

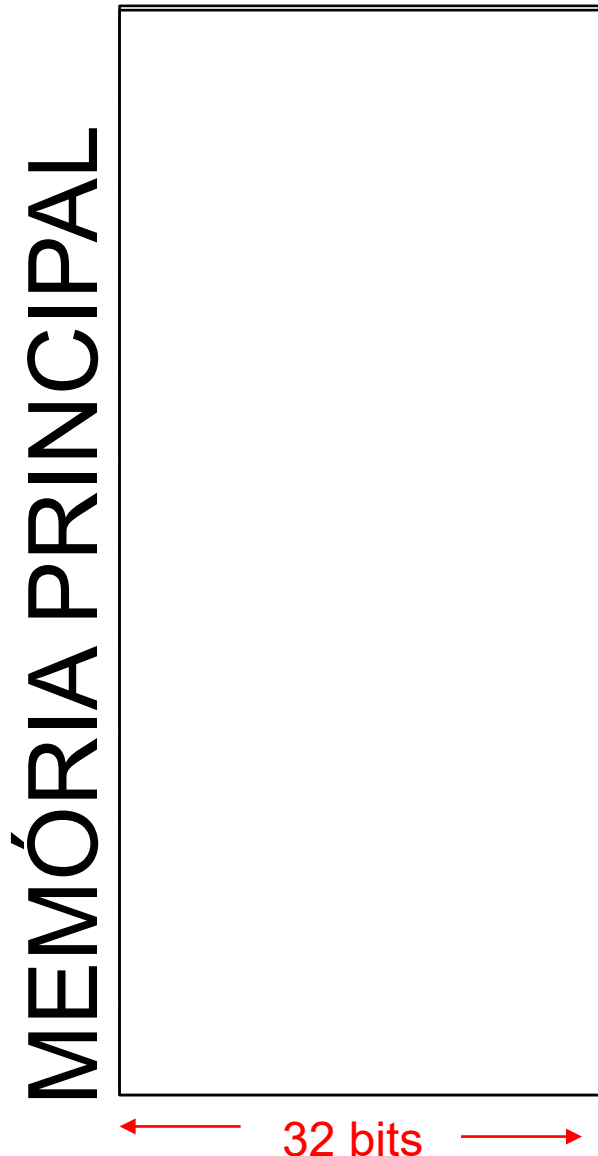
Memórias Cache

Prof. Gustavo Girão

Como funcionam?

- Baseiam-se no princípio da localidade
 - Espacial
 - Temporal
- Atuam entre as solicitações do processador e a memória principal
- Recebem a solicitação do processador e resultarão em dois casos:
 - Cache Hit
 - ✧ O dado é imediatamente enviado ao processador
 - Cache Miss
 - ✧ O dado é solicitado à memória principal e teremos um tempo de espera chamado de Penalidade de Miss

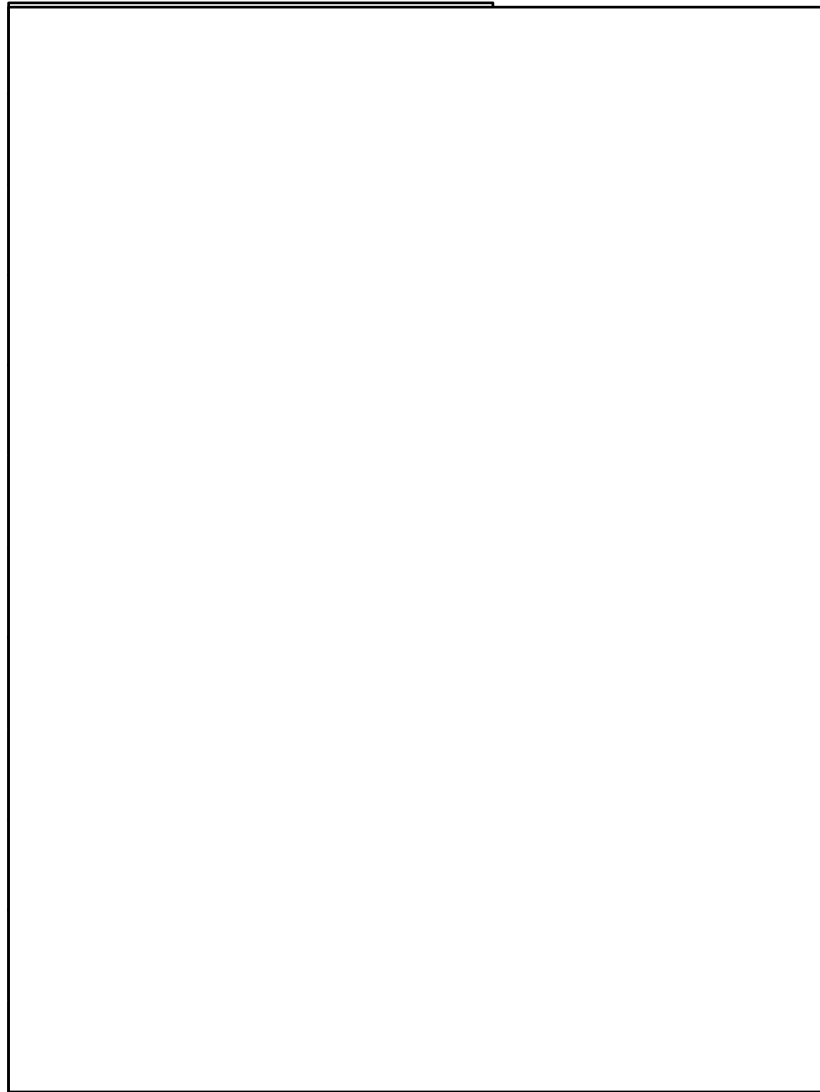
Divisão da Memória



- Decisão de projeto:
 - Qual o tamanho da unidade mínima de transferência da memória?
 - ✧ Normalmente está associado ao tamanho palavra do processador
 - ✧ Isso é comumente chamado de “largura” da memória

Divisão da Memória

MEMÓRIA PRINCIPAL

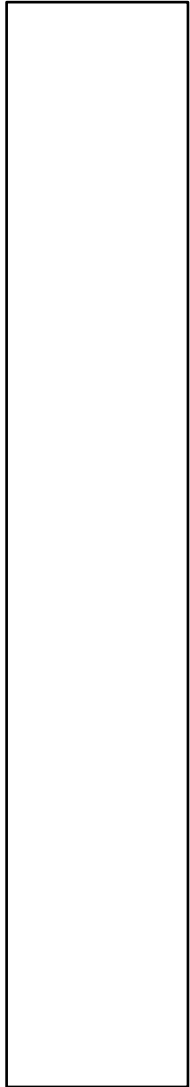


64 bits

- Decisão de projeto:
 - Qual o tamanho da unidade mínima de transferência da memória?
 - ✧ Normalmente está associado ao tamanho palavra do processador
 - ✧ Isso é comumente chamado de “largura” da memória

