Организация памяти

Иерархия памяти

Общая идея **иерархической (много-уровневой)** организации памяти заключается в использовании на одном компьютере нескольких различных типов запоминающих устройств (ЗУ), которые характеризуются разным временем доступа к ЗУ, его объемом и стоимостью. (*Время доступа* к ЗУ -- это время между операциями чтения/записи, которые выполняются по случайным адресам.)

Каждому типу ЗУ в зависимости от его характеристик назначается определенный уровень в иерархии памяти.

Порядок расположения уровней в иерархии

С увеличением уровня иерархии должно происходить:

- увеличение объема памяти данного уровня;
- увеличение времени доступа;
- уменьшение стоимости хранения единицы данных на данном уровне;
- уменьшение частоты обращений к уровню иерархии со стороны процессора.

Последние три требования легко выполняются в рамках технологических решений. Четвертое требование, как прави-ло, тоже выполняется, поскольку является следствием ***принципа локальности ссылок*** (обращений к памяти).

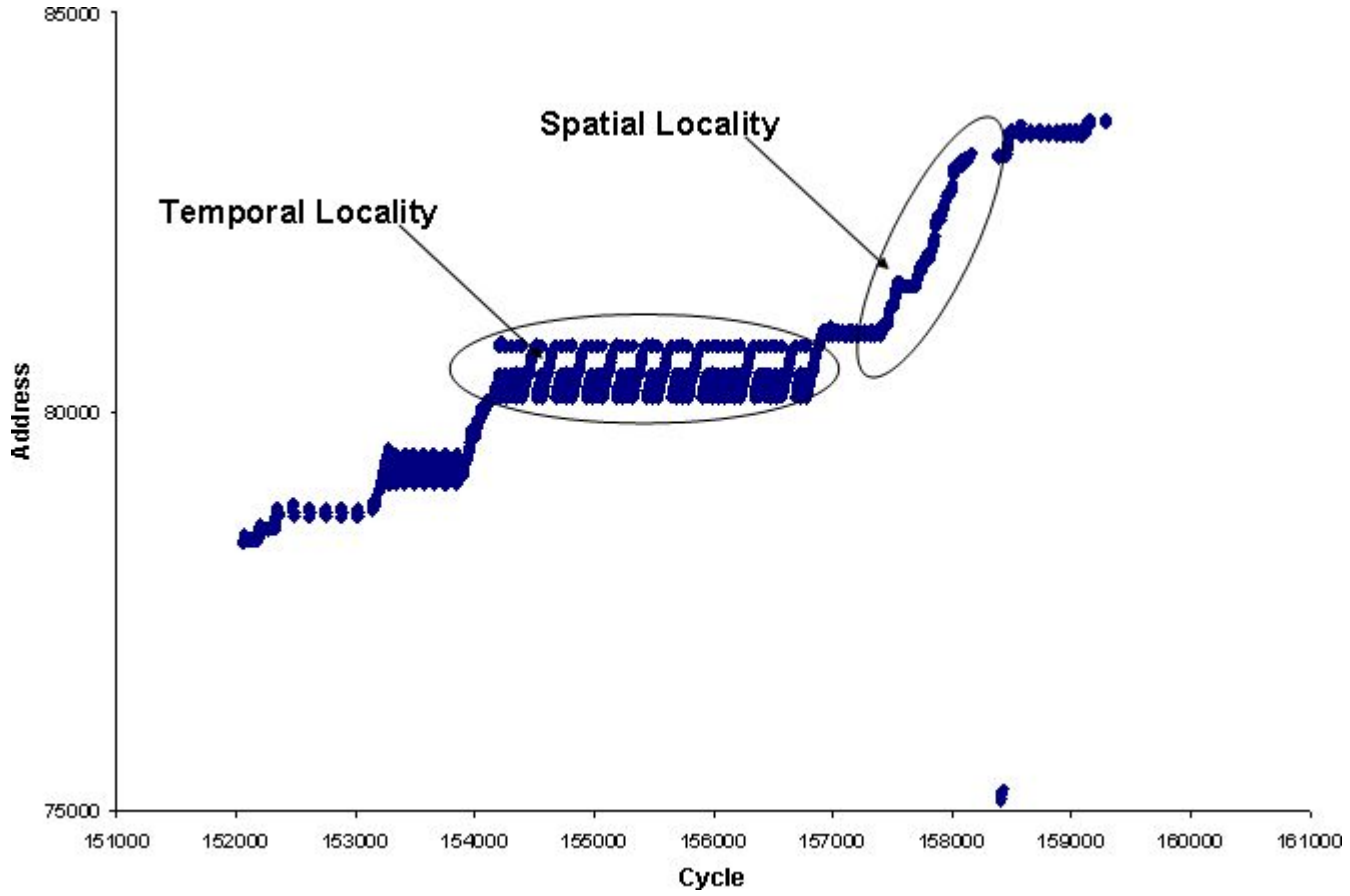
Принцип локальности ссылок

Принцип локальности состоит в том, что большинство программ, **выполняе-мых процессором, обладает свойства-ми локальности ссылок во времени и в пространстве.**

