Аппаратные средства телекоммуникационных систем

Модель памяти процессоров

Особенности модели памяти процессоров

Аппаратные средства телекоммуникационных систем. Модель памяти процессоров

Модель памяти процессора

- **Модель памяти процессора** – метод организации пространства ОЗУ (доступа к памяти) с аппаратной и/ или программной точки зрения.
- **Плоская модель памяти** — это метод организации адресного пространства оперативной памяти, в котором программная память и память данных находятся в одном адресном пространстве (*Используется в современных ЭВМ*).
  - Для 16-битных процессоров плоская модель памяти позволяет адресовать 64 кБ оперативной памяти; для 32-битных процессоров 4 ГБ, для 64-битных — до 16 эксабайт (для amd64 размер ограничен 256 ТБ).
    - Например адресное пространство для 32 битного режима будет состоять из 2^32 ячеек памяти пронумерованных от 0 и до 2^32-1
- **Сегментная модель памяти (модель адресации памяти) –** модель в которой память разбита на сегменты, каждый из которых характеризуется своими настройками доступа (напр, программы ОС не могут получить доступ памяти ядра ОС).

Модель памяти процессора

- **Виртуальная память** — модель памяти ПК, в которой процессор работает с виртуальной памятью, транслируемой в физические адреса устройством управления памятью (MMU).
  - Преимущество – виртуальное расширение ОЗУ за счет файла подкачки.
- **Страничная виртуальная модель памяти –** способ организации виртуального адресного пространства, в котором единицей отображения виртуального адреса в физический является область фиксированного размера (страница).
  - Страницы могут быть не только в ОЗУ, но и сброшены в файл подкачки на жесткий диск.
  - Физический адрес = адрес страницы + смещение внутри нее.
  - Организацией работы страниц занимается операционная система.

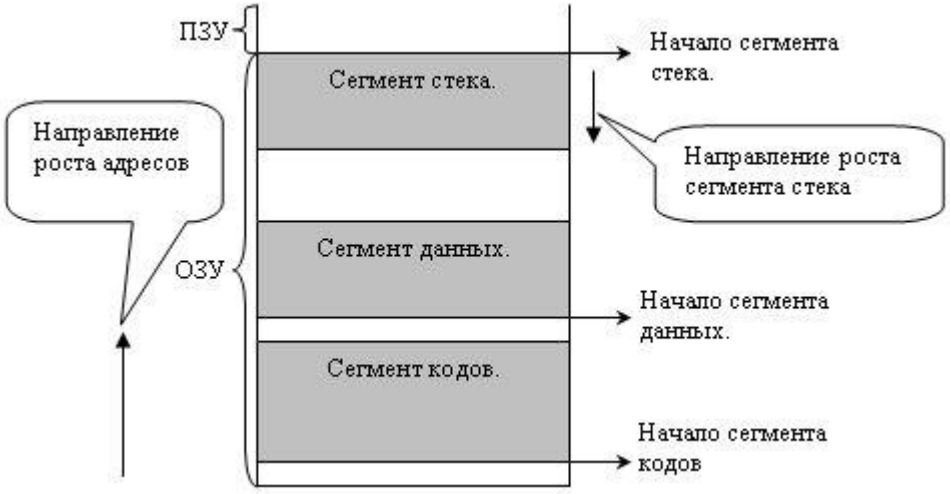
Особые виды памяти

- **Регистровая память** – набор регистров процессора, ячейки памяти в самом процессоре.
  - **Basic program registers** (Основные программные регистры) - это для обслуживания процессора и обработки целочисленных данных.
    Floating Point Unit registers (FPU, X87) – это набор регистров для работы с данными в формате с плавающей точкой.
    MMX и XMM registers - это регистры для систематизированной обработки увеличенного количества операндов.
  - Когда процессор совершает какие-то операции со значением или с памятью, он берет эти значения непосредственно из регистров или из стека.
- **Стек** - специальный раздел оперативной памяти, предназначенный для быстрого безадресного доступа к элементам (по принципу last input first output).

Сегменты памяти

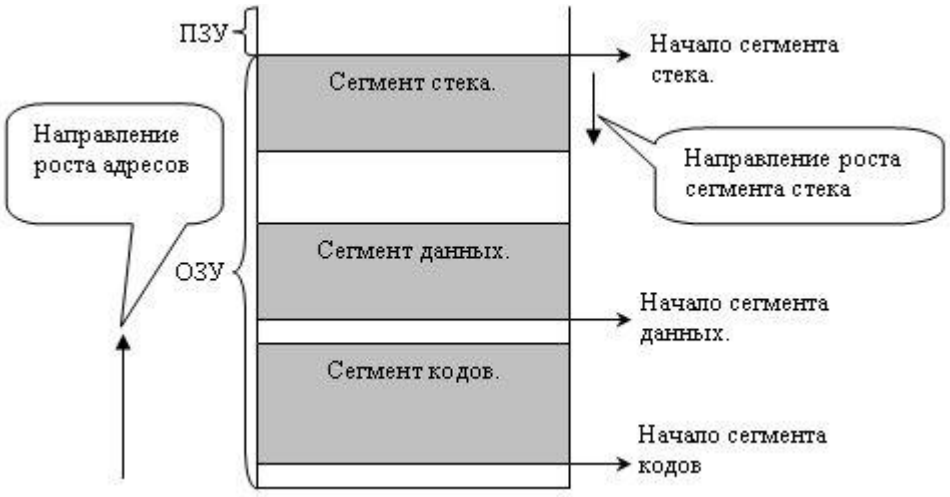
Логически память разделена на 3 основных сегмента памяти.

- **CS** - **сегмент кода**, содержит машинные команды (проргамму);
- **DS** - **сегмент данных** – содержит данные, то есть константы и рабочие области, необходимые программе;
- **SS - сегмент стека** – содержит адреса возврата в точку вызова подпрограмм.
- адресу определенному ОС.



