12. Организация памяти ПЭВМ

Организация памяти представляет собой аппаратный механизм, позволяющий операционной системе создавать для выполняющихся программ упрощенную среду. Например, при одновременном выполнении нескольких программ каждой из них должно быть дано независимое адресное пространство. При разделении всеми этими программами одного и того же адресного пространства каждая из них должна была бы выполнять сложные и занимающие много процессорного времени проверки, чтобы избежать влияния на другие программы.

Организация памяти состоит из сегментации и подкачки страниц. Сегментация служит для того, чтобы дать каждой программе несколько независимых, защищенных адресных пространств.

Подкачка страниц используется для поддержки среды, в которой большие адресные пространства моделируются на базе небольшой области оперативной памяти и некоторой дисковой памяти. Разработчики систем могут использовать как оба этих механизма, так и любой из них. При одновременном выполнении нескольких программ для защиты программ от влияния на них других программ можно использовать любой механизм.

Сегментация позволяет иметь полностью неструктурированную и простую память, подобную модели памяти 8-битового процессора, либо высоко структурированную память, с использованием трансляции адресов и защиты.

Средства организации памяти работают с единицами, которые называются сегментами.

Каждый сегмент представляет собой независимое, защищенное адресное пространство.

Доступ к сегментам управляется данными, в которых описаны их размер, уровень привилегированности, который нужен для доступа к ним, типы ссылок к памяти, применимые к этому сегменту (выборка команды, помещение или извлечение из стека, операция чтения, операция записи и т.д.), а также его присутствие в памяти.

В случае простой архитектуры памяти все адреса относятся к одному и тому же адресному пространству. Это модель памяти, используемая 8-битовыми микропроцессорами, такими как 8080, где логический и физический адреса совпадают.

Процессор x86 может быть использован подобным же образом посредством отображения всех сегментов в одном и том же адресном пространстве и запрещения подкачки страниц памяти.

Такая модель может понадобиться при адаптации старых разработок для 32-разрядного

В процессорах x86 предусматривается разделение пространств *памяти* и *ввода-вывода.*

Пространство памяти **(Memory Space) предназначено для хранения кодов инструкций и данных, для доступа к которым имеется 24 способа адресации.**

Память для процессоров представляется в виде линейной последовательности *байт.*

Память для 32-разрядных процессоров x86 подразделяется на байты (8 бит), слова (16 бит), двойные слова (32 бит) и учетверенные слова (64 бит).

Слово (word) записывается в двух смежных байтах, начиная с младшего. Адресом слова является адрес его младшего байта (Low byte). Следующий байт (адрес на единицу больше) содержит старший (High) байт слова.

Все пространство памяти разбивается на *параграфы* — области из 16 смежных байт, начиная с нулевого адреса. Выравнивание по границе параграфа означает, что четыре младших бита адреса — нулевые.

Более крупными единицами организации памяти являются страницы и сегменты.

Память может логически организовываться в виде одного или множества сегментов переменной длины (в реальном режиме — фиксированной). Кроме сегментации, в защищенном режиме возможно разбиение (Paging) логической памяти на страницы размером 4 Кбайт, каждая из которых может отображаться на любую область физической памяти. Начиная с 5-го поколения процессоров x86, появилась возможность увеличения размера страницы до 4 Мбайт.

Сегментация и разбиение на страницы могут применяться в любых сочетаниях.

Сегментация является средством организации логической памяти на прикладном уровне.

Разбиение на страницы применяется на системном уровне для. управления физической памятью.

Сегменты и страницы могут выгружаться из физической оперативной памяти на диск и по мере необходимости подкачиваться с него обратно в физическую память. Таким образом реализуется виртуальная память.

Применительно к памяти различают три адресных пространства: логическое, линейное и физическое.

Основным режимом работы 32-разрядных процессоров считается защищенный режим, в котором работают все механизмы преобразования адресных пространств.

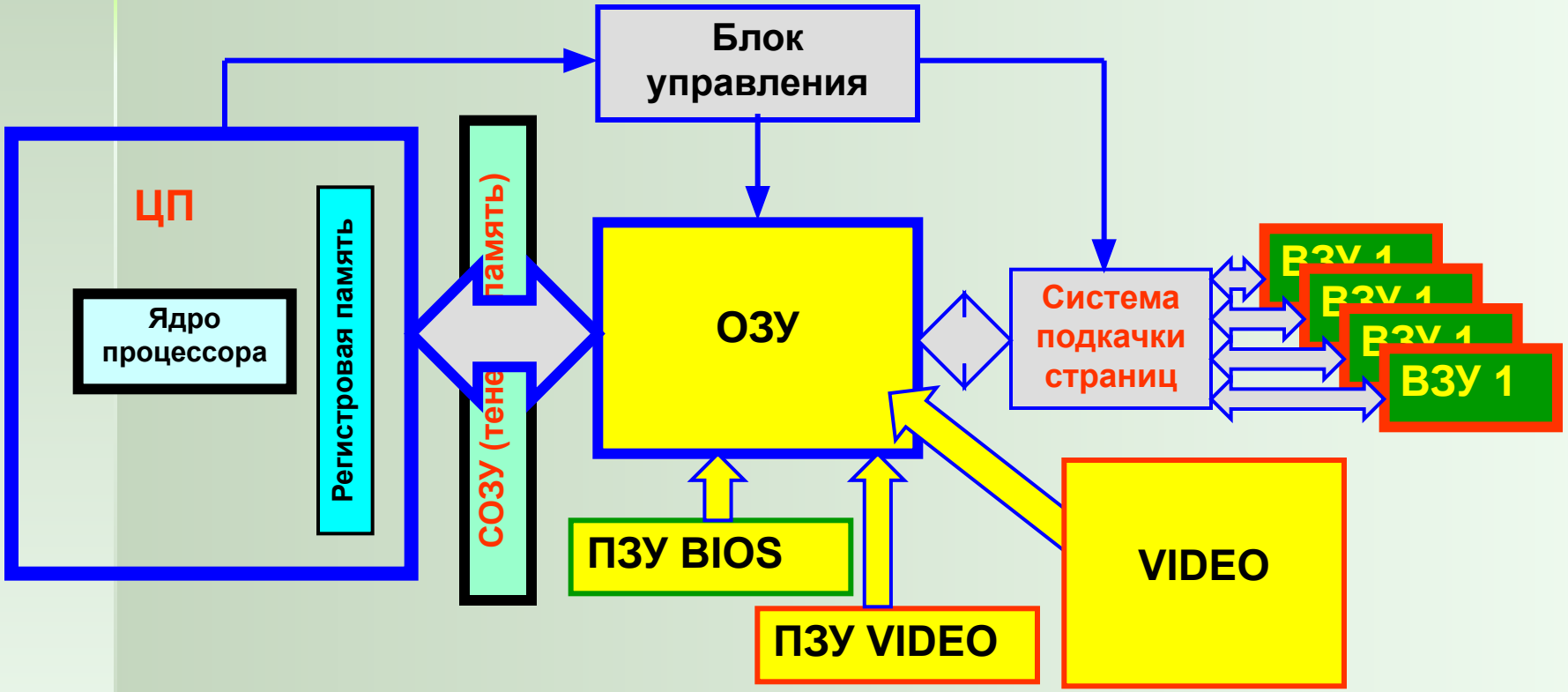
Концепция многоуровневой памяти ПЭВМ

Память определяет эффективность работы ПЭВМ.

Рассматривается три характеристики памяти: *объем, быстродействие, стоимость*.

Эти требования являются взаимно противоречивыми, поэтому невозможно реализовать один тип памяти, который бы отвечал всем этим требованиям.

В современных ПЭВМ реализована многоуровневая схема памяти, обеспечивающая оптимальные характеристики для эффективной работы:



Режимы адресации памяти центрального процессора

Процессор ПЭВМ может работать в одном из пяти режимов адресации памяти и переключаться между ними достаточно быстро как в ту, так и в другую сторону:

***Real Address Mode* (Real Mode) — режим реальной адресации, полностью совместим с 8086.**

В этом режиме возможна адресация до 1 Мбайт физической памяти.

***Protected Virtual Address Mode* (Protected Mode) — защищенный режим виртуальной адресации.**

В этом режиме процессор позволяет адресовать до 4 Гбайт физической памяти, через которые при использовании механизма страничной адресации могут отображаться до 64 Тбайт виртуальной памяти каждой задачи.

***Virtual 8086 Mode* — режим виртуального процессора 8086.**

Этот режим является особым состоянием задачи защищенного режима, в котором процессор функционирует как 8086. На одном процессоре в таком режиме может параллельно исполняться несколько задач с изолированными друг от друга ресурсами. При этом использование физического адресного пространства памяти управляется механизмами сегментации и трансляции страниц. Попытки выполнения недопустимых команд, выхода за рамки отведенного пространства памяти и разрешенной области ввода-вывода контролируются системой защиты.

