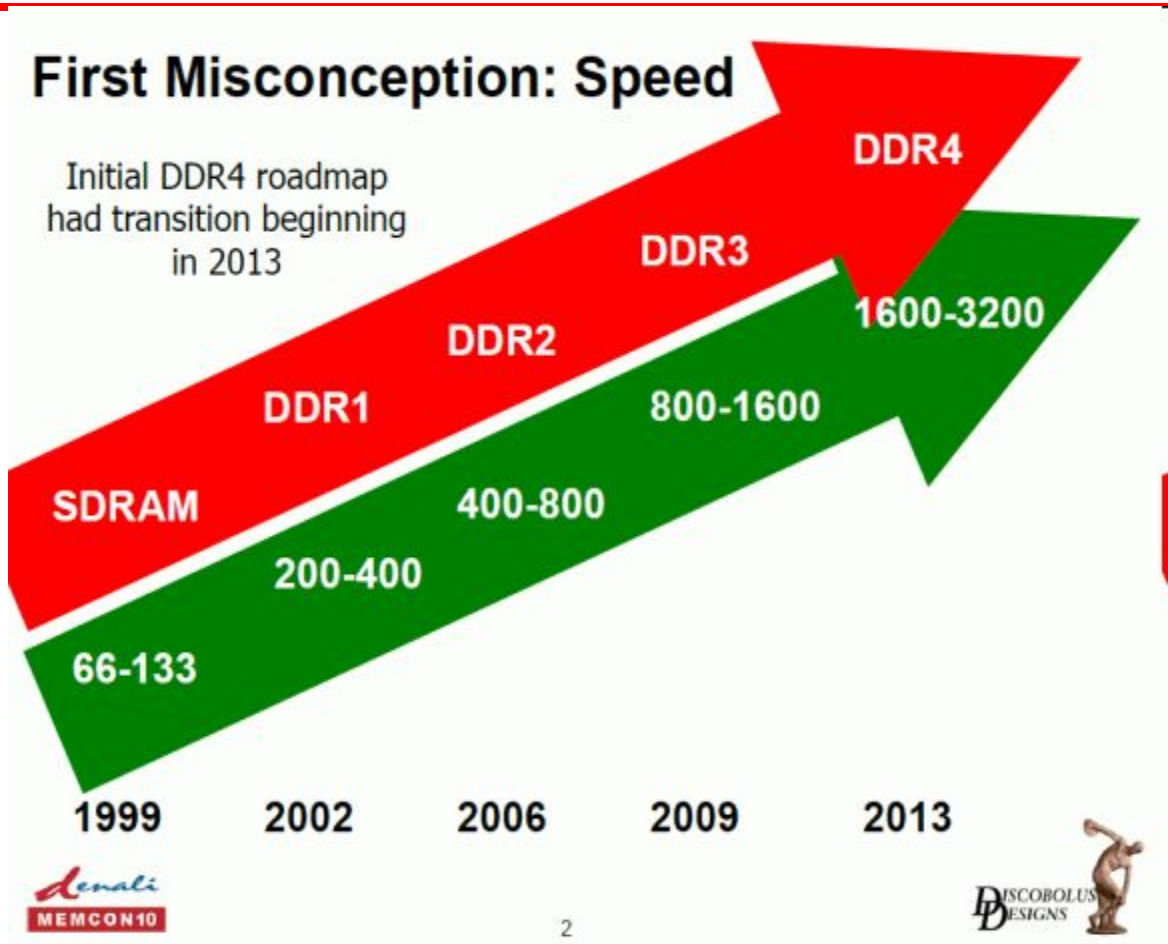
Lection №7

Организация памяти. Модель памяти в реальном режиме. Режимы адресации.



Физическая организация памяти

Основная (главную, оперативную, физическую)

- Основная память представляет собой упорядоченный массив однобайтовых ячеек, каждая из которых имеет свой уникальный адрес (номер).
- Процессор извлекает команду из основной памяти, декодирует и выполняет ее.
- Для выполнения команды могут потребоваться обращения еще к нескольким ячейкам основной памяти.
- Обычно основная память изготавливается с применением полупроводниковых технологий и теряет свое содержимое при отключении питания.

Вторичная (внешнюю) память

- Вторичную память (это главным образом диски) также можно рассматривать как одномерное линейное адресное пространство, состоящее из последовательности байтов.
- В отличие от оперативной памяти, она является энергонезависимой, имеет существенно большую емкость и используется в качестве расширения основной памяти.

Слово (16 бит)

Двойное слово (32 бит)

Байт (8бит)

Учетверённое слово (64бит)

ВСЁ ПРОСТРАНСТВО ПАМЯТИ

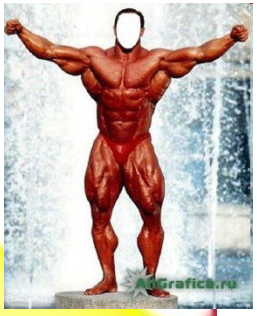
Параграфы
(области из 16 смешных байт)

Сегменты

Единицы памяти

Страницы

Три адресных пространства



Линейное

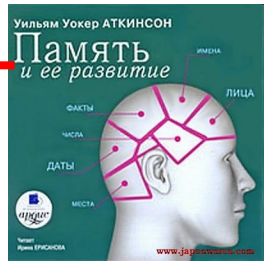
Логическое

Физическое

ЭВУ ruleZZZ!!

