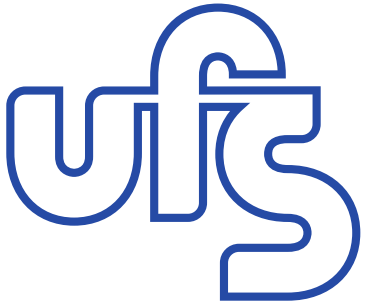
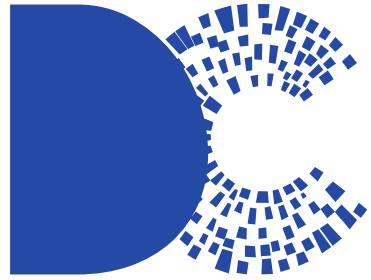




Hierarquia de memória

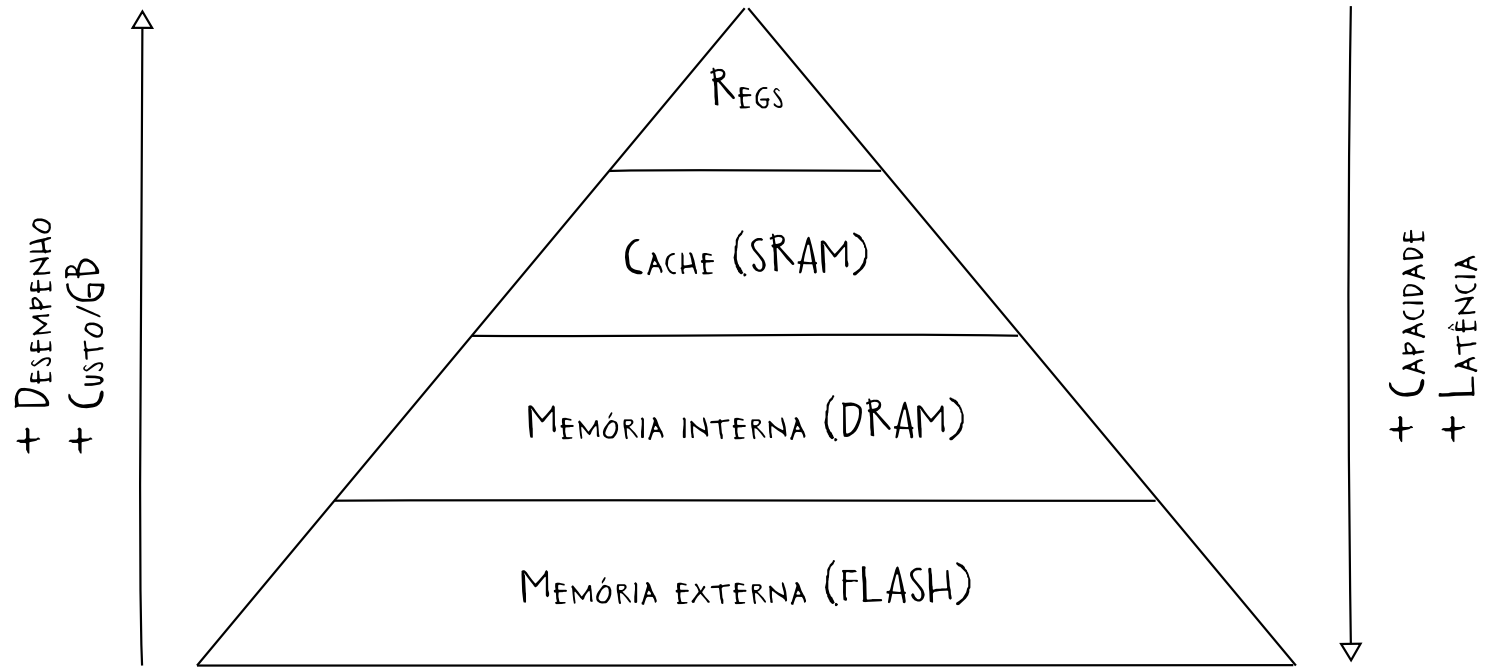
Arquitetura de Computadores

Bruno Prado

Departamento de Computação / UFS

Introdução

- O que é hierarquia de memória?
  - É uma organização de memória que utiliza componentes com diversas tecnologias, capacidades e desempenhos



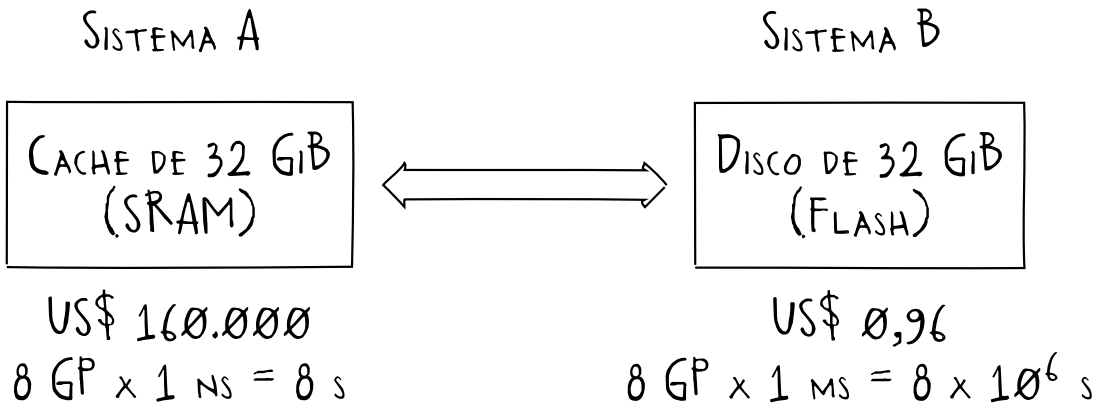
Introdução

- Análise comparativa das memórias

Tipo	Capacidade	Custo	Latência
Imediato	1 <-> 3 bytes	-	-
SRAM	2 KiB <-> 32 Mbit	~US$ 5k / GiB	0,20 <-> 2 ns
DRAM	1 <-> 16 GiB	~US$ 1 / GiB	~10 ns
FLASH	0,1 <-> 32 TB	~US$ 0,03 / GB	~1 ms

Introdução

- Qual o propósito da hierarquia de memória?
  - Abstrair e combinar as tecnologias que estão sendo utilizadas e reduzir as limitações associadas
  - Otimizar a relação entre desempenho e custo das tecnologias de armazenamento



Introdução

- Qual o propósito da hierarquia de memória?
  - Combinar diferentes tecnologias de armazenamento para maximizar o desempenho e reduzir o custo total
  - Considerando que o dado solicitado está disponível com probabilidade de acerto de 90% na cache

Sistema C

Cache de 16 MiB (SRAM)
Memória de 32 GiB (DRAM)
Disco de 256 GB (Flash)

US$ 78,13 + US$ 32 + US$ 7,68 = US$ 117,81

7,2 GP x 1 ns + 0,8 GP x 10 ns = 15,2 s

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

- Categorização dos dispositivos de memória
  - Localização
    - Interna: são diretamente acessados ou controlados pelo processador, como os registradores do processador e as memórias cache ou principal do sistema (DRAM)
    - Externa: o acesso é feito através de controladores de E/S (PCI Express, SATA ou USB) para as unidades de armazenamento, ex: disco de estado sólido (FLASH)

Introdução

- Categorização dos dispositivos de memória
  - Localização
    - Interna: são diretamente acessados ou controlados pelo processador, como os registradores do processador e as memórias cache ou principal do sistema (DRAM)
    - Externa: o acesso é feito através de controladores de E/S (PCI Express, SATA ou USB) para as unidades de armazenamento, ex: disco de estado sólido (FLASH)
  - Estrutura de armazenamento e endereçamento
    - Tamanho de palavra: define quantos bytes são processados por vez, geralmente refletindo a capacidade da arquitetura (múltiplos de 1 byte com 8, 16 ou 32 bits)
    - Bloco de dados: são conjuntos de dados com tamanhos muito maiores que a palavra do sistema (múltiplos de 512 bytes), sendo utilizados em dispositivos de memória externa

Introdução

- Categorização dos dispositivos de memória
  - Método de acesso
    - Sequencial: os dados estão em sequência na memória (fita magnética), ou seja, para obter o último elemento da sequência, todos os dados precisam ser acessados

Introdução

- Categorização dos dispositivos de memória
  - Método de acesso
    - Sequencial: os dados estão em sequência na memória (fita magnética), ou seja, para obter o último elemento da sequência, todos os dados precisam ser acessados
    - Direto: cada bloco de dados possui um endereço físico único, possibilitando a sua busca direta pelos dados na unidade de armazenamento (SSD)

Introdução

- Categorização dos dispositivos de memória
  - Método de acesso
    - Sequencial: os dados estão em sequência na memória (fita magnética), ou seja, para obter o último elemento da sequência, todos os dados precisam ser acessados
    - Direto: cada bloco de dados possui um endereço físico único, possibilitando a sua busca direta pelos dados na unidade de armazenamento (SSD)
    - Aleatório: os dados são armazenados como no método direto, entretanto, o tempo de acesso é constante e independente da última posição acessada (RAM)

Introdução

- Categorização dos dispositivos de memória
  - Método de acesso
    - Sequencial: os dados estão em sequência na memória (fita magnética), ou seja, para obter o último elemento da sequência, todos os dados precisam ser acessados
    - Direto: cada bloco de dados possui um endereço físico único, possibilitando a sua busca direta pelos dados na unidade de armazenamento (SSD)
    - Aleatório: os dados são armazenados como no método direto, entretanto, o tempo de acesso é constante e independente da última posição acessada (RAM)
    - Associativo: o acesso é aleatório (SRAM) por indexação ou associação do dado ao seu endereçamento na memória (cache), reduzindo o tempo de acesso

