



UNIVERSIDADE
FEDERAL DE
SERGIPE



DEPARTAMENTO
DE COMPUTAÇÃO

Gerenciamento de cache

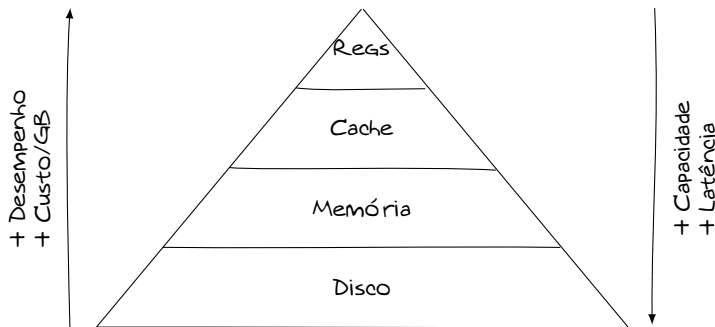
Arquitetura de Computadores

Bruno Prado

Departamento de Computação / UFS

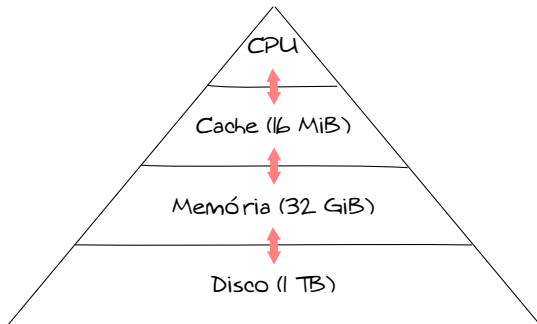
Introdução

- ▶ Por que usar a hierarquia de memória?
 - ▶ Compromisso entre capacidade e desempenho
 - ▶ Redução do custo de armazenamento dos dados



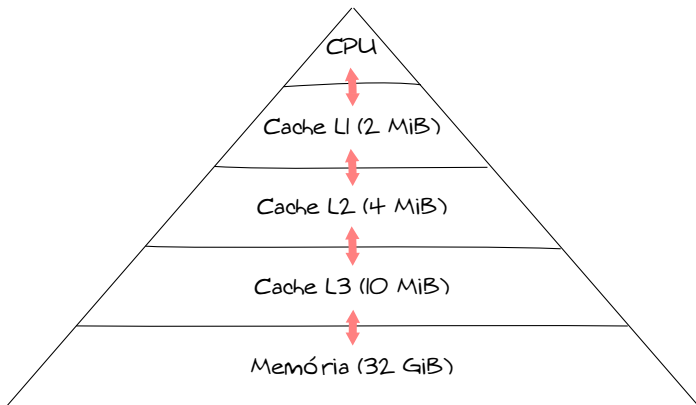
Introdução

- ▶ Por que usar a hierarquia de memória?
 - ▶ A tecnologia da memória cache possui o melhor desempenho, funcionando como interface rápida para as memórias mais lentas



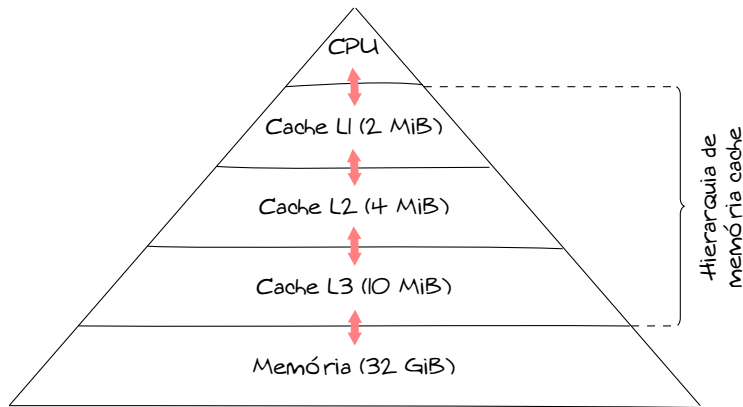
Introdução

- ▶ O que é uma memória cache?
 - ▶ Alto desempenho e baixa latência
 - ▶ Tamanho limitado por causa do custo elevado
 - ▶ Método de acesso associativo



Introdução

- ▶ O que é uma memória cache?
 - ▶ Alto desempenho e baixa latência
 - ▶ Tamanho limitado por causa do custo elevado
 - ▶ Método de acesso associativo



Introdução

- ▶ Por que a hierarquia de memória funciona?

Introdução

- ▶ Por que a hierarquia de memória funciona?
 - ▶ Os princípios de localidade espacial e temporal que definem que várias regiões da memória são repetidamente e sequencialmente acessadas em um determinado intervalo de tempo
 - ▶ Execução sequencial das instruções do software
 - ▶ Repetição através de controles iterativos

Introdução

- ▶ Localidade espacial
 - ▶ A execução do software é iterativa e sequencial
 - ▶ É feito o acesso recorrente para um conjunto pequeno e repetido de instruções e dados

```
1 // Biblioteca de E/S padrão
2 #include <stdio.h>
3 // Função principal
4 int main() {
5     // Variáveis inteiras
6     uint32_t a = 1, i;
7     // Controle iterativo
8     for(i = 0; i < 1024; i++) {
9         a = 2 * a;
10    }
11    // Retorno sem erros
12    return 0;
13 }
```


Introdução

- ▶ Localidade espacial
 - ▶ A execução do software é iterativa e sequencial
 - ▶ É feito o acesso recorrente para um conjunto pequeno e repetido de instruções e dados

```
1 // Biblioteca de E/S padrão
2 #include <stdio.h>
3 // Função principal
4 int main() {
5     // Variáveis inteiras
6     uint32_t a = 1, i;
7     // Controle iterativo
8     for(i = 0; i < 1024; i++) {
9         a = 2 * a;
10    }
11    // Retorno sem erros
12    return 0;
13 }
```

Instruções do controle iterativo

Introdução

- ▶ Localidade espacial
 - ▶ A execução do software é iterativa e sequencial
 - ▶ É feito o acesso recorrente para um conjunto pequeno e repetido de instruções e dados

```
1 // Biblioteca de E/S padrão
2 #include <stdio.h>
3 // Função principal
4 int main() {
5     // Variáveis inteiras
6     uint32_t a = 1, i;
7     // Controle iterativo
8     for(i = 0; i < 1024; i++) {
9         a = 2 * a;
10    }
11    // Retorno sem erros
12    return 0;
13 }
```

Dados acessados nas iterações

Introdução

- ▶ Localidade temporal
 - ▶ Quando uma determinada posição de memória é referenciada (instrução ou dado), provavelmente ela será acessada novamente pelo fluxo de execução

```
1 // Biblioteca de E/S padrão
2 #include <stdio.h>
3 // Função principal
4 int main() {
5     // Variáveis inteiras
6     uint32_t a = 1, i;
7     // Controle iterativo
8     for(i = 0; i < 1024; i++) {
9         a = 2 * a;
10    }
11    // Retorno sem erros
12    return 0;
13 }
```

Introdução

- ▶ Localidade temporal
 - ▶ Quando uma determinada posição de memória é referenciada (instrução ou dado), provavelmente ela será acessada novamente pelo fluxo de execução

```
1 // Biblioteca de E/S padrão
2 #include <stdio.h>
3 // Função principal
4 int main() {
5     // Variáveis inteiras
6     uint32_t a = 1, i;
7     // Controle iterativo
8     for(i = 0; i < 1024; i++) {
9         a = 2 * a;
10    }
11    // Retorno sem erros
12    return 0;
13 }
```

Inicialização de variáveis

Introdução

- ▶ Localidade temporal
 - ▶ Quando uma determinada posição de memória é referenciada (instrução ou dado), provavelmente ela será acessada novamente pelo fluxo de execução

```
1 // Biblioteca de E/S padrão
2 #include <stdio.h>
3 // Função principal
4 int main() {
5     // Variáveis inteiras
6     uint32_t a = 1, i;
7     // Controle iterativo
8     for(i = 0; i < 1024; i++) {
9         a = 2 * a;
10    }
11    // Retorno sem erros
12    return 0;
13 }
```

Repetição da operação por 1024 vezes

Introdução

- ▶ Localidade temporal
 - ▶ Quando uma determinada posição de memória é referenciada (instrução ou dado), provavelmente ela será acessada novamente pelo fluxo de execução

```
1 // Biblioteca de E/S padrão
2 #include <stdio.h>
3 // Função principal
4 int main() {
5     // Variáveis inteiras
6     uint32_t a = 1, i;
7     // Controle iterativo
8     for(i = 0; i < 1024; i++) {
9         a = 2 * a;
10    }
11    // Retorno sem erros
12    return 0;
13 }
```

Finalização da execução

Introdução

- ▶ Qual é o papel da memória cache na implementação da hierarquia de memória?

