**ВОПРОС 21**

Сегментация памяти i8086

**Сегментные регистры**

Сегментный регистр 8086 - рассматривали ранее где-то в 9 лк.

16-битная x86-архитектура, благодаря наличию четырёх сегментных регистров, позволяет одновременно иметь доступ к четырём сегментам памяти. Назначение сегментных регистров:

DS (data segment) — сегмент данных;

CS (code segment) — сегмент кода;

SS (stack segment) — сегмент стека;

ES (extra segment) — дополнительный сегмент.

Каждый из этих регистров 16-ти разрядный. И каждый из них содержит начальный адрес соответствующего сегмента. В процессе работы программы содержимое сегментого регистров можно менять и соответственно переходить к другому сегменту, это не означает, что весь код должен быть в одном сегменте кода.

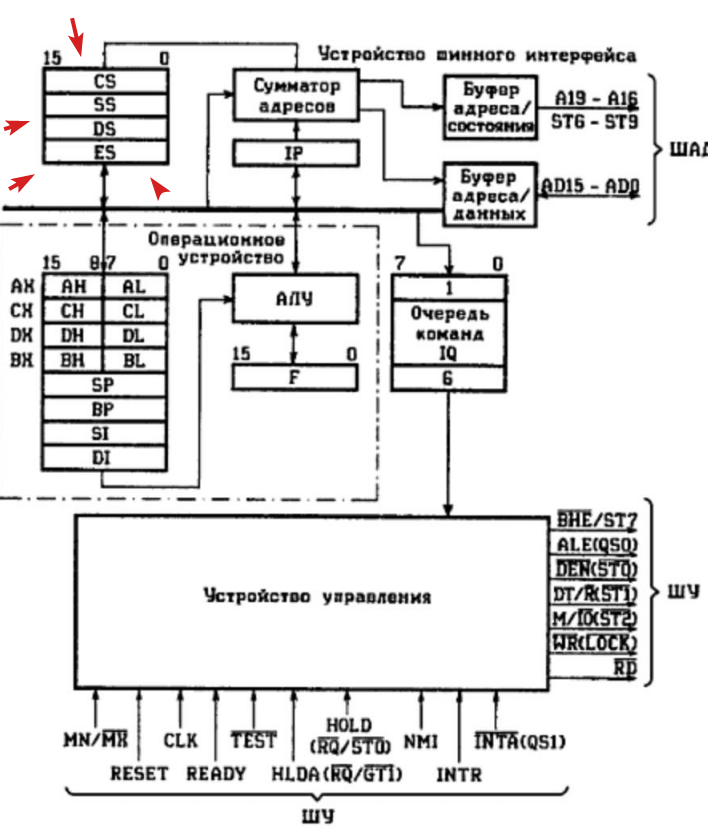
Логический адрес на такой платформе принято записывать в виде сегмент:смещение, где сегмент и смещение задаются в шестнадцатеричной системе счисления.

В реальном режиме для вычисления физического адреса байта памяти происходит сдвиг влево на 4 разряда значения соответствующего сегментного регистра, а затем добавляется смещение.

Адресных линий 20 в 8086. А регистры 16-иразрядные. Чтобы получить 20-иразрядное значение, мы сдвигаем на 4 разряда влево, а затем добавляем смещение.

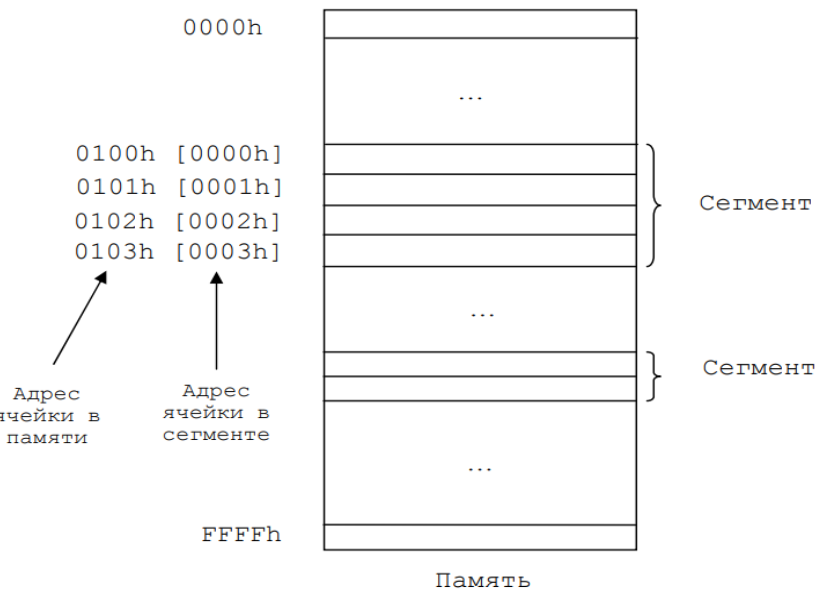
Например, логический адрес 7522:F139 дает 20-битный физический адрес: 75220 + F139 = 84359

**Внутренняя структура**



Сегменты в левой верхней части, содержимое этой верхней части подаются в сумматор адресов, куда также подаются данные с внутренней шины и содержимое регистра IP, который является счетчиком команд. Сумматор адресов - это то устройство, которое осуществляет сложение со сдвигом. Именно это устройство производит сложение со сдвигом и формирует из 16-ти разрядных регистров 20-ти разрядные адреса. Передает его в буфер.