Divisão da Memória

MEMÓRIA PRINCIPAL

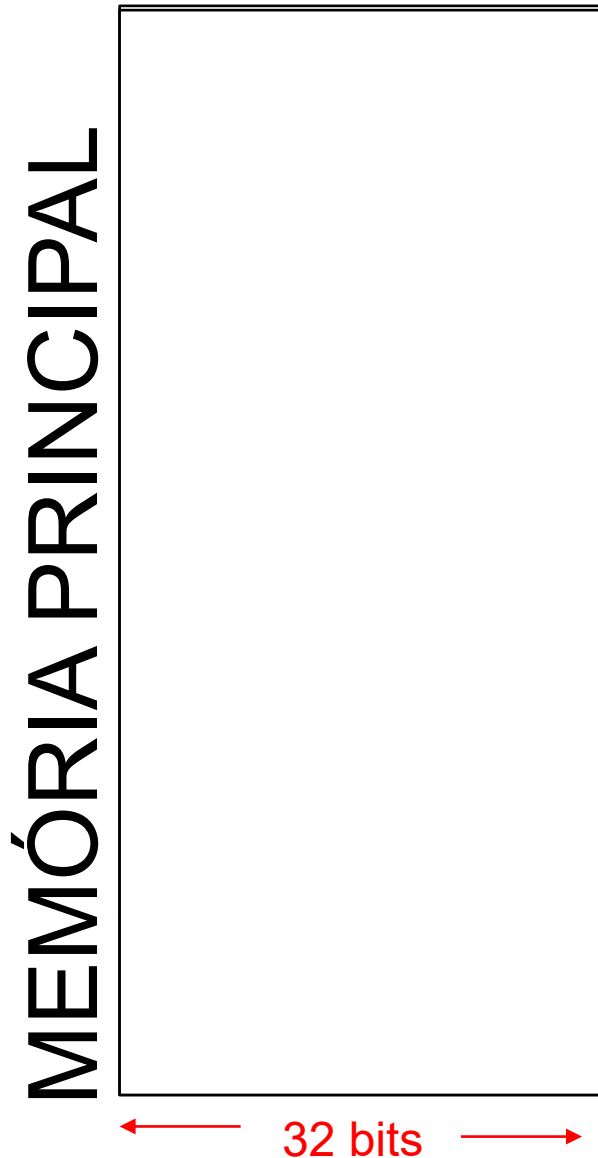


8 bits*

*como o MIPS, por exemplo

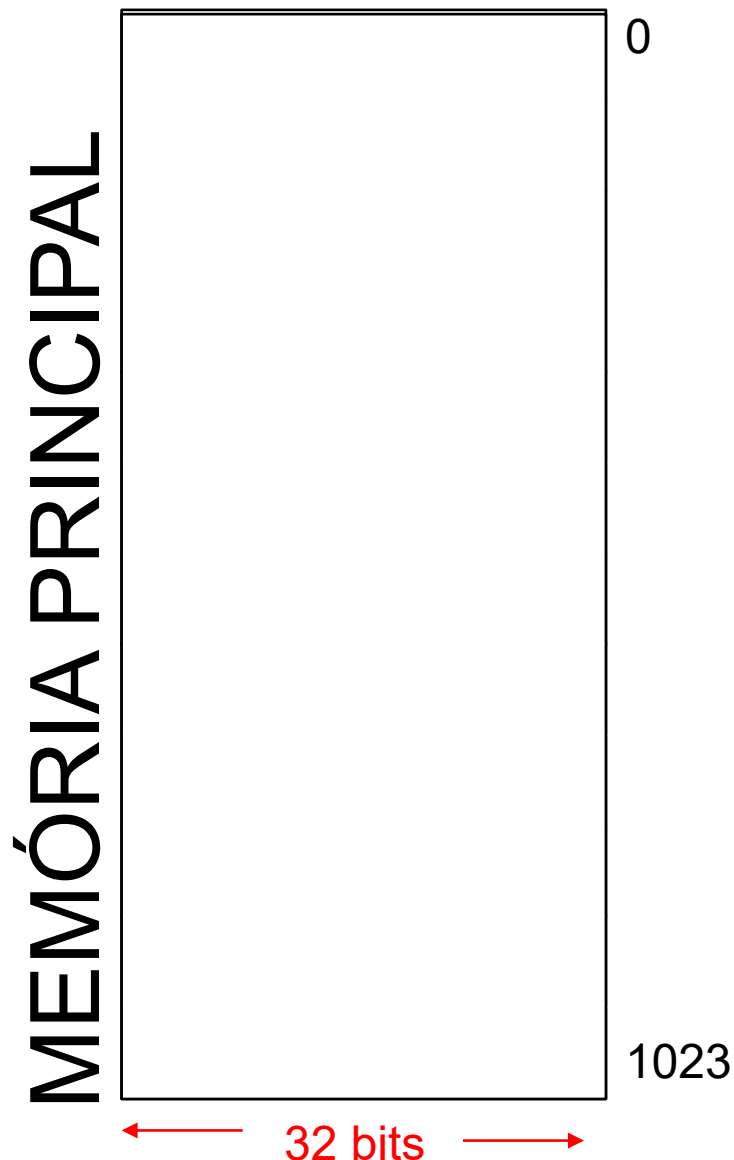
- Decisão de projeto:
 - Qual o tamanho da unidade mínima de transferência da memória?
 - ✧ Normalmente está associado ao tamanho palavra do processador
 - ✧ Isso é comumente chamado de “largura” da memória

Divisão da Memória



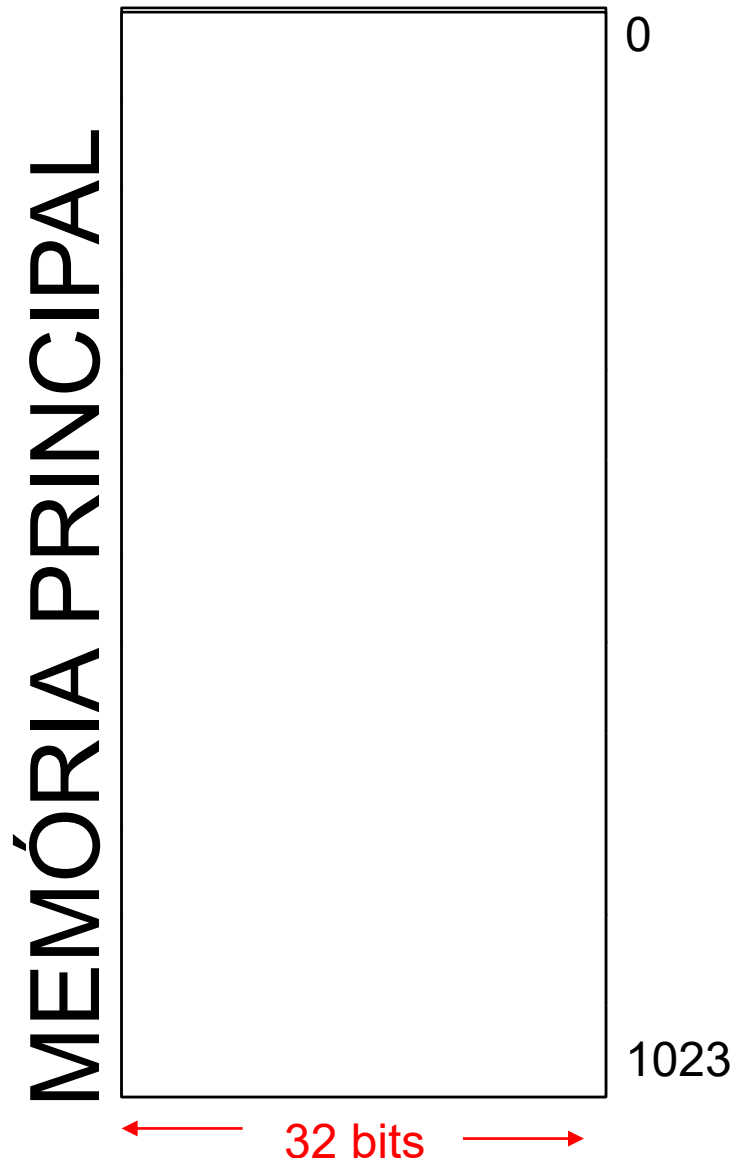
- Decisão de projeto:
 - Neste exemplo a largura é igual ao tamanho da palavra: 32 bits

