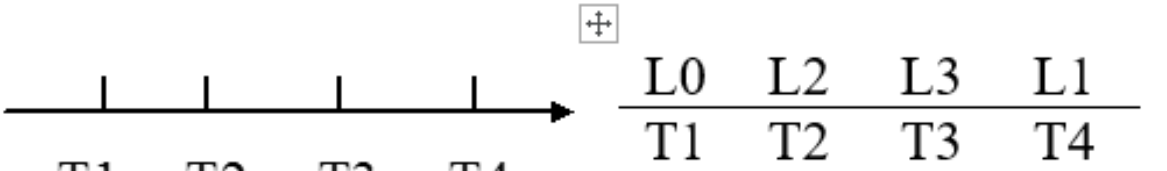
21. Стратегия Pseudo LRU для выбора блока-кандидата на удаление из кэш.

Реализация стратегии PLRU основывается на использовании 3-х битов (b0, b1, b2). Для каждого множества блоков КЭШ памяти. Модификация этих битов осуществляется при любом обращении к блоку в рамках этого множества (по чтению и по записи). Выбор блока, кандидата на удаление (одного из 4-х в рамках множества) осуществляется на основе значений битов LRU в соответствии с таблицей.

Значения битов LRU			Блок, кандидат на удаление
B0	B1	B2	
0	0	X	L0
0	1	X	L1
1	X	0	L2
1	X	1	L3

Алгоритм выбора блока кандидата на удаление, реализованный в процессорах Intel, в некоторых случаях выбирает не последний блок, к которому осуществлялось обращение, а предпоследний (отсюда и название PLRU). Примером использования нестрогого принципа LRU может служить временная последовательность обращений к блокам следующего вида:



T1	T2	T3	T4
t	B0	B1	B2
T1	1	1	X
T2	0	1	1
T3	0	1	0
T4	1	0	0

В соответствии с рассмотренной последовательностью обращений кандидатом на удаление должен быть блок L0 с наиболее ранним по времени обращением. В соответствии же с таблицей, комбинация битов LRU на момент времени T4 (100) соответствует выбор блока L2; не наиболее раннего, а предпоследнего. Стратегия PLRU лишь немногим уступает по эффективности.

27. Обобщение понятия кэширования.

В общем плане, кэширование следует рассматривать как универсальный метод, используемый для ускорения доступа к ОП, к диску и другим видам запоминающих устройств. Если кэширование применяется для уменьшения среднего времени доступа к ОП, то в качестве кэш-памяти используют быстродействующую статическую память (SRAM). Если кэширование используется системой ввода/вывода для ускорения доступа к данным, хранящимся на диске, то в этом случае роль кэш-памяти выполняют буферы в ОП, в которых оседают наиболее активно используемые данные. Виртуальную память так же можно считать одним из вариантов реализации принципа кэширования данных, при котором ОП

выступает в роли кэш-памяти по отношению к внешней дисковой памяти. Однако, в этом случае кэширование используется не столько для того, чтобы уменьшить время доступа к данным, сколько для того, чтобы заставить дисковую память подменить ОП за счет перемещения временно неиспользуемых кода и данных на диск с целью освобождения места для активных процессов в ОП. В результате, наиболее интенсивно используемые данные оседают в ОП, в то время как остальная информация хранится в более объемной и менее дорогостоящей внешней памяти.

28. Задача защиты памяти. Основные способы защиты памяти.

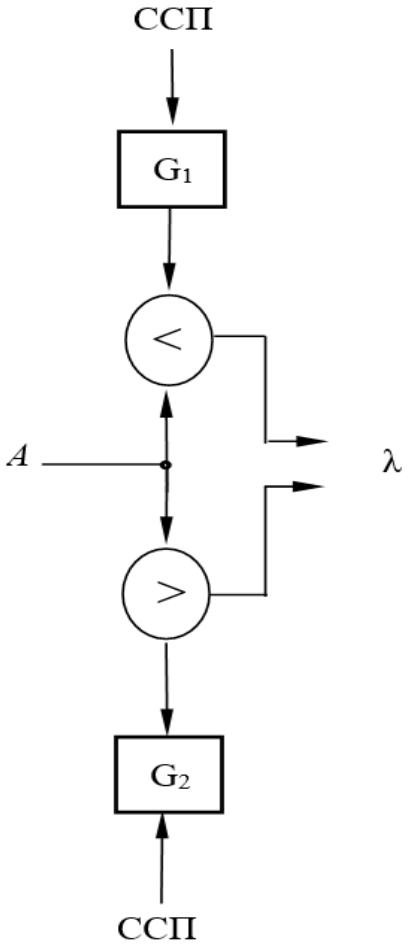
Средства защиты памяти обеспечивают проверку адреса при каждом обращении к памяти. Адрес считается корректным, если он принадлежит области памяти, которая выделена программе, выполняемой процессором или внешним устройством. Адрес считается некорректным, если он не принадлежит области памяти, выделенной программе. Когда в программе появляется некорректный адрес, обращение к памяти блокируется, и программа прерывается по причине нарушения защиты памяти. Прерванная программа исключается из процесса обработки. Искажение информации, не принадлежащей задаче, может произойти только при записи информации по некорректному адресу. С точки зрения программиста неправильная адресация как при записи, так и при чтении информации одинаково важна, поскольку свидетельствует о наличии ошибки в программе. По этой причине в компьютере контролируются любые обращения к памяти – и запись, и чтение.

