-ти уровней защиты информации. И все это при том, что размеры современной большой ЭВМ стали соизмеримы с размером бытового холодильника. На момент создания этого учебника последние модели этого семейства называются по-русски «зед-серией» (zSeries), а архитектура, на базе которой они построены, «зед-архитектурой» (z/Architecture) (рисунок 5).

#### 2.1. Ключевые вехи эволюции мейнфреймов



Рисунок 6 – Эволюция мейнфреймов

На рисунке 6 приведена схема, которая иллюстрирует эволюцию мейнфреймов с момента появления и до наших дней. На схеме отмечены шесть вех. Первой вехой отмечен 1964 год – год появления мейнфреймов. В 1970-1971 годах была разработана усовершенствованная система System/370 (здесь число 70 связывают с 70-ми годами), в ней впервые реализованы принципы виртуализации памяти. В 1983 году появилась расширенная архитектура 370/XA, (eXtended Architecture), в которой была введена 31-разрядная адресация и разработана новая канальная подсистема. В 1988 году была внедрена технология логического разделения ресурсов LPAR (Logical Partition Access Resources) и появилась архитектура ESA/370, в которой появился новый тип регистров – регистры доступа (access registers). В 1990 году появилась архитектура ESA/390, в которой появились новые каналы ESCON, появился дополнительный процессор, выполняющий криптографические функции, предложена новая технология объединения многих компьютеров в единый комплекс Sysplex Timer (ETR).

На 1990х годах стоит задержаться и сказать еще несколько слов. В 90-х годах прошлого века большие ЭВМ пережили сложные времена. Среди большинства программистов сложилось мнение, что все задачи можно решить при помощи персональных компьютеров, в крайнем случае, персональные компьютеры можно объединить в сеть. Известный журналист (ИМЯРЕК) опубликовал в 1993 году в журнале Computer World статью под названием «Вымерли как динозавры», в которой предрекал быстрое завершение жизни мейнфреймов. И действительно, персональные компьютеры расширили область применения вычислительной техники в бизнесе и в быту, компания IBM в этом году не продала ни одного мейнфрейма. Казалось, журналист прав. Однако, целый ряд задач по-прежнему невозможно было решить без мейнфреймов. Не нашлось замены мейнфреймам в банковском бизнесе, от мейнфреймов не отказались международные страховые компании, компании авиаперевозок, железнодорожные компании. Список можно продолжать долго, но мы остановимся, чтобы дальше в учебнике разобраться, почему мейнфреймы выжили и «не вымерли, как динозавры». Подробныйрассказ о журналисте, написавшем статью «Вымерли как динозавры», см. в файле в электронном виде.

В 2000 была объявлена новая серия мейнфреймов под названием z/Series, в этой архитектуре введена 64-х разрядная архитектура и сделано много прогрессивных новшеств. Поэтому в 2003 году этот журналист вынужден был прилюдно «съесть свои слова» - правда, отлитые из шоколада. А в 2009 году компания IBM отпраздновала 45-летний юбилей бесперебойной работы мейнфреймов.

#### Современный сервер - это мейнфрейм

Большие ЭВМ современной серии zSeries предназначены для развития электронного бизнеса. Они используются для выполнения Интернет-услуг. В рамках этой серии заявлено новое направление в развитии вычислительной техники в компании IBM – построение саморегулируемых (самонастраивающихся и самоорганизующихся) систем (Autonomic Computing). В основу этих систем положен принцип саморегуляции, заимствованный из живой природы. Вычислительная система такого типа должна иметь интеллектуальные механизмы внутренней самоорганизации и динамической адаптации к условиям деловой активности предприятия.

Мэйнфреймы используются в критически важных областях деятельности компаний, там, где требуется высокая производительность и надежность информации. Они нашли свое место в эпоху Internet: На момент создания этого учебника 85% информации, находящейся в Интернете, хранится в базах данных на мейнфреймах. Более 500 крупнейших мировых корпораций имеют вычислительные комплексы, оснащенные мэйнфреймами. Везде, где высоки требования к быстродействию, надежности, безопасности, используются именно мейнфреймы. Банки, страховые компании, транспортные корпорации активно используют сейчас технологии электронной коммерции (электронный обмен информацией, электронные транзакции и др.), обслуживание пользователей с использованием современных информационных технологий.

Можно сказать, что мейнфрейм - это большая вычислительная машина, доступная пока (из-за ее высокой стоимости) только крупной компании, предназначенная для хранения в ней коммерческих баз данных, серверов транзакций, приложений, требующих более высоких уровней безопасности и надежности, а также постоянства доступа; это вычислительная система, которая предоставляет большие возможности с точки зрения скорости вычислений и их объемов; это вычислительная система, которая используется для решения большого количества разнообразных задач, выполняет огромное количество различных типов транзакций и нагрузок.

## Основные достоинства мейнфреймов

Какой ценой получены такие достижения? Перечислим ключевые особенности архитектуры zArchitecture, благодаря которым мейнфреймы «выжили», а не исчезли «как диназавры».

Остановимся на перечислении достоинств этой машины. Основные характеристики современной большой вычислительной машины, это надежность, доступность и обслуживаемость (RAS – reliability, availability, serviceability). Например, в появившейся летом 2005 года новой архитектуре Z9 отсутствует единая точка отказов. Это значит, что каждый компонент машины с этой архитектурой может быть заменен новым без остановки рабочих нагрузок.

Централизованное управление – еще одна важная характеристика мэйнфрейм, это много дешевле, чем децентрализованное управление. Уникальная возможность мэйнфрейм - управление нагрузками.

Несомненным достоинством этих машин является и непрекращающаяся совместимость, что означает, что до сих пор в мэйнфреймах используются приложения, написанные в 70-е годы.

Признаки самонастраивающихся систем:

* многопроцессорная, многоузловая реализация с включением процессоров разных типов;
* практически неограниченные возможности масштабирования серверов и их консолидация;
* наращивание ресурсов по требованию;
* обеспечение логического разделения - partition, при котором один сервер представляется в виде нескольких отдельных виртуальных компьютеров с разными операционными средами и единым центром управления и средствами взаимодействия;
* широкий спектр средств контроля и восстановления работоспособности;
* гарантии высочайшего уровня готовности;
* минимизация эксплуатационных затрат.

Мэйнфреймы zSeries образуют одну из линеек серверов eServer. Буква Z в названии zSeries, архитектура z/Architecture означает сокращение от "zero down time": нулевое время простоя - высочайшая надежность, позволяющая непрерывно поддерживать работу сервера на заданном уровне производительности по схеме 7 (семь дней в неделю) × 24 ( 24 часа в сутки) × 365 (365 дней в году).

Эти преимущества достигаются централизацией вычислительной мощности в рамках одного сервера с развитой системой контроля и возможностью "горячей" замены отказавших элементов.

Масштабируемость архитектуры сервера: по количеству процессоров, по объему памяти, по средствам ввода-вывода и другим параметрам.

Высокая концентрация вычислительной мощности достигается за счет использования КМОП - интегральной технологии и многочиповых модулей MultiСhip Module (MCM). Каждый такой модуль может содержать несколько десятков интегральных чипов микропроцессоров, КЭШ-памяти, управления и обмена, а также относительно небольшой корпус размером примерно 12×12 см.

Отличительной особенностью мейнфрейма является наличие у него подсистемы ввода-вывода, которая, как правило, занимает отдельный шкаф (фрейм) и имеет свои специализированные процессоры.

Высокая пропускная способность подсистемы ввода-вывода обеспечивается большим числом разнообразных высокоскоростных каналов, управляемых этими процессорами. Такие процессоры выполняют канальные программы, находящиеся в основной памяти и реализующие операции ввода-вывода. Это, с одной стороны, освобождает центральные процессоры от большинства рутинных и медленных операций, связанных с вводом-выводом, а с другой стороны, распараллеливает выполнение таких операций.

Кластеризация серверов zSeries позволяет создавать системы IBM Parallel Sysplex, в которых могут быть консолидированы серверы разных поколений.

Архитектура серверов серии zSeries – это архитектура, имеющая большое количество функций, недоступных на других платформах:

* подсистема ввода-вывода с контрольными блоками,
* виртуализация и распределение функций,
* возможности параллельных вычислений,
* ориентированное на цели управления рабочими нагрузками, и другие.

## Виды работ, выполняемые на мейнфреймах

Мэйнфреймы могут выполнять два вида работ, которые соответственно представляют собой обслуживание двух абсолютно различных типов рабочих нагрузок (см. рисунок 6):

* *Пакетная обработка заданий* (Batch job), когда компьютер выполняет работу без участия человека. Используется в случае значительных объемов данных на входе.
* *Обработка заданий в реальном времени* (On-line), например, транзакционные системы, такие как система приобретения железнодорожных билетов, система оплаты по кредитной карте и т.п.

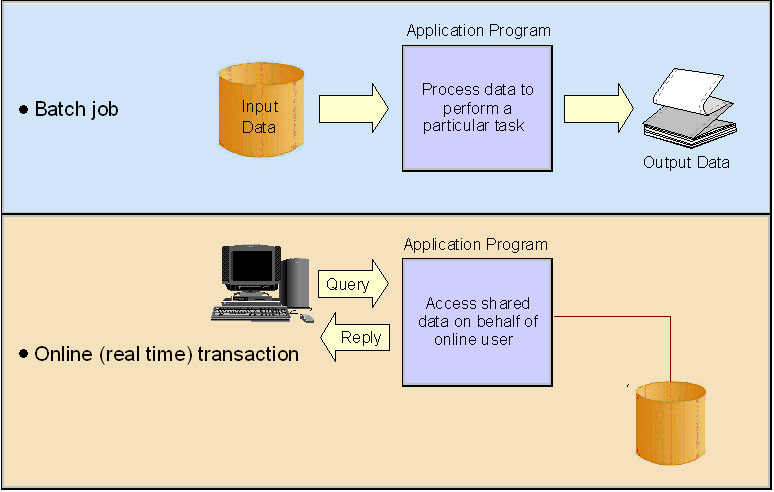


Рисунок 6 – Виды работ, выполняемых на мейнфреймах

Главное отличие этих видов работ - в объеме вводимых и выводимых данных. Если требуется соблюсти заведомо оговоренное самое высокое быстродействие, при этом обслужить значительное количество пользователей, стоит использовать мэйнфрейм, операционная система которой сможет координировать всю эту работу.

## Вопросы для контроля усвоения материала

1. Что такое мейнфрейм?
2. Что такое перфокарта?
3. Объясните название S/360.
4. Перечислите основные вехи развития мейнфреймов.
5. Какие виды работ выполняются на мейнфрейме?
6. Перечислите первые языки программирования на большой ЭВМ.

## Глава 2. Конструкция серверов zSeries

Что нас ждет? BD04972_

После прочтения этой главы Вы сможете:

1. перечислить основные конструктивные узлы многочипового процессорного модуля;
2. рассказать о двух технологиях конструирования компании IBM;
3. объяснить, почему двухядерный процессорный узел повышает надежность работы машины;
4. назвать два варианта исполнения центрального электронного комплекса;
5. объяснить как организована работа кольцевой структуры в современной ЭВМ;
6. объяснить названия шкафов в мейнфрейме;
7. перечислить основные типы процессорных узлов в мейнфрейме;
8. назвать основные элементы системы управления сервером;
9. перечислить функции управления сервером.

## Модульный принцип конструктивного исполнения

Общий принцип конструктивного исполнения серверов zSeries пояснен на рисунке 7. Схемотехническую базу серверов образует набор интегральных схем - чипов, выполненных на одном кристалле и реализующих различные узлы и устройства вычислительной машины. Основными чипами являются процессорный чип (PU – processor unit), чипы кэш-памяти второго уровня (L2 SD - second level storage device), системного контроллера (SC – system controller), адаптера памяти (MBA – memory bus adapter), контроллера памяти (MSC –memory storage control) и другие. Самый малый узел - процессорный чип, например, z990 имеет размер 14,1×18,9 мм и содержит 122 млн. транзисторов, у чипа системного контроллера (SD) эти параметры равны соответственно 17,5×17,5 мм и 521 млн. транзисторов.

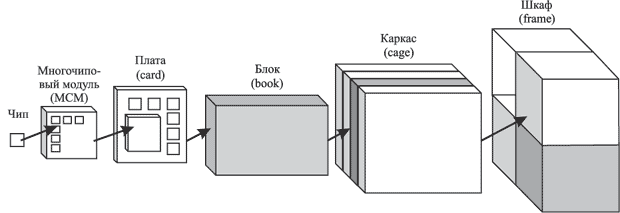


Рисунок 7 – Общий принцип конструктивного исполнения мейнфрейм

Чипы объединяются в мульти-чиповый модуль MCM (Multi-Chip-Module), который является основой процессорного ядра сервера. Многочиповый процессорный модуль MCM представляет собой многослойную подложку, на верхнем слое которой размещается необходимый набор чипов, а межсоединения выполняются в нескольких нижележащих слоях. (см. рисунок 8).

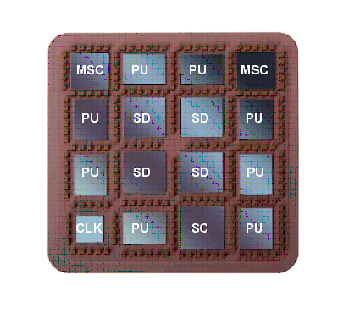


Рисунок 8 – Многочиповый процессорный модуль MCM.

Слои межсоединений реализуются с использованием стеклокерамических подложек (например, на показанном на рисунке 2.1. кристалле z9 таких слоев – 102) , обеспечивающих время распространения сигналов 7,8 ps/mm. При этом в каждом слое межсоединений используется трехуровневая технология Triplate, состоящая из двух слоев ортогональных проводников (X, Y слои) и слоя питания или земли, обеспечивающая снижение электромагнитных помех между слоями. Общая длина внутренних соединений для модели z9 – 0,476 км. В нижней части модуля MCM устанавливается разъем PGA (Pin Grid Array) с необходимым количеством контактов (например, для MCM z990 количество контактов - 5184). Разъем выполнен по технологии IBM Harcon (high density aria connection). Нагревание MCM контролируется несколькими термисторами с использованием охлаждающей системы.

Примеры конструктивного выполнения процессорных узлов для нескольких современных больших ЭВМ серии mainframe IBM приведены на рисунке 9.

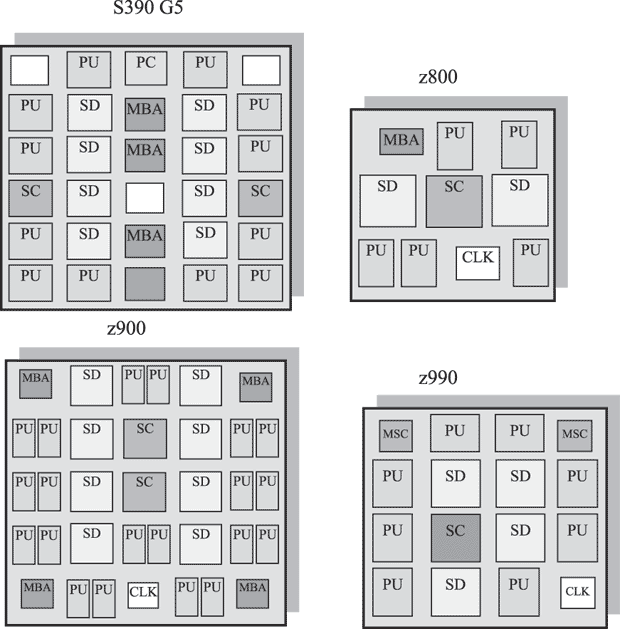


Рисунок 9 – Примеры выполнения МСМ для серверов IBM серий S390G5, z800, z900 и z990

## Двухядерный процессорный узел

Современная технология позволяет на сегодня создавать сразу два процессорных узла в одном чипе, так называемые «двухядерная технология» (dual core). В серверах z990 четыре из восьми процессорных чипов содержат по два процессора PU, имеющих общий интерфейс с системным контроллером SC и КЭШ второго уровня. Остальные четыре процессорных чипа содержат по одному процессору, что обеспечивает до 12 процессоров в одном MCM. Во время выполнения операций обеспечивается их дублирование, что повышает надежность их выполнения. Следует отметить, однако, что в случае отказа одного из сдвоенных процессоров второй процессор также останавливается, и оба процессора запускают диагностические процедуры. Схема размещения вычислительных узлов двух процессоров в одном чипе показана на рисунке 10.

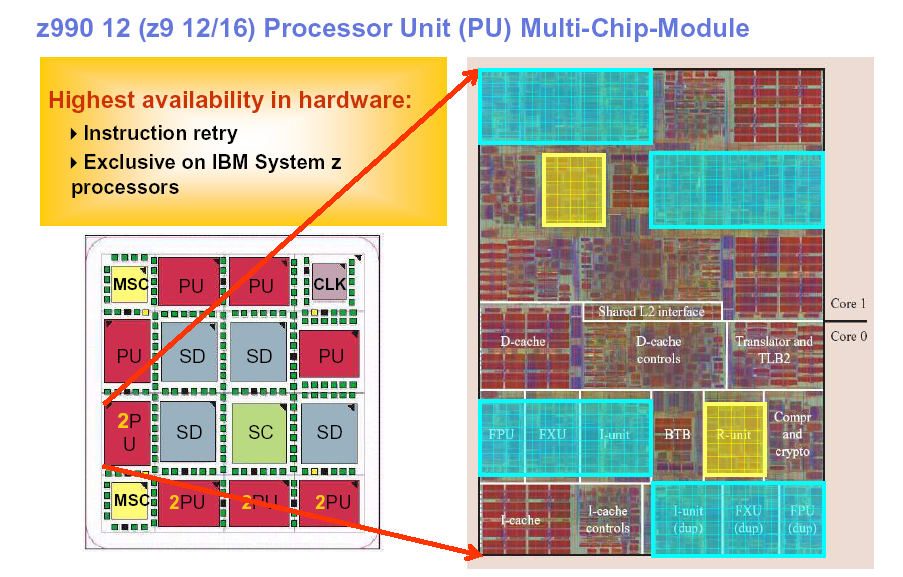


Рисунок 10 – Двухядерная технология процессорного узла IBM

## Плата центрального электронного узла CEC

Следующим конструктивным уровнем серверов серии zSeries является плата центрального электронного комплекса CEC (Cenral Electronic Complex), которая объединяет многочиповые модули MCM, модули памяти и модули интерфейса с подсистемой ввода-вывода. Плата CEC для серверов серии z900 - это многослойная печатная плата размером 553×447 мм с 10-ю сигнальными слоями и 24-мя слоями питания и земли. Модули размещаются на плате CEC с двух сторон, что обеспечивает более плотную упаковку. Как и в MCM, в комплексе CEC используется технология “Triplate” для уменьшения взаимных электромагнитных помех между слоями

Установка платы CEC выполняется двумя способами. В серверах z900 и более ранних моделях один комплекс CEC размещается в процессорном каркасе, содержащем также системы питания, вентиляции и управления системой (см. рисунок 11).

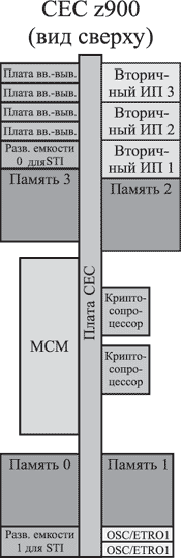


Рисунок 11 - Плата центрального электронного комплекса CEC для сервера z900

Второй вариант, используемый в серверах z990, z890, допускает установку нескольких CEC, каждый из которых выполняется в виде отдельного конструктивного блока (book), содержащего собственно CEC и вспомогательные системы питания, вентиляции и управления (см. рисунок 12). Для защиты от электромагнитных помех блок полностью закрыт металлическим кожухом и имеет габаритные размеры 56×14 см и вес 32 кг. Для объединения блоков используется центральная плата, имеющая с одной стороны четыре слота для подключения до четырех блоков CEC, а с другой - восемь слотов для установки вторичных источников питания.

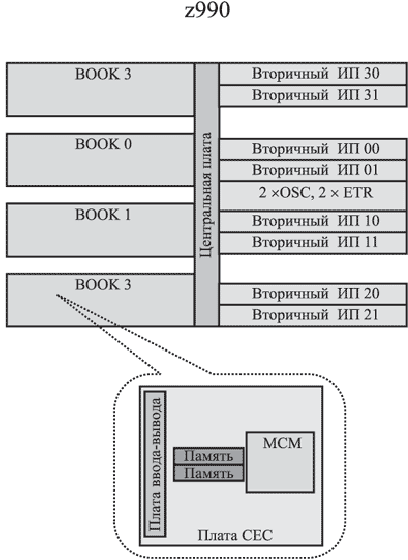


Рисунок 12 - Плата центрального электронного комплекса CEC для серверов z990 и z890

На рисунке 13 показана одна из четырех «книжек» сервера z9. Вид сбоку позволяет увидеть размещение модуля МСМ. Внутри книжки могут быть установлены дополнительно до 8-ми модулей памяти общим объемом до 128 GB.

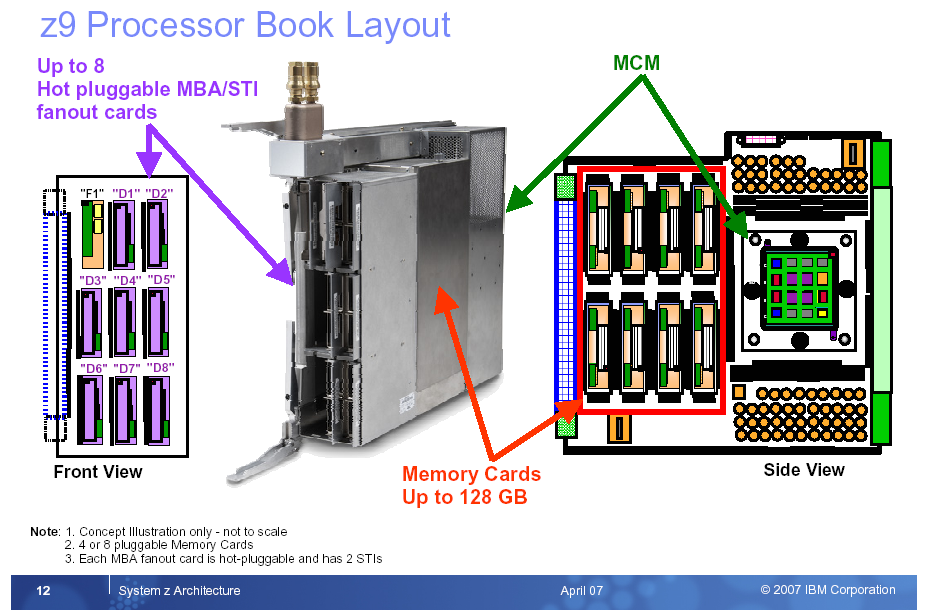


Рисунок 13 - Плата центрального электронного комплекса CEC для сервера z9

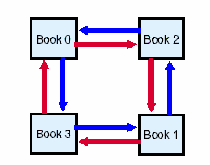
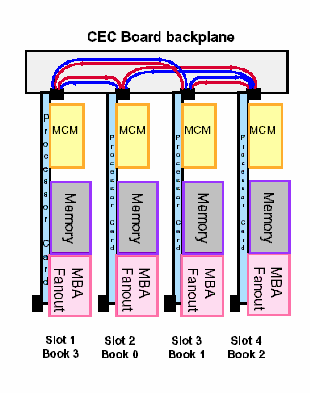


Рисунок 14 – Кольцевая структура подключения комплекса CEC

Для обеспечения согласованной безостановочной работы всех включенных процессоров, находящихся в отдельных книжках, для объединения памяти в единую замкнутую систему соединения между книжками реализованы в виде двух кольцевых структур (рисунок 14). Одно кольцо соединяет книжки по часовой стрелке, другое – против часовой стрелки. Если книжек две или три, кольцо замыкается с помощью переключателей, установленных на плате CEC. Если сервер имеет в своей конфигурации только одну книжку, тогда эти переключатели не работают. Книжки могут быть извлечены или добавлены без нарушения работоспособности машины беспрепятственно. Более того, не останавливая работу машины, можно модернизировать машину, добавляя книжки, либо добавлять те или иные блоки внутри книжки, например, заменяя испорченные части.

Важно отметить, что смена или добавление модулей памяти осуществляется в «горячем» режиме, без останова машины. Видео-ролик, иллюстрирующий операцию замены блока памяти, имеется на CD-ROMе, который прилагается к этому учебнику (файл MemInstall.wmv).

#### Шкафы для размещения большой ЭВМ и контроллеры каркасов

Для размещения процессорного каркаса и каркасов ввода-вывода используются специальные А и Z шкафы (A-frame, Z-frame). На рисунке 15 приведено условное изображение шкафов и их наполнение, используемое в серверах z990 и z890. Шкаф A-frame предназначен для размещения процессорных каркасов и одного каркаса ввода-вывода, а шкаф Z-frame – источников питания и для установки двух дополнительных (по выбору) каркасов ввода-вывода. Габаритные размеры шкафов равны 154×158×194 см, а вес - 790 кг (A) и 767 кг (Z). Шкаф Z входит в конфигурацию всегда, даже при отсутствии второго и третьего каркасов ввода-вывода, так как в этом шкафу располагаются первичные источники питания, формирующие постоянное напряжение 350 В. Для повышения надежности питания предусмотрено подключение сервера к двум внешним фидерам трехфазного напряжения. Постоянное напряжение 350В разводится по шкафам для подачи на вторичные источники питания (на рисунке не показаны), находящиеся в каждом из каркасов. В верхней части шкафов размещены батареи автономного питания IBF (Integrated Battery Function), обеспечивающие питание сервера в случае отключения внешнего питания.

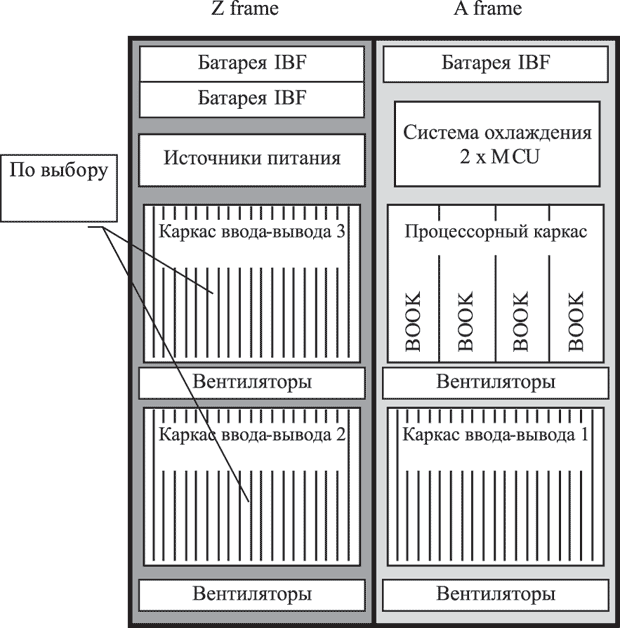


Рисунок 15 – Схема состава мейнфрейма

Для охлаждения блоков сервера используются две системы охлаждения MCU (Modular Cooling Unit). Каркасы ввода-вывода подключены к процессорному каркасу посредством самосинхронизирующих интерфейсов (Self-Timing Interface – STI). Адаптеры ввода-вывода, устанавливаемые в слотах, обеспечивают различные типы каналов для подключения периферийных устройств и для межсистемного обмена.

#### Типы процессорных узлов

Особенностью мейнфреймов является то, что в этих машинах одновременно сосуществуют процессоры разных конфигураций: кроме центральных процессоров, реализующих основные вычислительные функции машины, существуют также специализированные процессоры, выполняющие тот или иной набор задач.

Существует целая классификация процессорных узлов:

* Центральный процессор СР (central processor) реализует систему команд z/Architecture и ESA/390, этот процессор может работать с операционными системами z/VM, z/OS, Linux, TPF и другими;
* Процессор межсистемного взаимодействия ICF (Internal Coupling Facility) предназначен для реализации системного программного обеспечения Coupling Facility Control Code (CFCC), используемого при организации межсистемного обмена, может быть включен только в LPAR, выделенный для реализации таких функций;
* Процессор Java-приложений для серверов zSeries zAAP (zSeries Applications Assist Processor) ориентирован на эффективное исполнение Java-приложений под управлением виртуальной машины IBM Java Virtual Machine (JVM);
* Сервисный процессор SAP (System Assist Processor) используется для управления операциями ввода-вывода путем исполнения милликодов канальной подсистемы, причем, один из процессоров SAP выделен в качестве Master SAP для реализации обменов между CEC, размещенными в модулях – «книжках», и элементом поддержки SE;
* Процессор поддержки операционной системы Linux - IFL (Integrated Facility for Linux) оптимизирован для реализации операционной среды Linux и ее приложений;
* Процессор гибкой поддержки FSP (Flexible Support Processors) - это специальный элемент управления локальных сетей внутри мейнфрейма.

## Конфигурация и реконфигурация серверов

Серверы семейства zSeries допускают реконфигурацию двух типов: добавление аппаратных средств (процессоров, памяти, каналов ввода-вывода) и реконфигурацию (в том числе фоновую) за счет имеющихся в сервере резервных средств.

При этом реконфигурация может быть плановой и неплановой.

#### Виды плановой реконфигурации

**Реконфигурация по требованию CUoD** (Capacity Upgrade on Demand) выполняется путем добавления процессоров (загрузкой милликодов), памяти и каналов ввода-вывода. Такой тип реконфигурации не ограничен по времени действия и выполняется сервисной службой IBM.

**Реконфигурация по запросу пользователя CIU** (Customer Initiated Upgrade) позволяет увеличить количество процессоров и объем памяти по инициативе пользователя. Выполняется путем Web-запроса через веб-страницу компании IBM «Resource Link» в соответствии с предварительно оформленным контрактом и с использованием CUoD процедур.

**Временная реконфигурация On/Off CoD** (On/Off Capacity on Demand) позволяет подключить дополнительные процессоры на любое заданное время и выполняется по CIU запросу при наличии соответствующего контракта. Такой вид реконфигурации может использоваться для преодоления пиковых нагрузок сервера.

#### Неплановая реконфигурация

Неплановая реконфигурация CBU (Capacity BackUp) предназначена для временного подключения центральных процессоров CP при потере производительности вследствие аварийных ситуаций, она не может быть использована для преодоления пиковых нагрузок

сервера. Автоматическая реконфигурация CBU возможна при наличии резервных PU и разрешается после заключения контракта и загрузки в сервер специального кода.

#### Режимы выполнения изменений

Изменения, связанные с восстановлением и конфигурированием, могут выполняться различными способами. Наиболее часто осуществляется фоновое восстановление или конфигурирование системы в рабочем состоянии без начальной загрузки IPL (Initial Program Loading) или сброса по питанию POR (Power-on Reset). Некоторые изменения в конфигурации машины могут быть проведены без начальной загрузки IPL или сброса по питанию POR (Power-on Reset), но с отключением каналов и изменением идентификатора физического канала CHPID, поскольку потребует последующего подключения периферийных устройств. Более сложные изменения могут потребовать начальной загрузки IPL без сброса по питанию POR для отдельных LPAR и запущенных в них операционных систем. В редких случаях может потребоваться выполнение сброса по питанию (POR) для всей системы на относительно короткое время с последующим выполнением процедур начальной загрузки (IPL) для всех LPAR. Ремонт или изменения, требующие отключения питания всей системы, случаются крайне редко.

## Система управления сервером (мейнфреймом)

#### Процессоры гибкой поддержки FSP

Система управления (администрирования) сервером основана на использовании внутрисерверных локальных сетей, узлами которых являются специальные элементы - процессоры поддержки FSP (Flexible Support Processors). В некоторых моделях они называются *контроллерами каркасов* CC (Cage Control). Процессор поддержки FSP построен на основе микропроцессора PowerPC и подключен, с одной стороны - к внутренней сети сервера, а с другой - обеспечивает интерфейс SSI (SubSystem Interface)c управляемым модулем сервера. Интерфейс SSI определяется типом модуля иможет состоять из нескольких последовательных интерфейсов и отдельных управляющих линий. Различные компоненты сервера содержат свои FSP, обеспечивая требуемую полноту контроля и управления сервером.

#### Элемент поддержки SE

Структура системы управления сервером показана на рисунке 16. В системе используются две внутренние локальные сети Ethernet, подключенные к основному и альтернативному элементам поддержки SE (Support Element). Оба элемента поддержки SE устанавливаются в шкафу Z, и активен всегда только один из них. Каждый элемент поддержки SE реализован на ноутбуке ThinkPad и имеет два сетевых адаптера Ethernet или Token Ring для подключения к внутренним сетям. Возможно переключение элемента поддержки SE с одной внутренней сети на другую. Процедуры управления сервером выполняются путем передачи из элемента поддержки SE в процессор FSP сообщений с командами для различных модулей сервера. Процессор FSP шаг за шагом выполняет требуемые операции с подключенным к нему модулем и формирует сообщение в элемент поддержки SE с результатами выполнения команды.

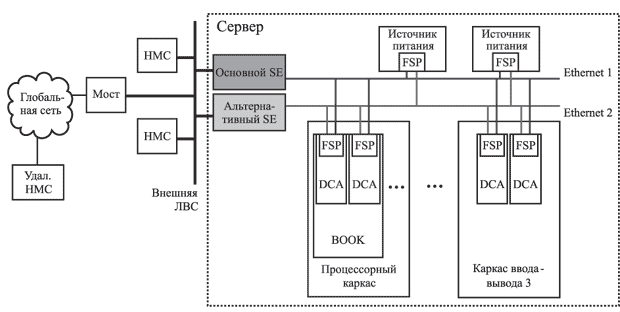


Рисунок 16 – Система управления сервером

#### Консоль управления НМС

Как уже было сказано выше, элементы поддержки SE соединены с одной или несколькими внешними сетями по протоколу Ethernet или Token Ring, по этим сетям они подключены к консолям управления HMC (Hardware Management Consoles). Таким образом, возможно удаленное управление сервером через удаленное подключение консоли управления HMC. Консоль управления HMC представляет собой рабочую станцию на основе персонального компьютера (ноутбука) с операционной средой OS/2 и коммуникационным сервером, в которой исполняются приложения HMCA (HMC Аpplications) для удаленного управления системами и выполнения других работ (экранная форма интерфейса такого приложения показана на рисунке 2.11). Управление сервером обычно осуществляется через HMC путем передачи команд в элементы поддержки SE, в то же время, при необходимости управление сервером может выполняться непосредственно рядом с машиной из любого элемента поддержки SE. Отметим, что с одной консоли HMC можно управлять до 100 элементами поддержки SE, и каждый элемент SE может получать команды от 32-х консолей управления HMC.

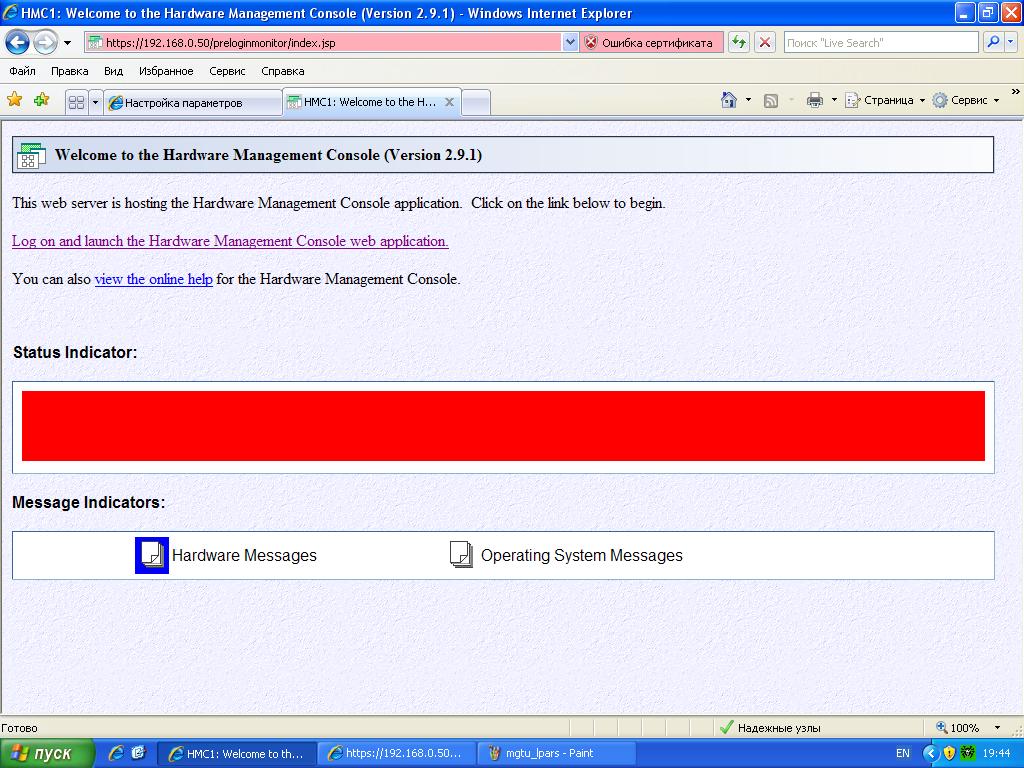


Рисунок 17 – Интерфейс консоли управления НМС

#### Основные функции консоли управления HMC

Основными функциями консоли управления HMC и элемента поддержки SE являются контроль состояния модулей сервера и выполнение управляющих процедур для модернизации, восстановления и других операций с модулями. Контроль состояния модулей осуществляется непрерывно для выявления ситуаций, требующих выполнения процедур HMC. Обычно такие ситуации являются следствием отказов в дублирующих каналах модулей и требуют реакции от HMC до наступления следующего отказа (отказ каналов источников питания, отсутствие резервных процессорных узлов (PU) и тому подобное). Функции консоли управления и элемента поддержки следующие:

* интерфейс с оператором;
* информирование о состоянии системы;
* определение проблемных состояний и их автоматический анализ;
* конфигурирование системы;
* загрузка милликодов;
* выполнение плановых операций;
* удаленное управление и другие.