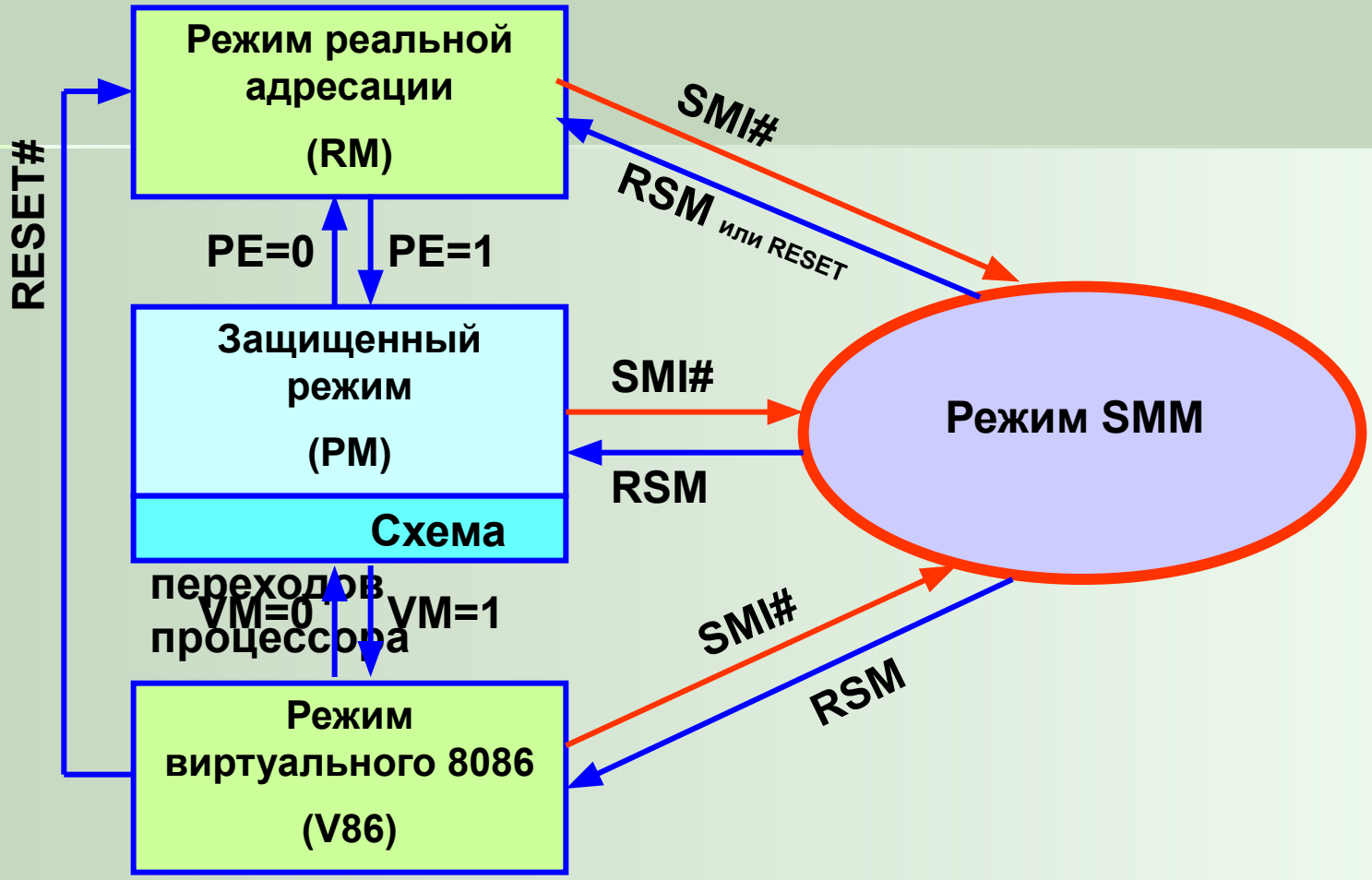
***Big Real Mode* (*Unreal Mode*) – *«нереальный» режим,* который поддерживают все 32-разрядные процессоры, позволяет адресоваться ко всему 4-гигабайтному пространству памяти.**

В этом режиме инструкции исполняются так же, как и в реальном режиме, но с помощью дополнительных сегментных регистров FS и GS программы получают непосредственный доступ к данным во всей физической памяти.

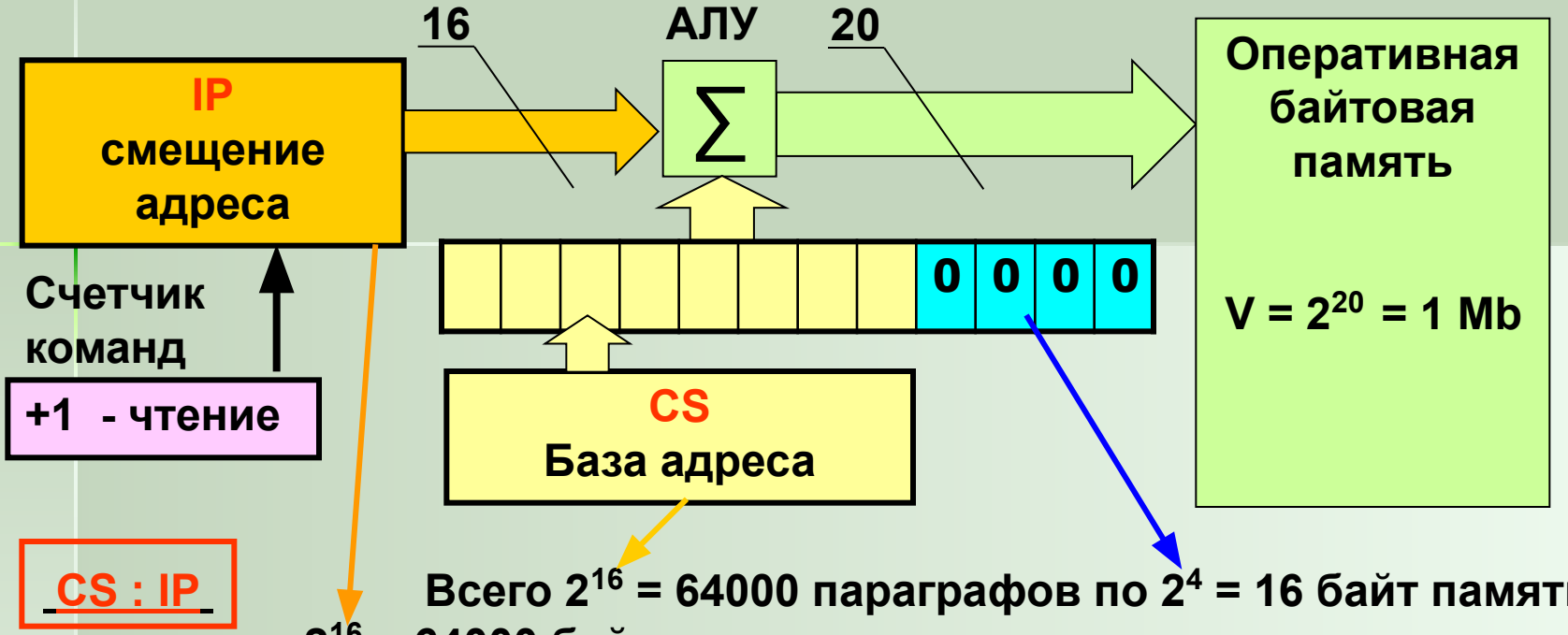
***System Management Mode* (*SMM*) - особый режим системного управления при котором процессор выходит в иное, изолированное от остальных режимов пространство памяти.**

Этот режим используется в служебных и отладочных целях.

Схема переходов процессора между режимами адресации.



13. Схема сегментного распределения памяти в режиме прямой адресации



Всего 2^{16} = 64000 параграфов по 2^4 = 16 байт памяти

размер сегмента 2^{16} = 64000 байт, начало сегмента определяется параграфом.