ADDRESS MEMORY SPACES

Логическая память



- Аппаратная организация памяти в виде линейного набора ячеек не соответствует представлениям программиста о том, как организовано хранение программ и данных.
- Большинство программ представляет собой набор модулей, созданных независимо друг от друга.
- Иногда все модули, входящие в состав процесса, располагаются в памяти один за другим, образуя линейное пространство адресов.

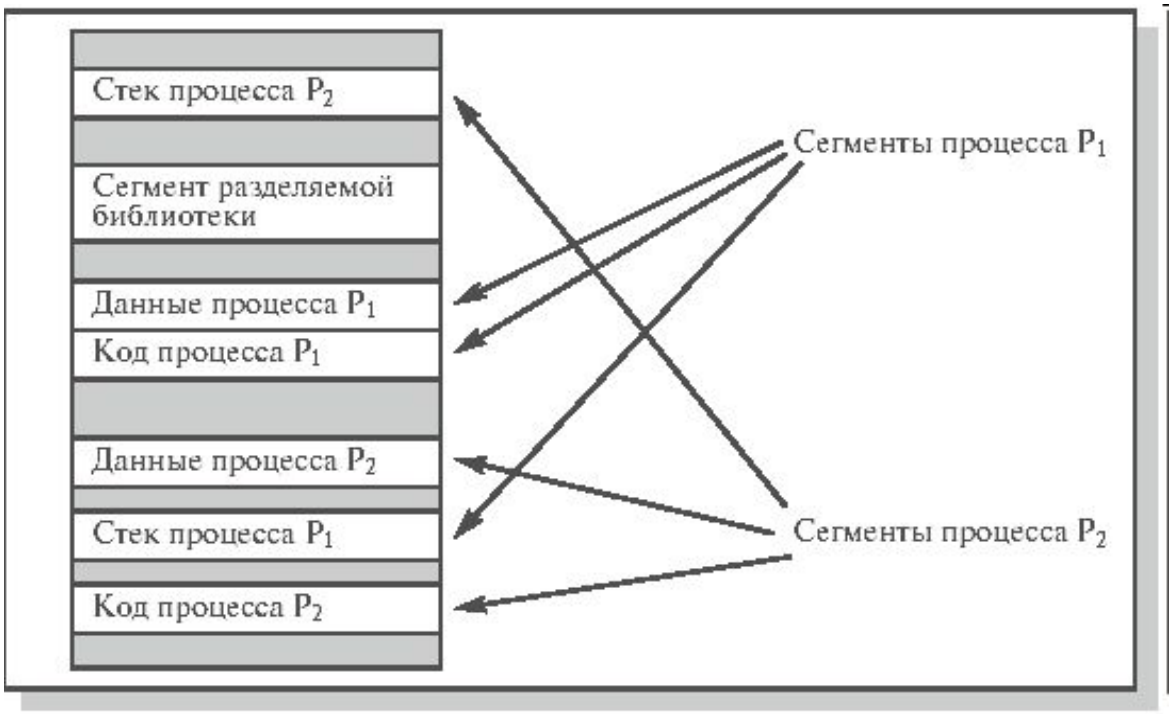
! Однако чаще модули помещаются в разные области памяти и используются по-разному.

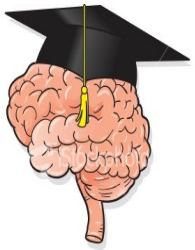
Схема управления памятью, поддерживающая этот взгляд пользователя на то, как хранятся программы и данные, называется сегментацией.

Организация памяти

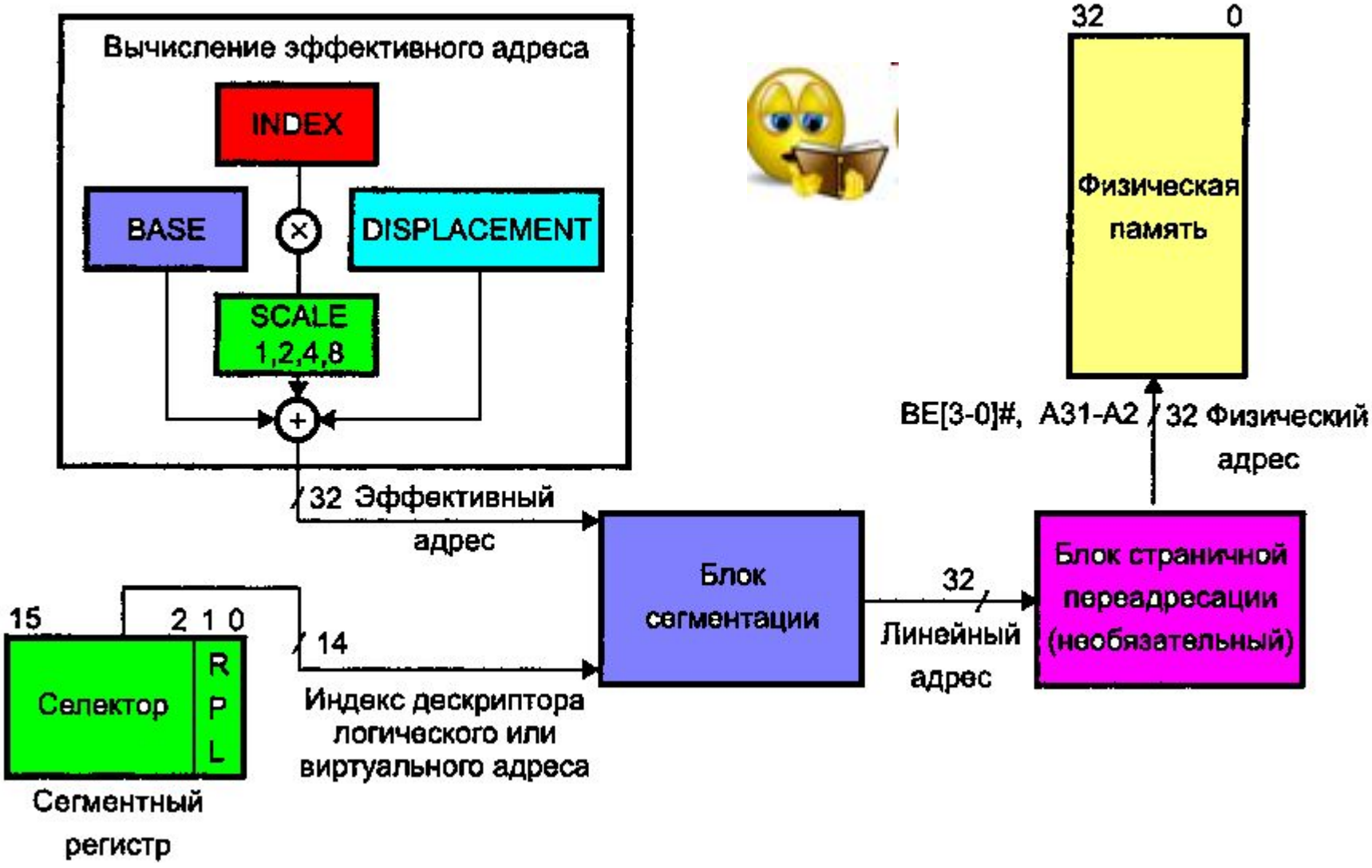
Сегмент – область памяти определенного назначения, внутри которой поддерживается линейная адресация.