Introdução

- Categorização dos dispositivos de memória
  - Capacidade e desempenho
    - Latência: é o tempo gasto para realizar uma operação de E/S que é constante no acesso aleatório

$$Latência = t_{busca} + t_{operação}$$

Introdução

- Categorização dos dispositivos de memória
  - Capacidade e desempenho
    - Latência: é o tempo gasto para realizar uma operação de E/S que é constante no acesso aleatório

$$Latência = t_{busca} + t_{operação}$$

    - Taxa de transferência: é a taxa com que os dados podem ser escritos ou lidos de um dispositivo, dependente da frequência de operação e da operação realizada

$$Taxa\ de\ transferência = \frac{\#bits}{Latência}$$

Memória externa

- Dispositivos de armazenamento não volátil
  - Eletromecânicos
    - Disco e fita magnéticas (HDD, DAT, etc)
    - Mídias óticas removíveis (CD, DVD, etc)

$$t_{acesso} = t_{busca} + t_{rotação} + t_{E/S}$$

Memória externa

- Dispositivos de armazenamento não volátil
  - Eletromecânicos
    - Disco e fita magnéticas (HDD, DAT, etc)
    - Mídias óticas removíveis (CD, DVD, etc)

$$t_{acesso} = t_{busca} + t_{rotação} + t_{E/S}$$

  - Estado sólido (memória FLASH)
    - Disco de estado sólido (SSD)
    - Mídias eletrônicas removíveis (*pendrive*)

$$t_{acesso} = t_{busca} + t_{E/S}$$

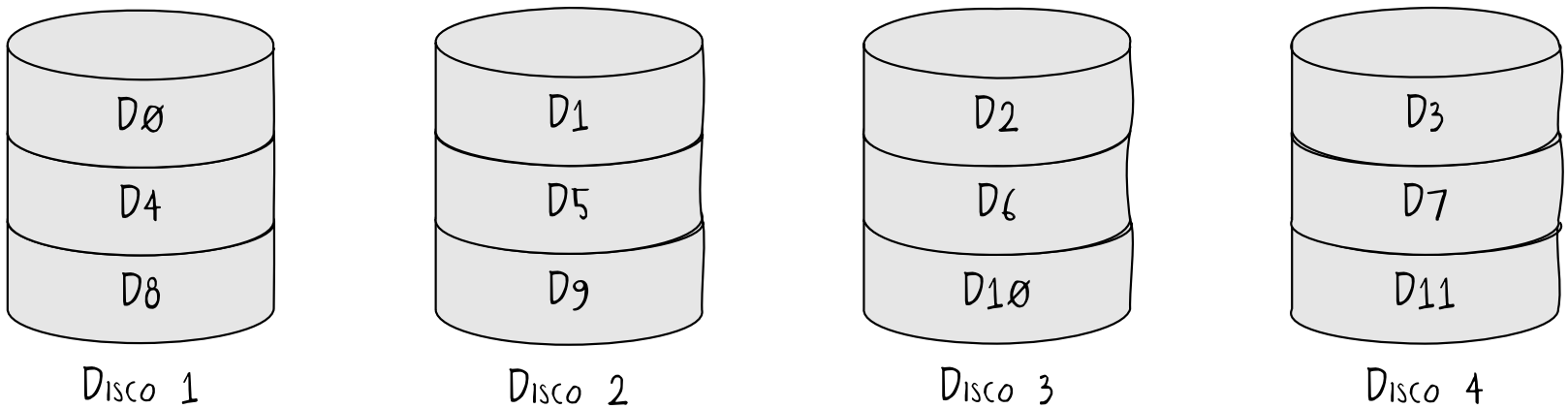
Memória externa

- *Redundant Array of Independent Disks* (RAID)
  - Técnicas para aumentar o desempenho e redundância utilizando múltiplos discos

Nível	Categoria	# Discos	Descrição
0	Divisão	N	Sem redundância
1	Espelhamento	2N	Redundância dos dados
2	Acesso paralelo	N + logN	Redundância com Hamming
3		N + 1	Paridade com bits intercalados
4	Acesso independente dos discos	N + 1	Paridade com blocos intercalados
5		N + 1	Paridade distribuída com blocos intercalados
6		N + 2	Paridade dupla distribuída com blocos intercalados

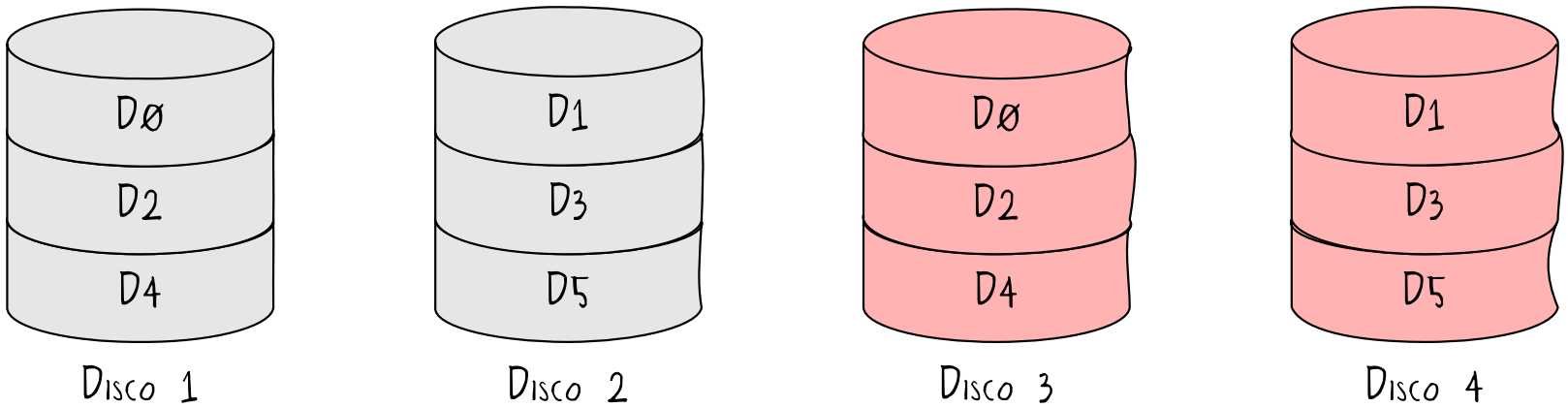
Memória externa

- *Redundant Array of Independent Disks* (RAID)
  - Nível 0 (dados divididos nos discos)



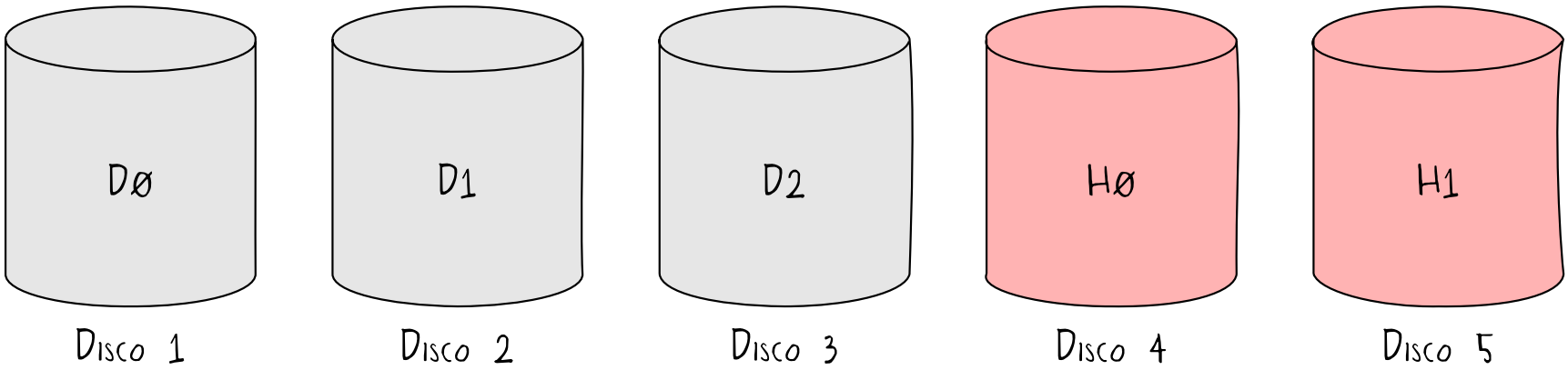
Memória externa

- *Redundant Array of Independent Disks* (RAID)
  - Nível 1 (redundância com espelhamento nos discos)



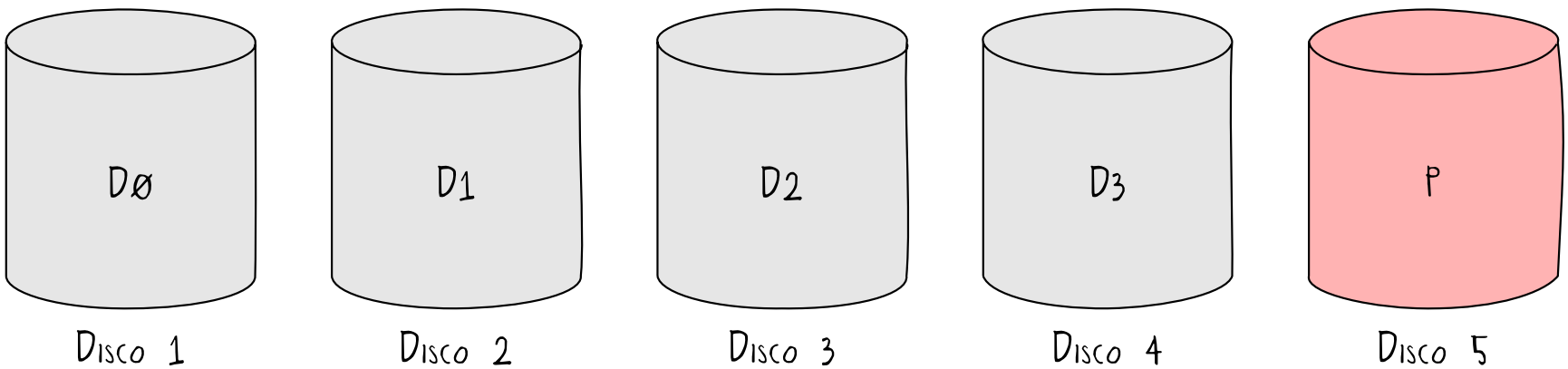
Memória externa

- *Redundant Array of Independent Disks* (RAID)
  - Nível 2 (redundância por código de Hamming)



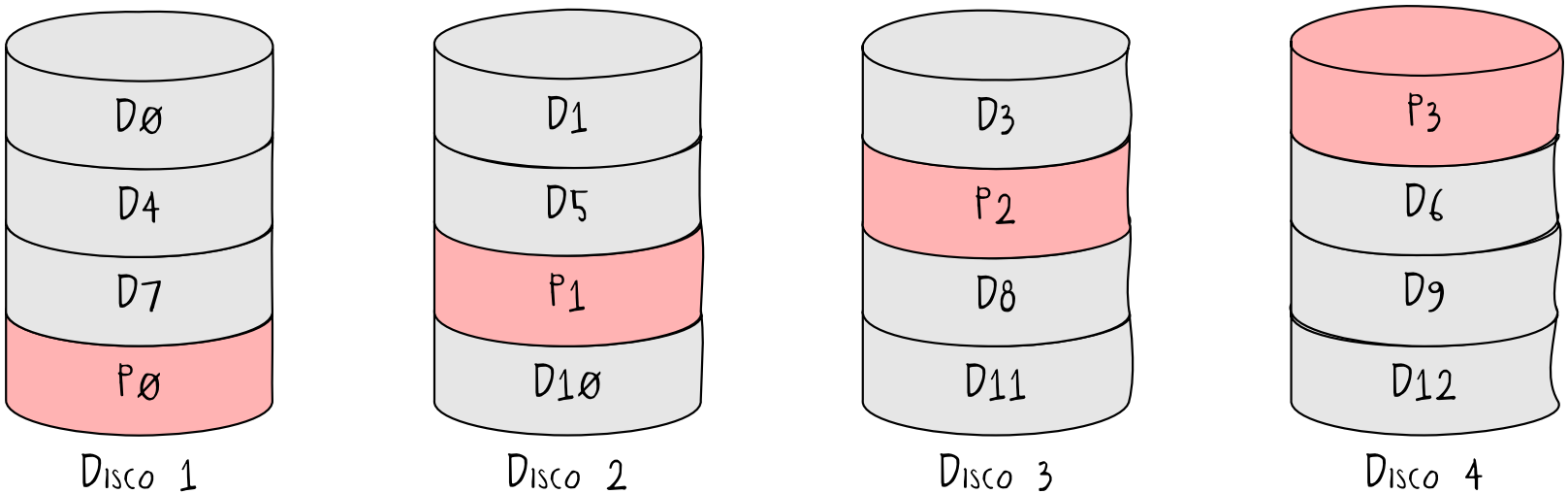
Memória externa

- *Redundant Array of Independent Disks* (RAID)
  - Níveis 3 e 4 (paridade de bits e de blocos)



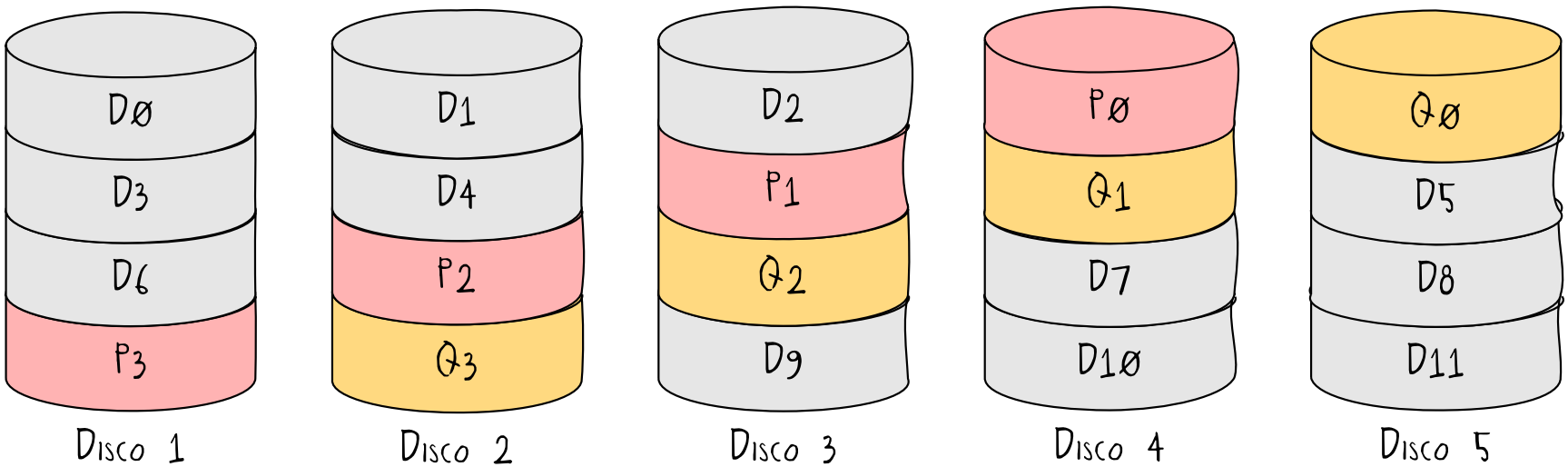
Memória externa

- *Redundant Array of Independent Disks* (RAID)
  - Nível 5 (paridade distribuída de blocos)



Memória externa

- *Redundant Array of Independent Disks* (RAID)
  - Nível 6 (paridade dupla distribuída de blocos)



Memória interna

- Dispositivos de armazenamento volátil
  - Registrador: é definido pela arquitetura e opera na mesma frequência do processador, mas com uma capacidade e quantidade de armazenamento reduzidas

Memória interna

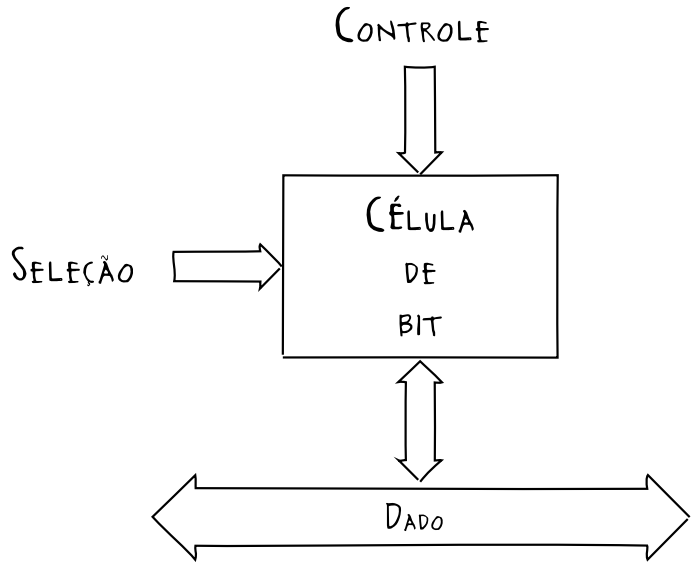
- Dispositivos de armazenamento volátil
  - Registrador: é definido pela arquitetura e opera na mesma frequência do processador, mas com uma capacidade e quantidade de armazenamento reduzidas
  - Memória cache (SRAM): é embarcada dentro do processador e armazena os últimos valores acessados pela memória principal, para melhorar o desempenho

Memória interna

- Dispositivos de armazenamento volátil
  - Registrador: é definido pela arquitetura e opera na mesma frequência do processador, mas com uma capacidade e quantidade de armazenamento reduzidas
  - Memória cache (SRAM): é embarcada dentro do processador e armazena os últimos valores acessados pela memória principal, para melhorar o desempenho
  - Memória principal (DRAM): tem o papel de armazenamento dos código e dados utilizados pelas aplicações carregados da memória externa

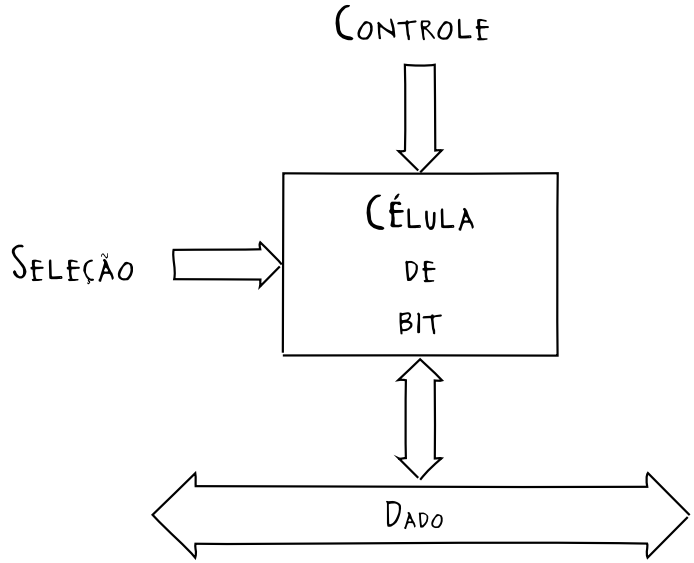
Memória interna

- A memória interna utiliza um sistema binário para armazenar os dados em unidades básicas (células)
  - É necessário controlar e selecionar as células para realizar as operações de escrita ou leitura dos bits



Memória interna

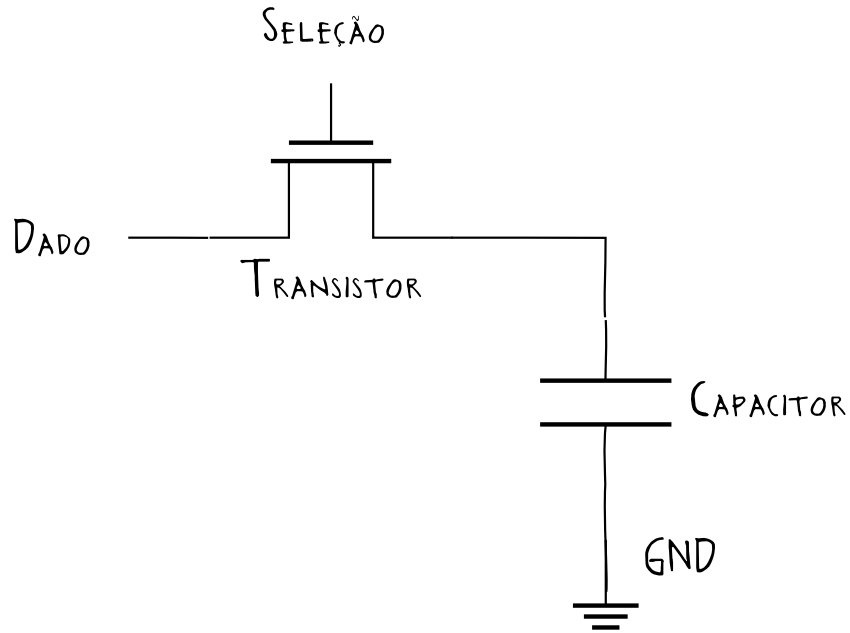
- A memória interna utiliza um sistema binário para armazenar os dados em unidades básicas (células)
  - É necessário controlar e selecionar as células para realizar as operações de escrita ou leitura dos bits



CADA BIT É ARMAZENADO INDIVIDUALMENTE
(EM 4 GB EXISTEM 34.359.738.368 CÉLULAS)

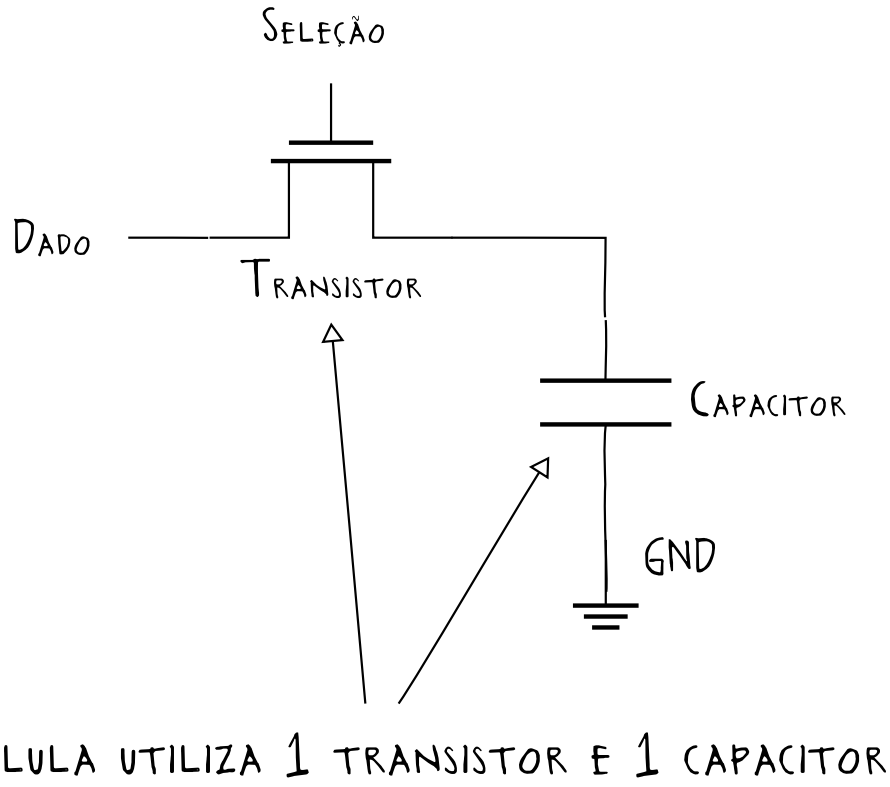
Memória interna

- Memória principal (DRAM)
  - A tecnologia dinâmica de armazenamento utiliza capacitores para armazenar o valor binário 0 (descarregado) ou 1 (carregado) da célula
  - Como existe uma tendência natural de perder a carga, o capacitor precisa ser periodicamente recarregado



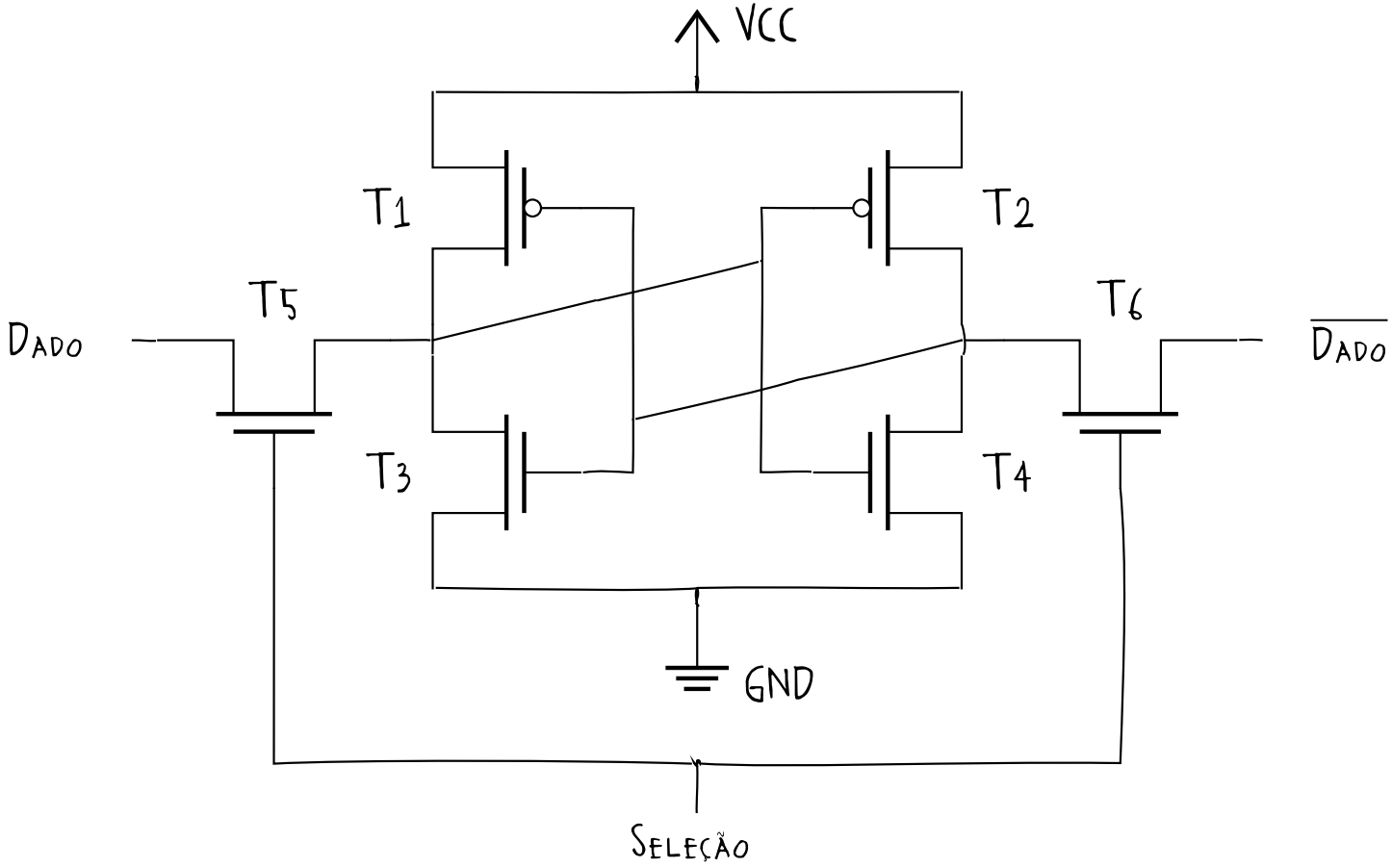
Memória interna

- Memória principal (DRAM)
  - A tecnologia dinâmica de armazenamento utiliza capacitores para armazenar o valor binário 0 (descarregado) ou 1 (carregado) da célula
  - Como existe uma tendência natural de perder a carga, o capacitor precisa ser periodicamente recarregado



Memória interna

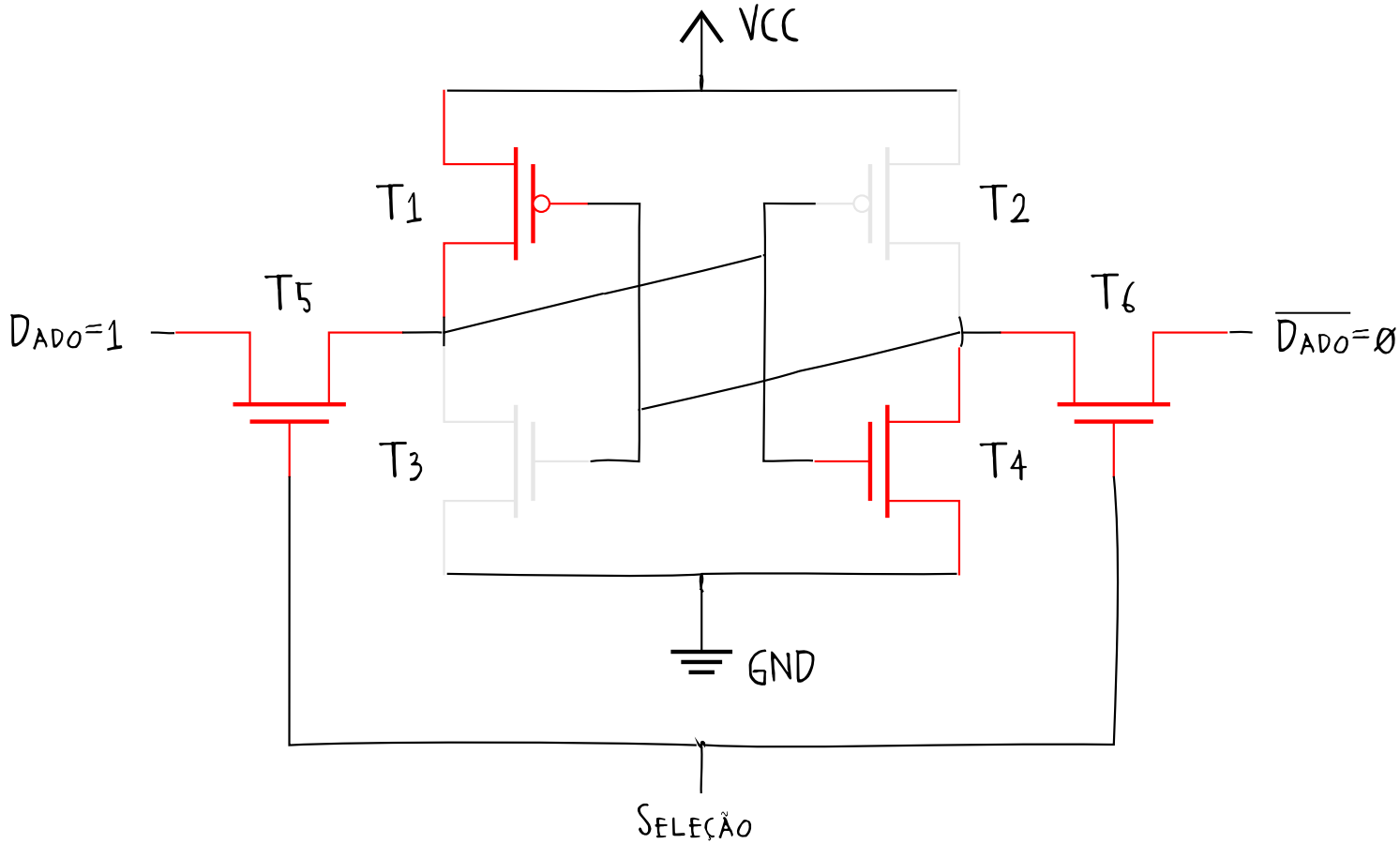
- Memória cache (SRAM)
  - Utiliza o mesmo tipo de componente do processador (transistor) para armazenar os valores binários 0 e 1



CADA CÉLULA UTILIZA 6 TRANSISTORES

Memória interna

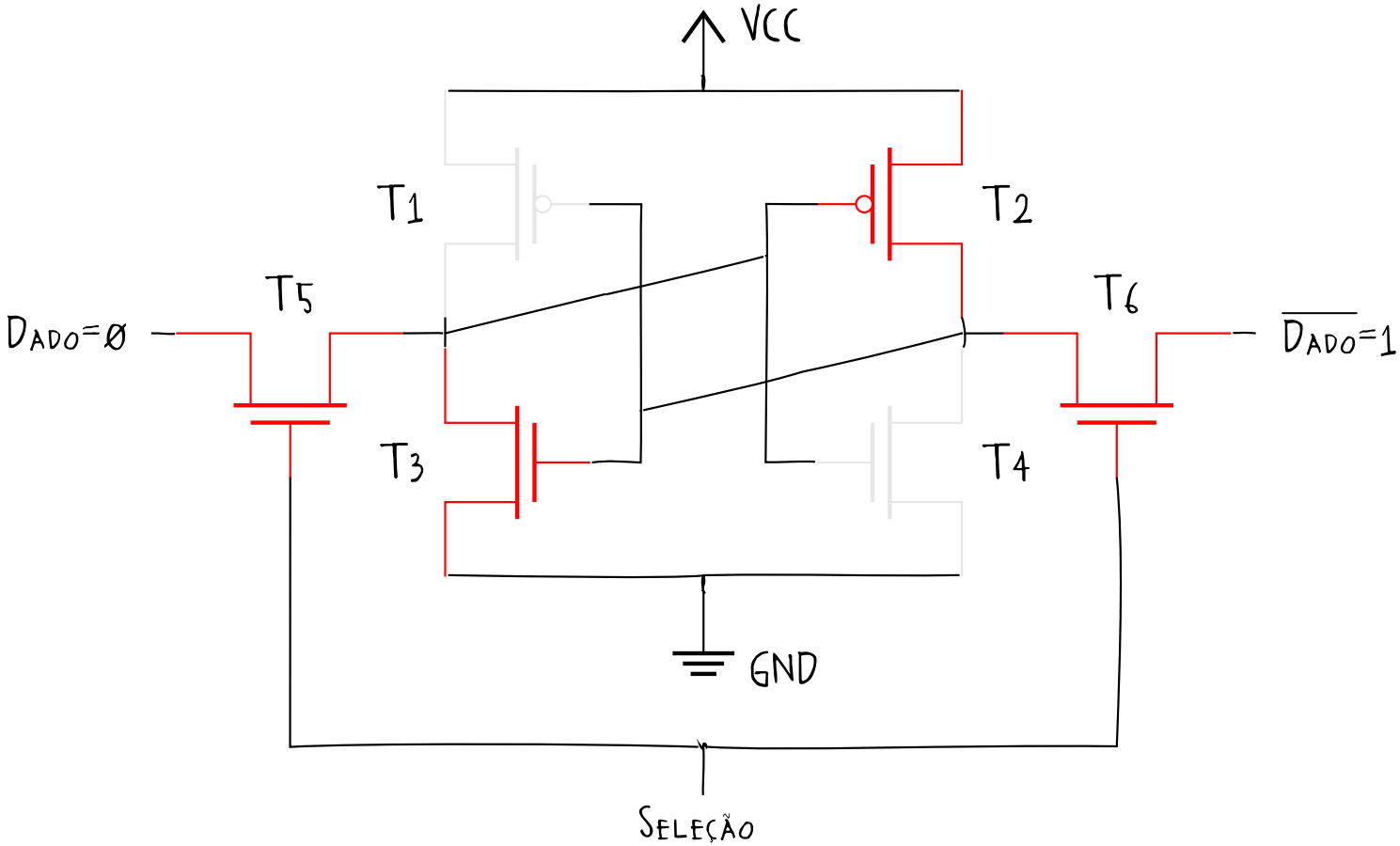
- Memória cache (SRAM)
  - Utiliza o mesmo tipo de componente do processador (transistor) para armazenar os valores binários 0 e 1



CADA CÉLULA UTILIZA 6 TRANSISTORES

Memória interna

- Memória cache (SRAM)
  - Utiliza o mesmo tipo de componente do processador (transistor) para armazenar os valores binários 0 e 1



CADA CÉLULA UTILIZA 6 TRANSISTORES

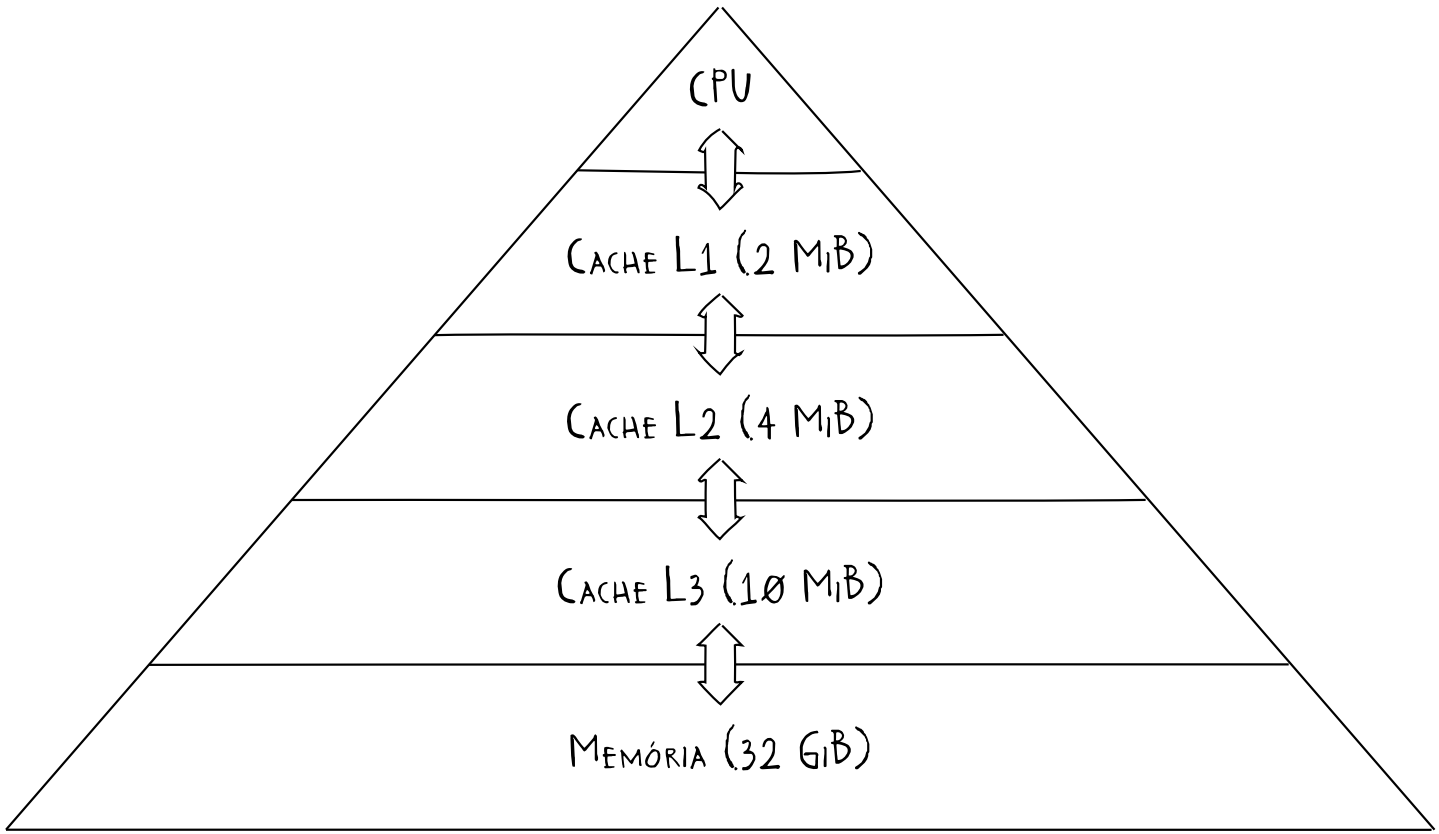
Memória interna

- Comparativo SRAM x DRAM

Característica	SRAM	DRAM
Área e custo de cada célula	↑	↓
Desempenho das operações	↑	↓
Consumo de potência	↓	↑
Densidade de armazenamento	↓	↑

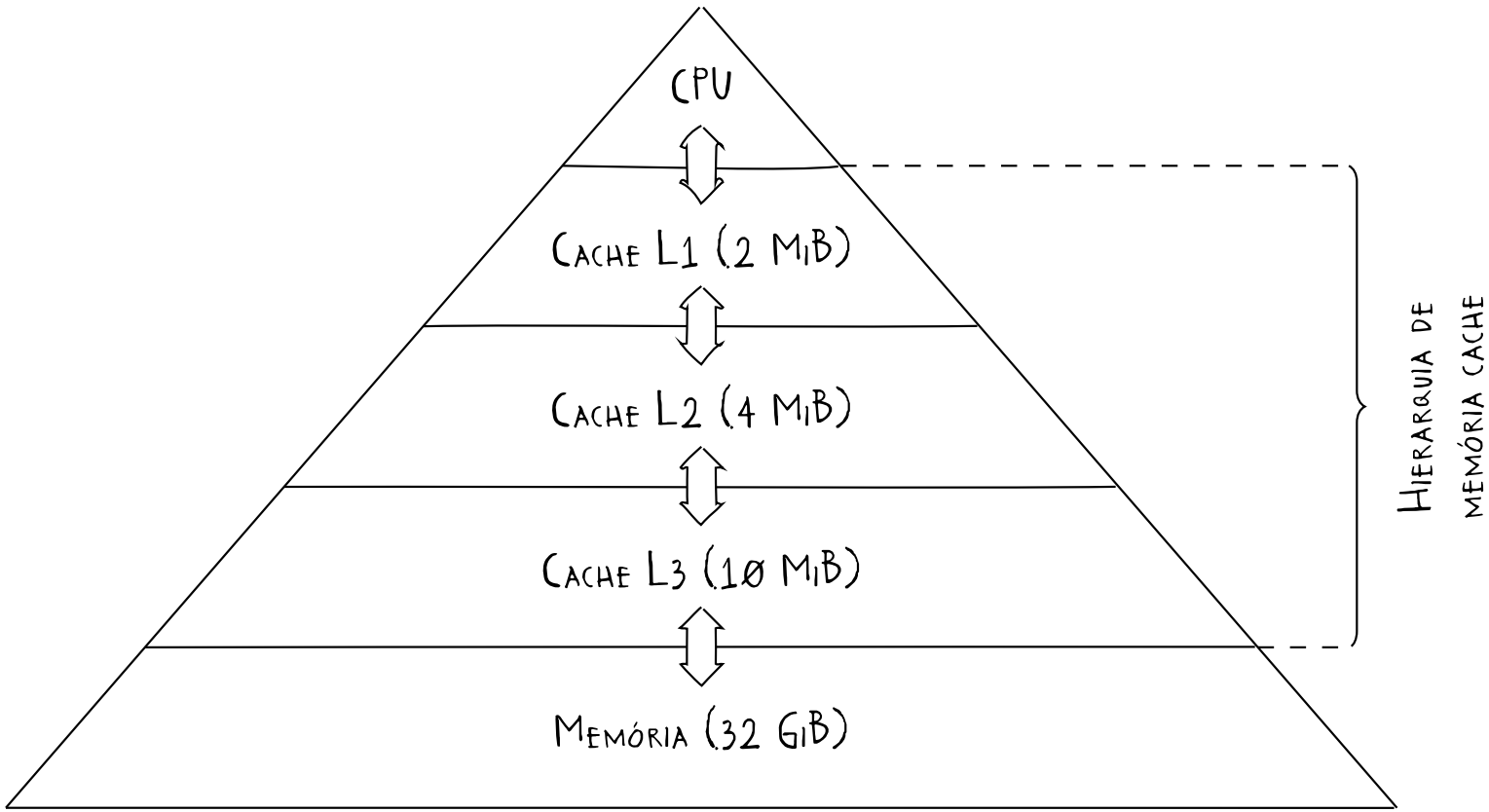
Memória cache

- Características principais
  - Alto desempenho e baixa latência
  - Tamanho limitado por causa do custo elevado
  - Método de acesso associativo da memória principal



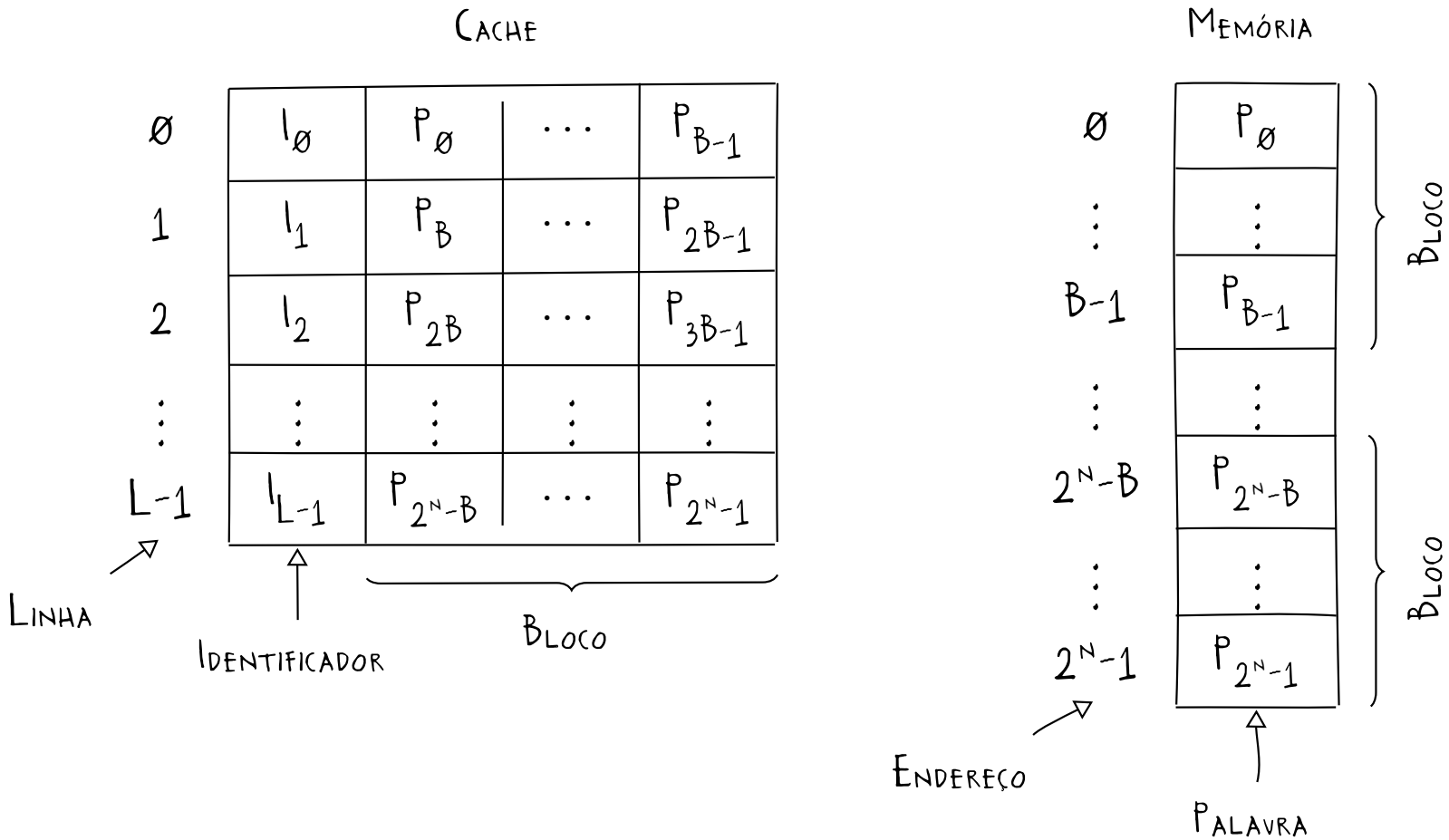
Memória cache

- Características principais
  - Alto desempenho e baixa latência
  - Tamanho limitado por causa do custo elevado
  - Método de acesso associativo da memória principal



Memória cache

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



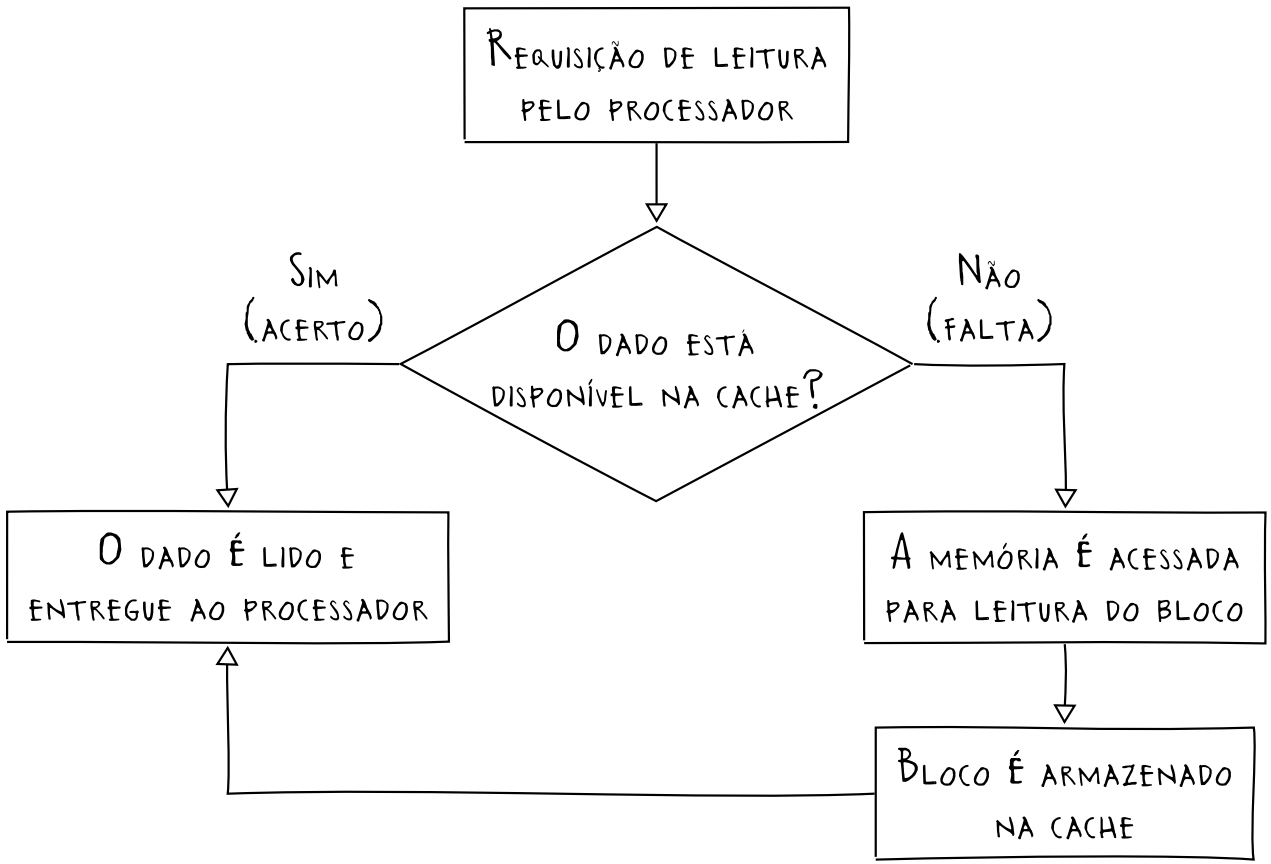
Memória cache

- 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

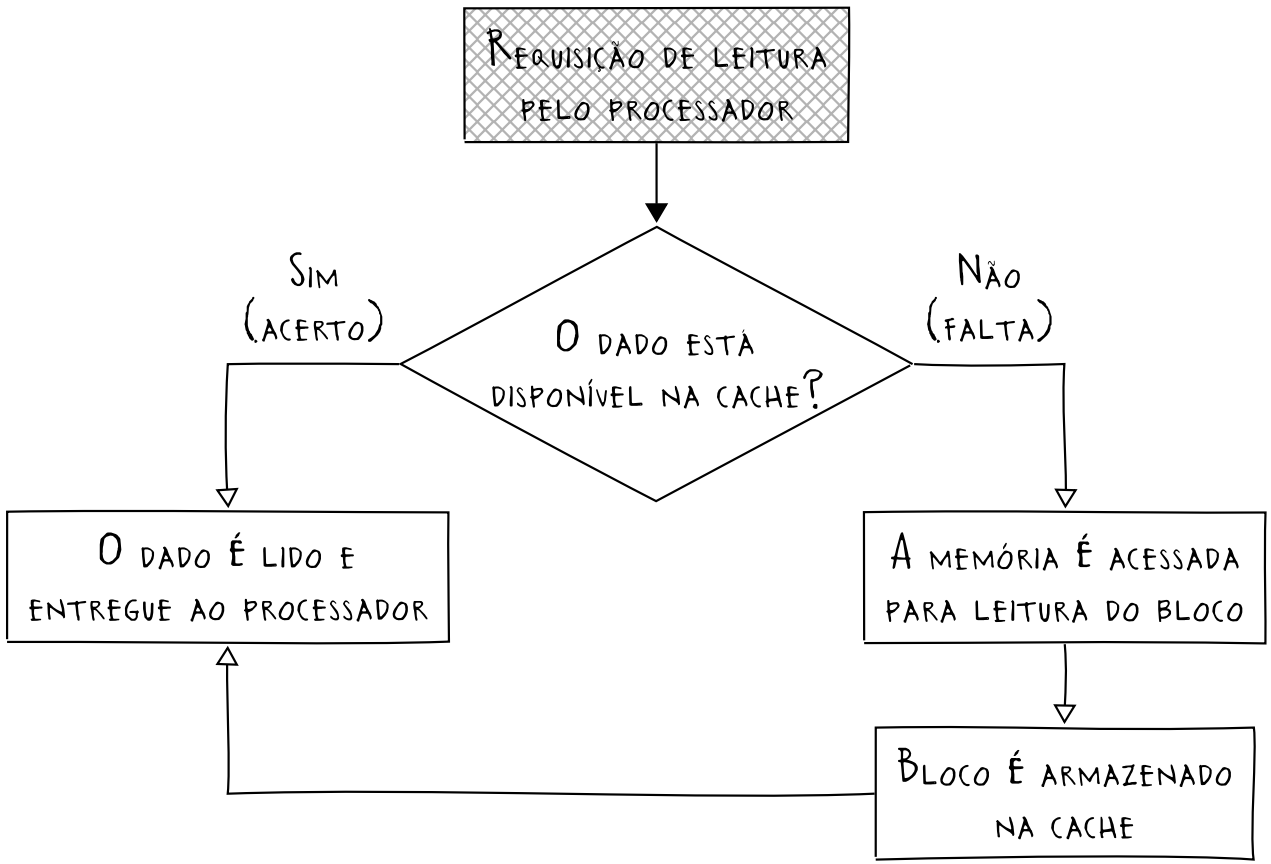
Memória cache

- 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



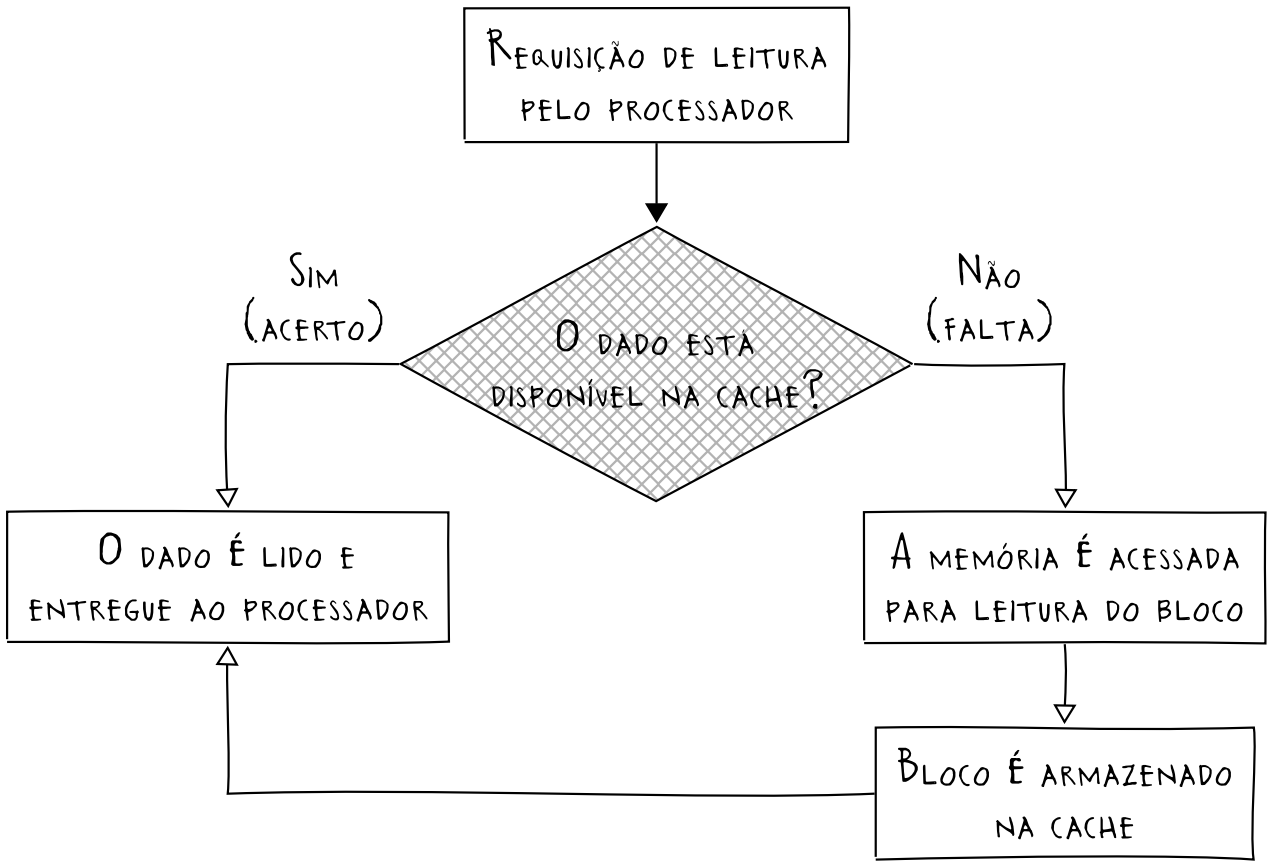
Memória cache

- 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



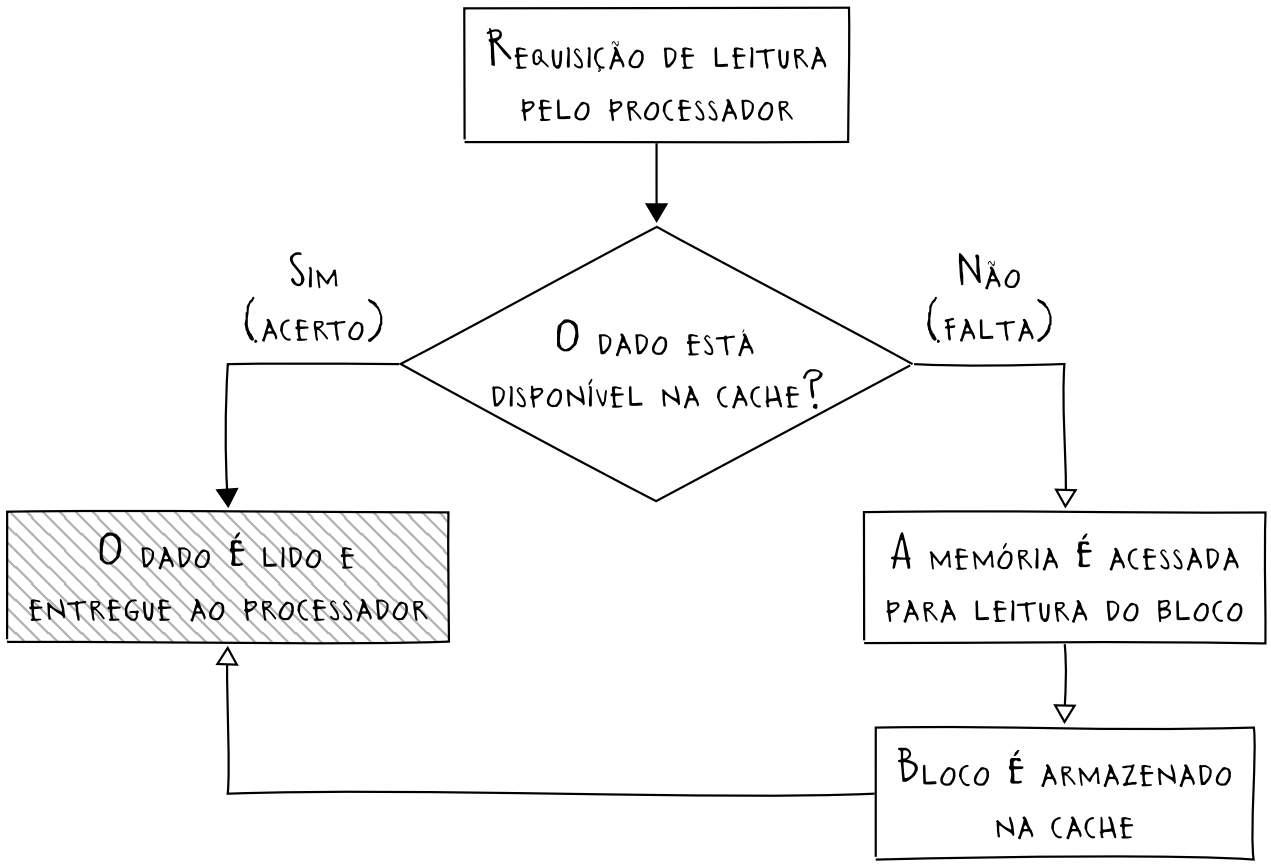
Memória cache

- 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



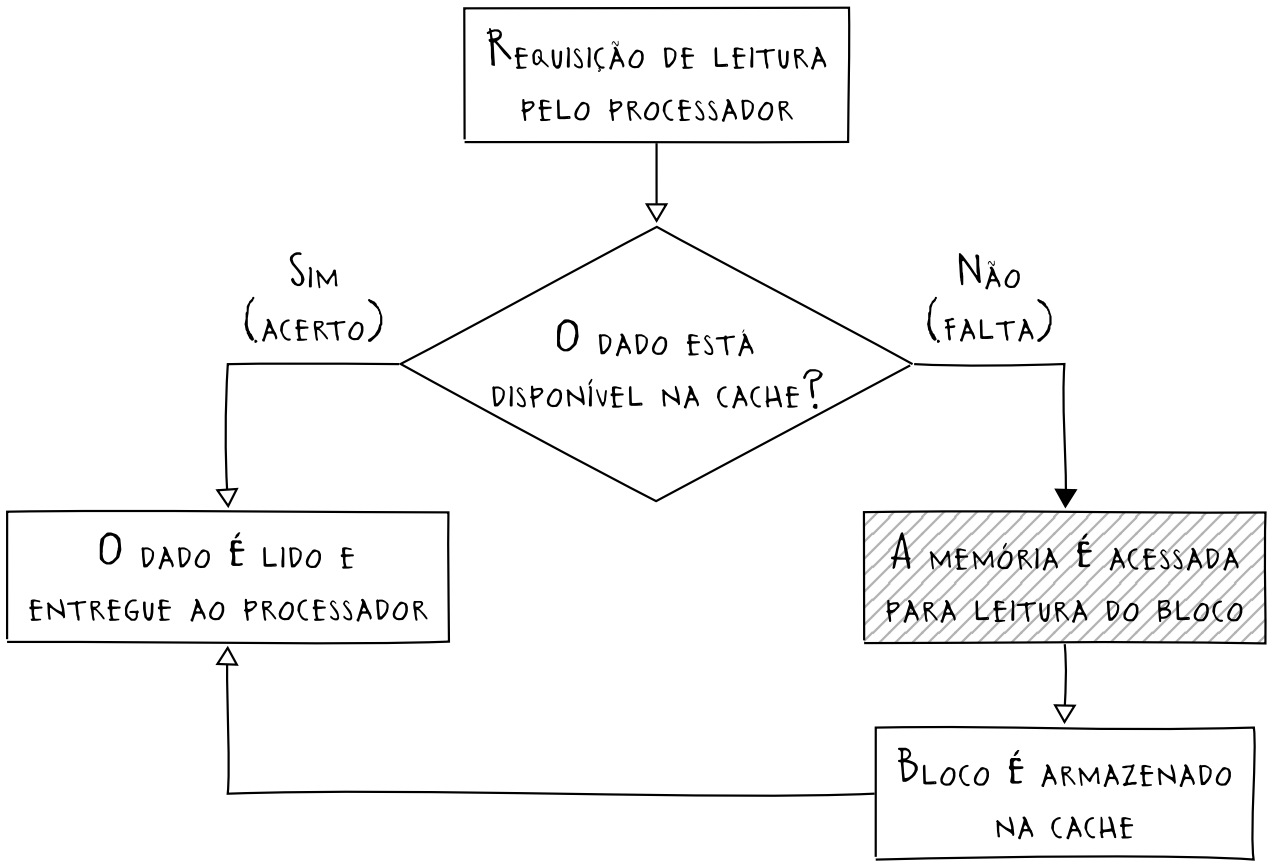
Memória cache

- 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



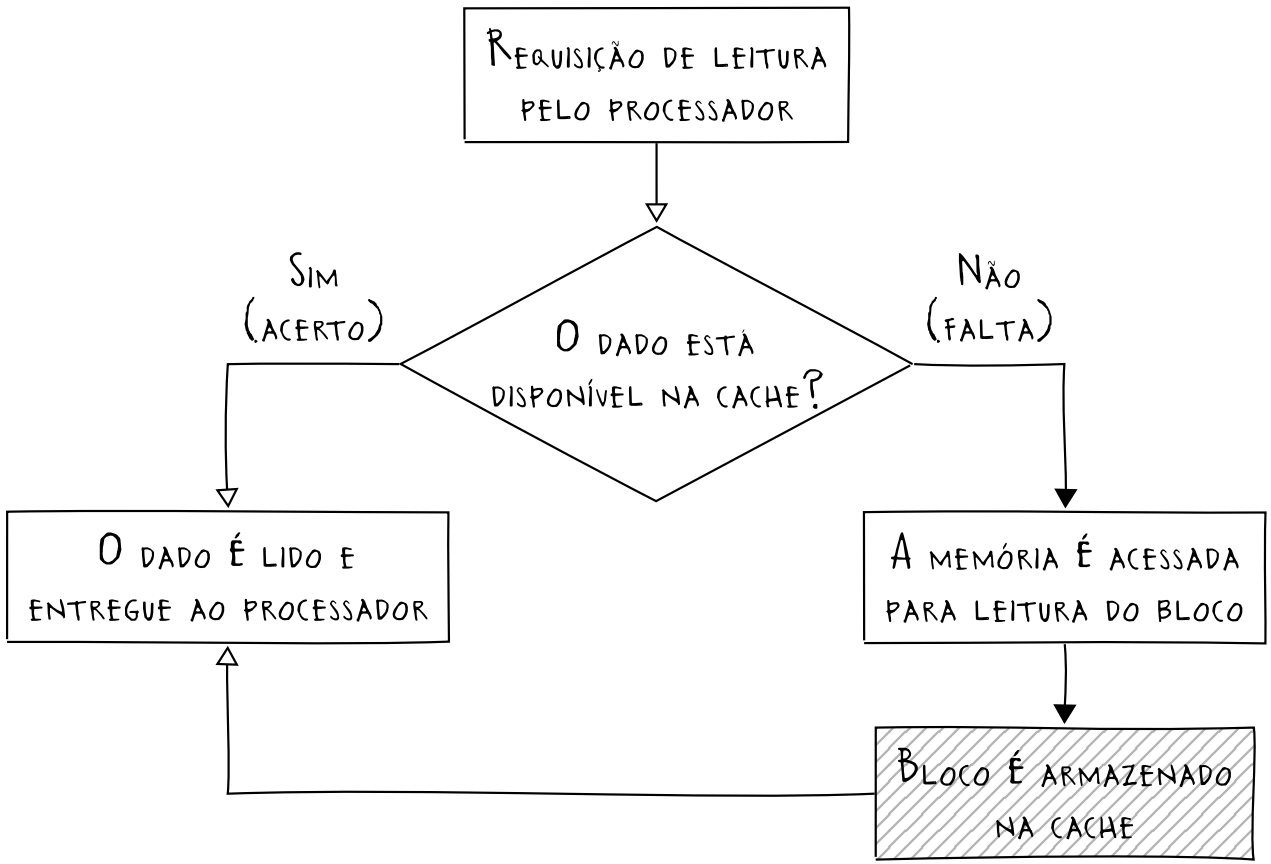
Memória cache

- 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



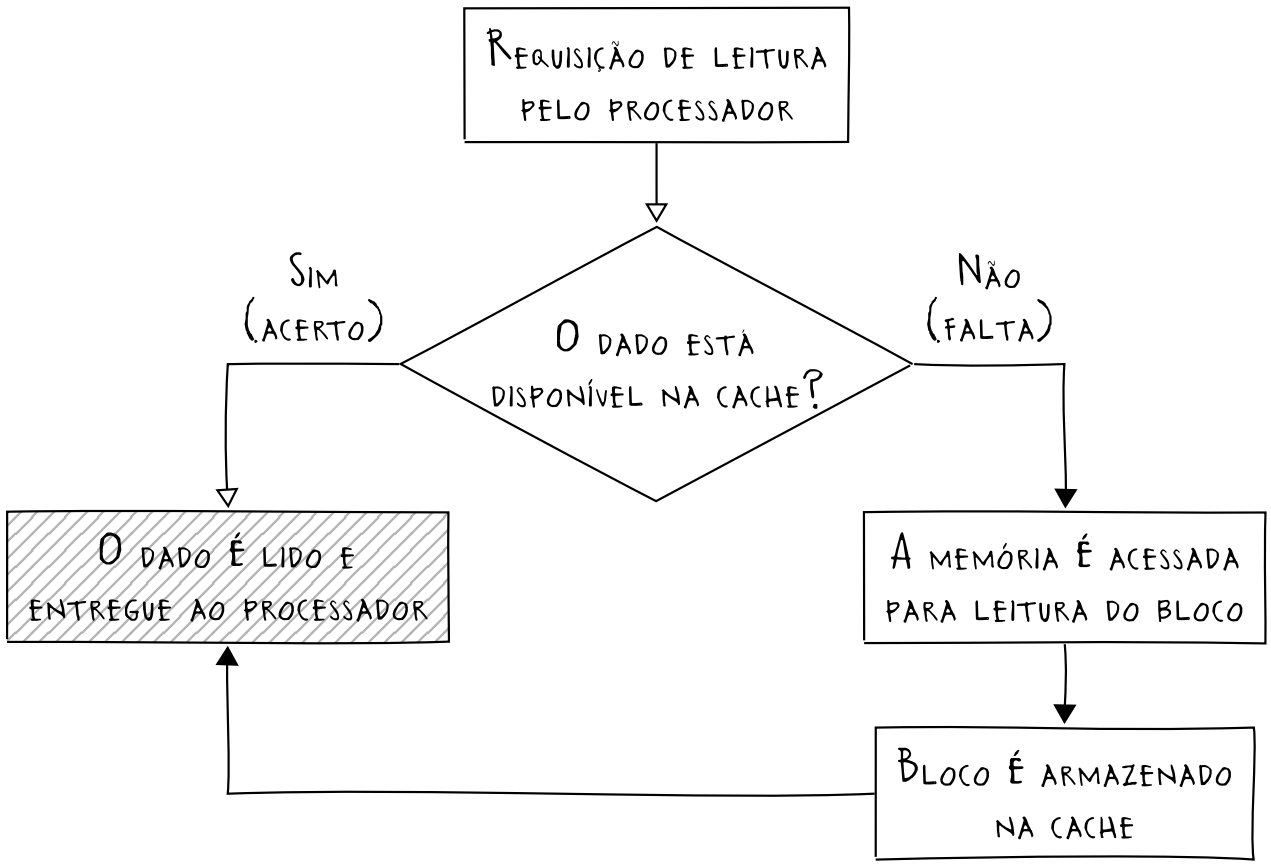
Memória cache

- 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



Memória 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



Memória 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

Memória 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

Memória 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

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