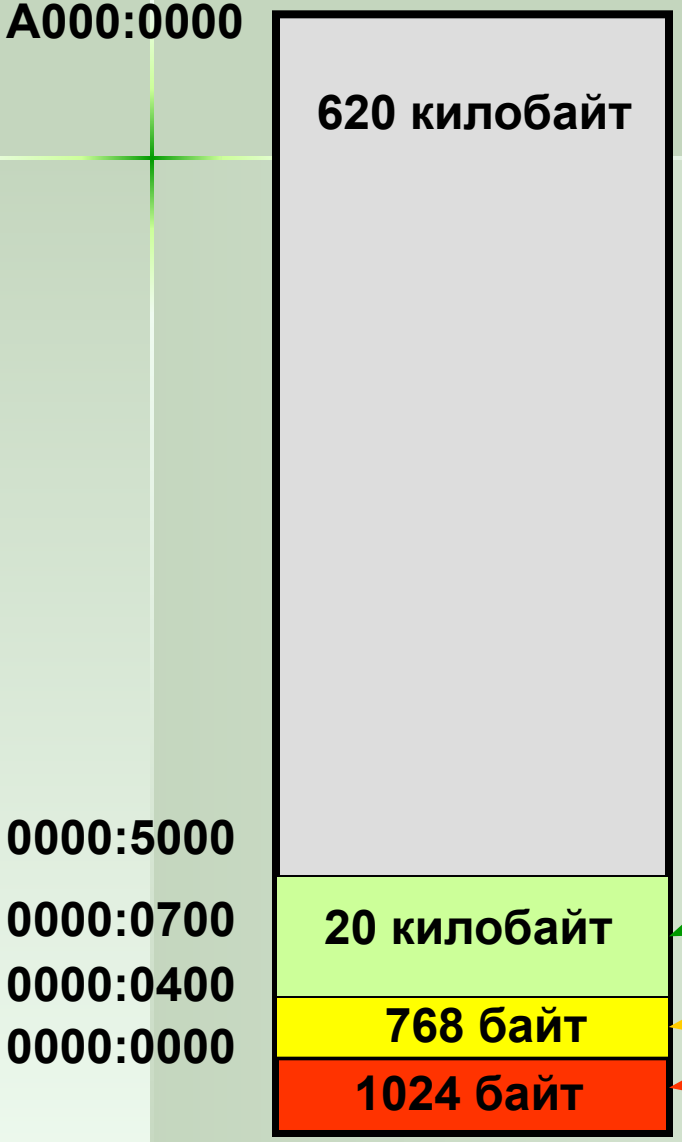
Недостатки сегментной адресации:

1. Размещение сегментов в памяти произвольное (от свободного параграфа).
2. Сегменты могут перекрывать друг друга.
3. Защита памяти от несанкционированного доступа выполнена программными средствами.
4. При обращении к «чужой» памяти происходит зависание программы.

Для устранения этих недостатков в современных ПЭВМ используется система защищенной адресации, а сегментная адресация применяется в виртуальных задачах для обеспечения преемственности ПО.

Распределение памяти ПЭВМ в режиме прямой адресации.

Логическая структура памяти по адресам от 00000 до A0000 (640 Kb).



Распределение памяти сегментное. По 64 килобайта на сегмент. Разбито на параграфы по 16 байт на параграф.

Область прикладных программ пользователя.

Загрузчик программ:

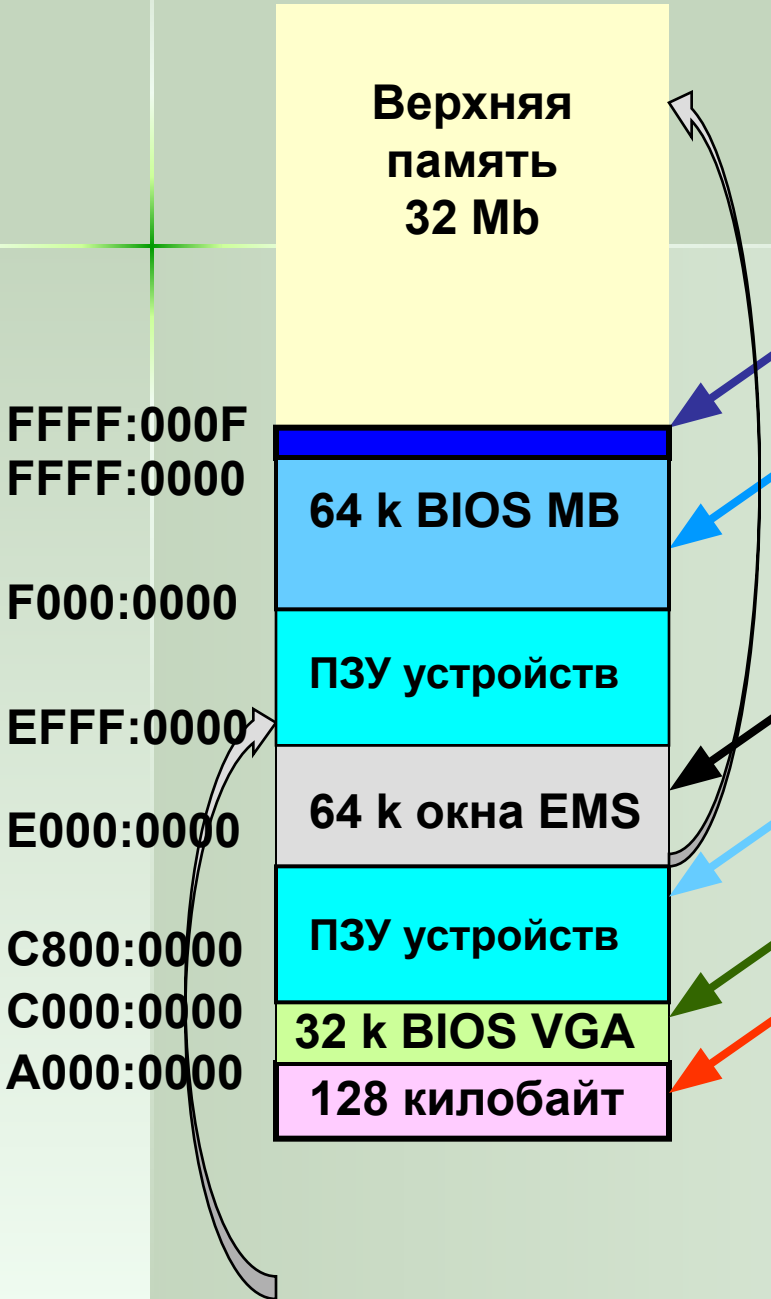
Для DOS - command.com
Для Windows – ntldr.

Резидентные программы, драйвера периферийных устройств, утилиты DOS.

Область констант BIOS, переменные таймера, буфер клавиатуры и т.д.

256 4-х байтных элементов таблицы векторов прерываний BIOS и DOS: CS:IP (указателей на программы-обработчики).

Логическая структура памяти по адресам от A0000 до FFFFF (384 Kb).



Область верхней памяти (32 мегабайта) в режиме эмуляции 8086 доступна в XMS-режиме (программа emm386.exe) через 4 окна EMS по 16 Kb.

Окно FFFF0-FFFFF (16 килобайт) доступно через драйвер himem.sys.

Область программ BIOS MB отображается в сегменте параграфа F000.