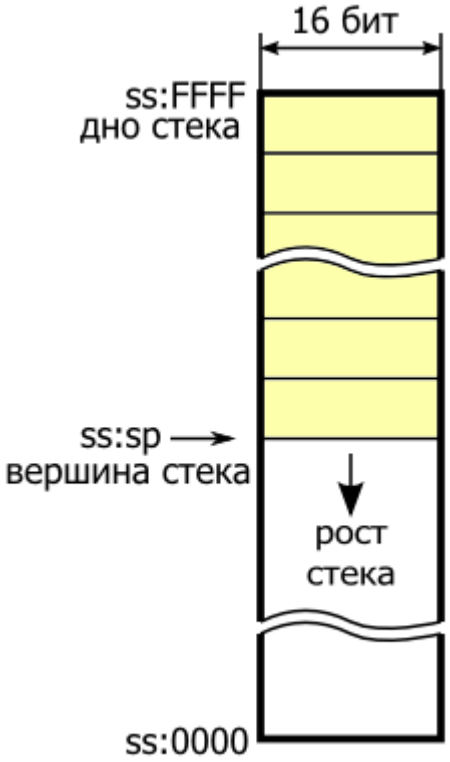
Сегменты памяти

- За распределение сегментов, их начальный (базовый) адрес и их размер отвечает дескриптор сегментов – 64 бита памяти, расположенных по адресу определенному ОС.
- В современных системах базовые адреса всех сегментов могут быть одним нулевым адресом (то есть ОС сама распределяет где какой сегмент будет сама).



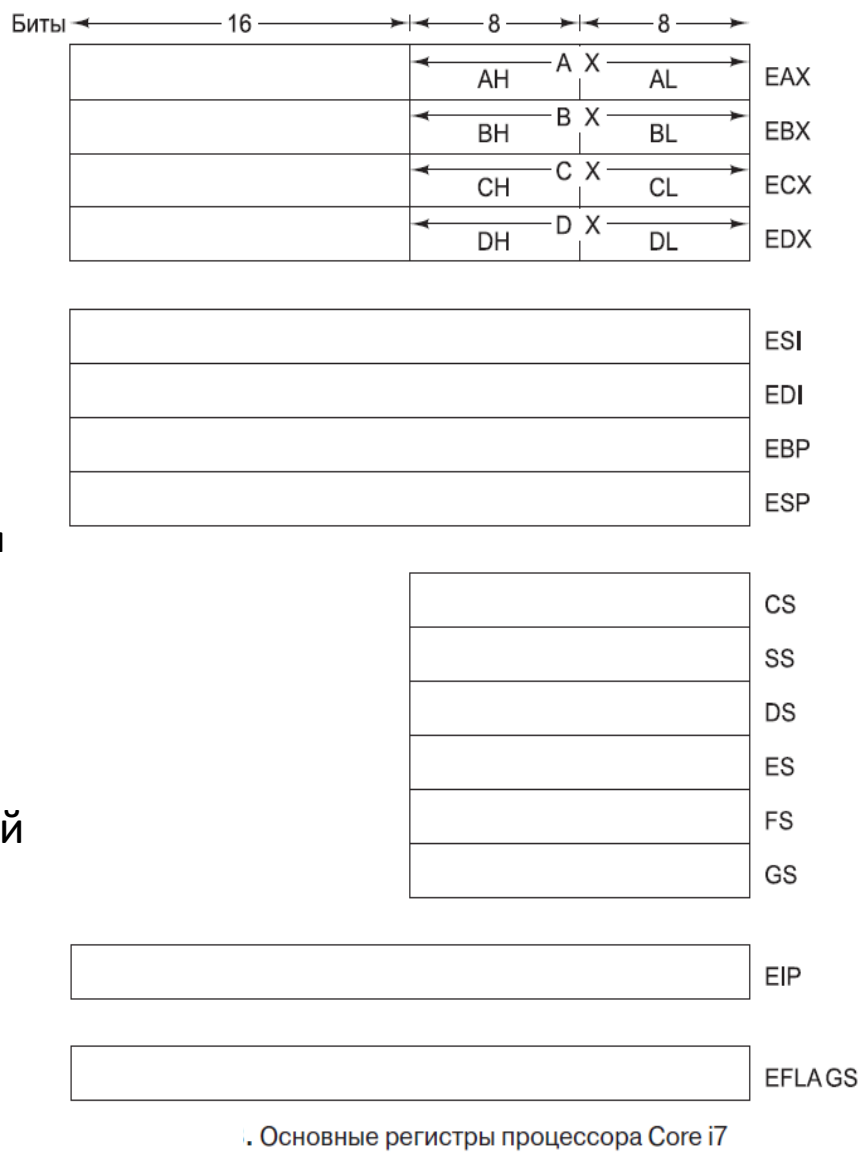
Стек

- Стек служит для загрузки каких-то данных в процессе работы, например при вызове прерывания в стек помещают значения счетчика команд и последних операндах и результате работы АЛУ.
- **Ширина стека** называется размер элементов, которые можно помещать в него или извлекать.
  - Чаще всего ширина стека равна двум байтам 16 бит.
  - Специальный регистр процессора SP (указатель стека) содержит адрес последнего добавленного элемента - вершина стека. Противоположный конец стека называется дном.
  - **Дно стека** находится в верхних адресах памяти.
  - При добавлении новых элементов в стек значение регистра SP уменьшается, то есть стек растёт в сторону младших адресов.



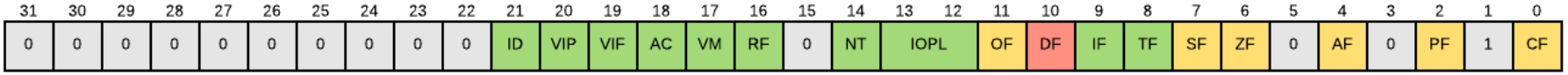
Регистровая память IA-32 (X86).