Периодичность контроля может быть, например, два раза в день при параллельном отображении состояния системы из основного в альтернативный элемент поддержки SE, или может быть организовано событийное управление, причем события могут определяться как результат выполнения специальных процедур, таких как первоначальная загрузка милликода, завершение восстановления и другие.

Стоит сказать о таком понятии, как милликод, отдельно. Этот термин принят в компании IBM, означает дополнительный уровень управления, реализующий подмножество команд, в том числе, наиболее сложных команд. Именно милликоды обеспечивают возможность прямого управления аппаратными средствами процессора. Для этого существует особый режим выполнения – с помощью набора теневых регистров MGR (Millicode General Purpose Registers).

## Вопросы для контроля усвоения материала

1. Какие чипы содержит многочиповый модуль МСМ?
2. Какие преимущества дает использование двухядерного процессора?
3. Назовите два вариант установки СЕС.
4. Какая топология соединения центральных электронных комплексов в мейнфрейме?
5. Какие шкафы имеются у мейнфрейма? Назовите основные узлы, входящие в каждый шкаф.
6. Перечислите основные типы процессорных узлов.
7. Опишите состав базовой конфигурации сервера.
8. Из каких элементов состоит система управления сервером?
9. Перечислите функции консоли управления сервером.

## Краткие выводы по главе 2

Оглянемся назад? 

1. Многочиповый процессорный модуль МСМ содержит:

* процессорные чипы (PU – processor unit),
* чипы кэш-памяти второго уровня (L2 SD - second level storage device),
* чипы системного контроллера (SC – system controller),
* чипы адаптера памяти (MBA – memory bus adapter),
* чипы контроллера памяти (MSC –memory storage control).

1. Технологии конструирования MCM:
   1. Triplate - два слоя отрогональных проводников и слой земли и питания;
   2. Harcon - разъемы с тысячами количеством контактов.
2. Двухядерный процессорный узел повышает надежность выполнения операций.
3. Два варианта установки центрального электронного комплекса:
   * + процессорный каркас (cage);
     + отдельный конструктивный блок «книжка» (book).
4. Кольцевая топология подключения центральных электронных комплексов (книжек): два соединения – кольца, соединенных по часовой и против часовой стрелки.
5. Состав шкафов мейнфрейма:
   * + 1. **А- шкаф:** процессорный каркас и каркас ввода-вывода;
       2. **Z-шкаф**: два дополнительных каркаса ввода-вывода, источники питания, система охлаждения**.**
6. Типы процессорных узлов:

* Центральный процессор СР;
* Процессор межсистемного взаимодействия ICF;
* Процессор для JAVA-приложений ZAAP;
* Сервисный процессор SAP;
* Процессор для LINUX;
* Процессор гибкой поддержки FSP.

1. Состав системы управления сервером:

* Процессор поддержки FSP
* Элемент поддержки SE;
* Консоль управления HMC.

1. Функции консоли управления и элемента поддержки:

* интерфейс с оператором;
* представление информации о состоянии системы;
* определение проблемных состояний и их анализ;
* конфигурирование системы;
* загрузка милликодов;
* выполнение плановых операций;
* удаленное управление и др

## Глава 3. Архитектура центральных процессоров мейнфреймов zSeries

Что нас ждет? BD04972_

После прочтения этой главы Вы сможете:

1. назвать ключевые особенности архитектуры zArchitecture;
2. перечислить базовый состав архитектуры zArchitecture;
3. перечислить семь классов команд процессора;
4. описать регистровую модель процессора;
5. дать табличное представление регистровой модели процессора;
6. назвать целочисленные границы команд;
7. описать слово состояния процессора.

## Ключевые особенности архитектуры zArchitecture

Современная большая машина класса IBM мейнфрейм разработана таким образом, что сохраняет преемственность более ранних машин: система может работать в режиме архитектуры ESA/390, выполняя программы с 24-разрядной и 31-разрядной адресацией, и в современном режиме архитектуры zArchitecture с 64-разрядной адресацией. Изначально машина загружается в режиме ESA/390, для перехода на современную архитектуру используется специальная команда сигнала процессору SIGP (SIGNAL PROCESSOR), все процессоры при этом должны быть остановлены, то есть, должны быть переведены в состояние СТОП (stopped state). Архитектура с 64-х разрядной адресацией и 64-х разрядными целочисленными арифметическими командами включает в себя также набор команд прежней архитектуры с 31-разрядной адресацией и 32-х разрядными целочисленными арифметическими командами. И это - первая особенность архитектуры zArchitecture. Именно эта особенность обеспечивает совместимость снизу-вверх: бизнес-приложения, разработанные в 60-70х годах, работают на самых современных мейнфреймах. Можно сказать, что 24-х разрядная и 31-разрядная архитектуры встроены в 64-х разрядную архитектуру современного мейнфрейма. Возможная несовместимость между архитектурами обычно касается лишь работы операционных систем, но не работы приложений.

Второй важнейшей особенностью новой архитектуры является постоянная готовность системы к работе (availability), что достигается многократным дублированием вычислительных операций. Дублирование осуществляется на уровне работы процессора, когда работают зеркальные каналы в блоках выполнения команд. Дублирование осуществляется на уровне вычислительного узла - в одной машине могут работать до четырех центральных электронных комплексов. Дублирование осуществляется и на уровне предприятия, когда вычислительные центры обработки информации обязательно имеют резервную систему, параллельно выполняющую все те же действия рабочей системы и находящуюся на удаленном от вычислительного центра расстоянии.

Отличительные черты новой архитектуры касаются также новых форматов 128-ми разрядного слова состояния программы PSW (Program State Word), в частности, его размер расширен до 16-ти байтов, чтобы слово могло хранить команды с большими адресами. Слово PSW также содержит вновь заданный разряд, который определяет 64-х разрядную адресацию.

В динамическом преобразовании адресов в архитектуре zArchitecture добавлены три дополнительных уровня преобразования, реализованные с помощью таблиц перекодировки, так называемых «таблиц регионов», при этом осталось использование уже существующих таблиц сегментов и страниц.

Дополнительно к существующим в предыдущей архитектуре способам задания виртуальных адресных пространств разработан управляющий элемент (иногда его называют код) адресного пространства ASCE (Address Space Control Element), с помощью которого можно создавать новые адресные пространства. Элемент ASCE может служить указателем реального адресного пространства, и это еще одна важная особенность архитектуры zArchitecture: возможность быстрого перехода от виртуального к реальному адресному пространству. Элемент ASCE дает возможность создавать такие виртуальные адреса, которые могут рассматриваться как реальные адреса без использования таблиц преобразований.

Префиксная область увеличена до 8 Кбайт, что позволяет кроме сохранения старых и новых слов состояния программы PSW, сохранять еще и значения регистров при сохранении состояния процесса.

Еще одной ключевой особенностью архитектуры zArchitecture является особым образом организованная работа подсистемы ввода-вывода. Чтобы сохранить преемственность в работе 31-разрядной адресации периферийных устройств, введен специальный разряд в блоке запроса на обслуживание ORB (operation-request-block), который определяет один из двух форматов работы подсистемы ввода-вывода: обмен блоками по 2 кбайта или по 4 кбайта.

Подробному рассмотрению того, как были достигнуты перечисленные достоинства архитектуры zArchitecture, посвящена эта и последующие главы учебника.

## Базовая архитектура большой ЭВМ класса мейнфрейм

Архитектура большой ЭВМ класса IBM мейнфрейм может быть представлена следующим образом (см. рисунок 3.1).

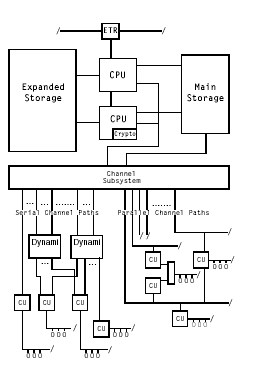


Рисунок 3.1 – Базовая архитектура zSeries.

Базовая архитектура zArchitecture большой ЭВМ класса мейнфрейм состоит из набора центральных процессоров СРU (Сentral Processor Units), согласованную работу которых синхронизирует внешний таймер ETR (External Timing Removed), основной памяти Main Storage, расширенной или дополнительной памяти Extended Storage, подсистемы ввода-вывода (далее по тексту мы будем называть ее канальной подсистемой) Channel SubSystem (CSS), в составе которой есть последовательные и параллельные каналы. Последовательные каналы управляются с помощью устройств селекции Dynami и контрольных блоков CU (Control Unit), через которые машина общается с периферийными устройствами D (Device). Контрольные блоки CU, их называют также контроллерами ввода-вывода или управляющими блоками, – очень важная часть канальной подсистемы.

Соединение между канальной подсистемой и контрольным блоком называется канальным путем (Channel Path). Канальный путь использует протокол параллельной либо последовательной передачи данных и соответственно называется либо параллельным, либо последовательным канальным путем. Последовательный канальный путь может подсоединяться к контрольному блоку через динамический переключатель Dynami, который предоставляет возможность организации различных внутренних соединений между портами этого переключателя.

Криптографический узел может быть выполнен встроенным в процессор, а внешний таймер (ETR) может быть подключен к системе.

Физическая реализация описанного набора основных компонентов архитектуры может варьироваться в зависимости от модели системы.

Специальные процессоры могут различаться по своим внутренним характеристикам, функциям, числу подканалов; канальные пути и контрольные блоки могут различаться по способу подключения периферийных устройств к канальной подсистеме, основная (и расширенная) память может различаться размерами, а система в целом будет отличаться характеристиками производительности.

Система, рассматриваемая без периферийных устройств ввода-вывода, называется конфигурацией. Все физическое оборудование, независимо от того, входит оно в состав конфигурации или нет, называется инсталляцией.

Процессорная память – это память внутри самого процессора, она недоступна для программ.

Внутренняя или процессорная память имеет многоуровневую структуру, включающую до трех уровней буферной памяти, называемой КЭШ-памятью. Уровни КЭШ-памяти предназначены для увеличения быстродействия основной памяти и "прозрачны" для процессора, то есть программно недоступны.

### 2.1. Основная память

Непосредственно связанное с центральными процессорами запоминающее устройство, в котором хранятся программные команды и данные, в которое записываются результаты вычислений перед пересылкой их во вспомогательное запоминающее устройство, имеющее большую емкость, или на устройство вывода данных. Это – оперативная память, особенно та ее часть, которая используется в качестве среды для хранения команд и данных, необходимых для текущего процесса в процессоре [1].

Основная память MS является многоабонентной и допускает одновременные обращения, как всех центральных процессоров, так и процессоров канальной подсистемы.

Основная память является напрямую адресуемой, поэтому она предоставляет возможность высокоскоростной обработки данных процессором и канальной подсистемой. И данные и программы должны быть загружены в основную память с входных устройств, прежде чем будут обработаны. Размер основной памяти, доступной в системе, зависит от конкретной модели. Память измеряется блоками по 4 КВ. Вначале канальная подсистема и все центральные процессоры в конфигурации имеют доступ к тем же самым блокам памяти и обращаются к конкретным блокам основной памяти, используя одни и те же абсолютные адреса.

Основная память может включать в себя буферную память, имеющую меньшее время доступа, называемую кэш-памятью. Кэш-память представляет собой буфер между регистрами процессора как самыми быстродействующими запоминающими устройствами и ячейками основной памяти. Кэш предназначен для выравнивания степени доступности этих двух типов за счет временного хранения содержимого ячеек оперативной памяти. Каждый центральный процессорный узел может иметь кэш ассоциативного типа.

Ассоциативная кэш – это память, которая «способна определять», содержится ли требуемый элемент данных – *слово-признак* – по одному из ее адресов или в одной из ее ячеек. Это может быть достигнуто различными способами [1].

В некоторых случаях - с помощью параллельной комбинационной логики производится анализ каждого слова в памяти, причем, одновременно выполняется проверка на совпадение со словом-признаком. В других случаях осуществляется последовательный сдвиг слова-признака синхронно со всеми словами в памяти; каждый бит слова-признака затем сравнивается с соответствующим битом каждого слова памяти, причем, используется столько однобитовых схем совпадения, сколько имеется слов в памяти. По мере совершенствования техники ассоциативных запоминающих устройств появились методы маскирования слов-признаков, а также методы поиска, основанные на «близком» совпадении (в отличие от точного совпадения).

Поскольку реализация параллельных операций над большим количеством слов требует больших затрат (в смысле аппаратных ресурсов), используются разнообразные приемы «аппроксимации» работы ассоциативной памяти без выполнения полного перебора, который описывался выше.

Один из них заключается в использовании процедуры хеширования с целью генерации «наилучшей оценки» истинного адреса, за которой следует проверка содержимого ячейки с вычисленным адресом. Некоторые ассоциативные кэш строятся по обычному принципу записи-считывания (параллельно по словам), в других же реализуется метод последовательного сравнения; устройства второго типа называются ортогональными кэш.

### 2.2. Расширенная память

Память, используемая для хранения справочной информации, данных, непосредственно не участвующих в работе программ, называется расширенной, дополнительной или внешней памятью [1].

Расширенная память (ES) может быть встроенной в центральный электронный комплекс в некоторых моделях. Расширенная память, когда она существует, может быть программно доступна для всех процессорных узлов в конфигурации с помощью специальных процессорных команд PAGE IN и PAGE OUT, которые пересылают блоки данных по 4 кбайта (страницами) из расширенной памяти в основную память и наоборот. Поэтому расширенную память называют еще страничной памятью.

Дополнительная или расширенная память может представлять собой физическую память, внешнюю по отношению к мейнфрейму. Это могут быть устройства с прямым доступом DASD, таких, как дисковые приводы и приводы для магнитных лент (см. рисунок 2).

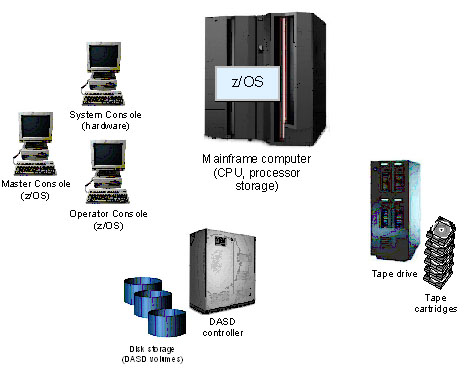


Рисунок 2 – Аппаратные ресурсы мейнфрейм

### 2.3. Центральный процессорный узел

Центральный процессорный узел (CPU) - это управляющий центр системы. Он содержит упорядочивающие и вычислительные возможности для выполнения команд, обработки прерываний, временных функций, начальной загрузки и других машинных функций.

Физическое исполнение CPU может изменяться от модели к модели, но логические функции остаются теми же. Результат выполнения команды должен быть одинаков для любой модели, именно это и обеспечивает совместимость.

Центральный процессорный узел (CPU) во время выполнения команд использует двоичную целочисленную арифметику, числа с плавающей запятой (двоичные и шестнадцатеричные) фиксированной длины, десятичные целые числа переменной длины, а также логические величины фиксированной и переменной длины. Обработка данных может быть параллельная или последовательная, размер элементов обработки, способы умножения, степень параллельности выполнения различных типов арифметики различается от одной модели процессора к другой, но результат вычислений остается тем же.

Команды, которые узел CPU выполняет, могут быть разделены на семь классов: общие, десятичные, с поддержкой плавающей точки (FPS), двоичные с плавающей точкой (BFP), шестнадцатеричные с плавающей точкой (HFP), управления и ввода-вывода.

Общие команды используются для выполнения двоичных целочисленных арифметических и логических операций, перехода и других неарифметических операций.

Десятичные команды оперируют с данными в десятичном формате.

Команды BFP и HFP оперируют с числами соответствующих форматов, тогда как команды FPS оперируют с данными с плавающей запятой независимо от формата или преобразуют их из одного формата в другой.

Привилегированные команды управления и команды ввода-вывода могут выполняться только тогда, когда процессор работает в режиме супервизора, псевдо-привилегированные команды управления могут выполняться в пользовательском режиме, после прохождения механизма авторизации.

Процессорный узел CPU предоставляет регистры, которые доступны программам, но не имеют адресуемого представительства в основной памяти. К таким регистрам относятся текущее слово состояния программы (PSW), регистры общего назначения, регистры управления операциями с плавающей точкой, регистры доступа, регистр префикса, регистры управления и регистры для компаратора времени и процессорного таймера. Каждый CPU в инсталляции имеет доступ к часам реального времени, которые используют все CPU в инсталляции. Код операции в команде определяет тип регистра, который должен быть использован в операции.

## Регистровая модель процессора

Регистровая модель процессора zArchitecture включает группы регистров, перечисленные в таблице 3.1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Обозначение** | **Наименование регистра** | **Разрядность** | **Кол-во** |
| **GR** | Общего назначения | **64** | **16** |
| **FPR** | С плавающей точкой | **64** | **16** |
| **FPC** | Управления  операциями  с плавающей точкой | **32** | **1** |
| **AR** | Доступа | **32** | **16** |
| **CR** | Управления | **64** | **16** |
| **PSW** | Слова состояния программы | **128** | **1** |
| **PR** | Префикса | **64** | **1** |
| **TOD** | Часов | **32** | **1** |
| **TR** | Таймера | **64** | **1** |
| **CCR** | Компаратора времени | **64** | **1** |

Итак, как показано в таблице, процессор содержит 10 типов регистров, из них 6 типов регистров имеют размер в 64 разряда, 3 типа регистров – 32-х разрядные и один регистр – слова состояния программы имеет размер в 128 разрядов. По количеству есть однотипные регистры количеством в 16 штук, это регистры общего назначения, регистры управления операциями с плавающей точкой, регистры доступа и регистры управления.

#### 3.1. Регистры общего назначения

Шестнадцать регистров общего назначения GR (General Register) используются для вычисления адресов, а также для выполнения целочисленных вычислений со знаком и без знака. Каждый регистр содержит 64 разряда, которые нумеруются от 0 до 63. Для архитектуры ESA/390 используются только младшие 32 разряда (разряды 32-63). Команды могут задавать информацию в одном или более регистрах общего назначения.

Каждый регистр содержит 64 разряда. Регистры общего назначения идентифицируются числами от 0 до 15, и эти числа указываются в 4-х разрядном поле R команды. Некоторые команды используют для адресации несколько регистров общего назначения, тогда они имеют несколько полей R. Для некоторых команд использование специального регистра общего назначения не требует его назначения в поле R.

Для некоторых команд разряды 32-63 или разряды 0-63 двух смежных регистров общего назначения спариваются, обеспечивая 64-х или 128-ми разрядный формат, соответственно. В таких операциях программа должна указывать номер того регистра, который содержит самые левые (высокого уровня) 32 или 64 разряда. Следующий регистр содержит (низкий уровень) самые правые 32 или 64 разряда.

Дополнительно к их использованию в качестве сумматоров в общей арифметике и логических операциях, регистры GR15 и GR16 также используются как регистры базового адреса и индекса при генерации адреса. В этих случаях регистры определяются с помощью четырехразрядного поля B или Х в команде. Число 0 в поле B или Х означает, что база или индекс не используются и таким образом, регистр общего назначения GR0 не может содержать базовый адрес или индекс.

#### 3.2. Регистры для выполнения операций с плавающей точкой

Все команды с плавающей точкой (FPS, BFP и HFP) используют одни и те же регистры с плавающей точкой. Один процессорный узел CPU имеет 16 регистров с плавающей точкой. Они используются для двоичных, десятичных и шестнадцатеричных вычислений и идентифицируются числами от 0 до 15 и указываются в поле R команд с плавающей точкой. Каждый регистр имеет 64 разряда и может содержать короткий (32-х разрядный) или длинный (64-х разрядный) операнд. Как показано на рисунке 2, пара регистров с плавающей точкой могут быть использована для расширенного (128-ми разрядного) операнда. Для более точных арифметических вычислений используются пары регистров, например, с номерами 0/2, 1/3,4/6, 5/7, и т.д.Каждая из восьми пар определяется меньшим номером регистра в паре.

#### 3.3. Регистр управления операциями с плавающей точкой

Регистр управления операциями с плавающей точкой FPC содержит маски исключений (IEEE exception) и флаги, он также определяет режим округления. Разрядность регистра – 32.

#### Регистры управления

Шестнадцать регистров управления CR используются операционной системой для управления прерываниями, виртуальной памятью, выполнением программ трассировки и т.д. Каждый регистр имеет 64 разряда. В режиме ESA/390 используются только младшие разряды 32 разряда (bits 32-63). Они идентифицируются числами от 0 до 15 и указываются в поле R команд управления LOAD CONTROL и STORE CONTROL. Эти команды могут адресовать сразу несколько регистров.

#### 3.5. Регистры доступа

Шестнадцать регистров доступа AR используются для задания адресных пространств. Каждый регистр имеет разрядность 32.

Каждый регистр имеет непрямое описание элемента ASCE (Address Space Control Element). Элемент ASCE – это параметр, используемый механизмом динамического преобразования адреса DAT (Dynamic Address Translation) для того, чтобы перенаправить ссылку к соответствующему адресному пространству. Когда центральный процессорный узел (CPU) находится в так называемом режиме регистров доступа, а такой режим, в свою очередь, контролируется разрядами в слове состояния программы PSW, поле В команды используется для определения логического адреса в обращениях к операнду памяти, и элемент ASCE определяет регистр доступа. Для некоторых команд поле В используется вместо поля R. Команды для загрузки и хранения содержимого регистров доступа и для перемещения содержимого регистров доступа от одного к другому.

Каждый из 16-ти регистров доступа может задать любое адресное пространство, включая пространство текущей команды – главное адресное пространство. Регистр доступа AR0 всегда задает текущее адресное пространство, его называют еще адресное пространство текущей команды. Когда один из регистров доступа AR1 - AR15 используется для задания адресного пространства, узел CPU определяет, какое адресное пространство задано, транслируя содержимое регистра доступа. Когда используется регистр доступа AR0 для задания адресного пространства, узел CPU рассматривает регистр доступа как задающий пространство текущей команды (то есть, главное адресное пространство) и не проверяет действительное содержимое регистра доступа.

Таким образом, 16 регистров доступа могут задавать одновременно одно главное адресное пространство и 15 других адресных пространств.

#### Регистр префикса

Регистр префикса PR используется для определения абсолютных адресов выделенного места в памяти для каждого процессора. Это 32-х разрядный регистр, доступ к нему имеет только операционная система.

Кроме перечисленных, каждый процессор имеет еще 128-ми разрядный регистр слова состояния процессора, 64-х разрядный регистр таймера и 32-х разрядный регистр программируемого поля часов реального времени TOD.

Графическое изображение регистровой модели показано на рисунке 3.3.

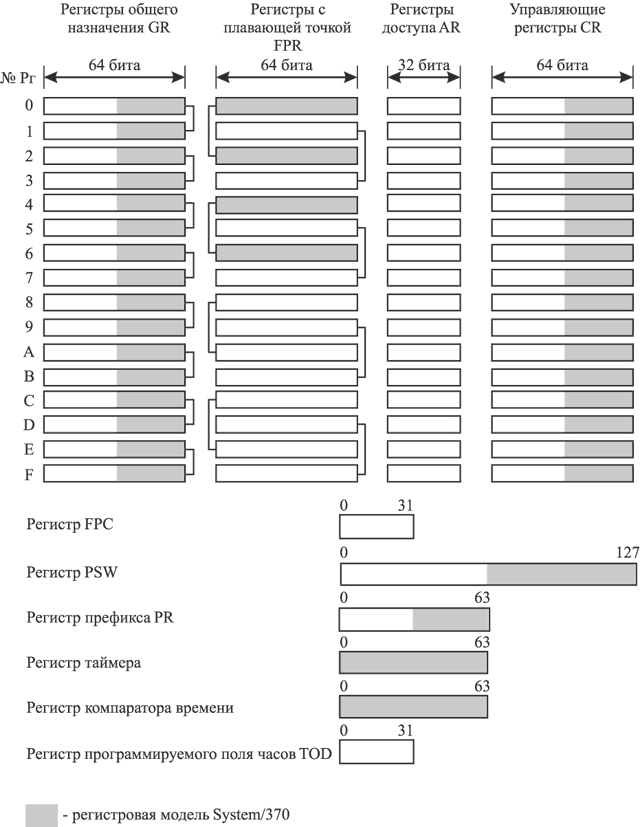


Рисунок 3.2 – Регистровая модель zArchitecture

**Замечание:** Стрелки на рисунке 3.3 показывают, что два регистра могут быть соединены как регистровая пара, задаваемая регистром меньшего номера в поле R. Например, пара регистров для операций с плавающей точкой FPR13 и FPR15 идентифицируются как 1101 в поле R.

#### Слово состояния процессора PSW

Слово состояния программы PSW включает в себя адрес команды, код условия и другую информацию, используемую для управления последовательностью команд и для определения состояния центрального процессора CPU.

Активное или управляющее слово PSW называется текущим PSW. Оно управляет программой во время ее исполнения. Процессор CPU имеет возможности управления прерываниями, которые позволяют ему быстро переключаться на другую программу в ответ на появление запросов на прерывание, исключительных событий или внешних воздействий. Когда случается прерывание, центральный процессор помещает текущее слово PSW в выделенное место в памяти, называемое ячейкой старого слова PSW, причем, конкретному классу прерываний выделяется свое место в памяти. Центральный процессор CPU выбирает новое слово PSW из другой заранее заданной ячейки памяти. Это новое слово PSW определяет следующую программу, которая должна быть выполнена. Когда процесс обслуживания прерывания будет закончен, программа, управляющая прерыванием, может заново загрузить старое слово PSW, сделав его опять текущим PSW. Таким образом, прерванная программа может быть продолжена.

Существует шесть классов прерывания: внешние (или от событий, которые произошли в периферийных устройствах), ввода-вывода, machine check, программные, рестарта и прерывания от супервизора. Каждый класс имеет пару ячеек памяти для записи старого и нового слов состояния программ, постоянно закрепленные в реальной памяти.

Формат слова состояния процессора показан на рисунке 3.3.

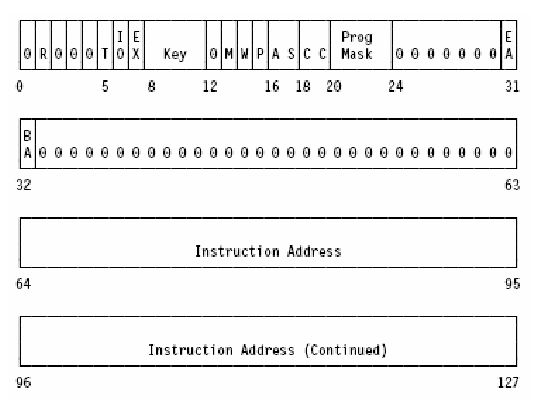
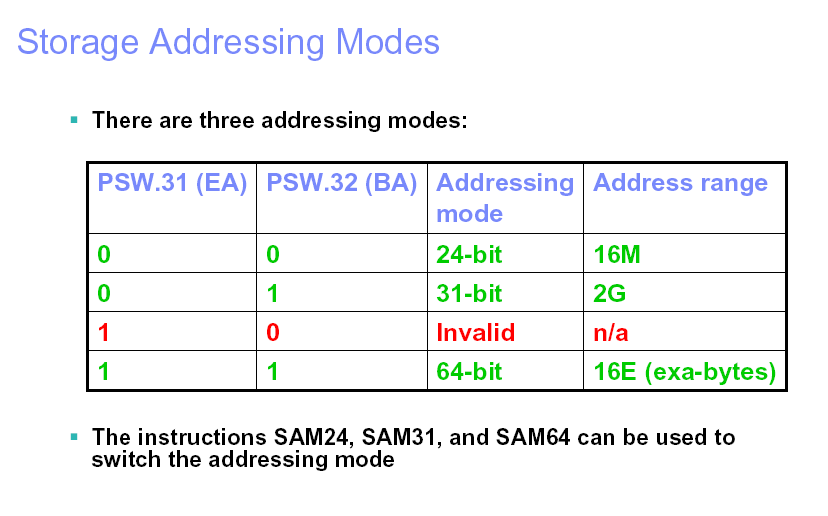


Рисунок 3.3 – Формат слова состояния процессора PSW

Cлово состояния процессора имеет 128 разрядов, поделенных на следующие поля:

* поле R, (разряд 1) определяет наличие или отсутствие записи программных событий (PER);
* поле Т (разряд 5) определяет наличие или отсутствие динамического преобразования адресов виртуальной памяти;
* поле IO (разряд 6) определяет разрешение или запрет прерывания по вводу-выводу;
* поле E (разряд 7) определяет наличие или отсутствие прерывания от внешних источников;
* поле Key (разряды 8-11) определяет режим защиты памяти;
* поле М (разряд 13) определяет наличие машинной проверки;
* поле W (разряд 14) говорит о наличии или отсутствии состоянии ожидания;
* поле Р (разряд 15) показывает, в каком режиме выполняются программы процессором (1 – пользовательский режим или 0 – режим супервизора);
* поле AS (разряды 16-17) определяет, какой режим адресного пространства используется:
* 00 – режим главного (первичного) адресного пространства;
* 01 – режим вторичного адресного пространства;
* 10 – режим адресного пространства с регистрами доступа;
* 11 – режим домашнего адресного пространства.
* поле СС (разряды 18-20) определяет код условия выполнения команды.
* поле Prog Mask (разряды 20-23) отвечает за маски прерывания программы:
* разряд 20: отслеживает переполнение при вычислениях с фиксированной точкой;
* разряд 21: отслеживает переполнение при вычислениях с плавающей точкой;
* разряд 22: отслеживает отрицательное переполнение при шестнадцатеричных вычислениях с плавающей точкой;
* разряд 23: отслеживает значащий разряд при шестнадцатеричных вычислениях с плавающей точкой;
* поле адреса команды Instruction Address начинается с 33-го разряда и определяется на основании длины текущей команды;
* поле EA (разряд 31) и поле BA (разряд 32) отвечают за расширенную адресацию памяти:
  + если бит ЕА=1 и ВА=1, значит, используется 64-разрядная адресация;
  + если ЕА=0, а ВА=0 , значит адресация 24-х разрядная;
  + если ЕА=0, а ВА=1 , значит адресация 31-разрядная.

В таблице 3.2. представлены три возможные режима адресации: 24-х разрядная, 31-разрядная и 64-х разрядная, и способы их задания с помощью перечисленных бит EA и ВА.



## 4. Два режима выполнения команд процессором

В архитектуре zSeries cуществуют два режима выполнения команд процессором: пользовательский и режим супервизора.

В пользовательском режиме (Problem state) программа не может выполнять привилегированные команды, у нее нет доступа к регистрам управления и к регистру таймера, она не управляет ключами защиты памяти, а имеет только доступ к «некритической» части слова состояния процессора PSW. Другими словами, в этом режиме отсутствует доступ к архитектурным возможностям, которые являются жизненно важными для системы в целом. В некоторых случаях, при выполнении определенных условий и авторизации программа пользовательского режима может выполнять определенные псевдо-привилегированные команды.

В режиме супервизора или, другими словами, в режиме ядра, программа может использовать все возможности архитектуры.

Переход из пользовательского режима в режим супервизора осуществляется через прерывание и загрузку нового слова состояния процессора PSW - такого, который будет показывать режим супервизора.

Переход из режима супервизора в пользовательский режим осуществляется с использованием привилегированной команды LOAD PSW [EXTENDED].

#### Целочисленные границы команд

Для «общения» программ с процессорами используются полуслова, слова, двойные слова и квадро-слова (см. рисунок 3.4). При этом команды всегда определяются в рамках границ в полуслово. Операнды непривилегированных команд обычно не имеют такого жесткого требования и могут быть заданы на основании внутренних границ. Исключение составляют только две команды: Compare и Swap.

Операнды привилегированных команд обычно должны быть заданы в рамках интегральных границ.

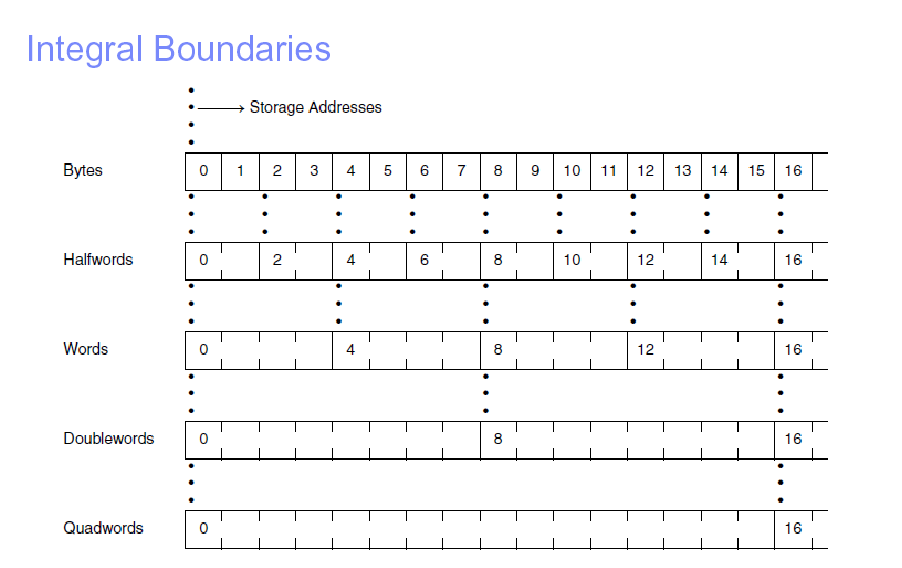


Рисунок 3.4 – Целочисленные границы команд

#### Форматы адресов и команд

Команды бывают размером в 2, 4 или 6 байт в длину, причем, длина команды всегда кратна полуслову. Первые два бита команды определяют ее длину следующим образом:

* 00 : 2 байта
* 01 или 10: 4 байта
* 11: 6 байт.

Код операции состоит из 8, 12 или 16 бит, причем, 8-ми разрядный код операции всегда находится в первом байте: 12-ти разрядный код операции занимает первый байт и вторую половину второго байта. 16-битовый код операции занимает первый байт, и либо второй байт, либо последний байт 6-байтной команды.

На рисунке 3.5 приведены все возможные форматы команд. Серым цветом выделены команды базовой архитектуры. Команды предназначены для управления регистрами.

Обозначения полей в форматах команд, приведены в таблице 3.3.

|  |  |
| --- | --- |
| **Таблица 3.3. Поля, используемые в форматах команд z/Architecture** | |
| **Обозначение** | **Назначение** |
| COP | Код операции команды, определяющий тип операции |
| 1, 2, 3, 4 | Индексы, указывающие на 1, 2, 3 и 4 операнды команды |
| R1, R2, R3 | Номера регистров |
| B1, B2, B4 | Номера базовых регистров |
| X2 | Номера индексных регистров |
| D1, D2, D4, DL1, DH1, DL2, DH2 | Поля смещения (L - младшая часть смещения, H - старшая часть смещения) |
| M1, M3, M4 | Поля маски |
| I, I2 | Поля непосредственных операндов |
| L, L1, L2 | Поля длины операндов |

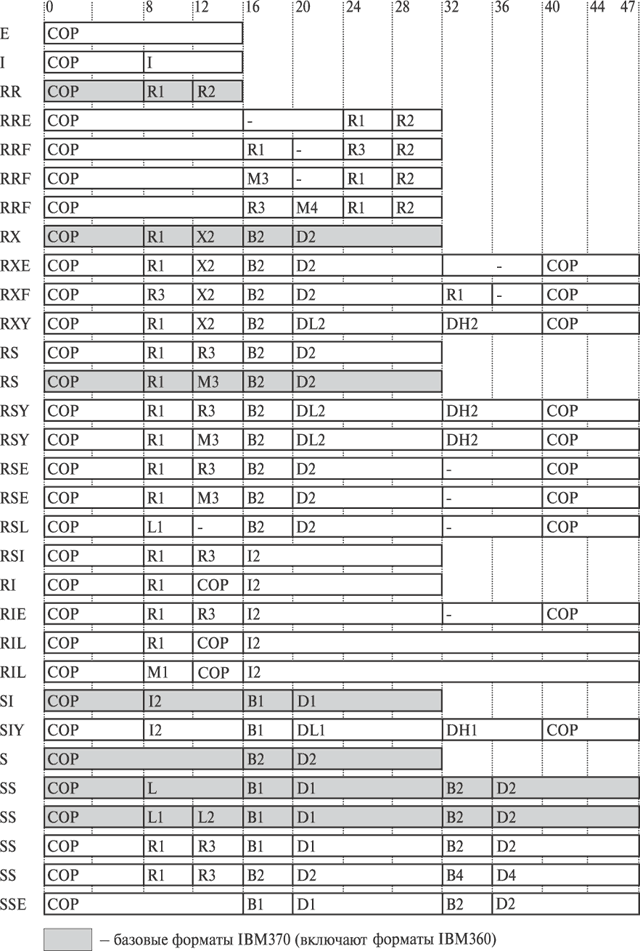


Рисунок 3.5 – Форматы команд архитектуры zArchitecture

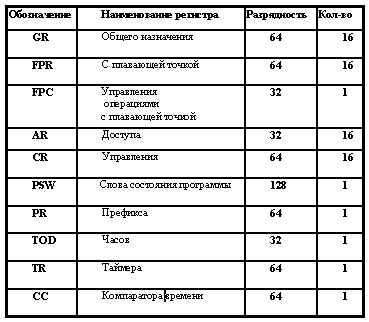
## Краткие выводы по главе «Архитектура процессора zSeries»

Оглянемся назад? 