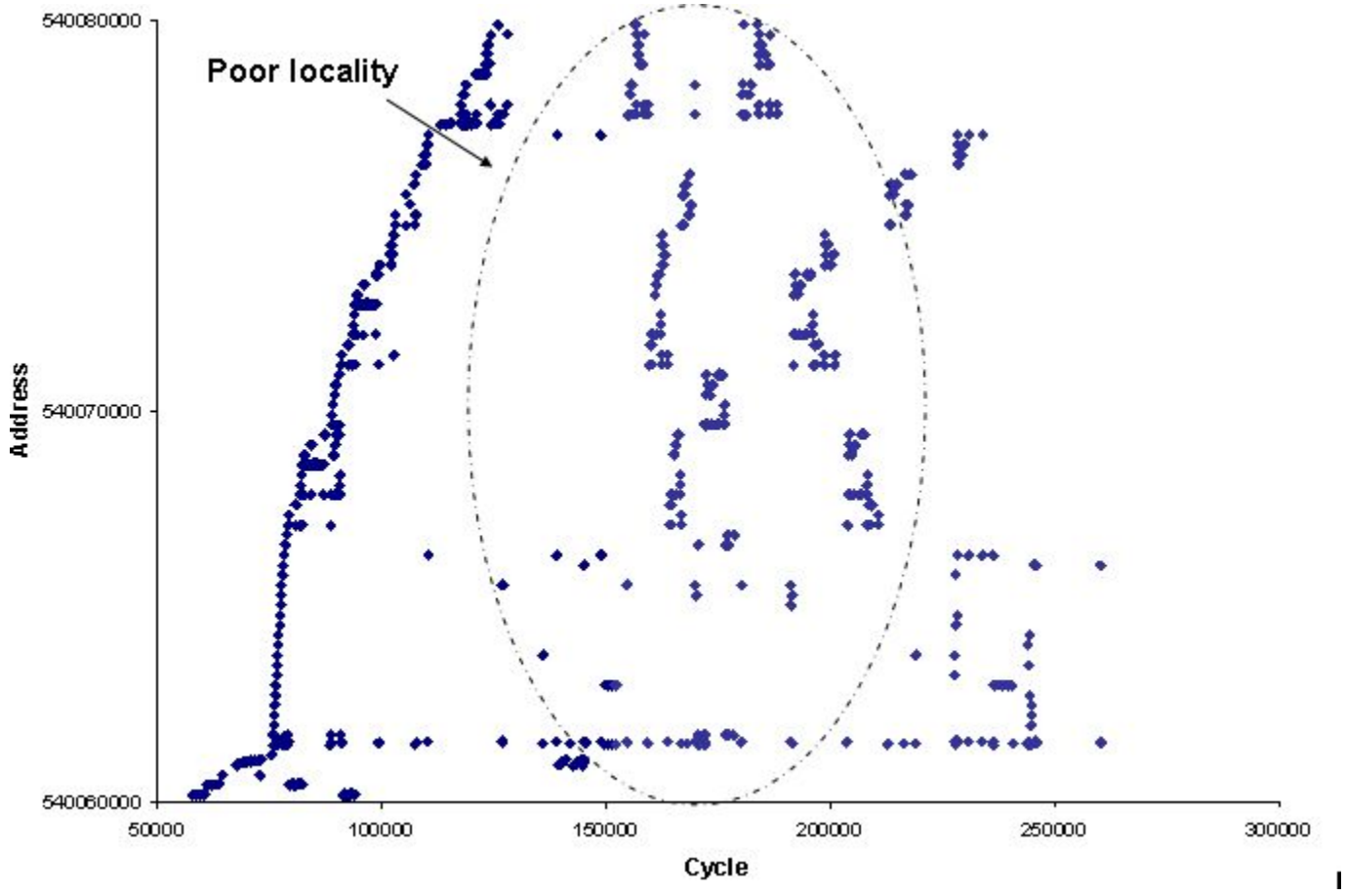
Локальность во времени состоит в том, что процессор многократно использует одни и те же команды и данные.

Локальность в пространстве состоит в том, что если программе нужен доступ к слову с адресом **A**, то скорее всего, следую-щие ссылки будут к адресам, расположен-ным по близости с адресом **A**.

Пример «хорошей» локальности



Пример «плохой» локальности



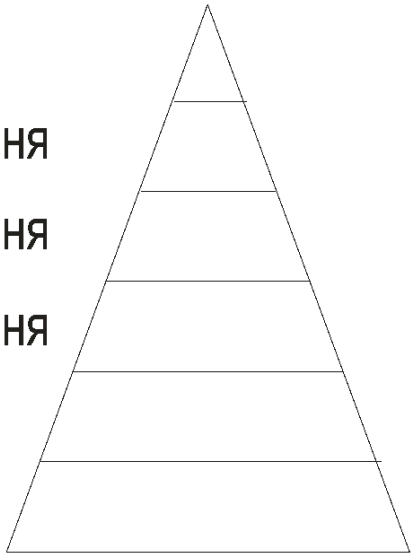
Иерархия памяти

Из свойства локальности ссылок следует, что в типичном вычислении обращения к памяти концентрируются вокруг небольшой области адресного пространства и более того, выборка идет по последовательным адресам. Время доступа к иерархически организованной памяти уменьшается благодаря следующему

- **сокращению количества обращений к оперативной памяти и**
- **совмещению обработки текущего фрагмента программы и пересылки данных из основной памяти в буфер-ную память.**

Схема иерархического построения памяти

регистровая память	64-256 слов, 1 такт
кэш-память 1-го уровня	до 64 КБ, 1-4 такта
кэш-память 2-го уровня	до 6 МБ, 5-15 тактов
кэш-память 3-го уровня	до 32 МБ, 10-30 тактов
основная память	до 256 ГБ, 100-200 тактов
внешняя память	до десятков ТБ, от 10^6 тактов