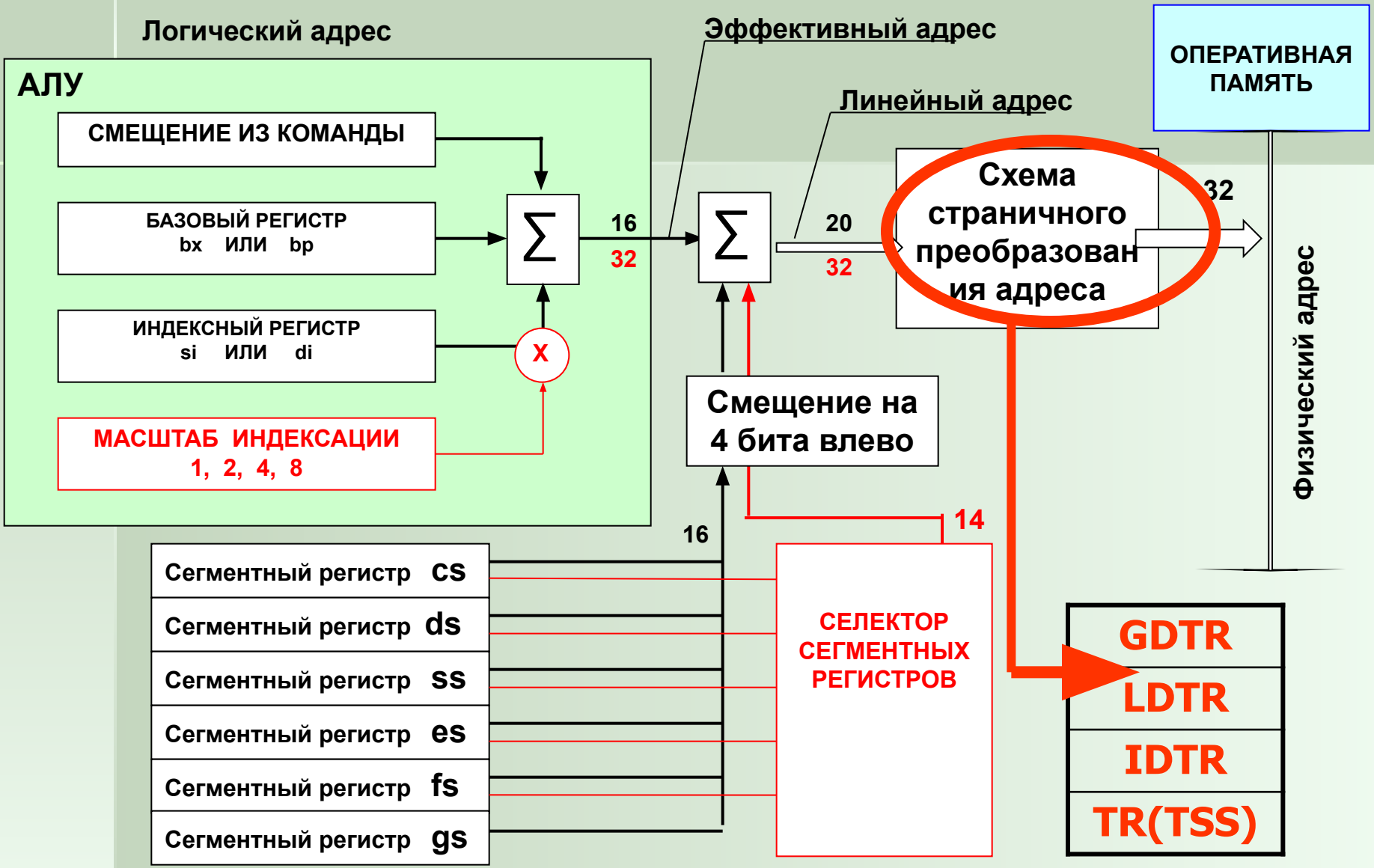
4 окна EMS доступа к верхней памяти

Область отображения ПЗУ адаптеров устройств MB и периферийных устройств.

Область отображения BIOS VGA

2 видеостраницы по 64 килобайта, доступные CPU ПЭВМ для подготовки примитивов графических изображений для видеоадаптеров

Схема формирования физического адреса памяти ПЭВМ.



Способы адресации операндов команды.

1. **Неявная адресация.** Операнд задается неявно на микропрограммном уровне. Такие команды не содержат операндов.
2. **Непосредственная адресация.** Операнд находится в самой команде и является частью ее кода. Непосредственный операнд (константа) может быть только вторым операндом (источником).
3. **Регистровая адресация.** Операнд находится в одном из регистров РОН.
4. **Адресация I/O.** Операндом является порт ввода-вывода.
5. **Стековая адресация.** Операнд находится в стеке.
6. **Прямая адресация.** Эффективный адрес находится в самой команде.
   - 6.1. **Абсолютная прямая адресация.** Эффективный адрес берется из поля смещения в самой команде.
   - 6.2. **Относительная прямая адресация.** К полю смещения команды добавляется значение регистра – указателя команд IP/EIP.
7. **Косвенная адресация.** Операнд находится в оперативной памяти ПЭВМ.
   - 7.1. **Косвенная базовая адресация.** Эффективный адрес в одном из РОН.
   - 7.2. **Косвенная базовая адресация со смещением.** К базовому адресу добавляется значение смещения, находящееся в другом РОН.
   - 7.3. **Косвенная индексная адресация.** Адрес в индексном регистре.
   - 7.4. **Косвенная базовая индексная адресация.** Эффективный адрес формируется, как сумма базового РОН и индексного регистра.
   - 7.5. **Косвенная базовая индексная адресация со смещением.** Эффективный адрес- сумма базового, индексного и РОН вычисляемого смещения.

14. Защищенный режим адресации памяти центрального процессора

Защита необходима при работе в мультизадачном режиме.

Средства защиты могут быть использованы для предотвращения взаимного влияния одновременно выполняемых задач. Например, может выполняться защита от затирания одной задачей команд и данных другой задачи.

При разработке конкретной программы механизм защиты поможет получить более четкую картину программных ошибок.

Когда программа выполняет неожиданную ссылку к недопустимой в данный момент области памяти, механизм защиты может блокировать данное событие и сообщить о нем в процессе исполнения программы.

При сбое в программе действие его распространится таким образом лишь на ограниченный домен памяти.

Операционная система при этом может быть защищена от нарушений ее областей, что позволяет зарегистрировать диагностическое сообщение об ошибке и сделать попытку автоматического восстановления программы.

Защита может применяться к сегментам и страницам памяти.

Два бита регистра процессора определяют уровень привилегированности текущей выполняемой программы (этот уровень называется текущим уровнем привилегированности, или CPL). Во время трансляции адреса для доступа к сегменту или странице выполняется проверка CPL.

Хотя для выключения механизма защиты специального управляющего регистра или бита режима не предусмотрено, этот эффект можно получить, назначив уровень привилегированности 0 (это старший уровень привилегированности) всем селекторам сегментов, дескрипторам сегментов и элементам страничной таблицы исполняемой задачи.

Системные регистры, используемые в защищенном режиме.

Регистры управления.

Cr0 – регистр переключения режимов адресации.

PE (Protect Enable) – бит 0 =0 реальная адресация, =1 защищенный режим

MP (Math Present) – бит 1 наличие матсопроцессора всегда =1.

TS (Task Switched) – бит 3 =1 переключение задач.

AM (Alignment Mask) – бит 18 =1 устанавливает контроль выравнивания байтов.

CD (Cache Disable) - бит 30 =1 запрещение КЭШ-памяти.

PG (PaGing) - бит 31 =1 разрешение страничного преобразования.

Cr1 – зарезервирован разработчиком центрального процессора (не используется).

Cr2 – регистр страничной организации памяти (свопинга) (хранит 32-битный линейный адрес, по которому был получен последний отказ страницы памяти).

Cr3 - регистр каталога страниц первого уровня (содержит 20-битный физический базовый адрес каталога страниц текущей задачи).

Cr4 – регистр (присутствует с x86 пятого поколения) разрешения новых архитектурных расширений центрального процессора, например, увеличения размера страницы с 4-х Кбайт до 4-х Мбайт, поддержка блока XMM и др.

Системные адресные регистры.

GDTR (Global Descriptor Table Register) - регистр глобальных дескрипторов,

IDTR (Interrupt Descriptor Table Register) - регистр таблицы дескрипторов прерываний,

TR (Task Register) - регистр селектора текущей задачи в таблице GDT, описывающий текущий сегмент состояния задачи TSS (Task Segment Status)

LDTR (Local Descriptor Table Register) - регистр локальных дескрипторов задачи.

Регистры отладки.

dr0, dr1, dr2, dr3 – линейные адреса точек останова исполнения текущей задачи.

dr6 и dr7 – регистры состояния и управления процессом отладки.

Схема сегментного распределения памяти в защищенном режиме.

Логический адрес состоит из 16-битового селектора сегмента и 32-битового смещения в этом сегменте.

Логический адрес транслируется в линейный адрес сложением смещения и базового адреса сегмента.

Базовый адрес берется из дескриптора сегмента - структуры данных в памяти, которая содержит информацию о размере и расположении данного сегмента в памяти, а также информацию для управления доступа.

Дескриптор сегмента берется в одной из двух таблиц, таблицы глобальных дескрипторов (GDT) или таблицы локальных дескрипторов (LDT).

Для всех программ в системе существует единая таблица GDT, а также по одной таблице LDT для каждой отдельной выполняемой в текущий момент программы. Каждый логический адрес связан со своим сегментом (даже если система отображает все сегменты в одно и том же линейное адресное пространство).

Хотя программа может иметь тысячи сегментов, для немедленного использования доступны только шесть из них.

Имеется шесть сегментов, селекторы которых загружены в процессор.

Селектор сегмента содержит информацию, используемую для транслирования логического адреса в соответствующий линейный адрес.