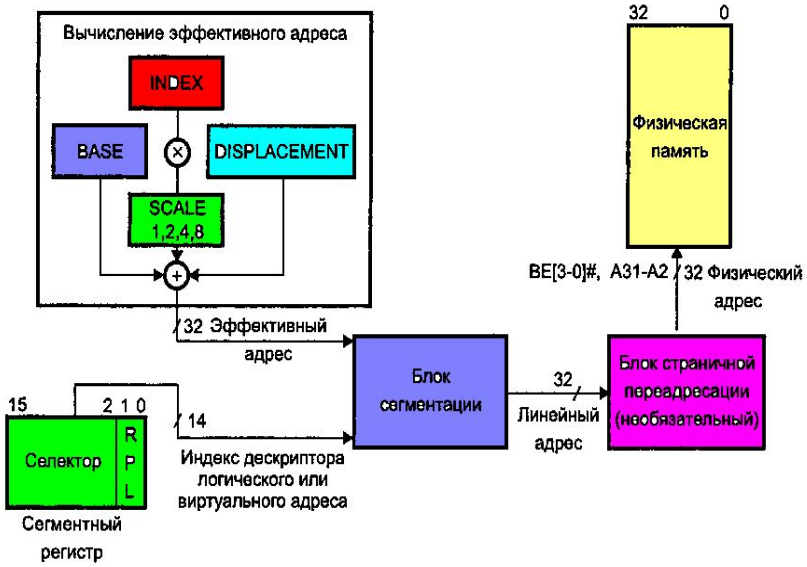
Сегменты содержат процедуры, массивы, стек или скалярные величины, но обычно не содержат информацию смешанного типа.





Организация памяти



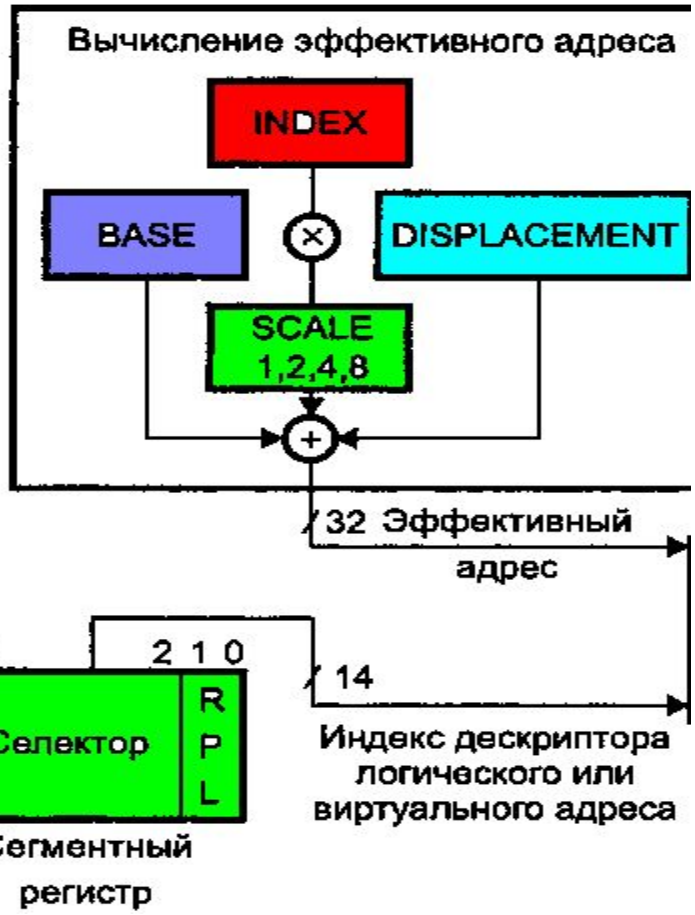


Селектор сегмента хранится в старших 14 битах сегментного регистра (CS,DS,ES,FS,GS), участвующего в адресации конкретного элемента памяти

Логический адрес, также называемый виртуальным, состоит из селектора сегмента (в реальном режиме – просто адреса сегмента) и эффективного адреса, называемого также смещением.