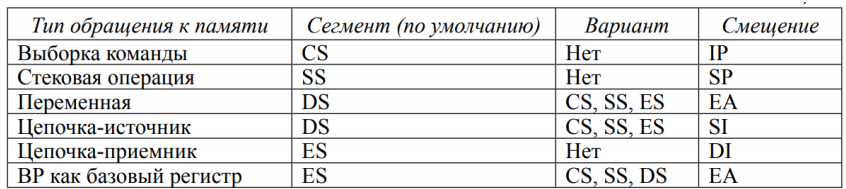
**Пример организации памяти**



Память представляет из себя некий массив, который делится на сегменты. Сегментов может быть множество. Внутри сегментов существует своя адресация, которая определяется смещением и идет с нулевой точки (адреса сегмента).

Для команд, для данных и для стека используется различные комбинации сегментных регистров и различные операции смещения.

**Источник логического адреса**

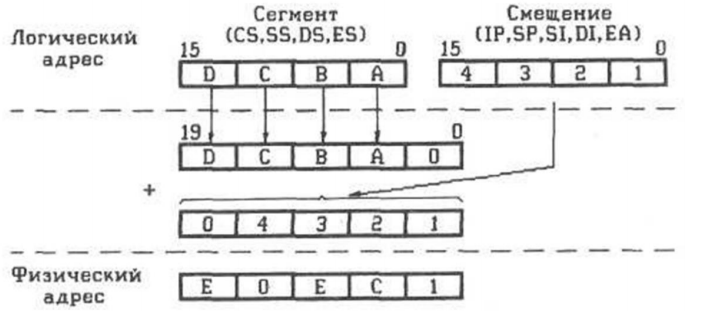


ЕА – эффективный адрес, вычисляемый в соответствии с заданным способом адресации

При декодирования команды определяется смещение. Меняя комбинации регистров, мы можем передать управление в любую точку команды. Все стековые операции выполняются с регистром стека и указателя стека регистра SP.

С данными можно использовать любой сегментный регистр.

**Пример формирования адреса**



На схеме сегментный регистр 16-разрядный.

На первом шаге сдвигаем содержимое сегментного регистра влево на 4 разряда, справа появляется 4 нуля.

После прибавляем смещение и получаем физический адрес. Это значение и будет передано по шине адреса.

**Особенности сегментации памяти в микропроцессоре i8086**

- Сегменты памяти определяются только сегментными регистрами. (какое значение запишем в сегментный регистр, там и будет начинаться сегмент памяти)

- Начальный адрес сегмента кратен 16. (Что бы мы ни написали в сегментный регистр, всё равно начальный регистр будет кратен 16, потому что это содержимое регистра сдвигается на 4 и справа всегда появляется 4 нуля)

- Никаких средств проверки правильности использования сегментов нет. (мы сами определяем сегментацию, но только цикловая структуры памяти и цикловая структура сегментов не даёт выходить за пределы памяти)

- Размещение сегментов в памяти достаточно произвольно. (сами определяем или компилятор)

- Сегменты могут частично или полностью перекрываться, или не иметь общих частей. (одинаковое содержимое записать в сегментный регистр стека и кода, тогда сегмент стека совпадёт с сегментов кода - ничего хорошего, но это возможно, размер сегмента стека 64 кб)

- Программа может обращаться к любому сегменту как для считывания, так и для записи данных и команд. (можно менять на ходу содержимое программы)

- Для защиты памяти от несанкционированного доступа других программ требуются специальные "внешние" схемы или программы (ну или самому написать в коде).

- Система не делает различий между сегментами данных, кода и стека. (на выходе формируется 20-иразрядный физический адрес, не важно куда обращается)

- Нет никаких препятствий для обращения к физически не существующей памяти.

- При обращении к несуществующей памяти результат непредсказуем (все зависит от разработчика аппаратного обеспечения компьютера.)