Для защиты памяти в мультипрограммных компьютерах используются следующие способы:

- **Защита памяти по граничным адресам.**
- **Защита памяти по ключам.**
- **Защита памяти по уровням привилегий.**

29. Защита памяти по граничным адресам.

Каждой программе выделяется область основной памяти, начинающаяся от ячейки с адресом G_1. Предполагается, что различным программам супервизор выделяет непрекрывающиеся области. Адреса A команд и операндов, генерируемые в процессе выполнения программы, проверяются на корректность путем сравнения их с граничными адресами $G_1 \leqslant A \leqslant G_2$ ($G_1<G_2$). Адрес некорректен, если $A<G_1$ или $A>G_2$. Проверка адресов производится схемой рис. 3.17, встраиваемой в процессор. При инициировании программы из слова состояния программы СПП (уровня прерывания) в процессор загружаются значения граничных адресов G_1, G_2. Адрес A, сформированный в программе, перед обращением к памяти сравнивается на меньше–больше с граничными адресами G_1, G_2. Если $A<G_1$ или $A>G_2$ формируется сигнал прерывания λ, по которому прекращается выполнение программы и управление передается супервизору, который выводит на печать информацию о команде, породившей некорректный адрес.



Если в компьютере используется страничная организация памяти, то программе выделяются области, состоящие из целого числа страниц. В этом случае границы G_1, G_2 определяются адресами страниц, которыми начинается и заканчивается область памяти. В m-разрядном адресе выделяются старшие разряды, представляющие адрес страницы, к которой производится обращение, и сравниваются с граничными значениями G_1, G_2 адресов страниц.

Способ защиты памяти по граничным адресам имеет следующий недостаток: необходимо, чтобы для размещения программы выделялся сплошной участок памяти.

30. Организация защиты памяти по ключам.

При страничной организации памяти в компьютере общего назначения широко используется способ защиты информации по ключам. *Ключ* – это r-разрядный двоичный код. Например, могут использоваться 4-разрядные коды, которые позволяют сформировать 16 различных ключей 0,1,…,15. Ключи присваиваются программам и страницам памяти. Каждой программе присваивается свой ключ, называемый *ключом программы*. Этот же ключ присваивается всем страницам, выделенным программе. Ключи страниц хранятся в памяти ключей (рис. 3. 18): ключ K_0 страницы 0 – в ячейке 0; ключ K_1 страницы 1 – в ячейке 1 и т. д. Ключи записываются в память ключей супервизором в момент подготовки программы к выполнению. Для этого используется привилегированная операция УСТАНОВИТЬ КЛЮЧ ПАМЯТИ *A, P*, где *A* – адрес ячейки, в которой хранится значение ключа; *P* – адрес страницы, которой присваивается ключ. В результате выполнения этой команды странице *P*, закрепленной за программой с ключом $K_П$, присваивается ключ $K_П$, который становится ключом страницы.

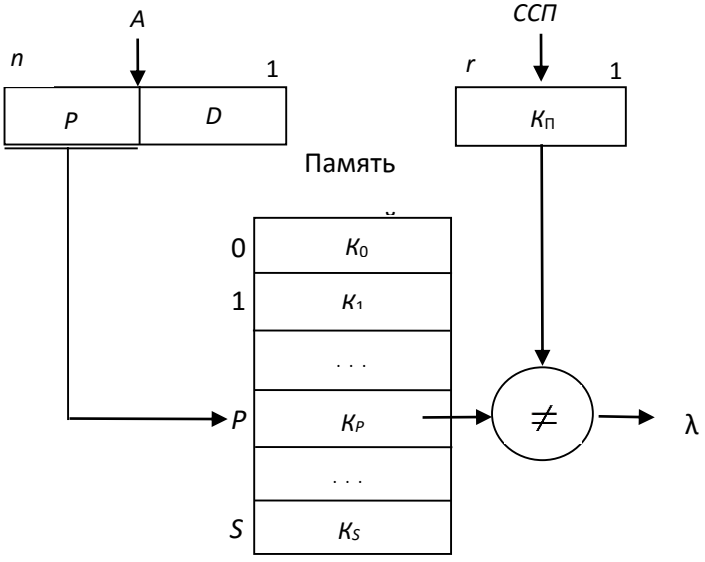


Рис. 3.18. Защита памяти по ключам

Проверка адресов организуется следующим образом (рис. 3.18). В момент инициирования программы ее ключ $K_П$, указанный в ССП (уровне прерывания), загружается в процессор и хранится там, пока процессор выполняет программу. В адресе *A*, генерируемом программой при обращении к памяти, выделяется поле *P*, определяющее адрес страницы, к которой производится обращение. По адресу *P* выполняется обращение к памяти ключей, из ячейки *P*