Организация памяти

Эффективный адрес формируется суммированием компонентов base, index, displacement с учётом масштаба scale.



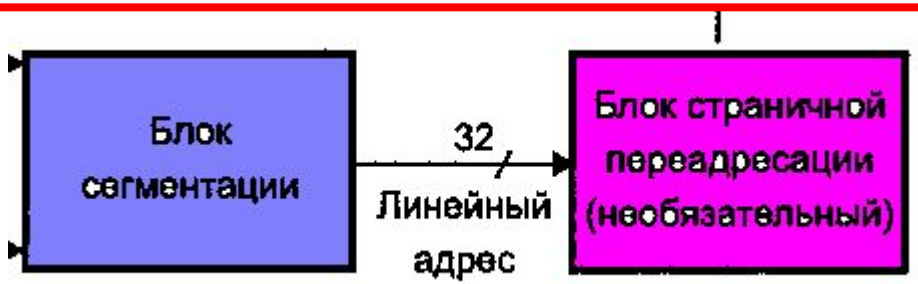
Каждая задача может иметь до 16Кбайт селекторов (2^{14})

X

Смещение, ограниченное размером сегмент, может достигать 4Гбайт

Логическое пространство для каждой задачи может достигать 64 Тбайт

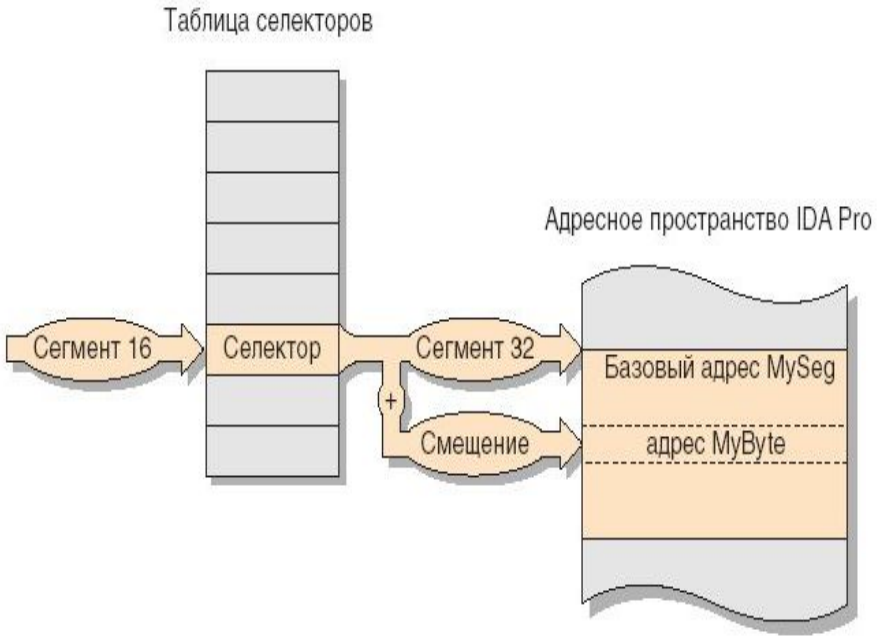
Организация памяти



Блок сегментации транслирует логическое адресное пространство в 32(64)-битное пространство линейный адресов.