Divisão da Memória



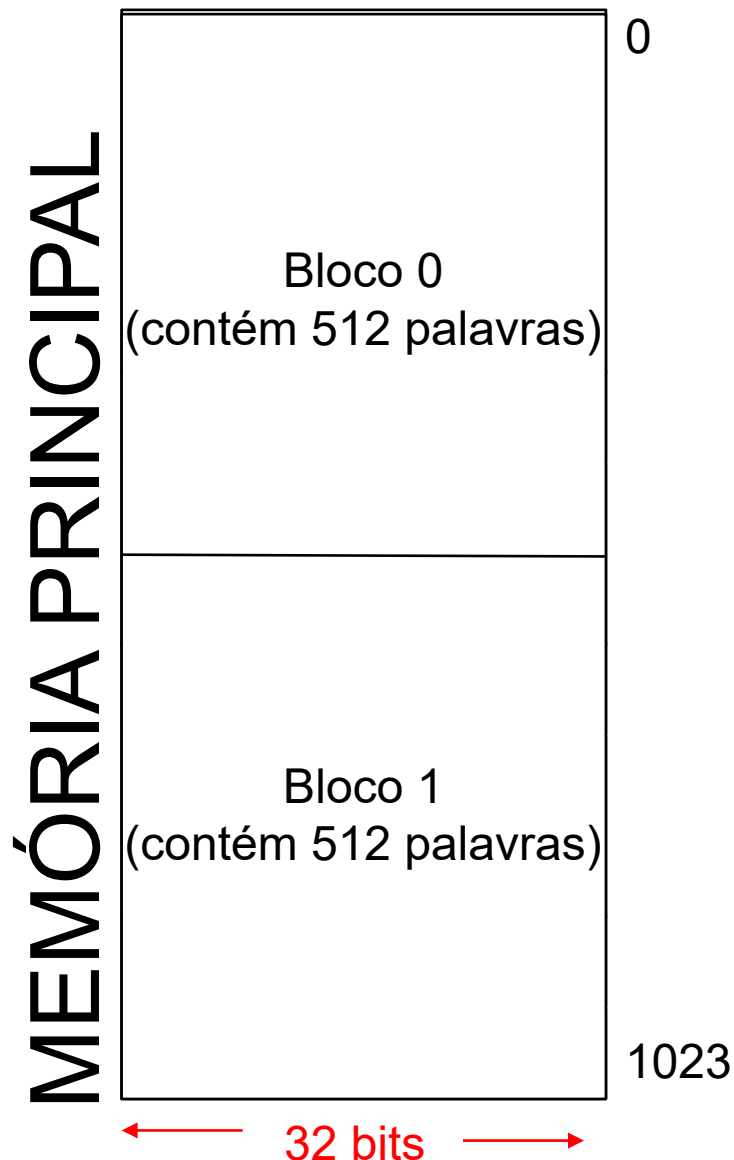
- Decisão de projeto:
 - Neste exemplo a largura é igual ao tamanho da palavra: 32 bits
- Quando o processador solicita um dado à cache, ele entrega um endereço.
- Cada endereço corresponde, neste caso a uma palavra
- Os endereços possíveis de serem referenciados chamamos de **espaço de endereçamento**. Neste caso, de 0 a 1023.

Divisão da Memória



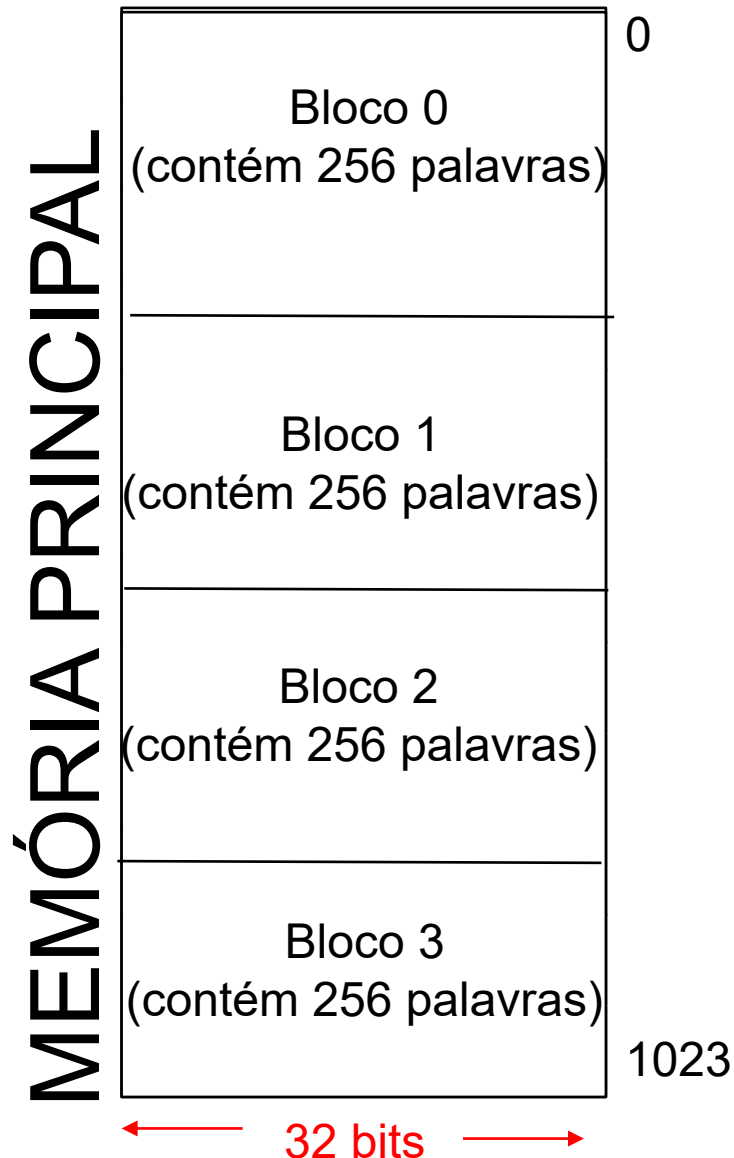
- A cache responde ao processador com uma única palavra conforme solicitado
- Mas quando solicita da memória, os dados vêm em blocos (ou linhas).
- Nova decisão de projeto:
 - Quantas palavras devem formar um bloco?