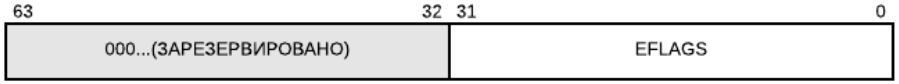
- **EAX, EBX, ECX и EDX – 32 разрядные GPR**
  - (GPR - регистры общего назначения);
  - также допустимы регистры 8 и 16 бит
  - **EAX** — основной арифметический регистр;
    - EBX предназначен для хранения указателей (адресов памяти);
  - **ECX** связан с организацией циклов;
  - **EDX** нужен для умножения и деления
    - вместе с EAX 64-разрядные произведения и делимые.
- **ESI и EDI** - указатели строковых команд:
  - **ESI** указывает на исходную строку,
  - **EDI** — на целевую.
- **EBP** предназначен для хранения указателей **(указатель кадра)**.
- **ESP** — это указатель стека
- **EIP** – счетчик команд
- **EFLAGS** — флаговый регистр.
- **CS-GS** сегментные регистры.



. Основные регистры процессора Core i7

Регистр флагов.





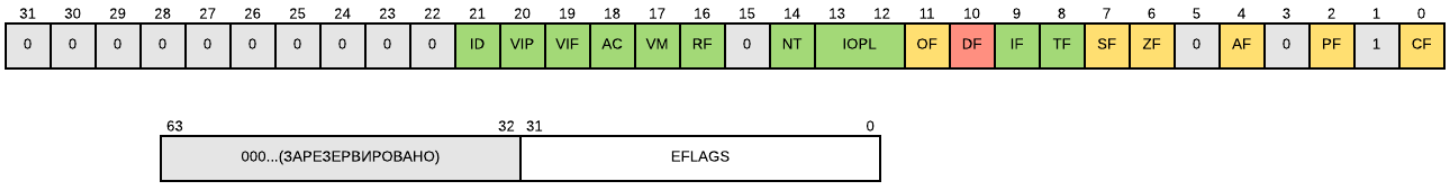
После выполнения очередной команды процессор сохраняет результат выполнения команды в регистре флагов

Регистр флагов служит для индикации процессору о результате выполнения каждой команды или текущем состоянии программы (напр. о прерываниях).

–EFLAG тоже, что и RFLAG но для 32 бит

- **Флаг CF, Carry Flag (бит 0)** (флаг переноса для без знаковых чисел)
- **Флаг PF, Parity Flag (бит 2)** 1 если значение результата АЛУ четное.
- **Флаг AF, Auxiliary Carry Flag (бит 4)** перенос для двоично-десятичных чисел
- **Флаг ZF, Zero Flag (бит 6)** 1, если результат последней операции 0
- **Флаг SF, Sign Flag (бит 7)** 1, если результат последней операции <0
- **Флаг OF, Overflow Flag (бит 11)** 1, если переполнение (перенос знаковых чисел)
- **Флаг DF, Direction Flag (бит 10)** 1 авто-декремент (обрабатывать строки от верхнего адреса к нижнему), 0 - наоборот .