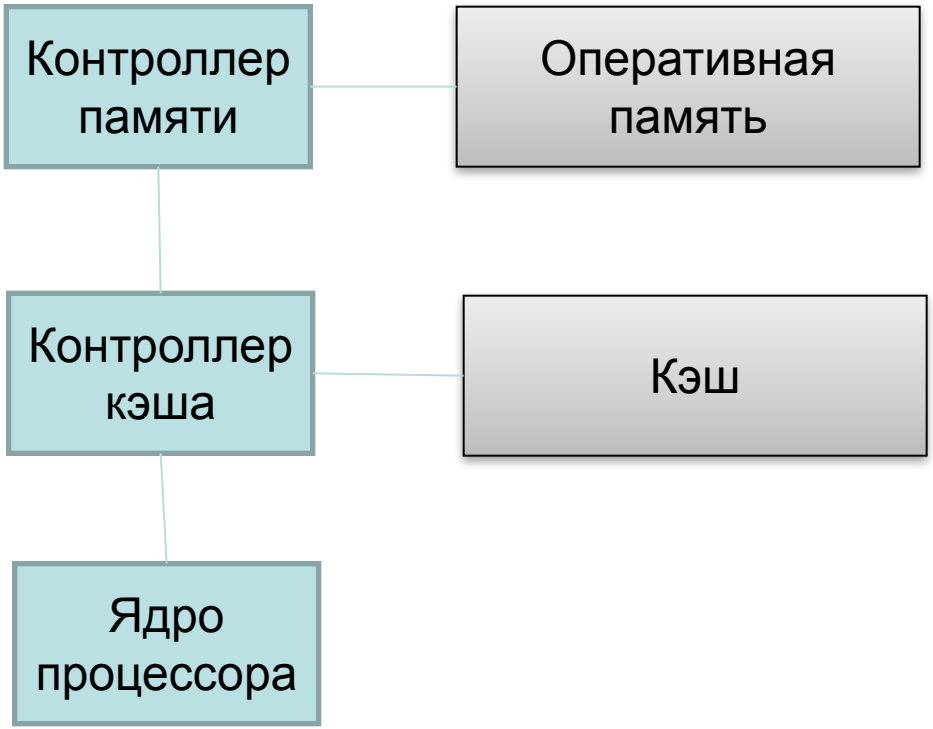


Организация кэш-памяти

Кэш-память это высокоскоростная память небольш**е**го размера с прямым доступом**.** Она предназначена для временного хранения фрагментов ко-да и данных**.** Кэш-память охватывает все адресное пространство памяти**,** но в отличие от оперативной памяти**,** она не адресуема и невидима для про-граммиста**.**

Работой кэш-памяти управляет кэш- контроллер **(**он интегрирован в процес-сор**).**

Схема работы кэш-контроллера



Функции кэш-контроллера

- Загрузка (выгрузка) копии кода и данных из ОП в кэш-память блоками слов с последовательными адресами, равными размеру строки кэш-памяти за один цикл чтения, даже, если процессор обращается только к одной ячейке памяти из данного блока.
- Контроль запросов процессора к ОП и проверка, есть ли действительная копия информации в кэш-памяти. Если копия присутствует (кэш попадание), то слово считывается из кэш-памяти и передается в процес-сор. Если действительная копия информации отсут-ствует в кэш-памяти (кэш промах), тогда запрос адресуется к ОП, и требуемая копия записывается на одну из строк кэш-памяти.
- Обеспечение когерентности, т.е., согласованности данных кэш-памятей с данными основной памяти.