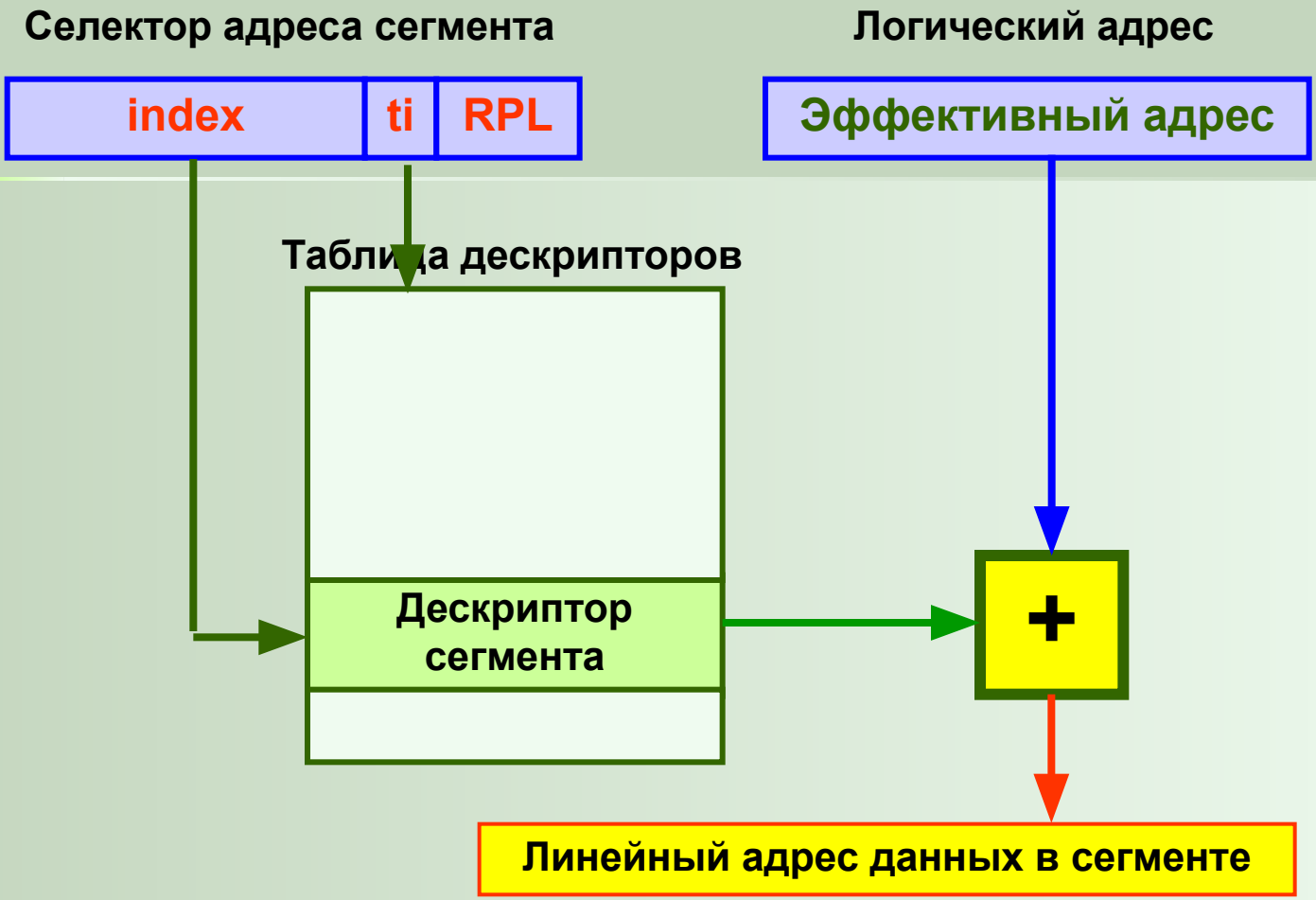
Для каждого вида ссылки к памяти (пространству кода, пространству стека и пространствам данных) в процессоре существуют отдельные сегментные регистры.

Они содержат селекторы текущих используемых сегментов.

Доступ к другим сегментам требует загрузки сегментного регистра с помощью разновидности команды MOV.

Одновременно может быть доступно до четырех сегментов данных, что в сумме с остальными составляет шесть сегментных регистров.

Схема формирования линейного адреса в защищенном режиме

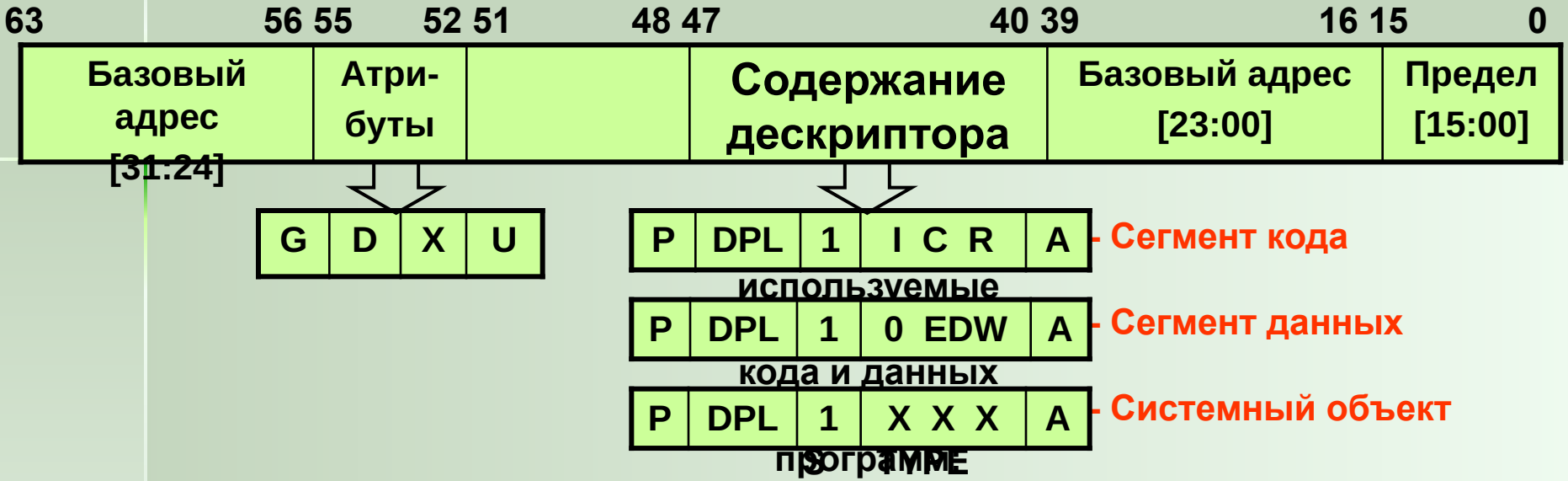


Бит ti определяет глобальную (0) или локальную (1) таблицу дескрипторов,

Два бита RPL определяют требуемый уровень привилегий.

Содержание дескриптора

Дескрипторы, используемые для сегментов кода и данных прикладных программ:



G - Грануляция (=0 граница от 1 байта до 4 Мбайт, =1 от 4 Кбайт до 4 Гбайт)

D - Размер операндов по умолчанию (Распознается только в дескрипторах кодового сегмента: 0=16-битовый размер; 1=32-битовый размер)

X - Зарезервировано Intel (не подлежит использованию)

U - Доступно для использования системным программным обеспечением

P - Присутствие сегмента (наличие указанной области памяти)

DPL - Уровень привилегированности дескриптора

S - Тип дескриптора (=0 системный; =1 прикладной)

TYPE - Тип сегмента (сегмент кода, сегмент данных, системный объект, определяет тип доступа и направление наращивания от базового адреса сегмента)

A - Бит обращения (определяет наличие обращения к сегменту)

Сегмент кода

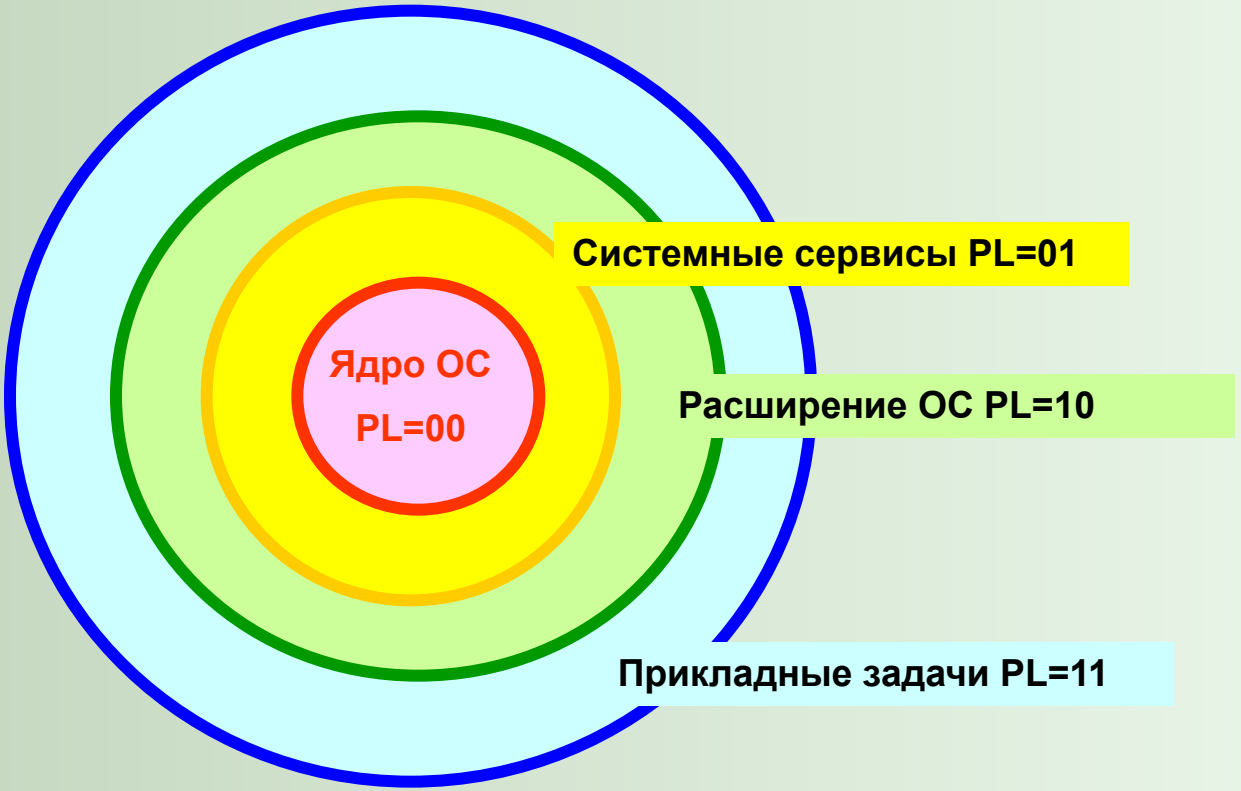
Уровни привилегий задачи

Четырехуровневая иерархическая система привилегий предназначена для управления использованием привилегированных инструкций и доступам к дескрипторам.