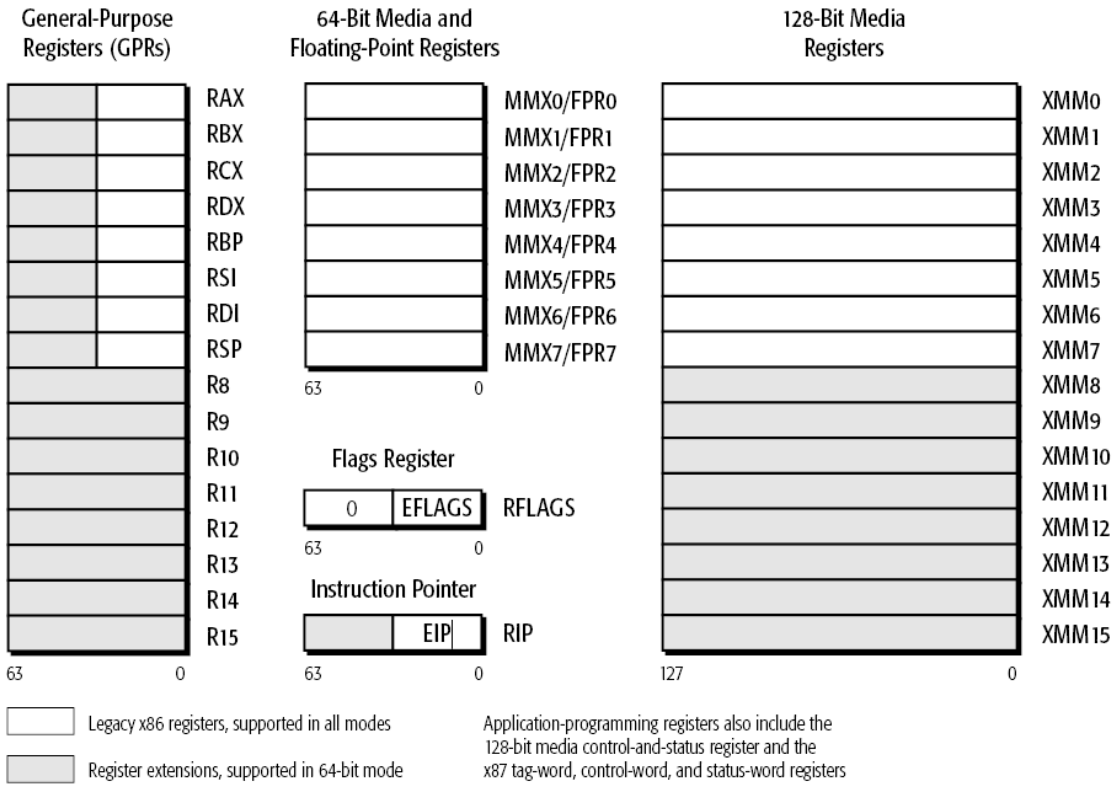
Регистр флагов. Системные флаги



- Системные флаги - **не должны изменяться прикладными программами.**
- **Флаг TF, Trap Flag (бит 8)** 1 для дебага step-to-step.
- **Флаг IF, Interrupt enable flag (бит 9)** 1 если процесс в прерывании
- **Поле IOPL, I/O privilege level field (биты 12 и 13)** уровень приоритета обрабатываемого устройства В-В
- **Флаг NT, Nested task flag (бит 14)** 1 если текущая команда связана с предыдущей.
- **Флаг RF, Resume Flag (бит 16)** 1 исключения в режиме дебага.
- **Флаг VM, Virtual-8086 mode flag (бит 17)** 1 для виртуальной совместимости с 8086.
- **Флаг AC, Alignment check (or access control) flag (бит 18)** режим выравнивания бит
- **Флаг VIF, Virtual interrupt flag (бит 19)** доп. Флаг порываний для предотвращения конфликта прерываний с разными приоритетами.
- **Флаг VIP, Virtual interrupt pending flag (бит 20)** 1 если прерывание ожидает (напр. Окончания другого прерывания).
- **Флаг ID, Identification flag (бит 21)** поддержка режима получения инф. О процессоре.

Регистровая память AMD64 (X64)

- 16 регистров общего назначения 64-битные (RAX-RDX~=EAX-EDX);
- 8 128-битных XMM регистров (SSE команды);
- 8 64-битных регистров MMX (3D Now команды)
- **специальный режим "Long Mode":**
  - до 64-бит виртуальных адресов;
  - 64-битные счетчик команд (RIP);
  - 64 битный регистр флагов RFLAGS



AMD64 имеет 3 режима доступа к памяти:

- реальный режим,
- защищённый режим
- 64-разрядный режим, или long mode

Виды команд X86-X64

- **Команды общего назначения. Основные x86 целочисленные команды.**
  - Большинство из них предназначены для загрузки, сохранения, обработки данных, расположенных в регистрах общего назначения или памяти. Некоторые из этих инструкций управляют потоком команд, обеспечивая переход к другому месту в программе.
- **x87 команды. Обрабатывают данные в x87 регистрах (FUP сопроцессор).**
  - Предназначены для работы с плавающей точкой в x87 приложениях.

Виды команд X86-X64

- **128-битные медиа-команды. SSE, SSE2 и SSE3 (streaming SIMD extension).**
  - Команды предназначенные для загрузки, сохранения, или обработки данных, расположенных в 128-битных XMM регистрах.
  - Команды выполняют операции целочисленные или с плавающей точкой над векторными (упакованными) и скалярными типами данных.
  - Векторные инструкции могут независимо выполнять одну операцию над множеством данных (SIMD) командами.
  - Векторные команды используются для медиа- и научных приложений для обработки блоков данных.
- **64-битные медиа-команды. Multimedia extension (MMX) и 3DNow! команды**.
  - Команды сохраняют, восстанавливают и обрабатывают данные, расположенные в 64-битных MMX регистрах.
  - Команды выполняют операции целочисленные и с плавающей точкой надо векторными (упакованными) и скалярными данными как и XMM.

Виды адресов X86-X64

- **Физический адрес** – это адрес в системной памяти компьютера, именно тот адрес, который выставляется на шину адреса.
- **Логический адрес** – адрес с указанием сегмента в формате – «сегмент:смещение»
  - сегмент указывается в сегментном регистре (cs, ds, ss) или непосредственно значением (это значение может быть только 16-битным), а адрес – в обычном регистре или непосредственно значением (это значение может быть 16-, 32-, 64-битным в зависимости от режима).
  - способ преобразования логического адреса в физический зависит от режима процессора.

Виды адресов X86-X64

- **Линейный адрес** – адрес полученный после преобразования логического адреса
  - После преобразования адреса получается абсолютный 20-, 32-, 64-битный адрес (в зависимости от режима); этот адрес называется линейным.
  - В режиме реальных адресов физический адрес сразу выставляется на шину адреса.
- **Виртуальный адрес** – линейный адрес, полученный в 64 совместимом режиме при помощи механизма трансляции. (Механизм задается ОС, при отсутствии механизма виртуальный адрес = линейный).

Страничная организация виртуальной памяти

- **Страничная организация памяти** - способ организации виртуальной памяти, при котором единицей отображения (трансляции) виртуальных адресов на физические является регион постоянного размера (т. н. страница).
- Типичные размеры страницы — 4 кБ и 4 Мб, 2 Мб, 1 ГБ (long-mode).
  - **Страницы по 4 кБ объединены в таблицу страниц (1024 таблиц), таблицы объединены в каталог страниц(1024 таблицы).**
  - *Страницы по 4 МБ таблицы объединяются в каталог страниц.*
    - Адрес ячейки = 10(каталог)+10(таблица)+12=32 бита.