Divisão da Memória



- A cache responde ao processador com uma única palavra conforme solicitado
- Mas quando solicita da memória, os dados vêm em blocos (ou linhas).
- Nova decisão de projeto:
 - Quantas palavras devem formar um bloco?
 - 512?

Divisão da Memória



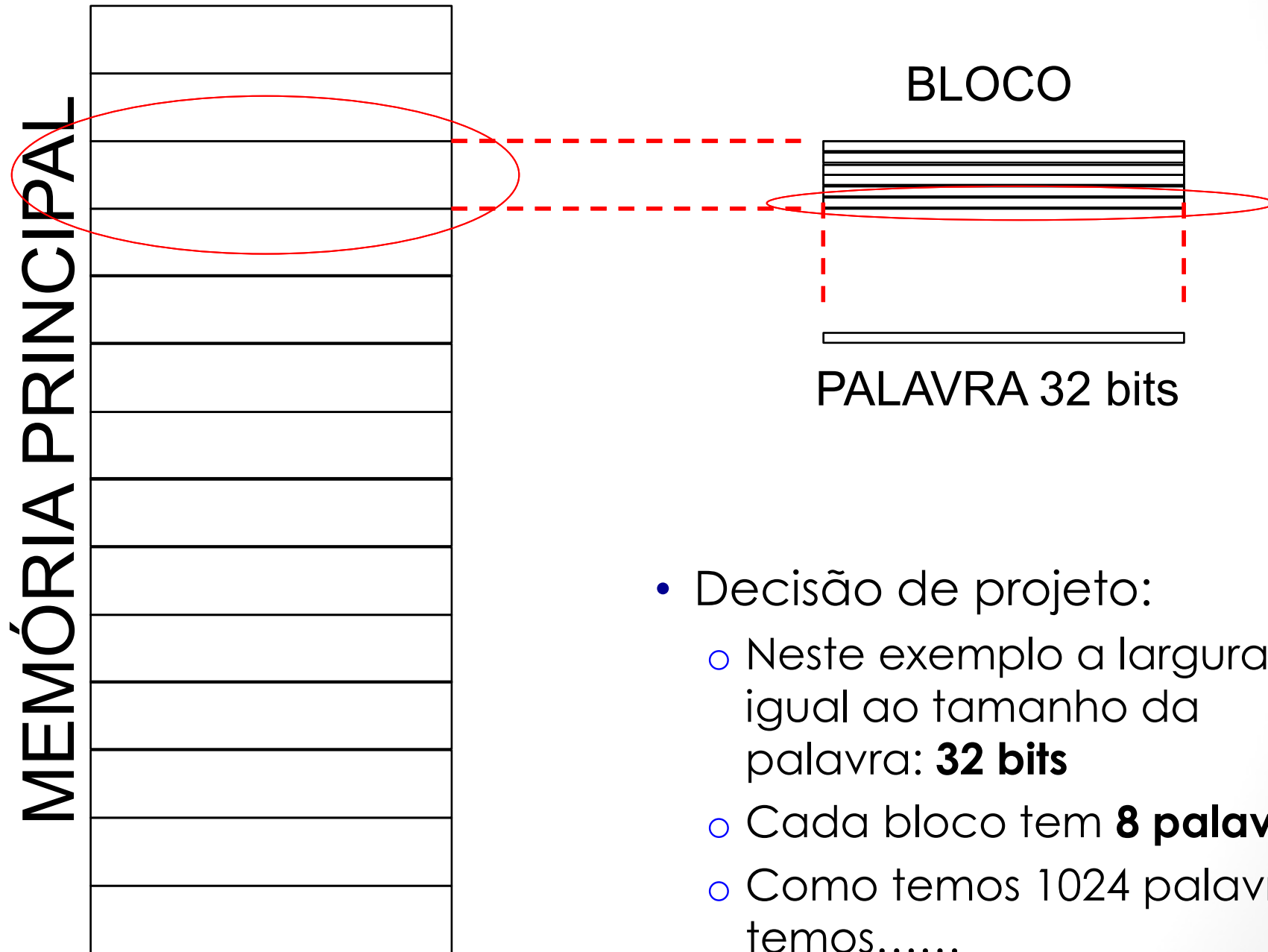
- A cache responde ao processador com uma única palavra conforme solicitado
- Mas quando solicita da memória, os dados vêm em blocos (ou linhas).
- Nova decisão de projeto:
 - Quantas palavras devem formar um bloco?
 - 512?
 - 256?

Divisão da Memória



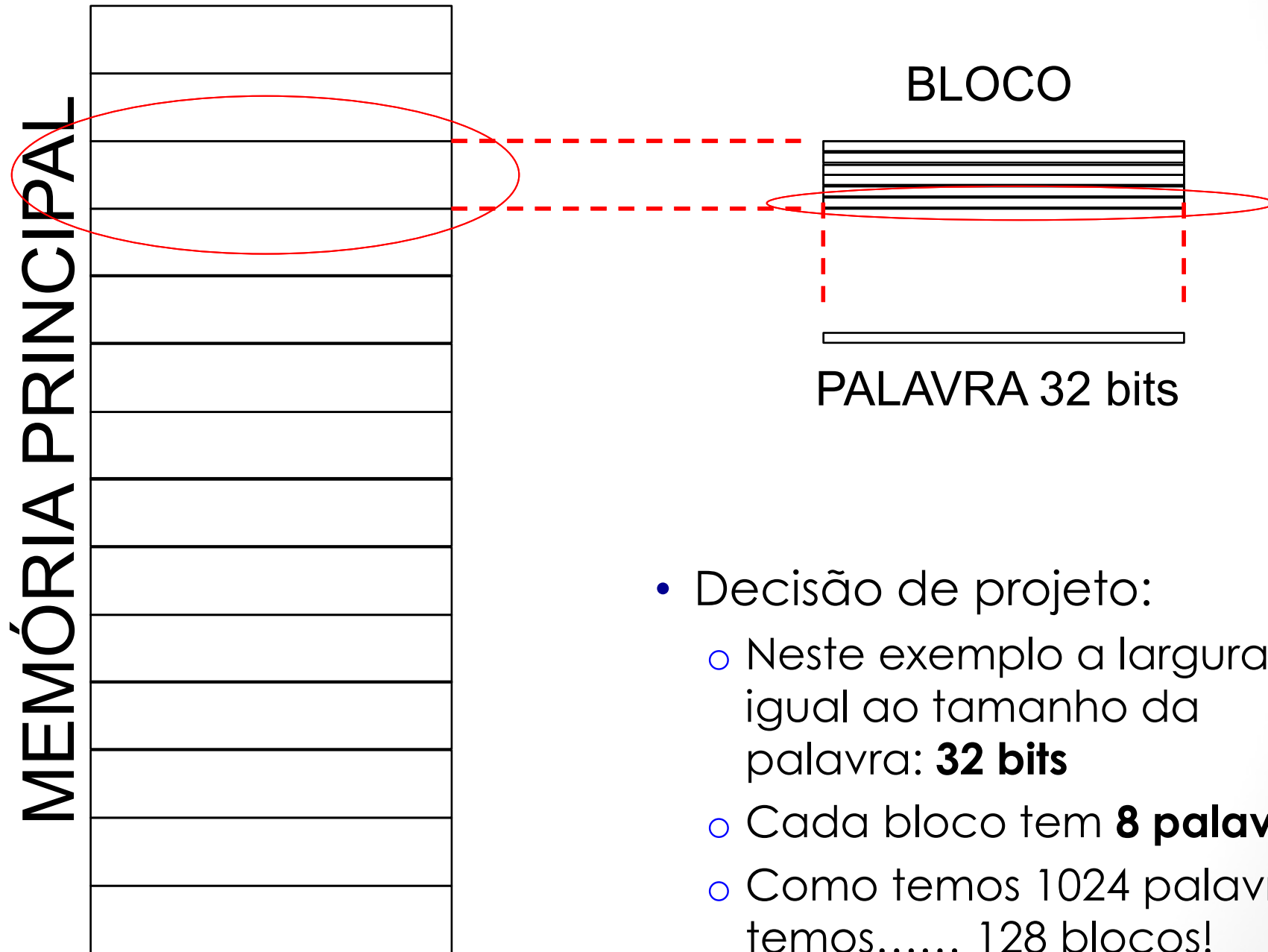
- A cache responde ao processador com uma única palavra conforme solicitado
- Mas quando solicita da memória, os dados vêm em blocos (ou linhas).
- Nova decisão de projeto:
 - Quantas palavras devem formar um bloco?
 - 512?
 - 256?
 - 128?

Divisão da Memória



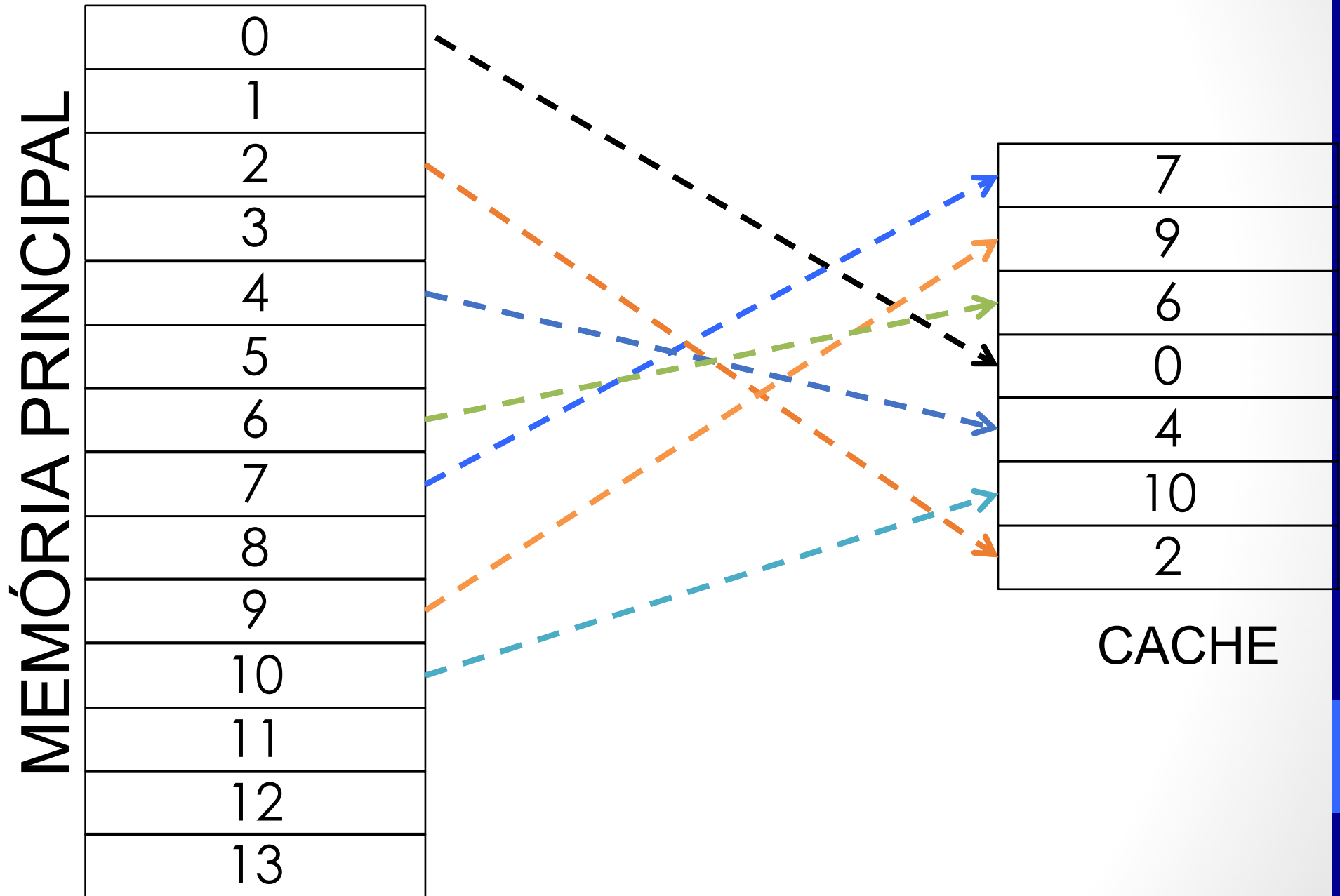
- Decisão de projeto:
 - Neste exemplo a largura é igual ao tamanho da palavra: **32 bits**
 - Cada bloco tem **8 palavras**
 - Como temos 1024 palavras, temos.....

Divisão da Memória



- Decisão de projeto:
 - Neste exemplo a largura é igual ao tamanho da palavra: **32 bits**
 - Cada bloco tem **8 palavras**
 - Como temos 1024 palavras, temos..... 128 blocos!

Divisão da Memória



Projeto da Cache

- É importante lembrar que a memória cache é **menor** que a memória principal e, portanto, comporta **menos blocos** que ela.
 - Entretanto, o **tamanho dos blocos** nas duas memórias **tem que** ser o mesmo!
- Ao longo da execução do processador, serão feitas referencias a endereços (para leitura ou escrita)
- Os blocos vão sendo copiados para a cache a medida que vão sendo solicitados pelo processador
 - Princípio da Localidade
 - ✧ Espacial: são trazidos blocos, ou seja, dados vizinho àquele solicitado também entram na cache
 - ✧ Temporal: dados mais recentemente solicitados ficam mais perto do processador, ou seja, na cache.

Exemplo

- Quando o processador solicita um dado da memória ele envia um endereço.
 - No nosso exemplo o espaço de endereçamento da memória é de 1024 palavras (endereços entre 0 e 1023).
- A cache recebe este endereço e a primeira coisa a ser feita é saber a que bloco **B** pertence ao endereço **E**.
 - Para isso precisamos de um outro número: o número de palavras por bloco (**N**).
- Logo, temos:
 - **$B = E \text{ div } N$**
 - ✧ Div é uma operação de **divisão inteira**

Exemplo



MEMÓRIA CACHE
(Com blocos de 8 palavras)

7
9
6
10
4

MEMÓRIA
PRINCIPAL

0
1
2
3
4
5
6
7
8
9
10
11
12
13

Exemplo



Pede Leitura do
Endereço 47



MEMÓRIA CACHE
(Com blocos de 8 palavras)

7
9
6
10
4

MEMÓRIA
PRINCIPAL

0
1
2
3
4
5
6
7
8
9
10
11
12
13

- Passo 1: o processador solicita o endereço **47**

Exemplo



MEMÓRIA CACHE
(Com blocos de 8 palavras)

7
9
6
10
4

MEMÓRIA PRINCIPAL

0
1
2
3
4
5
6
7
8
9
10
11
12
13

- Passo 2: A cache calcula qual o bloco do endereço **47**
 - $B = 47 \text{ div } 8 = 5$

Exemplo



MEMÓRIA CACHE
(Com blocos de 8 palavras)

7
9
6
10
4

MEMÓRIA
PRINCIPAL

0
1
2
3
4
5
6
7
8
9
10
11
12
13

- Passo 3: A cache agora verifica se tem o bloco 5

Exemplo



MEMÓRIA CACHE
(Com blocos de 8 palavras)

7
9
6
10
4

MEMÓRIA
PRINCIPAL

0
1
2
3
4
5
6
7
8
9
10
11
12
13

- Passo 3: A cache agora verifica se tem o bloco 5
- **CACHE MISS!**



Exemplo



MEMÓRIA CACHE
(Com blocos de 8 palavras)

7
9
6
10
4

Pede uma
cópia do
bloco 5



MEMÓRIA
PRINCIPAL

0
1
2
3
4
5
6
7
8
9
10
11
12
13

- Passo 4: A cache deve solicitar o bloco 5 à memória principal.

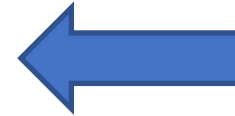
Exemplo



MEMÓRIA CACHE
(Com blocos de 8 palavras)

7
9
6
10
4
5

Recebe e
armazena



MEMÓRIA
PRINCIPAL


0
1
2
3
4
5
6
7
8
9
10
11
12
13

- Passo 5: A cache recebe a cópia e armazena.

Exemplo



Conteúdo do
endereço 47



MEMÓRIA CACHE
(Com blocos de 8 palavras)

7
9
6
10
4
5

MEMÓRIA
PRINCIPAL

0
1
2
3
4
5
6
7
8
9
10
11
12
13

- Passo 6: A cache entrega ao processador o conteúdo do endereço 47 que se encontra dentro do bloco 5.

Questões Básicas na Hierarquia de Memória

I. **Posicionamento do bloco.**

Onde o bloco deve ser colocado na memória de nível mais alto?

II. **Substituição de bloco.**

Quais blocos serão trocados em um miss?

III. **Estratégia de gravação.**

O que acontece em uma escrita?

Questões Básicas na Hierarquia de Memória

I. **Posicionamento do bloco.**

Onde o bloco deve ser colocado na memória de nível mais alto?

II. **Substituição de bloco.**

Quais blocos serão trocados em um miss?

III. **Estratégia de gravação.**

O que acontece em uma escrita?

I – Posicionamento do bloco

- ◆ **Mapeamento direto (*directed-mapped*)**

- ◆ Cada bloco pode ser colocado em uma única posição na cache

- ◆ Vantagem: hardware mais simples

- ◆ Para determinar se houve *cache hit* basta consultar UMA linha da cache.

- ◆ Desvantagem: potencial desperdício de espaço

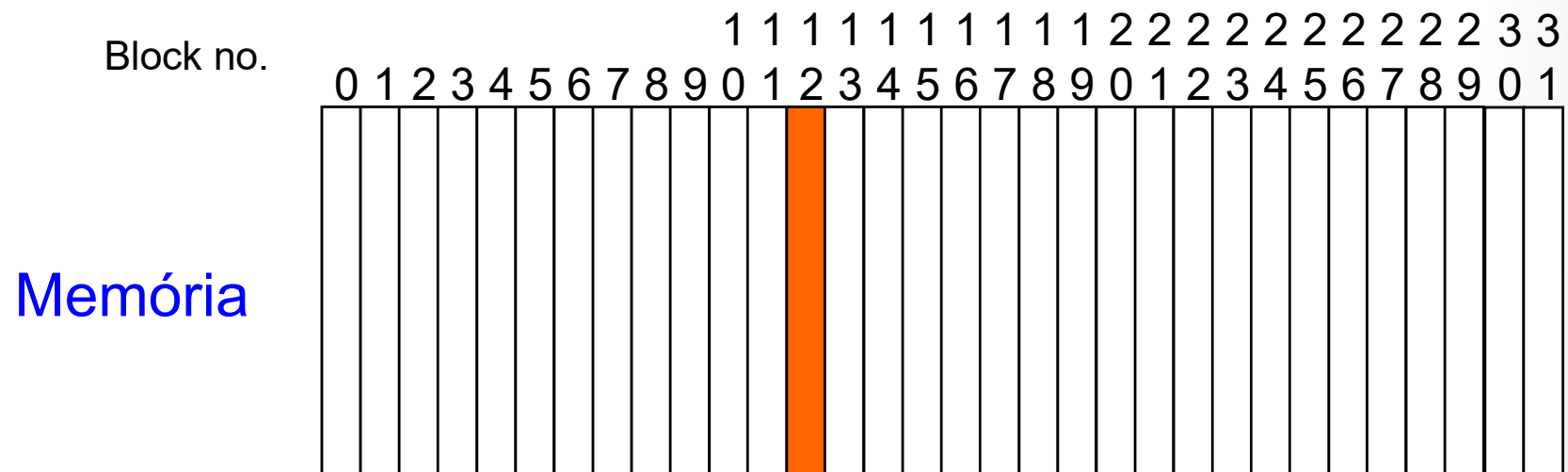
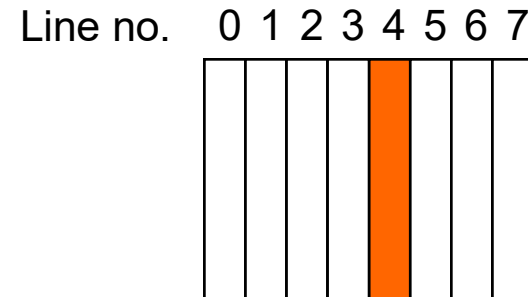
- ◆ Pode existir o caso em que exista espaço livre na cache, porém são espaços que não podem ser ocupados pelo bloco vindo da memória (regra de mapeamento)

I – Posicionamento do bloco

Mapeamento direto

Regra de mapeamento:

do bloco % # de linhas da cache
(12 % 8) = 4



I – Posicionamento do bloco

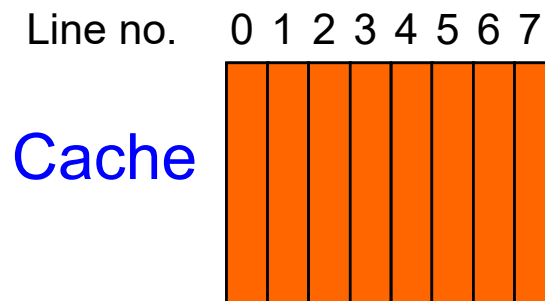
- ◆ **Totalmente associativa (*fully associative*)**
 - ◆ Cada bloco pode ser colocado em qualquer posição na cache
 - ◆ Vantagem: Nunca há desperdício de espaço
 - ◆ Enquanto houver espaço disponível na cache, o bloco vindo da memória pode ser alocado.
 - ◆ Desvantagem: hardware mais complexo
 - ◆ Para saber se houve *cache hit* é preciso comparar TODAS as linhas da cache.
 - ◆ Utilizada somente em memórias muito pequenas.