Уровни привилегий нумеруются от 0 до 3.

0-й уровень соответствует максимальным возможностям доступа к памяти.

3-й уровень имеет самые ограниченные права и отводится прикладным задачам.



Защита

Защита предназначена для предотвращения несанкционированного доступа к памяти и выполнения критических инструкций – команд HLT (останов процессора), команд ввода-вывода, управления флагом расширения прерываний и команд, влияющих на сегменты кода и данных.

Механизм защиты вводит следующие ограничения:

Ограничение использования сегментов: могут использоваться только описанные в GDT и LDT.

Ограничение доступа к сегментам через правила привилегий.

Ограничение набора инструкций: исключение некоторых системных команд)

Ограничение возможности межсегментных вызовов и передачи управления.

Проверка механизма страничной адресации.

Переключение задач

Процессор x86 обеспечивает аппаратную поддержку мультизадачности.

Задачей называется программа, выполняемая в текущий момент, либо ожидающая выполнения во время работы другой программы.

Задача запускается прерыванием, исключением, переходом или вызовом.

Существует два типа задаче-ориентированных дескрипторов, которые могут находиться в таблице дескрипторов: дескрипторы сегмента состояния задачи и шлюзы задачи.

Когда управление передается любому из таких дескрипторов, происходит переключение задачи.

Переключение задачи похоже на вызов процедуры, но оно выполняет сохранение большего количества информации о состоянии процессора.

Вызов процедуры сохраняет только содержимое регистров общего назначение, а в некоторых случаях содержимое только одного регистра (EIP). При вызове процедуры содержимое сохраняемых регистров помещается в стек, чтобы процедура имела возможность вызвать сама себя.

Когда процедура вызывает сама себя, она называется реентерабельной.

Переключение задачи передает выполнение в полностью иную среду, среду задачи. Для этого требуется сохранить содержимое практически всех регистров процессора, таких как регистр EFLAGS.

В отличие от процедур, задачи не реентерабельны.

Переключение задачи ничего не помещает в стек.

Информация о состоянии процессора сохраняется в структуре данных в памяти, которая называется сегмент состояния задачи.

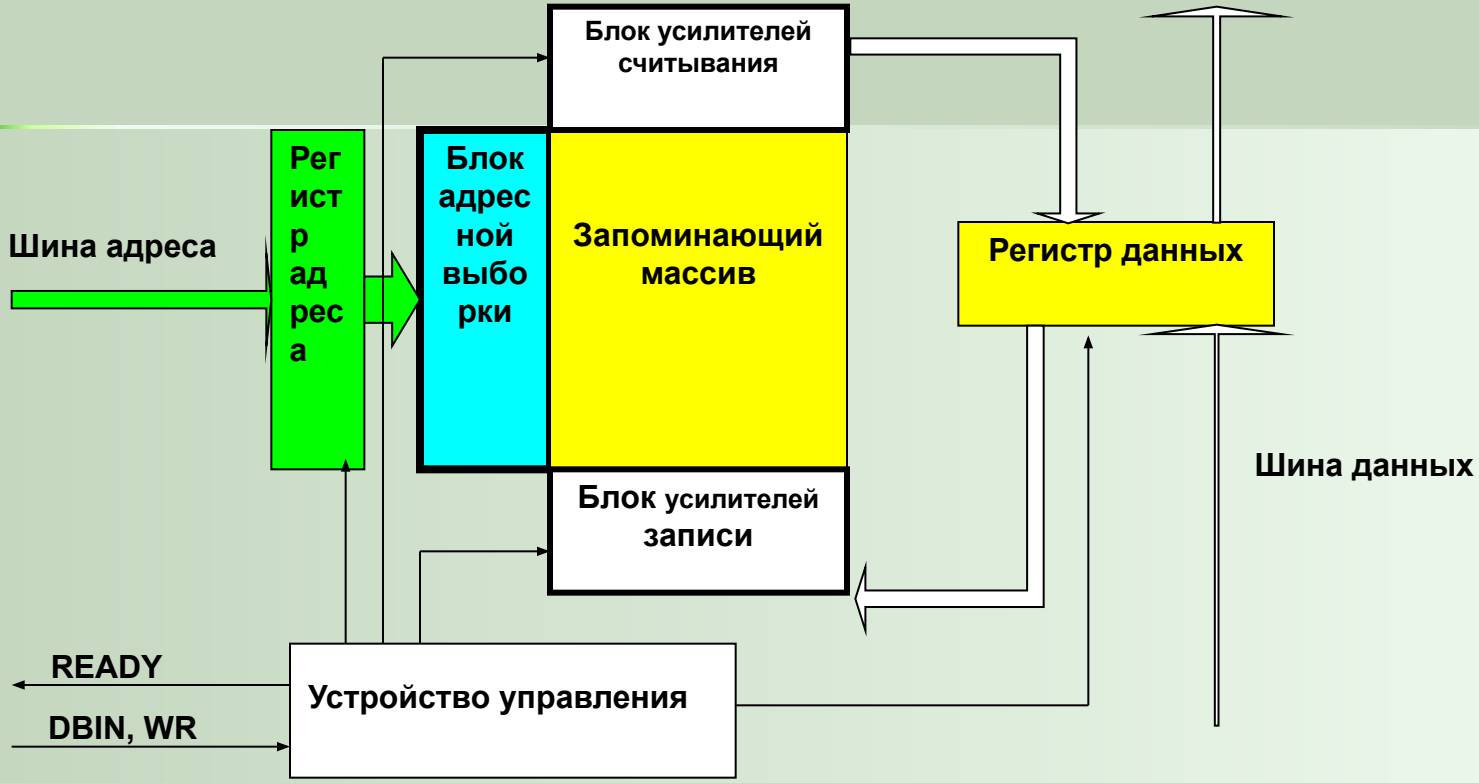
В число регистров и структур данных, поддерживающих мультизадачность, входят:

- **Сегмент состояния задачи.**
- **Дескриптор сегмента состояния задачи.**
- **Регистр задачи**
- **Дескриптор шлюза задачи.**

Используя эти структуры, процессор x86 может переключать выполнение с одной задачи на другую, сохраняя контекст текущей задачи, допуская тем самым рестарт другой задачи.