Схема работы кэш-контроллера

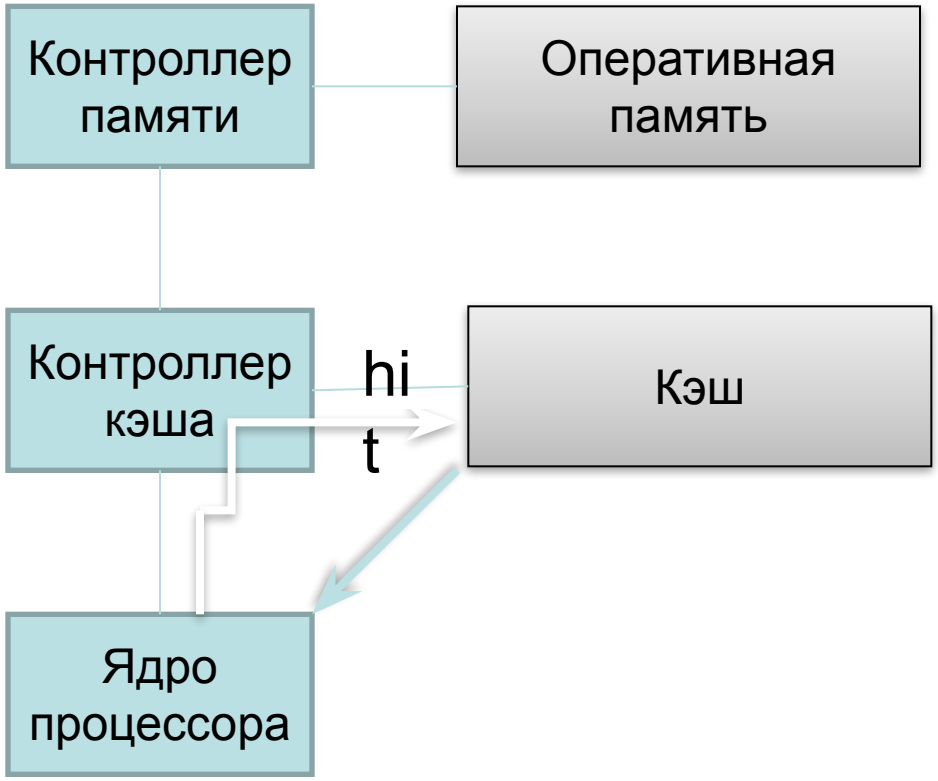
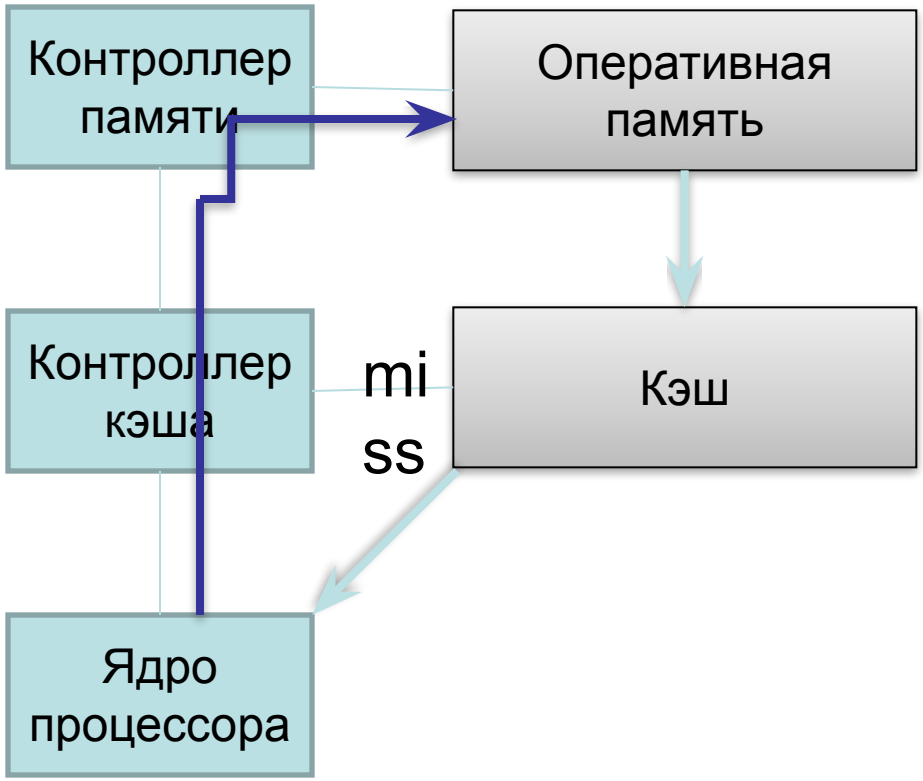
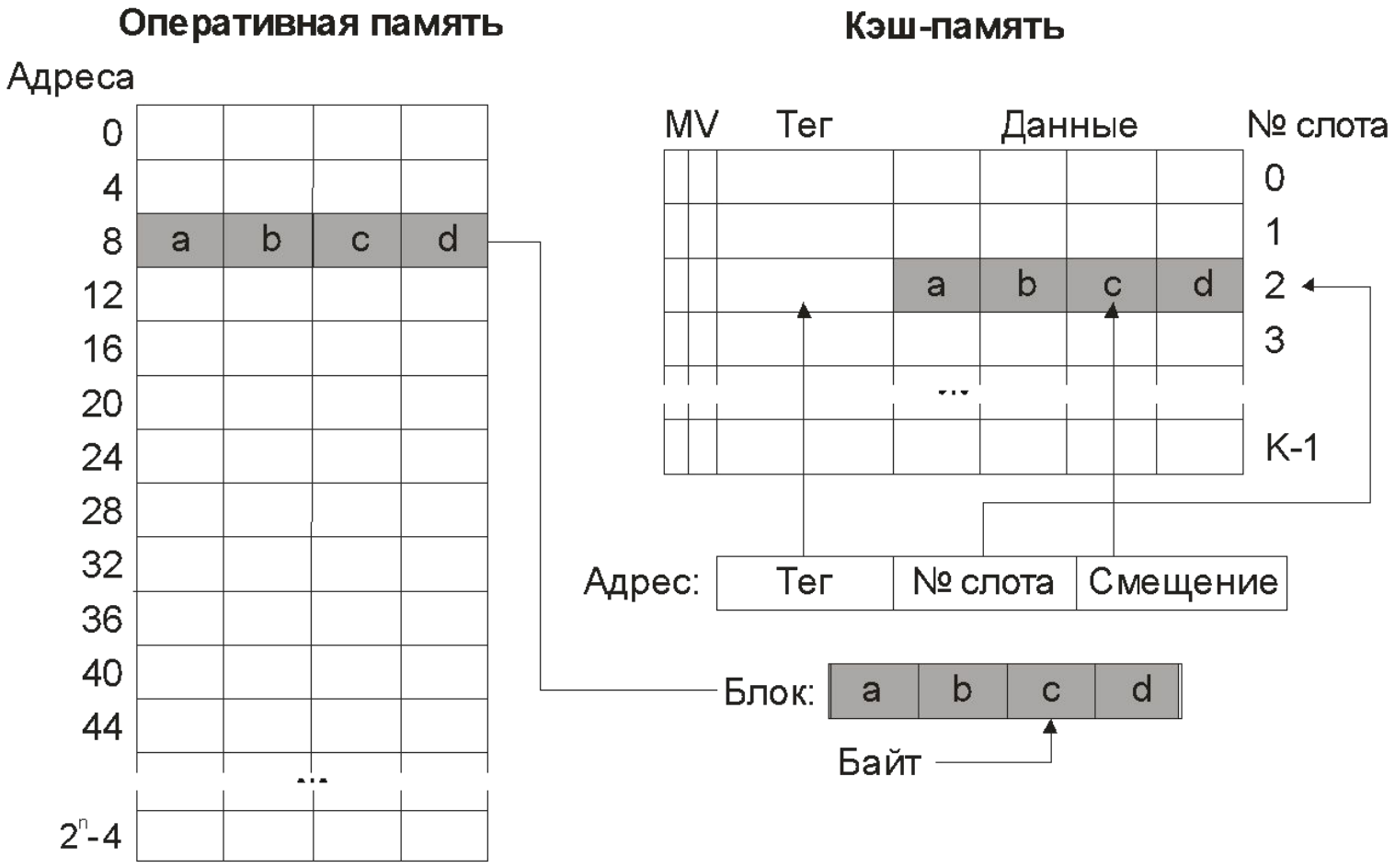


Схема работы кэш-контроллера



Структура кэш-памяти



Организация кэш-памяти

Когда контроллер выполняет поиск данных в памяти?