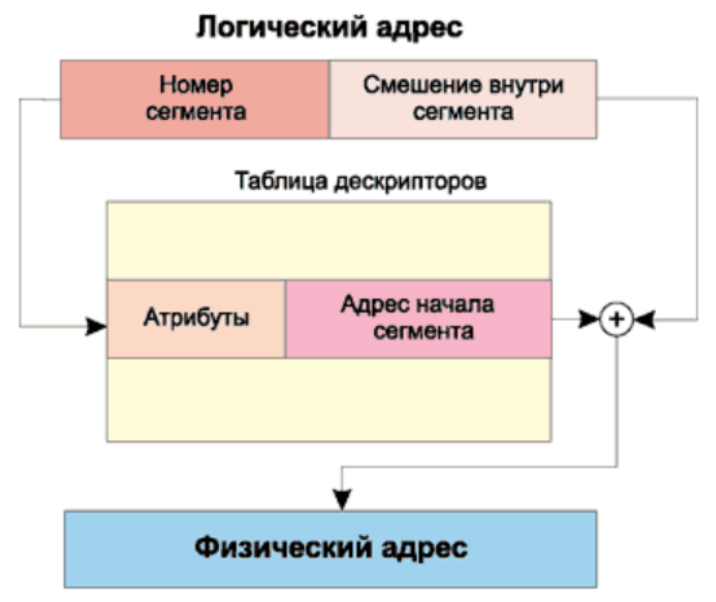
**Защищенный режим**

Появился в процессоре 80286, сохранив понятия сегмента и смещения, и радикально модернизировал механизм сегментации, предложив концепцию логического (виртуального) адреса.

В защищенном режиме работы процессора содержимое сегментных регистров используется не как слагаемое при вычислении адреса, а как индекс или селектор, выбирающий дескриптор сегмента в специальной структуре, называемой дескрипторной таблицей и описывающей свойства каждого сегмента: базовый адрес, размер и атрибуты, связанные с управлением привилегиями и защитой памяти.

По сути, сегментные регистры сохранились, но только при инициализации компьютера, заполняется в памяти таблица, по которой вычисляются физические адреса. Содержание сегментного регистра не является слагаемым при формировании.

**Преобразованию логического адреса в физический**



Номер сегмента используется не напрямую, при формировании физического адреса, а используется для поиска соответствующего дескриптора, который содержит определённые атрибуты, влияющие на защиту, расположение в памяти. Оттуда уже извлекается оттуда соответствующая запись, которая суммируется, но для этого куча проверок проводится.

Используя дескрипторов, можно адресовать больше физической памяти, так как в таблице дескрипторов мы определяем расположение сегментов, у нас уже иная система адресация, которая определяется логически в соответствующей таблице памяти.

Грубо говоря у нас есть таблица, где номеру сегмента соответствует определенный физический адрес.

При запуске там опрашивается что где, и соответственно ответсвтенность с программиста снимается, потому что нельзя обратиться к несуществующей памяти

**Таблица дескрипторов**

Любой процессор х86 (80286) начинает свою работу в реальном режиме. Затем его переключает специальная программа в защищенный режим.

Программа, переключающая процессор в защищенный режим, должна подготовить в памяти управляющие структуры, используемые процессором в этом режиме. Одна из таких структур — таблица дескрипторов сегментов, описывающая свойства адресуемых областей памяти.

При старте платформы и выполнении процедуры POST, обязанности по управлению контекстом процессора возлагаются на BIOS или UEFI, что подразумевает использование таблиц страниц и сегментов, построенных firmware. То есть в современном компьютере, все эти таблицы, с которыми работают прикладные программы готовит BIOS или UEFI. Готовятся таблицы необходимые, опрашивается всё железо, вносится в таблицу, чтобы мы не имели прямого доступа.

При выполнении Legacy-boot (режима совместимости), firmware (программное обеспечение БИОСа или УЕФИ) передает управление загрузчику в 16-битном режиме Real Mode (реального времени). Переключение процессора в Protected Mode и управление его контекстом входит в обязанности ОС. Либо не выполняется, в случае 16-битной системы. Вместе с тем, до момента передачи управления от BIOS к загрузчику ОС, в частности во время инициализации и тестировании памяти, процессор может временно переводиться в Protected Mode процедурами POST.

По сути, после подготовки таблицы, в реальном режиме передаётся управление ОС, а ОС при помощи инструкций переводит процессор в защищенный режим, а затем работает с таблицами. Может не перевести и работать в 16-битном режиме. (БИОС сам может включить защищённый режим и выполнить некоторые операции, но всё равно передаётся управление ОС в реальном времени. Далее ОС сама.

При UEFI-boot, в момент передачи управления от firmware к загрузчику операционной системы, процессор уже работает в Protected Mode, тем не менее, ОС повторно инициализирует CPU (процессор), предварительно создав собственные таблицы дескрипторов и сегментов, прекращая полномочия контекста UEFI, вызовом функции ExitBootServices().

**Виртуальная память**

В 32-битном процессоре 80386, сегментация была дополнена механизмом трансляции страниц или Paging, лежащим в основе организации виртуальной памяти.

Виртуальная память - схема адресации памяти компьютера, при которой память представляется программному обеспечению непрерывной и однородной, в то время как в реальности для фактического хранения данных используются отдельные (разрывные) области различных видов памяти, включая кратковременную (оперативную) и долговременную (жёсткие диски, твердотельные накопители).

Виртуальная память позволяет использовать ЖД в качестве памяти, оперативной, к примеру. Мы обращаемся к линейной структуре (виртуальной), которая может быть представлена в виде системы различных устройств. Уровень абстракции возрастает.