Introdução

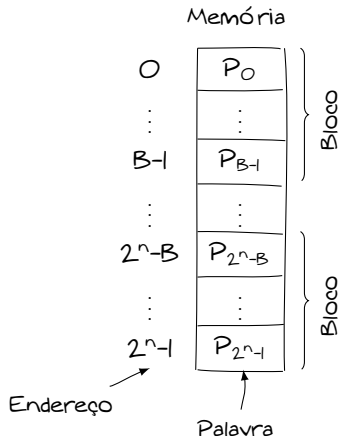
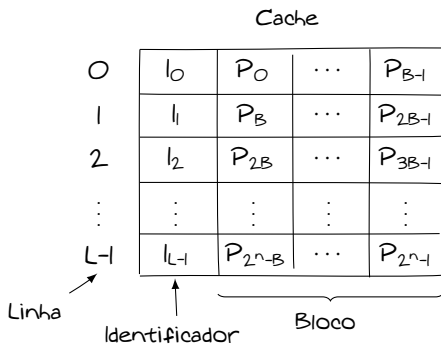
- ▶ Qual é o papel da memória cache na implementação da hierarquia de memória?
 - ▶ É a memória que possui menor latência, armazenando as instruções e os dados de acordo com os princípios de localidade, e realiza o acesso por associação do endereço da memória principal

Introdução

- ▶ Qual é o papel da memória cache na implementação da hierarquia de memória?
 - ▶ É a memória que possui menor latência, armazenando as instruções e os dados de acordo com os princípios de localidade, e realiza o acesso por associação do endereço da memória principal
 - ▶ O gerenciamento de cache vai explorar estes conceitos para aumentar disponibilidade dos dados na cache e o desempenho do sistema
 - ▶ Estrutura de armazenamento
 - ▶ Mapeamento dos endereços
 - ▶ Algoritmos de substituição
 - ▶ Política de escrita dos dados
 - ▶ Análise de desempenho

Gerenciamento de cache

- Estrutura de armazenamento
 - Linhas x Endereços
 - Blocos x Palavras

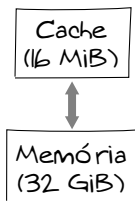


Gerenciamento de cache

- ▶ Estrutura de armazenamento
 - ▶ A cache possui L linhas com blocos de $B = 4$ palavras
 - ▶ A memória principal de 32 GiB possui $M = \frac{2^{33}}{B}$ blocos

Gerenciamento de cache

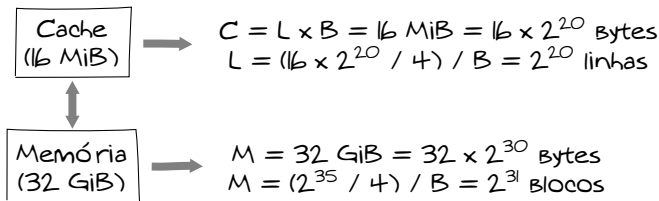
- ▶ Estrutura de armazenamento
 - ▶ A cache possui L linhas com blocos de $B = 4$ palavras
 - ▶ A memória principal de 32 GiB possui $M = \frac{2^{33}}{B}$ blocos



Gerenciamento de cache

► Estrutura de armazenamento

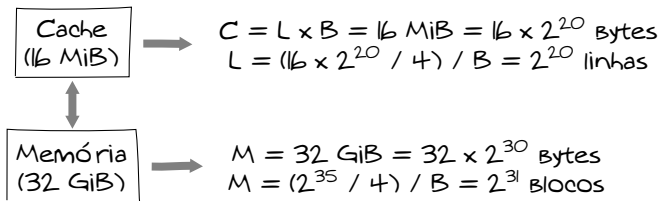
- A cache possui L linhas com blocos de $B = 4$ palavras
- A memória principal de 32 GiB possui $M = \frac{2^{33}}{B}$ blocos



Gerenciamento de cache

► Estrutura de armazenamento

- A cache possui L linhas com blocos de $B = 4$ palavras
- A memória principal de 32 GiB possui $M = \frac{2^{33}}{B}$ blocos



Esta cache de 16 MiB armazena $4 \times 2^{20} \div 2^{31} \approx 0,2\%$ do conteúdo total da memória principal de 32 GiB

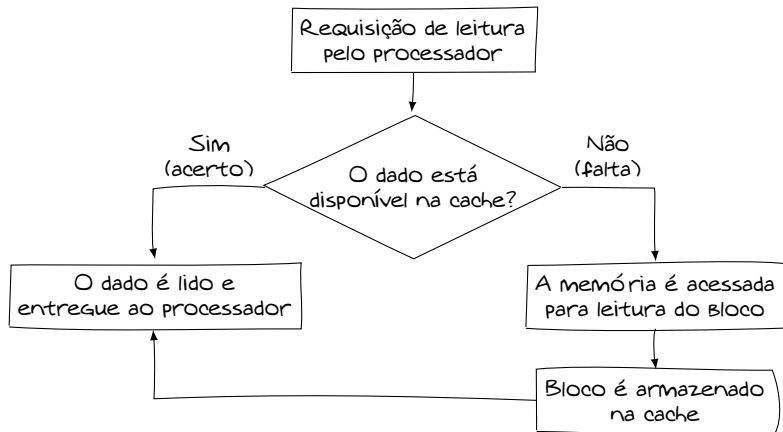
Gerenciamento de cache

► Estrutura de armazenamento

| Cache separada | Cache unificada |
|---|--|
| Duas caches distintas para instruções e dados | Uma única cache para instruções e dados |
| Permite o acesso paralelo das informações | A capacidade da cache é melhor aproveitada |
| Replicação de componentes e da lógica de controle | Somente um componente é utilizado e controlado |

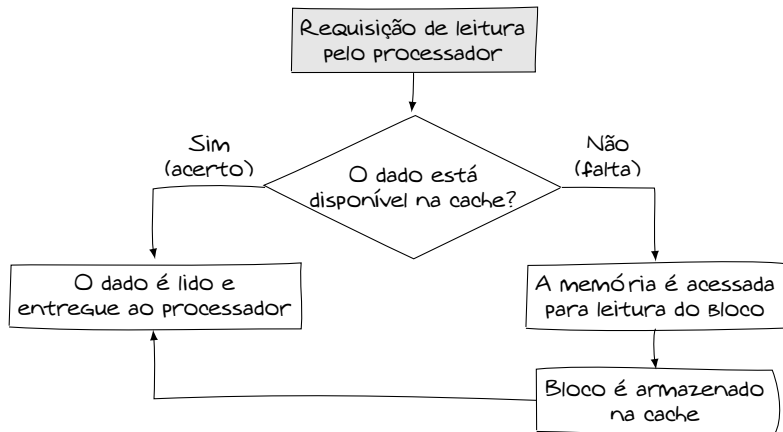
Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Cada bloco da memória possui um identificador
 - ▶ Este identificador é usado para verificar se o dado está armazenado em alguma linha da cache



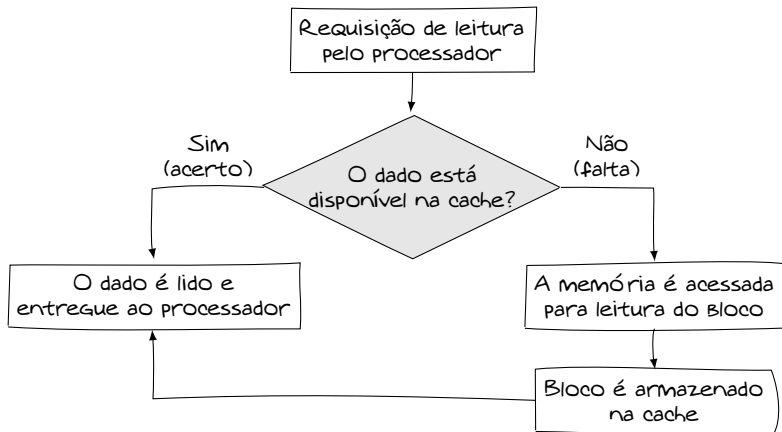
Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Cada bloco da memória possui um identificador
 - ▶ Este identificador é usado para verificar se o dado está armazenado em alguma linha da cache



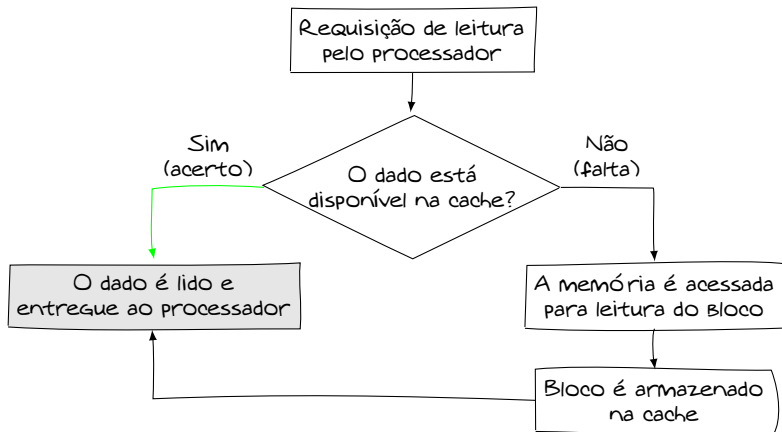
Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Cada bloco da memória possui um identificador
 - ▶ Este identificador é usado para verificar se o dado está armazenado em alguma linha da cache



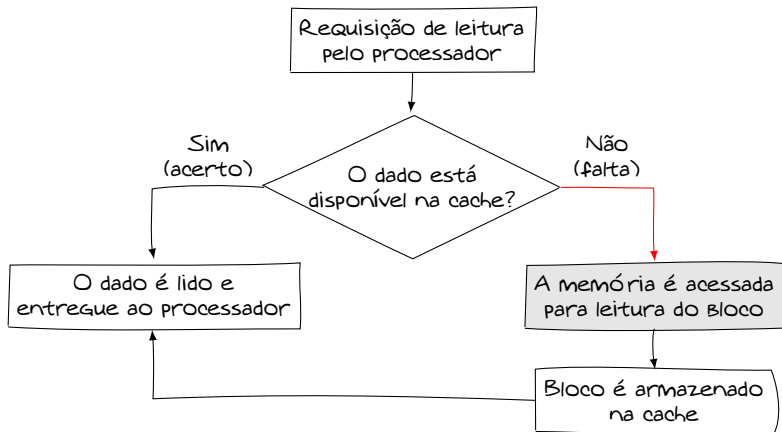
Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Cada bloco da memória possui um identificador
 - ▶ Este identificador é usado para verificar se o dado está armazenado em alguma linha da cache



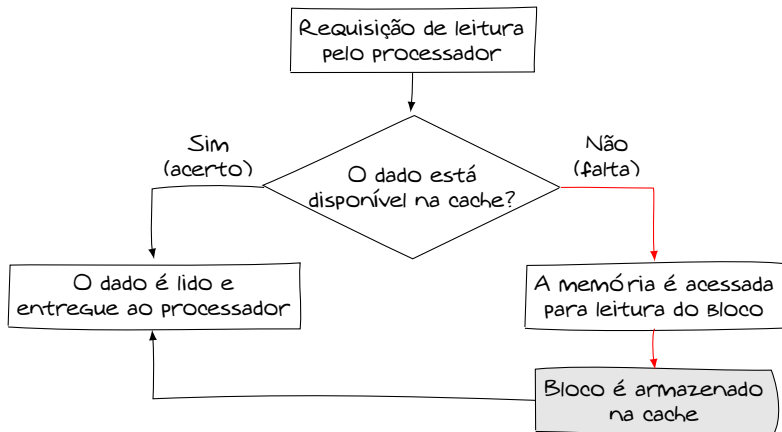
Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Cada bloco da memória possui um identificador
 - ▶ Este identificador é usado para verificar se o dado está armazenado em alguma linha da cache



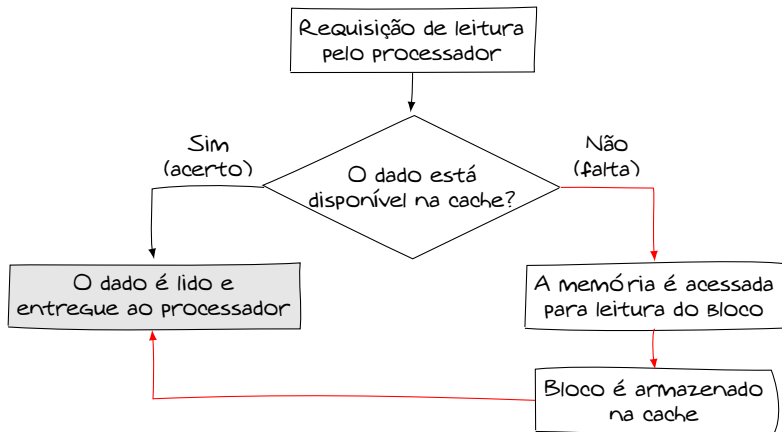
Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Cada bloco da memória possui um identificador
 - ▶ Este identificador é usado para verificar se o dado está armazenado em alguma linha da cache



Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Cada bloco da memória possui um identificador
 - ▶ Este identificador é usado para verificar se o dado está armazenado em alguma linha da cache



Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Se o dado solicitado para leitura pelo processador estiver disponível na cache, ocorre um acerto (*hit*)
 - ▶ Não existe retenção do processador
 - ▶ Os níveis inferiores da hierarquia não são acessados

Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Se o dado solicitado para leitura pelo processador estiver disponível na cache, ocorre um acerto (*hit*)
 - ▶ Não existe retenção do processador
 - ▶ Os níveis inferiores da hierarquia não são acessados
 - ▶ Caso o dado requisitado pelo processador não esteja armazenado na cache, ocorre uma falta (*miss*)
 - ▶ Ocorre a retenção de execução no processador
 - ▶ O bloco é buscado nos níveis inferiores da hierarquia

Gerenciamento de cache

- ▶ Fluxo de leitura de um dado da cache
 - ▶ Se o dado solicitado para leitura pelo processador estiver disponível na cache, ocorre um acerto (*hit*)
 - ▶ Não existe retenção do processador
 - ▶ Os níveis inferiores da hierarquia não são acessados
 - ▶ Caso o dado requisitado pelo processador não esteja armazenado na cache, ocorre uma falta (*miss*)
 - ▶ Ocorre a retenção de execução no processador
 - ▶ O bloco é buscado nos níveis inferiores da hierarquia

$\uparrow \%Acerto \longleftrightarrow \uparrow Desempenho$

Gerenciamento de cache

- ▶ Tipos de falta em cache
 - ▶ Compulsória
 - ▶ É decorrente do processo de inicialização da cache, enquanto os dados estão sendo armazenados em posições ainda sem nenhum dado

Gerenciamento de cache

- ▶ Tipos de falta em cache
 - ▶ Compulsória
 - ▶ É decorrente do processo de inicialização da cache, enquanto os dados estão sendo armazenados em posições ainda sem nenhum dado
 - ▶ Conflito
 - ▶ Dependendo das técnicas de endereçamento da cache, ocorrem colisões de endereços já utilizados que causam substituição dos dados

Gerenciamento de cache

- ▶ Tipos de falta em cache
 - ▶ Compulsória
 - ▶ É decorrente do processo de inicialização da cache, enquanto os dados estão sendo armazenados em posições ainda sem nenhum dado
 - ▶ Conflito
 - ▶ Dependendo das técnicas de endereçamento da cache, ocorrem colisões de endereços já utilizados que causam substituição dos dados
 - ▶ Capacidade
 - ▶ São faltas que ocorrem pela necessidade de substituição por falta de espaço disponível dos dados armazenados que serão referenciados futuramente

Gerenciamento de cache

- ▶ Mapeamento dos endereços
 - ▶ Define como os endereços de instruções e de dados solicitados pelo processador para a memória principal são mapeados nas linhas da cache
 - ▶ Direto
 - ▶ Totalmente associativo
 - ▶ Associativo por conjunto

Gerenciamento de cache

- ▶ Mapeamento direto

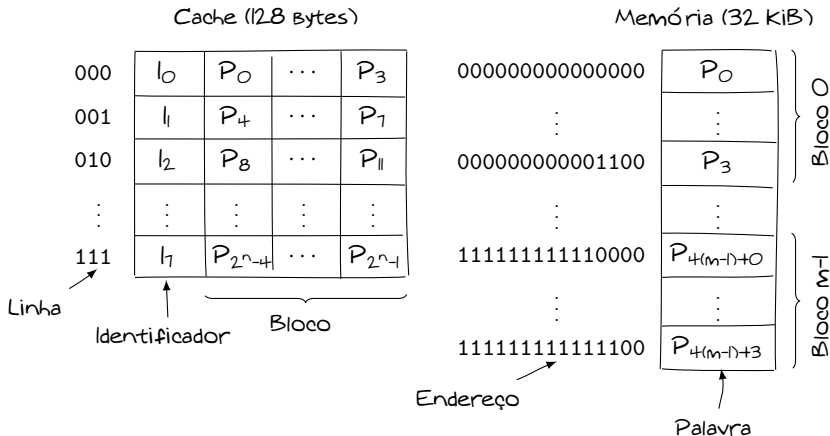
- ▶ É a técnica mais simples para associação de endereços na cache, consistindo na aplicação de uma função de módulo (*hash*) para determinar qual linha da cache deve ser indexada

$$i = j \bmod L$$

- ▶ A linha da cache é endereçada pela variável i
 - ▶ A variável j representa o endereço do bloco
 - ▶ O número total de linhas da cache é descrito por L

Gerenciamento de cache

- Mapeamento direto
 - A cache possui 8 linhas de dados
 - Cada linha armazena um bloco com 4 palavras

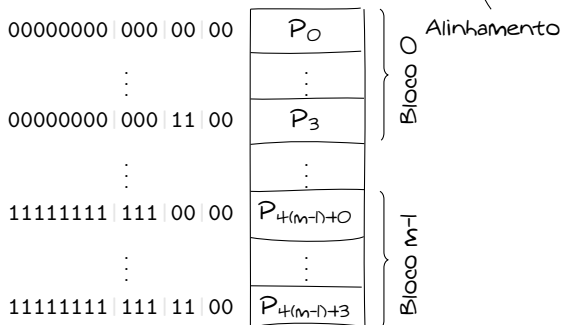


Gerenciamento de cache

► Mapeamento direto

- O identificador (ID) possui $N - NL - NP - 2$ bits, onde $NL = \log_2 L = 3$ e $NP = \log_2 B = 2$ são o número de bits para indexar as linhas e as palavras, respectivamente

| Identificador (ID) | Linha | Palavra | - |
|--------------------|-------|---------|---|
| $N - NL - NP - 2$ | NL | NP | 2 |



Memória (32 KiB)

Gerenciamento de cache

- ▶ Mapeamento direto
 - ▶ O endereço é usado para indexar linhas e palavras
 - ▶ O bit de validade indica se o dado está disponível

Cache de 128 Bytes com 8 linhas
(estado inicial)

| | | | | | | |
|-----|---|--|--|--|--|--|
| 000 | N | | | | | |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

↑
Linha

↑ Validade ↑ Identificador ↑ P_0 ↑ P_1 ↑ P_2 ↑ P_3

Gerenciamento de cache

► Mapeamento direto

- Solicitação de endereço 00000000|000|00|00
- Identificador = 00000000, Linha = 000 e Palavra = 00

O dado solicitado não está disponível
(falta compulsória)

| | | | | | | |
|-----|---|--|--|--|--|--|
| 000 | N | | | | | |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Linha

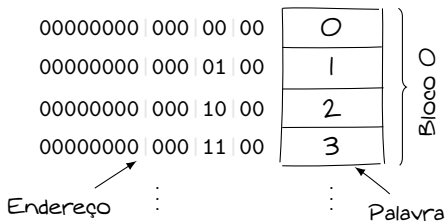
Validade Identificador P₀ P₁ P₂ P₃

Gerenciamento de cache

► Mapeamento direto

- Solicitação de endereço 00000000|000|00|00
- Identificador = 00000000, Linha = 000 e Palavra = 00

É feita a leitura do
bloco 0 da memória



Gerenciamento de cache

► Mapeamento direto

- Solicitação de endereço 00000000|000|00|00
- Identificador = 00000000, Linha = 000 e Palavra = 00

A palavra é obtida pelo processador
e o Bloco é armazenado na memória cache

| | | | | | | |
|-----|---|----------|---|---|---|---|
| 000 | S | 00000000 | 0 | 1 | 2 | 3 |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Linha

Validade

Identificador

P₀

P₁

P₂

P₃

Gerenciamento de cache

► Mapeamento direto

- Solicitação de endereço 00000000|000|11|00
- Identificador = 00000000, Linha = 000 e Palavra = 11

A palavra está disponível
(acerto na memória cache)

| | | | | | | |
|-----|---|----------|---|---|---|---|
| 000 | S | 00000000 | 0 | 1 | 2 | 3 |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Linha

Validade

Identificador

P₀

P₁

P₂

P₃

Gerenciamento de cache

► Mapeamento direto

- Solicitação de endereço 00000001|000|01|00
- Identificador = 00000001, Linha = 000 e Palavra = 01

Apesar da linha da cache ser válida,
o identificador não corresponde ao endereço
(falta por conflito)

| | | | | | | |
|-----|---|----------|---|---|---|---|
| 000 | S | 00000000 | 0 | 1 | 2 | 3 |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Linha

Validade

Identificador

P₀

P₁

P₂

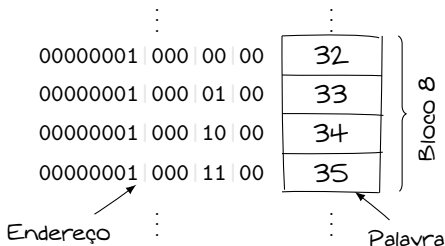
P₃

Gerenciamento de cache

► Mapeamento direto

- Solicitação de endereço 00000001|000|01|00
- Identificador = 00000001, Linha = 000 e Palavra = 01

É feita a leitura do
Bloco 8 da memória



Gerenciamento de cache

► Mapeamento direto

- Solicitação de endereço 00000001|000|01|00
- Identificador = 00000001, Linha = 000 e Palavra = 01

A palavra é obtida pelo processador
e o bloco é armazenado na memória cache

| | | | | | | |
|-----|---|----------|----|----|----|----|
| 000 | S | 00000001 | 32 | 33 | 34 | 35 |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Linha

Validade

Identificador

P₀

P₁

P₂

P₃

Gerenciamento de cache

- ▶ Mapeamento direto
 - ▶ Cenário ideal
 - ▶ Acesso sequencial e repetitivo da memória
 - ▶ Todas as linhas são utilizadas sem substituição

Após todos os blocos serem referenciados, não ocorrem mais faltas na memória cache

| | | | | | | |
|-----|---|----------|----|----|----|----|
| 000 | S | 00000000 | 0 | 1 | 2 | 3 |
| 001 | S | 00000000 | 4 | 5 | 6 | 7 |
| 010 | S | 00000000 | 8 | 9 | 10 | 11 |
| 011 | S | 00000000 | 12 | 13 | 14 | 15 |
| 100 | S | 00000000 | 16 | 17 | 18 | 19 |
| 101 | S | 00000000 | 20 | 21 | 22 | 23 |
| 110 | S | 00000000 | 24 | 25 | 26 | 27 |
| 111 | S | 00000000 | 28 | 29 | 30 | 31 |

Diagram illustrating a direct mapping cache scenario. The cache is represented as a table with 8 lines (000 to 111). Each line contains a status (S for valid), an identifier (00000000), and four data points (P0, P1, P2, P3). The data points are sequential values from 0 to 31. Arrows indicate the mapping from the cache line to the data points.

Labels below the table:

- Linha (points to the first column)
- Validade (points to the second column)
- Identificador (points to the third column)
- P₀ (points to the fourth column)
- P₁ (points to the fifth column)
- P₂ (points to the sixth column)
- P₃ (points to the seventh column)

Gerenciamento de cache

- ▶ Mapeamento direto
 - ✓ Implementação de baixo custo e simples
 - ✓ A indexação depende somente da seleção de bits do endereço de memória solicitado

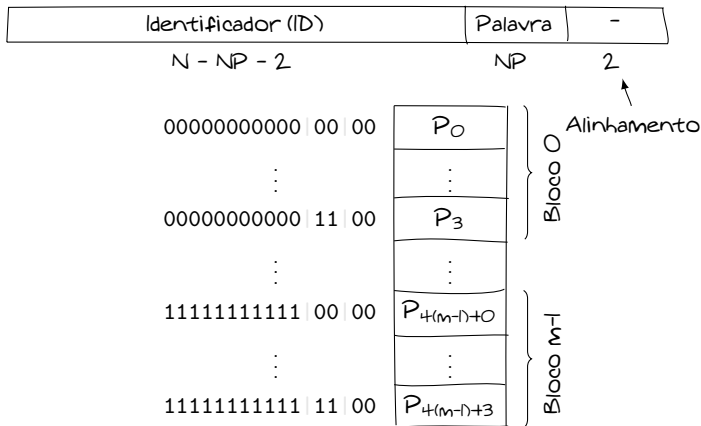
Gerenciamento de cache

► Mapeamento direto

- ✓ Implementação de baixo custo e simples
- ✓ A indexação depende somente da seleção de bits do endereço de memória solicitado
- ✗ Todos os blocos podem ser mapeados na mesma linha da cache (falta por conflito)
- ✗ A substituição dos blocos aumentam as faltas que degradam o desempenho do sistema

Gerenciamento de cache

- Mapeamento totalmente associativo
 - Todos os identificadores são comparados em paralelo
 - Somente as palavras do bloco são indexadas



Memória (32 KiB)

Gerenciamento de cache

- Mapeamento totalmente associativo
 - Redução de colisões de mapeamento
 - Escolha da política de substituição

Todos os identificadores são comparados em paralelo

| | | | | | |
|---|--------------|-----|-----|-----|-----|
| S | 000000000000 | 0 | 1 | 2 | 3 |
| S | 00000001000 | 32 | 33 | 34 | 35 |
| S | 00000010000 | 64 | 65 | 66 | 67 |
| S | 00000011000 | 128 | 129 | 130 | 131 |
| S | 00000100000 | 160 | 161 | 162 | 163 |
| S | 00000101000 | 196 | 197 | 198 | 199 |
| S | 00000110000 | 224 | 225 | 226 | 227 |
| S | 00000111000 | 256 | 257 | 258 | 259 |

↑ ↑ ↑ ↑ ↑ ↑

Validade Identificador P₀ P₁ P₂ P₃

Gerenciamento de cache

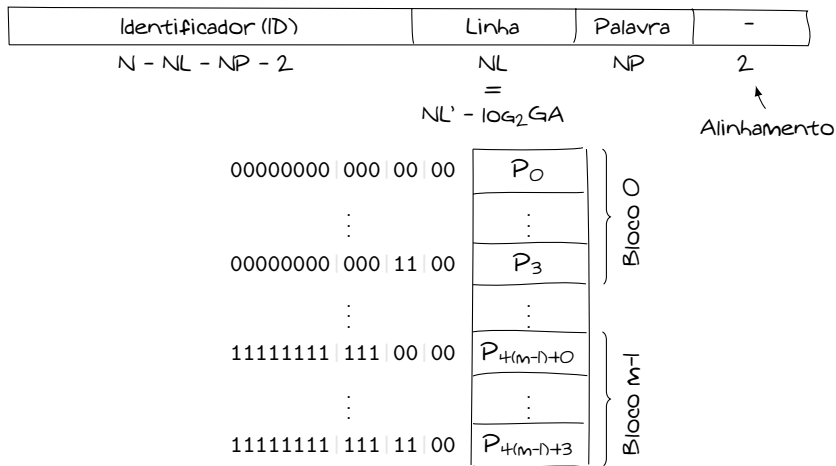
- ▶ Mapeamento totalmente associativo
 - ✓ Não existe conflito de endereçamento, uma vez que os blocos podem ser associados a qualquer linha da cache

Gerenciamento de cache

- ▶ Mapeamento totalmente associativo
 - ✓ Não existe conflito de endereçamento, uma vez que os blocos podem ser associados a qualquer linha da cache
 - ✗ Alto custo de comparação paralela de todos os identificadores armazenados nas linhas

Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Compromisso entre o mapeamento direto (simples) e o totalmente associativo (sem conflitos)



Memória (32 KiB)

Departamento de Computação / UFS

Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 1 (mapeamento direto)

| | | | | | | |
|-----|---|--|--|--|--|--|
| 000 | N | | | | | |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Diagram illustrating a direct-mapped cache structure with 8 lines (000 to 111). The first column is labeled "Validade" (Validity) and contains 'N' for all lines. The second column is labeled "Identificador" (Identifier) and is shaded gray. The remaining four columns are labeled P_0 , P_1 , P_2 , and P_3 , representing the data bits. Arrows point from the labels to their respective columns. A label "Linha" (Line) points to the row indices.

Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 1 (mapeamento direto)

| | | | | | | |
|-----|---|--|--|--|--|--|
| 000 | N | | | | | |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Diagram illustrating a direct-mapped cache structure with 8 lines (000 to 111). The cache is organized into columns: Validade, Identificador, and three pointer columns (P_0 , P_1 , P_2 , P_3). The first column (Validade) contains 'N' for all lines. The second column (Identificador) is shaded gray for lines 000 and 001. The third column (P_0) is shaded gray for lines 000 and 001. The fourth column (P_1) is shaded gray for lines 000 and 001. The fifth column (P_2) is shaded gray for lines 000 and 001. The sixth column (P_3) is shaded gray for lines 000 and 001.

Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 1 (mapeamento direto)

| | | | | | | |
|-----|---|--|--|--|--|--|
| 000 | N | | | | | |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Diagram illustrating a direct-mapped cache structure with 8 lines (000 to 111). Each line contains a tag (N) and a validity bit (V). The validity bit is highlighted in the 010 line. The diagram also shows the mapping of memory blocks P₀, P₁, P₂, and P₃ to the cache lines.

Labels below the table:

- Linha (points to the row index)
- Validade (points to the validity bit)
- Identificador (points to the tag)
- P₀, P₁, P₂, P₃ (point to the data fields)

Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 1 (mapeamento direto)

| | | | | | | |
|-----|---|--|--|--|--|--|
| 000 | N | | | | | |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Diagram illustrating a direct-mapped cache structure with 8 lines (000 to 111). The cache is organized into columns: Validade, Identificador, and four data columns (P_0 , P_1 , P_2 , P_3). The first column (Validade) contains 'N' for all lines. The second column (Identificador) is highlighted in gray for line 011. The data columns (P_0 , P_1 , P_2 , P_3) are empty for all lines.

Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 1 (mapeamento direto)

| | | | | | | |
|-----|---|--|--|--|--|--|
| 000 | N | | | | | |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Diagram illustrating a direct-mapped cache structure with 8 lines (000 to 111). Each line contains a tag (N) and a validity bit (V). The cache is divided into two sets of four lines each. The first set (lines 000-011) is shaded gray. The second set (lines 100-111) is also shaded gray. The validity bit (V) is indicated by an arrow pointing to the first column. The identifier (Identificador) is indicated by an arrow pointing to the second column. The pointers P_0 , P_1 , P_2 , and P_3 are indicated by arrows pointing to the third, fourth, fifth, and sixth columns respectively. The label "Linha" points to the first column.

Gerenciamento de cache

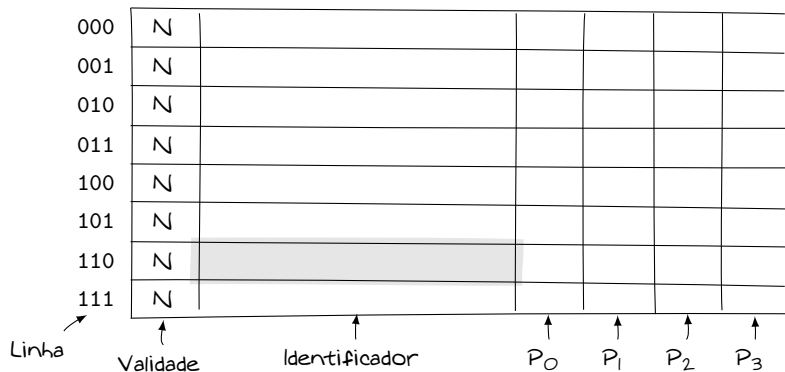
- Mapeamento associativo por conjunto
 - Grau de associatividade 1 (mapeamento direto)

| | | | | | | |
|-----|---|--|--|--|--|--|
| 000 | N | | | | | |
| 001 | N | | | | | |
| 010 | N | | | | | |
| 011 | N | | | | | |
| 100 | N | | | | | |
| 101 | N | | | | | |
| 110 | N | | | | | |
| 111 | N | | | | | |

Diagram illustrating a direct-mapped cache structure with 8 lines (000 to 111). The cache is organized into columns: Validade (Validity), Identificador (Identifier), and four pointers (P₀, P₁, P₂, P₃). The first column (Validade) contains 'N' for all lines. The second column (Identificador) is shaded gray for lines 101 and 110, indicating they are valid. The third column (P₀) is shaded gray for lines 101 and 110, indicating they are mapped to P₀. The fourth column (P₁) is shaded gray for lines 101 and 110, indicating they are mapped to P₁. The fifth column (P₂) is shaded gray for lines 101 and 110, indicating they are mapped to P₂. The sixth column (P₃) is shaded gray for lines 101 and 110, indicating they are mapped to P₃.

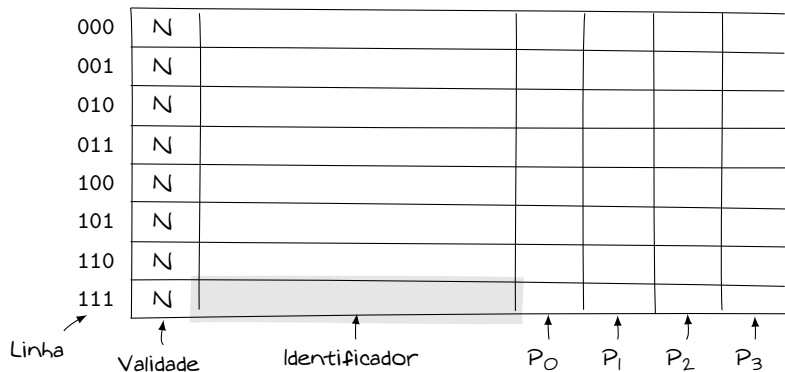
Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 1 (mapeamento direto)



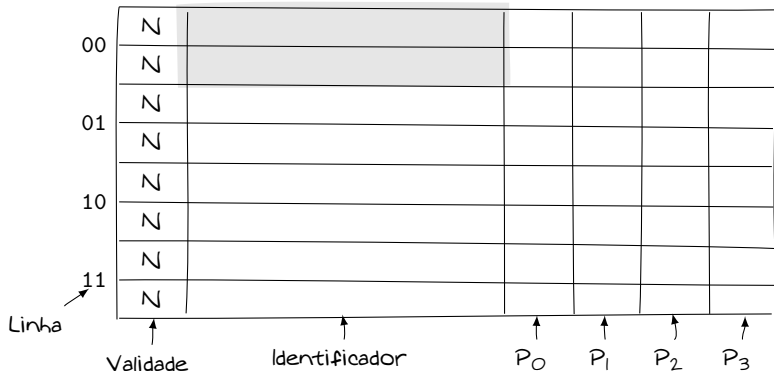
Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 1 (mapeamento direto)



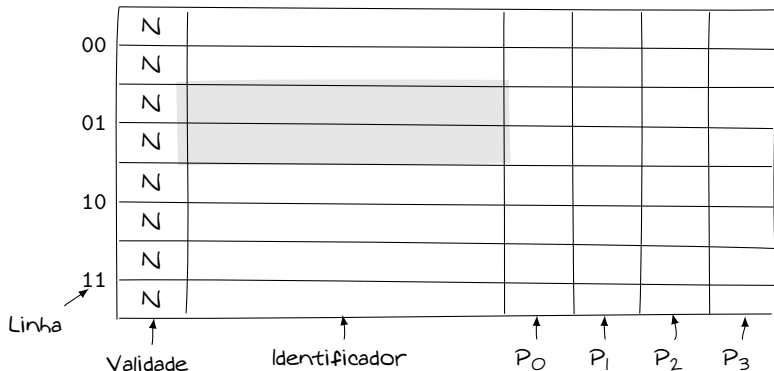
Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 2



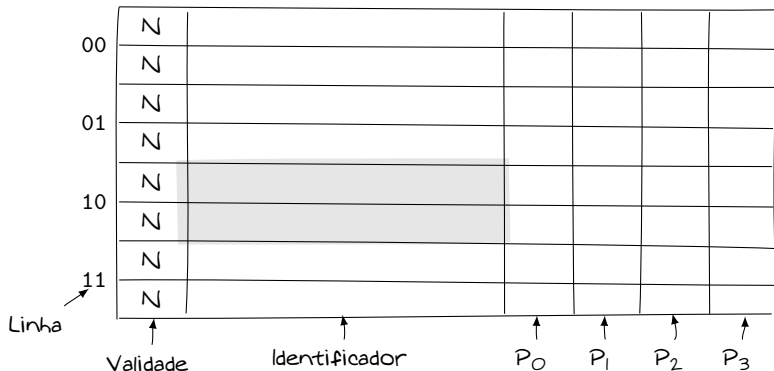
Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 2



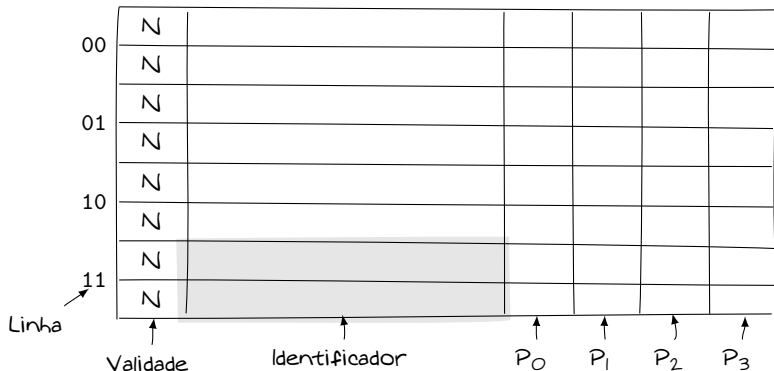
Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 2



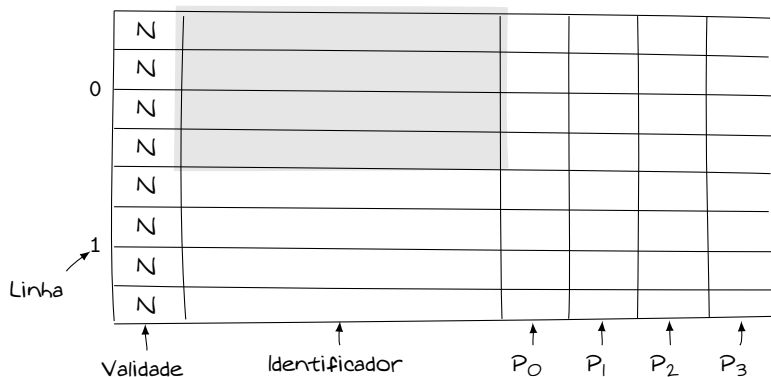
Gerenciamento de cache

- ▶ Mapeamento associativo por conjunto
 - ▶ Grau de associatividade 2



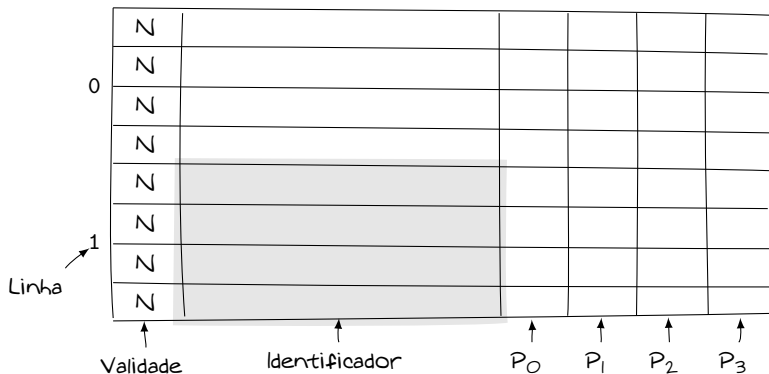
Gerenciamento de cache

- ▶ Mapeamento associativo por conjunto
 - ▶ Grau de associatividade 4



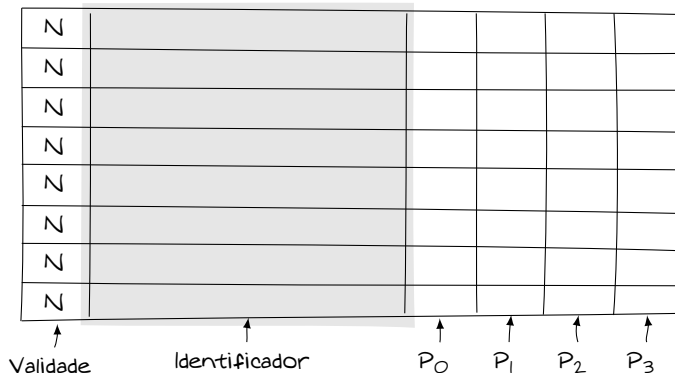
Gerenciamento de cache

- ▶ Mapeamento associativo por conjunto
 - ▶ Grau de associatividade 4



Gerenciamento de cache

- Mapeamento associativo por conjunto
 - Grau de associatividade 8 (totalmente associativo)



Gerenciamento de cache

- ▶ Substituição de dados da cache
 - ▶ Somente é aplicável para mapeamento associativo por conjunto ou totalmente associativo

Gerenciamento de cache

- ▶ Substituição de dados da cache
 - ▶ Somente é aplicável para mapeamento associativo por conjunto ou totalmente associativo
 - ▶ É necessário realizar a substituição devido a capacidade limitada de armazenamento e para manter o dado disponível o máximo de tempo

Gerenciamento de cache

- ▶ Substituição de dados da cache
 - ▶ Somente é aplicável para mapeamento associativo por conjunto ou totalmente associativo
 - ▶ É necessário realizar a substituição devido a capacidade limitada de armazenamento e para manter o dado disponível o máximo de tempo
 - ▶ Define qual será o critério ou a política adotada para substituir o dado armazenado na cache
 - ▶ *First-In First-Out* (FIFO)
 - ▶ *Last Frequently Used* (LFU)
 - ▶ *Last Recently Used* (LRU)
 - ▶ Randômico

Gerenciamento de cache

- ▶ Algoritmos de substituição
 - ▶ FIFO
 - ▶ O bloco mais antigo sempre é substituído, independente de quantas referências são realizadas
 - ▶ Pode ser implementado utilizando uma estrutura de fila

Gerenciamento de cache

- ▶ Algoritmos de substituição
 - ▶ FIFO
 - ▶ O bloco mais antigo sempre é substituído, independente de quantas referências são realizadas
 - ▶ Pode ser implementado utilizando uma estrutura de fila
 - ▶ LFU
 - ▶ O bloco que é menos frequentemente acessado pelo processador é substituído na cache
 - ▶ É feito um controle na linha da cache para medir a frequência de acesso (contador)

Gerenciamento de cache

- ▶ Algoritmos de substituição
 - ▶ LRU
 - ▶ O bloco que foi referenciado a mais tempo pelo processador é substituído na cache
 - ▶ Cada linha possui um controle de idade que é reinicializado a cada acesso e incrementado caso não seja referenciado

Gerenciamento de cache

- ▶ Algoritmos de substituição
 - ▶ LRU
 - ▶ O bloco que foi referenciado a mais tempo pelo processador é substituído na cache
 - ▶ Cada linha possui um controle de idade que é reinicializado a cada acesso e incrementado caso não seja referenciado
 - ▶ Randômico
 - ▶ Os blocos são aleatoriamente substituídos
 - ▶ Necessita de um gerador de números aleatórios para calcular a posição que será descartada

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Quando uma solicitação de escrita é realizada pelo processador, este dado é armazenado na cache

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Quando uma solicitação de escrita é realizada pelo processador, este dado é armazenado na cache
 - ▶ Assim como ocorre na leitura, o objetivo é reduzir a latência das operações de acesso a memória

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Quando uma solicitação de escrita é realizada pelo processador, este dado é armazenado na cache
 - ▶ Assim como ocorre na leitura, o objetivo é reduzir a latência das operações de acesso a memória
 - ▶ Já foi visto que a cache possui capacidade limitada e que os dados armazenados podem ser substituídos ao longo da execução do software

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Quando uma solicitação de escrita é realizada pelo processador, este dado é armazenado na cache
 - ▶ Assim como ocorre na leitura, o objetivo é reduzir a latência das operações de acesso a memória
 - ▶ Já foi visto que a cache possui capacidade limitada e que os dados armazenados podem ser substituídos ao longo da execução do software

Quais são as formas de escrever os dados na memória antes que sejam substituídos?

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Escrita direta (*write through*)
 - ▶ Toda vez que uma operação de escrita é realizada, o dado é imediatamente transferido para a memória

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Escrita direta (*write through*)
 - ▶ Toda vez que uma operação de escrita é realizada, o dado é imediatamente transferido para a memória
 - ▶ É a técnica mais simples para garantir que os valores estejam consistentes em todos os níveis da hierarquia

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Escrita direta (*write through*)
 - ▶ Toda vez que uma operação de escrita é realizada, o dado é imediatamente transferido para a memória
 - ▶ É a técnica mais simples para garantir que os valores estejam consistentes em todos os níveis da hierarquia
 - ▶ A grande desvantagem desta técnica é o tráfego intenso com a memória que pode gerar retenção nos acessos ao barramento do sistema

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Escrita atrasada (*write back*)
 - ▶ Todos os dados são escritos somente na cache até que seja feita a substituição da linha

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Escrita atrasada (*write back*)
 - ▶ Todos os dados são escritos somente na cache até que seja feita a substituição da linha
 - ▶ Para evitar perda de dados durante a operação de substituição, existe um campo de uso na linha da cache para indicar que o dado ainda não foi escrito nos níveis inferiores da hierarquia de memória

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Escrita atrasada (*write back*)
 - ▶ Todos os dados são escritos somente na cache até que seja feita a substituição da linha
 - ▶ Para evitar perda de dados durante a operação de substituição, existe um campo de uso na linha da cache para indicar que o dado ainda não foi escrito nos níveis inferiores da hierarquia de memória
 - ▶ Reduz o tráfego com a memória e a retenção, uma vez que todas as operações são realizadas na cache

Gerenciamento de cache

- ▶ Política de escrita dos dados
 - ▶ Escrita atrasada (*write back*)
 - ▶ Todos os dados são escritos somente na cache até que seja feita a substituição da linha
 - ▶ Para evitar perda de dados durante a operação de substituição, existe um campo de uso na linha da cache para indicar que o dado ainda não foi escrito nos níveis inferiores da hierarquia de memória
 - ▶ Reduz o tráfego com a memória e a retenção, uma vez que todas as operações são realizadas na cache

Não pode ser utilizada em operações de E/S mapeada em memória!

Gerenciamento de cache

- ▶ Política de escrita dos dados em caso de falta
 - ▶ Escrita com alocação (*write allocate*)
 - ▶ É feita a requisição de leitura do bloco da memória
 - ▶ A palavra endereçada está disponível na cache

Gerenciamento de cache

- ▶ Política de escrita dos dados em caso de falta
 - ▶ Escrita com alocação (*write allocate*)
 - ▶ É feita a requisição de leitura do bloco da memória
 - ▶ A palavra endereçada está disponível na cache
 - ▶ Escrita sem alocação (*no write allocate*)
 - ▶ O dado é atualizado diretamente na memória
 - ▶ Evita que blocos inteiros sejam carregados em rotinas de inicialização que escrevem zeros na memória

Gerenciamento de cache

- ▶ Análise qualitativa de desempenho
 - ▶ Tamanho da cache
 - ▶ Uma maior capacidade de armazenamento reduz as substituições de dados até um certo limite
 - ▶ A adoção de múltiplos níveis de cache (L1, L2 e L3) colabora para a redução da penalidade que é o tempo de acesso em caso de falta

Gerenciamento de cache

- ▶ Análise qualitativa de desempenho
 - ▶ Tamanho da cache
 - ▶ Uma maior capacidade de armazenamento reduz as substituições de dados até um certo limite
 - ▶ A adoção de múltiplos níveis de cache (L1, L2 e L3) colabora para a redução da penalidade que é o tempo de acesso em caso de falta
 - ▶ Grau de associatividade
 - ▶ Quanto maior a associatividade, menor o número de colisões e o número de substituições
 - ▶ A política de substituição impacta diretamente na taxa de faltas da cache (desempenho)

Gerenciamento de cache

- ▶ Análise qualitativa de desempenho
 - ▶ Tamanho da cache
 - ▶ Uma maior capacidade de armazenamento reduz as substituições de dados até um certo limite
 - ▶ A adoção de múltiplos níveis de cache (L1, L2 e L3) colabora para a redução da penalidade que é o tempo de acesso em caso de falta
 - ▶ Grau de associatividade
 - ▶ Quanto maior a associatividade, menor o número de colisões e o número de substituições
 - ▶ A política de substituição impacta diretamente na taxa de faltas da cache (desempenho)
 - ▶ Política de escrita
 - ▶ A escrita direta simplifica a coerência, mas aumenta o tráfego com barramento e a memória
 - ▶ Com a escrita atrasada, a latência e o tráfego são reduzidos, mas é preciso controlar as substituições

Gerenciamento de cache

- ▶ Análise quantitativa de desempenho
 - ▶ Tempo de acerto da cache (TA)
 - ▶ Taxa de falta da cache (TF)
 - ▶ Medição da penalidade (P) para acessar os níveis inferiores da hierarquia de memória

$$Latência\ média = TA + TF \times P$$

Gerenciamento de cache

- ▶ Análise quantitativa de desempenho
 - ▶ Frequência de operação de 2 GHz
 - ▶ Tempo de acerto de 3 ciclos (cache)
 - ▶ Penalidade com tempo de 10 ciclos (memória)
 - ▶ Taxa de falta de 5%

$$\begin{aligned}\textit{Latência média} &= 3 + 0,05 \times 10 \\ &= 3,5 \textit{ ciclos} \\ &= 1,75 \textit{ ns}\end{aligned}$$

Gerenciamento de cache

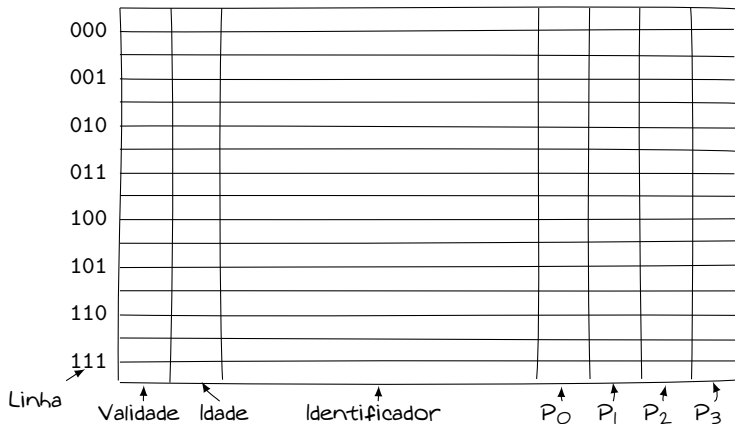
- ▶ Análise quantitativa de desempenho
 - ▶ Frequência de operação de 2 GHz
 - ▶ Tempo de acerto de 3 ciclos (cache)
 - ▶ Penalidade com tempo de 10 ciclos (memória)
 - ▶ Taxa de falta de 5%

$$\begin{aligned}\textit{Latência média} &= 3 + 0,05 \times 10 \\ &= 3,5 \textit{ ciclos} \\ &= 1,75 \textit{ ns}\end{aligned}$$

A cache proporciona um aumento médio de $10 \textit{ ciclos} \div 3,5 \textit{ ciclos} \approx 3$ vezes no desempenho

Exercício

- Implemente duas caches separadas para dados (D) e instruções (I), com capacidade de 256 bytes, blocos de 4 palavras e associatividade de grau 2



- Implemente a política LRU com escrita direta (*write through*) sem alocação (*no write allocate*), excluindo os endereços mapeados em memória

Exercício

- ▶ Os eventos de acerto ou de falta nos acessos às caches devem ser exibidos durante a execução de cada uma das instruções, com as taxas de acerto de das caches D e I no final da execução

```
[START OF SIMULATION]
.
.
.
0x?????????: ?_read_hit [u]->[u] ID=0x??????,DATA={...}
0x?????????: ?_read_miss [u] [u]{VAL=u,AGE=u,ID=0x??????},...
0x?????????: ?_write_hit [u]->[u] ID=0x??????,DATA={...}
0x?????????: ?_write_miss [u] [u]{VAL=u,AGE=u,ID=0x??????},...
.
.
.
[CACHE]
D_hit_rate: ?.??%
I_hit_rate: ?.??%
[END OF SIMULATION]
```