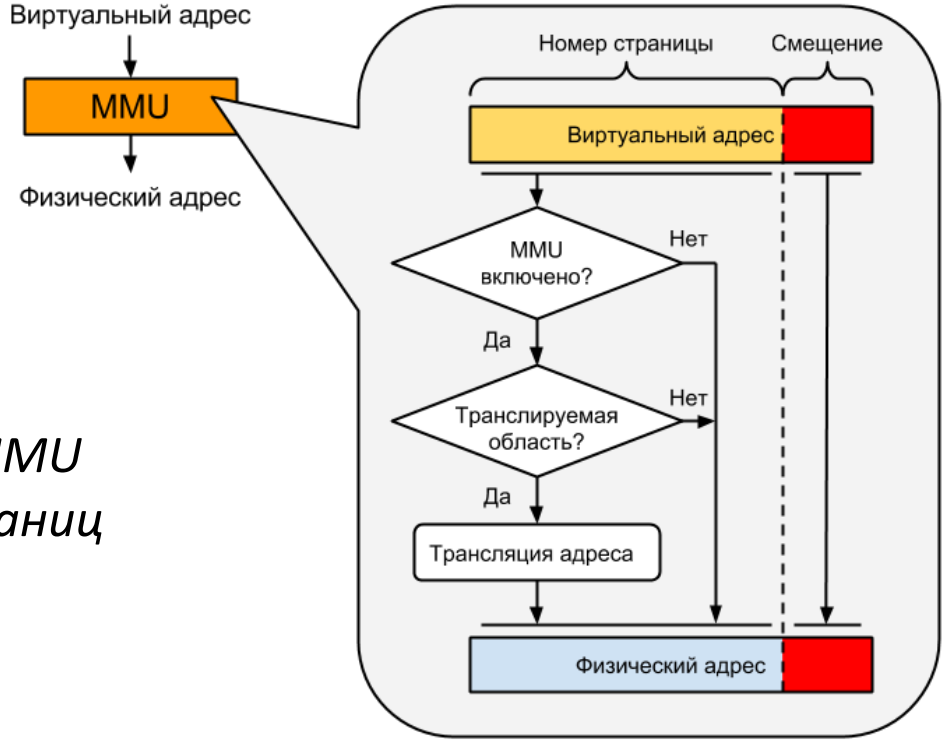
- **Виртуальный адрес страницы и физический могут не совпадать.**
  - *у одной страницы могут быть разные виртуальные адреса и один физический.*
- При отсутствии ресурсов в ОЗУ часть страниц могу быть сброшены на жесткий диск пока память в ОЗУ не будет освобождена
  - Такая область памяти на жёстком диске называется - файл подкачки.

Управление виртуальной памятью

Устройство управления памятью –MMU – транслирует виртуальные адреса в физические.

- Трансляция полностью аппаратная.
- Виртуальная память как правило страничная.
- *Трансляция адресов осуществляется через специальную **кэш память TLB**.*
  - Данная память хранит адреса часто обращаемых виртуальных страниц.

- *TLB вектор содержит сопоставления физических и виртуальных адресов для недавно использовавшихся страниц и атрибуты защиты каждой страницы*
- *Если адреса нет в TLB, то модуль MMU ищет адрес по всем таблицам страниц каждого процессора.*



Режимы работы процессоров

Аппаратные средства телекоммуникационных систем. Модель памяти процессоров

Режимы работы процессора.

Основные режимы

– **Реальный режим** (при включении, 64 кБ RAM) и виртуальны 8086 для совместимости с 16 битными приложениями.

– **Защищенный режим** (32 битный, 4 ГБ RAM) и **расширенный** (64 ГБ RAM)

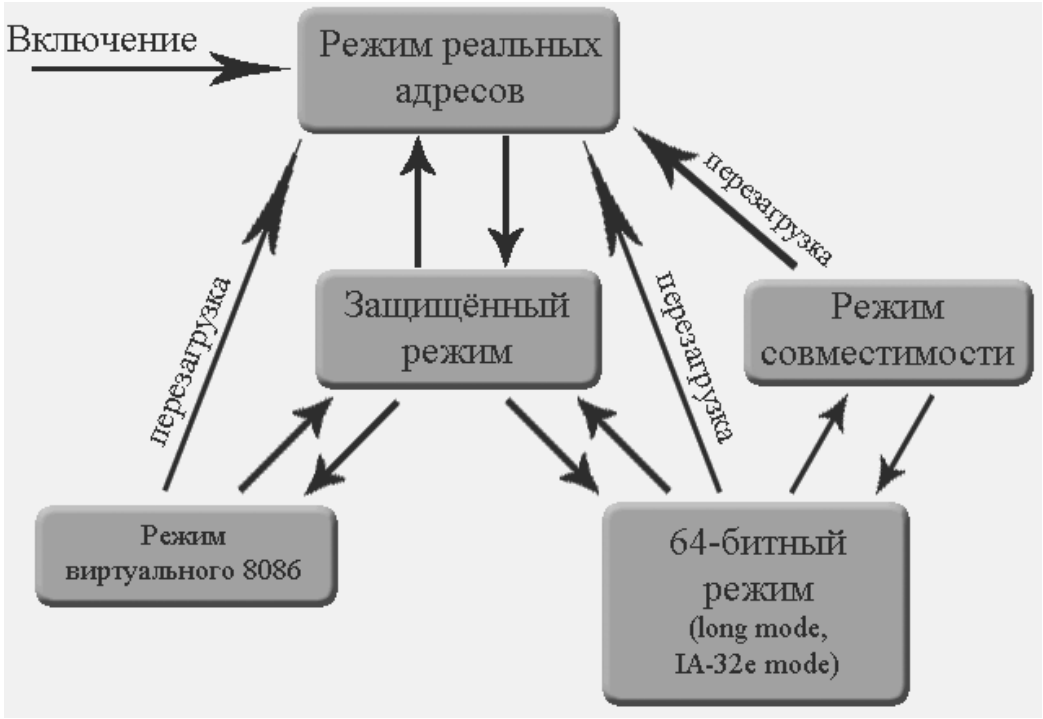
– **Длинный режим** (64 битный, 256 ТБ RAM) и **режим совместимости**.

Режимы отличаются:

- методом и уровнями доступа к памяти,
- Допустимым объемом памяти
- Набором команд
- Размером слова

Защищенный режим - доступ к 4 ГБ виртуальной памяти.

Расширенный режим - доступ к 64 ГБ памяти.