- после фиксации промаха (сквозной просмотр).
- одновременно с поиском блока в кэш-памяти, в случае кэш-попадания, обращение к оперативной памяти прерывается (отложенный просмотр).

Организация кэш-памяти

Когда контроллер помещает данные в кэш- память?

- Загрузка по требованию **(on demand).**
- Спекулятивная загрузка **(speculative load).**
  Алгоритм предполагает помещать данные в кэш-память задолго до того, как к ним произойдет реальное обращение. У кэш-контроллера есть несколько алгоритмов, которые указывают, какие ячейки памяти потребуются процессору в ближайшее

  время.

Основные вопросы организации кэш-памяти

- Алгоритм отображения адресов основной памяти в кэш-память.
- Алгоритм записи данных и команд из кэш-памяти в основную память.
- Алгоритм замещения строки в кэш-памяти.
- Размер кэш-памяти.
- Длина строки в кэш-памяти.

Алгоритмы отображения

- Прямой **(direct mapping).**
- Ассоциативный **(full associative mapping).**
- Множественно-ассоциативный **(set-associative mapping).**

Отображение блока ОП на линию кэш-памяти

i = j modulo m,

i – номер линии кэш-памяти,

j – номер блока ОП**,**

m – количество строк в кэш-памяти

кэш-линия	блоки ОП
0	**0, m, 2m,…**
1	**1, m+1, 2m+1,…**
m-1	**m-1, 2m-1, 3m-1,…**

Пример

Block – 4 Bytes

MM=16 Mbytes (2^{24}) (4M blocks of 4 bytes each)

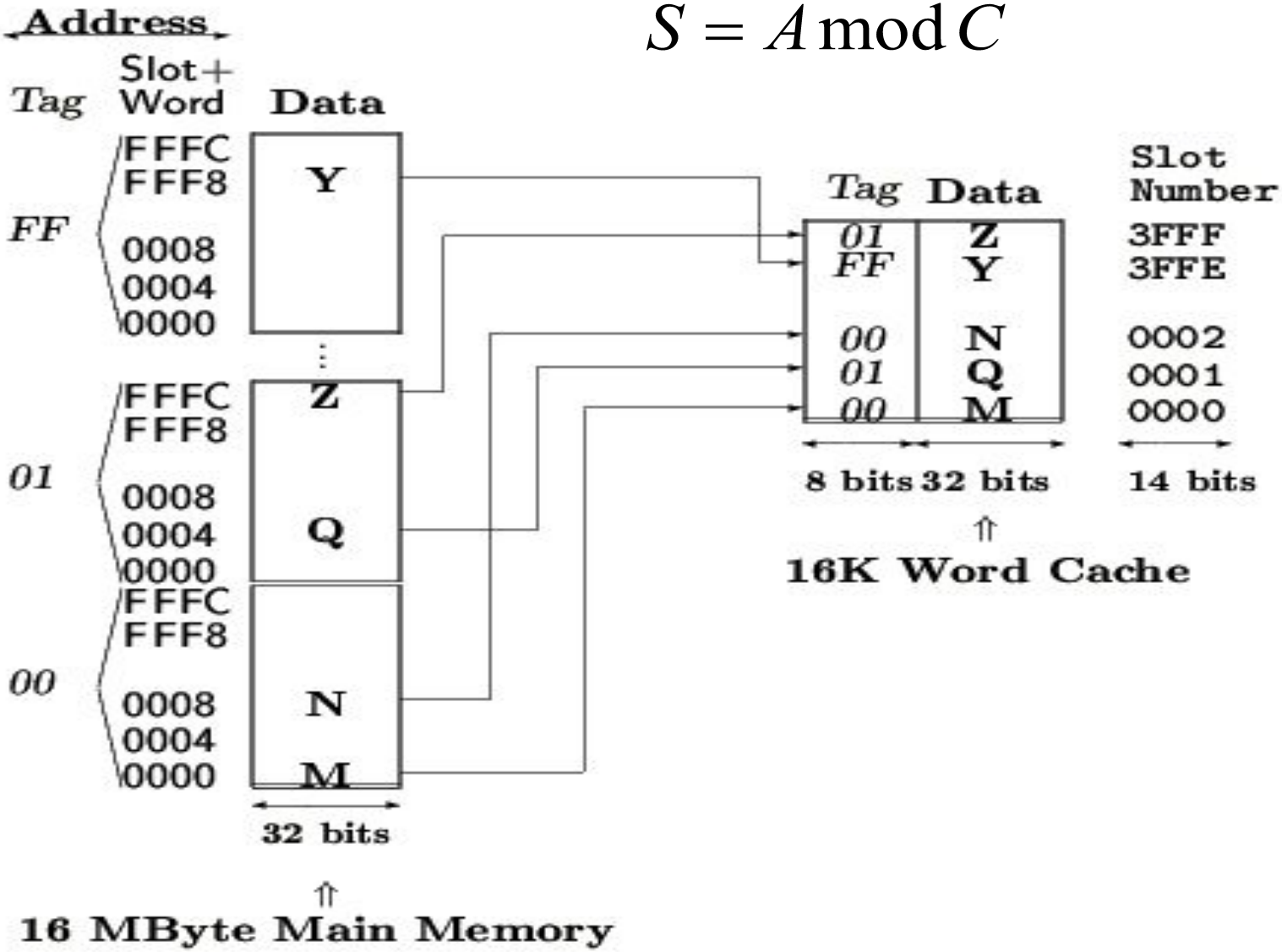
Cache = 64 Kbytes (2^{14}) rows of 4bytes each)