1. Ключевые особенности архитектуры zArchitecture:

* **Преемственность, совместимость снизу вверх** - архитектуры 24-х и 31-разрядные встроены в 64-х разрядную архитектуру
* **Постоянная готовность к работе –** за счет многократного дублирования вычислительных операций
* **Слово состояния программы PSW** - 128- разрядное может хранить команды с большими адресами
* **Динамическое преобразование адресов DAT**– пятиуровневое, таблицы трех регионов используются для преобразования 64-х разрядных виртуальных адресов
* **Элемент ASCE** создает новые адресные пространства
* **Префиксная область** увеличена до 8КВ для хранения старых и новых PSW и значений регистров
* **Работа подсистемы ввода-вывода** – обмен блоками по 2 кбайта или по 4 кбайта

1. Базовый состав архитектуры zSeries:
   1. набор центральных процессоров CP с процессорной памятью;
   2. основная память MS;
   3. дополнительная память ES;
   4. канальная подсистема CSS
   5. внешний таймер ETR
2. Классы команд архитектуры zSeries:
   * + 1. **Общие команды -** оперируют данными в двоичном формате
       2. **Десятичные команды -** оперируют данными в десятичном формате
       3. **Команды двоичных операций с поддержкой плавающей точки ВFP -** оперируют данными двоичного формата
       4. **Команды шестнадцатеричных операций с поддержкой плавающей точки HFP -** оперируют данными шестнадцатеричного формата
       5. **Команды с поддержкой плавающей точки FPS -** оперируют данными независимо от формата
       6. **Команды управления -** выполняются в режиме супервизора
       7. **Команды ввода-вывода -** выполняются в режиме супервизора
3. Регистровая модель процессора представлена в таблице



Оглянемся назад? 

1. Слово состояния процессора имеет 128 разрядов, может быть
   1. **Текущее PSW** - управляет программой во время ее исполнения,
   2. **Старое PSW** - прерванной программы сохраняется в памяти и
   3. **Новое PSW** - определяет программу, которая должна быть выполнена в результате прерывания.
2. Два режима выполнения команд процессором:

**Пользовательский режим -** не выполняются привилегированные команды

**Режим супервизора -**  выполняются привилегированные команды

1. Целочисленные границы команд определяются как:

* Полуслово = 2 байта;
* Слово = 4 байта;
* Двойное слово = 8 байт;
* Квадро-слово = 16 байт.

**Новые термины и определения** (с номерами страниц, где они встречаются):

Преемственность архитектур, 2 Совместимость, 2

Дублирование вычислений, 2 Слово состояния программы PSW, 3 Таблицы регионов, 3 Динамическое преобразование адресов DAT, 3

Элемент ASCE, 3, 16 Префиксная область, 4 Внешний таймер ETR, 5

Контрольный блок, Канальный путь, 6 Динамический переключатель, 6 Конфигурация , 6 Процессорная память, 6 Кэш-память, 6

Основная память, 7 Ассоциативная кэш-память, 7 Расширенная память, 10 Центральный процессорный узел, 11 Общие команды, 11 Десятичные команды, 11

Команды двоичных операций с поддержкой плавающей точки ВFP, 12

Команды шестнадцатеричных операций с поддержкой плавающей точки HFP, 12

Команды управления, 12 Команды ввода-вывода, 12

Регистр общего назначения GR, 13 Идентификатор регистра общего назначения GR, 14

Регистр с плавающей точкой FPR, 15 Регистр управления, 15

Регистр управления операциями с плавающей точкой FPC, 15 Регистр доступаAR, 16 Регистр префикса PR, 17 Текущее PSW, 18 Старое PSW, 18 Новое PSW, 18

Регистр управления СR, 7 Пользовательский режим, 21

Режим супервизора, 21

## Термины и определения

## Вопросы для контроля усвоения материала главы 3

1. Перечислите ключевые особенности архитектуры zArchitecture.
2. Из каких компонентов состоит базовый набор архитектуры zArchitecture?
3. Какие типы регистров содержит процессор ZSeries?
4. Для чего служит регистр префикса?
5. Сколько разрядов в слове состояния процессора?
6. Назовите два режима выполнения команд процессором.
7. Что такое целочисленная граница слова?

**Содержание главы**

[Глава 4. Микроархитектура процессоров zSeries 2](#_Toc237520226)

[1. Типовая архитектура мультипроцессорной системы с общей памятью 2](#_Toc237520227)

[1.1. Способы размещения строк в кэш-памяти 4](#_Toc237520228)

[1.2 Два условия обеспечения когерентности 5](#_Toc237520229)

[1.3. Протоколы когерентности кэш-памяти и способы размещения строк 6](#_Toc237520230)

[1.3.1. Протоколы записи с аннулированием 7](#_Toc237520231)

[1.3.2. Протокол записи с обновлением 7](#_Toc237520232)

[1.3.3. Сквозная запись 7](#_Toc237520233)

[2. Алгоритм MESI 8](#_Toc237520234)

[3. Структура дублированного процессора zSeries 10](#_Toc237520235)

[4. Функциональная схема процессора 15](#_Toc237520236)

[4.1. Блок команд 15](#_Toc237520237)

[2.2. Блок операций 15](#_Toc237520238)

[3.3. Блок управления BCE 16](#_Toc237520239)

[3.4. Сопроцессор 17](#_Toc237520240)

[3.4.1. Блок сжатия 17](#_Toc237520241)

[3.4.1. Блок транслятора 18](#_Toc237520242)

[3.4. Блок восстановления 18](#_Toc237520243)

[5. Краткие выводы по главе «Микроархитектура процессоров zSeries» 20](#_Toc237520244)

[6. Вопросы для контроля усвоения материала 21](#_Toc237520245)

[7. Литература 21](#_Toc237520246)

# Глава 4. Микроархитектура процессоров zSeries

Что нас ждет? BD04972_

После прочтения этой главы Вы сможете:

1. объяснить причину возникновения проблемы с когерентностью кэш-памяти в мультипроцессорной архитектуре с общей памятью;
2. дать определение когерентной кэш-памяти;
3. назвать три способа размещения строк в кэш-памяти;
4. назвать два класса протоколов, обеспечивающих когерентность кэш-памяти;
5. объяснить назначение алгоритма MESI;
6. перечислить основной состав структуры дублированного процессора zSeries;
7. сказать, что такое конвейеризация, и какие преимущества она дает;
8. объяснить, почему нужен буфер адресов переходов при распаковке команд ветвления в блоке команд;
9. привести пример алгоритма, реализующего словарный метод сжатия.

## Типовая архитектура мультипроцессорной системы с общей памятью

Сервер zSeries имеет до 20-ти и более процессоров, размещенных на одной плате МСМ. При этом все процессоры используют общую физическую (основную) память. Архитектура zSeries – это пример использования многоуровневой кэш-памяти. Первый уровень – находится внутри процессорного чипа. Второй уровень – находится внутри многочипового модуля МСМ.

При этом кэш-память первого уровня содержит разделяемые, так и частные данные. Частные данные - это данные, которые используются одним процессором, в то время как разделяемые данные используются многими процессорами, по существу именно таким способом обеспечивая обмен между ними.

Когда кэшируется элемент частных данных, их значение переносится в кэш для сокращения среднего времени доступа, а также требуемой полосы пропускания. Поскольку никакой другой процессор не использует эти данные, этот процесс идентичен процессу для однопроцессорной машины с кэш-памятью.

Если кэшируются разделяемые данные, то разделяемое значение реплицируется и может содержаться одновременно в кэшах нескольких процессоров.

Кроме сокращения задержки доступа и требуемой полосы пропускания такая репликация данных способствует также общему сокращению количества обменов.

Однако кэширование разделяемых данных вызывает новую проблему: когерентность кэш-памяти.

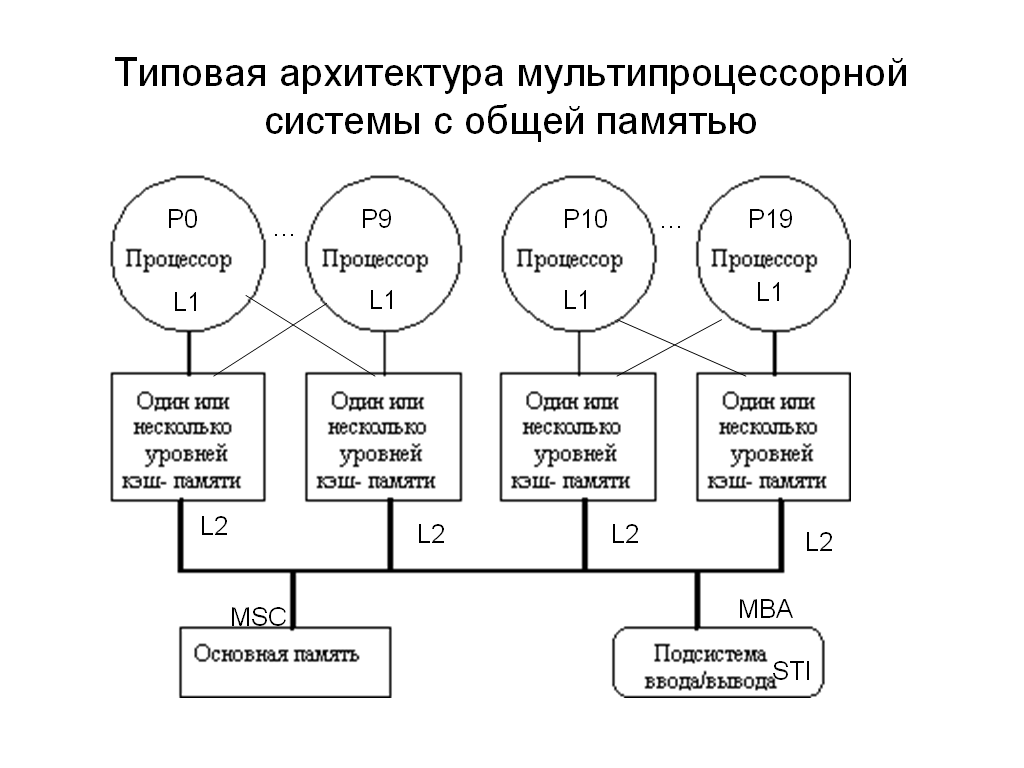


Рисунок 4.1 - Типовая архитектура мультипроцессорной системы с общей памятью

Когерентность кэш-памяти - это согласованное состояние памяти, гарантия того, что любое считывание элемента данных возвращает последнее по времени записанное в этот элемент значение.

Как показано на рисунке 4.1, типовая архитектура мультипроцессорной системы с общей памятью имеет несколько уровней кэш-памяти, то есть, буферной памяти. Первый уровень кэш-памяти – это буферная память, встроенная в процессор, следующие несколько уровней памяти – это промежуточная память между основной памятью и процессорами. Доступ к каждой такой кэш-памяти имеют все процессоры, то есть, они могут считывать и записывать информацию в кэш-память. Встает вопрос: как исключить такие ситуации, когда, например, два или больше процессора обращаются (хотят записать информацию) в одну и ту же ячейку памяти?

Проблема, о которой идет речь, возникает из-за того, что значение элемента разделяемых данных в памяти, хранящееся в двух разных процессорах, доступно этим процессорам только через их индивидуальную кэш-память. На рисунке 4.2 показан простой пример, иллюстрирующий эту проблему. На рисунке 4.2 показаны три процессора CP1, CP2 и CP3, в ячейках А’ и B’ записаны кэшированные копии элементов А и В основной памяти. Для процессора ЦП1 кэш и память когерентны (вариант а). Для процессора CP2 предполагается использование кэш-памяти с отложенным обратным копированием, когда CP2 записывает значение 550 в ячейку А’. В результате ячейка А’ хранит новое значение, а в основной памяти в ячейке А осталось старое значение 100. При попытке вывода А из памяти будет получено старое значение. Вариант б) – некогерентности кэш и памяти.

В третьем варианте (в) – подсистема ввода-вывода вводит в ячейку памяти В новое значение 440, а в кэш-памяти осталось старое значение В – 200.

Проблема когерентности памяти состоит в необходимости гарантировать, что любое считывание элемента данных возвращает последнее по времени записанное значение.

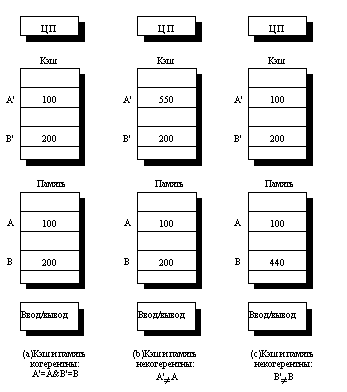


Рисунок 4.2 – Примеры вариантов когерентности и некогерентности кэш и памяти

### 1.1. Способы размещения строк в кэш-памяти

Существует три основных способа размещения блоков (строк) в кэш-памяти: кэш-память с прямым отображением (direct-mapped cache), частично ассоциативная кэш-память или множественно ассоциативная кэш-память (set-associative cache) и полностью ассоциативная кэш-память (fully associative cache).

Кэш-память называется с кэш-памятью с прямым отображением, если каждый блок основной памяти имеет только одно фиксированное место, на котором он может появиться в кэш-памяти. Это наиболее простая организация кэш-памяти. Все блоки основной памяти, имеющие одинаковые младшие разряды в своем адресе, попадают в один блок кэш-памяти. При таком подходе справедливо соотношение: (адрес блока кэш-памяти) = (адрес блока основной памяти) mod (число блоков в кэш-памяти).

Кэш-память называется полностью ассоциативной, если некоторый блок основной памяти может располагаться на любом месте кэш-памяти.

Кэш-память называется частично ассоциативной, если некоторый блок основной памяти может располагаться на ограниченном множестве мест в кэш-памяти.

В современных процессорах, как правило, используется либо кэш-память с прямым отображением, либо двух - (четырех-) канальная множественно-ассоциативная кэш-память.

Определение когерентности кэш-памяти не совсем корректно, поскольку невозможно требовать, чтобы операция считывания мгновенно видела значение, записанное в этот элемент данных некоторым другим процессором. Если, например, операция записи на одном процессоре предшествует операции чтения той же ячейки на другом процессоре в пределах очень короткого интервала времени, то невозможно гарантировать, что чтение вернет записанное значение данных, поскольку в этот момент времени записываемые данные могут даже не покинуть процессор.

Вопрос о том, когда точно записываемое значение должно быть доступно процессору, выполняющему чтение, определяется выбранной моделью согласованного (непротиворечивого) состояния памяти и связан с реализацией синхронизации параллельных вычислений.

Поэтому с целью упрощения предположим, что мы требуем только, чтобы записанное операцией записи значение было доступно операции чтения, возникшей немного позже записи и что операции записи данного процессора всегда видны в порядке их выполнения.

### Два условия обеспечения когерентности

С этим простым определением согласованного состояния памяти мы можем гарантировать когерентность путем обеспечения двух условий.

Условие 1. Операция чтения ячейки памяти одним процессором, которая следует за операцией записи в ту же ячейку памяти другим процессором получит записанное значение, если операции чтения и записи достаточно отделены друг от друга по времени.

Условие 2. Операции записи в одну и ту же ячейку памяти выполняются строго последовательно (иногда говорят, что они сериализованы): это означает, что две подряд идущие операции записи в одну и ту же ячейку памяти будут наблюдаться другими процессорами именно в том порядке, в котором они появляются в программе процессора, выполняющего эти операции записи.

Первое условие очевидно связано с определением когерентного (согласованного) состояния памяти: если бы процессор всегда бы считывал только старое значение данных, мы сказали бы, что память некогерентна.

Необходимость строго последовательного выполнения операций записи является более тонким, но также очень важным условием.

Представим себе, что строго последовательное выполнение операций записи не соблюдается. Тогда процессор CP1 может записать данные в ячейку, а затем в эту ячейку выполнит запись процессор CP2.

Если последовательность операций записи не соблюдается, то может возникнуть ситуация, когда какой-нибудь процессор будет наблюдать сначала операцию записи процессора CP2, а затем операцию записи процессора CP1, и будет хранить это записанное CP1 значение, хотя на его месте будет уже значение, записанное процессором CP2.

Как создать разумную модель порядка выполнения программ и когерентности памяти для пользователя – это и есть проблема.

Представьте, что третий процессор постоянно читает ту же самую ячейку памяти, в которую записывают процессоры CP1 и CP2; он должен наблюдать сначала значение, записанное CP1, а затем значение, записанное CP2. Возможно, он никогда не сможет увидеть значения, записанного CP1, поскольку запись от CP2 возникла раньше чтения. Если он даже видит значение, записанное CP1, он должен видеть значение, записанное CP2, при последующем чтении.

Подобным образом любой другой процессор, который может наблюдать за значениями, записываемыми как CP1, так и CP2, должен наблюдать идентичное поведение. Простейший способ добиться выполнения таких условий заключается в строгом соблюдении порядка операций записи: чтобы все записи в одну и ту же ячейку могли наблюдаться в том же самом порядке. Это условие и называется последовательным выполнением (сериализацией) операций записи (write serialization).

Вопрос о том, как процессор должен увидеть значение, записанное другим процессором, достаточно сложен и имеет заметное воздействие на производительность, особенно в больших машинах. Решается этот вопрос по разному – перейдем к рассмотрению методов поддержания когерентности кэш-памяти.

### 1.3. Протоколы когерентности кэш-памяти и способы размещения строк

Для того, чтобы соблюсти строгую очередность и порядок в записях (сериализацию), разработаны два класса протоколов:

* Протоколы на **основе справочника** (directory based). Информация о состоянии блока физической памяти содержится только в одном месте, называемом справочником, хотя физически справочник может быть распределен по узлам системы.
* Протоколы **наблюдения** (snooping). Каждый кэш, который содержит копию данных некоторого блока физической памяти, имеет также соответствующую копию служебной информации о его состоянии. Централизованная система записей отсутствует.

#### Протоколы записи с аннулированием

Один из методов заключается в том, чтобы гарантировать, что процессор должен получить исключительные права доступа к элементу данных перед выполнением записи в этот элемент данных. Этот тип протоколов называется протоколом *записи с аннулированием (write ivalidate protocol),* поскольку при выполнении записи он аннулирует другие копии.

Это наиболее часто используемый протокол. Исключительное право доступа гарантирует, что во время выполнения записи не существует никаких других копий элемента данных, в которые можно писать или из которых можно читать: все другие кэшированные копии этого элемента данных аннулированы.

#### Протокол записи с обновлением

Альтернативой протоколу записи с аннулированием является обновление всех копий элемента данных в случае записи в этот элемент данных. Этот тип протокола называется протоколом *записи с обновлением (write update protocol)* или протоколом *записи с трансляцией (write broadcast protocol).* Обычно в этом протоколе для снижения требований к полосе пропускания отслеживается, является ли слово в кэш-памяти разделяемым объектом, или нет, а именно, содержится ли оно в других кэшах. Если нет, то нет никакой необходимости обновлять другой кэш или транслировать в него обновленные данные.

Эти два альтернативных метода имеют много общего с методами работы кэш-памяти со сквозной записью и с записью с обратным копированием, которые применяются в архитектуре zArchitecture.

#### Сквозная запись

При сквозной записи память обновляется почти мгновенно после записи (возможно с некоторой задержкой в буфере записи). Подобным образом при использовании протокола записи с обновлением другие копии обновляются так быстро, насколько это возможно. Наиболее важное отличие в производительности протоколов записи с аннулированием и с обновлением связано с характеристиками прикладных программ и с выбором размера блока.

В кэш-памяти со сквозной записью последнее значение элемента данных найти легко, поскольку все записываемые данные всегда посылаются также и в память, из которой последнее записанное значение элемента данных может быть выбрано.

К недостаткам этого протокола можно отнести то, что наличие буферов записи может привести к некоторому усложнению.

Эффективность той или иной системы кэш-памяти зависит от стратегии управлений памятью. Стратегия управления памятью подразумевает ответ на такие вопросы: выбор метода отображения основной памяти в кэше; алгоритм взаимодействия между медленной основной и быстрой кэш-памятью; выбор стратегии замещения информации в кэше.

Один ответ на эти вопросы применительно к большим ЭВМ класса мейнфрейм – реализация алгоритма MESI (Modified, Exclusive, Shared, Invalid), которую мы рассмотрим в следующем разделе.

## Алгоритм MESI

Для обеспечения когерентности памяти вычислительной системы с раздельными кэшами и общей памятью можно использовать алгоритм MESI (Modified, Exclusive, Shared, Invalid) [5]. Алгоритм MESI представляет собой организацию когерентности кэш-памяти с обратной записью. Этот алгоритм предотвращает лишние передачи данных между кэш-памятью и основной памятью. Так, если данные в кэш-памяти не изменялись, то незачем их пересылать.

Каждая строка кэш-памяти может находиться в одном из следующих состояний:

* М - строка модифицирована (доступна по чтению и записи только в этом процессоре, потому что модифицирована командой записи по сравнению со строкой основной памяти);
* Е - строка монопольно копированная (доступна по чтению и записи в этом процессоре и в основной памяти);
* S - строка множественно копированная или разделяемая (доступна по чтению и записи в этом кэше, в основной памяти и в кэш-памятях других процессоров, в которых содержится ее копия);
* I - строка, невозможная к использованию (строка не доступна ни по чтению, ни по записи).

Информация о состоянии строки используется, во-первых, для определения процессором возможности локального, без выхода на шину, доступа к данным в кэш-памяти, а, во-вторых, - для управления механизмом когерентности.

Для управления режимом работы механизма поддержки когерентности в слове состояния процессора используется бит WT: состояние 1 которого задает режим сквозной (write-through) записи, а состояние 0 - режим обратной (write-back) записи в кэш-память.

Промах чтения в кэш-памяти заставляет вызвать строку из основной памяти и сопоставить ей состояние Е или S. Кэш-память заполняется только при промахах чтения. При промахе записи транзакция записи помещается в буфер и посылается в основную память при предоставлении шины.

|  |  |  |
| --- | --- | --- |
| **Исходное состояние строки** | **Состояние после чтения** | **Состояние после записи** |
| **I** | **Если WT=1, тогда Е, иначе S; Обновление строки путем ее чтения из основной памяти** | **Сквозная запись в основную память; I** |
| **S** | **S** | **Сквозная запись в основную память; Если WT=1 тогда Е, иначе S** |
| **Е** | **Е** | **М** |
| **М** | **М** | **М** |

Если установлено состояние I и WT=0, команда чтения данных из этой строки вызывает чтение строки из основной памяти, размещение ее в кэш-памяти и изменение состояния этой строки в кэш-памяти на Е или S. Состояние Е будет, если установлен режим сквозной записи, при котором запись производится и в строку кэш-памяти и в строку основной памяти. Состояние S устанавливается при режиме обратной записи, что позволяет модифицировать данные строки кэш-памяти без немедленной модификации строки основной памяти, что, в свою очередь, увеличивает производительность. До тех пор, пока к данным строки не будет доступа для других процессоров или внешних устройств, не будет и обратной записи, и процессор не будет использовать шину.

Если установлено состояние I и при этом WT=1, строки команда записи в эту строку изменяет только содержимое строки основной памяти (сквозная запись), но не изменяет содержимое кэш-памяти и сохраняет состояние строки I.

В состоянии S строки чтение данных из этой строки не меняет ее состояние. Если установлен режим сквозной записи, то после завершения записи состояние строки меняется на Е, при режиме обратной записи выполняется сквозная запись, но состояние строки остается прежним S.

Если состояние строки Е, то чтение сохраняет это состояние, а запись переводит строку в состояние М.

Наконец, если состояние строки М, то и команды чтения, и команды записи не меняют этого состояния.

Для поддержки когерентности строк кэш-памяти при операциях ввода/вывода и обращениях в основную память других процессоров на шине генерируются специальные циклы опроса состояния кэш-памятей. Эти циклы опрашивают кэш-памяти на предмет хранения в них строки, которой принадлежит адрес, используемый в операции, инициировавшей циклы опроса состояния. Возможен режим принудительного перевода строки в состояние I, который задается сигналом INV. При этом состояние строк определяется таблицей 3.

|  |  |  |
| --- | --- | --- |
| **Исходное состояние** | **INV=0** | **INV=1** |
| **I** | **I** | **I** |
| **S** | **S** | **I** |
| **Е** | **S** | **I** |
| **М** | **S; обратная запись строки** | **I; обратная запись строки** |

## Структура дублированного процессора zSeries

Процессоры серверов z/Architecture имеют дублированную структуру с зеркальным исполнением команд в двух процессорных каналах (см. рисунок 4.3).

Процессор этой серии содержит блоки команд и операций, сопроцессоры, блок управления и блок восстановления, который, в свою очередь, имеет в своем составе блок сжатия.

Каждый такой канал состоит из последовательно включенных блока команд (I-unit) и блока операций (E-unit). Блок команд I предназначен для выборки и дешифрации команд из кэш-памяти команд первого уровня (L1-I), вычисления адресов операндов и запуска процессов их выборки из кэш-памяти данных первого уровня (L1-D), и формирования очереди команд для исполнения в блоке E. Блок операций Е выполняет операции, заданные в командах, над операндами из регистров или памяти и запись результатов в регистры или память. Дополнительные тракты обработки реализованы в сопроцессоре COP.

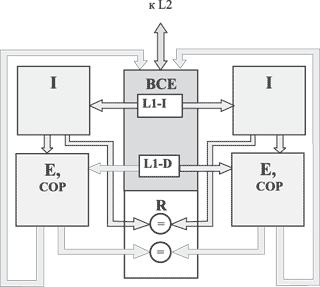


Рисунок 4.3 – Структура дублированного процессора zSeries

КЭШ-память первого уровня реализована в одном чипе с процессорными каналами и в моделях архитектуры z/Architecture разделена на две независимые КЭШ команд и КЭШ данных. В предшествующих архитектурах использовалась объединенная КЭШ команд и данных. Обе КЭШ и их контроллеры сосредоточены в блоке управления BCE (buffer control element).

Каждый из процессорных каналов имеет конвейерную организацию (рисунок 4.4) с оптимальным срабатыванием уровней за один такт. При исполнении сложных команд или в случае конвейерных сбоев отдельные уровни срабатывают за большее число тактов.

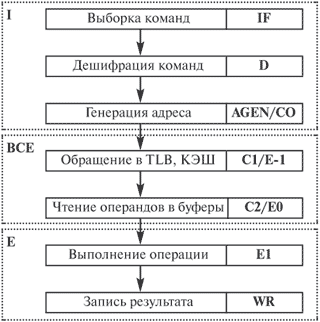


Рисунок 4.4 - Конвейерная организация работы процессора

Конвейерная обработка представляет собой такой вид обработки данных, когда интервал времени, требуемый для прохождения процесса через какой-либо функциональный узел вычислительной системы (например, через арифметико-логическое устройство с плавающей запятой), продолжительнее, чем интервалы, в которые эти данные могут быть введены в этот функциональный узел. Тогда этот функциональный узел выполняет процесс в несколько этапов: при завершении первого этапа данные передаются на второй этап, в на первый поступают новые данные. Технология конвейеризации обеспечивает быструю обработку последовательных процессов, хотя и требует более сложной реализации управляющего устройства, ведущего учет выполняемых одновременно операций [3].

Задание для самостоятельной работы: Для понимания принципа конвейеризации проведите аналогию с обработкой ягод для получения вина (сбор, очистка, сушка, выжимка сока) и разгрузкой кирпича.

Основные функции первых трех уровней конвейера реализуются в блоке команд I. Первый уровень (IF) реализует выборку команд за несколько тактов, и иногда рассматривается как отдельный блок, а не как уровень конвейера. Выбранные из КЭШ L1-I (КЭШ L2 или основной памяти) команды помещаются в буфер команд, из которого передаются во второй уровень (D) для распаковки и дешифрации. На этой стадии из заданных в адресной части регистров считываются компоненты адреса (база, индекс) операнда, и команда передается в очередь команд для исполнения в блоке операций Е.

Уровень конвейера (AGEN) используется для вычисления логического адреса операнда путем сложения базы, индекса и смещения. Конвейер оптимизирован для двухадресных команд формата RX с размещением одного операнда в регистре, а второго - в памяти. Вычисленный адрес передается в блок BCE для обращения в память.

В блоке управления BCE реализуются еще два уровня конвейера С1 и С2, связанных с выборкой операндов из КЭШ L1-D (КЭШ L2 или основной памяти). Первый из этих уровней запускает одновременные процессы обращения в директорию КЭШ и буфер преобразования адреса TLB, а второй выполняет чтение операнда и размещение его в буфере операндов в блоке операций E с подстыковкой к ранее считанным операндам.

В блоке операций Е выполняются операции команд, считываемых из очереди команд, над операндами, находящимися в программно-доступных регистрах и буфере операндов. Для этого используются два уровня, первый из которых (E1) реализует выполнение операции в арифметико-логическом устройстве процессора (АЛУ), а второй (WR) - сохранение результата в регистре или в памяти. Выполнение большинства операций в АЛУ с фиксированной точкой занимает один такт, а в АЛУ с плавающей точкой - три такта, для сложных операций число тактов может достигать 100 и более. Для ускорения операций в АЛУ используется внутренний конвейер. Запись результата операций осуществляется в регистры или КЭШ-память.

Рассмотрим функциональную схему процессора – состав блока команд, блока операций, блока управления, сопроцессора и блока восстановления и их взаимосвязи. Функциональная схема процессора приведена на рисунке 4.5.

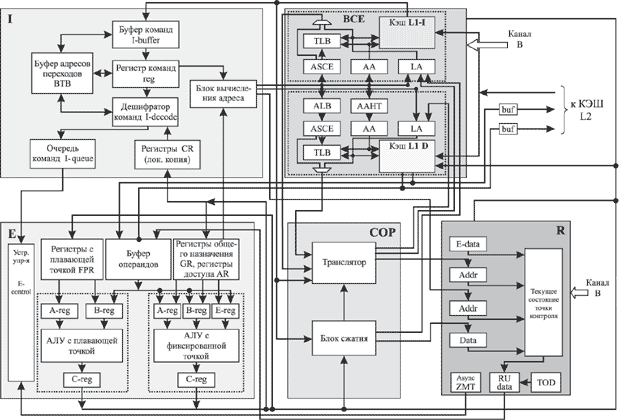


Рисунок 4.5 – Функциональная схема процессора zSeries

## Функциональная схема процессора

### Блок команд

Блок команд **I** процессора архитектуры z/Architecture разработан так, что поддерживает три режима адресации (24, 31, 64 разрядную адресацию). Буфер команд I-buffer предназначен для хранения последовательных участков исполняемых программ и состоит из нескольких фрагментов, каждый из которых содержит одну строку из КЭШ-памяти команд. Каждая из строк в буфере распаковывается и передается по командно в регистр команд. После завершения распаковки очередной строки запускается чтение новой строки из КЭШ команд. Чтение из КЭШ выполняется с использованием буфера адресов переходов BTB (branch target buffer) емкостью 8К строк для ускорения формирования адресов в командах условного перехода путем их предсказания. Буфер адресов переходов BTB представляет собой таблицу, каждая строка которой содержит информацию, занесенную при выполнении команды ветвления: адрес команды ветвления, фактически использованный адрес перехода и другую вспомогательную информацию.

**Замечание:** Команда ветвления – это команда, прерывающая последовательное выполнение команд в программе, пример такой команды - Jump

При выполнении очередной команды ветвления по адресу этой команды осуществляется поиск в таблице, и при успешном поиске из соответствующей строки буфера адресов переходов BTB считывается предсказываемый адрес перехода. Далее запускается выборка из КЭШ соответствующей ветви программы до момента фактического вычисления адреса ветвления и его сравнения с предсказанным адресом перехода. При совпадении конвейер процессора продолжает функционировать без задержки, в противном случае выполняется восстановление работы конвейера, требующее нескольких дополнительных тактов. Для команд ветвления, завершающих циклические участки программ, вероятность успешного предсказания перехода высока и определяется одним - двумя неправильными предсказаниями на все число повторений цикла. Отключение буфера адресов переходов BTB исключает "лишние" обращения в КЭШ и может использоваться для разгрузки КЭШ в пользу других абонентов.

### 2.2. Блок операций

Блок операций **E** включает в себя регистровую память и два арифметико-логических устройства АЛУ для выполнения операций с фиксированной и плавающей точкой. В качестве источников операндов АЛУ используются программно доступные регистры общего назначения GR и регистры с плавающей точкой FPR, регистры доступа AR, а также буфер операндов, считанных из памяти. Результаты операций могут быть сохранены в регистрах GR, FPR или в памяти.

АЛУ для операций с плавающей точкой (FPU) включает 2 входных регистра A-reg, B-reg и выходной регистр C-reg. Это 64-разрядные регистры с конвейеризированным трактом обработки чисел в форматах HFP, BFP. Конвейер FPU состоит из 5 уровней и выполняет большинство операций с пропускной способностью одна операция за один такт.

Ускорение операций в АЛУ достигается введением конвейерной обработки, дополнительные уровни которой (**E-1**, **E0** на рисунке 2) размещены в уровнях основного конвейера, предшествующих выполнению операций в АЛУ. Уровень **E-1** формирует многоразрядное управляющее слово для выборки операндов из регистровой памяти и формирования маски. Уровень **E0** используется для чтения операндов.

### 3.3. Блок управления BCE

Блок управления обращениями в память **BCE** (Buffer Control Element) включает две независимые КЭШ команд и данных, а также интерфейс с КЭШ второго уровня L2. Обе КЭШ имеют одинаковый объем 256 KB, сходные схемы управления и относятся к частично-ассоциативным (four-way-set-associative) КЭШ с размером строки 256 байт. Разрядность выборки из КЭШ команд - четыре слова (16 байт). КЭШ данных допускает сквозную (write-through) запись и расслоена на два блока, допускающих одновременные обращения двойными словами, например, для выполнения чтения из одного и записи в другой блок. Обмен данными между КЭШ команд и данных, например для записи из КЭШ данных в КЭШ команд, может выполняться через КЭШ второго уровня L2, допускающую обратную (write-back) запись и реализующую протокол поддержки когерентности содержимого КЭШ команд и данных всех процессоров системы.

Поскольку КЭШ-память функционирует в абсолютном адресном пространстве, а блок команд I формирует логические адреса (LA), необходимо преобразование адреса с использованием в общем случае таблиц из буферов ALB, TLB, региональных, сегментных и страничных таблиц. Уменьшение времени трансляции адреса достигается за счет ее конвейеризации, это этапы C0, C1, C2 на рисунке 4.2. Кроме этого в КЭШ данных предусмотрена таблица AAHT (Absolute Address History Table), содержащая ранее использованные абсолютные адреса (AA) и компоненты соответствующих им логических адресов (LA).

При обращении на этапе C0 в блоке команд I по содержимому регистров, содержащих компоненты адреса, формируется индекс для поиска в таблице AAHT для предсказания части разрядов абсолютного адреса. При попадании в AAHT на этапе C1 выполняется обращение в директорию и запоминающий массив КЭШ.

При промахе на этапе C1 таблица AAHT модифицируется и осуществляется обращение в буфер преобразования адреса TLB с использованием соответствующего кода управления адресным пространством ASCE, получаемого из таблицы ALB на этапе C0 или соответствующим преобразованием. Если поиск в TLB не дает положительного результата, в блоке транслятора сопроцессора COP запускается полное преобразование адреса с использованием соответствующих таблиц в памяти.

В КЭШ команд абсолютные адреса формируются аналогично, однако абсолютные адреса могут быть предсказаны не по таблице AAHT, а из буфера BTB в блоке выборки команд.

При попадании в КЭШ первого уровня обмен данными выполняется за один такт, в противном случае запускается цикл обращения в КЭШ второго уровня L2.

### 3.4. Сопроцессор

Дополнительные тракты обработки информации, а также аппаратные средства для динамического преобразования адреса реализованы в виде сопроцессора **COP**, состоящего из двух блоков:

блок сжатия, предназначенный для упаковки/распаковки данных и преобразования символьной информации;

блок транслятора, реализующий преобразование виртуальных адресов в реальные с использованием таблиц, размещенных в памяти.

#### 3.4.1. Блок сжатия

Упаковка/распаковка информации в блоке сжатия осуществляется по команде CMPSC, задающей исходную область памяти, с которой выполняется операция, область размещения результата, адрес словаря, используемого в операции, и дополнительные управляющие биты, уточняющие выполняемую операцию. Для упаковки используется метод Зива-Лемпеля LZ2 (Lempel-Ziv 2) со статическими словарями.

Преобразование символьной информации, выполняемое блоком сжатия, определяется командами TR, TRT, TRE и др. и заключается в побайтном преобразовании исходного операнда в операнд результата с использованием таблицы преобразования, индексируемой байтами исходного операнда.

Входные данные для блока поступают в сопроцессор с выходной шины блока обработки E, а результат выдается в блок R с последующей записью в память. В блоке сжатия предусмотрены входной и выходной буферы по 256 байт. Кроме этого блок имеет интерфейс с памятью через КЭШ команд для чтения из памяти словарей и таблиц, необходимых для выполнения преобразований. При упаковке/распаковке словарь может достигать объема 64 КВ и считывается построчно. Таблица для команд символьных преобразования имеет объем 256 байт и полностью считывается в имеющийся в блоке дополнительный буфер.

#### 3.4.1. Блок транслятора

Блок транслятора реализует функции динамического преобразования виртуальных адресов в реальные с использованием трех региональных, сегментной и страничной таблиц. Каждое преобразование запускается в случае промаха при поиске в таблице TLB в блоке BCE и может потребовать до пяти обращений в память. Вместе с виртуальным адресом из блока BCE передается тип адресного пространства, код ASCE и другие параметры, необходимые для формирования базового адреса первой из используемых таблиц преобразования. Выборка строк таблиц из памяти выполняется через КЭШ команд с использованием общего с блоком сжатия интерфейса. Блок транслятора позволяет ускорить преобразования адресов, что особенно важно в случае использования режимов виртуальных ЭВМ, когда одновременно функционируют host-система и несколько guest-систем. В этом случае реальный адрес, полученный в системе более низкого уровня (guest-системе), рассматривается как виртуальный в системе более высокого уровня (host-системе или guest-системе), что может потребовать большого числа обращений в память для каждого преобразования. Например, при двухуровневом преобразовании при каждом обращении guest-системы в одну из пяти таблиц может потребоваться до пяти обращений в таблицы host-системы, что приводит к необходимости выполнять до 25 обращений в память при полном преобразовании.