Режимы работы x86-x64. Реальный режим

- **Реальный режим** – это режим, в который переходит процессор после включения или перезагрузки.
  - Стандартный 16-разрядный режим,
  - доступно 1 Мб физической памяти и возможности процессора почти не используются
  - все адреса, к которым обращаются программы, являются физическими, т. е. без какого-либо преобразования будут выставлены на шину адреса.
  - Размер слова 2 байта (WORD);
  - Вся память делится на сегменты размером 64 Кб;
    - *физический_адрес = сегмент * 10h + смещение*
    - Смещение 16-бит значение в РОН или const.
    - Первые 1024 байт заняты таблицей порываний (256 адресов подпрограмм прерываний) с нулевого адреса.

Режимы работы x86-x64. Защищенный режим

- **Защищённый режим (protected mode, или legacy mode (AMD))**– это 32-разрядный режим (X86);
  - доступ к 4-гигабайтному физическому адресному пространству,
    - *при включении механизма трансляции адресов можно получить доступ к 64 Гб физической памяти.*
    - Размер слова 4 байта, или двойное слово (*DWORD double word*).
  - Все операнды, которые выступают как адреса, 32-битные;
  - **За работу защищенного режима отвечает операционная система (ОС);**
  - Защищённый режим называется так потому, что позволяет защитить данные операционной системы от приложений,
    - Например часть памяти резервируется по привилегированные данные ОС.
    - В режиме возможна многозадачность.
  - *Если в защищённом режиме происходит нарушение условий защиты, то процессор генерирует специальное прерывание – исключение.*

Защищенный режим. Уровни привилегий

- **Уровни привилегий доступа к памяти:**
  - уровень 0: ядро операционной системы;
  - уровень 1: драйверы ОС;
  - уровень 2: интерфейс ОС;
  - уровень 3: прикладные программы.
  - Иногда уровни 0 - 2 могут иметь одинаковые привилегии уровня 0, но уровень 3 в защищённом режиме всегда имеет привилегии ниже чем другие.

– **Программы и данные ограничены внутри своих уровней привилегий.**

— **Каждый уровень привилегий имеет свой сегмент в памяти**

– Каждый сегмент и объект, который управляется процессором описываются в области памяти – дескрипторе,

— Дескриптор описывает: адреса сегмента, уровни и особенности доступа.

— Все дескрипторы объединены в **таблицу дескрипторов**.

Режимы работы x86-x64. Защищенный режим. Дескрипторы

- **Глобальные дескрипторы** – (таблица *GDT, ее размер в регистре* GDTR).
  - —**Селектор** – это индекс дескриптора в GDT.
- **Локальные дескрипторы (таблица LDT, размер в LDTR) -** дескрипторы, создаваемые ОС под каждый процесс .
  - *Главное отличие LDT от GDT - в ней нельзя определять дескрипторы системных объектов – объектов, которые использует процессор.*
- **Таблица векторов прерываний** (**таблица IDT, размер в IDTR**)- дескрипторы адресов и настроек (шлюз порывания) .
  - —Адрес соответствует подпрограмме прерываний (действиям в ответ на вызов прерывания, напр. обработка нажатия клавиши на клавиатуре).
  - —Прерывания могут быть аппаратные, исключения и программные.
  - —Прерывания находят на 0 уровне привилегий, поэтому обращение к ним через «шлюз».

Режимы работы x86-x64. Длинный режим

- **long mode («длинный режим», или IA-32e по документации Intel)** – это 64-разрядный режим.
  - По принципу работы он почти полностью сходен с защищённым режимом.
  - В 64-разрядный режим можно перейти только из защищённого режима.
  - Размеры слов - это двойное слово (DWORD), но можно оперировать данными размером в 8 байт (QWORD). Размер адреса всегда 8-байтовый.

Дополнительные режимы x86-x64

- **Режим системного управления** (System Management Mode) - режим в который процессор переходит при получении специального прерывания SMI.
  - Режим предназначен для выполнения некоторых действий с возможностью их полной изоляции от прикладного программного обеспечения и даже операционной системы. Например использоваться для реализации системы управления энергосбережением компьютера или функций безопасности и контроля доступа.
  - Переход в этот режим возможен только аппаратно.
- **Режим виртуального процессора 8086** – это подрежим защищённого режима для поддержки старых 16-разрядных приложений.
  - Его можно включить для отдельной задачи в многозадачной операционной системе защищённого режима;
- **Режим совместимости для long mode.** В режиме совместимости приложениям доступны 4 Гб памяти и полная поддержка 32-разрядного и 16-разрядного кода;
  - Размер слова - двойное. Размер адреса 32- битный, а размер операнда не может быть 8-байтовым.
  - Режим совместимости можно включить для отдельной задачи в многозадачной 64- битной операционной системе.

Сравнение работы процессоров в защищенном и длинном режимах

Operating Mode		Operating System Required	Application Recompile Required	Defaults		Register Extensions	Typical
				Address Size (bits)	Operand Size (bits)		GPR Width (bits)
Long Mode	64-Bit Mode	New 64-bit OS	yes	64	32	yes	64
	Compatibility Mode		no	32		no	32
				16	16		16
Legacy Mode	Protected Mode	Legacy 32-bit OS	no	32	32	no	32
				16	16		
	Virtual-8086 Mode			16	16		16
	Real Mode	Legacy 16-bit OS					

Register or Stack	Legacy and Compatibility Modes			64-Bit Mode		
	Name	Number	Size (bits)	Name	Number	Size (bits)
General-Purpose Registers (GPRs)	EAX, EBX, ECX, EDX, EBP, ESI, EDI, ESP	8	32	RAX, RBX, RCX, RDX, RBP, RSI, RDI, RSP, R8–R15	16	64
128-Bit XMM Registers	XMM0–XMM7	8	128	XMM0–XMM15	16	128
64-Bit MMX Registers	MMX0–MMX7	8	64	MMX0–MMX7	8	64
x87 Registers	FPR0–FPR7	8	80	FPR0–FPR7	8	80
Instruction Pointer	EIP	1	32	RIP	1	64
Flags	EFLAGS	1	32	RFLAGS	1	64
Stack	–		16 or 32	–		64