которой читается ключ страницы $K_Р$, ключ $K_Р$ сравнивается на равенство с ключом программы $K_П$. Если $K_Р \neq K_П$, адрес A считается некорректным и формируется сигнал λ прерывания по защите памяти. В противном случае разрешается обращение к памяти по адресу A.
Чтобы супервизору был обеспечен доступ к любой области памяти и несколько программ могли использовать для обмена информацией общую область памяти, используется универсальный ключ 0, который присваивается супервизору и страницам, общим для нескольких программ. При сравнении ключа программы $K_П$ с ключом страницы $K_Р$, ключ 0 считается равным любому другому ключу. Поэтому, если один из ключей $K_П$, $K_Р$ имеет нулевой код, адрес считается корректным и сигнал прерывания λ не формируется.

31. Организация защиты памяти по уровням привилегий.

В целях разграничения доступа к системным ресурсам со стороны прикладных и системных программ, в современных моделях процессоров в том или ином виде существует два основных режима функционирования:

- прикладной
- системный.

В системном режиме допускается исполнение любых машинных команд. В прикладном режиме не допускается исполнение так называемых привилегированных команд. В простейшем случае режим работы процессора задаётся с помощью специального бита, находящегося в каком – либо системном регистре. Например, в процессоре IBM 370 бит режима находится в слове состояния программы PSW (Program Status Word).Два альтернативных состояния процессора называются Task/Supervisor.

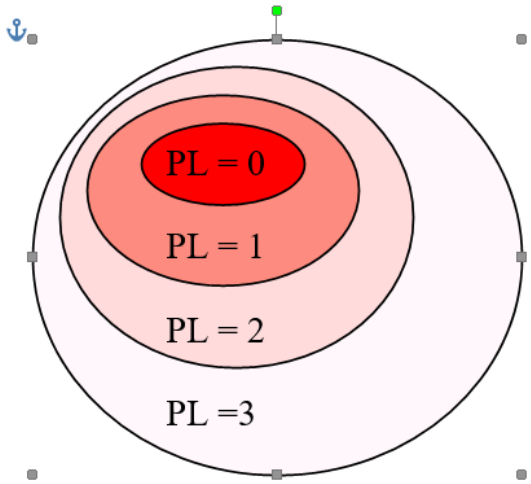
Аналогичный бит режима System/User имеет место в моделях системы VAX(DEC), который находится в PS (Processor Status).

В процессорах семейства 80X86 Pentium используется более сложная интерпретация (способ) для задания режима работы процессора в плане разделения системного и пользовательского режимов. С помощью специального бита PE (Protect Enabled) в управляющем регистре CR0 задаётся или реальный режим (PE = 0) или защищённый режим (PE = 1).

При использовании защищённого режима вступают в действие различные средства защиты. Одним из средств защиты, поддерживающимся на аппаратном уровне, является защита по уровням привилегий (кольцам защиты).

Аппаратно поддерживаются 4 уровня привилегий для сегментов и 2 уровня привилегий для страниц. Идея защиты по уровням привилегий состоит в присвоении различным сегментным объектам, в частности сегментам кода, данных, стека, определённого уровня привилегий. Этот уровень отражается соответствующим двухбитным полем DPL(Descriptor Privilege Level), размещаемом в дескрипторе (описателе сегмента).

Уровень привилегий определяет степень важности и доступности сегмента. Наивысшим уровнем привилегий является PL = 0, и он присваивается программам ядра, наинизший уровень PL = 3, он присваивается прикладным программам.



Общее правило защиты предполагает возможность обращений или доступа из внутренних колец. Во внешние. Попытки обращений из внешних колец во внутренние в общем случае пресекаются средствами защиты с выходом на прерывание специального типа Тип 13 (“Нарушение общей защиты”).

Пользовательский режим ассоциируется с уровнем привилегии PL = 3, системный режим с уровнем привилегий PL = 0.

Современные операционные системы, в частности Windows и Unix поддерживают только 2 уровня привилегий (User/Supervisor). Соответствующие биты для двух уровней используются на страничном уровне и размещаются в страничных дескрипторах. В процессорах семейства Intel к привилегированным командам относятся:

1) Команды загрузки и сохранения системных регистров.
2) Команды манипуляции флагом IF: CLI (0 - > IF), STI (1 - >IF).
3) Команды ввода/вывода: IN/OUT, INS/OUTS.
4) Команда останова процессора - HLT.

Попытка выполнения привилегированных команд в пользовательском режиме (PL = 3) приводит к выводу на прерывание 13. Почти все привилегированные команды, за исключением команд ввода/вывода и манипуляций флагом IF требует для своего выполнения наивысшего уровня привилегий PL = 0.