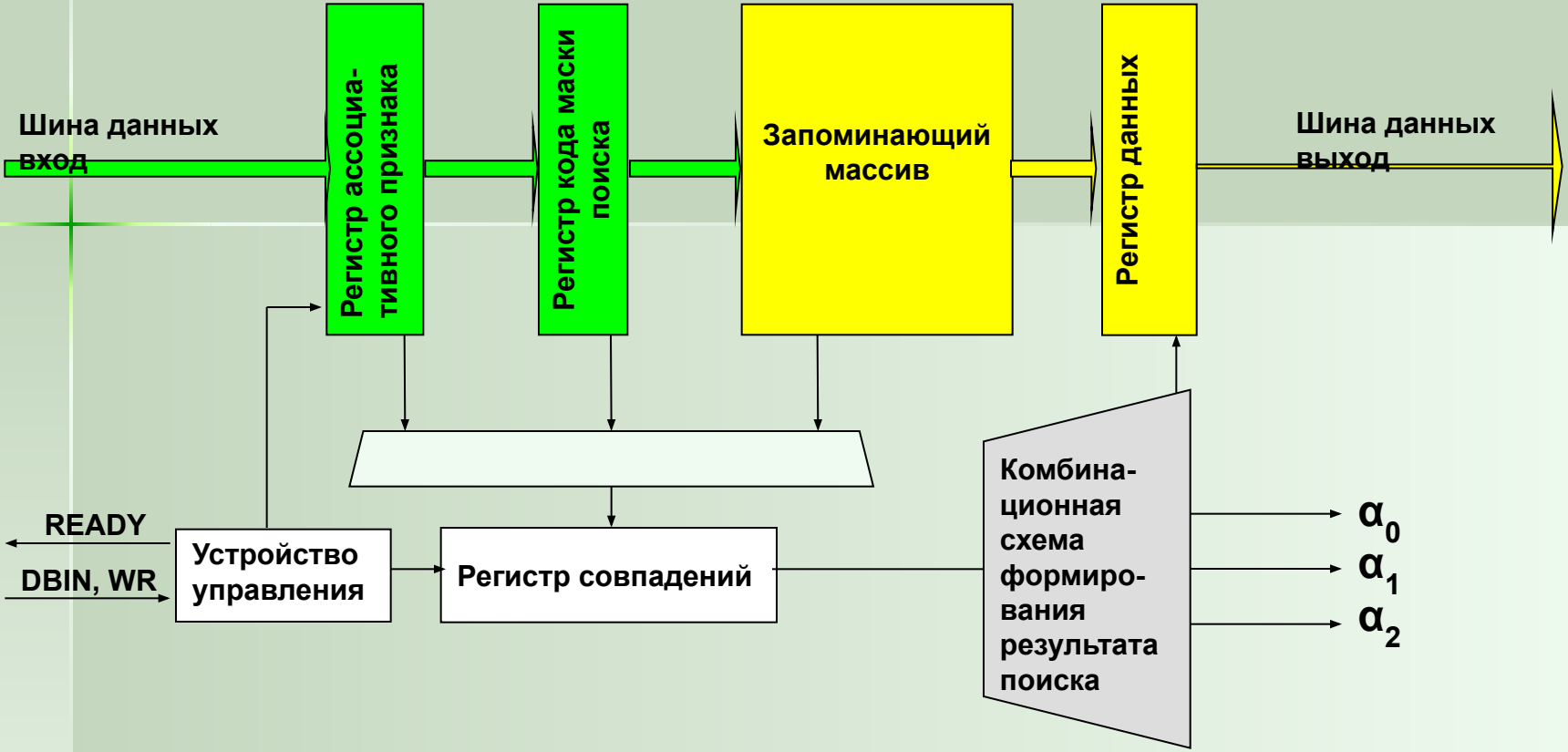
15. Схемы организации памяти ПЭВМ.

1. Адресная память (память произвольного доступа)



Процесс выборки адреса завершается записью или считыванием соответствующего байта данных, сопровождаемый сигналами запроса со стороны центрального процессора и сигналом готовности контроллера оперативной памяти.

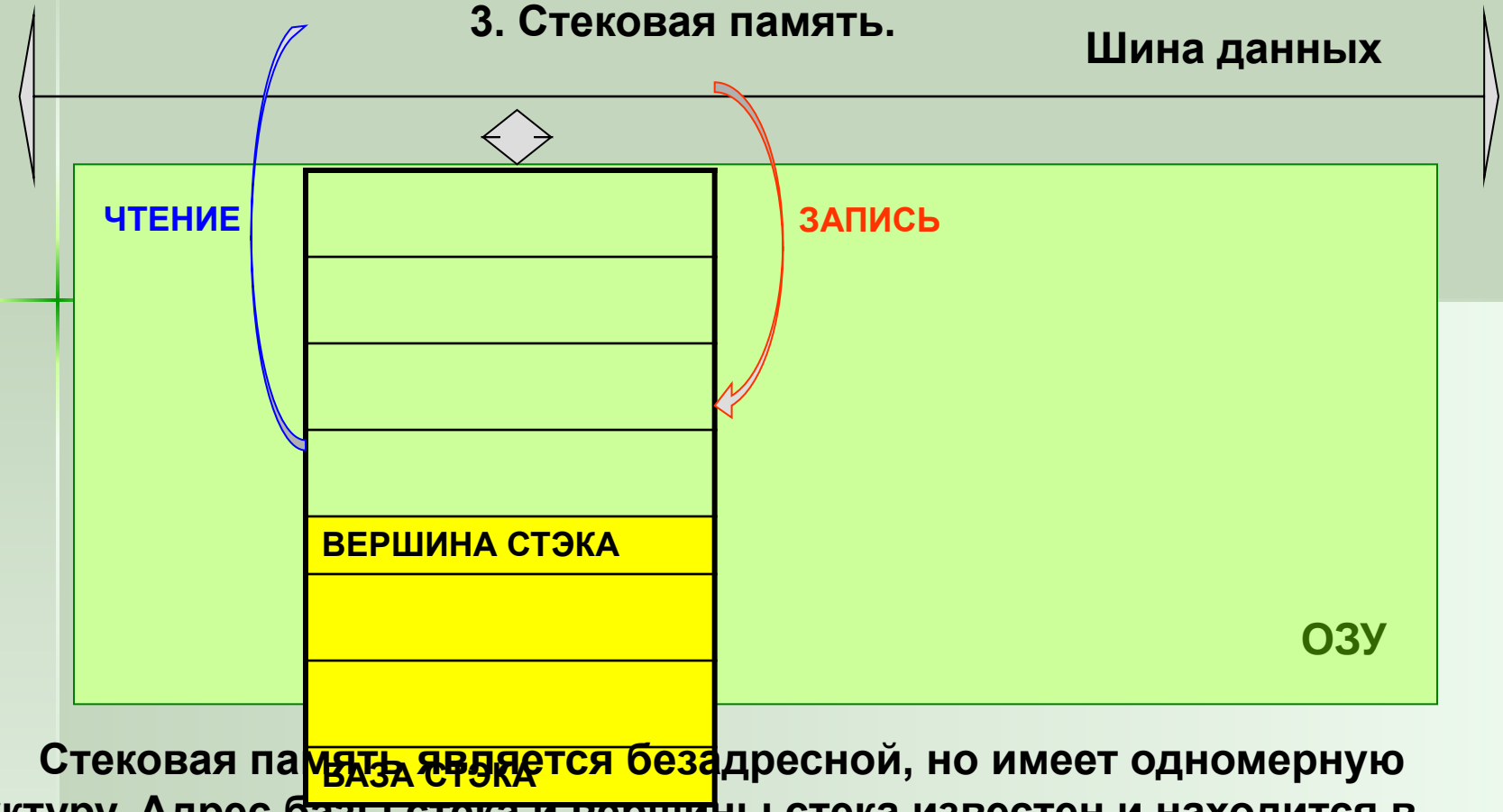
2. Ассоциативная память (сверхоперативная память или КЭШ-память).



Процесс выборки данных осуществляется не по адресу, а по содержанию информации (ассоциативному признаку или по отдельным разрядам этого признака):

α_0 – считывание данных запрещено (данных в памяти не найдено),
α_1 – считывается найденное слово,
α_2 – считывается слово из ячейки, имеющей наименьший номер среди отмеченных по результатам поиска (наименьшее время хранения).

3. Стековая память.