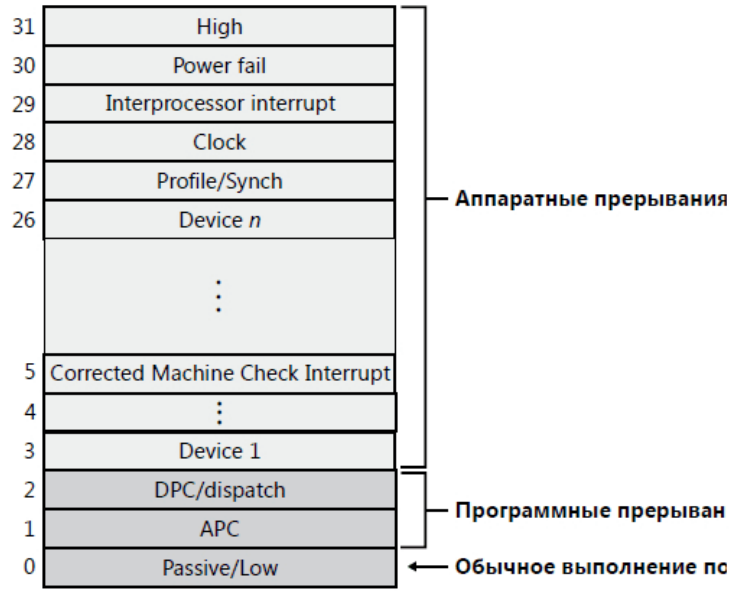
Особенности системы прерываний

Аппаратные средства телекоммуникационных систем. Модель памяти процессоров

Модель памяти X86-X64, Прерывания

Прерывание (от англ. interrupt) - это *прекращение выполнения текущей команды или текущей последовательности команд для обработки некоторого события специальной программой – обработчиком прерывания, с последующим возвратом к выполнению прерванной программы.*

Прерывание используется для быстрой реакции процессора на особые ситуации, возникающие при выполнении программы и взаимодействии с внешними устройствами.



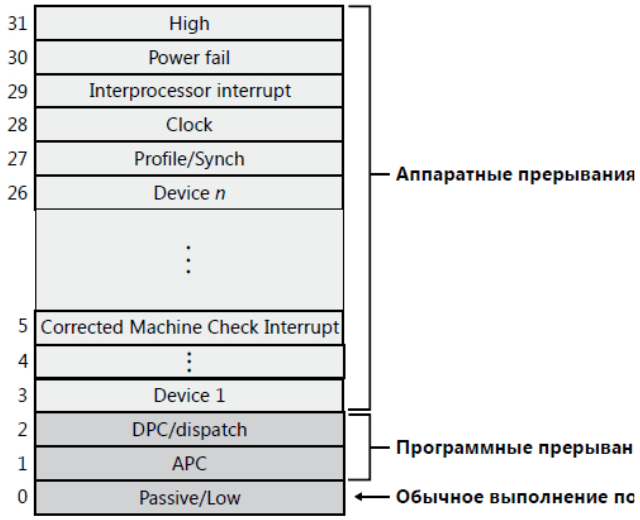
Модель памяти X86-X64, Прерывания

3 типа прерываний процессора:

- Аппаратные (по запросу от внешних устройств, напр. контроллер USB)
- Программные (по программному запросу, напр. обработка клавиатуры)
- Исключения (ошибки и аварии)

- Прерывания могут иметь свои приоритеты (в случае одновременного вызова двух прерываний)

- При вызове прерывания основная программа останавливается, ее параметры (напр. номер последней команды) записываются в стек.
  - После этого вызывается программа прерывания
  - После окончания программы прерывания основная программа возобновляется, данные выгружаются из стека



Модель памяти X86-X64.
Защищённый режим. Модель шлюзов

В защищенном режиме прерывания находятся в таблице прерываний и описываются шлюзами.

1. Шлюз задачи (Шлюз задачи может находиться в любом декрипторе, служит для перехода между разными уровнями).
2. Шлюз прерывания (аппаратные и программные прерывания).
3. Шлюз ловушки (для отладки ПО).

Переход между шлюзами осуществляется при помощи уровней стеков.
Для перехода
Между шлюзами
Каждая задача должна иметь свои стеки на каждый уровень.

Особенности организации многозадачности

Аппаратные средства телекоммуникационных систем. Модель памяти процессоров

Модель памяти X86-X64. Особенности многозадачного режима работы

- *Многозадачность* – способность процессора выполнять несколько задач на уровне ОС.
  - Основная единица измерения в многозадачной системе – это задача, или поток (в дальнейшем – задача).
- *Задача* – это самостоятельная последовательность команд (программа), которая выполняется в своём окружении.
  - Основные параметры, которыми характеризуется окружение задачи
  - состояние регистров общего назначения,
  - состояние сегментных регистров
  - адресное пространство (если операционная система обеспечивает изолированность задач друг от друга), характеризуется регистром CR3, а также состоянием регистров математического сопроцессора.
- Многозадачность реализуется путём быстрого переключения задач.
- На процессорах x86 многозадачность реализуется аппаратно, при помощи процессора
- Для описания каждой задачи используется область памяти, где хранятся все её параметры (как состояние её регистров и данные). *сегментом состояния задачи*, или Task State Segment (TSS).

Модель памяти X86-X64.
Особенности многозадачного режима работы

- На процессорах x86 многозадачность реализуется аппаратно, при помощи процессора:
  - Процессор сохраняет состояние прерываемой задачи в специально отведённой области памяти
  - процессор запускает очередную задачу, предварительно восстановив ее состояние из такой же области памяти.
  - У каждой задачи может быть 4 стека для каждого из уровней привилегий.
  - Для каждой задачи может использоваться своё виртуальное адресное пространство, (напр. свой каталог страниц),
    - это позволяет обеспечить изолированность каждой задачи от других и от памяти системы и тем самым повысить защищённость операционной системы.