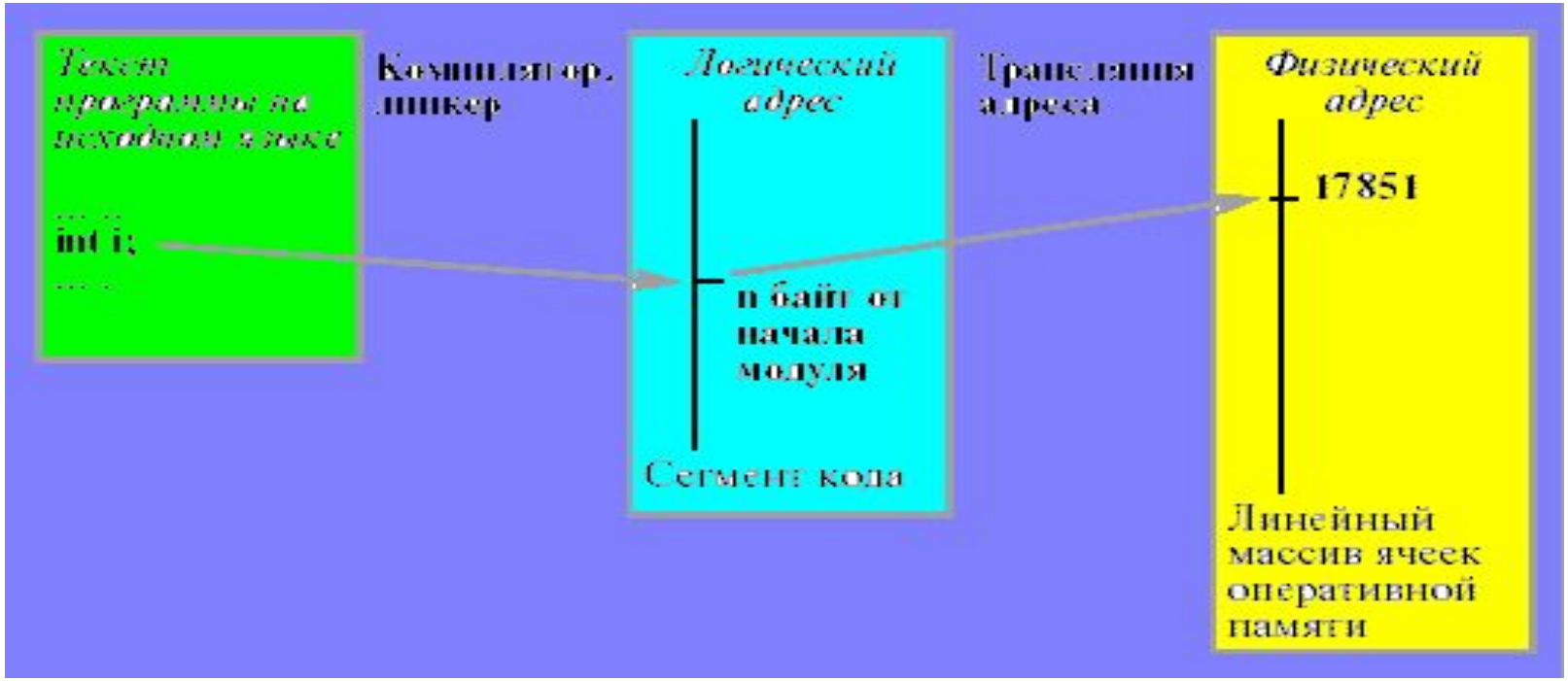
Линейный адрес образуется сложением базового адреса сегмента с эффективным адресом.

- Базовый адрес сегмента в реальном режиме образуется умножением содержимого используемого сегментного регистра на 16.
- В защищенном режиме базовый адрес загружается из дескриптора, хранящегося в таблице, по селектору, загруженному в используемый сегментный регистр.



Перевести сегментный адрес в линейный очень просто: ea = segment * 0x10 + offset

Организация памяти



- **Физический адрес** памяти образуется после преобразования линейного адреса блока страничной переадресации.
- В простейшем случае он совпадает с линейным.

- Блоком страничной переадресации осуществляет трансляцию линейного адреса в физический страницами (4Кбайт/2Мбайта/4Мбайта)

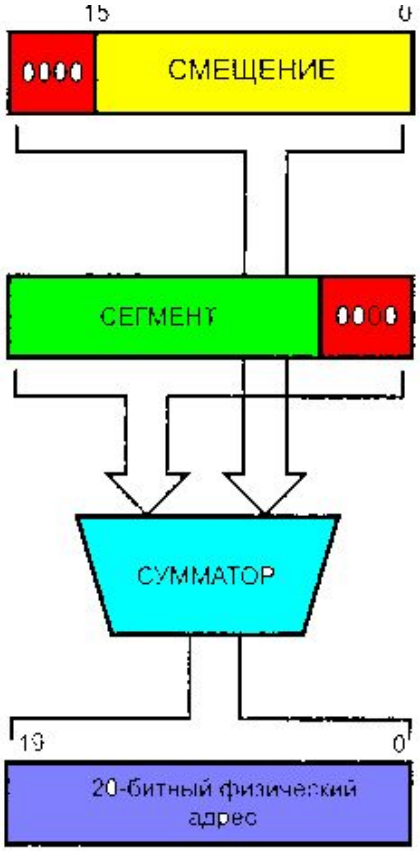
Организация памяти

Физические адреса - это реальные адреса, используемые для выбора микросхем физической памяти, содержащих данные.

- Физическая память организована в виде последовательности 8-разрядных байтов.
- Каждому байту присвоен уникальный адрес, который может изменяться от 0 до 2^{32}-1 (или 2^{36}-1 у P6+).
- Физическая память составляет единое целое с компьютером и управляется процессором (с некоторой помощью средств прямого доступа к памяти).

Сдвиг адреса сегмента на 4 бита влево эквивалентен умножению на 16, следовательно, физический адрес

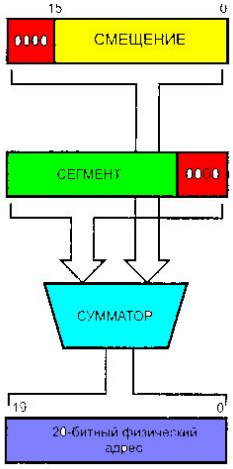
PA=16*Seg +EA



Формирование физического адреса памяти процессором, с использованием 16-разрядных регистров

Физический адрес памяти состоит из двух(четырех)-битных частей – адреса сегмента(Seg) и исполнительного адреса EA (executive address),суммируемых со смещением на 4 бита.

Модель памяти в реальном режиме



Исполнительный адрес, также называемый *эффективным адресом*, может быть константой, содержимым регистра, содержимым ячейки памяти или суммой нескольких величин (например, двух регистров и константы), но эта сумма является *16-разрядной* (перенос игнорируется). Таким образом, физический адрес *никогда* не перейдет границу 64-килобайтного сегмента, на начало которого указывает текущий сегментный указатель. Сегмент получается как бы свернутым в кольцо (wrapped): по мере увеличения суммируемых компонентов исполнительный адрес растет, но после достижения значения FFFFh снова обнуляется и начинает расти с начала. С одной стороны, это свойство обеспечивает некоторую защиту сегментов друг от друга (хотя некорректно написанная программа может легко перезагрузить указатель сегмента и повредить данные другого сегмента)

Поскольку свернутым в кольцо является всё пространство памяти: по мере увеличения исполнительного адреса и адреса сегмента физический адрес растет, но только до значения FFFFFh, после чего обнуляется и начинает расти с начала.