32. Понятие виртуальной памяти. Механизмы поддержки виртуальной памяти.

Под ***виртуальной памятью***, по мнению ДПС, понимается такой способ организации двухуровневой памяти (первый уровень – ОП, второй – внешняя дисковая память), при котором эта память воспринимается пользователем и, соответственно, программами, как большая одноуровневая память. При этом все пересылки между уровнями памяти являются невидимыми (прозрачными) для выполняемых программ.

Поддержка механизмов виртуальной памяти реализуется специальными аппаратными и программными средствами. Аппаратные средства в современных компьютерах обычно представлены специальным блоком **MMU – Memory Management Unit**, входящим в состав центрального процессора (CPU). В отношении процессоров семейства Intel 80x86 этот блок впервые появился в процессоре Intel 80286 и был предназначен для реализации сегментированной виртуальной памяти. Начиная со следующей модели Intel 80386, была реализована аппаратная поддержка как сегментной, так и страничной организации памяти. При этом блок MMU был разделен на две относительно независимые части в виде SU – Segment Unit и PU – Page Unit.

Программные средства поддержки виртуальной памяти входят в состав ОП, точнее, в ее ядро и обычно называются ***супервизором памяти***.

33. Виртуальный и физический адреса при страничной организации виртуальной памяти

СОСНУЛ

34. Страничная организация виртуальной памяти

В большинстве современных операционных систем виртуальная память организуется с помощью страничной адресации. Оперативная память делится на страницы: области памяти фиксированной длины (например, 4096 байт), которые являются минимальной единицей выделяемой памяти (то есть даже запрос на 1 байт от приложения приведёт к выделению ему страницы памяти). Исполняемый процессором пользовательский поток обращается к памяти с помощью адреса виртуальной памяти, который делится на номер страницы и смещение внутри страницы. Процессор преобразует номер виртуальной страницы в адрес соответствующей ей физической страницы при помощи буфера ассоциативной трансляции (TLB). Если ему не удалось это сделать, то требуется дозаполнение буфера путём обращения к таблице страниц, что может сделать либо сам процессор, либо операционная система (в зависимости от архитектуры). Если страница была выгружена из оперативной памяти, то операционная система подкачивает страницу с жёсткого диска. При запросе на выделение памяти операционная система может «сбросить» на жёсткий диск страницы, к которым давно не было обращений. Критические данные (например, код запущенных и работающих программ, код и память ядра системы) обычно находятся в оперативной памяти (исключения существуют, однако они не касаются тех частей, которые отвечают за обработку аппаратных прерываний, работу с таблицей страниц и использование файла подкачки).

35. Сегментная и страничная организация виртуальной памяти

Механизм организации виртуальной памяти, при котором виртуальное пространство делится на части произвольного размера — сегменты. Этот механизм позволяет, к примеру, разбить данные процесса на логические блоки. Для каждого сегмента, как и для страницы, могут быть назначены права доступа к нему пользователя и его процессов. При загрузке процесса часть сегментов помещается в оперативную память (при этом для каждого из этих сегментов операционная система подыскивает подходящий участок свободной памяти), а часть сегментов размещается в дисковой памяти. Сегменты одной программы могут занимать в оперативной памяти несмежные участки. Во время загрузки система создает таблицу сегментов процесса (аналогичную таблице страниц), в которой для каждого сегмента указывается начальный физический адрес сегмента в оперативной памяти, размер сегмента, правила доступа, признак модификации, признак обращения к данному сегменту за последний интервал времени и некоторая другая информация. Если виртуальные адресные пространства нескольких процессов включают один и тот же сегмент, то в таблицах сегментов этих процессов делаются ссылки на один и тот же участок оперативной памяти, в который данный сегмент загружается в единственном

экземпляре. Система с сегментной организацией функционирует аналогично системе со страничной организацией: время от времени происходят прерывания, связанные с отсутствием нужных сегментов в памяти, при необходимости освобождения памяти некоторые сегменты выгружаются, при каждом обращении к оперативной памяти выполняется преобразование виртуального адреса в физический. Кроме того, при обращении к памяти проверяется, разрешен ли доступ требуемого типа к данному сегменту.

Виртуальный адрес при сегментной организации памяти может быть представлен парой (g, s), где g — номер сегмента, а s — смещение в сегменте. Физический адрес получается путём сложения начального физического адреса сегмента, найденного в таблице сегментов по номеру g, и смещения s.

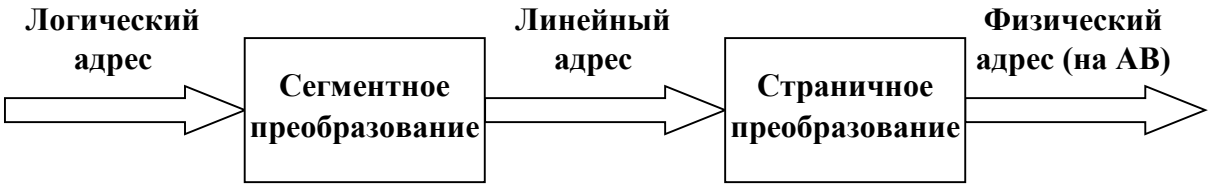
Недостатком данного метода распределения памяти является фрагментация на уровне сегментов и более медленное по сравнению со страничной организацией преобразование адреса.