Модель памяти X86-X64.
Особенности многозадачного режима работы

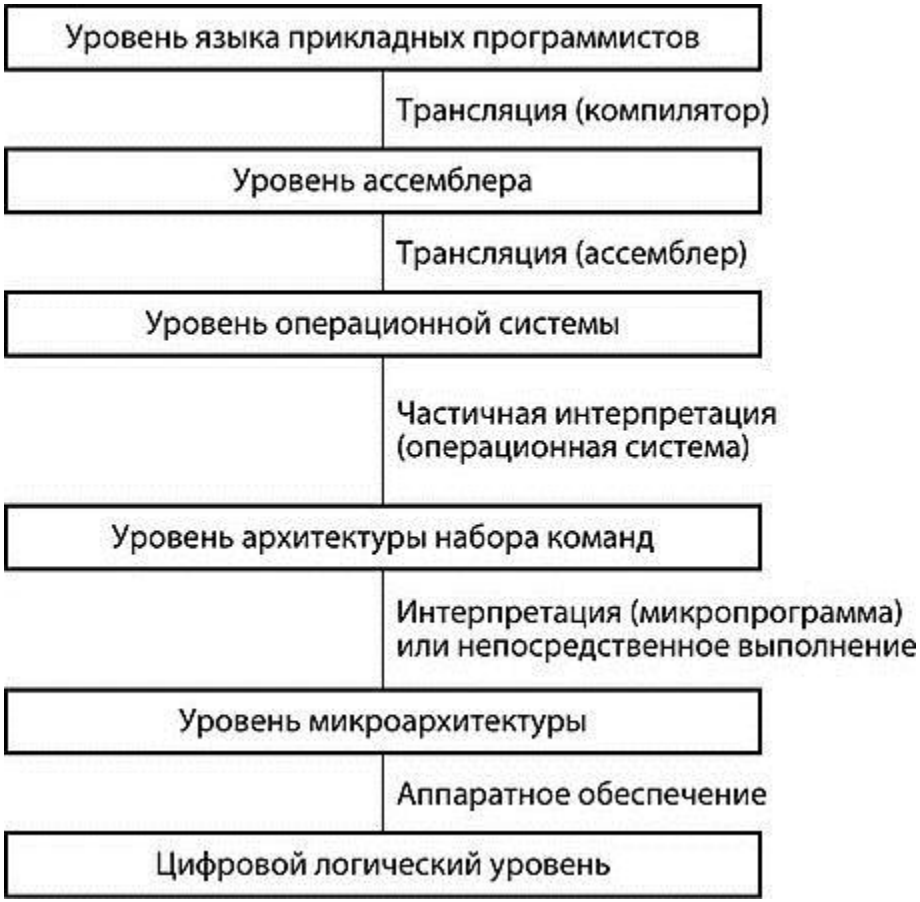
- *Карта разрешения ввода-вывода* – список разрешенных и защищенных портов ввода-вывода для каждой задачи.
- Каждая задача может использовать свою таблицу, локальную дескрипторную таблицу (LDT).
- *Регистр TR (Task Register)* – 16-разрядный регистр, содержит селектор TSS текущей задачи, бит перевода процессора в режим многозадачности.

Программный уровень работы с процессором

Аппаратные средства телекоммуникационных систем. Модель памяти процессоров

Уровень архитектуры набора команд

- **Программы, написанные на различных языках высокого уровня, должны транслируются в общую** для всех промежуточную **форму** — уровень архитектуры набора команд;
- *Аппаратное обеспечение ориентируется на непосредственное выполнение программ этого уровня.*
- Уровень архитектуры набора команд связывает компиляторы и аппаратное обеспечение. Это язык, который понятен и компиляторам, и устройствам.



Языки Assembler.

Уровень определяется моделью памяти, набором регистров, поддерживаемыми типами данных (напр. Float), поддерживаемыми командами (напр. AMD64).

Ассемблер (Assembly) — язык программирования, понятия которого отражают архитектуру электронно-вычислительной машины.

Достоинства языка: скорость работы и трансляции в машинный код, приближенность к архитектуре процессора, что позволяет оптимизировать ПО под каждый процессор.

Недостатки: низкий уровень абстракции, увеличенное время разработки по сравнению с ЯП высокого уровня, непереносимость кода, требуется знание архитектур каждого процессора.

Языки Assembler.

Язык ассемблера — символьная форма записи машинного кода, использование которого упрощает написание машинных программ.

- Язык ассемблера обеспечивает доступ к регистрам, указание методов адресации и описание операций в терминах команд процессора.
- **Для одной и той же ЭВМ могут быть разработаны разные языки ассемблера.**
- *Для идеального микропроцессора, у которого система команд точно соответствует языку программирования, ассемблер вырабатывает по одному машинному коду на каждый оператор языка.*
  - **На практике для реальных микропроцессоров может потребоваться несколько машинных команд для реализации одного оператора языка.**
- Ассемблер может содержать средства более высокого уровня абстракции:
  - *встроенные и определяемые макрокоманды, соответствующие нескольким машинным командам,*
  - *средства описания структур данных.*

Языки Assembler. Практическое применение

1. **Требование к низкому объему памяти и высокой скорости исполнения кода**
   - программы-загрузчики,
   - встраиваемое программное обеспечение (драйвера),
   - программы для микроконтроллеров и процессоров с ограниченными ресурсами, вирусы,
   - программные защиты;

Языки Assembler. Практическое применение

2. **Быстродействие и оптимизация кода**

вставки в «узкие места ПО» программы, написанные на языке ассемблера выполняются гораздо быстрее, чем программы-аналоги, написанные на языках программирования высокого уровня абстракции.

Быстродействие зависит от оптимизации работы под конкретную модель процессора, реальный конвейер на процессоре, размер кэша, тонкости работы операционной системы.

В результате, программа начинает работать быстрее, но теряет переносимость и универсальность.