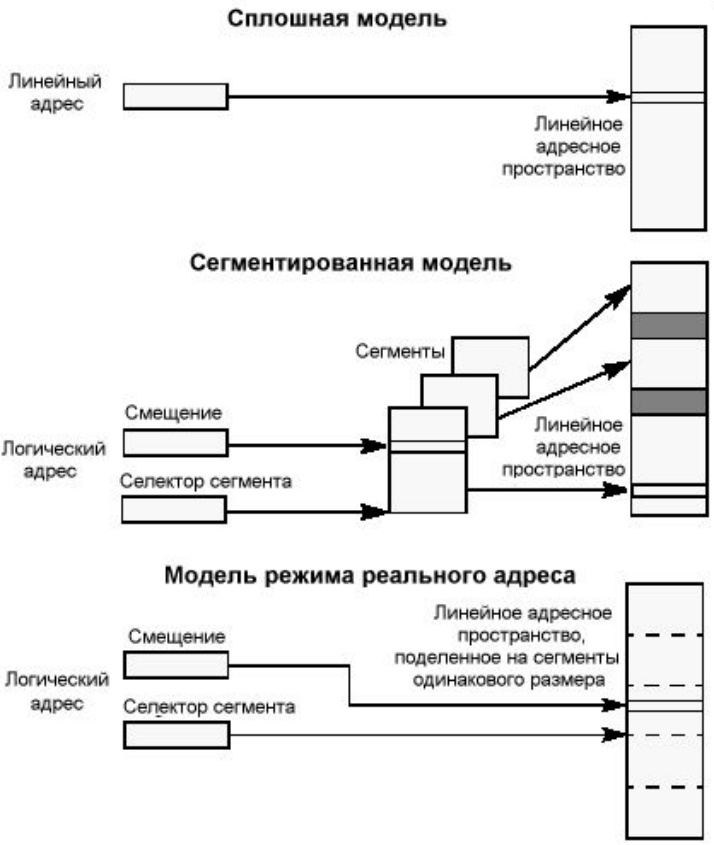
Модель памяти в реальном режиме

С сегментацией связаны понятия ближнего и дальнего адреса (вызова, перехода). При *ближнем* (Near) или *внутрисегментном* обращении доступ к требуемой ячейке памяти осуществляется только указанием смещения, а адрес сегмента определяется текущим содержимым соответствующего регистра сегмента. При *дальнем* (Far) или *межсегментном* обращении указывается полный адрес, содержащий 16-битное значение сегмента (загружаемое в соответствующий сегментный регистр) и 16-битное смещение. Естественно, что дальние обращения выполняются медленнее (хотя бы из-за пересылки большего количества байт адреса).

Процессор может обращаться как к одному байту памяти, так и к *слову*, состоящему из двух байт, или к *двойному слову* (4 байта). Двойное слово обычно используется для хранения полного адреса, и в нем располагается сначала слово смещения (в порядке I–II), а затем сегмента (в том же порядке). Сегментация памяти и порядок I– II являются характерной чертой процессоров x86.

Модель памяти в реальном режиме

При вычислении физического адреса ( ) возможно возникновение переполнения, которое с 20-разрядной шиной адреса приводило к сворачиванию пространства в кольцо. Если, например, Seg = FFFFh и EA = FFFFh, физический адрес, вычисленный по формуле PA = 16 × Seg + EA, получается равным 10FFEF. Процессором 8086 он трактуется как 0FFEF — адрес, принадлежащий первому мегабайту. Однако на выходе A20 процессора в этом случае установится единичное значение, что соответствует адресу ячейки из второго мегабайта физической памяти. Для обеспечения полной программной совместимости с 8086 в схему IBM PC/AT был введен специальный вентиль *Gate A20*, принудительно обнуляющий бит A20 системной шины адреса. 32-разрядные процессоры имеют специальный вход A20M, по которому в реальном режиме принудительно обнуляется внешний бит адреса A20. Вентиль в PC управляется через программно-управляемый бит контроллера клавиатуры 8042 или более быстрым способом (Gate A20 Fast Control), определяемым чипсетом системной платы.