Страничная – см. 34

36. Двухступенчатая схема преобразования логического адреса

При использовании виртуальной памяти одним из базовых механизмов является ***механизм преобразования*** (трансляции) логического (виртуального) адреса в физический.

В большинстве современных компьютеров поддерживается двухступенчатая схема преобразования следующего вида:



AB – Address Bus

37. Понятие селектора и дескриптора сегмента

Селектор — число, хранящееся в сегментном регистре; это 16-битная структура данных, которая является идентификатором сегмента. Селектор указывает не на сам сегмент в памяти, а на его дескриптор, в таблице дескрипторов. Селектор «живёт» в сегментном регистре (CS, DS, ES, FS, GS, SS).

В реальном режиме содержимое каждого сегментного регистра представляет собой номер параграфа — 16-байтового участка памяти, выровненного на границу 16 байт.

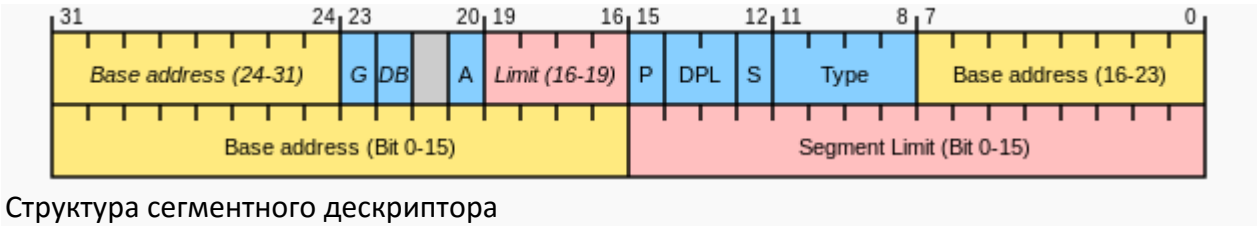
В защищённом режиме каждый сегментный регистр делится на три части, как показано ниже:

```
15      3 2  1  0
 +-------------------+
 | Index   | TI | RPL|
 +-------------------+
```

Бит TI в этом случае указывает, какая таблица дескрипторов должна использоваться (нуль соответствует глобальной таблице дескрипторов, единица — локальной таблице дескрипторов).

Поле Index является номером (индексом) дескриптора в таблице дескрипторов, который должен использоваться при вычислении линейного адреса. Наконец, поле RPL используется для контроля прав доступа программы к сегменту и является запрошенным уровнем привилегий. Частным случаем RPL является текущий уровень привилегий — CPL, чьё значение в любой момент времени находится в сегментном регистре CS.

Дескриптор сегмента — служебная структура в памяти, которая определяет сегмент. Длина дескриптора равна 8 байт .



Структура сегментного дескриптора

- База (жёлтые поля, 32 бита) — начало сегмента в линейной памяти
- Лимит (красные поля, 20 бит) — (размер сегмента в байтах)-1 (База+Лимит = линейный адрес последнего байта)
- Права доступа (синие поля, 12 бит) — флаги, определяющие наличие сегмента в памяти, уровень защиты, тип, разрядность + один пользовательский флаг

Байт прав доступа (AR, англ. Access Rights, биты 8-15 на рисунке):

- Бит P определяет доступность сегмента (0 — сегмента нет, 1 — есть). При обращении к сегменту со сброшенным битом P происходит исключение #NP, обработчик которого может загрузить/создать сегмент.
- Номер привилегий DPL содержит 2-битный номер (0-3), определяющий, к какому уровню (кольцу) защиты относится этот сегмент.
- Тип сегмента (биты 8-12 на рисунке). Старший бит (S) определяет сегмент как системный (S=0) или пользовательский (S=1).

38. Структура логического адреса. Сегментные регистры, их виды

Логический адрес в 16- и 32-разрядных системах сострит из двух компонентов: селектора сегмента и смещения. Однако в 64-разрядных системах логический адрес состоит только из смещения, так как сегментная организация памяти в этом случае не используется.

Сегментные регистры используются при формировании линейных адресов памяти. Вторым компонентом для формирования линейного адреса является смещение, называемое также эффективным адресом.