Word – 2 bits, rows -14 bits, tag – 8 bits

000000, 010000, …, FF0000 □ 0

000004, 010004, …, FF0004 □ 1

$$S = A \bmod C$$



MM Address =

Tag	Slot	Word
8	14	2

Figure 2: **Direct Mapping**

Пример «буксования» кэш-памяти (32 K) (*cache trashing)*

```
real*8 a(4096),b(4096),c(4096)
  common a,b,c
  do i=1,4096
    c(i)=a(i)+b(i)
  enddo
```

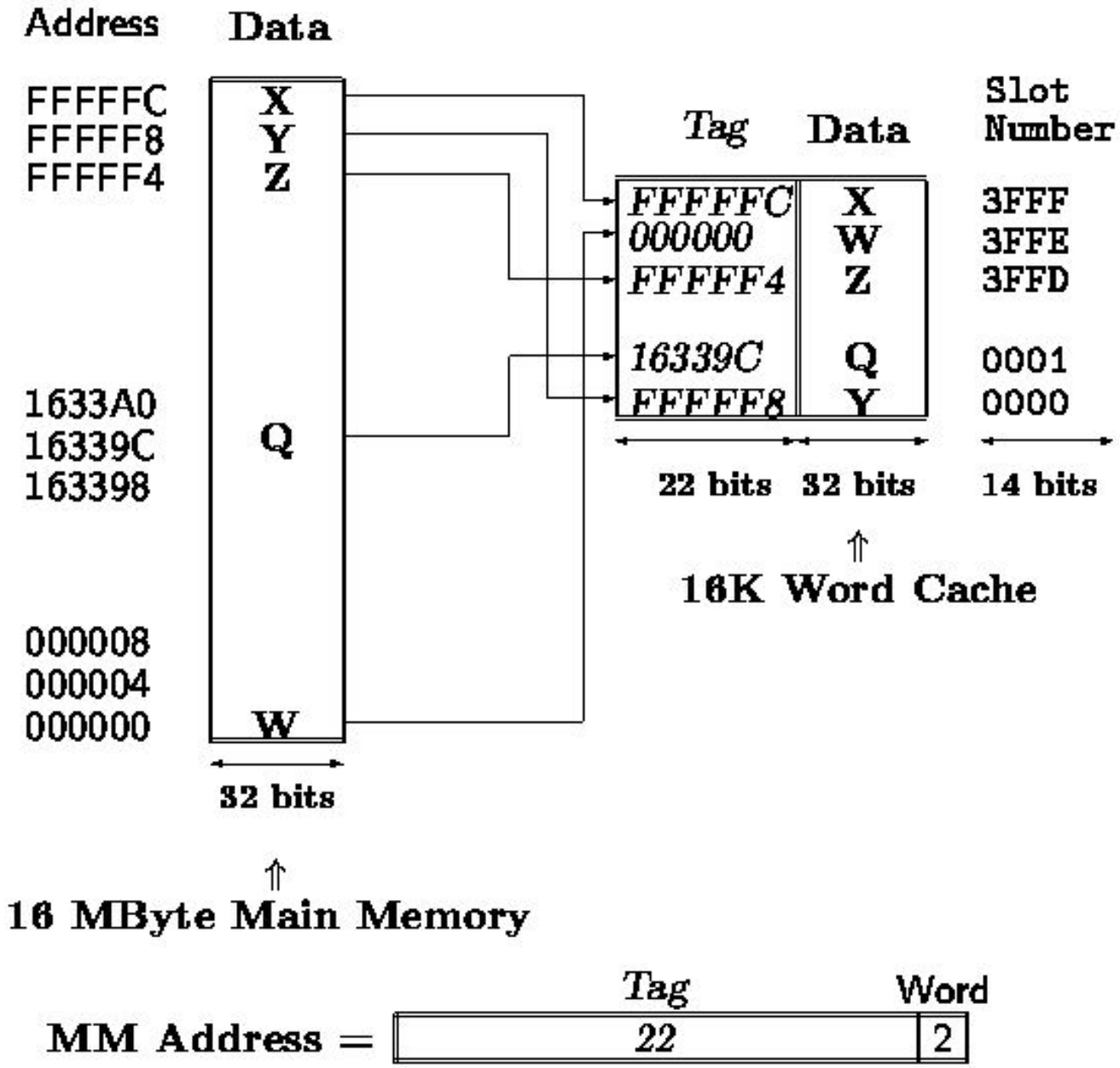
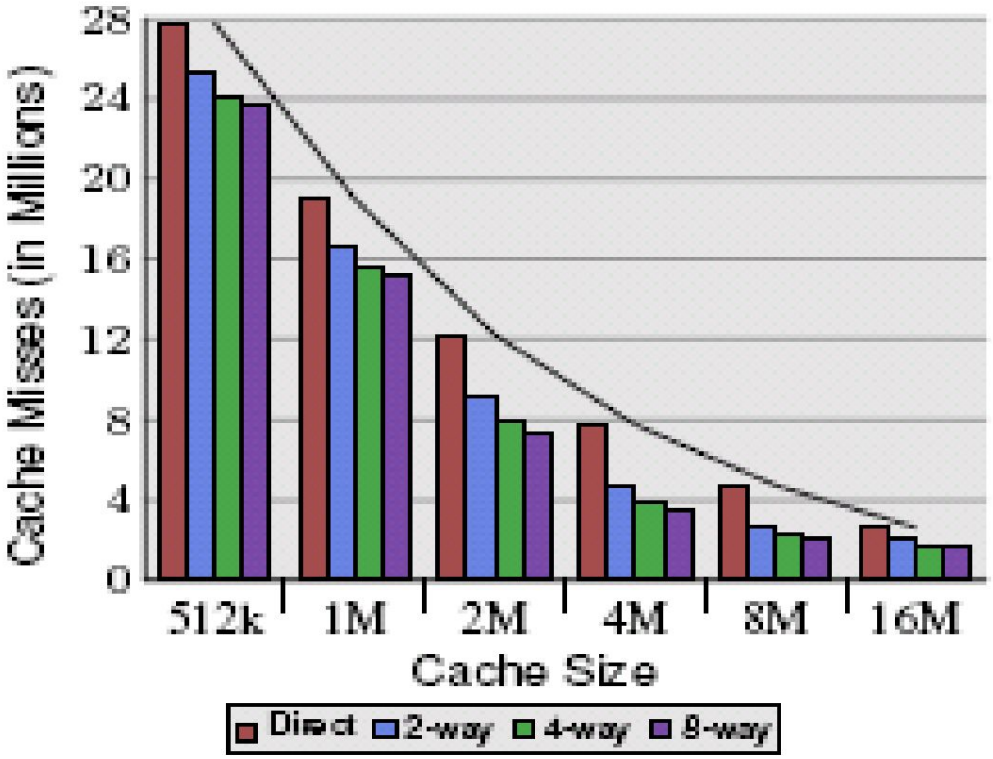
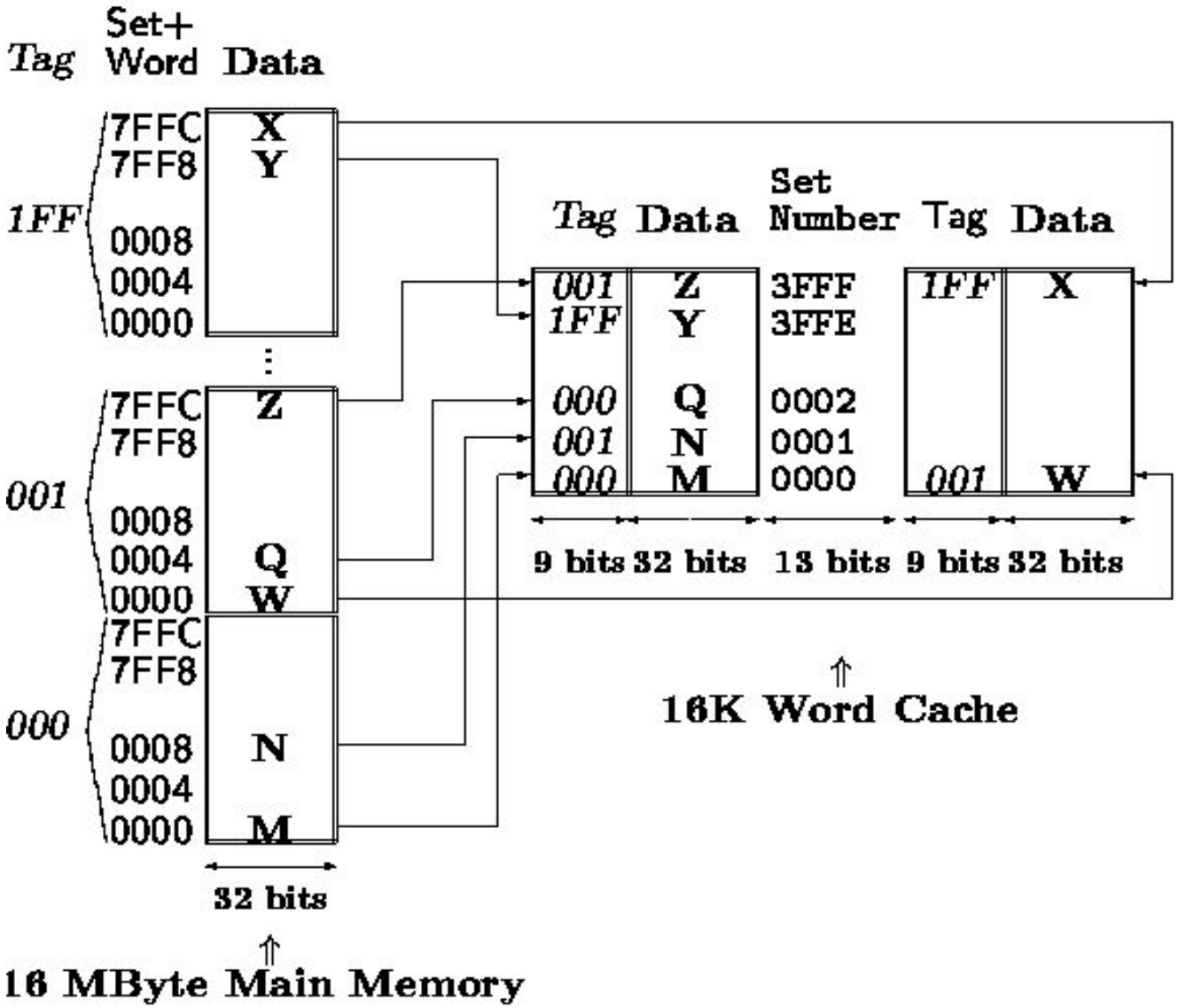


Figure 3: **Associative Mapping**

Зависимость количества промахов в кэш-память в зависимости от объема кэш-памяти и степени ассоциативности для длины строки 32 байта





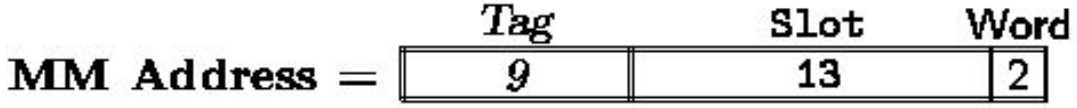


Figure 4: **Two-way Set-Associative Mapping**

Сравнение алгоритмов отображения адресов

- Прямой
  - 1 блок – 1 строка
  - Плюс: быстрый поиск, маленькие теги, простая реализация
  - Минус: пробуксовка кэша
- (Полностью) ассоциативный
  - 1 блок – любая строка
  - Плюс: нет пробуксовки кэша
  - Минус: медленный поиск, большие теги, сложная реализация
- Множественно-ассоциативный
  - 1 блок – несколько строк
  - Компромиссный вариант

Алгоритмы записи

- Сквозная запись **(Write Through (WT)).**
- Сквозная запись с буфери-зацией **(Write Combining).**
- Обратная запись **(Write Back (WB)).**

Алгоритмы замещения кэш-строк

- **Least Recently Used (LRU)**
- **Most Recently Used (MRU)**
- **Pseudo-Least Recently Used (PLRU)**

Алгоритм замещения (алгоритм псевдо-**LRU**)

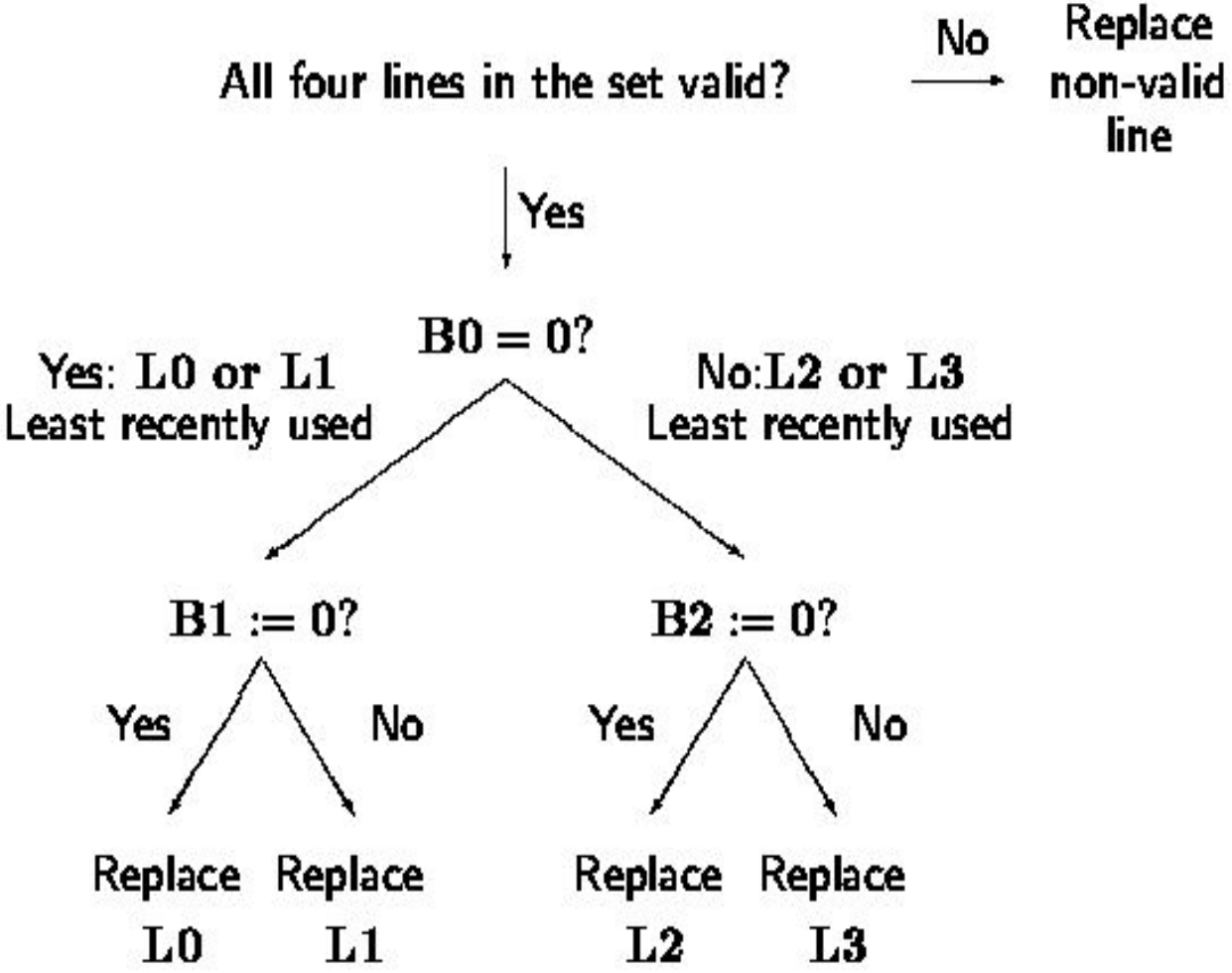


Figure 6: **Intel 80486 on-chip cache replacement algorithm**

Какими должны быть основные параметры кэша?

- **Размер кэша**
  - **Большой**, чтобы вместить рабочие данные
  - **Маленький**, для быстрого доступа
- **Степень ассоциативности кэша**
  - **Большая**, чтобы избегать пробуксовки
  - **Маленькая**, для быстрого доступа
- **Размер строки кэша**
  - **Большой**, чтобы использовать локальность
  - **Большой**, чтобы уменьшить теги
  - **Маленький** (доля полезных данных в кэше больше, если данные в памяти распределены произвольным образом)

Предвыборка команд и данных

Предвыборка **команд и данных** – это механизм уменьшения простоев процес-сора, связанные с ожиданием команд и данных. Этот механизм заключается в загрузке команд и данных в кэш-память из ОП до того, как они реально потре-буются. В результате при первом обра-щении к соответствующей ячейке ОП не возникает кэш-промах, поскольку запра- шиваемые данные и команды уже нахо-дятся в кэш-памяти.

Два типа предвыборки

Программная предвыборка

(программист или компилятор явно вставляет в программу команды предвыборки данных по тому или иному адресу в ОП.)

Аппаратная предвыборка

(кэш-контроллер анализирует, по каким адресам и в каком порядке программа обра-щается к ОП и пытается предугадать, какие данные вскоре могут понадобиться програм-ме, и осуществляет их автоматическую пред-выборку в кэш-память.)