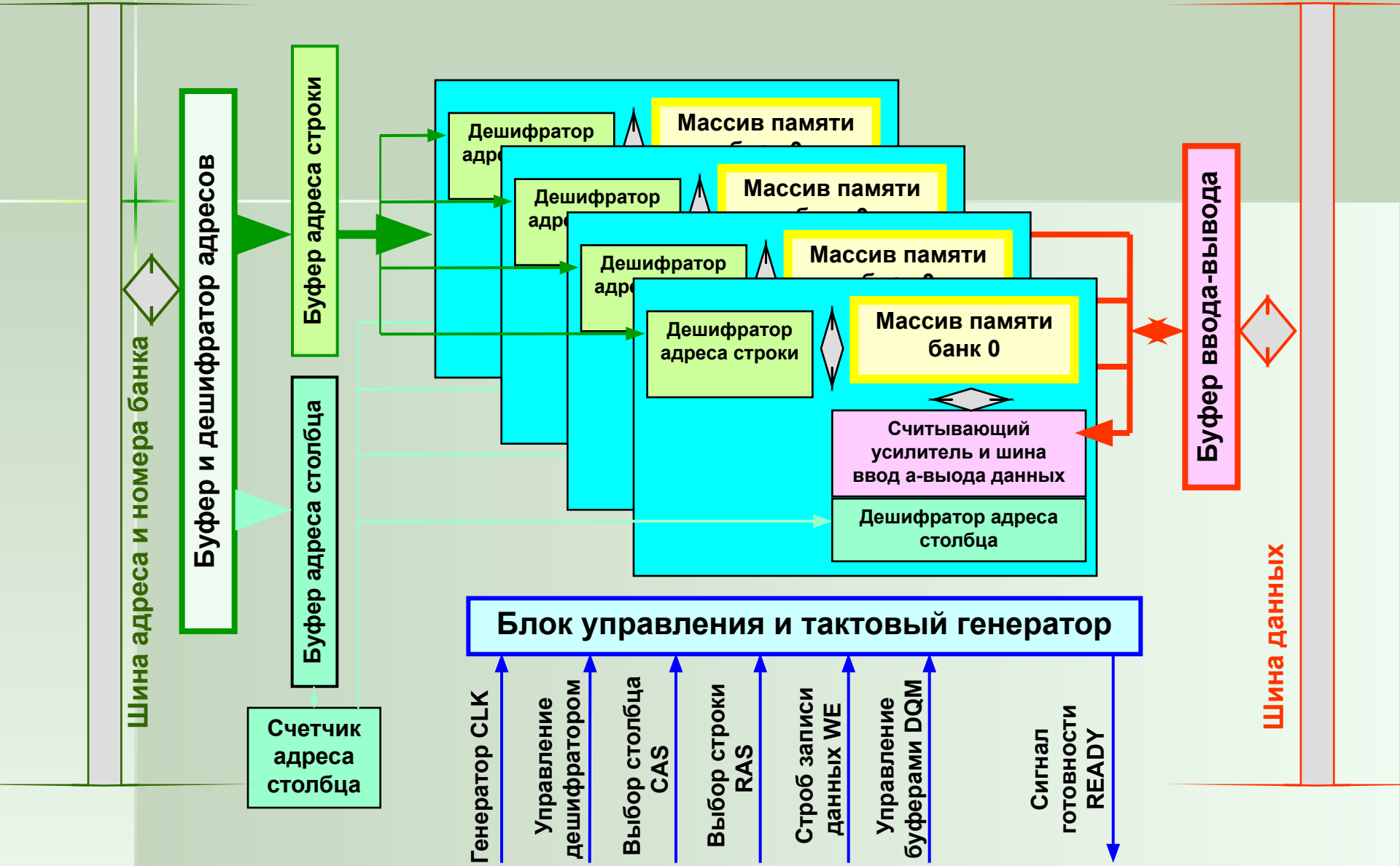


Стековая память является безадресной, но имеет одномерную структуру. Адрес базы стека и вершины стека известен и находится в регистрах SS-BP и SS-SP соответственно. Запись и считывание возможны только из строго определенных ячеек памяти: чтение осуществляется из вершины стека (последней занятой ячейки стека), при этом значение смещения адреса SP уменьшается на единицу, запись осуществляется в следующую свободную ячейку стека и ее адрес заносится в указатель вершины стека.

Типы исполнения памяти (классификация памяти).

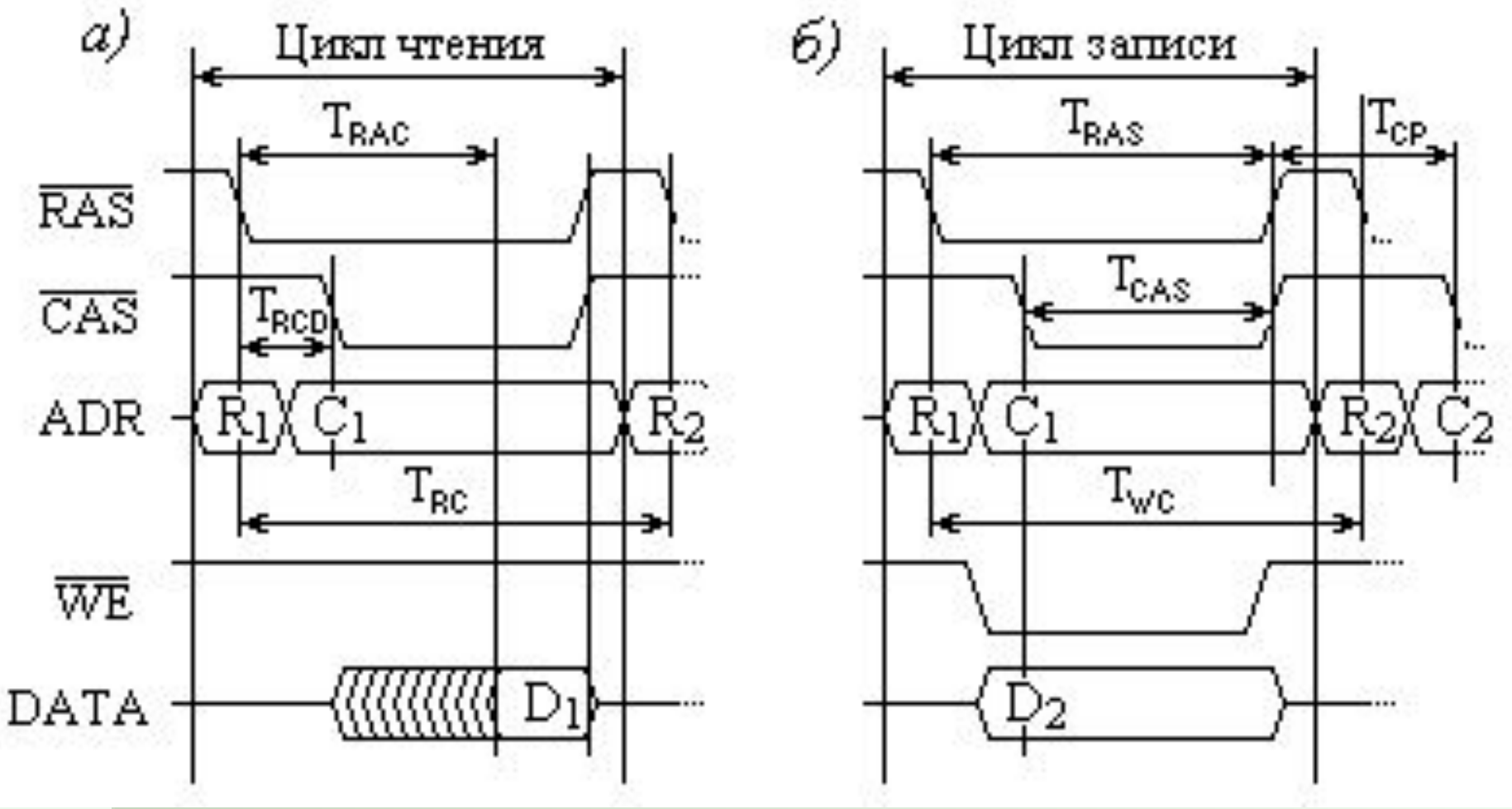
(DDR II SD RAM) DDR – удвоенная скорость передачи данных достигается за счет выполнения обращения к памяти не по уровню, а по фронту, т.е. два раза за время одного цикла.

Структурная схема доступа к ячейкам памяти произвольного доступа.

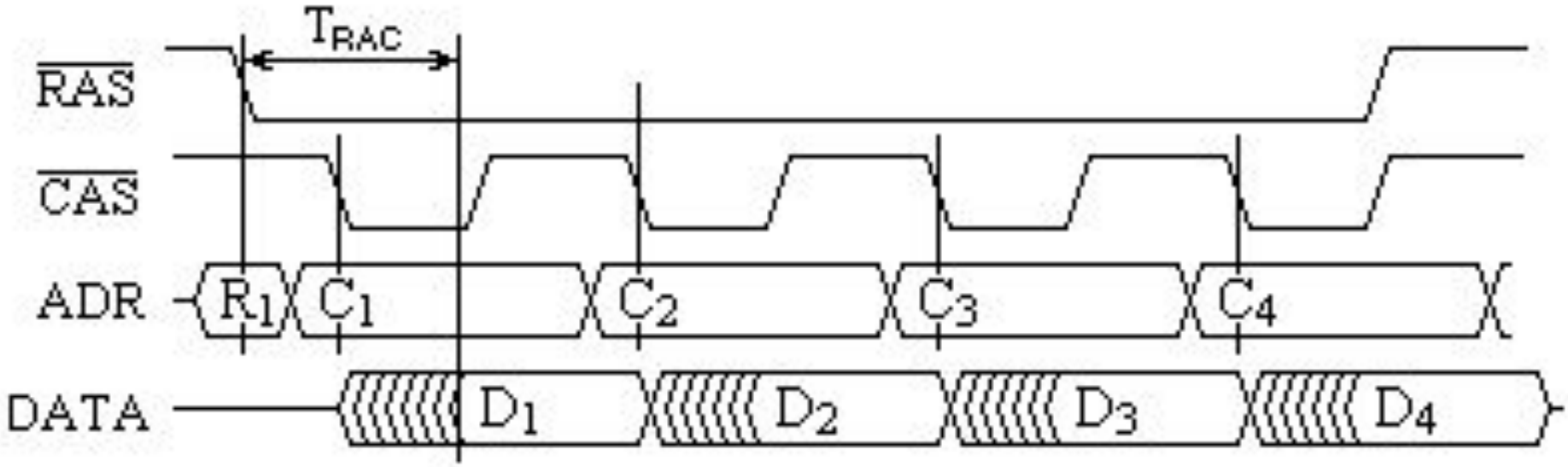


Временные диаграммы простых циклов:

а) чтения асинхронной динамической памяти
б) записи асинхронной динамической памяти



Временные диаграммы циклов чтения асинхронной динамической памяти произвольного доступа при пакетной обработке EDO DRAM:



Временные диаграммы циклов чтения и записи с одиночной и двойной скоростью передачи в синхронной конвейерно-пакетной статической памяти произвольного доступа PB SRAM: