Organização de Computadores Digitais

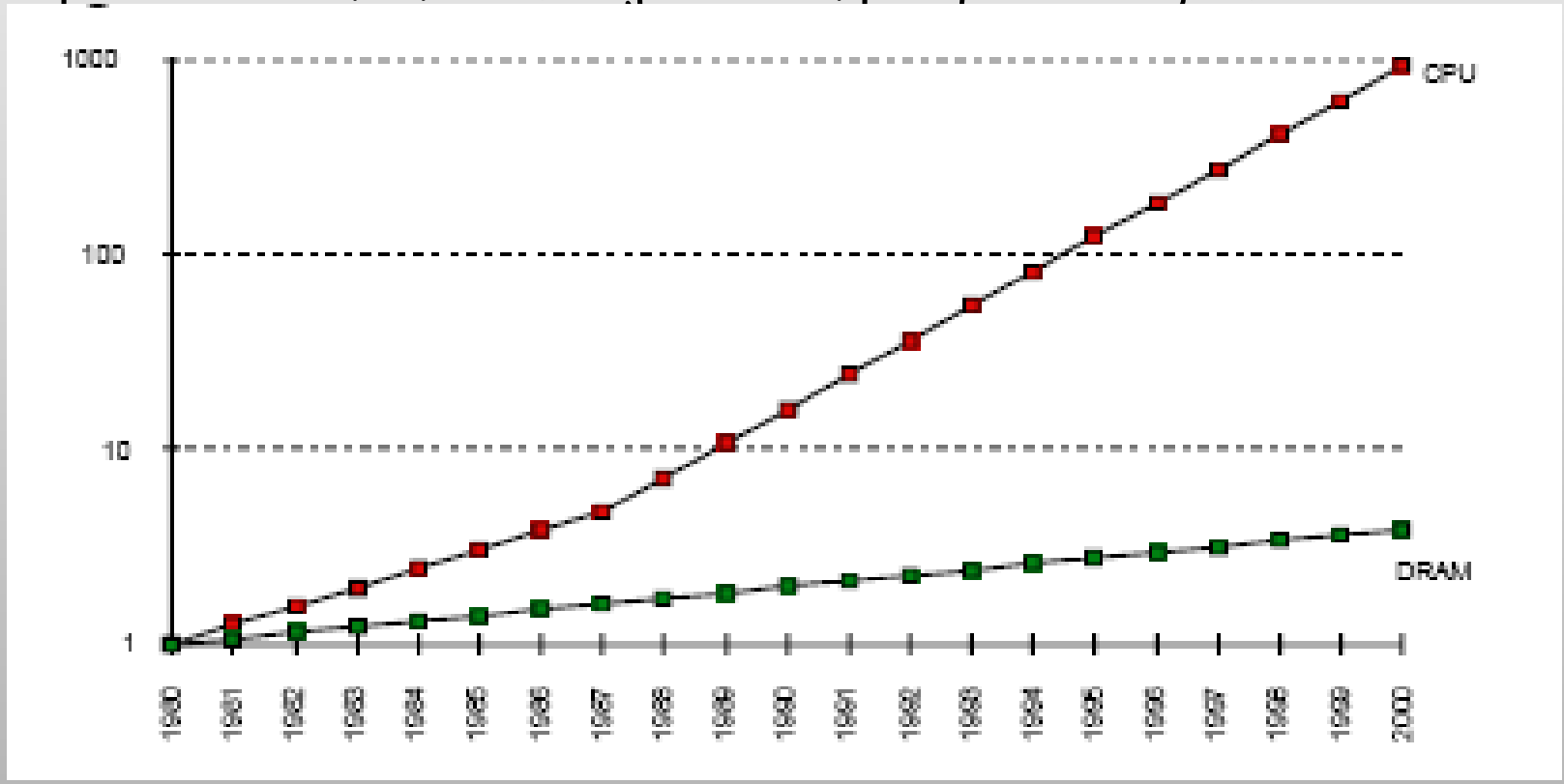
Capítulo 4 – Memória Cache

Porque a memória é importante?

- Os dispositivos de memória servem para armazenar programas (instruções) e dados (sequências de palavras em memória que não incluem qualquer código de operação)

Porque a memória é importante?

- O desempenho dos processadores aumentou a uma taxa muito superior a da memória. Veja o exemplo que se segue.



Características

- Localização
- Capacidade
- Unidade de transferência
- Método de acesso
- Desempenho
- Características físicas
- Tipo físico
- Organização

Localização

- A organização da memória rege-se por uma hierarquia de sistemas de memória. As principais são as seguintes:
  - A memória interna refere-se à memória principal, diretamente acessada pela CPU
  - A CPU necessita da sua própria memória local, na forma de registradores
  - A memória externa (memória secundária) é referida a todos os dispositivos usados para armazenar dados e que acessem a CPU através de controladores/módulos de Entrada/Saída

Capacidade

- Expressa em bytes ou words para a memória interna
- Os tamanhos mais usuais de palavras são de 8,16, 32 bits
- A capacidade de memória externa é expressa somente em bytes

Unidade de Transferência

- Interna
  - No nível da memória interna, refere-se ao número de linhas de dados que entram e saem do módulo de memória. Geralmente é igual à largura da palavra, mas nem sempre
  - É geralmente determinada pelo largura do barramento de dados
- Externa
  - É geralmente um bloco que é maior do que uma palavra
- Unidade Endereçável
  - O menor local de memória que pode ser endereçado

Métodos de Acesso(1)

- Sequencial
  - Começa no início e lê os dados sequencialmente
  - O tempo de acesso depende da localização dos dados e da posição anterior
  - Por exemplo, uma fita cassete
- Direto
  - Blocos individuais têm endereços únicos
  - O acesso é feito por salto e por procura sequencial
  - O tempo de acesso depende na localização dos dados e da localização anterior
  - Por exemplo, um disco

Métodos de Acesso(2)

- Aleatório
  - Endereços individuais identificam locais exatos
  - O tempo de acesso é independente do local ou de acessos anteriores
  - Por exemplo, a RAM
- Associativa
  - Os dados são localizados por comparação de parte dos dados armazenados
  - O tempo de acesso é independente do local ou de acesso anteriores
  - Por exemplo, a cache

Desempenho (1)

- Tempo de acesso (Access time)
  - Para uma memória de acesso aleatório: é o tempo que demora a executar uma operação de leitura ou escrita, ou seja, é o tempo a partir do instante que o endereço é apresentado à memória até ao instante que os dados foram arquivados ou tenham ficado disponíveis para uso
  - Para os outros tipos de memória, é o tempo que demora o mecanismo de leitura-escrita a posicionar-se na localização desejada

Desempenho (2)

- Tempo de ciclo de memória (Memory cycle time)
  - Compreende o tempo de acesso e o tempo necessário a memória para recuperar antes que outro acesso possa ser realizado
  - Tempo de ciclo de memória = acesso + recuperar
- Taxa de transferência (Transfer Rate)
  - Taxa a qual os dados podem ser transferidos
  - Para memória aleatórias é 1/(Tempo de Ciclo)
  - Para as outras memórias: TN = TA + N/R
    - TN - Tempo médio de leitura ou escrita de N bits
    - TA - Tempo médio de acesso
    - N - Número de bits
    - R - Taxa de transferência, em bits por segundo (bps)

Tipos de memória física

- Semicondutora
  - RAM
- Magnética
  - Disco & Fita
- Ótica
  - CD & DVD
- Outros
  - Magneto -optica

Memória
Questões importantes

- As questões no desenho de uma memória de um computador podem ser resumidas a três questões
  - Dimensão
  - Velocidade
  - Custo
- A relação entre estas características terá de ser considerada na escolha da memória
  - Quanto menor o tempo de acesso, maior o custo por bit
  - Quanto maior a capacidade, menor o custo por bit
  - Quanto maior a capacidade, maior o tempo de acesso

Hierarquia de memória

- A forma escolhida para ter em consideração todas estas relações foi não ter apenas uma única memória mas sim ter uma hierarquia de memórias
- À medida que descemos na hierarquia da memória temos:
  - custo por bit decrescente
  - capacidade crescente
  - tempo de acesso crescente
  - frequência de acesso à memória pelo CPU decrescente

Lista da Hierarquia

Maior capacidade

Menor custo por bit

- Registradores
- L1 Cache
- L2 Cache
- Memória principal
- Cache de disco
- Disco
- Óptico
- Fita

Mais Rápida

Maior Frequência de Uso

Hierarquia de memória

Tempo de acesso

Mais Rápida

- Registradores — 5 nanoseg
- L1 Cache
- L2 Cache
- Memória principal — 70 nanoseg
- Cache de disco
- Disco — 10.000.000 nanoseg(10 miliseg)
- Óptico
- Fita — 10.000.000.000 nanoseg (10 seg)

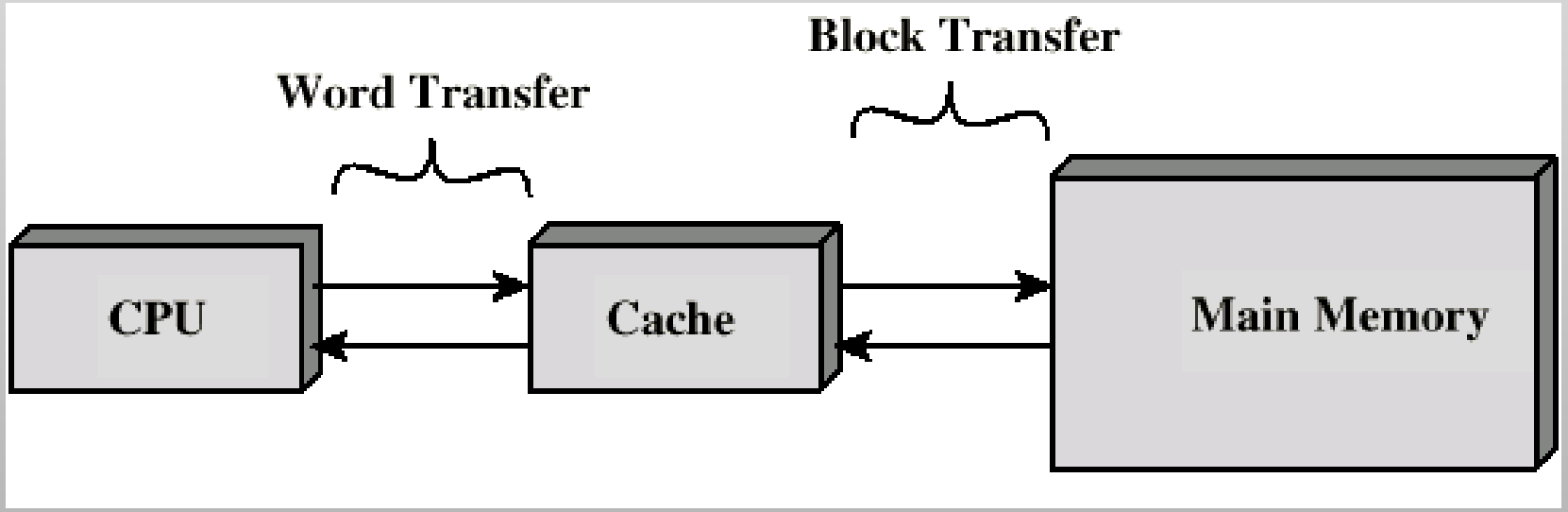
Memória Cache

Memória Cache

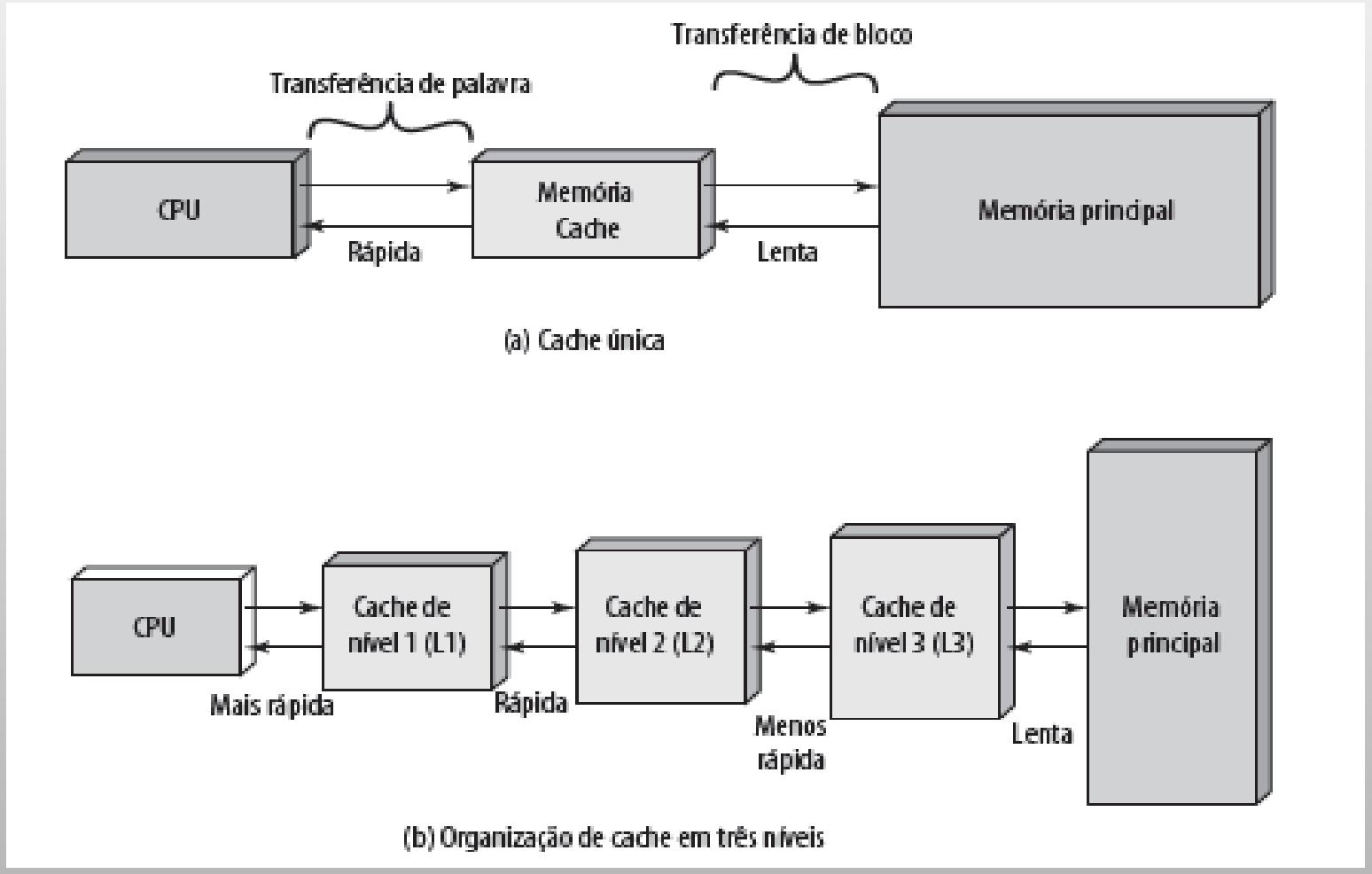
- O tempo de execução de uma instrução é inferior ao tempo de acesso à memória
- A taxa que o CPU pode executar as instruções é claramente limitada pelo ciclo de acesso à memória
- Como o tempo de acesso à memória não acompanha o tempo de ciclo do processador, a solução consiste em colocar uma memória pequena e rápida entre o processador e a memória principal
- Esta memória é denominada memória cache.

Cache

- Pequena memória rápida
- Posiciona-se entre a memória principal e a CPU
- Pode estar localizada na CPU ou num módulo



Cache e memória principal



Operações da Cache

- A CPU pede dados a memória principal
- A cache é pesquisada a procura dos dados
- Se estes estão presentes, os dados são obtidos rapidamente
- Se não estão presentes, um bloco de dados é lido da memória principal para a cache
- Os dados são então entregas da cache a CPU
- A cache inclui identificadores (tags) para identificar os blocos de memória principal presentes na cache

Estrutura da Cache

- Para uma memória principal de 2^n palavras endereçadas, dividida em blocos de K palavras ($M=2^n/K$ blocos), a memória cache é organizada em C slots de tamanho fixo que contêm K palavras cada
- Como é óbvio $C<<M$
- Por razões de eficiência, as leituras para (e da) cache são feitas em bloco e não palavra a palavra

Estrutura da Cache

- Como existem mais blocos de memória que slots (C<<M), um slot individual não pode estar permanentemente dedicado a um bloco
- Então cada slot inclui uma tag que identifica o bloco que está ser guardado
- Esta tag é usualmente parte do endereço da memória principal

Estrutura da Cache

- A CPU gera um endereço de uma palavra para ser lida
- Se a palavra estiver em cache, é entregue a CPU, caso contrário, o bloco que contém a palavra é carregado da cache e entregue a CPU

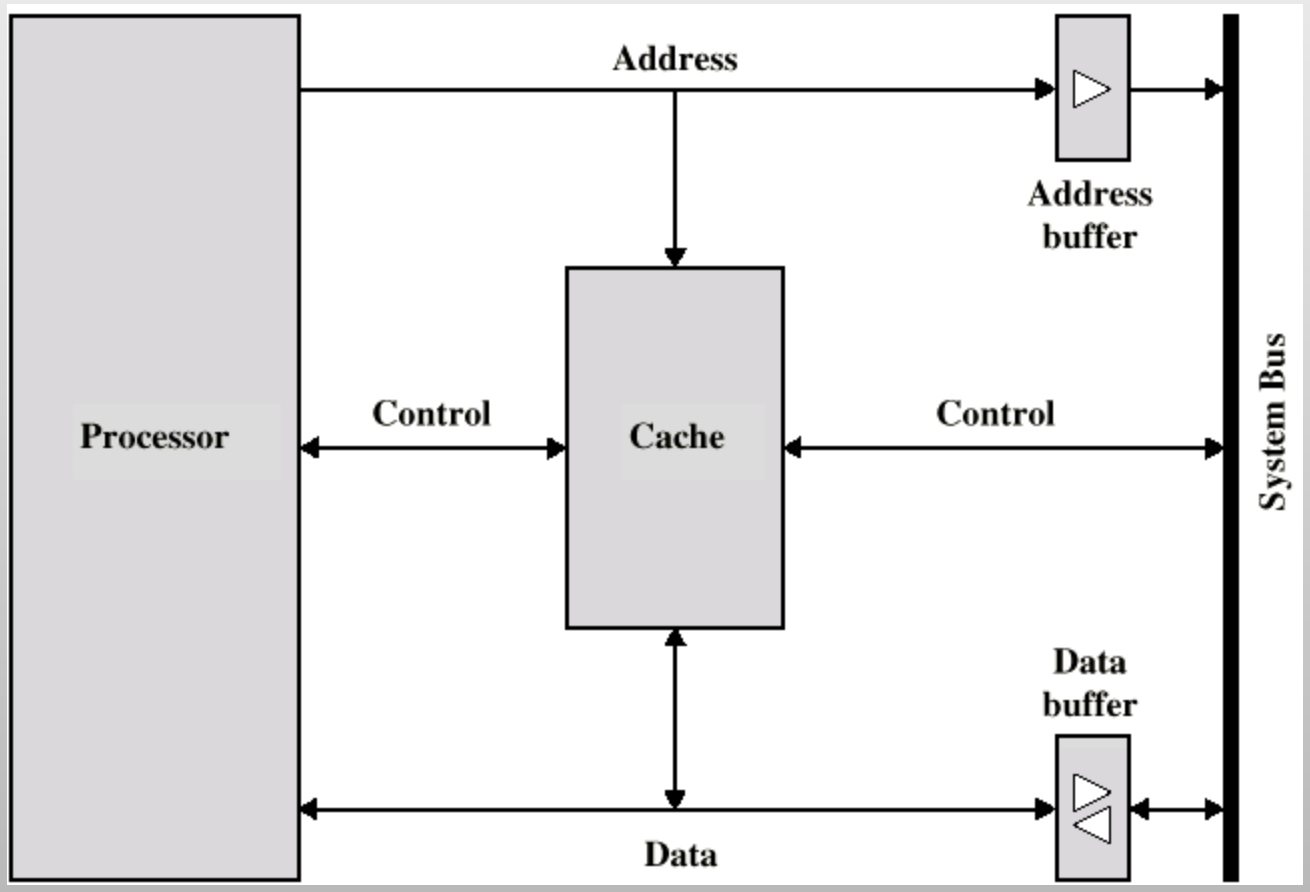
Cache Design

- Tamanho
- Função de Mapeamento
  - Direto, associativo, associativo por conjuntos
- Algoritmo de Substituição
  - LRU, FIFO, LFU
- Regras de Escrita
  - Write-through, write-back, write-once
- Dimensão dos Blocos
- Número de Caches
  - Um ou dois níveis
  - Unificada ou separada

Tamanho é importante

- Custo
  - Ter uma maior cache é mais dispendioso
- Velocidade
  - Uma maior cache aumenta a velocidade (até um limite)
  - Pesquisar a cache a procura de dados demora tempo

Organização Típica da Cache



Função de Mapeamento Exemplo

- 16MBytes memória principal
- Endereços de 24 bit
  - (2^{24}=16M)
- Cache de 64kByte
- Cache com blocos de 4 bytes
  - i.e. cache tem 16k (2^{14}) linhas de 4 bytes

Mapeamento Direto

- Tamanho da cache = 32KB
- Tamanho do bloco = 16 bytes
  - i.e. cache tem 2K (2^{11}) linhas de 16 bytes
- Memória principal =16 MB
  - Endereços de 24 bit (2^{24}=16M)
  - 2^{20} (1M) blocos de 2^{4} bytes

Mapeamento Direto

- Cada bloco da memória principal é mapeado a uma só linha da cache
  - i.e. se um bloco está na cache, este tem de estar numa posição específica
- Fórmula para mapear um bloco de memória a uma linha de cache : $i = j \bmod c$
  - i= Número da Linha da Cache
  - j= Número do Bloco da Memória Principal
  - c= Número de Linhas na Cache

Estrutura dos Endereços
Mapeamento Direto

- Os w bits menos significativos identificam uma palavra numa linha da cache
- Os próximos s bits significativos identificam em que slot um endereço é mapeado
- Os últimos t bits significativos identificam um bloco de memória

Mapeamento Direto com C=4

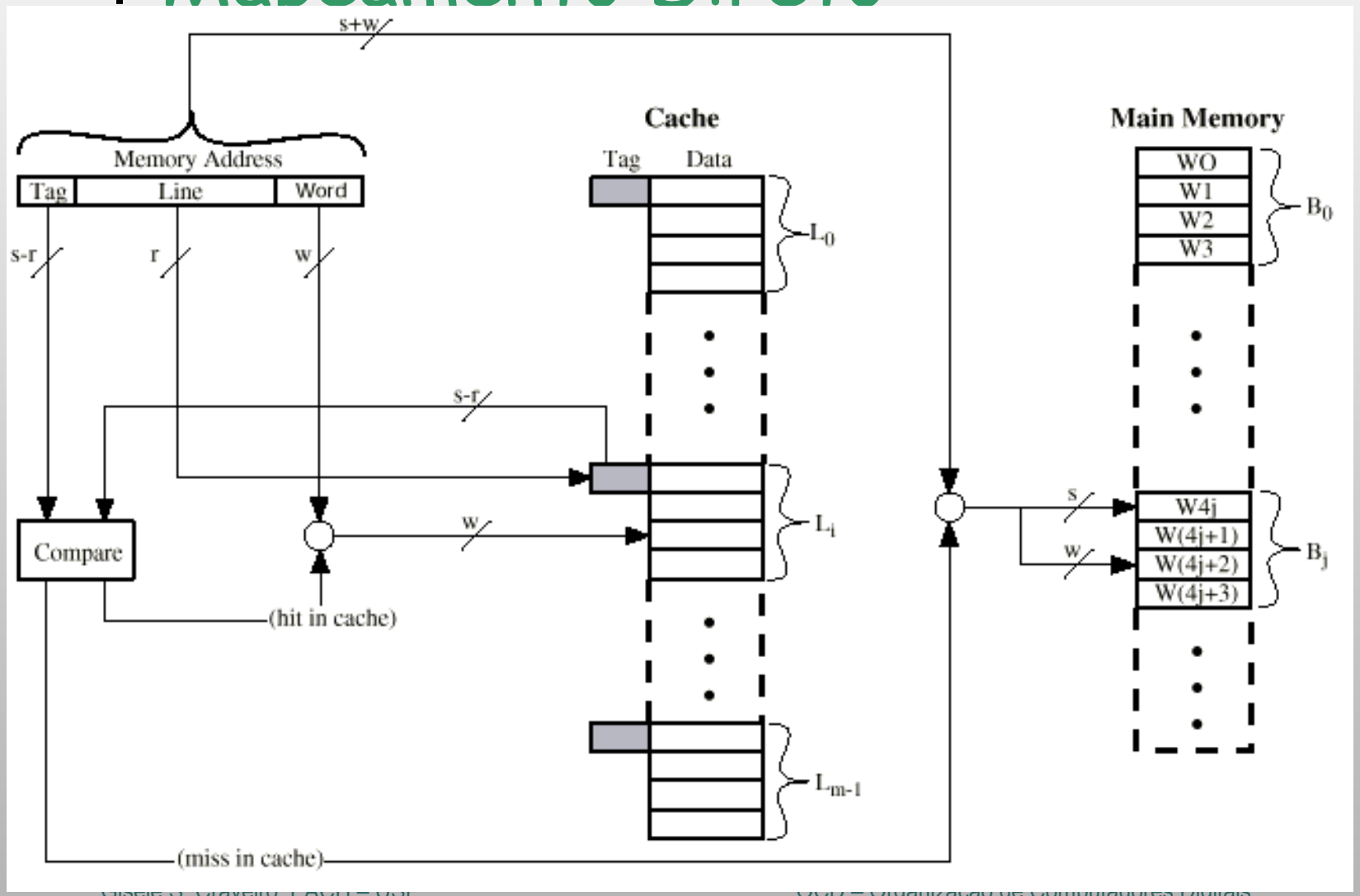
- Linha da Cache — Bloco de Memória Principal
  - 0 — 0, 4, 8, …
  - 1 — 1, 5, 9, …
  - 2 — 2, 6, 10, …
  - 3 — 3, 7, 11, …
- Regra Geral:
  - 0 — 0, C, 2C, 3C, …
  - 1 — 1, C+1, 2C+1, 3C+1, …
  - 2 — 2, C+2, 2C+2, 3C+2, …
  - 3 — 3, C+3, 2C+3, 3C+3, …

Estrutura dos Endereços
Mapeamento Direto

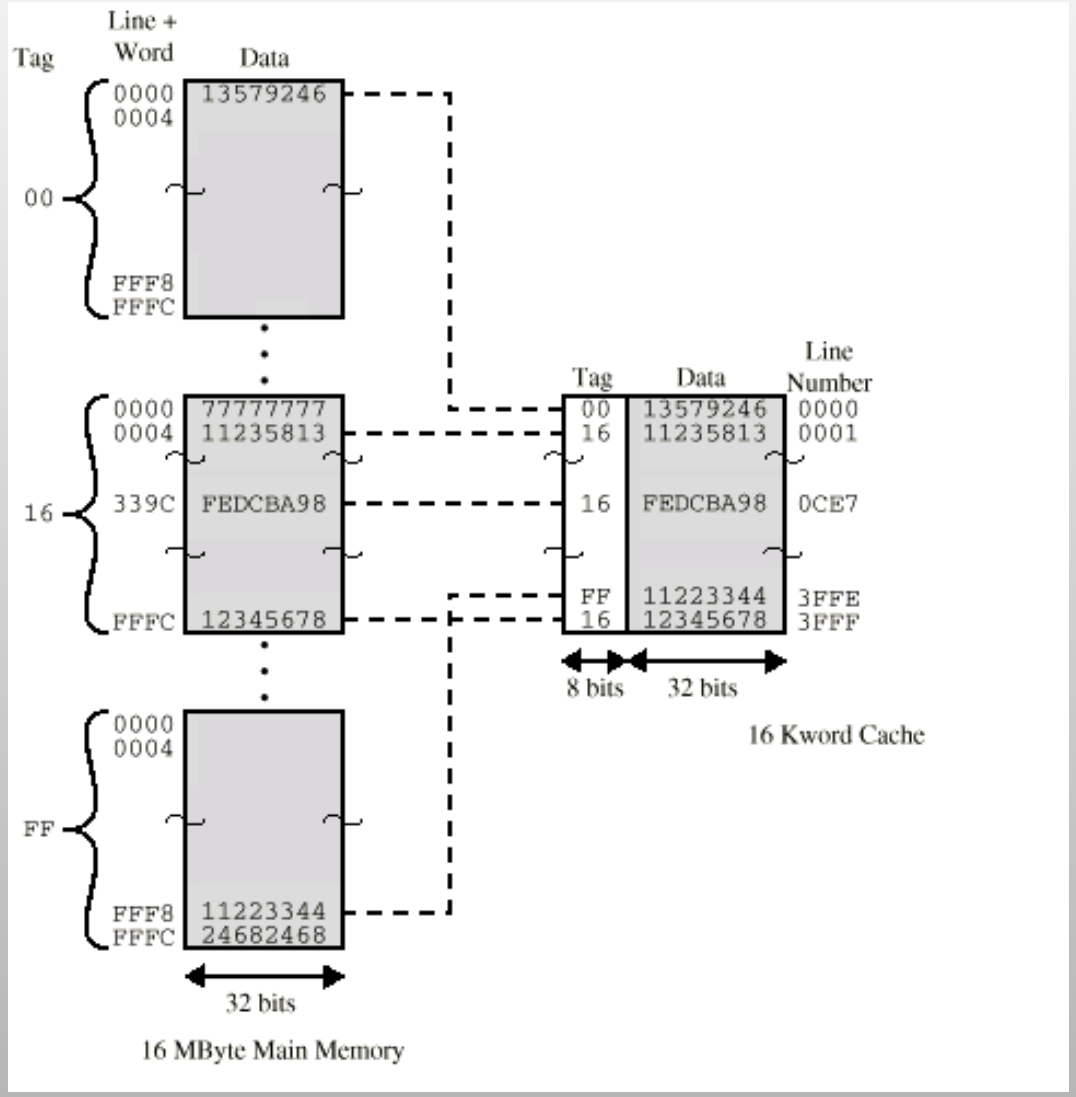
- Endereços de 24 bit
- Identificador de palavra: 2 bits (blocos de 4 bytes)
- Identificador de bloco: 22 bit
  - Tag de 8 bit (=22-14)
  - Linhas de 14 bit
- Dois blocos na mesma linha nunca têm o mesmo Tag
- Encontram-se dados na cache procurando a linha (r) e comparando a Tag (s-r)

Mapeamento Direto

Mapeamento Direto



Direct Mapping Exemplo



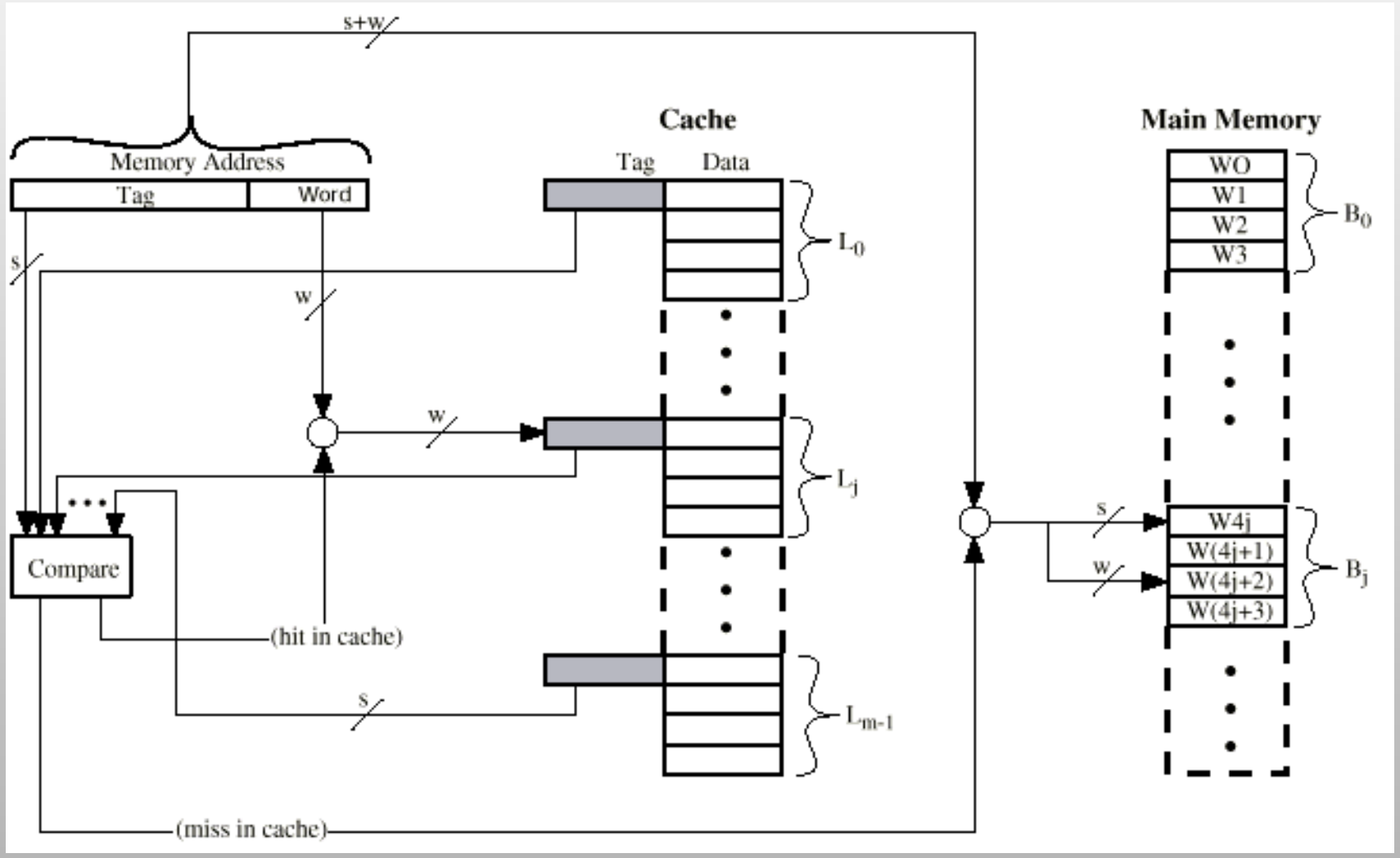
Mapeamento Direto prós & contras

- Simples
- Barato
- Localização fixa para os blocos
  - Se um programa acede a dois blocos que mapeiam na mesma linha repetidamente, os cache misses tornam-se muito elevados

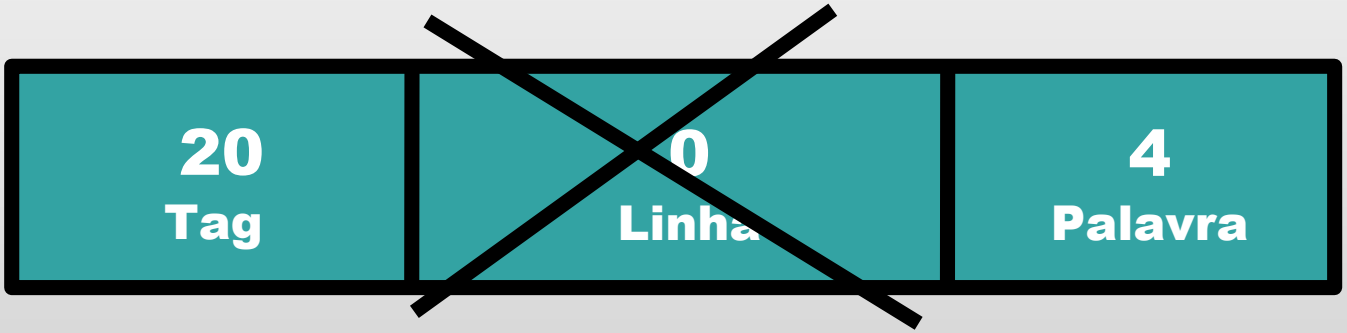
Mapeamento Associativo

- Um bloco de memória pode ser carregado em qualquer linha da cache
- Um endereço de memória é composto por tag e palavra
- O tag identifica o bloco de memória
- Cada linha de uma tag é examinada a procura de um match

Organização de Cache Associativa

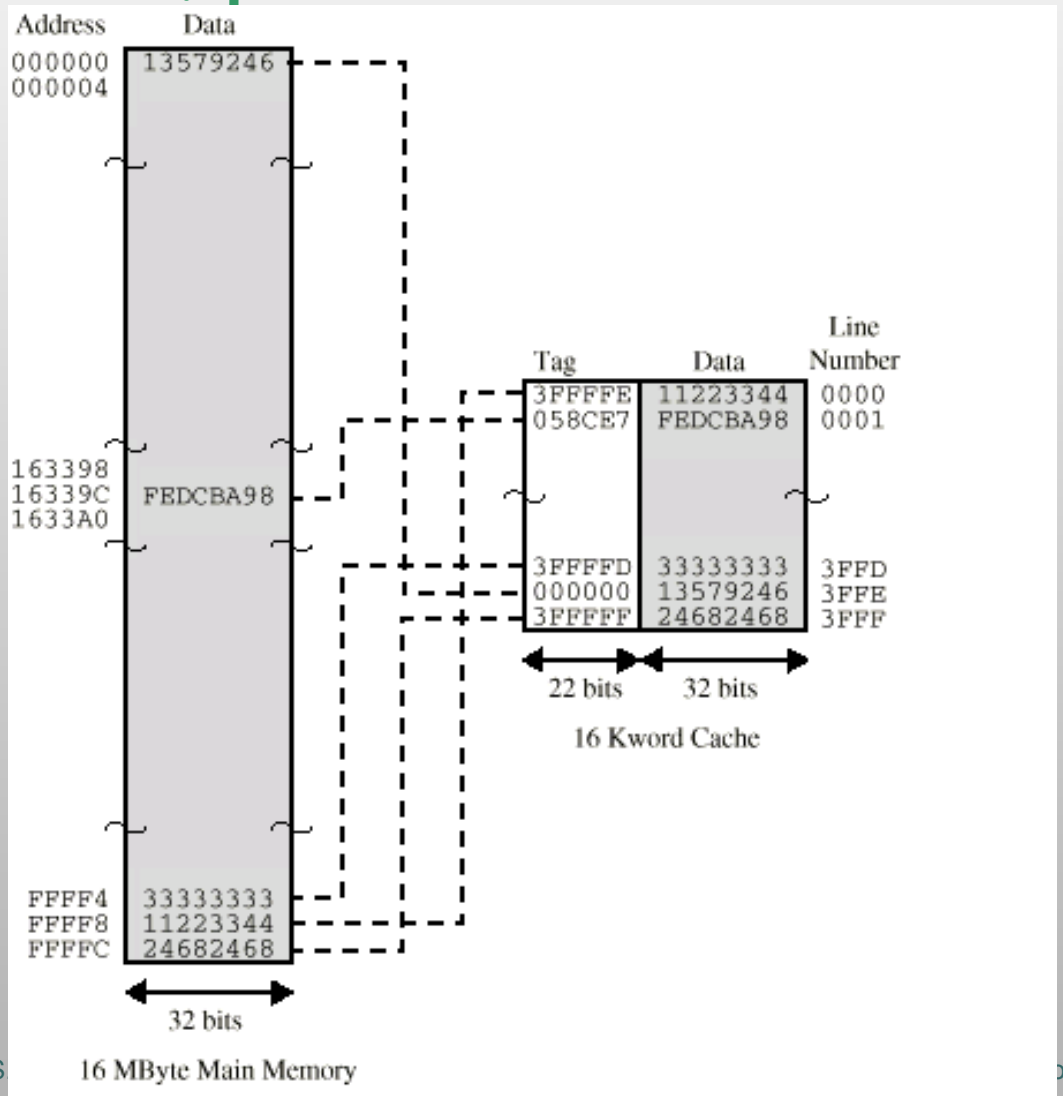


Estrutura dos Endereços
Mapeamento Associativo



- Tamanho da cache = 32KB
- Tamanho do bloco = 16 bytes
- Memória principal =16 MB
- 2^{24} bytes

Mapeamento Associativo Exemplo



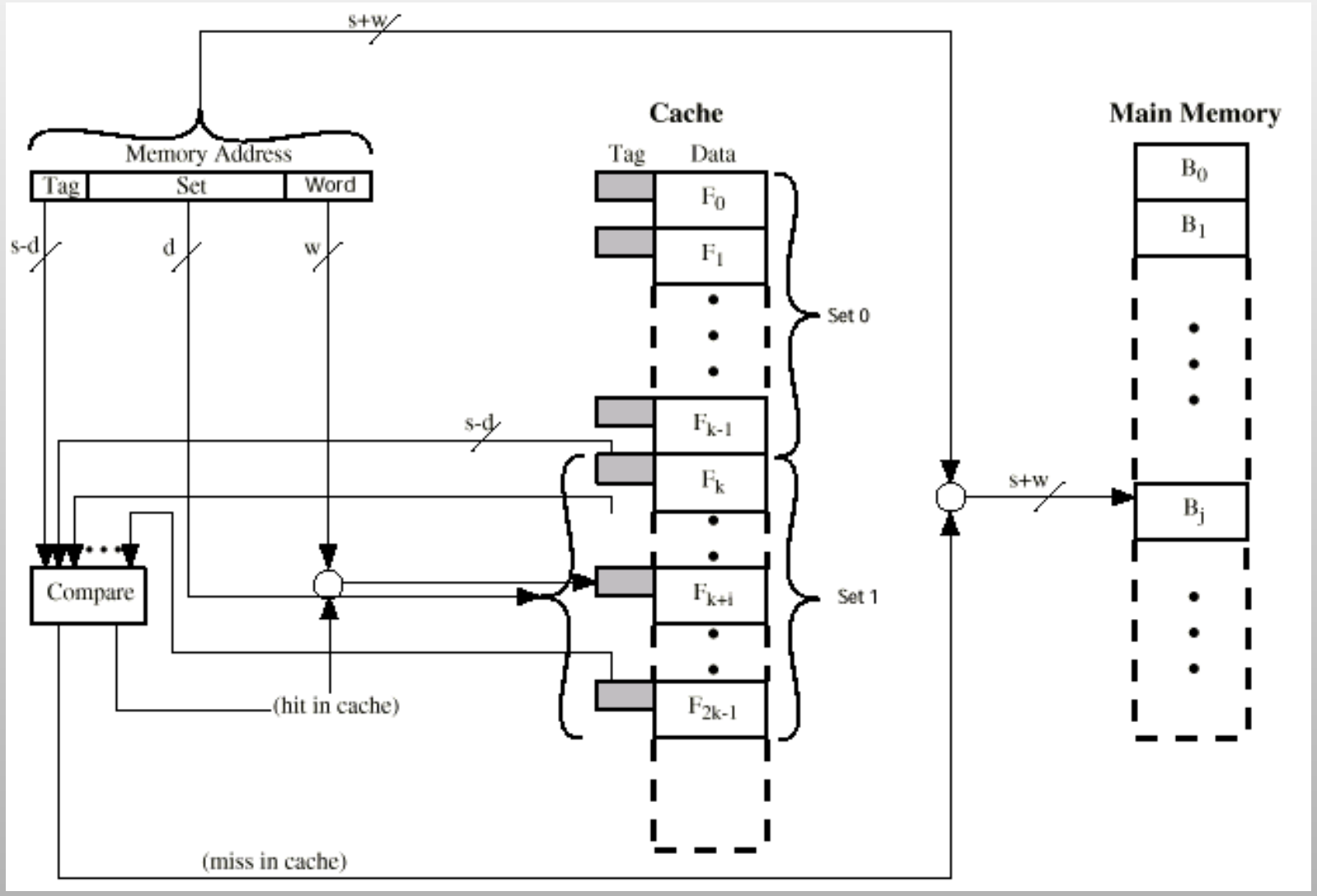
Mapeamento Associativo

- A pesquisa da cache é cara
  - Idealmente seria necessário um circuito que pode simultaneamente examinar todas as tags ao mesmo tempo
  - Implicaria a utilização de muitos circuitos
    - custo elevado
- Como qualquer dado pode ser expulso da cache é necessário desenvolver regras de substituição

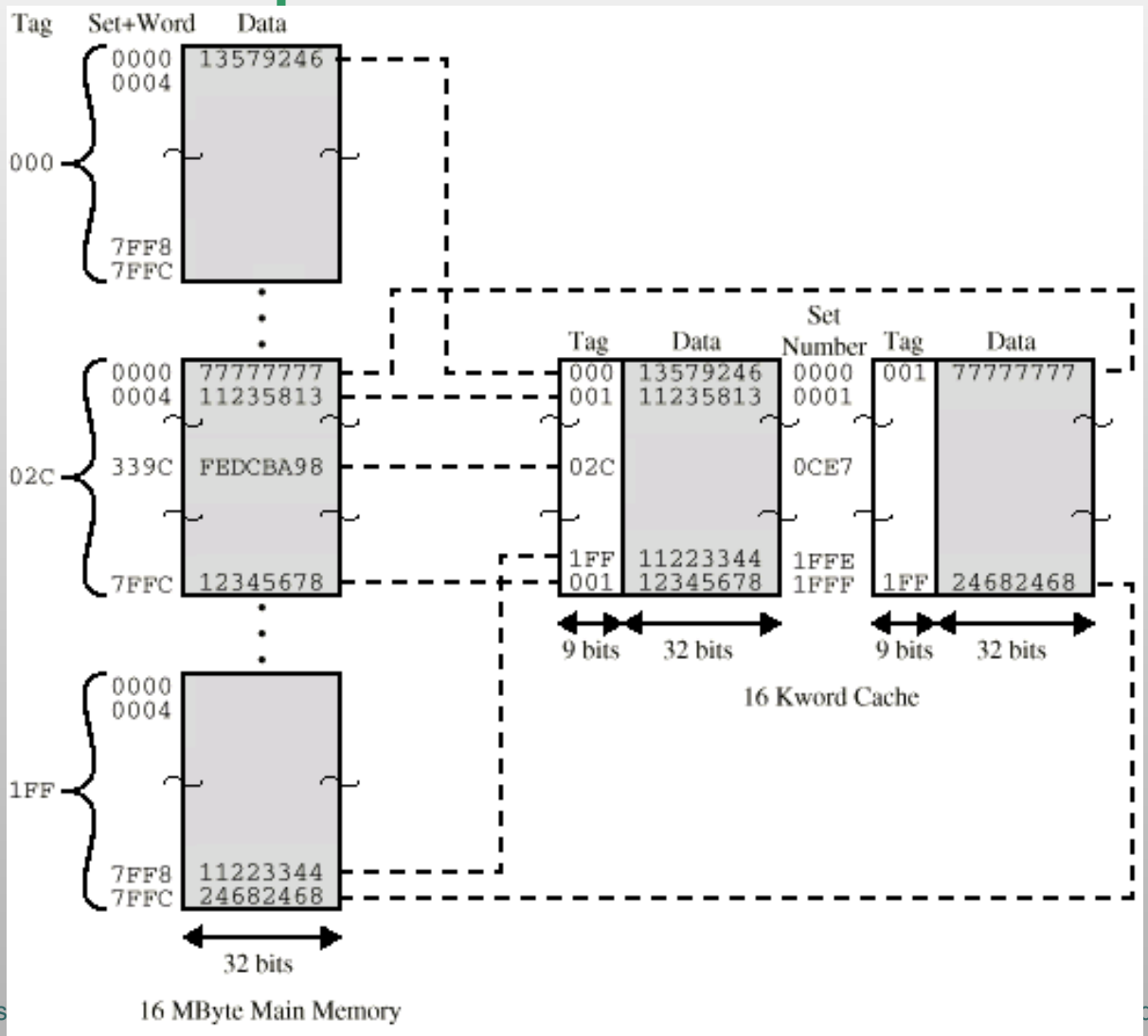
Mapeamento Associativo por Conjuntos

- Divide a cache em v conjuntos de k linhas
  - m = v x k
  - i= j modulo v , onde
    - i = número do conjunto na cache
    - j = número do bloco na memória principal
    - m = número de linhas da Cache
- Com este tipo de mapeamento, o bloco B pode ser mapeado em qualquer linha do conjunto i
- Permite capturar as vantagens dos dois métodos anteriores
  - Nos casos extremos se v= m, k=1(direta); se v=1, k = m(associativa)