I – Posicionamento do bloco

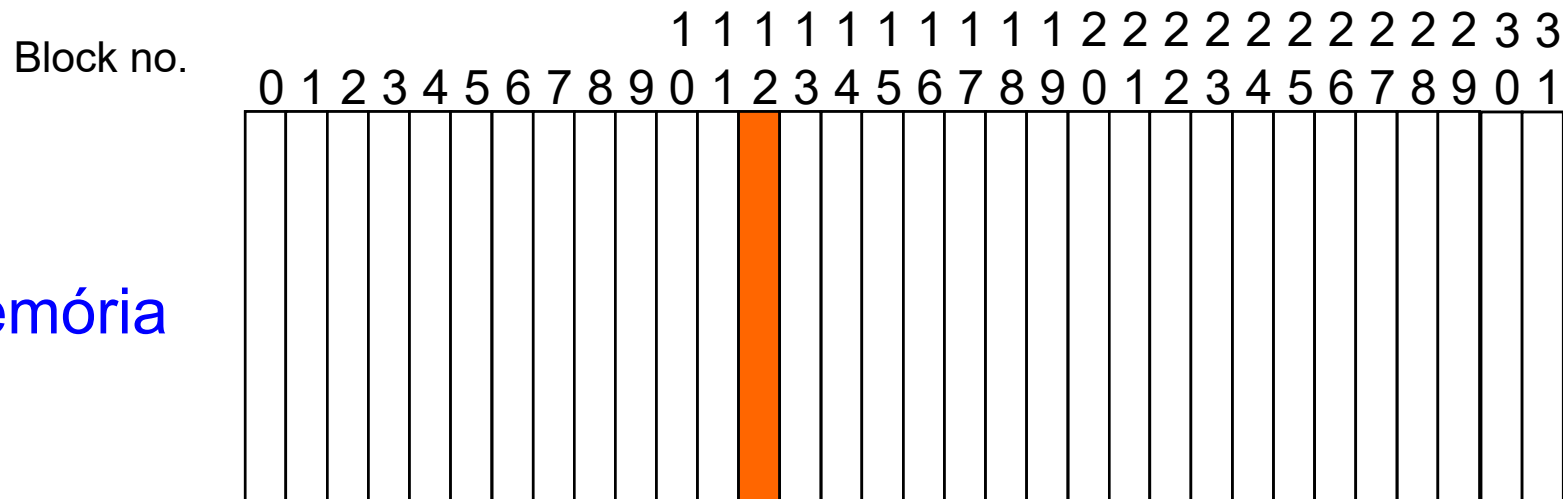
Totalmente
Associativa

Mapeamento Totalmente Associativo

Regra de mapeamento:
A próxima linha disponível



Memória



I – Posicionamento do bloco

- ◆ **Associativa por conjunto (*n*-way set-associative)**
 - ◆ Cada bloco pode ser colocado em um conjunto restrito de posições na cache.
 - ◆ Solução de compromisso entre as duas anteriores.
 - ◆ Se o conjunto for do tamanho da própria cache
 - ◆ Completamente Associativo
 - ◆ Se o conjunto for de tamanho 1
 - ◆ Direto
 - ◆ “way” (ou **vias**) identifica quantos **linhas** por conjunto existem.

I – Posicionamento do bloco

Associativa por Conjunto:

Regra de mapeamento:

do bloco % # de conjuntos
(12 % 4) = Set 0

Set no.	0	1	2	3				
Line no.	0	1	2	3	4	5	6	7

Diagram illustrating a memory layout with blocks numbered 0 to 33. The blocks are arranged in a single row. Block 12 is highlighted in orange, indicating it is the current block being processed. The label 'mória' (memory) is positioned to the left of the blocks.

I – Posicionamento do bloco

- ▶ Em processadores Intel core i7:
 - Cache de dados 32kB L1: associativas de conjunto 8 vias (8-way).
 - Cache de dados 256kB L2: associativas de conjunto 8 vias.
 - Cache 8MB L3 off-core: associativas de conjunto 16 vias.
- ▶ O número de vias aumenta o número de locais na cache onde o conteúdo de cada endereço de memória pode ser colocado.
- ▶ Assim, faz mais sentido ter mais locais para instruções, onde saltos e desvios abrem maior intervalo de endereçamento.

II – Substituição do bloco

- ▶ Uma falha de cache (ou miss) ocorre quando o processador requisita dados que **não estão presentes** na memória cache.
- ▶ Nesse caso, o processador **congela seu funcionamento** (*stall*), até que a memória cache busque no nível inferior o dado requisitado.
- ▶ Existem algumas alternativas à parada do pipeline. Uma delas é a mudança de contexto para a execução de uma thread diferente como vimos em aulas passadas (*multithreading*)

II – Substituição do bloco

- ▶ **Mapeamento direto:** somente o bloco não encontrado é substituído, simplificando o hardware.
- ▶ Demais mapeamentos:
 - ▶ **Aleatória:**
o bloco a substituir é escolhido aleatoriamente.
 - ▶ **Menos recentemente usado (LRU):**
substitui-se o bloco que não é usado há mais tempo.
 - ▶ **Menos frequentemente usado (LFU):**
substitui-se o bloco que foi menos utilizado.
 - ▶ **Primeiro a entrar, primeiro a sair (FIFO):**
substitui-se o bloco mais antigo (ainda que tenha sido recentemente usado).

II – Substituição do bloco

- ▶ Como são Implementados?

- ▶ Aleatória:

- Através de um gerador de números aleatórios (depende da máquina) o bloco a ser removido é escolhido.

- ▶ Baixa custo de implementação

- ▶ Menos recentemente usado (LRU):

- Uma referencia temporal é associada a cada bloco quando este chega na cache ou é acessado, esta referencia é atualizada (incrementada). O bloco a ser retirado é aquele com menor referencia temporal

- ▶ Alto custo de implementação: um registrador de referencia temporal para cada bloco.
 - ▶ De tempos em tempos precisa ser redimensionado

II – Substituição do bloco

- ▶ Como são Implementados?

- ▶ Menos frequentemente usado (LFU):

Faz uso de um contador de uso (registrador). Sempre que o bloco é acessado (i.e. lido ou escrito) o contador é incrementado. A linha a ser retirada é a de menor contador.

- ▶ Alto custo de implementação: um contador por linha
 - ▶ Também precisa de redimensionamento devido à saturação do contador

- ▶ Primeiro a entrar, primeiro a sair (FIFO):

Utiliza-se uma referencia de ordem dentro da cache. Cada linha trazida para um bloco da cache recebe uma referencia de quando chegou na cache. A linha a ser retirada é a de menor valor. Ao ser retirada as referencias de todas as outras linhas mudam.

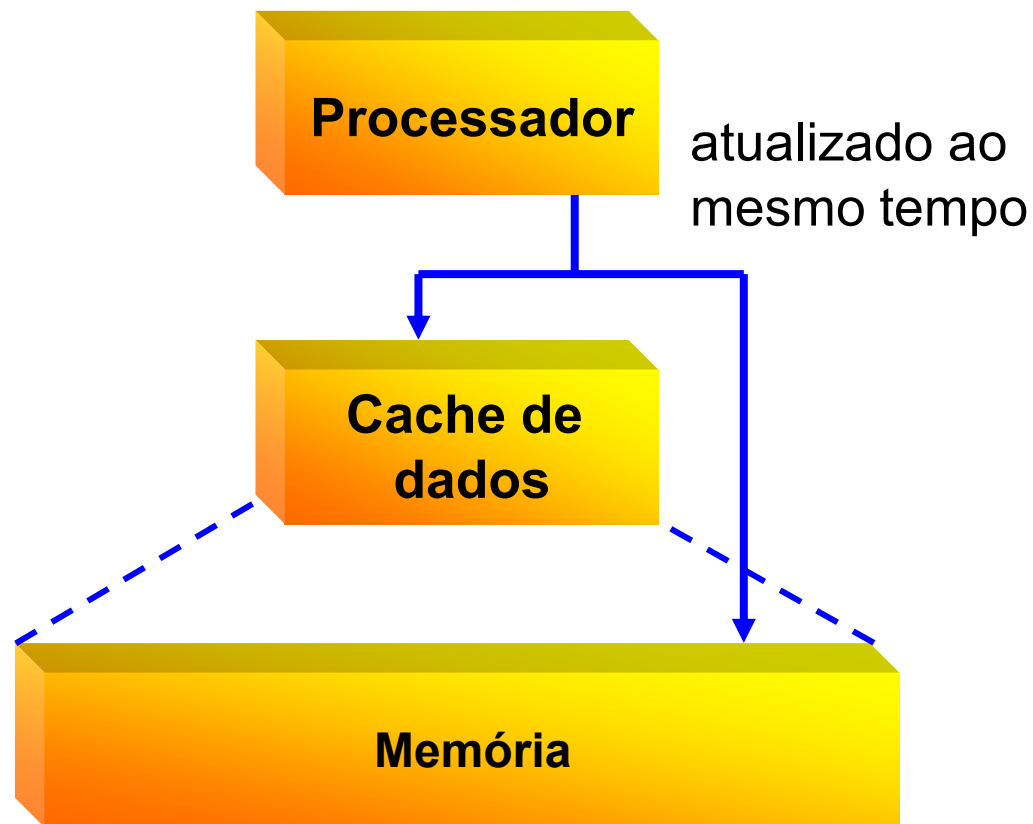
- ▶ Custo moderado de implementação e manutenção: um registrador para cada linha, porém tem resolução (numero de bits), pequena.

III – Estratégia de gravação

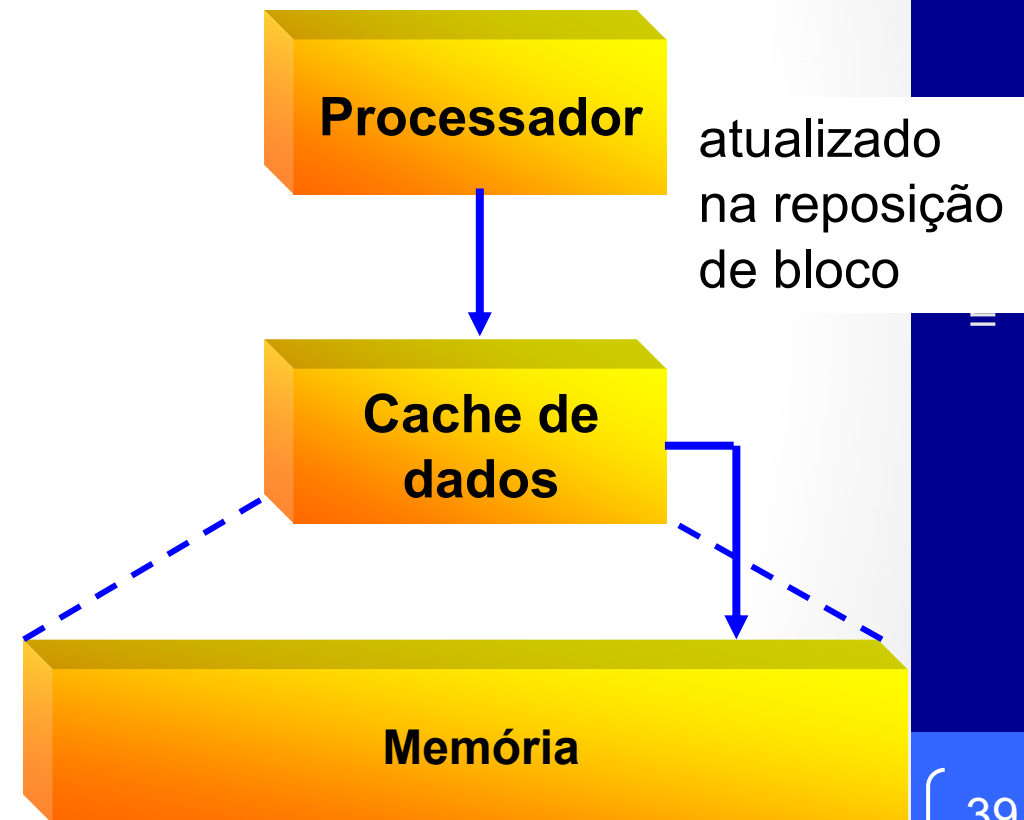
- ▶ Dois modos de escrita de dados:
 - **Write-through:** a informação é escrita tanto para o bloco da cache quanto para a memória de nível mais baixo.
 - **Write-back:** a informação é escrita somente para o bloco da cache. Este bloco só escrito na memória de nível inferior quando for trocado.

III – Estratégia de gravação

Write-through



Write-back



III – Estratégia de gravação

► Write-back

- **Servidores**, por consumir menos largura de banda de memória
- **Sistemas embarcados**, por poupar energia ao utilizar menos recursos de hardware para escrita de dados

► Write-Through

- **Dispositivos com duplo processamento**, onde ambos compartilham uma memória comum.



Bibliografia

- ▶ PATTERSON, D.A. & HENNESSY, J. L.
Organização e Projeto de Computadores -
A Interface Hardware/Software. 3ª ed.
Campus, 2005. **CAPÍTULO 7**
- ▶ STALLINGS, William. Arquitetura e
organização de computadores. 8. ed. São
Paulo: Pearson, 2010. **Capítulo 4**

Próxima aula

- Memória Virtual!