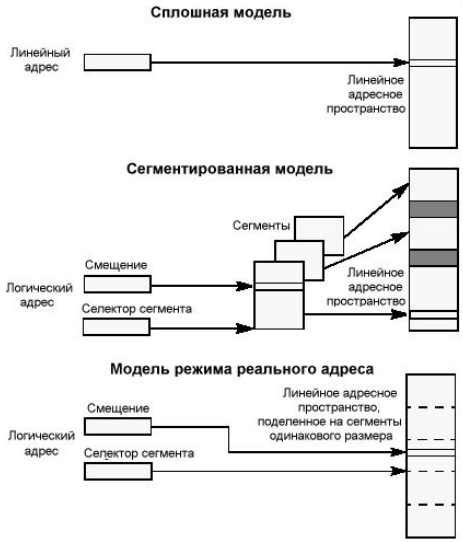


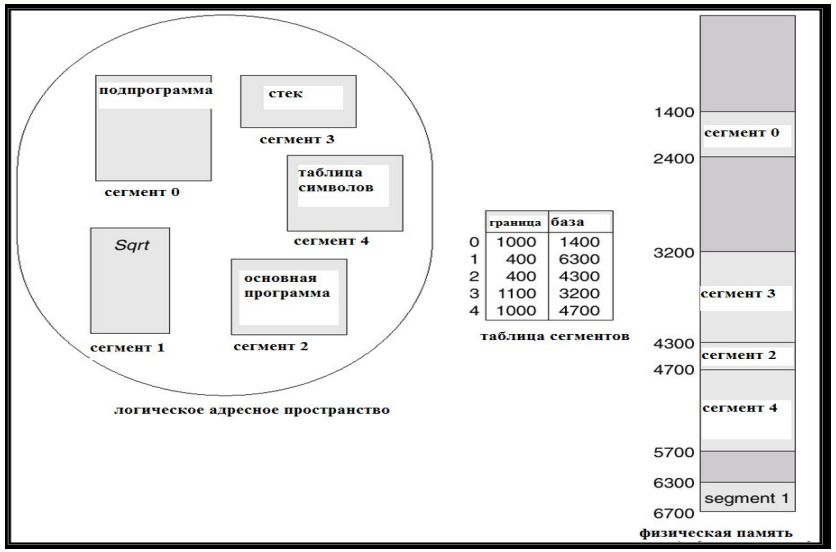
При использовании **сплошной модели** (Flat Model) памяти программа оперирует единым непрерывным адресным пространством - линейным адресным пространством. В нем содержатся и код, и стек, и данные программы, адресуемые смещением в пределах от 0 до 2^{32}-1. Такое 32-битное смещение называется линейным адресом.

WIKIPEDIA

При использовании средств процессора для управления памятью, программа может использовать одну из трех моделей доступа к памяти

- сплошная ("плоская") модель памяти
- сегментированная модель памяти
- модель режима реального адреса

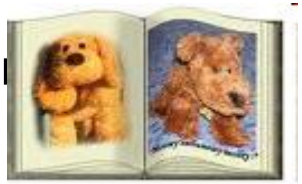




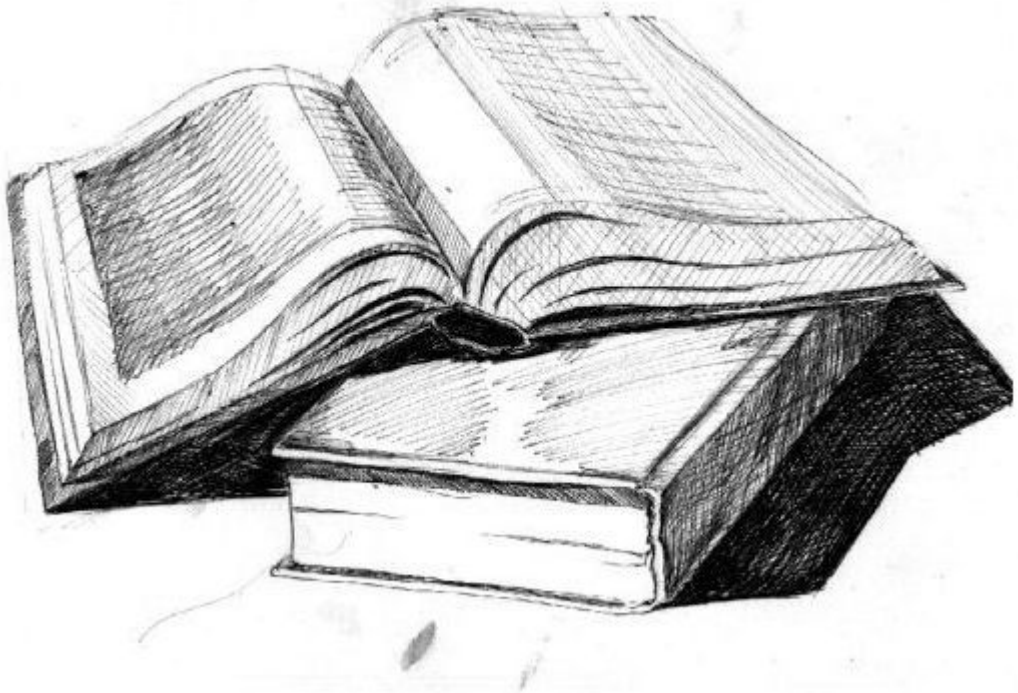
При использовании **сегментированной модели** (Segmented Model) для программы память представляется группой независимых адресных блоков, называемых сегментами. Для адресации байта памяти программа должна использовать логический адрес, состоящий из селектора сегмента и смещения. Селектор сегмента выбирает определенный сегмент, а смещение указывает на конкретный байт в адресном пространстве выбранного сегмента. Каждая задача в защищенном режиме может иметь до 16383 сегментов, размером до 4 Гбайт каждый, таким образом, общий объем памяти, адресуемой программой составляет 64 Тбайт. Микропроцессор при помощи блока сегментации отображает логический адрес в линейное адресное пространство. Сегментация позволяет эффективно управлять пространством логических адресов. Сегменты используются для объединения областей памяти с общими атрибутами. Каждый сегмент имеет несколько связанных с ним атрибутов: размер, расположение, тип (стек, программа или данные) и характеристики защиты.