### 3.4. Блок восстановления

Блок восстановления **R** входит в состав двухканального процессора для реализации функций контроля правильности функционирования путем сравнения информации, формируемой процессорными каналами, а также для накопления информации о состояниях процессора в точках контроля с целью восстановления работы процессора в случае сбоев. Для этого в блоке предусмотрена регистровая память, в которую заносятся управляющие коды различных режимов, коды прерываний, адреса и операнды, используемые командах, содержимое всех программно доступных регистров (GR, AR, FPR, CR). Перед записью информация от двух каналов сравнивается, и при обнаружении ошибок выполняется следующая последовательность мер по восстановлению:

* завершаются все записи в КЭШ L2, инициированные исполненными командами;
* в блоках I, E, BCE выполняется сброс;
* буферы TLB, ALB, директория и массив КЭШ очищаются;
* запоминается прерывание по восстановлению, которое будет исполнено после завершения процесса восстановления;
* все записи из регистровой памяти блока R проверяются на правильность по контрольным кодам и восстанавливаются во всех блоках, откуда они поступили, при наличии неустраняемых ошибок процессор переводится в состояние СБОЙ (подробнее о состояниях процессора описано в главе 8);
* осуществляется запуск выполнения команды в соответствии с восстановленным состоянием, если в ближайшее после запуска время сбой повторяется, процессор переводится в состояние СБОЙ, иначе продолжается нормальная работа процессора.

Поскольку блоки восстановления R и управления BCE не дублируются, их контроль осуществляется путем использования избыточного кодирования для обнаружения и коррекции ошибок.

## Краткие выводы по главе «Микроархитектура процессоров zSeries»

Оглянемся назад? 

1. Проблема когерентности кэш-памяти в архитектуре zSeries является следствием кэширования разделяемых данных
2. Способы размещения строк в кэш-памяти:

* **Кэш-память с прямым отображением –** каждый блок основной памяти имеет одно фиксированное место
* **Полностью ассоциативная кэш-память –**блок основной может располагаться в любом месте кэш-памяти
* **Частично ассоциативная кэш-память –**блок основной памяти может располагаться на ограниченном множестве мест

1. **Когерентная кэш-память –** согласованное состояние памяти, когда любое считывание элемента разделяемых данных возвращает последнее по времени записанное в этот элемент значение
2. Классы протоколов обеспечения когерентности кэш-памяти:
   * + 1. **Протоколы на основе справочника –** информация о блоке физической памяти хранится в одном месте
       2. **Протоколы наблюдения –** каждый кэш содержит данных о блоке физической памяти
3. **Алгоритм MESI** предназначен для реализации протокола когерентности кэш-памяти
4. Состав дублированного процессора zSeries:

* **Блок команд I** для выборки и дешифрации команд
* **Блок операций Е** выполняет операции над операндами, записывает результаты в регистры или в память
* **Блок управления BCE** объединяет кэш команд и данных и их контроллеры
* **Блок восстановления -** для реализации функций контроля с целью восстановления работы процессора в случае сбоев
* **Сопроцессор,** в составе которого
  + **Блок сжатия -** для упаковки-распаковки данных и преобразования символьной информации
  + **Блок транслятора -** для преобразования виртуальных адресов в реальные

## Вопросы для контроля усвоения материала

1. Дайте определение когерентности кэш-памяти.
2. Из каких блоков состоят дублирующие каналы в процессоре zSeries?
3. Какие протоколы использует алгоритм MESI?
4. Для чего служит сериализация?
5. Сколько блоков входит в состав сопроцессора?
6. Назовите основные протоколы обеспечения когерентности кэш-памяти.
7. Что такое когерентность кэш-памяти?

**Содержание главы**

[Глава 4. Логическое разделение ресурсов 2](#_Toc207596148)

[1. Уровни конфигурирования системы 2](#_Toc207596149)

[2. Логический раздел LPAR 6](#_Toc207596150)

[3. Параметры логического раздела LPAR 7](#_Toc207596151)

[4. Процессоры логического раздела LPAR 8](#_Toc207596152)

[5. Распределение памяти в логических разделах LPAR 9](#_Toc207596153)

[6. Взаимодействие логического раздела с подсистемой ввода-вывода 11](#_Toc207596154)

[6.1. Принципы адресации периферийных устройств 11](#_Toc207596155)

[6.1. Способы выделения каналов ввода-вывода для LPAR 13](#_Toc207596156)

[7. Кластеры логических разделов (LPAR-claster) 13](#_Toc207596157)

[7.1. Кластер совмещенных дисков прямого доступа DASD 14](#_Toc207596158)

[7.2. Кольцо канал-канал (CTC/GRS rings) 15](#_Toc207596159)

[7.3. Параллельный системный комплекс Parallel SysPlex 17](#_Toc207596160)

[8. Использование логических разделов в мейнфрейм в университете 20](#_Toc207596161)

[9. Система распределения ресурсов Intelligent Resource Director (IRD) 23](#_Toc207596162)

[6. Краткие выводы по главе «LPAR» 24](#_Toc207596163)

[7. Термины и определения 25](#_Toc207596164)

[8. Вопросы для контроля усвоения материала 25](#_Toc207596165)

[9. Литература 25](#_Toc207596166)

# Глава 5. Логическое разделение ресурсов

Что нас ждет? BD04972_

После прочтения этой главы Вы сможете:

1. перечислить возможные уровни конфигурирования сервера zSeries;
2. дать определение логическому разделу LPAR;
3. перечислить параметры логического раздела;
4. сказать, какие процессоры могут быть выделены в логический раздел;
5. сказать, какие ограничения накладываются при распределении памяти в логических разделах;
6. привести основные принципы адресации периферийных устройств в логическом разделе;
7. перечислить способы выделения каналов для логического раздела;
8. привести практический пример использования логических разделов.

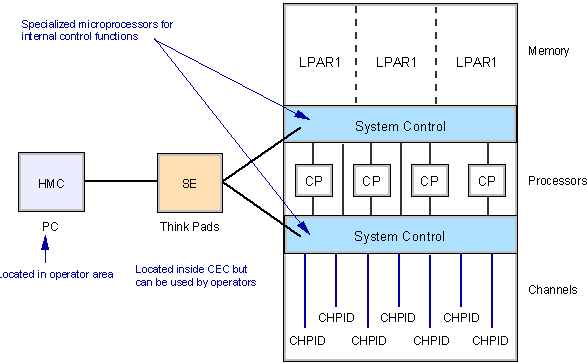
## Уровни конфигурирования системы

Сервер zSeries обеспечивает три уровня конфигурирования и может функционировать с использованием одного, двух или трех уровней управления: как базовая система, как логический раздел (logical partition - LPAR) и как виртуальная система.

Принципы управления первого уровня, когда сервер конфигурирован как базовая система, нам знакомы по предыдущим лекциям. Повторим, базовая система состоит из набора центральных процессоров, памяти, подсистемы ввода-вывода, канальных путей и внешнего таймера. Все ресурсы в базовой системе управляются централизованно единовременно одним из процессоров, принявшим на себя «командование» - выдавшим команду SIGP (Signal Processor).

В этой главе мы рассмотрим следующий уровень управления – логический раздел. Что имеется в виду, когда говорится, что машина поделена на логические разделы? Это значит, что в рамках одной физической машины созданы такие образования – логические разделы, которые позволяют «считать» одну машину набором нескольких вычислительных, причем независимых, систем – логических разделов.

Третий уровень управления – виртуальные машины. На этом уровне управления непосредственный доступ к аппаратным ресурсам полностью отсутствует.



**Рисунок 1 –** схема управления системой и логическое разделение ресурсов

На рисунке 1 показан пример организации внутренней системы управления современной мейнфрейм, имеющей второй уровень управления – логические разделы LPAR. Память системы поделена между тремя логическими разделами. Голубым цветом выделены два слоя управления системой System Control, каждый представляет собой набор специализированных микропроцессоров, разработанных для выполнения функций внутреннего управления системой. Эти микропроцессоры, выполняющие функции внутренних контроллеров, используют более простую организацию и набор команд, чем процессоры zSeries. Они обычно называются контроллеры (controllers), чтобы не спутать с процессорами zSeries.

Именно система управления разделяет начальную, базовую систему на несколько логических разделов (*logical partitions -* LPAR). Каждый логический раздел LPAR содержит ресурсы (процессоры, память, устройства ввода-вывода) и работает как независимая система. Как показано на рисунке 1, несколько логических разделов могут существовать на одном аппаратном обеспечении мейнфрейма.

Долгие годы предельное число логических разделов LPAR на мейнфрейме было 15; последние машины имеют ограничение - 60, в перспективе возможно и большее число логических разделов. Практически ограничения зависят от размера памяти, доступности систем ввода-вывода, вычислительные мощности которых обычно ограничивают число логических разделов (LPAR-ов).

Аппаратно-программное обеспечение, которое обеспечивает логическое разделение ресурсов, называется системный менеджер процессорных ресурсов PR/SM (Processor Resource/System Manager). Одной их функцией системного менеджера процессорных ресурсов PR/SM является создание и запуск логических разделов LPAR. При задании нового логического раздела с помощью этого средства системные администраторы определяют память, процессоры и каналы ввода-вывода (CHPIDs), которые будут постоянно закреплены за этим разделом либо они будут общие для всех логических разделов. Это действие выполняется при профилировании системы с помощью элемента поддержки SE (Support Element) в центральном электронном комплексе CEC (Central Electronic Complex) либо удаленно через консоль управления HMC (Hardware Management Console).

Вносимые изменения записываются в файл, отвечающий за конфигурацию системы (Logical Partition Image Profile), и в набор данных об организации каналов ввода-вывода IOCDS (Input Output Control Data Set), после записи изменений обязательно должен быть произведен сброс по питанию (POR).

Управляющий слой, отвечающий за операции ввода-вывода, использует специальный файл, в котором содержится конфигурационный набор данных параметров оборудования (IOCDS – Input\Output Configuration DataSet). В этом файле физические адреса ввода-вывода переводятся в номера устройств, которые используются операционной системой машины z/OS.

Номера устройств устанавливаются системным программистом при создании файла определения параметров конфигураций ввода-вывода IODF (Input\Output Definition File) и IOCDS и с учетом их приоритетов (а не случайным образом!). В современных машинах они содержат три или четыре шестнадцатеричных цифры.

Все аппаратные ресурсы машины могут быть поделены между этими логическими разделами. Каждый центральный процессор, каждый канальный путь, может быть закреплен за определенным логическим разделом либо разделен в заранее заданном соотношении между несколькими логическими разделами.

Рассмотрим более подробно, что собой представляет логический раздел LPAR.

## Логический раздел LPAR

Можно дать такое определение логическому разделу LPAR: это логический сервер, в котором обеспечивается изолированное от других LPAR исполнение собственной операционной системы (z/OS, OS/390, z/VM, Linux, VSE/ESA) или Coupling Facility Control Code (CFCC).

LPAR всегда должен быть соотнесен с одной из логических канальных подсистем LCSS, допускающих подключение до 60-ти LPAR .

Логический раздел может быть активирован или деактивирован в любой момент "на ходу", хотя для создания нового или удаления ранее созданного раздела необходима остановка системы с использованием процедуры сброса по питанию POR.

Логический раздел LPAR, практически является эквивалентом отдельной мейнфрейм. Каждый LPAR имеет свою собственную операционную систему. Это может быть любая операционная система мейнфрейм, не обязательно устанавливать операционную систему zOS на каждой LPAR. Во время планирования установки можно выбрать общие устройства ввода-вывода для нескольких логических разделов, но это будет локальным решением.

Системный администратор может определить один или более центральных процессоров для исключительного использования в LPAR. Либо, администратор может разрешить всем процессорам быть использованными на некоторых или даже на всех LPARs. В этом случае функции системного управления (известные как микрокод или firmware) предоставляют Диспетчеру право распределять процессоры среди выбранных логических разделов LPAR.

Администратор может также установить весовые коэффициенты (weightings) для разных логических разделов; например, определив, что первый логический раздел (LPAR1) должна получать вдвое больше процессорного времени, чем второй логический раздел (LPAR2).

Операционная система в каждой LPAR является отдельно загружаемой (Initial Program Loaded), имеет свою собственную копию, может иметь собственную консоль оператора (если необходимо), и т.д. Если система в одном логическом разделе (LPAR) сломалась, это никак не повлияет на другие логические разделы.

Администратор может также определить максимальное число одновременно работающих в каждом логическом разделе LPAR процессоров.

## Параметры логического раздела LPAR

Следующие параметры логического раздела LPAR задаются в процессе его открытия:

* количество выделяемых для раздела ресурсов сервера: процессоров, памяти и каналов ввода-вывода;
* весовой коэффициент, присваиваемый разделу для его соотношения с другими разделами в составе одного сервера при использовании общих разделяемых ресурсов;
* ограничение на использование логическим разделом процессоров в большем количестве, чем допускается весовым коэффициентом (LPAR capping) и
* другие параметры

Параметры и информация о выделенных для раздела ресурсах формируются через консоль управления (HMC) и сохраняются в дисковой памяти элемента поддержки SE в виде профиля раздела (Logical Partition Image Profile).

## Процессоры логического раздела LPAR

В логический раздел могут быть выделены процессоры разных типов: центральные процессоры (CPU), процессоры межсистемного взаимодействия (ICF), процессор поддержки операционной системы Linux (IFL) или процессор для исполнения Java-приложений (zAAP).

Каждый из процессоров может быть:

1. постоянно закреплен за одним разделом либо
2. являться общим ресурсом для всех логических разделов LPAR, что является более эффективным вариантом конфигурации.

## Распределение памяти в логических разделах LPAR

Распределение основной и расширенной памяти между разделами осуществляется в процессе создания логических разделов LPAR и предусматривает выделение каждому из разделов части адресного пространства памяти.

Максимальный объем памяти, выделяемой для одного раздела, ограничен объемом используемой в сервере памяти (для z/Architecture - 256 GB, в режиме ESA/390 - 2 GB) без учета области системной памяти HSA (Hardware Save Area).

Использование общих фрагментов основной или расширенной памяти разными LPAR не допускается.

Объем памяти, выделяемой для раздела LPAR, измеряется блоками, размер которых зависит от объема инсталлированной в сервере памяти.

**Таблица 1** – Соотношение объема инсталлированной памяти и размера LPAR

|  |  |
| --- | --- |
| **Объем инсталлированной памяти (GB)** | **Размер блока**  **для LPAR (MB)** |
| **5 ÷ 8** | **16** |
| **8 ÷ 16** | **32** |
| **16 ÷ 32** | **64** |
| **32 ÷ 64** | **128** |

Абсолютная память логического раздела должна быть непрерывной. Активизация и деактивизация логических разделов ведет к тому, что появляется фрагментация памяти.

**Пример.** Посмотрим, как мы сможем задать всю неиспользованную память одному логическому разделу. Предположим, наша машина имеет три логических раздела LPAR1, LPAR2 и LPAR3, как показано на рисунке 2.

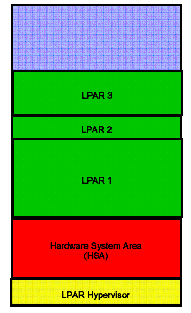
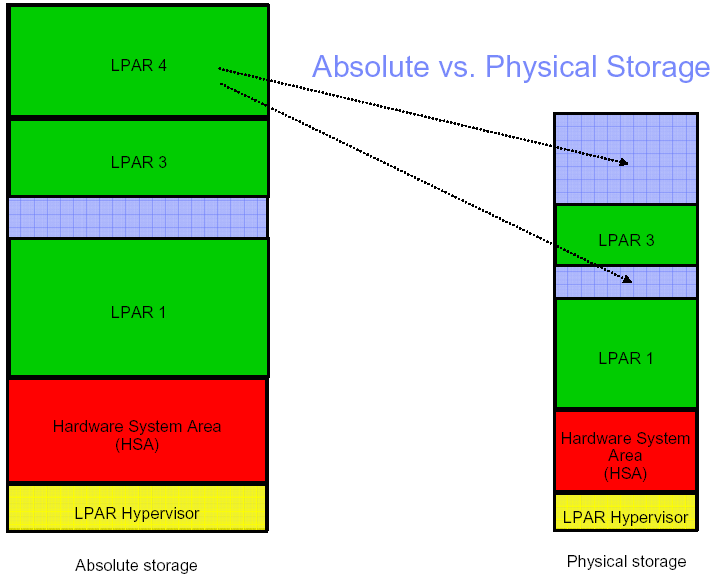


Рисунок 2 – распределение абсолютной памяти между логическими разделами

Логический раздел LPAR2 был деактивирован, в результате деактивации освободилась область абсолютной памяти, которую занимал этот логический раздел. Вновь созданный логический раздел LPAR4, размещенный в абсолютной памяти следующим за логическим разделом LPAR3, на самом деле использует также тот фрагмент физической памяти, который был занят деактивированным третьим логическим разделом, как это показано на рисунке 3.

Эта задача решается следующим образом. Абсолютная память отображается на физическую память так, чтобы размер абсолютной памяти был больше, чем размер физической памяти в 2 раза. Отображение позволяет размещать порции логического раздела (LPAR4), выглядящие логически смежными объектами в абсолютной памяти, в несмежных областях физической памяти.

Если осуществлять предпланирование, возможно расширить размер памяти логического раздела «на лету».



**Рисунок 3** – Отображение абсолютной памяти логического раздела на физическую память

## Взаимодействие логического раздела с подсистемой ввода-вывода

### 6.1. Принципы адресации периферийных устройств

Самые современные машины zSeries на момент написания книги, имеют два слоя преобразования адресов I/O между реальными элементами подсистемы ввода-вывода и программным обеспечением операционной системы. Второй слой был добавлен, чтобы сделать переход к новым машинам легче.

Слой управления вводом-выводом (I/O control layer) использует управляющий файл (control file), известный как конфигурационный набор данных IOCDS (I/O Configuration Data Set). Этот файл переводит физические адреса, состоящие из номеров идентификатора канала (CHPID numbers), номеров портов переключателей (switch port numbers), адреса контрольного блока и адресов блоков, в номера периферийных устройств. Именно эти номера периферийных устройств используются операционной системой для доступа к устройствам.

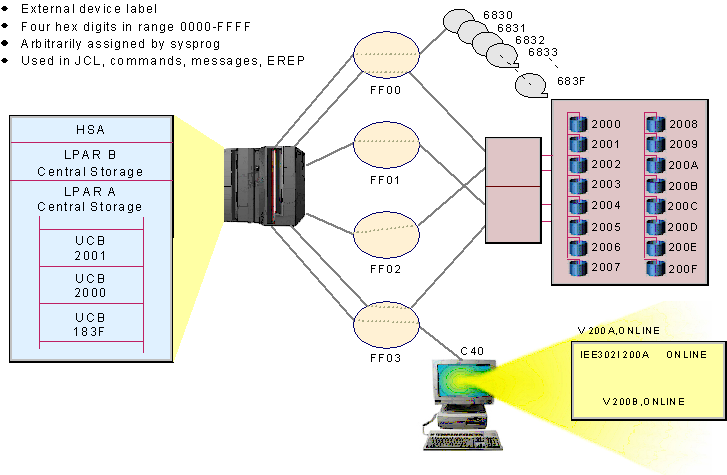


Рисунок 4 – адресация периферийных устройств

Они загружаются в системную область памяти (Hardware Save Area HSA) при включении питания и могут быть модифицированы динамически. Номер устройства выглядит как адрес, который использовали для ранних машин серии S/360, отличие в том, что он может иметь три или четыре шестнадцатеричные цифры.

Многие пользователи все еще употребляют номер устройства как адрес (“addresses”), хотя номер устройства – это арбитражные номера между x'0000' и x’FFFF’

### Способы выделения каналов ввода-вывода для LPAR

Существует четыре способа выделения каналов ввода-вывода для логического раздела:

* 1. **Закрепленные каналы** предназначены для использования только в одном логическом разделе;
  2. **Реконфигурируемые каналы** в каждый момент времени используются только одним LPAR;
  3. **Совмещенные каналы** относятся к разделяемым ресурсам внутри одной логической подсистемы ввода-вывода (LCSS) и
  4. **Объединенные каналы,**  когда каналы, входящие в разные логические подсистемы ввода-вывода (LCSS) допускают подключение к логическим разделам (LPAR).

## Кластеры логических разделов (LPAR-claster)

Кластеризация была разработана много лет назад в нескольких формах. Кластеризация производилась несколькими способами:

* + Обычные диски DASD общего доступа;
  + Кольца CTC/GRS;
  + Основной (Basic Sysplex) и Параллельный системный комплекс (Parallel Sysplex).

Для описания одиночной операционной системы используется образ системы, который может быть отдельной физической системой или логическим разделом внутри большой машины.

### Кластер совмещенных дисков прямого доступа DASD

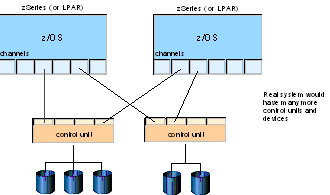


Рисунок 4 – Кластер совмещенных дисков прямого доступа

Оборудование основных совмещенных дисков прямого доступа показано на рисунке 4. Рисунок показывает образ операционных систем z/OS, хотя на месте операционной системы z/OS может быть любая версия более ранней операционной системы. Это могут быть два логических раздела в одной и той же системе, или две отдельные машины, абсолютно нет разницы в самом понятии кластера или его работы.

Возможности такой системы ограничены. Операционные системы автоматически выдают команды RESERVE и RELEASE дисководу перед изменениями (модификациями) основного каталога тома (VTOC) или каталога на диске. Эти каталоги содержат метаданные для дисковода, которые показывают, где на диске размещаются различные наборы данных.

Команда RESERVE от диска к системе ограничивает доступ к заполненному диску, и она работает до тех пор, пока не появится команда RELEASE. Эти две команды хорошо работают в течение ограниченного периода времени (такого, как, например, модификация метаданных). Приложения могут также выдать дисковые команды RESERVE/RELEASE, чтобы защитить свои наборы данных во время выполнения приложения. Это делается не автоматически на этом оборудовании, и это редко делается на практике, потому что это может закрыть доступ к диску на слишком долгий срок для других систем.

Система основных совмещенных дисков прямого доступа обычно используется, когда рабочий штат контролирует, какую работу отдать на выполнение какой системе, и проверяет, что нет никаких конфликтов выполнения, таких как, например, когда обе системы пытаются изменить одни и те же данные одновременно. Несмотря на это ограничение, оборудование основных совмещенных дисков прямого доступа очень полезно для тестирования, восстановления и аккуратной балансировки загрузки (load balancing).

Итак, основные черты такой конфигурации:

* Ограниченные возможности
* Резервирование и снятие резервирования для диска в целом
* Доступ к диску во время обновления ограничен.

### Кольцо канал-канал (CTC/GRS rings)

Рисунок 5 иллюстрирует следующий уровень кластеризации. Он имеет такие же основные совмещенные диски прямого доступа, которые описывались выше, но имеет два соединения канал-канал CTC *(channel-to-channel*) между системами. Иначе такое соединение называется кольцо (*CTC ring)*. Аспект кольца более obvious, когда более чем две системы вовлечены.

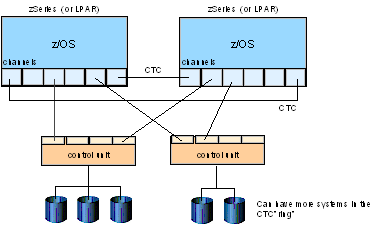


Рисунок 5 – Кластерная конфигурация «Кольцо CTC»

Операционная система z/OS может использовать кольцо CTC, чтобы передавать управляющую информацию среди всех систем кольца. Информация, которая может быть передана таким способом, включает использование и блокирование наборов данных на дисках. Это позволяет системе автоматически предотвращать нежелательное дублирование доступа к наборам данных. Эта блокировка основана на спецификациях языка JCL, предоставляемых для тех видов работ, которые задаются системе, как это будет подробнее описано в главе “Использование языка JCL и подсистемы SDSF” (том 2).

Информация об очереди задач такая, что все системы в кольце может принять задачи от единственной входной очереди. Likewise, все системы могут отправлять задачи на печать в единственную выходную очередь.

Управление безопасностью позволяет принимать uniform решения по безопасности для всех систем.

Дисковые метаданные управляют так, что дисковые команды RESERVE и RELEASE уже не являются необходимыми.

Конфигурации с совмещенными дисками прямого доступа DASD, соединения кольцо CTC, и совмещенные очереди заданий известны как свободно связанные системы (*loosely coupled systems)*.

Основные черты такой конфигурации:

* Глобальный общий доступ к ресурсам (GRS) используется для передачи информации по кольцу CTC
* Request ENQueue on a dataset, update, the DEQueue
* Свободно связанные системы

### 7.3. Параллельный системный комплекс Parallel SysPlex

Самая современная конфигурация кластера называется параллельный системный комплекс Parallel Sysplex. Для его реализации используется специальный компонент, называемый устройство сопряжения Coupling Facilities (CFs). Устройство сопряжения CF – это процессор серии zSeries, с памятью и специальными каналами, встроенный в операционную систему. Он не имеет периферийных устройств, никаких других, кроме специальных, каналов, и очень небольшую операционную систему.

Специальные каналы имеют название межсистемных каналов InterSystem Channels (ISCs), они интегрированы в шины кластера Integrated Cluster Buses (ICBs) для обмена информацией с центральными электронными комплексами (CEC). Отметим, что традиционные каналы большой машины, такие как последовательные ESCON и параллельные FICON каналы ввода-вывода не используются устройством сопряжения.

Для работы оператора используется консоль, интегрированная с элементом поддержки (SE). Управляющая программа, которая общается с центральными электронными комплексами через межсистемные каналы и шину кластера, называется управляющий код устройства сопряжения CFCC (Coupling Facility Control Code).

Код CFCC – это аппаратно-программное обеспечение (firmware), которое поставляется как часть системы. Код CFCC может также работать на виртуальной машине в операционной системе zVM.

Функции устройства сопряжения CF обширны, это аналог быстрого scratch pad. Он используется для достижения трех целей:

* Для блокирования информации, которая является общей для всех подключенных в кластер систем
* кэширование информации (такой как базы данных) общей для всех подключенных в кластер систем
* просмотр информации, которая является общей для всех подключенных в кластер систем

Информация в устройстве сопряжения CF размещается в памяти и устройство сопряжения CF обычно имеет память большого размера.

Устройство сопряжения CF может быть отдельной системой или логическим разделом LPAR, используемым как устройства сопряжения CF. Рисунок 6 иллюстрирует небольшой параллельный системный комплекс, содержащий два образа операционной системы z/OS.

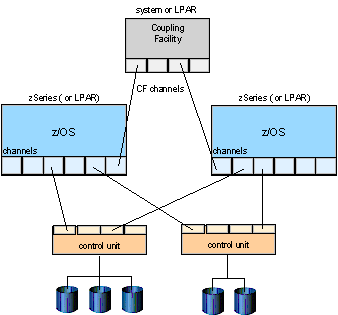


Рисунок 6 – Конфигурация кластера Parallel Sysplex

Повторим, эта конфигурация может быть составлена из трех логических разделов в рамках одной системы, это может быть три отдельные системы, либо эта конфигурация может быть комбинацией отдельных систем и логических разделов.

Во многих случаях система с кластером Parallel Sysplex выглядит как одна большая система. Она имеет один интерфейс оператора, который контролирует все системы. При определенном планировании и исполнении (которые не являются тривиальными), рабочие нагрузки комплекса могут быть совмещенными любой или всеми системами в параллельном системном комплексе Parallel Sysplex, и восстановлении (с помощью другой системы в комплексе Parallel Sysplex), которые могут быть автоматическими для многих рабочих нагрузок.

Основные черты такой конфигурации:

* В этом расширении кольца CTC используется выделенная устройство сопряжения (Coupling Facility) для процедуры storeENQ data для GRS.
* Работает существенно быстрее.
* Устройство сопряжения также может быть использовано для хранения общей информации приложений (такой как таблицы DB2).
* Кластер может быть представлен как единая система.

## Использование логических разделов в мейнфрейме в университете

В рамках национальной программы «Образование» в МГТУ им. Баумана была приобретена большая ЭВМ класса IBM мейнфрейм в комплекте с дисковой подсистемой DS8000 и ленточной библиотекой IBM LTO Ultrium 3584. Основной целью использования мейнфрейм была поставлена организация объединенного хранилища информации административного и научного характера. Кроме того, предполагалось использовать мейнфрейм в учебных целях, выпускники университета должны знать самые современные вычислительные машины не «по-наслышке», одни должны уметь работать на них, другие их обслуживать, третьи – разрабатывать, модернизировать, создавать новые.

Университет имеет свое представительство в Интернете, это официальный веб-портал университета, а также большое количество распределенных физических серверов, на которых установлены веб-сервера и веб-страницы филиалов, факультетов, кафедр, научных школ, центров, многих других организаций. В планах руководства объединить все программное обеспечение и базы данных веб-портала и перевести его на платформу IBM WebSphere Portal и базу данных DB2.

Поставленная машина имеет серию z9 BC, на момент сдачи ее в эксплуатацию в машине были задействованы один центральный процессор POWER5, реализующий архитектуру zArchitecture, специализированные процессоры zAAP для поддержки Java-приложений, сервисный процессор SAP, используемый для управления операциями ввода-вывода, и процессор IFL для поддержки операционной системы LINUX и ее приложений.

Второй центральный процессор POWER5 уже во время работы машины был открыт компанией IBM в качестве гранта для поддержки созданного на кафедре «Компьютерные системы и сети» Академического центра компетенции компании IBM в области больших ЭВМ. Этот процессор предназначен для использования в учебном процессе всего университета.

Для того, чтобы выполнить поставленные задачи, при этом обеспечив надежность и защищенность использования административных данных, совместив в одном физическом корпусе большой ЭВМ задачи по обслуживанию административной информации (электронный университет, базы данных отдела кадров, бухгалтерии, то есть, такой информации, которая должна быть очень хорошо защищена извне) и информации, предоставляемой всем желающим узнать об университете – веб-портал университета, веб-сайты кафедр, библиотеки, музей и так далее, и третья задача – использование машины в учебном процессе – выполнение лабораторных работ студентами, а это значит, что надо заведомо быть готовыми к сбоям и поломкам аппаратуры и программного обеспечения. Эти три на первый взгляд несовместимые для решения противоречивые задачи могут быть прекрасно решены с применением технологии логического разделения ресурсов.

На машине реализована возможность логического деления ресурсов: созданы три логических раздела, назовем их условно как административный, учебный и представительский. Первый центральный процессор совместно используется административным и представительским логическим разделами, второй – только учебным логическим разделом. Все специализированные процессоры являются совместно используемыми.

В дисковой подсистеме DS8000 реализована также возможность логического деления дисков (LPAR) с помощью технологии виртуализации (Virtualization Engine) компании IBM на базе процессора POWER5. Данная технология применения LPAR в системах хранения данных позволила создать две полностью независимых подсистемы расширенной памяти, которые работают под управлением различных операционных систем. Одна из подсистем применяется для поддержания рабочей среды университетского хранилища данных. Другая подсистема предназначена для использования ее в учебном процессе университета, при этом обе подсистемы работают в рамках одного физического корпуса.

Основная память также поделана между логическими разделами, причем, пропорциональное соотношение организовано в пользу представительского LPARа, учитывалась необходимость быстро реагировать на запросы извне, страницы веб-портала должны открываться быстро, информация из хранилища данных должна предоставляться быстро.

## Система распределения ресурсов Intelligent Resource Director (IRD)

Как продолжение развития системы управления логическими разделами появилась система распределения ресурсов Intelligent Resource Director (IRD). Его основными функциями являются управление процессорами логических разделов LPAR (функция называется «LPAR CPU Management»), динамический менеджмент канальных путей DCM (dynamic channel path management) и организация приоритетных очередей в канальной подсистеме IOPQ (Channel Subsystem Priority Queuing).

Установка такой системы в машине уменьшает затраты на ее обслуживание, повышает надежность работы системы, увеличивает характеристики ее производительности.

## Вопросы для контроля усвоения материала главы 5

1. Перечислите компонентный состав базовой системы мейнфрейм.
2. Из каких аппаратных ресурсов состоит логический раздел LPAR?
3. Какие процессоры могут быть выделены в логический раздел?
4. Для чего служит устройство сопряжения CFs?
5. Сколько логических разделов может быть в одной физической системе?
6. Допускается ли использование общих фрагментов основной или расширенной памяти разными LPAR?
7. Что такое кластер?

## Краткие выводы по главе «LPAR»

Оглянемся назад? BD04972_

1. Уровни конфигурирования сервера zSeries:

* **Базовая система** – все аппаратные ресурсы управляются из одного центра,
* **Логический раздел** – содержит выделенные ресурсы и работает как независимая система,
* **Виртуальная машина** – не содержит ресурсов и работает как зависимая система.

1. **Логический раздел**– это логический сервер, в котором выполняется собственная операционная система.
2. **Параметры логического раздела LPAR**:
   1. **Количество выделяемых ресурсов**
   2. **Весовой коэффициент**
   3. **Ограничения на использование ресурсов**
3. Любой процессор конфигурации может быть выделен в логический раздел.
4. Соотношение объема инсталлированной памяти и размера LPAR:

|  |  |
| --- | --- |
| **Объем инсталлированной памяти (GB)** | **Размер блока для LPAR (MB)** |
| **5 ÷ 8** | **16** |
| **8 ÷ 16** | **32** |
| **16 ÷ 32** | **64** |
| **32 ÷ 64** | **128** |

1. **LPAR и канальная подсистема - адресация (**два слоя преобразования адресов)**:**
   1. **Конфигурационный набор данных IOCDS** – файл, управляющий вводом-выводом
   2. **Номер периферийного устройства** – это арбитражные номера между x'0000' и x’FFFF’.
2. **LPAR и канальная подсистема - способы выделения каналов:**

* **Закрепленные** - использование только в одном логическом разделе
* **Реконфигурируемые** - одновременно используются только одним LPAR;
* **Совмещенные -** относятся к разделяемым ресурсам внутри одной логической подсистемы ввода-вывода
* **Объединенные -**  каналы, входящие в разные логические подсистемы ввода-вывода допускают подключение к разным LPAR

1. Варианты построения кластеров логических разделов:
   * Кластер обычных дисков DASD общего доступа;
   * Кольца CTC/GRS;
   * Основной (Basic Sysplex) и Параллельный системный комплекс (Parallel Sysplex).
2. Пример использования логических разделов в университетской большой ЭВМ:

* административный логический раздел
* учебный логический раздел
* представительский логический раздел

# Глава 6. Основы мультипроцессирования

Что нас ждет? BD04972_

После прочтения этой главы Вы сможете:

1. назвать четыре состояния процессора;
2. перечислить средства управления работой процессора;
3. перечислить варианты сброса процессора;
4. дать определение понятия конфигурация;
5. назвать три основных принципа мультипроцессирования;
6. объяснить механизм прерываний;
7. перечислить шесть классов прерываний.
8. дать определение плавающему прерыванию;
9. перечислить средства временной синхронизации.

## Управление работой процессора

### 1.1. Состояния процессора

Процессор в большой ЭВМ класса IBM мейнфрейм может находиться в одном из четырех взаимоисключающих состояний:

* **СТОП,** когда все команды и прерывания, кроме прерывания для перезапуска, не выполняются;
* **РАБОТА,** когда команды и прерывания исполняются в соответствии с управляющими кодами в слове состояния программы PSW, в управляющих регистрах и в соответствии с режимом, заданным оператором;
* **ЗАГРУЗКА,** когда состояние устанавливается в процессе первоначальной загрузки в соответствии со стандартной архитектурой этой машины ESA/390;
* **СБОЙ**, в это состояние процессор переходит в результате машинных сбоев, выявленных в процессе функционирования процессора.

Переходы из одного состояния в другое могут быть вызваны либо определенными событиями, либо по команде SIGNAL PROCESSOR (SIGP) с адресом нужного процессора, либо по командам с пульта оператора. Условия переходов из одного состояния в другое приведены в таблице 1.

Таблица 1 - Состояния процессора и причины их возникновения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Следующее состояние** | **ЗАГРУЗКА** | **СТОП** | **РАБОТА** | **СБОЙ** |
| **Исходное состояние** |  |  |  |  |
| **ЗАГРУЗКА**  **(ETR)** |  | Пошаговый режим | Непрерывный режим | Сбой оборудования |
| **СТОП** | Переключатели загрузки |  | Переключатель перезапуска, приказ запуска в команде SYGNAL PROCESSOR |
| **РАБОТА**  **(ETR)** | Переключатели сброса процессора, приказ "Стоп" в команде SYGNAL PROCESSOR, завершение шага в пошаговом режиме работы, останов по совпадению адресов |  |
| **СБОЙ** | - | Сброс | - |  |

Как видно из таблицы, из состояния ЗАГРУЗКА (ETR) возможен переход в любое из оставшихся 3-х состояний: СТОП в случае пошагового режима выполнения программы, РАБОТА, если программа выполняется непрерывно и СБОЙ в случае неправильной работы оборудования. Из состояния СТОП возможен переход либо в состояние ЗАГРУЗКА, либо в состояние РАБОТА, если оператор нажал переключатель перезапуска или если был получен приказ запуска в команде SIGP. Из состояния РАБОТА возможен переход только в состояние СТОП, переход вызывают переключатели сброса процессора, приказ СТОП в команде SIGP, завершение шага в пошаговом режиме работы и останов в случае совпадения адресов. Из состояния СБОЙ возможен переход только в состояние СТОП по командам сброса внешнего управления процессором, причем существует пять вариантов осуществления сброса, что будет рассмотрено в следующем разделе.

### 1.2. Средства управления работой процессора

Основную роль в управлении работой процессора играют управляющие коды в слове состояния процессора PSW и управляющие регистры процессора.

При смене состояния процессора автоматически осуществляется смена значений всего слова PSW.

Примечание: Отметим, что смена состояния процессора – это не единственная причина смены значений PSW. Смена значений всего слова PSW или отдельных его полей может выполняться автоматически при смене состояний других процессов, таких как прерывания, а также программно - при выполнении некоторых команд (например, LOAD PSW, BRANCH и SET PROGRAM MASK).

В префиксной области основной памяти для каждого процессора имеется место, где хранятся слова состояния процессора PSW – текущее и старое. Это позволяет процессору вернуться в предыдущее состояние в целом ряде случаев. Текущее слово состояния процессора PSW хранится в регистре PSW процессора и содержит информацию, необходимую для управления процессом исполнения команд программы.

Полями слова состояния процессора PSW, определяющими последовательность исполнения команд, являются поле адреса команды (Instruction Address) и поле кода условия СС (формат слова состояния процессора PSW описан подробно в главе 3 и приведен на рисунке 3.3.). Поле адреса команды реализует функции программного счетчика, то есть формирует адрес очередной исполняемой команды. В зависимости от режима адресации, определяемого битами EA, BA в PSW, разрядность адреса может быть равна 64, 32 или 24. Двухразрядный код условия СС формируется по результатам исполнения большинства команд и используется в качестве признака перехода в командах ветвления. Другие поля слова PSW используются в различных архитектурных механизмах, реализуемых процессором, таких как прерывания, динамическое преобразование адреса, защита памяти и так далее.

### 1.3. Пять вариантов сброса процессора

Предусмотрено пять инициируемых извне функций управления процессором:

* сброс процессора;
* начальный сброс процессора;
* сброс подсистемы;
* сброс с очисткой;
* сброс по питанию.

#### 1.3.1. Сброс процессора

Функция внешнего управления процессором, осуществляющая сброс процессора переводит процессор в состояние СТОП во всех случаях, кроме случая, когда сразу после сброса следует процедура начальной загрузки ILP (Initial Program Loading), тогда процессор переходит в состояние ЗАГРУЗКА.

Функция «Сброс процессора» вынуждает процессор выполнить следующие действия:

* выполнить очистку указателей сбоев оборудования и устранить неопределенность состояния процессора, возникающую в результате таких сбоев, в том числе путем сохранения состояния для последующего анализа и восстановления;
* прекратить выполнение текущей команды или других действий, например, прерываний;
* сбросить все условия прерываний, кроме внешних плавающих прерываний;
* сбросить все предварительно выбранные команды и операнды, а также подготовленные для записи в память результаты;
* очистить строки буфера быстрого доступа ALB (Access Lookaside Buffer) и буфера быстрой переадресации TLB ((Translation-Lookaside Buffer).

Один из вариантов внешнего управления этого уровня - сброс, вызванный переключателем "загрузка - нормальная" любого из процессоров *конфигурации*. В этом случае после выполнения процессором перечисленных действий устанавливается режим архитектуры ESA/390, текущее слово PSW трансформируется в формат ESA/390 и сохраняется в префиксной области основной памяти для последующего восстановления режима z/Architecture по команде SIGNAL PROCESSOR.

Остановимся на объяснении понятия «конфигурация», впервые встретившемся читателю в предыдущем абзаце. В компании IBM принято обозначать этим понятием конкретную компоновку компьютерной системы с учетом сущности, количества и главных свойств ее функциональных модулей [1].

#### 1.3.2. Начальный сброс процессора

Начальный сброс процессора включает в себя операции сброса процессора с последующими дополнительными операциями **очистки и инициализации**.

Процедура **очистки** заключается в том, что устанавливаются в нуль содержимое:

* текущего PSW,
* старого PSW,
* регистров префикса,
* таймера CP,
* компаратора времени,
* программируемого регистра TOD и
* регистра управления операциями с плавающей точкой.

Процедура **инициализации** заключается в том, что в управляющих регистрах устанавливаются начальные состояния, соответствующие режиму z/Architecture.

#### Сброс подсистемы ввода-вывода

Сброс подсистемы ввода-вывода не связан напрямую с внешним управлением процессором, однако, он осуществляется совместно и предназначен для тех элементов конфигурации, которые не являются процессорами, путем выполнения следующих действий:

* в канальной подсистеме выполняется сброс системы ввода-вывода, включая сброс прерываний ввода-вывода и передачу системного сброса в устройства ввода-вывода;
* сбрасываются плавающие прерывания в конфигурации.

#### Сброс с очисткой

Сброс с очисткой объединяет операции начального сброса процессора и сброса подсистемы, плюс следующие операции инициализации:

* во всех процессорах конфигурации устанавливается режим архитектуры ESA/390;
* регистры общего назначения, регистры с плавающей точкой, регистры доступа устанавливаются в 0;
* содержимое основной памяти в конфигурации и соответствующие ключи памяти обнуляются;
* блокировки, применяемые в любом процессоре конфигурации при исполнении команды PERFORM LOCKED OPERATION, отменяются**.**



**Рисунок 1**. Пять функций внешнего управления

#### 1.3.5. Сброс по питанию

**Сброс по питанию** выполняется при включении питания. Он включает в себя сброс регистров общего назначения (GR), регистров с фиксированной точкой (FPR) и регистров доступа (AR) в нулевое состояние и установку режима архитектуры ESA/390.

Сброс по питанию может сопровождаться также сбросом часов реального времени TOD часов.

## Основные принципы мультипроцессирования

Мы рассмотрели состояния процессора и функции внешнего управления процессором, не принимая пока во внимание то, что мейнфреймы имеют важную особенность – наличие большого числа самых разнообразных процессоров. Их совместная работа внутри одной машины сопряжена с целым рядом проблем, начиная с того, что необходимо синхронизировать их работу, обеспечить целостность данных, к которым процессоры имеют совместный доступ, организовать межпроцессорный обмен информацией, и так далее.

Управление большим набором процессоров называется мультипроцессированием.

В современных больших машинах класса IBM мейнфрейм на момент написания книги может быть 16 процессоров как в модели мейнфрейма 216 серии z900), 32 процессора как в модели мейнфрейма D32 серии z990 и даже 54 процессора, как в модели мейнфрейма S54 серии z9 EC. Большое количество процессоров используется для распараллеливания вычислительных процессов с разделением данных и ресурсов и для обеспечения высокой готовности системы.

Мультипроцессирование включает в себя механизмы и процессы управления большим набором процессоров путем обеспечения их взаимодействия через общую память и средства межпроцессорных обменов.

Основу мультипроцессирования составляют:

* общая разделяемая память;
* межпроцессорное взаимодействие;
* синхронизация часов.

### 2.1. Взаимодействие процессоров через общую память

Разделяемая память допускает обращение нескольких процессоров в одни и те же ячейки, при этом процессорам разрешается обращение в страницу памяти емкостью 4 KB, содержащую общую ячейку. Общая ячейка, используемая процессорами и канальной подсистемой, должна определяться одним и тем же абсолютным адресом.

Взаимодействие процессоров через общую разделяемую память организовано так, что у каждого процессора имеется собственная уникальная префиксная область.

При этом процессоры могут использовать общие данные, обновление которых осуществляется только при участии механизма блокировки обращения в память с помощью следующих специальных команд:

* TEST AND SET (antique);
* COMPARE AND SWAP;
* PERFORM LOCKED OPERATION.

Для организации межпроцессорного взаимодействия введен адрес процессора в системе, присваиваемый каждому процессору при инсталляции системы и не изменяемый при ее реконфигурациях. Адрес процессора используется и в команде SIGNAL PROCESSOR для указания процессора, которому предназначен приказ, заданный в команде, а также идентифицирует процессор, сформировавший условия прерывания, его адресом, записанным в код прерывания.

### 2.2. Средства межпроцессорных обменов

Кроме того, что процессоры могут взаимодействовать через общую память, дополнительные средства мультипроцессирования могут быть обеспечены с помощью команды централизованного управления процессорами SIGNAL PROCESSOR (SIGP) и средств внешних прерываний. Рассмотрим эти средства подробнее.

#### 2.2.1. Команда SIGNAL PROCESSOR (SIGP)

Команда Signal Processor является основным средством взаимодействия процессоров путем сигнализации и получения ответа. Право генерировать такую команду единовременно имеет только один процессор, он приобретает при этом статус «главного», обеспечивающего централизованный принцип управления. Любой из процессоров может получить это право. Управляющий процессор с помощью этой команды обращается (адресует) к любому другому процессору. Такая команда адресует один процессор (есть вариант команды, в которой адресуются все процессоры) и передает ему один из приказов. В каждом процессоре предусмотрены средства для передачи, получения и выполнения приказов, а также для формирования ответа процессором, исполняющим команду Signal Processor. При определенных условиях адресуемый процессор формирует код состояния для процессора, исполняющего команду Signal Processor. Каждому приказу, исполняемому в адресуемом процессоре, может соответствовать несколько передаваемых условий.

#### 2.2.2. Механизм обслуживания прерываний

Прерывания процессора, как дополнительное средство межпроцессорного обмена, позволяют обеспечить быструю реакцию процессора при возникновении особых условий в самом процессоре, в подсистеме ввода-вывода, в других процессорах и вне системы. Прерывания допускаются только в режиме РАБОТА, за исключением прерывания для перезапуска, которое может быть выполнено в режимах РАБОТА или СТОП. Инициируются прерывания запросами от устройств, в которых возникают условия прерываний.

Все прерывания разбиты на шесть классов:

**Прерывание по вызову супервизора** возникает сразу после исполнения в процессоре команды SUPERVISOR CALL, основным назначением которой является переключение в режим СУПЕРВИЗОР. Данное прерывание не может быть запрещено маскированием (выполняется сразу после исполнения команды SUPERVISOR CALL).

**Программные прерывания** возникают при возникновении нарушений или особых ситуаций в процессе исполнения команд программы, например, при недопустимом коде операции, при исполнении привилегированной команды в режиме РАБОТА, при попытках несанкционированных обращений в память и др.

**Прерывания от схем контроля** формируются при неисправностях и сбоях аппаратных средств. Различают неотложные прерывания, требующие немедленной реакции процессора, и подавляемые прерывания, допускающие отложенную обработку.

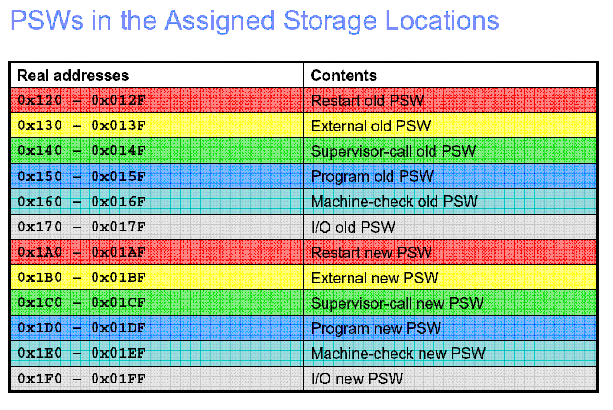
**Прерывания ввода-вывода** формируются в подсистеме ввода-вывода и предназначены для передачи в процессор информации о возникновении особых ситуаций в периферийном оборудовании и канальной подсистеме.

**Прерывание перезапуска** обеспечивает выполнение специальной программы и инициируется оператором путем нажатия кнопки рестарта на пульте управления или по команде SIGNAL PROCESSOR от другого процессора, адресующей данный процессор. Данное прерывание не может быть запрещено маскированием.

**Внешние прерывания** обеспечивают реакцию процессора на различные сигналы, возникающие внутри системы и вне нее. К ним относятся сигналы от кнопки прерывания с пульта управления (Interrupt key), оповещения о сбое (Malfunction alert), от компаратора времени (Clock comparator), от процессорного таймера (CPU timer) и другие. Именно этот класс прерываний обеспечивает возможность процессорам общаться между собой.

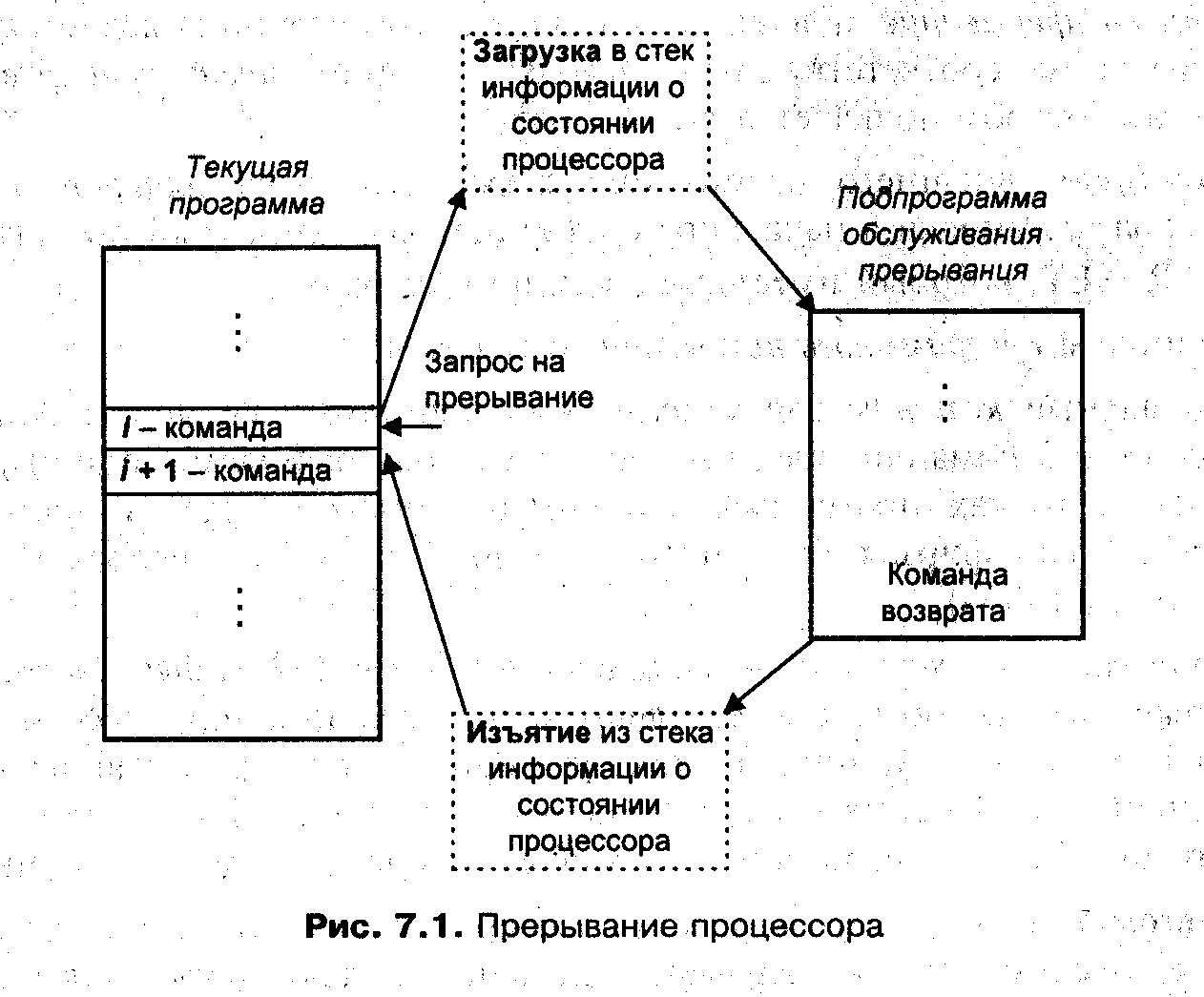
Каждому классу прерываний выделены две фиксированные области памяти, в одной сохраняется старое(текущее) PSW прерываемой программы, а в другой - новое PSW прерывающей программы. Таблица 3 иллюстрирует соответствие между реальным адресом в памяти и словами состояния процессора для каждого класса прерываний.

Таблица 3 – Размещение слов PSW в памяти компьютера



Каждый процессор имеет свои области в памяти за счет механизма префиксации. Старое PSW обычно содержит адрес команды, которая выполнялась бы следующей, если бы не произошло прерывания. Это позволяет продолжить исполнение прерванной программы после отработки процедуры обслуживания прерывания. При программных прерываниях или вызове супервизора в старом PSW сохраняется код длины последней выполненной команды, что позволяет идентифицировать команду, вызвавшую прерывание.

В некоторых случаях прерываний, когда требуется повторное исполнение команды, вызвавшей прерывание, в старом PSW сохраняется адрес этой команды. Возврат к прерванной программе выполняется путем восстановления в процессоре старого PSW (см. рисунок 5).



Обработка *одновременно* поступивших запросов осуществляется путем последовательной записи старых PSW и выборки новых PSW без исполнения команд прерывающих программ до тех пор, пока не будут обработаны все имеющиеся запросы. Далее в обратном порядке в соответствии с сохраненными старыми PSW выполняются прерывающие программы.

##### 2.2.2.1. Код прерываний

Причина прерывания внутри класса уточняется кодом прерывания этого класса, который в процессе прерывания заносится в отдельную область памяти, закрепленную за данным классом. В зависимости от класса длина кода прерывания может быть 16, 32 или 64 разряда.

Код прерывания используется прерывающей программой для определения того, какая процедура обслуживания прерывания должна быть выполнена.

Большинство команд допускают прерывания после полного окончания их исполнения. В процессе исполнения могут быть прерваны команды, называемые прерываемыми (interruptible). К ним относятся такие команды, как MOVE LONG, COMPARE LOGICAL LONG и др. Процесс исполнения этих команд состоит из последовательно исполняемых этапов (unit of operation). Прерывания допускаются после завершения текущего этапа, и прерванная команда считается частично выполненной.

##### 2.2.2.2. Процесс прерывания

Процесс прерывания включает следующие действия:

распознавание класса и причины прерывания;

сохранение текущего PSW как старого PSW распознанного класса;

сохранение информации, идентифицирующей прерывание;

выборка и размещение в процессоре нового PSW распознанного класса;

запуск прерывающей программы в соответствии с новым PSW.

Возврат к прерванной программе выполняется путем восстановления в процессоре старого PSW.

##### 3.2.2.3. Маскирование

Разрешение и запрет прерываний в процессоре реализуются с использованием **маскирования**.

Маски прерываний размещаются в зависимости от класса в текущем PSW, в управляющих регистрах и в регистре управления операциями с плавающей точкой FPC.

**Примеры:**

Прерывания ввода-вывода маскируются 6-ым разрядом в PSW и дополнительно, в регистре управления CR6 имеется по одному разряду для маскирования каждого подкласса.

Внешние прерывания маскируются 7-ым разрядом в PSW и дополнительно, в регистре управления CR0 имеется по одному разряду для маскирования каждого подкласса.

Подавляемые машинные проверки (repressible machine checks) маскируются 13-ым разрядом в PSW и дополнительно, в регистре управления CR0 имеется по одному разряду для маскирования каждого подкласса.

Некоторые программные прерывания маскируются с 20-го по 23-ий разрядами в PSW, такие как переполнение для чисел с фиксированной точкой, переполнение для десятичных чисел, отрицательное переполнение (hexadecimal-floating-point exponent underflow) или изменение знака (hexadecimal-floating-point significance).

Каждый разряд маски разрешает или запрещает соответствующее прерывание. Неразрешенные прерывания либо полностью игнорируются, либо остаются в состоянии ожидания.

Маской могут быть запрещены как все прерывания данного класса, так и отдельные типы прерываний.

Программные прерывания не могут быть маскированы, исключение составляют только случаи переполнение для чисел с фиксированной точкой, переполнение для десятичных чисел, отрицательное переполнение (hexadecimal-floating-point exponent underflow) или изменение знака (hexadecimal-floating-point significance).

##### 2.2.2.4. Плавающие прерывания

Прерывания, которые могут быть обработаны любым из процессоров конфигурации, называются плавающими прерываниями (floating interruption). Запрос на такое прерывание подается в первый из процессоров, в котором это прерывание не замаскировано, после чего сбрасывается для исключения повторных прерываний в других процессорах. К плавающим прерываниям относятся прерывания ввода-вывода, некоторые из внешних прерываний и от схем контроля.

#### 

#### 2.2.2.5. Приоритеты прерываний

Для обслуживания нескольких поступивших одновременно запросов на прерывания в определенном порядке используется механизм приоритетов запросов. Каждому классу прерываний и запросам внутри класса присваиваются приоритеты, в соответствии с которыми и обслуживаются прерывания.

Наивысшим приоритетом обладают неотложные прерывания от схем контроля. При их поступлении все текущие операции в процессоре сбрасываются и выполняется обработка обнаруженных неисправностей или сбоев.

Прерывания обрабатываются в следующем порядке:

* неотложные прерывания от схем контроля;
* прерывание по вызову супервизора;
* программные прерывания;
* прерывания от схем контроля, допускающие отложенную обработку;
* внешние прерывания;
* прерывания ввода-вывода;
* прерывание рестарта.

Внутри класса все прерывания также ранжируются путем присвоения соответствующих приоритетов.

### 2.3. Средства временной синхронизации

Для правильного отсчета времени и взаимной синхронизации работы всех процессоров в архитектуре z/Architecture предусмотрены три архитектурных компонента:

1) **часы** реального времени **TOD** (Time of day) для отсчета реального времени и ведения даты и времени суток; причем, в системе предусматривается использование одних часов TOD для всех процессоров, а в каждом процессоре имеется программируемый регистр часов реального времени TOD, связанный с этими часами;

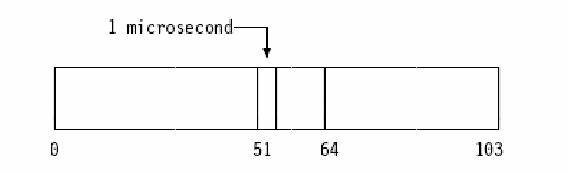
2) **компаратор времени**, имеющийся у каждого процессора и предназначенный для выработки прерывания, когда показания часов TOD превышают установленное программой значение;

3) **процессорный таймер** (CPU Timer), обеспечивающий измерение прошедшего времени и выработку прерывания в случае истечения заданного интервала времени; каждый процессор имеет собственный таймер.

Рассмотрим подробнее каждый из этих компонентов.

#### 2.3.1. Часы реального времени TOD

**Часы TOD** представляют собой 104-разрядный двоичный счетчик, который инкрементируется в типовом варианте каждую микросекунду добавляет единицу в 51-ый разряд (рисунок 4)



**Рисунок 4 –** Формат часов реального времени TOD

В зависимости от модели инкремент часов TOD может осуществляться и в другие разряды, но с частотой, соответствующей периоду переключения 1 мкс в разряде 51. Это позволяет менять разрешающую способность часов TOD с целью ее согласования со временем исполнения команд.

Нулевое показание часов TOD определено на 1 января 1900 года 00:00:00 (час:мин:сек) по стандарту UTC. Переполнение часов произойдет 17-го сентября 2042 года в 23:53. Команда STORE CLOCK (STCK) выдает информацию о первых 64-х битах часов TOD.

Работа часов TOD не зависит от состояния процессоров, кроме того, часы могут иметь отдельный источник питания. Часы могут находиться в одном из следующих состояний: выставлены, не выставлены, остановлены, неисправны или отключены, как это показано в таблице 2.

Таблица 2

|  |  |
| --- | --- |
| **Причина, вызвавшая состояние часов** | **Состояние часов** |
| Ручное управление CR0 (бит 9),  Автом. управление CR14 TOD-clock-sync-control) | Выставлены |
| Выключение питания | Не выставлены |
| Команда SET CLOCK | Остановлены |
| Прерывание процессора | Неисправны |
| Отсутствие питания | Отключены |

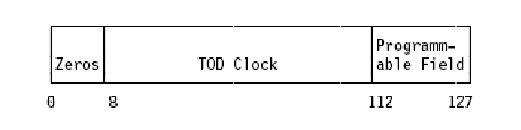
Переходы из одного состояния в другое задаются специальными командами или процедурами. Часы могут быть отключены при отсутствии питания или для технического обслуживания. После включения питания часы обнуляются и переходят в невыставленное состояние. Переход в выставленное состояние осуществляется только из остановленного состояния часов TOD, а останавливаются часы по команде SET CLOCK, загружающей новое значение показаний, если включено ручное управление TOD либо если установлен бит TOD-clock control override в регистре управления CR14.

Переход из остановленного состояния в выставленное выполняется в зависимости от значения бита управления часами в регистре управления CR0 (TOD-clock-sync-control bit) в процессоре, выполнившем команду SET CLOCK, причем переход осуществляется автоматически при сброшенном состоянии бита или с задержкой до момента сброса этого бита. В выставленном или в не выставленном состояниях часы идут, то есть инкрементируются.

Неисправное состояние часов устанавливается средствами контроля, зависящими от модели.

Установка показаний в часах выполняется командой SET CLOCK при наличии разрешающих условий. При этом в счетчик часов в инкрементируемые разряды загружается операнд команды, а остальные биты обнуляются. В многопроцессорных конфигурациях одновременное выполнение команд SET CLOCK в разных процессорах блокируется.

Считывание показаний часов выполняется командами STORE CLOCK, STORE CLOCK EXTENDED путем занесения, соответственно, разрядов 0÷63 или 0÷103 счетчика часов в 8-ми или в 16-ти байтный операнд в памяти. Формат показаний часов для команды STORE CLOCK EXTENDED показан на рисунке 5, где поле Programmable Field загружается из программируемого регистра TOD. Выполнение двух последовательных команд чтения в одном или разных процессорах приводит к разным показаниям часов. Выполнение команд чтения часов в отключенном или неисправном состоянии приводит к записи нулевых показаний.



**Рисунок 5 –** Формат часов TOD

Первые восемь бит пока должны быть нулевыми, они будут использоваться после 17-го сентября 2042 года (и вплоть до 36500 года A.D.) Тогда 59-ый разряд будет равен 1 микросекунде, а 111-ый разряд будет равен 222\*10-24секунды. Поле Programmable field используется для того, чтобы сгенерировать уникальное показание часов, устанавливаемое с помощью команды SET CLOCK Programmable field (SCKPF).

#### Компаратор времени

Компаратор времени предназначен для сравнения показаний часов TOD с загруженной в компаратор установкой. В типовом варианте сравниваются 48 старших разрядов счетчика часов и уставка такой же разрядности.

Загрузка уставки в регистр компаратора выполняется командой SET CLOCK COMPARATOR, а чтение - командой STORE CLOCK COMPARATOR. Компаратор времени имеет такой же формат, как и часы TOD, то есть, 51-ый разряд соответствует 1 микросекунде. Компаратор времени постоянно сравнивается с часами TOD, и если часы TOD достигают показание компаратора времени, срабатывает внешнее прерывание. Эта возможность используется при измерениях в реальном времени. Команды для управления компаратором: установка (или установка?) SET CLOCK COMPARATOR (SCKC), чтение STORE CLOCK COMPARATOR (STCKC).

#### 2.3.2. Процессорный таймер

Процессорный таймер является двоичным счетчиком в формате, соответствующем 64-ем старшим разрядам счетчика часов TOD, исключая старший бит 0, рассматриваемый как знаковый. Таймер декрементируется вычитанием единицы из 51-го разряда каждую микросекунду.

Таймер имеет такой же формат, как и часы TOD (51-ый разряд соответствует 1 микросекунде), но разряд 0 рассматривается как знаковый.

В моделях процессоров с большей или меньшей разрешающей способностью таймера для декремента может быть выбран другой разряд таймера с сохранением периода переключения 1 мкс в 51-ом разряде. Прерывание от таймера вырабатывается при достижении отрицательного значения, то есть, при установке единицы в разряде 0. При загрузке в таймер положительной величины прерывание не срабатывает. В некоторых моделях для более точного измерения фактического времени исполнения программы таймер приостанавливается во время других операций, например, при интенсивных операциях ввода-вывода. Загрузка показаний в таймер выполняется командой SET CPU TIMER (SPT), а чтение - командой STORE CPU TIMER (STPT).

Когда процессорный таймер меньше нуля, генерируется внешнее прерывание. Останавливается таймер тогда, когда останавливается процессор.

#### 2.3.2. Расхождения в отсчетах времени (Timer Stepping)

В реальной системе часы для отсчета реального времени и ведения даты и времени суток TOD и процессорный таймер отсчитывают время с одинаковой скоростью. В виртуальной системе процессорный таймер запускается только тогда, когда к виртуальной системе обращается Диспетчер, поэтому может получиться, что отсчет процессорного таймера виртуальной системы окажется медленнее, чем у часов TOD.

### 3. Запуск мультипроцессорной операционной системы

При запуске мультипроцессорной операционной системы последовательность действий заключается в шести основных шагах:

1. старт процессора 0, остальные процессоры пока остановлены;
2. выдача команды SIGP “SET ARCH” для переключения центрального электронного комплекса (CEC) c архитектуры S/390 на архитектуру zArchtecture;
3. определение префиксной области в основной памяти для других процессоров;
4. инициализация перезапуска новых слов RSW в префиксных областях для других процессоров (см. таблицу 3);
5. выдача команды SIGN «SET PREFIX» установки префикса для всех остальных процессоров (CPUs);
6. выдача команды SIGN «RESTART» перезапуска для всех остальных процессоров (CPUs).

## 4. Краткие выводы по главе 6 «Основы мультипроцессирования»:

Оглянемся назад? 

1. Четыре состояния процессора:

* СТОП – все команды и прерывания не выполняются (кроме RESET),
* РАБОТА - все команды и прерывания не выполняются,
* ЗАГРУЗКА – первоначальная загрузка в соответствии с архитектурой S/390,
* СБОЙ – неопределенное состояние.

1. Средства управления работой процессора:

* Управляющие коды PSW
* Управляющие регистры

1. Пять вариантов сброса процессора:

* сброс процессора;
* начальный сброс процессора;
* сброс подсистемы;
* сброс с очисткой;
* сброс по питанию.

1. Конфигурация – конкретная компоновка вычислительной системы.
2. Основные принципы мультипроцессирования:

* общая разделяемая память;
* межпроцессорное взаимодействие;
* синхронизация часов.

1. Механизм обслуживания прерываний - дополнительное средство межпроцессорного обмена.
2. Шесть классов прерываний:
   * + По вызову супервизора,
     + Программные,
     + От схем контроля,
     + Ввода-вывода,
     + Перезапуска,
     + Внешние.
3. Плавающие прерывания - Прерывания, которые могут быть обработаны любым из процессоров конфигурации.
4. Средства временной синхронизации:

* **Часы TOD –** отсчет реального времени один для всех процессоров
* **Компаратор времени –** срабатывает, когда показание часов TOD превышает заданное программой значение
* **Процессорный таймер –** измеряет интервал заданный времени

## Вопросы для контроля усвоения материала главы 6

* Какая глубина прерывания у мейнфрейма?
* Что такое процессор?
* Что будет, если запросы на выполнение той или иной задачи поступят одновременно?
* В каких случаях требуется изменить показания часов ?
* Какие состояния бывают у процессора?
* Разные PLAR имеют разные часы TOD или общие?
* Что такое плавающие прерывания?
* Какая глубина прерывания у мейнфрейма?
* Как обрабатываются запросы на прерывания, поступившие одновременно?
* Где хранятся таблицы трассировки?
* Как часто обновляются записи при регистрации программных событий PER?
* Как взаимосвязаны регистрация программных событий PER и производительность системы?

## Глава 7. Работа с основной памятью. Адресация памяти

Что нас ждет? BD04972_

После прочтения этой главы Вы сможете:

1. назвать три типа физической памяти большой ЭВМ архитектуры zArchitecture;
2. перечислить типы адресов и адресных пространств;
3. назвать четыре режима виртуальной адресации и четыре типа виртуальных адресов;
4. объяснить механизм префиксации;
5. рассказать, как организована защита памяти;
6. описать форматы и типы используемых адресов памяти;
7. назвать способы преобразования одного типа адреса в другой тип

## Три типа физической памяти

В этой главе обсуждаются способы представления информации в основной памяти большой ЭВМ класса IBM мейнфрейм, а также адресация, защита памяти и то, как реализуются ссылки на те или иные ячейки памяти и как осуществляются изменения в записях. Мы рассмотрим также такие аспекты адресации, как форматы и типы адресов; дадим понятие адресного пространства и рассмотрим способы преобразования одного типа адреса в другой.

Физическая память в большой машине представляет собой три типа памяти:

* **внутренняя или процессорная память** имеет многоуровневую структуру, включающую до трех уровней буферной памяти, называемой кэш-памятью (Cache), причем, уровни кэш-памяти предназначены для увеличения быстродействия основной памяти и «прозрачны» для процессора, то есть программно недоступны;
* **основная память (Main Storage)** является многоабонентной и допускает одновременные обращения всех центральных процессоров, а также и процессоров канальной подсистемы, конструктивно выполнена в виде отдельных чипов на многочиповом модуле MCM;
* **расширенная память (Extended Storage)** программно доступна и допускает операции чтения-записи страниц с использованием специальных процессорных команд управления PAGE IN, PAGE OUT, конструктивно выполнена в виде отдельных плат, расположенных на плате центрального электронного комплекса CEC.

Физическая конструкция кэш-памяти или тип конкретного средства реализации кэш-памяти никак не сказывается результатах работы системы, за исключением увеличения общей производительности, более того, эффект наличия или отсутствия кэш-памяти не видим для программ. Обычно физически кэш связан с центральным процессором или с процессором ввода-вывода.

Отметим, что основная память может быть реализована на энергозависимом либо на энергонезависимом запоминающем устройстве. Содержимое энергозависимого запоминающего устройства не сохраняется при отключении питания. Для основной памяти, построенной на основе энергонезависимого запоминающего устройства (на момент написания книги наиболее популярное устройство последнего типа – флеш-память), отключение питание и затем включение никак не влияет на содержимое основной памяти. Однако все процессоры при отключении питания переходят в состояние СТОП (подробнее об этом см. главу «Управление работой процессора») и никакие изменения в основной памяти не могут быть сделаны, пока питание отключено. При этом для обоих типов памяти содержание ключей защиты памяти не обязательно останется сохраненным, если питание основной памяти будет отключено.

Основная память предоставляет системе быстрый доступ за счет специальным образом организованной системы адресации самой памяти (обратите внимание – не хранящихся данных, а ячеек самой памяти!). Основная память может в своем составе также иметь несколько уровней с разной скоростью доступа к данным. Мы знаем, что и данные и программы должны быть загружены в основную память (с устройств ввода), прежде чем они будут обработаны с помощью процессора. В кэш-памяти сохраняются те данные и коды программы, к которым наиболее часто обращается процессор.

Пример того, как может быть распределена основная память по назначению, а также того, как соотносятся реальные и абсолютные адреса, приведен в Приложении (см. п.10) к этой главе.

Память можно изобразить в виде длинной горизонтальной строки битов. Для большинства операций доступ к памяти представляет собой последовательное чтение слева направо этой строки битов. Строка битов может быть поделена на блоки по 8 битов в каждом, такой блок называется байтом, и это основной строительный блок для всех информационных форматов. Когда память рассматривается как набор байтов, ее представляют в виде листа, в котором строки по восемь бит (байты) располагаются снизу вверх или сверху вниз, и каждый байт имеет свой порядковый номер, нумерация памяти непрерывная, начинается с нуля. Фактически, получается адресация байтов. Адрес байта представляет собой двоичное 24-х , 31-о или 64-х разрядное число. Адреса будут подробно описаны в отдельном разделе.

### 1.1. Информационные форматы

Информация перемещается между памятью и центральным процессором или подсистемой ввода-вывода байтами или группой байтов во времени. Если это отдельно не определено, то группа байтов в памяти адресуется по адресу самого левого байта в группе. Число байтов в группе определяется выполняемой операцией. Когда байты используются в работе центрального процессора, группа байтов называется полем.

Внутри каждой группы байтов разряды нумеруются последовательно слева направо. Самый левый разряд иногда считается разрядом «высокого уровня», а самый правый разряд - разрядом «низкого уровня». Номера разрядов – это не адреса памяти. Адресуемыми могут быть только байты. Чтобы работать с конкретными разрядами в байте, необходимо получить доступ к конкретному байту.

Разряды в байте нумеруются с 0 до 7 слева направо. Разряды в адресе могут нумероваться следующим образом:

* для 24-х разрядной адресации: 8-31 или 40-63;
* для 31-й разрядной адресации: 1-31 или 33-63;
* для 64-х разрядной адресации: 0-63.

В рамках любого другого формата фиксированной длины большого количества байт, образующие формат разряды нумеруются последовательно начиная с нуля.

Для целей выявления ошибок в некоторых моделях один или более проверочных разрядов могут добавлены в каждый байт или в группу байтов. Такие проверочные разряды генерируются автоматически машиной и не могут напрямую управляться программой. Емкость любой памяти выражается в байтах.

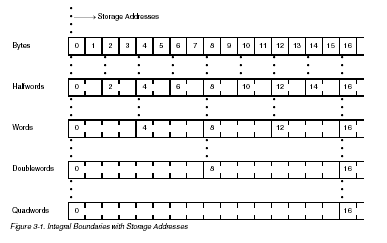
Когда длина поля операнда не определена кодом операции, поле заведомо имеет фиксированную длину, которая может быть равна один, два, четыре, восемь или шестнадцать байт. Более длинные поля тоже существуют, но они будут определяться отдельно для некоторых команд.

Когда длина поля операнда задана явно, тогда считается, что поле имеет переменную длину. Операнды переменной длины меняют длину, увеличивая ее побайтно.

Когда информация помещается в память, содержание только тех байт замещается, которые включены в обозначенное поле, даже хотя ширина физического пути к памяти может оказаться больше, чем длина поля, которое должно быть сохранено.

### 1.2. Целочисленные границы в памяти

Определенные блоки информации должны укладываться в целочисленные границы в памяти. Границы называются целочисленные для блока информации, когда их адрес в памяти умножается на длину блока в байтах. Специальные имена даны полям размерами в 2, 4, 8 и 16 байт в интегральных границах. Полуслово – это группа из двух последовательных байт внутри двух-байтной границы и это основной строительный блок для команд. Слово - это группа из четырех последовательных байт внутри четырех-байтной границы. Двойное слово - это группа из восьми последовательных байт внутри восьми-байтовой границы. Квад-слово - это группа из шестнадцати последовательных байт внутри шестнадцати-байтной границы. Рисунок 1 иллюстрирует эти целочисленные границы.



**Рисунок 1** – целочисленные границы адресов памяти

Когда адреса в памяти обозначают полуслово, слово, двойное слово или квад-слово, двоичное представление адреса содержит один, два, три или четыре самых правых нулевых разряда, соответственно.

Команды должны быть вписываемы в рамки двухсловных целочисленных границ. Слово управления каналом CCW (Channel Control Word), IDAWs (Indirect Address Word), MIDAWs (Multiple Indirect Address Word), и операнды памяти определенных команд должны быть в других целочисленных границах. Операнды памяти большинства команд не имеют требований к заданию границ.

**Замечание для программистов**: для операций с фиксированной длиной поля длины поля, которые являются степенью числа 2, замечено значительное ухудшение производительности когда операнды памяти не позиционированы по адресам, которые представляют собой целочисленные произведения длины операнда. Для повышения производительности часто используемые операнды памяти должны быть соотнесены с целочисленными границами.

## Типы адресов и адресных пространств основной памяти

Итак, адрес памяти – это адрес нахождения байта в памяти. Адресация памяти – это адресация байтов. Каждый байт имеет свой номер в памяти.

**Преподавателю**: Здесь уместно провести аналогию с почтовой адресацией, электронной почтой и идентификацией автомобилей.

Для адресации основной памяти используются три основных типа адресов: абсолютный, реальный и виртуальный. Эти адреса различаются по тому, какие преобразования с ними необходимо провести, чтобы получить доступ к основной памяти. Динамическое преобразование адреса DAT (Dynamic Address Translation) переводит виртуальный адрес в реальный, префиксация переводит реальный адрес в абсолютный. Кроме этих трех основных типов адреса существуют еще дополнительные типы. Каждый из дополнительных типов, так или иначе, является модификацией одного из этих основных типов, в зависимости от команды и от текущего режима работы процессора.

В соответствии с основными типами адресов существуют три типа адресных пространств в большой ЭВМ класса IBM мейнфрейм (см. рисунок 2):

* абсолютное, одно на систему;
* реальные, их будет столько, сколько в системе работающих центральных процессоров;
* виртуальные, о их количестве поговорим отдельно.

Внутри абсолютного адресного пространства особым, хитрым, образом вписываются адреса реальных адресных пространств центральных процессоров системы и виртуальные адресные пространства отдельных операционных систем, программ, пользователей и даже каждой сессии одного пользователя, если он подключается к системе удаленно. При этом исключаются конфликтные ситуации, степень защиты данных –высочайшая.

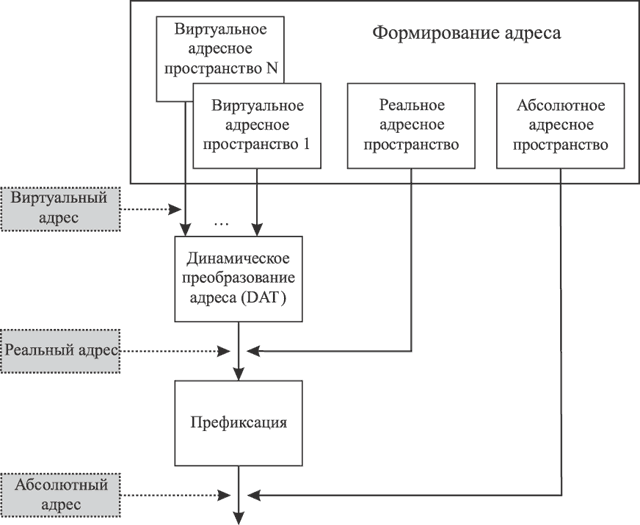


Рисунок 2 – типы адресов и адресных пространств в БЭВМ

Как видно из рисунка 2, при формировании адреса используются следующие типы адресов:

* абсолютный адрес, получаемый после преобразования в префиксном регистре;
* реальный адрес, который транслируется в абсолютный с помощью регистра префикса;
* виртуальный адрес, транслируемый в реальный с использованием механизма динамической переадресации (DAT).

При работе программ используется также понятие эффективный адрес, видимый программой, он может быть виртуальным либо реальным адресом двух типов: логический адрес или адрес команды.

### 2.1. Абсолютный адрес и абсолютное адресное пространство

Абсолютный адрес – это адрес однозначно определяемого байта основной памяти. Абсолютный адрес используется для доступа к памяти без каких-либо преобразований над ним.

Все центральные процессоры и подсистема ввода-вывода в конкретной конфигурации системы ссылаются на определенное место совместно используемой основной памяти, используя один и тот же абсолютный адрес.

Абсолютное адресное пространство позволяет системе не запутаться, когда она обслуживает множество процессоров, множество операционных систем, множество программ, при этом для каждого из перечисленных создается свое адресное пространство. Именно абсолютное адресное пространство обеспечивает правильную работу системы, целостность данных при обращении к ним одновременно нескольких процессоров. Соответственно, абсолютный адрес непосредственно, без каких-либо преобразований определяет ячейку физической памяти. Каждой ячейке физической памяти соответствует не более чем один абсолютный адрес.

Абсолютное адресное пространство – одно для одной системы (машины или логического раздела LPAR). В отличие от абсолютного адресного пространства, реальных адресных пространств столько, сколько реально работающих процессоров в системе.

Абсолютное адресное пространство – это такая нумерация байтов памяти, в которой каждой ячейке каждого физического типа памяти имеется одно строго определенное место. Внутри абсолютного адресного пространства вписаны и реальное адресное пространство, и виртуальные адресные пространства. Причем, последние имеют свою классификацию: они могут быть первичными и вторичными адресными пространствами.

Абсолютное адресное пространство – это то пространство адресов физической памяти, в котором для каждой ячейки физической памяти существует один и только один адрес. При этом физическая память – это может быть внутренняя память процессора, основная память, дополнительная память и даже внешняя, расположенная на внешних носителях (магнитные ленты и диски, лазерные диски и т.д.).

Мы можем представить абсолютное адресное пространство в виде одного длинного свитка бумаги с линованными строчками, где каждой строчке соответствует номер, и всего таких номеров в абсолютном адресном пространстве большой ЭВМ класса IBM мейнфрейм – 264 , как это показано на рисунке 2.

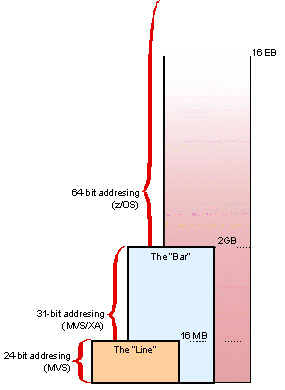


Рисунок 2 – визуальное представление абсолютного адресного пространства машины

### 2.2. Реальный адрес и реальные адресные пространства

Реальный адрес определяет положение в реальной памяти. Реальный адрес – это адрес ячейки реальной памяти, преобразуемый в абсолютный с помощью механизма префиксации. Всегда существует один реальный адрес, отображаемый в один абсолютный адрес для каждого процессора в системе.

В многопроцессорной системе каждый процессор работает в своем собственном адресном пространстве – реальном, где он «знает», в каком месте его адресного пространства записаны те или иные необходимые ему для работы коды (например, слово состояния программы для различных классов прерывания) и где, в каком месте, у него записаны адреса, по которым он должен обращаться к тем или иным данным (например, к накопителю на диске прямого доступа к данным DASD).

Понятие «реальное адресное пространство» используется в многопроцессорных системах для того, чтобы каждый процессор при работе «чувствовал себя» так, как будто он один владеет всей машиной, как будто в его распоряжении все пространство адресов. А поскольку процессоров в современном мейнфрейме бывает до 54-х (на момент написания этой книги), для исключения конфликтов одновременного обращения нескольких процессоров к одной и той же ячейке физической памяти, введен механизм префиксации. Суть механизма префиксации заключается в том, что каждый процессор работает с адресным пространством, нумерация которого начинается с нуля, причем, нулевая ячейка обозначает физическую память, соответствующую внутренней памяти именно этого процессора. Следовательно, необходимо отобразить начальную область реальной памяти каждого процессора на одну из областей физической памяти с разными абсолютными адресами. Эту задачу и решает механизм префиксации.

Когда центральный процессор использует реальный адрес, чтобы попасть в основную память, адрес преобразуется в абсолютный с помощью префиксации. Адрес изменяется на величину, которая хранится в регистре префикса каждого центрального процессора.

Память, состоящая из последовательности байтов, соответствующих их реальным адресам, называется реальная память.

Реальных адресных пространств в машине столько, сколько работающих центральных процессоров. Если процессоров много, сразу несколько из них могут обратиться к тем же данным на диске DASD (например, записать в одну и ту же ячейку свои данные).

Чтобы уменьшить число конфликтов при обращении различных процессоров в процессе обработки прерываний, и обеспечить четкую и бесконфликтную работу системы, был разработан механизм префиксации, преобразующий реальный адрес каждого процессора (и таких адресов может быть столько, сколько работает процессоров в системе) в абсолютный уникальный единственный для каждого работающего процессора адрес в системе.

### 2.3. Виртуальный адрес и виртуальные адресные пространства

Идея виртуализации памяти появилась в архитектуре машин, разрабатываемых в IBM, еще в 70-е годы прошлого столетия. Основная идея заключалась в том, чтобы увеличить число одновременно работающих программ, которые «чувствуют себя» так, как будто каждая безраздельно владеет всем абсолютным адресным пространством машины.

Введение механизма виртуализации позволило повысить безопасность работы программ. Более того, пользователи получили возможность иметь свои собственные адресные пространства. Поначалу количество создаваемых виртуальных адресных пространств было ограничено определенным числом – номером адресного пространства. В архитектуре zArchitecture их количество увеличено за счет введения дополнительных регистров доступа, так что каждый из 16-ти 32-х разрядных регистров доступа может создать еще 232 виртуальных адресных пространства.

**Преподавателю:** Привести аналогию виртуальной памяти – наличие различных амбарных книг в камере хранения, на складе – взять пример из книги Солдатова В.Н.

Виртуальный адрес определяет ячейку памяти в виртуальной памяти. Когда виртуальный адрес используется для доступа к основной памяти, он преобразовывается в реальный адрес с помощью механизма динамического преобразования адресов, а затем из реального в абсолютный с помощью механизма префиксации.

Тип виртуального адреса определяется режимомвиртуальной адресации, используемым в конкретном процессоре.

Виртуальные адреса формируются процессором при исполнении программ и транслируются в реальные посредством динамического преобразования адресов (Dynamic Address Translation - DAT).

Процесс динамического преобразования адресов (DAT) реализуется в машине с помощью многоуровневых взаимосвязанных (look-up) вспомогательных таблиц, которые и описывают преобразование адреса из виртуального в реальный. Основные указатели на эти таблицы хранятся в регистрах управления CR1, CR7 и CR13. Кроме того, таблицы могут быть определены с помощью регистров доступа. Подробнее динамическое преобразование адресов будет рассмотрено в следующей главе.

В результате одновременно могут быть доступны многочисленные виртуальные адресные пространства.

Адресное пространство, образуемое виртуальными адресами, называется виртуальным адресным пространством.

Виртуальное адресное пространство – это последовательность натуральных чисел (виртуальных адресов), в которой каждое число в результате специальных преобразований соотносится с определенным местом байта в основной памяти. Последовательность чисел начинается с нуля и счет идет снизу вверх для каждого виртуального адресного пространства, так же как это показано на рисунке 2 для абсолютного адресного пространства.

Когда виртуальный адрес используется центральным процессором, чтобы попасть в основную память, он сначала преобразуется в реальный адрес с помощью механизма динамического преобразования адреса (Dynamic Address Translation - DAT), а затем реальный адрес с помощью механизма префиксации преобразуется в абсолютный адрес. При этом механизм DAT может использовать параметры преобразований, которые хранятся в специальных таблицах DATT, при этом могут использоваться от пяти до двух таблиц преобразования. Эти таблицы называются соответственно сверху вниз: таблица первого региона, таблица второго региона, таблица третьего региона, таблица сегмента и таблица страницы. Запись (обозначение) (начальный адрес и длина) таблицы самого верхнего уровня для специального адресного пространства называется элемент ASCE (address space control element) – управляющий элемент адресного пространства. Этот элемент ASCE, необходимый для использования в DAT, находится в регистре управления или может быть определен в регистре доступа. Как альтернативный вариант, элемент ASCE для адресного пространства может быть записью реального пространства, что показывает, что механизм DAT преобразует виртуальный адрес, просто воспринимая его, как реальный адрес без использования таблиц.

В разное время механизм DAT использует элемент управления адресным пространством (ASCE), полученный в разных управляющих регистрах или определенный с помощью регистров доступа. Выбор определяется с помощью индикатора режима преобразования, который задается в текущем слове состояния программы PSW.

### 2.4. Режимы виртуальной адресации и типы виртуальных адресов

Взаимосвязь между режимами виртуальной адресации и типами используемых виртуальных адресов можно проследить с помощью рисунка 3. Существует четыре режима работы процессора с виртуальным адресным пространством:

* Режим главного адресного пространства (Primary-space mode);
* Режим вторичного адресного пространства (Secondary-space mode);
* Режим с адресным пространством, определяемым регистром доступа (Access-register mode) и
* Режим базового адресного пространства AS (Home-space mode).

В зависимости от того, в каком режиме работает процессор, будут использоваться различные виртуальные адреса:

* + главный виртуальный адрес (Primary Virtual Address),
  + вторичный виртуальный адрес (Secondary Virtual Address),
  + виртуальный адрес, определяемый регистром доступа (AR-specified Virtual Address) и
  + базовый виртуальный адрес - Home Virtual Address).

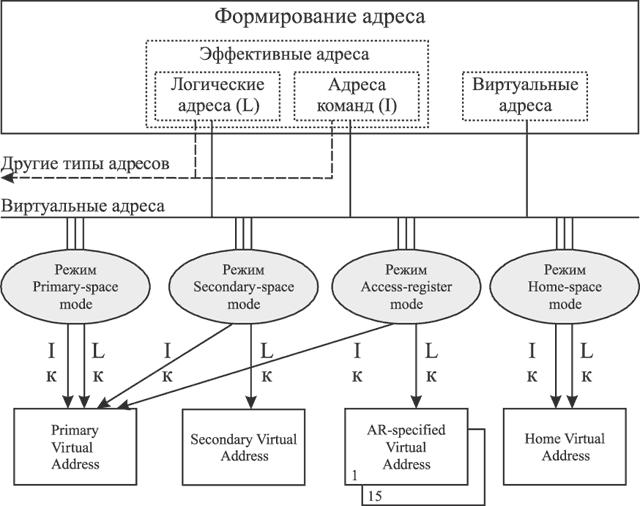


Рисунок 3 – Режимы адресации и типы адресов

Как видно из рисунка 3 в режимах главного или базового адресного пространства процессору доступны, соответственно, только главное или базовое виртуальные адресные пространства. В режиме вторичного адресного пространства процессор может транслировать адреса и главного, и вторичного адресного пространства. В режиме с адресным пространством, определяемым регистрами доступа, процессор оперирует в главном и в пятнадцати пространствах, определяемых регистрами доступа AR1÷AR15.

Рассмотрим подробнее четыре типа виртуальных адресов.

#### Главный виртуальный адрес (Primary Virtual Address)

Главный виртуальный адрес - это виртуальный адрес, который получается благодаря преобразованию адресов с использованием первичного элемента контроля адресного пространства PASCE (primary address-space-control element). Логические адреса считаются главными виртуальными адресами, если процессор работает в режиме главного адресного пространства. Адреса команд считаются главными виртуальными адресами, если процессор работает в режиме главного адресного пространства, вторичного адресного пространства или режиме регистров доступа. Адрес первого операнда команды MOVE TO PRIMARY и адрес второго операнда команды MOVE TO SECONDARY всегда считаются главными виртуальными адресами.

#### 2.3.2. Вторичный виртуальный адрес (Secondary Virtual Address)

Вторичный виртуальный адрес - это виртуальный адрес, который получается благодаря преобразованию адресов с использованием вторичного элемента контроля адресного пространства SASCE (secondary address-space-control element). Логические адреса считаются вторичными виртуальными адресами, если процессор работает в режиме вторичного адресного пространства. Адрес второго операнда команды MOVE TO PRIMARY и адрес первого операнда команды MOVE TO SECONDARY всегда считаются вторичными виртуальными адресами.

#### 2.3.3. Виртуальный адрес, определяемый регистром доступа (AR-Specified Virtual Address)

Виртуальный адрес, определяемый регистром доступа, это виртуальный адрес, получаемый с помощью определяемого регистром доступа элемента ASCE (address-space-control element). Логические адреса считаются адресами, определяемыми регистром доступа, если процессор работает в режиме регистра доступа.

#### 2.3.4. Базовый виртуальный адрес (Home Virtual Address)

Базовый виртуальный адрес – это виртуальный адрес который получается после преобразований с помощью элемента HASCE (home address-space-control element).

### Эффективный адрес

На рисунке 3 видно, что используется также понятие эффективный адрес. Эффективный адрес – это адрес, формируемый процессором до выполнения процессов DAT или префиксации.

Эффективный адрес задается ссылкой на регистр, в котором он уже был сформирован либо вычисляется с использованием адресной арифметики.

Эффективный адрес – это адрес, который существует до того, как были выполнены динамическое преобразование адресов и префиксация. Эффективный адрес может быть определен напрямую в регистре или может быть результатом адресной арифметики. Существует два способа вычисления эффективного адреса: 1) суммирование с базовым замещение или 2) вычисление на основе базового, индекса и замещение.

В зависимости от режима работы процессора (а их существует четыре, как мы только что рассмотрели), эффективный адрес может быть двух видов:

* Логический адрес (Logical Address - L) – адрес операнда для большинства команд, может быть реальным в реальном режиме или виртуальным четырех типов в зависимости от режима адресации;
* Адрес команды (Instruction Address – I) – используется для выборки команд, может быть реальным в реальном режиме или виртуальным адресом в режимах главного, вторичного и адресного режима, определяемого регистрами доступа, а также базовым в режиме базового адресного пространства.

#### 2.5.1. Логический адрес

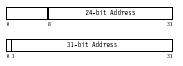
Если это не оговорено особо, адрес операнда памяти для большинства команд – это логические адреса. Логический адрес считается реальным в режиме реальной адресации, главным виртуальным адресов в режиме главного адресного пространства (Primary-space mode); вторичным виртуальным адресом в режиме

вторичного адресного пространства (Secondary-space mode), виртуальным адресом, определенным регистром доступа в режиме регистра доступа (Access-register mode) и базовым виртуальным адресом в режиме базового адресного пространства (Home-space mode). Некоторые команды имеют адреса операнда памяти или доступ к памяти ассоциированный с командой, которые не разрешают использовать правила для логических адресов. Во всех таких случаях, определитель команды содержит описание типа адреса.

#### 2.5.2. Адрес команды

Адреса, используемые, чтобы извлечь команду из памяти, называются адресами команд. Адреса команд считаются реальными адресами, если процессор работает в режиме реальной адресации, главными виртуальными адресами, если работает режим главного адресного пространства, вторичного адресного пространства или адресного пространства, определяемого регистрами доступа. Адреса команд считаются базовыми виртуальными адресами, если процессор работает в режиме базового адресного пространства. Адрес команды в текущем слове состояния программы PSW и целевой адрес команды EXECUTE – это адреса команд.

Разряды адреса независимо от режима адресации нумеруются с 1 по 31, когда имеет место 24-х и 31-о разрядная адресация (в этом случае нулевой разряд определяет режим: 24 или 31). Когда 24-х разрядный или 31-о разрядный адрес находится в четырех-байтовом поле, разряды нумеруются с 8-го по 31-ый или с 1-го по 31-й, соответственно (см. рисунок 4).



**Рисунок 4** – нумерация разрядов в полях адреса

### Размеры адресов

Размеры адресов зависят от максимального числа значимых разрядов, которые могут представлять адрес. Три размера адреса предоставляются: 24-разрядный, 31-разрядный и 64-х разрядный адрес.

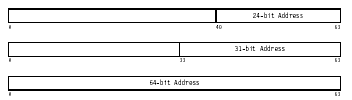
Адрес размером в 24 разряда может описать, однозначно определить, память размером максимум в 16777216 (16 мегабайт) байт, адрес размером в 31 разряд может однозначно описать память размером в 2147483648 (2 гегабайт) байт, и размер адреса в 64 разряда позволяет описать 18,446,744,073,709,551,661 (16 экзабайт) байт.

Разряды в 24-разрядном, 31-разрядном и 64-разрядном адресе, получаемые адресной арифметикой под контролем текущего режима адресации, нумеруются соответственно: 40-63, 33-63 и 0-63 (см. рисунок 5). Эта нумерация соотносится с нумерацией в базовом адресе и индексных разрядах регистров общего назначения.

Виртуальные адреса 24-х разрядный и 31-разрядный добавляются до 64 разрядов добавлением соответственно 40 или 33 нулей слева, прежде чем начнется процесс их динамического преобразования.

Реальные адреса 24-х разрядный и 31-разрядный точно так же расширяются до 64-х разрядов до процесса префиксации. Абсолютные адреса 24-х разрядный и 31-разрядный расширяются до 64-х разрядного прежде чем обращаться к основной памяти. Таким образом, 24-х разрядный адрес всегда обозначает ячейку в первом 16-ти мегабайтном блоке 16-ти экзабайтной памяти, адресуемой 64-х разрядным адресом. А 31-разрядный адрес всегда обозначает ячейку в первом 2-х гегабайтном блоке.

Когда бы машина ни генерировала для программы 24-х разрядный или 31-разрядный адрес, адрес становится доступным (помещен в память или загружен в регистр общего назначения) будучи вставленным в 32-х разрядное поле, соответственно с одним или восемью левыми нулями. Когда адрес загружается в регистр общего назначения, разряды 0-31 64-х разрядного регистра остаются неизменными.



**Рисунок 5** – формат адреса в полях регистра общего назначения

Размер эффективных адресов управляется разрядами 31 и 32 слова состояния программы PSW, которые определяют соответственно разряды расширенной адресации и базовой адресации. Когда разряды 31 и 32 равны нулю, это означает, что центральный процессор работает в режиме 24-х разрядной адресации, 24-х разрядный операнд и эффективные адреса команд определены. Когда разряд 31 равен нулю, а разряд 32 равен 1, это означает, что центральный процессор работает в режиме 31-разрядной адресации, 31-разрядный операнд и эффективные адреса команд определены. Когда разряды 31 и 32 равны единице, это означает, что центральный процессор работает в режиме 64-х разрядной адресации, 64-разрядный операнд и эффективные адреса команд определены.

## Механизм префиксации

Как уже говорилось в п.2, префиксация – это перевод реального адреса в абсолютный адрес. Префиксация предоставляет возможность задать промежуток реальных адресов в диапазоне от 0 до 8191 для разных блоков в абсолютной памяти для каждого центрального процессора CPU таким образом, чтобы два и более процессора могли одновременно использовать в своей работе основную память при наименьшем взаимодействии между собой, особенно во время обработки прерываний. При этом каждый CPU будет «думать», что он безраздельно и полностью владеет всей основной памятью.

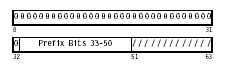
Префиксация вынуждает реальные адреса в диапазоне 0 – 8191 соответствовать один к одному 8-ми килобайтному блоку абсолютных адресов (префиксная область). Адреса этих блоков определяются величиной, записанной в разрядах 0-50 регистра префикса каждого CPU, а блок реальных адресов определенный величиной в префиксном регистре каждого CPU соответствует один к одному абсолютным адресам, находящимся в диапазоне 0 – 8191. Оставшиеся реальные адреса – те же самые, которые соответствуют абсолютным адресам. Это преобразование позволяет каждому CPU получить доступ ко всей основной памяти, включая первые 8 килобайт и к ячейкам памяти, определяющим префиксные регистры других процессоров.

Префикс - это 51-разрядная величина, находящаяся в разрядах от 0 до 50-го регистра префикса, предназначенная для преобразования реального адреса в абсолютный и наоборот.

**Преподавателю:** Можно привести аналогию с перемещением данных между двумя ячейками таблицы – чтобы не потерять запись, нужно использовать еще одну ячейку.

### 3.1. Формат регистра префикса

Регистр префикса имеет формат, показанный на рисунке 4.



**Рисунок 4 –** Формат регистра префикса

Другие названия префикса: расположение фиксированной памяти, ядро низкого уровня, префиксная область. Назначенное расположение памяти используется для обмена информацией между системой и программным обеспечением, то есть, для управления прерываниями. Префиксная область располагается по адресу 0 – 0x1FFF (в реальной памяти).

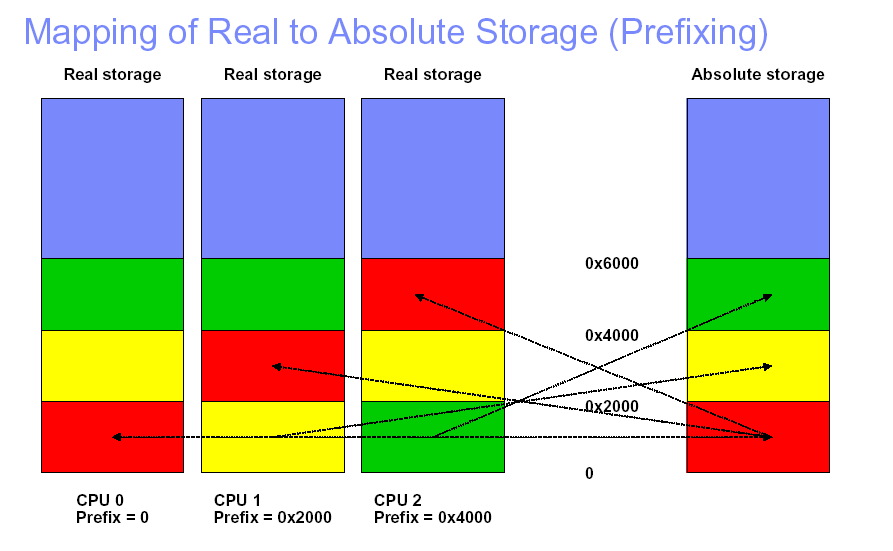
Механизм взаимодействия между реальной и абсолютной адресацией графически изображен на рисунке 5.

Каждый центральный процессор (CPU) должен управлять своей собственной информацией, поэтому для каждого процессора этот диапазон адресов должен быть представлен в отдельном месте в абсолютной памяти. Регистр префикса каждого центрального процессора определяет этот абсолютный адрес. Программное обеспечение должно удостовериться, что каждый центральный процессор имеет отдельный регистр префикса.

Существенно, что диапазон 0 – 0x1FFF и диапазон регистра префикса взаимозаменяемы. Каждый центральный процессор имеет отдельную область адресов, адресуемую как 0 – 0x1FFF.

Если регистр префикса равен нулю, это значит, что механизм префиксации не работает. Регистр префикса определяет адреса в диапазоне, меньшем 2 G (Гегабайт).

В архитектуре ESA/390, регистр префикса содержал адрес в пределах 4K и применялся только к диапазону адресов до 4K (0 – 0xFFF).



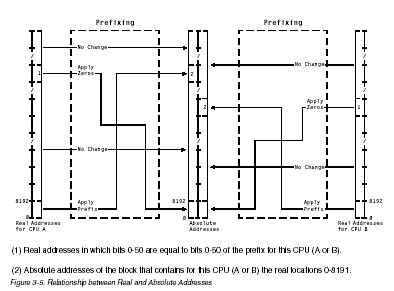
**Рисунок 5** – Отображение реальной памяти каждого процессора в абсолютную память системы

Рисунок иллюстрирует преобразование реальной памяти в абсолютную память с помощью префиксации. Предположим, машина имеет три центральных процессора. Префиксный регистр первого CPU0 равен 0, второго 0х2000, третьего 0х4000. Это означает, что реальный адрес третьего ЦП с диапазоном 0х1FFF преобразуется в диапазон 0х4000 – 0х5FFF в абсолютной памяти. А реальные адреса в диапазоне 0х4000 – 0х5FFF транслируются в диапазон 0х1FFF абсолютной памяти. Во всех других случаях реальные и абсолютные адреса идентичны.

Механизм префиксации поясняется и рисунком 6. Стрелки показывают, как соотносятся куски реальной памяти с абсолютной памятью. В реальной памяти у каждого процессора есть своя область с адресами 0-8191 (до 224), это та область, где находится важнейшая информация, относящаяся именно к этому процессору. В абсолютной памяти область с номерами 0-8191 должна быть общей для всех процессоров. Как этого добиться? В абсолютной памяти эти области для каждого процессора должны лежать в разных местах, то есть, должны иметь разные номера. Как выйти из положения? Вести «двойной учет»: (пример со складом) у каждого клиента (процессора) своя учетная книга (реальная память), в учетной книге все детали клиентов (их вещи, коробки) записаны по-порядку (реальные адреса). А для того, чтобы знать, на какой полке на складе на самом деле хранится конкретная коробка (в каком месте в абсолютной памяти находятся адреса конкретных физических ячеек памяти), надо заглянуть еще в одну книгу (таблицу переадресации или в регистр префиксации).

Перевод процессора в режим базового адресного пространства возможен только в режиме супервизора. Программа может все-же быть адресуемая из других адресных пространств, если будут использованы непривилигированные команды для изменения содержания регистров доступа или с помощью псевдопривилегированных или привилегированных команд, которые изменяют элементы ASCE в регистрах 1 и 7 каждого процессора.

Если программа обращается в реальную память конкретного процессора в область префиксации с адресами 0 ÷ 8191, обязательно осуществляется преобразование реального адреса в абсолютный путем замены старших разрядов 0 ÷ 50 на префикс, находящийся в регистре префикса каждого процессора. При обращении программы в реальную память в область с номером, равным префиксу, старшие разряды 0 ÷ 50 реального адреса заменяются в абсолютном адресе на 0. Остальные реальные адреса соответствуют абсолютным адресам без преобразования. Содержимое регистра префикса может быть загружено или прочитано командами процессора. Загрузка нулевого префикса означает идентичность реальных и абсолютных адресов.

****

**Рисунок 6** – Взаимоотношения между реальными и абсолютными адресами

На рисунке 6 показаны два реальных адресных пространства процессоров А и В и абсолютное адресное пространство, расположенное в центре рисунка. Цифрой 1 у каждого процессора обозначена область памяти реальных адресов, в которой разряды от 0 до 50 равны таким же разрядам префиксного регистра для этого процессора. Цифрой 2 обозначены области в абсолютном адресном пространстве (разные!), в которые записывается содержание префикса (первых 8 Кбайт) каждого процессора. Наличие областей, обозначенных цифрами 1 и 2, отличает реальное адресное пространство от абсолютного.

### Правила префиксации

Реальный адрес преобразуется в абсолютный адрес по одному из трех правил в зависимости от состояния разрядов 0-50 в реальном адресе:

1. Если все биты 0-50 реального адреса нулевые, тогда они заменяются битами 0-50 префикса.
2. Если биты 0-50 реального адреса равны битам 0-50 префикса, тогда они заменяются нулями.
3. Если не все биты 0-50 реального адреса нулевые и при этом они не равны битам 0-50 префикса, тогда оставляем все неизменным.

С помощью механизма префиксации преобразуются только те адреса, которые представлены в памяти. Содержание источника адреса остается неизменным.

Отличие между реальными и абсолютными адресами существует даже тогда, когда регистр префикса содержит одни нули, в этих случаях реальный адрес и соответствующий ему абсолютный адрес идентичны.

## Вопросы для контроля усвоения материала

* Какие типы физической памяти существуют в мейнфрейме?
* Что такое основная память?
* Какие типы адресов используются процессором?
* Какие режимы адресации бывают у процессора?
* Какие существуют режимы виртуальной адресации?
* Что такое префиксация?
* Для чего служит динамическое преобразование адресов?
* Чем отличаются логический и эффективный адреса?
* Где хранятся таблицы трассировки?

## Краткие выводы по главе 7

Оглянемся назад? 

1. Три типа физической памяти:

* **Кэш-память** – буферная память, обеспечивающая быстрый доступ к данным;
* **Основная память** - отдельный чип на многочиповом модуле МСМ;
* **Расширенная память** – отдельная плата на плате СЕС.

1. Типы адресов и адресных пространств памяти (АП):

* абсолютный адрес и абсолютное АП,
* реальный адрес и реальное АП и
* виртуальные адрес и виртуальные АП.

1. Типы виртуальных адресов:
   * + Главный виртуальный адрес;
     + Вторичный виртуальный адрес;
     + Виртуальный адрес, определяемый регистром доступа;
     + Базовый виртуальный адрес.
2. Механизм префиксации заключается в следующих шагах:
   * + 1. Префиксная область реального адресного пространства процессора записывается в определенную область абсолютного адресного пространства**;**
       2. Адрес области абсолютного АП сохраняется в регистре префикса процессора;
       3. В область префикса абсолютного АП записывается адрес реального АП, в которой хранится адрес области префикса процессора**.**

# Глава 8. Работа с основной памятью. Динамическое преобразование адресов.

Что нас ждет? 

После прочтения этой главы Вы сможете:

1. перечислить этапы динамического преобразования адресов;
2. назвать два варианта задания адресного пространства;
3. дать определение авторизации адресного пространства;
4. описать формат виртуального адреса;
5. сказать, какие типы строк использует буфер быстрой адресации.

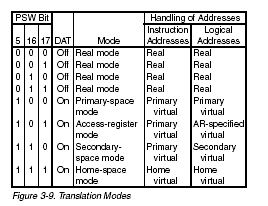
## Этапы динамического преобразования адресов

Процесс динамического преобразования виртуальных адресов в реальные состоит из трех этапов. На первом этапе выбирается один из двух способов задания виртуального адресного пространства (используя номер адресного пространства ASN или используя регистры доступа, которые в архитектуре zArchitecture были специально разработаны для увеличения количества адресных пространств). На втором этапе находится специальный управляющий элемент (или его называют еще код) ASCE (Address Space Control Element). Для этого выполняется так называемая ASN-трансляция, если адресное пространство задано номером, либо ALET-трансляция, то есть, использование записи в списке доступа, если при задании адресного пространства использовались регистры доступа. На третьем этапе преобразования с помощью полученного кода ASCE и многоуровневых таблиц преобразования, находящихся в основной памяти машины, выполняется собственно динамическое преобразование виртуального адреса в реальный.

В главе подробно рассматриваются принципы, на которых основаны эти трансляции. Рассмотрены форматы этих таблиц, организованные таким образом, что уже в процессе преобразований возможны изменения режимов виртуальных адресов и типов адресных пространств. Именно поэтому процесс называется динамическим преобразованием адреса (DAT). Основные указатели к этим таблицам хранятся в управляющих регистрах (CR1, CR7, и CR13) или описываются в регистрах

доступа. Отметим, что многочисленные адресные пространства могут быть доступны одновременно.

Первый раздел главы посвящен средствам, обеспечивающим процесс DAT.



В таблице 1 показаны восемь возможных режимов работы процессора и разряды слова состояния программы, которые показывают, в каком адресном пространстве работает процессор. Как процессор переходит из одного режима адресации в другой? Как создаются адресные пространства? Как происходит смена адресных пространств в процессоре? Информация в этой главе позволит вам ответить на эти вопросы.

## Два варианта задания адресного пространства в процессоре

Существуют два варианта задания адресного пространства.

Первый вариант предусматривает использование до 216-ти адресных пространств, задаваемых уникальным 16-разрядным номером адресного пространства ASN (Address Space Number).

Второй вариант задания адресного пространства предполагает использование регистров доступа AR1 – AR15, в каждый из которых может быть загружен код ALET (Access List Entry Token), определяющий адресное пространство.

Принцип трансляции номера ASN или кода ALET в коды ASCE пояснен на рисунке 1. Как видно из рисунка 1, код управления адресным пространством ASCE (Address Space Control Element) получается в результате трансляции номера ASN или трансляции кода ALET (Access List Entry Token) с использованием таблицы списков доступа, и именно код ASCE определяет параметры процесса динамического преобразования адресов DAT.

Для реализации первого способа необходимы две таблицы (AFT и AST), расположенные в основной памяти, и управляющий регистр CR14. Второй способ предполагает использование управляющих регистров CR2 и CR5, а также нескольких таблиц в основной памяти.

Смена адресных пространств в процессоре выполняется путем загрузки номера адресного пространства (ASN) в соответствующие управляющие регистры привилегированными и полупривилегированными командами или путем загрузки кодов в списке доступа ALET (Access List Entry Token), в регистры доступа непривилегированными командами. Далее код ASN или ALET транслируется в код управления адресным пространством ASCE (Address Space Control Element), определяющий параметры процесса динамического преобразования адреса в данном адресном пространстве.

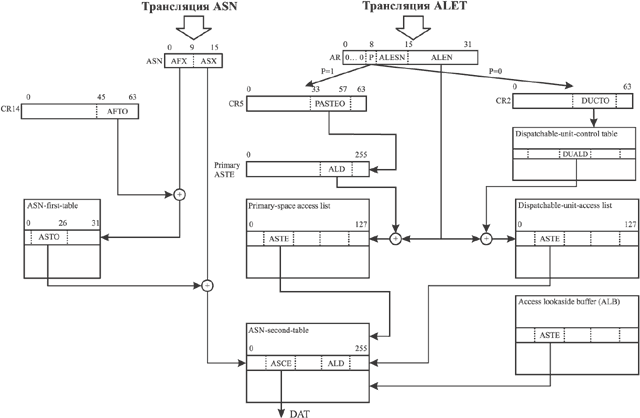


Рисунок 1 – преобразование кодов ASN и ALET в управляющий элемент ASCE

## Номер адресного пространства ASN

Адресное пространство может быть задано номером ASN (Address Space Number) с помощью управляющей программы. Номер ASN указывает с помощью двух-уровневых таблиц в основной памяти на запись в таблице ASTE (ASN-Second-Table-Entry), которая, в свою очередь, содержит информацию об адресном пространстве. Если таблица ASTE отмечена как доступная, она содержит управляющий элемент ASCE (Address Space Control Element), который определяет адресное пространство.

При определенных условиях псевдо привилегированные команды устанавливают новый код ASCE в управляющий регистр CR1 или CR7, считывая этот код ASCE из таблицы ASTE в основной памяти. Некоторые из этих команд используют механизм преобразования, так называемую ASN-трансляцию, которая по номеру ASN может определить нахождение в памяти таблицы ASTE.

Без знаковый 16-ти разрядный двоичный формат кода ASN позволяет создать 2 16 = 64К уникальных кодов ASN.

Номера ASN для главного и вторичного адресных пространств задаются положениями в управляющих регистрах. Номер ASN для главного адресного пространства называется главным номером PASN (primary ASN), он записывается в разряды 48-63 в четвертый управляющий регистр CR4, а для вторичного адресного пространства называется вторичный номер SASN и записывается в разряды 48-63 третьего управляющего регистра CR3 (см. рисунок 2).



**Рисунок 2** – формат главного и вторичного номеров ASN

Псевдо привилегированная команда, которая загружает главный номер (PASN) или вторичный номер (SASN) в соответствующий управляющий регистр (CR4 или CR3), загружает также соответствующий номер ASN в управляющие регистры CR1 или CR7.

Номер ASN для базового адресного пространства (home address space) не задает положение в управляющем регистре.

В регистре доступа наличие числа 0 или 1 определяет то, какое адресное пространство используется – главное или вторичное, и соответственно, в каком из управляющих регистров CR1 или CR7 будет находиться код ASCE. Регистр доступа, содержащий любую другую величину, будет определять вход в таблицу ALET, которая называется список доступа. Указанная таблица будет содержать реальный адрес таблицы ASTE для адресного пространства, задаваемого регистром доступа.

Код ASCE, задаваемый регистром доступа, находится в таблице ASTE. Преобразуя содержание регистра доступа мы получаем код ASCE для использования в динамическом преобразовании адреса, не используя при этом самого номера ASN.

**Замечание для программиста:** Поскольку таблица ASTE находится из списка доступа с помощью ее адреса вместо использования номера ASN, эту таблицу принято называть псевдо-ASTE, подчеркивая тот факт, что результат получен не средствами двух-уровневых таблиц ASN-трансляции, а по-другому. Номер уникального псевдо-ASTE может быть больше, чем номера уникальных ASN и ограничен только размером памяти, доступной для размещения записей таблицы ASTE.

## Номера таблицы ASTE

### 4.1. Текущий номер в таблице ASTE (ASTESN)

Запись в таблице ASTE состоит из текущего номера ASTESN, который может использоваться для контроля ссылок на память в зависимых адресных пространствах с помощью регистра доступа или за счет использования следующих команд:

* Команда установки вторичного адресного пространства с переключением пространств SET SECONDARY ASN with space switching (SSAR-ss)
* Команда установки вторичного адресного пространства с переключением пространств SET SECONDARY ASN WITH INSTANCE with space switching (SSAIR-ss)
* Загрузки параметров адресного пространства LOAD ADDRESS SPACE PARAMETERS.

Текущий номер адресного пространства ASTESN может использоваться для контроля связей или для возврата к адресному пространству с помощью следующих команд:

* BRANCH IN SUBSPACE GROUP
* PROGRAM CALL
* PROGRAM TRANSFER
* PROGRAM TRANSFER WITH INSTANCE
* PROGRAM RETURN

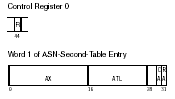
Подробное описание того, как эти команды используют номер ASTESN вы найдете в [1, стр. 3-17].

### 4.2. Начальный номер в таблице ASTE (ASTEIN) и повторное использование номера ASN

В системе может быть установлена функция повторного использования номера ASN и LX. В этом случае запись в таблице ASTE будет содержать еще один номер – ASTEIN и будет иметься четкое описание, относящееся к этому номеру и новый номер связи LSTESN (linkage-second-table-entry). Подробное описание перечисленных компонентов приведено в [1, стр.3-18].

Добавление этой новой возможности в системе вызывает использование следующих трех новых управляющих разрядов (см. рисунок 3):

* Контрольный разряд 44 (R) в управляющем регистре CR0
* Контрольный разряд 30 (CA) в слове 1 в записи таблицы ASTE
* Бит повторного использования 31 (RA) в слове 1 в записи таблицы ASTE



**Рисунок 3 –** Три контрольных разряда ASTE

**Замечания для программистов**: Бит повторного использования в таблице ASTE обеспечивает системе заявленные свойства RAS (Reliability, Availability и Serviceability). Именно введение этого бита позволило повторно использовать номер ASN. Стоит, однако, отметить, что свойство целостности (integrity) в этом случае не обеспечивается, так как номер ASTEIN с регистре общего назначения, использующем команды PROGRAM TRANSFER WITH INSTANCE или SET SECONDARY ASN WITH INSTANCE, которые задает программа. Его обеспечивает индекс авторизации AX, обычно используемый этими командами. Может случиться сбой, если номер ASN авторизован, а используется индекс AX неизмененной авторизации.

## Трансляция ASN

Трансляция ASN – это процесс преобразования 16-ти разрядного номера ASN для указания на таблицу ASTE, определяемой этим номером, в которой хранится код ASCE.

Запуск трансляции номера АП (ASN) осуществляется при смене адресных пространств, задаваемых в следующих командах:

* PROGRAM TRANSFER with space switching (PT-ss)
* PROGRAM TRANSFER WITH INSTANCE with space switching (PTI-ss)
* SET SECONDARY ASN with space switching (SSAR-ss)
* SET SECONDARY ASN WITH INSTANCE with space switching (SSAIR-ss)
* LOAD ADDRESS SPACE PARAMETERS.

Процесс ASN-трансляции одинаков для главного и вторичного адресного режимов, только результаты трансляции используются по-разному. Полученный код ASCE размещается в одном из управляющих регистров в зависимости от типа адресного пространства:

* для Primary Virtual Address в CR1;
* для Secondary Virtual Address в CR7;
* для Home Virtual Address в CR13.

Трансляция ASN может также выполняться, как часть команды PROGRAM RETURN (PR-ss) и команды PROGRAM CALL (PC-ss).

Преобразование ASN в код ASCE, необходимый для активизации адресного пространства в процессоре, выполняется с использованием двух таблиц:

* первичная ASN-таблица (ASN-first-table - AFT);
* вторичная ASN-таблица (ASN-second-table - AST).

Эти таблицы используются для указания на запись в таблице ASTE и на третью таблицу – таблицу авторизации, которая используется, когда выполняется авторизация.

Для целей трансляции 16-ти разрядный номер ASN состоит из двух частей: AFX (ASN-first-table-indeX) - индекс первичной ASN-таблицы состоит из 10-ти левых разрядов номера, и ASX – индекс вторичной таблицы (ASN-Second-table-indeX) состоит из 6-ти правых разрядов номера ASN.

Формат номера ASN показан на рисунке 4.



**Рисунок 4 –** Формат номера адресного пространства ASN

Индекс AFX используется, чтобы выбрать запись из первой таблицы AST (ASN-first-table). Изначально AST задается с помощью указателя ASTO (ASN-first-table-origin) в управляющем регистре CR14.

Сначала выполняется обращение в строку таблицы AFT по адресу, формируемому путем сложения базового адреса AFTO (AFT origin) из управляющего регистра CR14 и первого индекса AFX из кода ASN. В строке AFT задан базовый адрес ASTO (AST origin), который суммируется со вторым индексом ASX из кода ASN для формирования адреса строки ASTE (AST entry), содержащей код ASCE.

Например, в результате трансляции главного PASN во время выполнения команды PROGRAM CALL with space switching адрес, указанный в таблице ASTE, помещается в управляющий регистр CR5, как новый главный указатель PASTEO (primary ASTE origin).

### 5.1. Управление ASN-трансляцией

Управление ASN-трансляцией осуществляется с помощью управляющего разряда “T” и начального указателя AFTO (ASN-first-table-origin), которые размещены в управляющем регистре CR14.

Формат управляющего регистра CR14 приведен на рисунке 5.



**Рисунок 5 –** Формат номера адресного пространства ASN

Разряд 44 в управляющем регистре CR14 – это бит контроля трансляции номера адресного пространства. Этот бит предоставляет механизм, с помощью которого управляющая программа может указать, может ли быть исполнена трансляция ASN во время выполнения какой-либо программы, а также разрешено ли выполнение команды вызова программы с переключением адресного пространства PROGRAM CALL with space switching.

Разряды 45-63 в управляющем регистре CR14 имеют 12 нулей справа, таким образом, формируется 31-разрядный реальный адрес, который определяет начало первой таблицы ASN, которая называется AFT (Address space First Table).

### 5.2. Таблицы ASN-трансляции

Процесс ASN-трансляции состоит в использовании двухуровневых таблиц преобразований:

* первичная ASN-таблица (ASN-first-table - AFT);
* вторичная ASN-таблица (ASN-second-table - AST).

Эти таблицы находятся в основной памяти.

#### Формат записи в таблице AFT

Запись в таблице AFT имеет формат, показанный на рисунке 6.



**Рисунок 6 –** Формат записи в таблице AFT

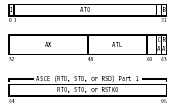
Поля в записи расположены следующим образом:

**Бит ошибки (I):** Разряд 0 обеспечивает проверку, является ли таблица ASTE с записью доступной. Если бит I=0, ASN-трансляция использует указанную таблицу ASTE. Если бит I=1, ASN-трансляция не может быть продолжена.

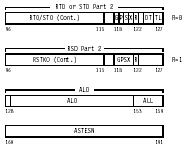
**Указатель начала второй таблицы ASTO** (ASN-Second-Table-Origin): разряды с 1 по 25, имеют справа дополнительные 6 нулей (до 31-го разряда), используются для формирования 31-разрядного реального адреса, указывает на начало второй таблицы AST.

#### Формат записи в таблице AST

Запись в таблице AST имеет длину в 64 байта, причем только первые 48 байт используются. Байты 0-47 имеют формат, показанный на рисунках 7 и 8.



**Рисунок 7 –** Формат первых 12-ти байт записи в таблице AST



**Рисунок 8 –** Формат оставшихся 36-ти байт записи в таблице AST

Поля в байтах 0-47 таблицы AST расположены следующим образом:

**Бит ошибки (I):** Разряд 0 обеспечивает проверку, является ли адресное пространство, ассоциированное с записью в таблице ASTE доступным. Если бит I=0, ASN-трансляция выполняется, если бит I=1, ASN-трансляция не может быть продолжена.

**Указатель начала таблицы авторизации ATO** (Authority-Table -Origin): разряды с 1 по 29 c двумя дополнительными нулями (до 31-го разряда) используются для формирования 31-разрядного реального адреса, который указывает на начало второй таблицы авторизации.

**Бит базового пространства (B):** разряд 31определяет, если В=1, что адресное пространство, связанное с записью в ASTE, является базовым в группе подпространств.

**Индекс авторизации (AX):** разряды с 32 по 47 используются для авторизации номера ASN, чтобы указать на положение битов авторизации в таблице авторизации.

**Длина таблицы авторизации (ATL):** разряды 48-59 определяют длину таблицы авторизации в блоках по четыре байта (словами), делая, таким образом, таблицу авторизации изменяемой одновременно в 16-ти записях.

**Бит контроля ASN (CA):** разряд 62 = 1 – это состояние исключения специальных операций в пользовательском режиме, то есть такие команды как PROGRAM TRANSFER WITH INSTANCE with space switching (PTI-ss) и SET SECONDARY ASN with space switching (SSAR-ss) не выполняются. Этот разряд игнорируется, когда процессор находится в режиме супервизора.

**Бит повторного использования номера ASN (RA):** разряд 63 в единице вынуждает команды PROGRAM TRANSFER with space switching и SET SECONDARY ASN with space switching остановить выполнение и в пользовательском режиме и в режиме супервизора.

**Управляющий элемент (ASCE):** разряды 64-127 – это восьми байтовый код, который может быть указателем на таблицу региона (RTD), на таблицу сегмента (STD) или на реальное адресное пространство (RSD). Внутри этих разрядов имеется целый ряд важных полей, назначения которых будут описаны ниже.

Следующие поля описаны подробно в [1, стр. 3-25]:

Указатель на список доступа (ALD): разряды с 128 по 153.

Номер (ASTESN): разряды с 160 – 191

Указатель на таблицу связей или на первую таблицу связей (LTD) или (LFTD): разряды192-223

Номер (ASTEIN): разряды с 352 – 383

Разряды 224-319 в записи таблицы доступны для программирования.

**Замечание для программистов:** Все неиспользуемые поля в записи таблицы ASTE, включая неиспользуемые поля в байтах 0-31, 40-43 и 48-63 должны быть установлены в 0.

## Трансляция кодов ALET

Трансляция кодов ALET из регистров доступа (AR) в коды ASCE выполняется разными способами в зависимости от значений полей ALET. Номер регистра доступа, из которого берется код ALET, определяется кодом B или R в командах, исполняемых в режиме адресации с адресным пространством, определяемым регистром доступа (см. рисунок 8). Если в команде задан нулевой номер регистра доступа, это соответствует не AR0, а нулевому содержимому AR, то есть, нет необходимости выполнять трансляцию, и код ASCE находится по умолчанию в первом управляющем регистре CR1 (Primary Virtual Address).

Цель такой трансляции: одновременный доступ ко многим адресным пространствам. Это обеспечивается, когда пятый разряд в слове состояния процессора равен единице (PSW.5 = 1), что означает в соответствии с таблицей 1 возможность динамического преобразования адреса (DAT), а разряды 16-17 соответствуют 0 (PSW.16 = 0) и 1 (PSW.17 = 1). Регистр доступа точно такой же, как регистр общего назначение (НЕ регистр индекса), команда смены адресного пространства выглядит следующим образом:

*LG R1,8(R3,R4),*

где четвертый регистр доступа (AR4) используется для трансляции логического адреса 8(R3,R4) в реальный адрес. Если содержимое регистра доступа равно нулю, тогда трансляция идет в режиме адресации главного адресного пространства с помощью первого управляющего регистра (CR1). Если содержимое регистра доступа = 1, тогда трансляция идет в режиме адресации вторичного адресного пространства с помощью седьмого управляющего регистра (CR7). Когда регистр общего назначения равен нулю, это означает отсутствие регистра доступа, который мог бы быть использован (НЕ содержание нулевого регистра доступа).



Рисунок 9 – способы преобразования адресов с использованием регистров доступа

Как видно на рисунке 1, разряд 8 в регистре доступа (бит Р) определяет один из двух способов трансляции кода ALET в управляющий элемент ASCE. При Р=1 используется таблица доступа главного адресного пространства (Primary-space access list), а при Р=0 - таблица доступа (Dispatchable-unit-access list).

Поле ALESN регистра доступа используется для контроля допустимости обращения к таблице доступа.

Поле ALEN задает индекс для обращения в таблицу AST.

При Р=1 преобразование выполняется с использованием таблицы доступа к главному адресному пространству (Primary-space access list). Текущее значение адреса строки для главного виртуального адреса PASTEO (Primary Virtual Address) из пятого управляющего регистра CR5 используется для чтения строки AST (Primary ASTE), в которой задан код ALD, содержащий базовый адрес и длину таблицы доступа Primary-space access list. Базовый адрес суммируется с индексом ALEN из регистра доступа для обращения к строке таблицы доступа Primary-space access list, откуда читается адрес строки AST, содержащей ASCE для виртуального адреса, определенного регистром доступа.

При Р=0 преобразование выполняется с использованием таблицы доступа Dispatchable-unit-access list. В этом случае по коду DUCTO из управляющего регистра CR2 осуществляется обращение в таблицу доступа Dispatchable-unit-control-table, откуда считывается базовый адрес таблицы доступа DUALD, используемый с индексом ALEN для чтения ASCE из таблицы AST (Address Space Secondary Table).

## Формат ASCE

(в разработке)

## Буфер быстрой адресации ALB (Access Lookaside Buffer)

Буфер быстрого доступа ALB (Access Lookaside Buffer) хранит коды ASCE и параметры обращений за операндом.

Поскольку в режиме адресации с адресными пространствами, определяемыми регистром доступа трансляция кода ALET выполняется при каждом обращении за операндом, для ускорения процесса преобразования адреса используется буфер ALB, в котором запоминаются значения параметров, полученных в процессе трансляции.

При последующих обращениях с теми же параметрами код ASCE считывается из таблицы ALB без обращений в другие таблицы.

## Авторизация адресных пространств

Под авторизацией адресных пространств понимается процесс проверки разрешения на открытие конкретного адресного пространства программой с текущим индексом авторизации.

Авторизация или санкционирование адресных пространств выполняется после трансляции задающего его кода ASN или кода ALET с целью проверки допустимости открытия адресного пространства программой, запустившей трансляцию. Общая схема процесса авторизации показана на рисунке 11.

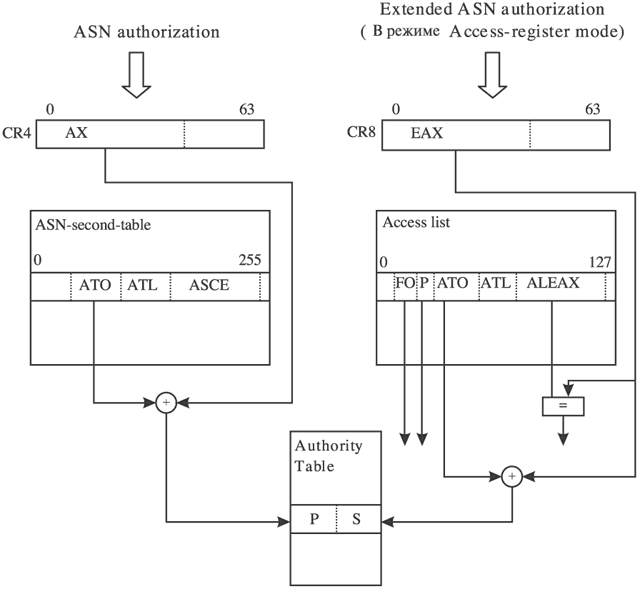


Рисунок 11 – Схема процесса авторизации адресных пространств

Для этого используется специальная таблица авторизации (Authority Table), каждая строка которой содержит два бита разрешения открытия главного (бит P) или вторичного (бит S) адресных пространств.

Выбор строки определяется индексом авторизации (Authorization Index - AX) или индексом расширенной авторизации (Extended Authorization Index - EAX), находящимися при исполнении текущей программы в управляющих регистрах CR4 и CR8 соответственно.

### 9.1. Авторизация с помощью кода ASN

ASN - авторизация выполняется после трансляции кода адресного пространства ASN. На завершающем этапе трансляции после выборки строки из таблицы ASN-second-table, содержащей код управления адресным пространством ASCE, из нее одновременно считываются два поля, определяющие местоположение таблицы авторизации в памяти: поле базового адреса таблицы авторизации (Authority Table Origin - ATO) и поле длины таблицы (Authority Table Length - ATL).

Индекс AX из управляющего регистра CR4 суммируется с базовым адресом ATO для формирования адреса строки таблицы авторизации с контролем границы таблицы по полю ее длины ATL. При единичном значении бита разрешения открытия, считанного из адресуемой строки, допускается открытие соответствующего типа адресного пространства.

### 9.2. Расширенная ASN - авторизация

Расширенная ASN - авторизация выполняется в режиме регистров доступа (Access-register mode), использующем регистры доступа для задания адресных пространств.

Базовый адрес и длина таблицы авторизации берутся из строки таблицы доступа (Access List), в которой задается также адрес строки таблицы ASN-second-table с используемым кодом управления адресным пространством ASCE.

Авторизация при этом выполняется в несколько этапов:

Вначале в выбранной строке таблицы Access List проверяется бит защиты памяти FO, единичное значение которого разрешает только чтение из памяти.

Затем проверяется бит P, управляющий процессом авторизации. При P=0 адресное пространство считается авторизированным независимо от индекса авторизации. При P=1 авторизация выполняется путем сравнения индекса расширенной авторизации EAX из управляющего регистра CR8 и поля индекса ALEAX из строки таблицы доступа. При совпадении использование адресного пространства допустимо, в противном случае выполняется дополнительное обращение в таблицу авторизации с использованием базового адреса и длины этой таблицы из строки таблицы доступа и индекса EAX из CR8.

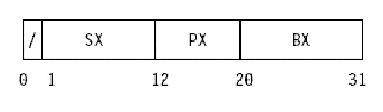
Авторизация завершается успешно при разрешающем значении бита S, считанного из таблицы авторизации.

## Форматы виртуальных адресов

### Формат 31-разрядного виртуального адреса

В архитектуре ESA/390 размер виртуальной памяти измерялся 2 гигабайтами и содержал 1 мегабайт сегментов, а каждый сегмент содержал 4 килобайта страниц.

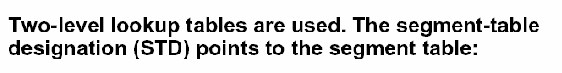
Формат 31-разрядного виртуального адреса для архитектцры S/390 представлен на рисунке 12.

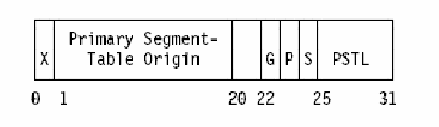


**Рисунок 12 –** формат 31-разрядного виртуального адреса

Виртуальный адрес состоял из следующих полей: индекса сегмента, индекса страницы и базового индекса.

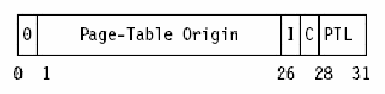
Процесс преобразования адресов был следующий. Использовались двухуровневые таблицы преобразования. Указатель STD (Segment Table Destination) указывал на таблицу сегментов (рисунок 13). Она была расположена в регистрах управления CR1, CR7 и CR13 либо была описана в регистре доступа. По историческим причинам, разряды с 8 по 12 в регистре управления CR0 при этом должны были быть 10110.





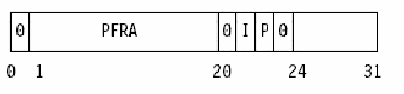
**Рисунок 13 –** формат таблицы сегментов

Поле PSTO таблицы сегментов, в свою очередь, указывало на таблицу страниц (рисунок 14).



**Рисунок 14 –** формат таблицы сегментов

И наконец, поле таблицы страниц PFRA содержало реальный адрес страничного фрейма (рисунок 15).

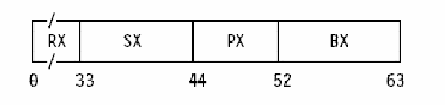


**Рисунок 15 –** формат таблицы страниц

### Формат 64-разрядного виртуального адреса

В архитектуре zArchitecture размер виртуальной памяти измеряется 16-ью экзабайтами и содержит 2 гигабайта регионов, каждый регион содержит 1мегабайт сегментов, а каждый сегмент содержит 4 килобайта страниц.

Формат 64-разрядного виртуального адреса представлен на рисунке 16.



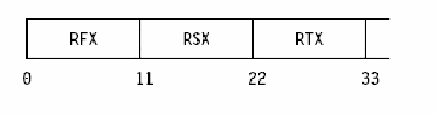
**Рисунок 16 –** формат 64-х разрядного виртуального адреса

Формат виртуального адреса z/Architecture приведен на рисунке 16. Разрядность адреса равна 64, при реализации архитектур с 24- или 31-разрядным адресом старшие биты соответственно 0÷39 или 0÷32 обнуляются. Формат в общем случае включает четыре типа индексов, используемых при обращении в таблицы DAT. Индексы сегмента SX, страницы PX и байта BX находятся в младших 24 или 31 разрядах виртуального адреса и соответствуют сегментно-страничной организации памяти. Эти индексы задают, соответственно, номера строк в сегментной и страничной таблицах, а также адрес байта внутри страницы. Расширенный 64-разрядный формат адреса помимо страниц и сегментов предполагает использование дополнительных уровней представления адресного пространства - регионов памяти. Емкости регионов, сегментов и страниц кратны соответственно 2 GB, 1 MB, 4 KB.

Максимально возможное число регионов - 8 GB. Номер региона, занимающий старшие 33 бита адреса 0÷32, в свою очередь, состоит из трех полей - первого (RFX), второго (RSX) и третьего (RTX) индексов региона.

Использование всех трех индексов допускает объем памяти до 16 EB (Exa Byte = 260 байт), без RFX - до 8 PB (Peta Byte = 250 байт), а без RFX и RSX - до 4 TB (Tera Byte = 240 байт).

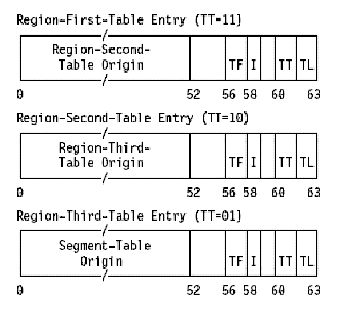
Поскольку виртуальный адрес теперь состоит из шести частей (первый регион, второй регион и третий регион, сегмент, страница, база), требуется пятиуровневая таблица преобразования. Эта таблица напрямую влияет на производительность системы. На сегодня не все уровни таблицы нужны, поскольку даже огромное адресное пространство размером в 4 терабайта может быть управляемо третьим регионом (первый и второй регионы по-прежнему остаются заполненными нулями).



**Рисунок 17 –** формат региона 64-х разрядного виртуального адреса

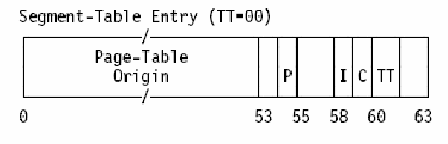
Динамическое преобразование адресов в архитектуре zArchitecture называется динамическим именно потому, что преобразование не обязательно проходит все уровни (все таблицы). Существует механизм, который позволяет определить, на каком уровне надо начать преобразование (первый регион, второй регион и третий регион или сегмент).

Все записи таблиц регионов имеют одинаковый формат, показанный на рисунке 18:



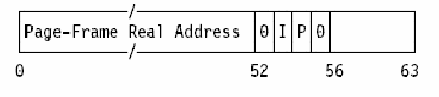
**Рисунок 18** – Формат записи в таблице регионов

Запись в таблице сегментов выглядит, как показано на рисунке 19:



**Рисунок 19** - Запись в таблице сегментов STE

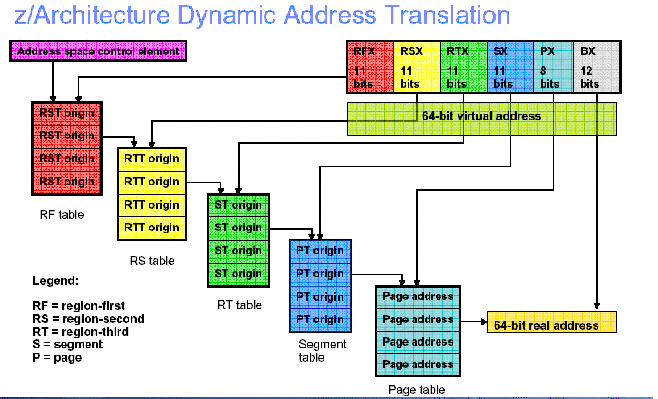
И наконец, на рисунке 20 представлена запись в таблице страниц:



**Рисунок 20** - Запись в таблице страниц

Строка каждой из таблиц содержит идентификатор своего типа (TT) и базовый адрес таблицы следующего уровня (RSTO - для второго уровня регионов, RTTO- для третьего уровня регионов, STO - для сегментной таблицы, PTO - для страничной таблицы) и после суммирования с соответствующим индексом из виртуального адреса определяет строку этой таблицы. Длина следующей таблицы (число блоков) указывается в поле TL. Кроме того, в поле TF указывается смещение начала следующей таблицы относительно конца текущей в тех же единицах измерения. Бит I определяет доступность регионов, сегментов и страниц, охватываемых строкой.

## Динамическое преобразование адресов



**Рисунок 21** – схема динамического преобразования адресов DAT

Полученный в результате трансляции кодов ASN или ALET код ASCE используется для динамического преобразования DAT виртуального адреса в реальный, как это схематично показано на рисунке 13. Справа вверху изображен 64-х разрядный виртуальный адрес, готовый к преобразованию в реальный адрес.

По коду ASCE и индексам виртуального адреса после прохождения пятиуровневых таблиц преобразования получается реальный адрес, показанный на рисунке 21 справа внизу.

Подробная схема процесса DAT представлена на рисунке 22.

Динамическое преобразование адреса начинается с обращения в регионально-сегментные строки буфера быстрой адресации TLB по базовому адресу из ASCE и индексам регионов и сегмента. При наличии совпадения из найденной строки считывается базовый адрес страничной таблицы, в противном случае осуществляется обращение в одну из региональных или сегментную таблицу в зависимости от числа совпавших элементов трассы трансляции. По считанному из строки этой таблицы базовому адресу и соответствующим индексам виртуального адреса выполняется повторное обращение в регионально-сегментные строки буфера быстрой адресации TLB. Такие обращения выполняются до получения базового адреса страничной таблицы, который вместе с индексом страницы из виртуального адреса используется для обращения в страничную строку буфера TLB. При наличии совпадения из информационного поля страничной строки считывается реальный адрес страницы (PFRA).

Если в коде ASCE указан режим реальной адресации, то виртуальный адрес используется как реальный. При этом по указателю реального адресного пространства из ASCE и виртуальному адресу страницы (индексы RFX, RSX, RTX, SX, PX) осуществляется обращение в строки реальных адресных пространств TLB, используемые для контроля допустимости обращения в реальное адресное пространство.

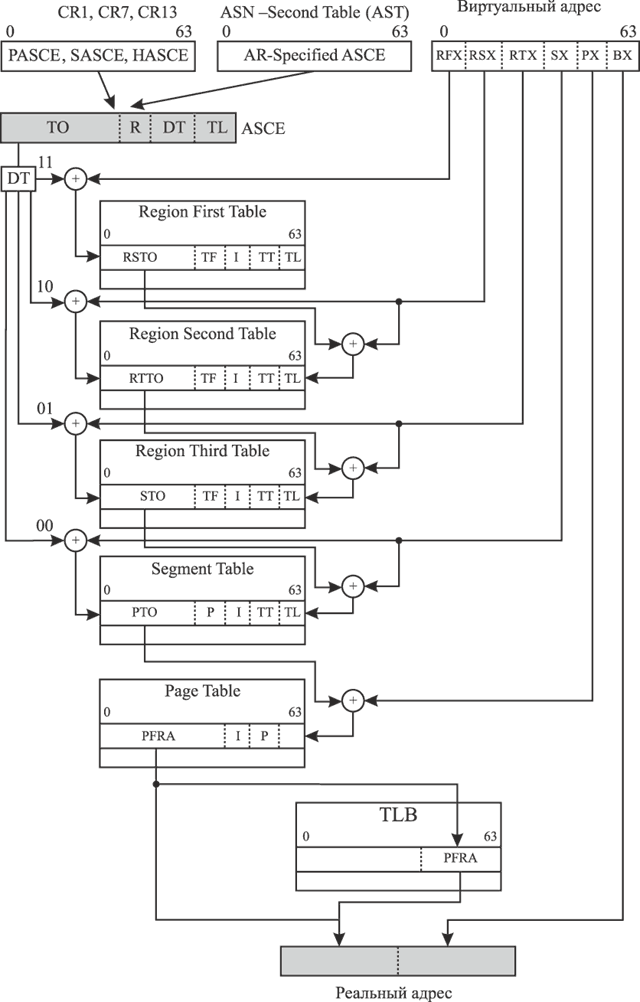


Рисунок 22 – Подробная схема процесса DAT

## Управление числом уровней преобразования

Формат ASCE включает базовый адрес первой таблицы (TO), используемой в преобразовании. Тип этой таблицы, а, следовательно, и число уровней преобразования, определяется полем DT в таблице 2. При DT=11 используются все пять уровней преобразования, и TO в ASCE задает базовый адрес первой региональной таблицы. При DT=00 региональные таблицы не используются, и ASCE задает базовый адрес сегментной таблицы. Кроме того, формат ASCE включает бит R, определяющий тип адресного пространства (реальное или виртуальное), и двухразрядное поле TL, указывающее длину таблицы (в блоках по 4 KB).

**Таблица 2** – Управление DAT

|  |  |  |
| --- | --- | --- |
| **Таблица 2** | | |
| **DT** | **Первая таблица DAT** | **Тип TO в ASCE** | |
| 11 | Первая региональная таблица  (Region First Table) | RSTO | |
| 10 | Вторая региональная таблица  (Region Second Table) | RTTO | |
| 01 | Третья региональная таблица  (Region Third Table) | STO | |
| 00 | Сегментная таблица  (Segment Table) | PTO | |

Поле DT из выбранного ASCE в соответствии с таблицей 2 определяет число уровней преобразования и таблицу, в которую осуществляется первое обращение. Для этого базовый адрес таблицы из поля ASCE суммируется с соответствующим индексом из виртуального адреса. При обращении в первую региональную таблицу используется индекс RFX, во вторую региональную таблицу - индекс RSX, в третью региональную таблицу - индекс RTX и в сегментную таблицу - индекс SX.

При DT=11 последовательно выполняются обращения в первую, вторую и третью региональные таблицы и в сегментную таблицу, при DT=10 - во вторую и третью региональные таблицы и в сегментную таблицу, при DT=01 - в третью региональную таблицу и в сегментную таблицу, а при DT=00 обращение осуществляется сразу в сегментную таблицу с пропуском всех региональных таблиц. Все старшие неиспользуемые индексы, соответствующие пропущенным таблицам, должны быть нулевыми, в противном случае преобразование прерывается. Контроль обращений осуществляется с учетом длины таблиц и пропусков между ними, заданных кодом ASCE и в строках таблиц, для исключения обращений за их пределы.

## Буфер быстрой переадресации TLB

Поскольку все таблицы находятся в реальной памяти, и в процессе DAT может потребоваться до пяти обращений в таблицы, время преобразования может быть недопустимо большим. Для его уменьшения в состав устройства DAT вводится буфер быстрой переадресации TLB (Translation-Lookaside Buffer). В литературе такой буфер иногда называют кэш-памятью адресов. В мультипроцессорных реализациях каждый процессор имеет собственный буфер TLB. Процессор обращается в таблицы DAT только при первом преобразовании виртуального адреса. Полученные из таблиц строки вместе с исходными атрибутами преобразования (базовые адреса, индексы, параметры) запоминаются в строках TLB. При последующих обращениях с теми же атрибутами выполняется обращение в TLB без обращений в память.

Поскольку строки таблиц DAT, размещенных в памяти, и соответствующие им строки TLB должны в процессе трансляции формировать одни и те же ссылки, содержимое этих строк должно выравниваться. С этой целью для строк таблиц DAT, хранящихся в памяти, вводятся два типа состояний: присоединенное (attached) и доступное (valid). Строка считается присоединенной к конкретному процессору, если он может использовать ее для DAT. Одновременно строка может быть присоединенной для нескольких процессоров. Строка является доступной, если регион, сегмент или страница, связанные с этой строкой, имеются в памяти (установлен бит доступности этих блоков в соответствующих строках).

Каждая строка TLB состоит из двух частей: поле атрибутов для поиска строки, соответствующей текущим атрибутам DAT (ассоциативное поле) и поле результата преобразования (информационное поле).

В буфере быстрой адресации TLB предусмотрено использование трех типов строк:

**Комбинированные регионально-сегментные строки** в ассоциативном поле содержат базовые адреса региональных (если регионы используются) и сегментной таблиц, использовавшихся при выполнении DAT; цепочка таких адресов называется трассой трансляции. Начало трассы задается в ASCE. Трасса может быть размещена в строке TLB, если все строки таблиц DAT, содержащие элементы трассы, присоединены и доступны. Помимо трассы, ассоциативное поле включает индексы регионов (RFX, RSX, RTX) и сегмента (SX) виртуального адреса. Информационное поле строки содержит базовый адрес страничной таблицы, полученный в результате преобразования.

**Страничные строки** в ассоциативном поле содержат базовый адрес страничной таблицы, который получен из присоединенной и доступной строки таблицы DAT, и индекс страницы (PX) из виртуального адреса. В информационном поле строки содержится реальный адрес страницы PFRA, который и является конечным результатом динамического преобразования адреса.

**Строки реальных адресных пространств** включают ассоциативное поле, состоящее из указателя реального адресного пространства Real-space Token Origin (входит в соответствующее ASCE при установленном бите R, задающем реальное адресное пространство), и индексы RFX, RSX, RTX, SX, PX, входящие в виртуальный адрес. Такие строки используются не для DAT, а для контроля допустимости обращения в реальное адресное пространство.

## Краткие выводы по главе

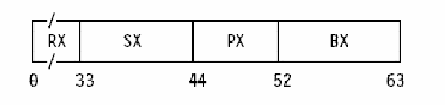
Оглянемся назад? 

1. Два способа задания адресных пространств:

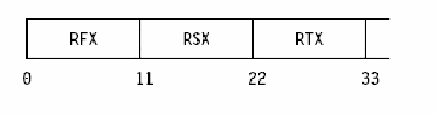
* С помощью номера адресного пространства ASN
* С помощью регистра доступа.

1. Авторизация адресного пространства – это процесс проверки разрешения на открытие программой адресного пространства
2. Формат виртуального адреса:

* включает четыре типа индексов: региона RX, сегмента SX, страницы PX и базы BX



* регион RX разделяется на три компонента: первый регион RFX, второй регион RSX и третий регион RTX.



1. Типы строк в буфере быстрой адресации:
   * + 1. Комбинированные регионально-сегментные строки**;**
       2. Страничные строки;
       3. Строки реальных адресных пространств**.**

## Вопросы для контроля усвоения материала

* Какие разряды в слове PSW отвечают за режим адресации?
* Что такое динамическое преобразование адресов?
* Что такое ASN-трансляция ?
* Какие существуют способы задания адресных пространств?
* Как создаются адресные пространства?
* Как процессор переходит из одного режима адресации в другой?

## Глава 9 Подсистема ввода-вывода мейнфреймов zSeries

Что нас ждет? BD04972_

После прочтения этой главы Вы сможете:

1. назвать основные принципы работы подсистемы ввода-вывода в современной большой ЭВМ;
2. перечислить основные компоненты подсистемы ввода-вывода большой машины;
3. объяснить механизм создания множественных образов каналов и канальных путей;
4. перечислить типы адресации каналов ввода-вывода;
5. назвать способы выделения каналов ввода-вывода для логических разделов;
6. объяснить, как идентифицируются объекты канальной системы.

## Основные принципы работы подсистемы ввода-вывода в большой ЭВМ

Современная большая машина класса IBM мейнфрейм обслуживает одновременно сотни тысяч и тысячи тысяч пользователей. Для выполнения операций обмена информацией между таким количеством пользователей и внутренней памятью машины должны быть предусмотрены процедуры настройки и управления самыми разными периферийными устройствами. Это подразумевает наличие современных интерфейсов в таком количестве, чтобы обеспечить *совместимость* большой ЭВМ с любым существующим периферийным устройством, включая такие стандартные устройства, как принтеры, плоттеры, модемы, сканнеры и так далее, а также специальные устройства, разработанные для конкретных целей предприятия, такие как адаптеры для ввода технологической информации (уровень газа в воздухе, давление в вакуумной камере и так далее). Это подразумевает быструю связь за счет *многоуровневого подключения* к большой машине, причем уровень выбирается автоматически в зависимости от быстродействия подключаемого периферийного устройства. Это подразумевает также *надежность обслуживания*, что обеспечивается специальным *механизмом* *создания множественных образов каналов и канальных путей* с четко определенной системой идентификации объектов внутри каждой канальной системы, которая обеспечивает *полную изоляцию пользователей* одних канальных путей от других.

Подсистема ввода-вывода со своими специализированными процессорами ввода-вывода освобождает центральные процессоры от рутинных операций ввода-вывода и функционирует параллельно с ними, что позволяет существенно увеличить как производительность сервера, так и пропускную способность системы ввода-вывода. Реализация функций подсистемы ввода-вывода осуществляется на *сервисных процессорах SAP* (System Assist Processor).

Высокая пропускная способность подсистемы ввода-вывода у серверов zSeries во многом определяется высокой степенью *распараллеливания работы* большого числа периферийных устройств. Такое распараллеливание достигается за счет использования в подсистеме ввода-вывода *большого числа* параллельно функционирующих *каналов*, в каждом из которых выполняется одна или несколько операций обмена информацией с периферийными устройствами.

Каждой операции ввода-вывода, реализующей обмен информацией между периферийным устройством и основной памятью, соответствует *канальная программа* (Channel Program), определяющая параметры обмена: приказы для периферийных устройств, количество передаваемых данных, используемые адреса основной памяти и др.

Для *автономного исполнения канальных программ* используются SAP-процессоры со специализированной системой команд. Центральные процессоры осуществляют только запуск этих операций одной из команд ввода-вывода с последующим минимальным контролем хода их исполнения через систему прерываний.

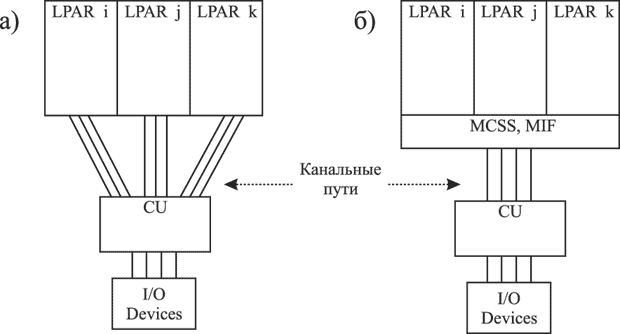
## Компоненты подсистемы ввода-вывода в большой ЭВМ

Подсистема ввода-вывода состоит из следующих компонентов.

**Периферийные устройства (I/O Devices)**, к которым относятся внешние запоминающие устройства и устройства ввода-вывода информации. Каждое периферийное устройство имеет свой номер в подсистеме ввода-вывода, который задается 16-разрядным кодом, соответственно допускается использование до 65536 периферийных устройств.

**Контрольные блоки (Control Unit),** называемые также устройствами управления или контроллерами периферийных устройств, обеспечивают адаптацию типовых процедур управления подсистемы ввода-вывода CSS к конкретным особенностям периферийных устройств. Контрольный блок CU принимает команды от подсистемы ввода-вывода CSS, дешифрирует их и вырабатывает последовательность управляющих сигналов для периферийного устройства, необходимую для выполнения требуемой операции. Контрольный блок CU может быть выполнен в виде отдельного устройства или встроен физически и логически в периферийное устройство, в подсистему ввода-вывода CSS.

**Канальные пути (Channel Path)** или каналы, каждый из которых является интерфейсом для обмена информацией между сервером и одним или несколькими контрольными блоками (CU). Через такой интерфейс передаются команды, состояния и данные, необходимые для операций ввода-вывода. Один контрольный блок CU может быть подключен к нескольким канальным путям, и одно периферийное устройство может быть связано с несколькими контрольными блоками CU. Число канальных путей для связи с одним периферийным устройством может достигать восьми, устройство может использовать любой из этих путей. Каждому канальному пути присваивается уникальный идентификатор (Channel Path Identifier - CHPID). Одна подсистема ввода-вывода поддерживает до 256 идентификаторов CHPID.



**Рисунок 1** Два варианта исполнения подсистемы ввода-вывода

**Подканалы (Subchannels)** предназначены для хранения управляющей информации об одной операции ввода-вывода в CSS в течение всего времени ее исполнения. Подканал выделяется каждому подключенному к подсистеме ввода-вывода периферийному устройству на время выполнения операции ввода-вывода и представляет собой фрагмент внутренней памяти, в котором хранится информация об операции ввода-вывода: адрес команды ввода-вывода, идентификатор канального пути CHPID, номер периферийного устройства, счетчик данных, состояния и другие данные. Каждому периферийному устройству подсистемы ввода-вывода CSS, доступному из программ, соответствует выделенный подканал. Число подканалов зависит от модели системы и может достигать 65536, если каждый подканал адресуется 16-разрядным кодом.

Мы рассмотрели построение подсистемы ввода-вывода, соответствующее рисунку 1.а), на рисунке 4.1.б) введен еще один компонент – **множественная канальная подсистема,** представляющая собой от одной до 256-ти логических канальных подсистем ввода-вывода.

В ранних моделях больших ЭВМ распределение компонентов было фиксированным, и допускалась лишь его ручная реконфигурация (рисунок 4.1а). В современных моделях компоненты могут совместно использоваться несколькими логическими разделами (рис. 4.1б) без ручной реконфигурации путем использования функций множественной канальной подсистемы MCSS (Multiple LCSS - MCSS) и механизма создания множественных образов канальной подсистемы MIF (Multiple Image Facility).

Множественная канальная подсистема MCSS состоит из одной или нескольких (до 256) логических канальных подсистем (Logical CSS - LCSS). Внутри каждой логической LCSS полностью поддерживается архитектура и возможности одной канальной подсистемы, что позволяет сохранить программную совместимость с предыдущими моделями серверов. Каждая из LCSS имеет свой идентификатор - CSSID, с помощью которого управляющая программа компьютера (hypervisor) реализует полное адресное пространство подсистемы ввода-вывода. В моделях больших ЭВМ серий z990, z890 используются две логических подсистемы ввода-вывода LCSS, что расширяет число канальных путей до 512.

Перечисленные компоненты подсистемы ввода-вывода CSS (для варианта на рисунке 4.1а) или множественной канальной подсистемы MCSS (для варианта на рисунке 4.1б) подключаются к центральному электронному комплексу CEC (central electronical complex).

## Механизм создания множественных образов канальной подсистемы

Общая организация канальной подсистемы современной большой ЭВМ класса мейнфрейм приведена на рисунке 2. К каждой логической подсистеме ввода-вывода LCSS, входящей в состав множественной подсистемы ввода-вывода MCSS, могут быть подключены один или более (вплоть до 15-ти) логических разделов LPAR1 - LPAR15, при этом один логический раздел LPAR0 используется управляющей программой (Control Program CP или hypervisor). С другой стороны, каждый из логических разделов LPAR соотносится с одной из LCSS и использует те канальные пути, которые выделены внутри логической канальной подсистемы LCSS для этого раздела.

В каждом отдельном логическом разделе LPAR может быть запущена собственная операционная система, которая будет считать, что она безраздельно владеет всей подсистемой ввода-вывода, то есть, должны быть реализованы полные архитектурных возможностей отдельной канальной подсистемы.

С этой целью и применяется механизм создания множественных образов канальной подсистемы Multiple Image Facility, с помощью которого для каждого логического раздела LPAR создаются MIF-образы полного набора из 64К подканалов Subchannel Images (SCI) и 256 канальных путей Channel Path Images (CPI). Совокупность всех образов подканалов SCI и образов канальных путей CPI для одного логического раздела LPAR называется MIF-образом раздела.

Совокупность MIF-образов всех разделов LCSS образует MIF-образ канальной подсистемы. Внутри каждой LCSS каждому MIF-образу присваивается идентификатор (IID) в диапазоне 1-15.

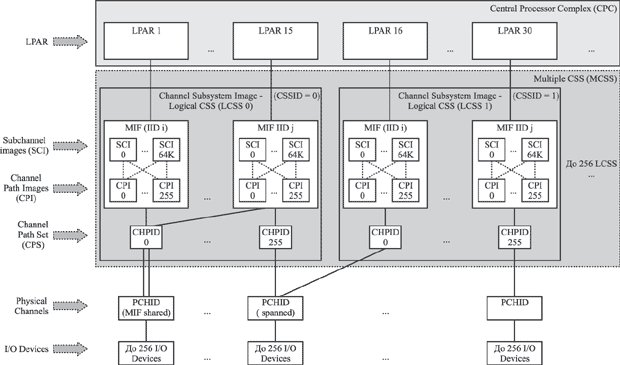


Рисунок 2 – Общая архитектура подсистемы ввода-вывода и идентификация подканалов

Программы, исполняемые в логическом разделе LPAR, оперируют образами подканалов SCI и канальных путей CPI. Образы канальных путей, используемые в LPAR внутри каждой LCSS, накладываются на множество из 256 канальных путей LCSS (Channel Path Set - CPS), каждый из которых имеет свой идентификатор CHPID в диапазоне 0-255. Канальные пути в разных логических канальных подсистемах LCSS могут иметь одинаковые идентификаторы CHPID. Соответствие канальных путей LCSS и их MIF-образов поддерживается управляющей программой (hypervisor) на уровне милликодов. Канальный путь логической канальной подсистемы LCSS, которому соответствуют несколько MIF-образов канальных путей, называется MIF-совмещенным (MIF shared).

Множество канальных путей всех логических канальных подсистем LCSS накладывается на множество физических канальных путей. Каждый физический канальный путь имеет уникальный идентификатор (Physical Channel ID - PCHID), соответствующий физическому интерфейсу и элементам его конструктивного исполнения: каркасу ввода-вывода, слоту, порту. Для идентификаторов CHPID устанавливается соответствие идентификаторам PCHID, поддерживаемое на уровне управляющей программы (hypervisor).

Каждой логической канальной подсистеме LCSS соответствует группа физических каналов, принадлежащих только этой подсистеме, кроме объединенных каналов (Spanning Channel). Объединенные каналы могут использоваться логическими разделами разных логических канальных подсистем (LCSS). Канальный путь считается объединенным, если один и тот же физический канальный путь (PCHID) соответствует канальным путям (CHPID) в разных логических канальных подсистемах (LCSS). Один физический канальный путь доступен всем логическим разделам LPAR и всем логическим канальным подсистемам LCSS.

В случае объединения внутренних каналов вместо физического канала может применяться виртуальный канал (Virtual Channel) с идентификатором VCHID.

Виртуальные каналы реализованы на уровне милликодов и позволяют реализовать обмен между логическими разделами без использования физических каналов.

## Адресация в подсистеме ввода-вывода

Какое назначение имеет адресация? Каждое подключаемое к машине периферийное устройство должно иметь возможность обмениваться информацией с машиной. Периферийных устройств, подключаемых к большой ЭВМ тысячи тысяч, как в них не запутаться? Надо задать каждому свой собственный адрес.

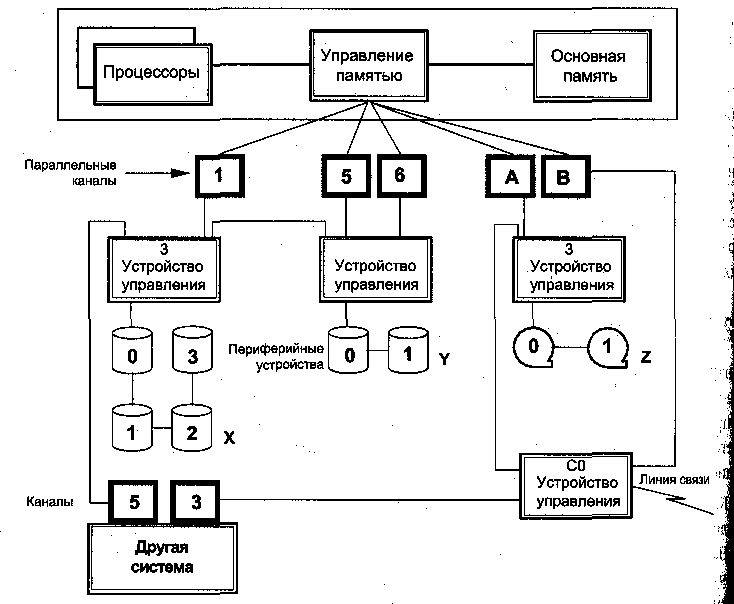
Преподавателю: здесь уместно привести аналогию с почтовыми адресами людей (ФИО, номер квартиры, номер дома, название улицы, название города, название страны) или телефонными номерами (код страны, код города, код района).

Аналогично почтовой адресации для людей, при подключении периферийного устройства раньше вручную, а на современных машинах автоматически, ему присваивается идентификатор.

В подсистеме ввода-вывода CSS в зависимости от типа канальной подсистемы существуют четыре типа адресации:

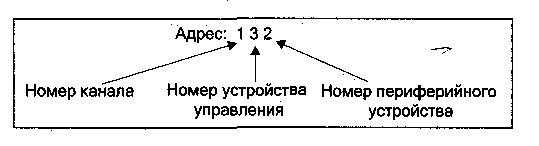
* идентификация канальных путей,
* нумерация подканалов,
* нумерация периферийных устройств и
* адресация, независимая от типа канала (хотя она и невидима для программ).

Каждый канал, устройство управления и периферийное устройства имеют адрес, выраженный шестнадцатеричным числом. Принципы адресации пояснены рисунками 3 и 4.



**Рисунок 3** Общая схема архитектуры S/360

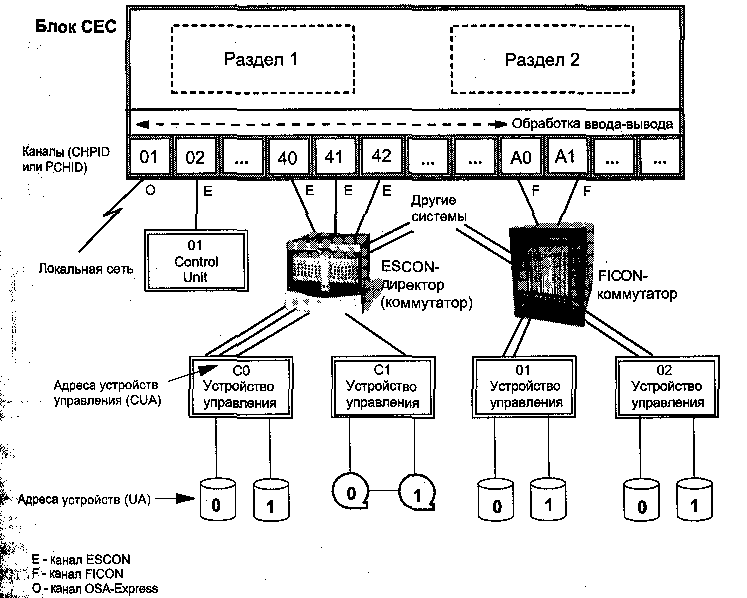
Дисковый привод, обозначенный на рисунке 3 буквой Х, имеет адрес 132, полученный способом, показанным на рисунке 4.



**Рисунок 4** Адрес устройства для архитектуры S/360

Дисковому приводу, обозначенному на рисунке 3 буквой Y, можно назначить несколько адресов: 171, 571 или 671, так как он подключен через три канала. Все три адреса могут использоваться операционной системой для доступа к периферийному устройству. Наличие нескольких путей полезно в целях обеспечения повышения производительности, надежности и доступности.

На рисунке 3 показана условно и другая вычислительная система (слева внизу), у которой два канала (5 и 3) подключены к контрольным блокам (устройствам управления), используемым первой системой. Такое совместное использование периферийных устройств характерно во всех инсталляциях мейнфреймов. Привод для носителей на магнитной ленте Z имеет адрес А31 в первой системе и адрес 331 во второй системе.



**Рисунок 5** Общая схема архитектуры zArchitecture

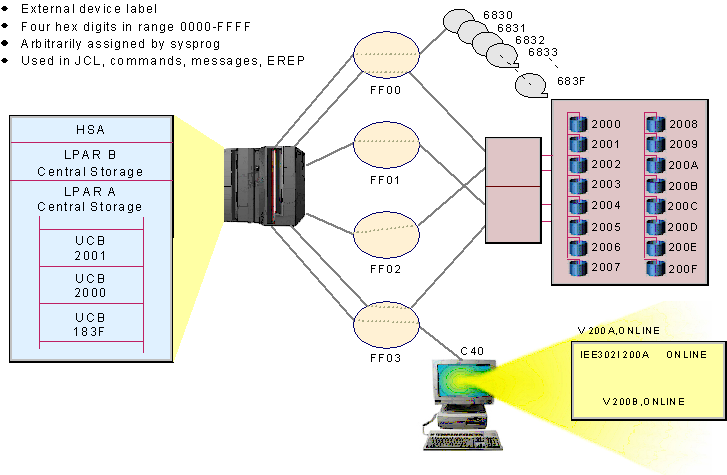
На рисунке 5 представлена современная схема архитектуры zArchitecture. Реальные мейнфреймы имеют большее число каналов и периферийных устройств, однако, этот рисунок иллюстрирует основные понятия.

Номер периферийного устройства выглядит как адрес, описанный для машин версии S/360, но теперь он может содержать три или четыре шестнадцатеричные цифры, как это показано на рисунке 6.

Полное адресное пространство подсистемы ввода-вывода складывается из системы идентификаторов логических подсистем ввода-вывода, идентификаторов канальных путей и номеров периферийных устройств.

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Обозначение идентификатора** | **Диапазон значений** |
| Логический раздел | LPAR | 1,2,..,30 |
| Канальная подсистема | CSSID | 0,1 ? |
| Логическая канальная подсистема | LCSS | 0,1 |
| Образ подсистемы | IID | SCI\*CPI |
| Образ подканала | SCI | 0,1,2,…, 65536 |
| Образ канального пути | CPI | 0,1,2,…,255 |
| Канальный путь | CHPID (две шестнадцатеричные цифры [3, стр. 40]) | 0,1,2,…, 255 Восьмиразрядное число (28)? |
| Физический канал | PCHID |  |
| Периферийное устройство |  | 0,1,…255 |

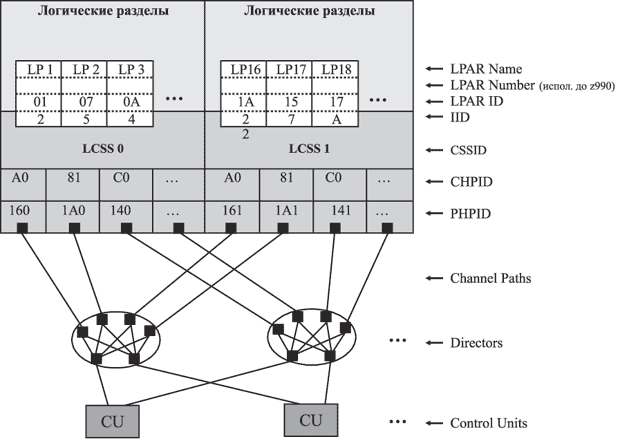
В таблице приведены идентификаторы компонентов подсистемы ввода-вывода и диапазон их значений.



**Рисунок 6** – адресация периферийных устройств

### 4.1. Идентификация объектов канальной подсистемы

Чтобы разобраться с тем, как организована адресация в подсистеме ввода-вывода, сначала рассмотрим принципы идентификации объектов канальной подсистемы. Общая структура идентификации объектов, используемая при адресации в канальной подсистеме, показана на рисунке 7.



**Рисунок 7** Структура идентификации объектов канальной подсистемы

Каждый логический раздел имеет уникальное для всего сервера имя (LPAR Name), идентификатор (LPAR ID) и номер (LPAR Number), использующийся в моделях, предшествующих z990. Кроме того, для каждого логического раздела LPAR задается MIF-образ подсистемы ввода-вывода своим идентификатором IID. В каждом MIF-образе логической подсистемы ввода-вывода (LCSS) используется полное пространство идентификаторов образов подканалов SCI и канальных путей CPI. Переход от этих образов к физическим канальным путям PHPID осуществляется с использованием идентификаторов MIF-образа канальной системы IID, CSSID и физического канального пути CHPID. Применение многопортовых коммутаторов (Directors) позволяет создавать несколько альтернативных канальных путей к одному контрольному блоку CU.

#### Идентификация канальных путей

Идентификатор канального пути CHPID - это уникальное для системы восьмиразрядное число, присваиваемое каждому установленному в системе канальному пути. Идентификатор CHPID используется для адресации канального пути, он определяется вторым операндом адреса команды RESET CHANNEL PATH и используется для задания того канального пути, который должен быть переустановлен (RESET). Канальные пути, с помощью которых подсоединяется периферийное устройство, идентифицируются в информационном блоке подканала SCHIB, каждый с помощью своего присвоенного CHPID, во время выполнения команды STORE SUBCHANNEL. Кроме того, идентификатор CHPID может также быть использован в сообщениях оператора, когда необходимо идентифицировать определенный канальный путь. Системная модель может предоставить до 256-ти канальных путей. Максимальное число канальных путей и присвоенных этим канальным путям идентификаторов зависит от конкретной модели системы.

#### Нумерация подканалов

Номер подканала - это уникальное для системы 16-ти разрядное число, используемое для адресации подканала. Это число уникальное (единственное) в канальной подсистеме. Адресация в подсистеме осуществляется с помощью восьми команд ввода-вывода: CANCEL SUBCHANNEL, CLEAR SUBCHANNEL, HALT SUBCHANNEL, MODIFY SUBCHANNEL, RESUME SUBCHANNEL, START SUBCHANNEL, STORE SUBCHANNEL, TEST SUBCHANNEL. Все функции ввода-вывода зависят от специфики конкретного периферийного устройства, определяются программой, которая назначает подканал для конкретного устройства ввода-вывода.

Подканалы всегда определятся номерами подканалов внутри одного диапазона непрерывных номеров. Самый низкий номер подканала - это подканал номер 0. Самый высокий номер подканала канальной подсистемы имеет подканальный номер, на единицу меньший, чем номер предоставляемого канала. Максимально может быть предоставлено 65536 подканалов. В нормальных условиях номера подканалов используются только в обмене между программой центрального процессора и канальной подсистемой.

#### Нумерация периферийных устройств

Номер устройства - это 16-ти разрядное число, которое определено как один из параметров подканала на то время, когда устройство закреплено за подканалом (присвоено подканалу). Номер устройства однозначно определяет (идентифицирует) устройство для программы. Каждый подканал, который имеет соответствующее ему присвоенное устройство ввода-вывода, а также содержит параметр, называемый номером устройства.

Номер устройства предоставляет средства идентификации этого устройства независимо от каких-либо ограничений, накладываемых моделью системы, конфигурацией машины или протоколами канальных путей. Номер устройства используется как средство связи, когда это касается устройства при обмене информацией между системным оператором и системой. Например, номер устройства вводится системным оператором, чтобы определить устройство ввода, с которого будет осуществлена начальная загрузка программы. Номер устройства однозначно обозначает устройство для программы.

*Замечания для программиста*: Номер устройства назначается на время установки устройства и может иметь любое значение. Однако пользователь может встретить различные запреты на значения номера устройства, которые могут ввести управляющая программа операционной системы, программы поддержки, конкретное устройство управления контрольный блок (control unit) или даже само периферийное устройство.

#### Идентификация периферийных устройств

Идентификатор периферийного устройства – это адрес, не видимый программой (not apparent to the program). Он используется канальной подсистемой для ее общения с периферийными устройствами. Тип используемого идентификатора периферийного устройства зависит от специфики канального пути и от применяемого протокола. Каждый подканал имеет один или несколько идентификаторов периферийного устройства.

Адрес устройства определяет конкретное устройство ввода-вывода и, если используется параллельный интерфейс ввода-вывода, то еще и конкретный блок управления (CU), ассоциированный с подканалом. Тогда идентификатор ПУ называется адресом устройства и адрес является 8-миразрядным числом.

Следует иметь в виду, что адрес периферийного устройства и, соответственно, размер идентификатора периферийного устройства, зависят от того, какой тип канального пути используется.

В частности, для канального пути с параллельным типом интерфейса ввода-вывода идентификатор устройства называется адресом устройства и представляет собой 8-ми разрядное число.

Для интерфейса ввода-вывода последовательного типа (ESCON) идентификатор устройства содержит четырехразрядный адрес устройства контроля (CU) и восьми разрядный адрес устройства, то есть, представляет собой 12-ти разрядное число.

Для интерфейса ввода-вывода типа FICON идентификатор устройства состоит из 8-ми разрядного идентификатора образа контрольного блока (устройства управления - control unit) и 8-ми разрядного адреса устройства, то есть, представляет собой 16-ти разрядное число.

Для интерфейса ввода-вывода типа FICON-converted идентификатор устройства состоит из 4-х разрядного адреса устройства управления (контрольного блока) и 8-ми разрядного адреса периферийного устройства, то есть, представляет собой 12-ти разрядное число.

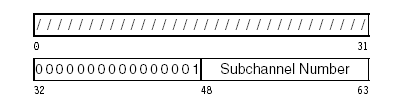
Более подробно об идентификации периферийных устройств с учетом определенного типа канального пути можно найти в соответствующих публикациях [3, 4].

#### Идентификация подсистемы ввода-вывода

Для преобразования номера подканала в адрес памяти, где фактически размещен подканал, управляющая программа (hypervisor) формирует идентификатор подсистемы - Subsystem Identification Word (SID).

Номер подканала, как правило, берется из команды ввода-вывода. Помимо номера подканала в идентификатор подсистемы SID включены также идентификатор используемой логической подсистемы ввода-вывода (LCSS) CSSID и идентификатор MIF-образа раздела IID.

Все команды ввода-вывода, в которых есть ссылки на подканал, используют содержимое регистра общего назначения GR1, в котором содержится идентификатор подсистемы SID (subsystem identification word). Формат SID представлен на рисунке 4.4.



**Рисунок 8 -** Формат идентификатора подсистемы ввода-вывода SID

### Управление назначением канальных путей

Управление назначением канальных путей в современных мейнфреймах осуществляется с помощью специально созданного для этого регистра CCW (Channel Command Word) – слова состояния канала.

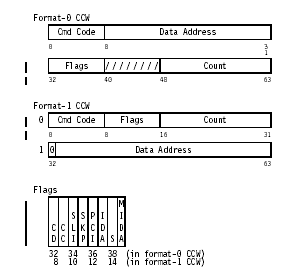
#### Слово состояния канала CCW

Слово состояния канала CCW определяет команду, которая должна быть выполнена и для команд, инициирующих определенные операции ввода-вывода, оно определяет область памяти, связанную с этими операциями, а также действия, которые должны быть предприняты, когда область памяти будет заполнена или когда она должна быть перенесена или другие действия.

Канальная программа состоит из одного или более слов состояния канала CCW, которые логически связаны между собой так, что они выбираются и выполняются канальной подсистемой либо последовательно, либо в другом порядке. Последовательные слова CCW соединены с использованием цепочек и флагов команд цепочек, а непоследовательные слова CCW соединяются с помощью команды передачи в канале «transfer-in-channel».

Положение первого слова CCW канальной программы определяет блок ORB, который является операндом команды START SUBCHANNEL. То, какой из двух форматов имеет слово CCW, определяет разряд 8 слова 1 блока ORB. Каждое дополнительное слово CCW вызывается канальной программой по мере необходимости. Выбор слов CCW канальной подсистемой не влияет на их размещение в основной па

Слово состояния канала имеет два формата – формат 0 и формат 1. Они различаются только размером адреса и размещением полей в слове CCW (рисунок 9).



**Рисунок 9 -** Формат слова состояния канала CCW

Формат-0 слова CCW может быть помещен в любом месте первых 224 (16 мегабит) байт абсолютной памяти, а формат-1 слова могут быть помещен в любом месте первых 231 (2 GB) байт абсолютной памяти.

#### Формат команды ввода-вывода

Команды ввода-вывода используют формат S, показанный на рисунке 10.



**Рисунок 10 -** Формат команды ввода-вывода

В зависимости от того, какая именно команда ввода-вывода будет выполняться, будет зависеть, где находится адрес второго операнда – в регистрах общего назначения GR1 и GR2. Однако, все команды ввода-вывода, которые ссылаются на подканал, используют содержимое регистра GR1, как на встроенный операнд. Для этих команд регистр GR1 содержит идентификатор подсистемы SID, формат которого приведен в п. 4.1.5.

#### Особенности выполнения операций ввода-вывода

Порядок доступа к полям операнда и к полям подканала – непредсказуем.

Как правило, результат выполнения команды ввода-вывода – это выставление кода условия вставляется в слово PSW. Существуют 4 типа кода условия, которые показаны в таблице.

|  |  |
| --- | --- |
| **Код условия** | **Результат выполнения команды ввода-вывода** |
| **0** | Ожидаемый или наиболее вероятный |
| **1** | Альтернативный или наиболее вероятный второй возможный результат |
| **2** | Неэффективный, так как канал был занят |
| **3** | Неэффективный, так как команда не выполнима |

#### Назначение канальных путей

Если в первом слове состояния канала CCW проверка правильности (validly test) проходит успешно, и не установлен флаг задержки 6-1, тогда канальная подсистема начинает выбор периферийного устройства, сначала выбирая канальный путь из группы доступных для выбора канальных путей. Контрольный блок, который определяет идентификатор устройства, подсоединяет себя логически к канальному пути.

Канальная подсистема посылает часть командного кода из слова состояния канала CCW в канальный путь и периферийное устройство отвечает статусным байтом, показывая, может ли быть выполнена команда. Контрольный блок CU может логически отключиться от канального пути в это время, а может остаться подключенным и начать передачу данных.

Если выбрать устройство не удалось из-за индикатора занятости периферийного устройства или из-за условия «путь не работает», тогда канальная подсистема начинает выбор периферийного устройства в альтернативном канальном пути, если такой есть. Если во всех доступных путях, которые были попытки выбора, везде был отказ индикатора занятости, операция выбора остается в состоянии ожидания, пока какой-либо путь не освободится. Если условие «путь не работает» обнаружено в одном или нескольких канальных путях, в которых предпринималась попытка выбора, программа получит сообщение с помощью последующего прерывания ввода-вывода.

## Три режима обмена информацией: frame-multiplex mode, burst mode, or byte-multiplex

При выполнении обменов между периферийными устройствами и логическим разделом LPAR (а точнее, памятью, отнесенной к LPAR) используются три режима: фрейм-мультиплексный, байт-мультиплексный режим и режим пакетной передачи.

Во фрейм-мультиплексном режиме устройство ввода-вывода может оставаться логически подключенным к канальному пути на время выполнения канальной программы. Возможности канального пути к выполнению операций во фрейм-мультиплексном режиме могут быть распределены между несколькими одновременно работающими устройствами ввода-вывода. В этом режиме информация, необходимая для того, чтобы завершить операцию ввода-вывода, поделена на фреймы так, что она может быть interleaved с фреймами, полученными от операций ввода-вывода с другими устройствами ввода-вывода.

Фреймы, помимо собственно управляющей информации и данных, включают в себя адреса источника (Source Logical Address - SLA) и приемника (Destination Logical Address - DLA). Это позволяет передавать информацию между разными источниками и приемниками по одним и тем же физическим канальным путям.

Во время пакетной передачи некоторые канальные пути могут быть чувствительными к отсутствию передачи данных, например, в течение полуминуты. Такое случается, когда имеется промежуток на читаемой магнитной ленте может быть зарегистрирован сбой работы оборудования, когда отсутствует передача данных в течение заранее установленного времени.

В байт-мультиплексном режиме периферийное устройство остается логически подключенным к канальному пути только на короткий интервал времени. Возможности канального пути к работе в байт-мультиплексном режиме могут быть поделены между несколькими одновременно работающими периферийными устройствами. В этом режиме все операции ввода-вывода поделены на короткие интервалы времени, во время которых по канальному пути передается только сегмент информации. Во время этого интервала времени только одно ПУ и ассоциированный с ним подканал логически подсоединены к канальному пути. Интервалы, ассоциированные с одновременными действиямимножества ПУ выстроены в последовательность в соответствии с запросами от ПУ.

Возможности канальной подсистемы, ассоциированные с подканалом Возможности канальной подсистемы, ассоциированные с подканалом, выполняют свое управление только одной операции только во время, требуемое для передачи сегмента информации. Сегмент может состоять из единственного байта данных, из нескольких байтов данных, отчета о состоянии устройства, или из управляющей последовательности, используемой для инициализации новой операции.

Изначально устройства, требующие высокой скорости передачи данных, оперируют с канальным путем во фрейм-мультиплексном режиме, более медленные устройства работают в пакетном режиме и самые медленные устройства работают в байт-мультиплексном режиме. Некоторые контрольные блоки имеют ручной переключатель для установки желаемого режима передачи данных.

Операция ввода-вывода, которая случается в канальном пути с параллельным интерфейсом, либо в режиме пакетной передачи, либо в байт-мультиплексном режиме, зависит от возможностей, предоставляемых каналом и устройством ввода-вывода.

Для увеличения производительности некоторые канальные пути и контрольные блоки предоставляют возможности для высокоскоростной передачи и потоковой передачи данных.

Операция ввода-вывода, которая случается в канальном пути интерфейсом типа с последовательной передачей случается во фрейм-мультиплексном режиме или в пакетном режиме. Для увеличения производительности, некоторые контрольные блоки, прикрепленные к интерфейсу последовательной передачи, предоставляют возможности для предоставления чувствительных важных данных программе одновременно с предоставлением информации о проверке контрольного блока, если такое разрешено программой.

В зависимости от контрольного блока или канальной подсистемы, доступ к устройству через подканал может быть ограничен единственным типом канального пути.

Режимы и качества, описанные выше действуют только на протокол, используемый для передачи информации через канальный путь и скорость передачи. Никаких эффектов не наблюдается в отношении центрального процессора или канальных программ относительно способа, с помощью которого эти программы выполняются.

## Вопросы для контроля усвоения материала

1. Перечислите основные принципы работы подсистемы ввода-вывода в большой ЭВМ.
2. Назовите основные компоненты подсистемы ввода-вывода в большой ЭВМ.
3. Как осуществляется распараллеливание работы периферийных устройств в большой ЭВМ?
4. Что такое MIF-образ раздела?
5. Что такое MIF-образ подсистемы?
6. Сколько реальных подсистем ввода-вывода может быть у мейнфрейма?
7. Сколько виртуальных подсистем ввода-вывода может быть у мейнфрейма?
8. Идентификатор периферийного устройства и номер периферийного устройства – это одно и то же?

## Краткие выводы по главе 9 «Подсистема ввода-вывода мейнфрейма zSeries»

Оглянемся назад? 

1. Основные принципы работы подсистемы ввода-вывода в большой ЭВМ:

* совместимость;
* многоуровневое подключение периферийных устройств,
* надежность;
* механизм множественных образов каналов,
* использование специальных процессоров SAP (System Assist Processor),
* распараллеливание работы периферийных устройств.

1. Основные компоненты подсистемы ввода-вывода в большой ЭВМ::
   * + **Периферийное устройство –**  запоминающие устройства и устройства ввода-вывода информации;
     + **Устройство управления CU (Control Unit)–**  устройство, вырабатывающее последовательность управляющих сигналов для периферийного устройства;
     + **Канальный путь –**  интерфейс для обмена информацией между сервером и одним или несколькими CU;
     + **Подканал –**  фрагмент внутренней памяти для хранения управляющей информации об одной операции ввода-вывода.
2. Механизм создания множественных образов:

* **MIF-образ** канальной подсистемы – совокупность MIF-образов раздела;
* **MIF-образ** раздела – совокупность всех SCI и CPI для одного LPAR;**;**
* **CPI -** Идентификатор MIF-образа канального пути;
* **SCI -** Идентификатор MIF-образа подканала

1. Типы адресации каналов ввода-вывода:

* Идентификация канальных путей,
* Нумерация подканалов,
* Нумерация периферийных устройств, и
* адресация, независимая от типа канала.

1. Три режима обмена информацией:

* Фрейм-мультиплексный;
* Пакетный;
* Байт-мультиплексный.

## Литература

1. Варфоломеев В.А., Лепский Э.К., Шмаков М.И., Яковлев В.В. Архитектура и технологии IBM eServer zSeries: учеб. пособие.– М.:Интернет-Ун-т Информационных Технологий, 2005.
2. Галямова Е.В. Способы общения пользователя с современной большой вычислительной машиной (mainframe)//Межвузовский сборник научных трудов. М.: МИРЭА . 2006.
3. Галямова Е.В. Большие вычислительные машины и их место в современном мире// Межвузовский сборник научных трудов,. М.: МИРЭА . 2006.
4. Построение сети хранения данных (Storage Area Network). International Business Machines Corporation, 2005.
5. IBM WebSphere Portal V5. Course materials. International Business Machines Corporation, 2004.
6. Mike Ebbers, Wayne O’Brien. “An Introduction to the Mainframe: z/OS Basics” - International Business Machines Corporation, 2004
7. Толковый словарь по вычислительным системам/ Под ред. В. Иллингуорта и др.: Пер. с англ. А.К. Белоцкого и др.; Под ред. Е.К. Масловского. – М.: Машиностроение, 1990. – 560 с.: ил.
8. zArchitecture Principles of Operation, chapter 2 «Organization»
9. Введение в новые мейнфреймы: основы zOS
10. Борковский А.Б. Англо-русский словарь по программированию и информатике (с толкованиями) – М.: Рус.яз., 1987. – 335 с.