В *микропроцессоре 8086* было четыре 16-разрядных сегментных регистра:

- CS — сегментный регистр кода
- DS — сегментный регистр данных
- ES — сегментный регистр дополнительных данных
- SS — сегментный регистр стека

Общие правила использования сегментных регистров процессором таковы:

- для выборки кода команды всегда используется сегментный регистр CS
- при обращении к стеку (смещение формируется с использованием регистров SP/ESP/RSP или BP/EBP/RBP) всегда используется сегментный регистр SS
- в строковых операциях при обращении к операнду-приёмнику (смещение в регистре DI/EDI/RDI) применяется сегментный регистр ES
- во всех остальных случаях, если не используется префикс замены сегмента, доступ к памяти осуществляется с использованием сегментного регистра DS. При наличии префикса замены сегмента вместо DS используется указанный префиксом сегментный регистр

39. Реальный и защищенный режимы. Особенности использования сегментных регистров в этих режимах

Реальный режим. После инициализации (системного сброса) центральный процессор находится в реальном режиме. В реальном режиме центральный процессор работает как очень быстрый i8086 с возможностью использования 32-битных расширений. Механизм адресации, размеры памяти и обработка прерываний (с их последовательными ограничениями) микропроцессор 8086 полностью совпадают с аналогичными функциями других микропроцессоров с 32-битной Intel архитектурой в реальном режиме.

Для программного обеспечения этого типа обычно используется однозадачный режим, т. е. одновременно может выполняться только одна программа. Нет никакой встроенной защиты для предотвращения перезаписи ячеек памяти одной программы или даже операционной системы другой программой; это означает, что при выполнении нескольких программ вполне могут быть испорчены данные или код одной из них, а это может привести всю систему к краху (или останову).

Защищенный режим является основным режимом работы микропроцессора. Ключевые особенности защищенного режима: виртуальное адресное пространство, защита и многозадачность. В защищенном режиме программа оперирует с адресами, которые могут относиться к физически отсутствующим ячейкам памяти, поэтому такое адресное пространство называется виртуальным. Для адресации виртуального адресного пространства используется сегментированная модель, в которой адрес состоит из двух элементов: селектора сегмента и смещения внутри сегмента. С каждым сегментом связана особая структура, хранящая информацию о нем, - дескриптор. Кроме "виртуализации" памяти на уровне сегментов существует возможность "виртуализации" памяти при помощи страниц - *страничная трансляция*.

Встроенные средства переключения задач обеспечивают *многозадачность* в защищенном режиме. Среда задачи состоит из содержимого регистров МП и всего кода с данными в пространстве памяти. Микропроцессор способен быстро переключаться из одной среды выполнения в другую, имитируя параллельную работу нескольких задач. Для некоторых задач может эмулироваться управление памятью как у процессора 8086. Такое состояние задачи называется *режимом виртуального 8086*.

Защита задач обеспечивается следующими средствами: контроль пределов сегментов, контроль типов сегментов, контроль привилегий, привилегированные инструкции и защита на уровне страниц. Контроль пределов и типов сегментов обеспечивает целостность сегментов кода и данных. Программа не имеет права обращаться к виртуальной памяти, выходящей за предел того или иного сегмента. Программа не имеет права обращаться к сегменту данных как к коду и наоборот. Архитектура защиты микропроцессора обеспечивает 4 иерархических уровня привилегий, что позволяет ограничить задаче доступ к отдельным сегментам в зависимости от ее текущих привилегий. Кроме того, текущий уровень привилегий задачи влияет на возможность выполнения тех или иных специфических команд (привилегированных инструкций). Функции страничной трансляции, впервые появившиеся в МП Intel386, обеспечивают дополнительные механизмы защиты на уровне страниц.

С помощью специального бита PE (Protect Enabled) в управляющем регистре CR0 задаётся или реальный режим (PE = 0) или защищённый режим (PE = 1).

При использовании защищённого режима вступают в действие различные средства защиты. Одним из средств защиты, поддерживающимся на аппаратном уровне, является защита по уровням привилегий (кольцам защиты).

Аппаратно поддерживаются 4 уровня привилегий для сегментов и 2 уровня привилегий для страниц. Идея защиты по уровням привилегий состоит в присвоении различным сегментным объектам, в частности сегментам кода, данных, стека, определённого уровня привилегий. Этот уровень отражается соответствующим двухбитным полем DPL(Descriptor Privilege Level), размещаемом в дескрипторе (описателе сегмента)

40. Организация RAID-массивов

В 1988 году исследователей из США предложила ряд способов построения системы из большого числа накопителей на жестких магнитных дисках, которые начали называться RAID (Redundant Array of Inexpensive Disks – Избыточный Массив Недорогих Дисков). Идея RAID весьма проста. Рядом с компьютером устанавливается RAID – массив, состоящий из

RAID – контроллера и значительного числа дисков, которые обеспечивают как увеличение производительности компьютера, так и значительное увеличение надежности хранения данных. Массив дисков, подключенных к компьютеру, распознается операционной системой как один диск большой емкости. Отказоустойчивость дискового массива повышается за счет использования известных способов повышения надежности хранения данных: алгоритмов Хэмминга, циклических кодов и т.д. Дисковое пространство массива представляет собой объединение сегментов всех дисков и подразделяется на следующие уровни 0, 1, 2, …, 7, 10, 53, а также на ряд модификаций этих уровней. Наибольшее распространение получили уровни 0, 1, 2 и 5. Остальные уровни имеют гораздо меньшее распространение.