Organização de Cache com Mapeamento Associativo por Conjunto



Mapeamento Associativo Exemplo



Algoritmos de Substituição

- Qual o bloco a substituir? (se o bloco não está na cache)
- Mapeamento Direto
  - Só existe uma opção
- Mapeamento Associativo
  - Cada bloco é um candidato
- Associativa de nível k
  - Um de k

Algoritmos de Substituição

- A substituição de blocos na cache pode ser
- implementada por diversos algoritmos
  - LRU (least recently used)
  - FIFO (Firt-in First Out)
  - LFU (least frequently used)
  - Random (aleatoriamente)

Política de Escrita

- Existem duas técnicas de escrita de blocos modificados na memória principal
  - Write-Throught - as operações são executadas em simultâneo na cache e na memória
  - Write-back - as operações são executadas apenas na cache e atualizando a memória principal apenas quando o bloco é substituído

Tamanho de linha

- Recupere não apenas a palavra desejada, mas também uma série de palavras adjacentes.
- Tamanho de bloco aumentado aumentará razão de acerto a princípio.
  - O princípio da localidade.
- Razão de acerto diminuirá à medida que o bloco se tornar ainda maior.
  - Probabilidade de uso de informações recém- buscadas torna-se menor que probabilidade de reutilizar informações substituídas.

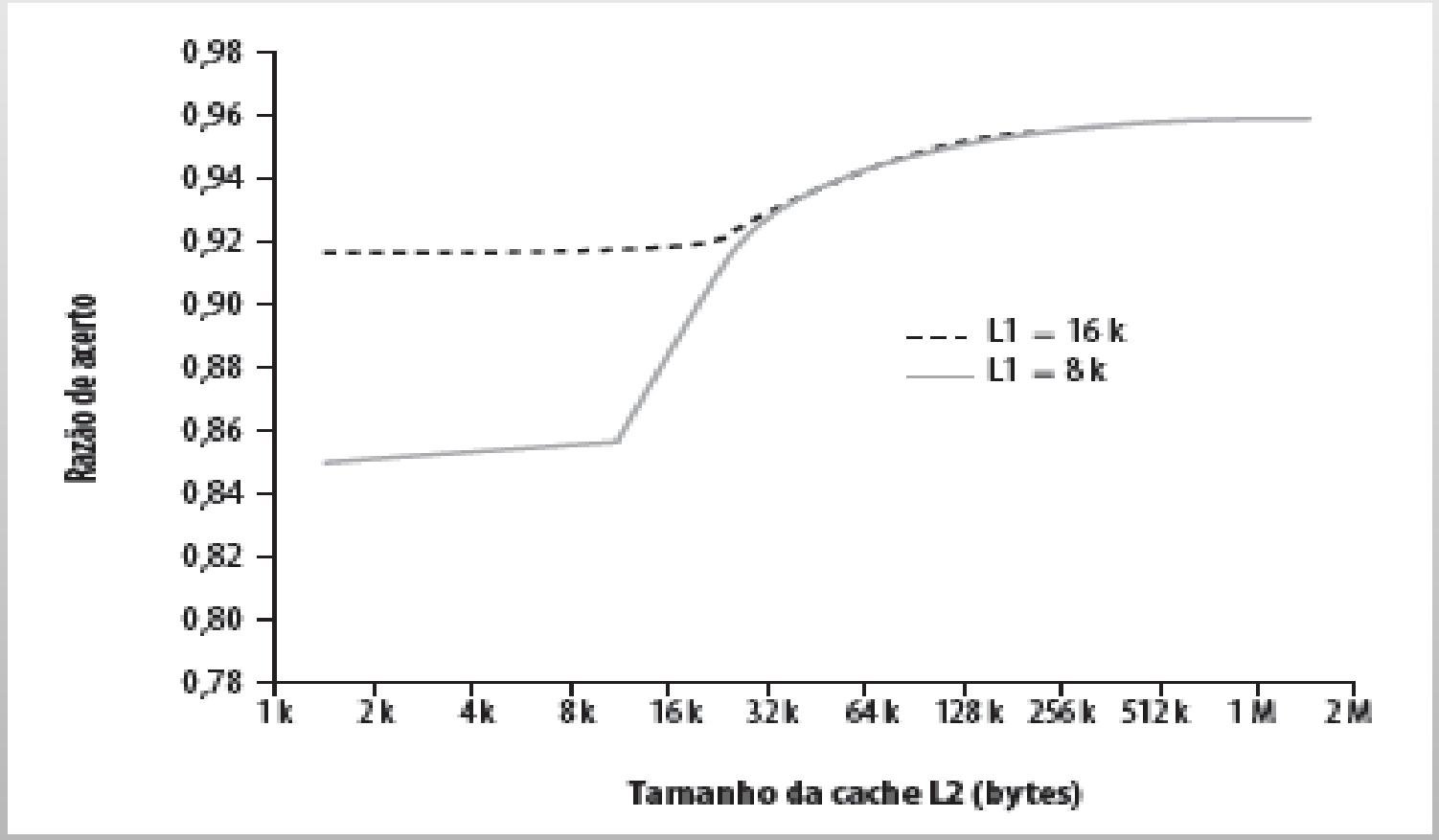
- Blocos maiores:
  - Reduzem número de blocos que cabem na cache.
  - Dados sobrescritos pouco depois de serem buscados.
  - Cada palavra adicional é menos local, de modo que é menos provável de ser necessária.
- Nenhum valor ideal definitivo foi descoberto.
- 8 a 64 bytes parece ser razoável.
- Para sistemas HP, 64 e 128 bytes mais comum.

Caches multinível

- Alta densidade lógica permite caches no chip.
  - Mais rápido que acesso ao barramento.
  - Libera barramento para outras transferências.
- Comum usar cache dentro e fora do chip.
  - L1 no chip, L2 fora do chip na RAM estática.
  - Acesso L2 muito mais rápido que DRAM ou ROM.
  - L2 normalmente usa caminho de dados separado.
  - L2 pode agora estar no chip.
  - Resultando em cache L3.
    - Acesso ao barramento agora no chip.

Razão de acerto total (L1 & L2)

Para L1 de 8 KB e 16 KB



Caches unificadas *versus* separadas

- Uma cache para dados e instruções ou duas, uma para dados e uma para instruções.
- Vantagens da cache unificada:
  - Maior taxa de acerto.
    - Equilibra carga entre buscas de instrução e dados.
    - Apenas uma cache para projetar e implementar.
- Vantagens da cache separada:
  - Elimina disputa pela cache entre a unidade de busca/decodificação de instrução e a unidade de execução.
    - Importante no pipeline de instruções.