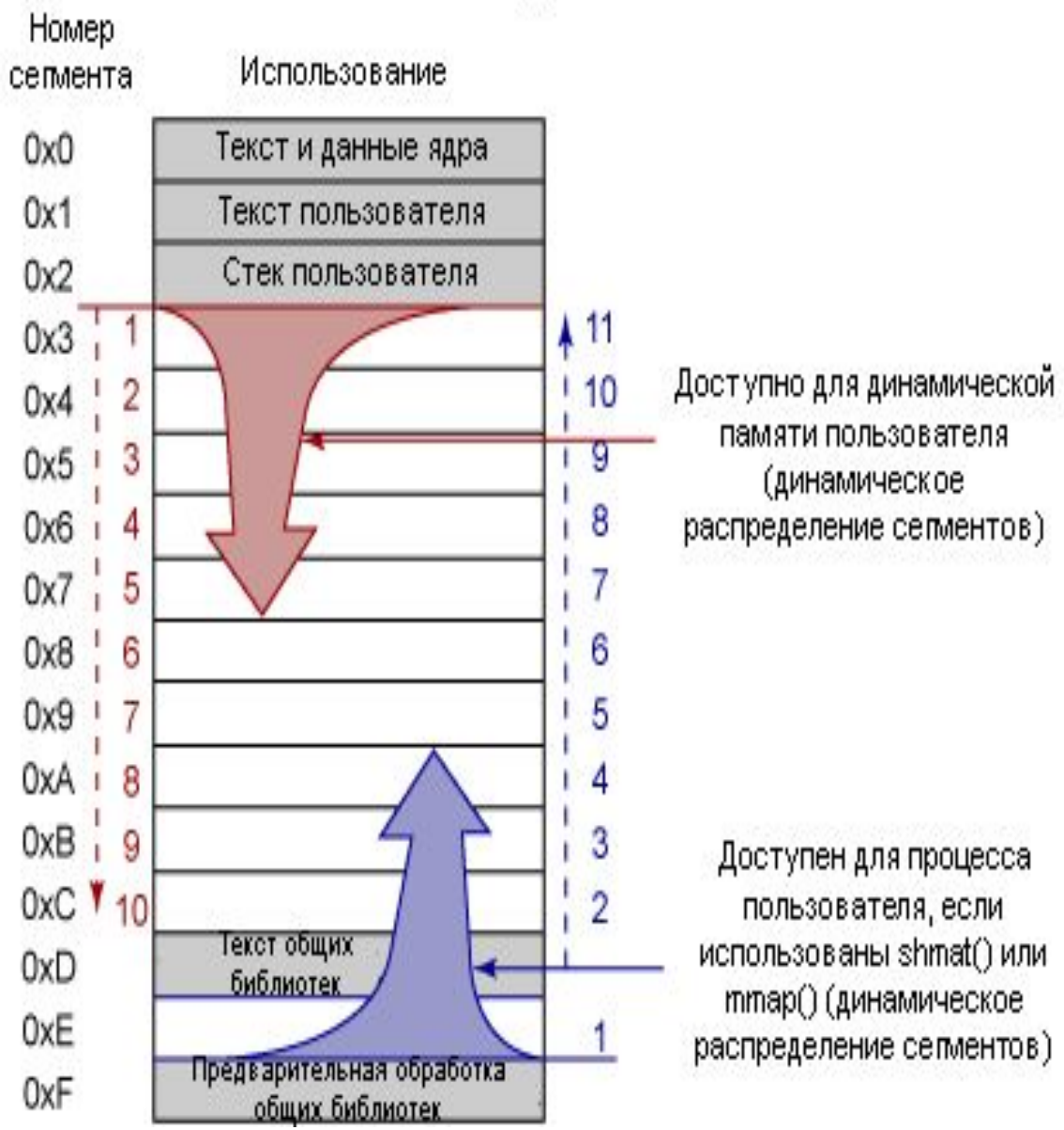
Архитектура может использовать **модель режима реального адреса** (Real-address Mode Model). Эта модель является специфическим случаем сегментированной модели. Линейное адресное пространство образуется из массива сегментов длиной по 64 Кбайт. Размер такого линейного адресного пространства - 1 Мбайт. В этой модели селектор сегмента непосредственно используется для вычисления базового адреса в линейном адресном пространстве путем сдвига значения селектора на 4 бита влево (умножение на 16). Это значит, все сегменты начинаются с адреса, кратного 16. 16-байтный блок памяти называют параграфом, поэтому говорят, что сегмент выравнивается по границе параграфа. Т.к. размер сегмента намного превышает размер параграфа, то имеет место перекрытие сегментов, т.е. один и тот же байт линейного адресного пространства может быть адресован различными логическими адресами (используя селекторы разных сегментов).

Используемая литература:



- Книга «Ассемблер. Учебник для ВУЗов», авторы Михаил Гу Юров
- Книга «Архитектура ЭВМ»,автор Мюллер
- *http://www.insidepro.com/kk/00*




Номер сегмента
Использование
0x0
0x1
0x2
0x3
0x4
0x5
0x6
0x7
0x8
0x9
0xA
0xB
0xC
0xD
0xE
0xF
Текст и данные ядра
Текст пользователя
Стек пользователя
Текст общих библиотек
Предварительная обработка общих библиотек
1
2
3
4
5
6
7
8
9
10
11
10
9
8
7
6
5
4
3
2
1
Доступно для динамической памяти пользователя (динамическое распределение сегментов)
Доступен для процесса пользователя, если использованы shmat() или mmap() (динамическое распределение сегментов)