41. Уровни RAID

См. 42-45 и 41

42. Структура RAID 0

В RAID уровня 0, называемого *уровнем чередования данных*, каждый файл размещается на нескольких дисках, за счет чего увеличивается скорость передачи данных и, следовательно, возрастает производительность компьютера. Однако этот уровень организации RAID никак не влияет на вероятность отказа дисков. Поэтому уровень RAID 0 называют *дисковым массивом без дополнительной отказоустойчивости*. Структура такого массива представлена на рисунке 1, откуда видно, что производительность компьютера,

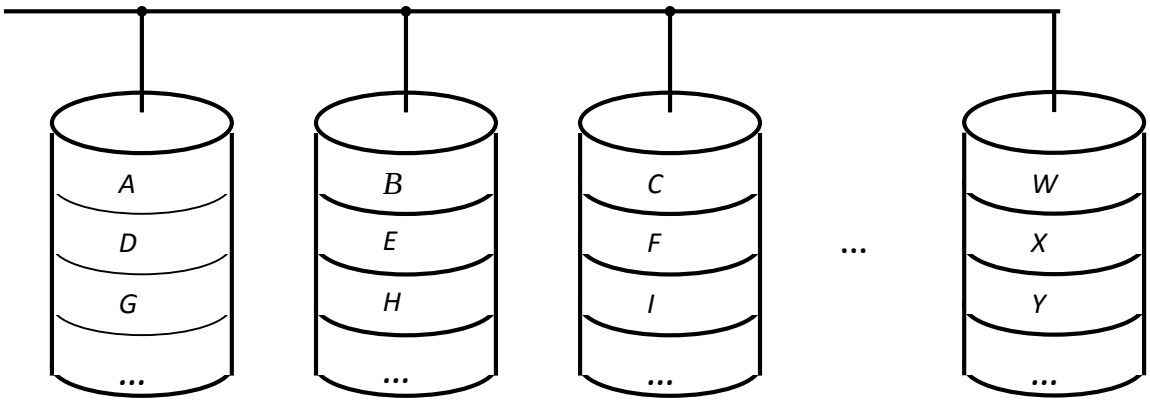


Рис. 1. Структура RAID 0

оснащенного системой RAID 0, возрастает за счет размещения записей *A*, *B*, *C*, …, *W*, … на нескольких дисках, в каждом из которых могут параллельно во времени происходить обращения для записи и чтения данных.

43. Структура RAID 1

Уровень RAID 1 обеспечивает зеркальное дублирование дисков, создавая на двух разных дисках две копии записей (рис. 2): основную и резервную, каждая из которых снабжается

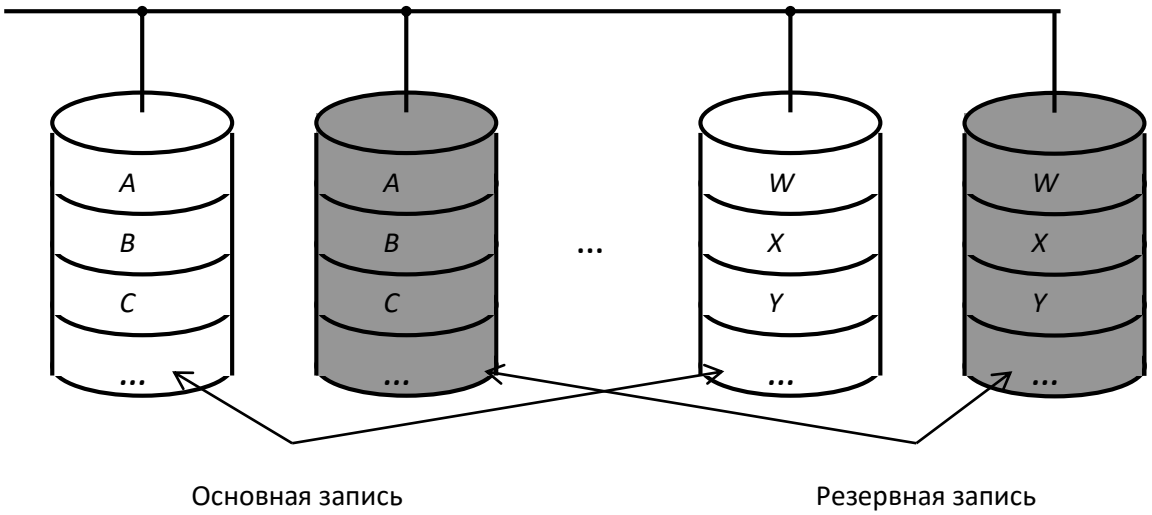


Рис. 2. Структура RAID 1

дополнительными разрядами, контролирующими корректность записанного кода. При чтении зеркальных копий обращение производится сразу к двум дискам, контроллер дисков сравнивает дополнительные разряды с суммой значений, записанных на секторах, и выдает на выходе совокупность значений, соответствующую значению дополнительных разрядов сектора. Если в одной из копий сумма значений не совпадает с дополнительными разрядами, то формируется сообщение о некорректной работе одного диска, передаваемого на экран системного администратора. Естественно, что если основная и резервная копии совпадают, то контроллер дисков выдает на выход одну из копий. Если в одной из копий зарегистрирован сбой данных, то на выход передаются данные от правильной копии данных. Недостаток этого способа запис данных – низкий коэффициент использования адресного пространства, который не превышает значения 0,5. Однако, если все диски работоспособны, то гарантируется полный контроль правильности считываемых данных.

44. Структура RAID 2

Уровень RAID 2 основан на использовании алгоритма Хемминга для проверки и восстановления данных. При этом для каждого сегмента данных вычисляется совокупность контрольных разрядов, набор которых записывается на отдельные диски (рис. 3). Алгоритм

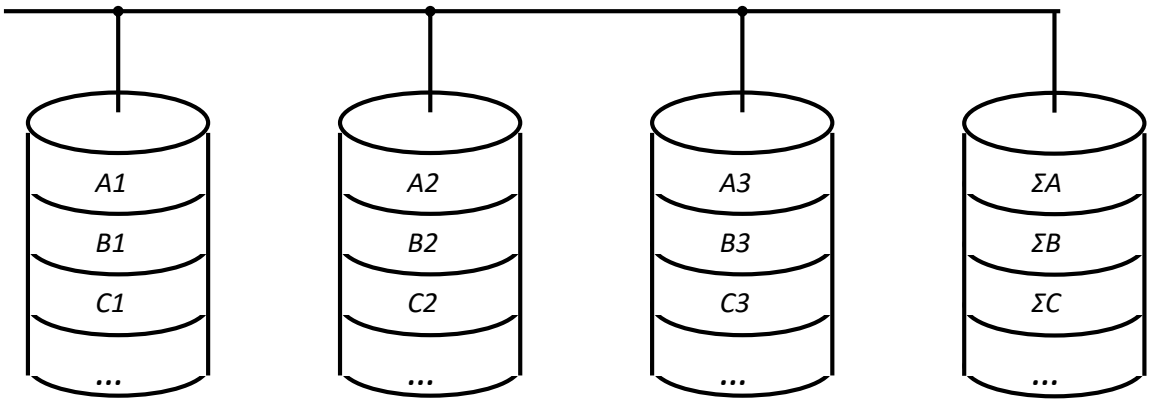


Рис. 3. Структура RAID 2

Хемминга позволяет устранить одну ошибку в любом секторе диска. По этой причине стремятся уменьшить количество информационных разрядов в каждом секторе, чтобы увеличить мощность системы контроля донных, т.е. чтобы контрольные разряды кода Хемминга приходились на небольшое число информационных разрядов. Естественно, что алгоритм Хемминга позволяет уменьшить количество разрядов секторов дисков во много раз, благодаря чему возрастает производительность дисковой системы.

45. Структура RAID 5

Уровень RAID 5 имеет структуру, изображенную на рис. 4. В отличии от ранее рассмотренных уровней организации RAID – систем, в уровне RAID 5 контрольные данные, формируемые по алгоритму Хемминга, размещаются во всех дисках равномерно,

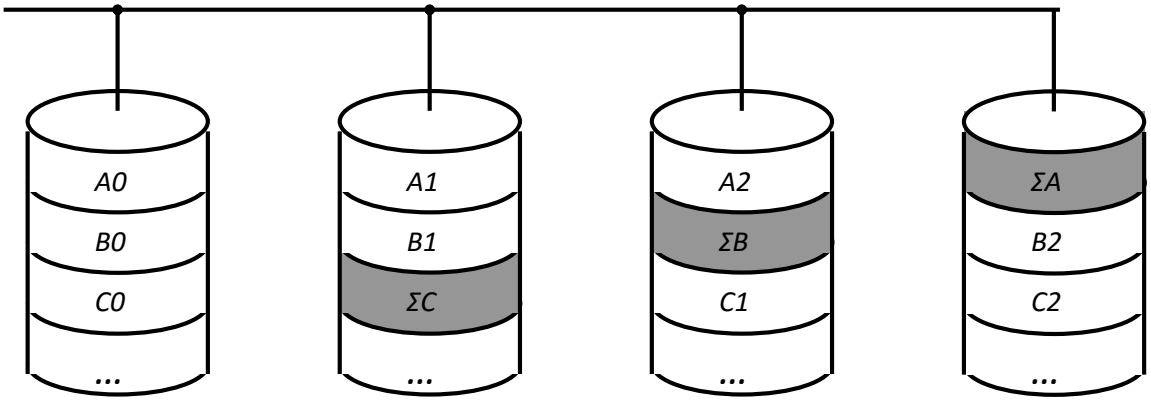


Рис. 4. Структура RAID 5

что обеспечивает высокую пропускную способность и защиту данных от сбоев и отказов аппаратуры.