Memória 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

Memória 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

Memória 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

Memória cache

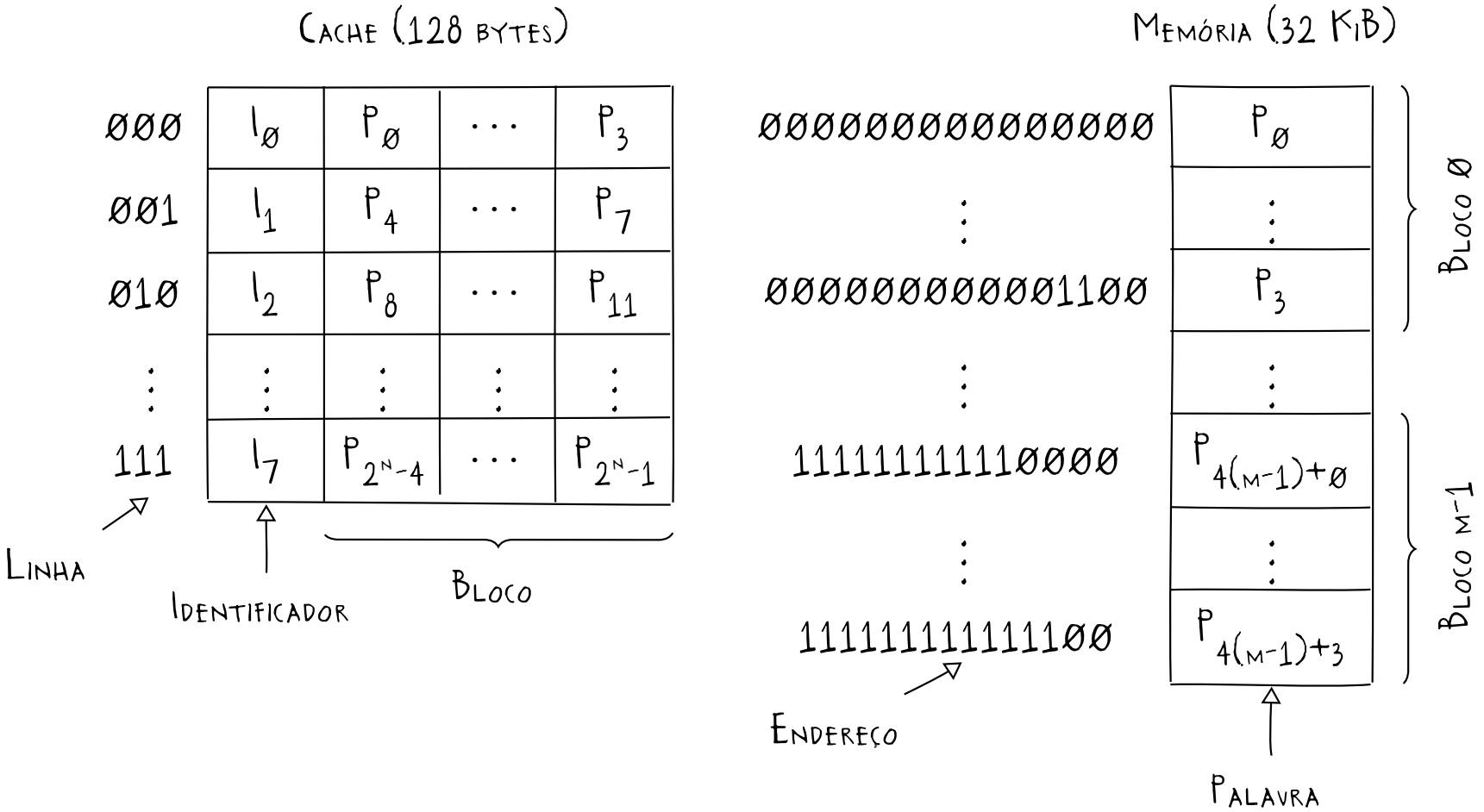
- Mapeamento direto
  - É a técnica mais simples para associação de endereços na cache, consistindo na aplicação de uma função de módulo (*hash*) para determinar qual linha deve ser indexada

$$i = j \bmod L$$

  - 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

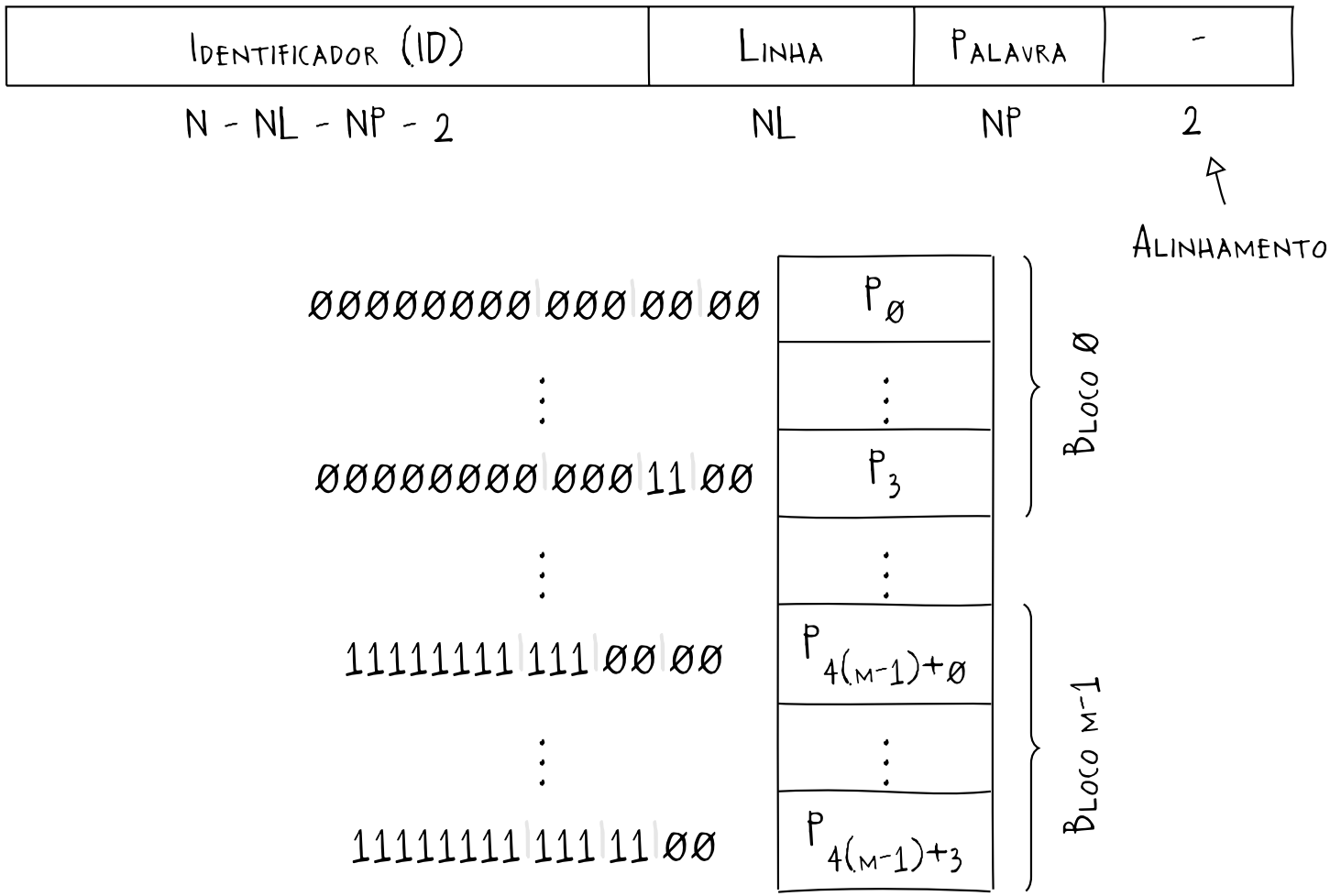
Memória cache

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



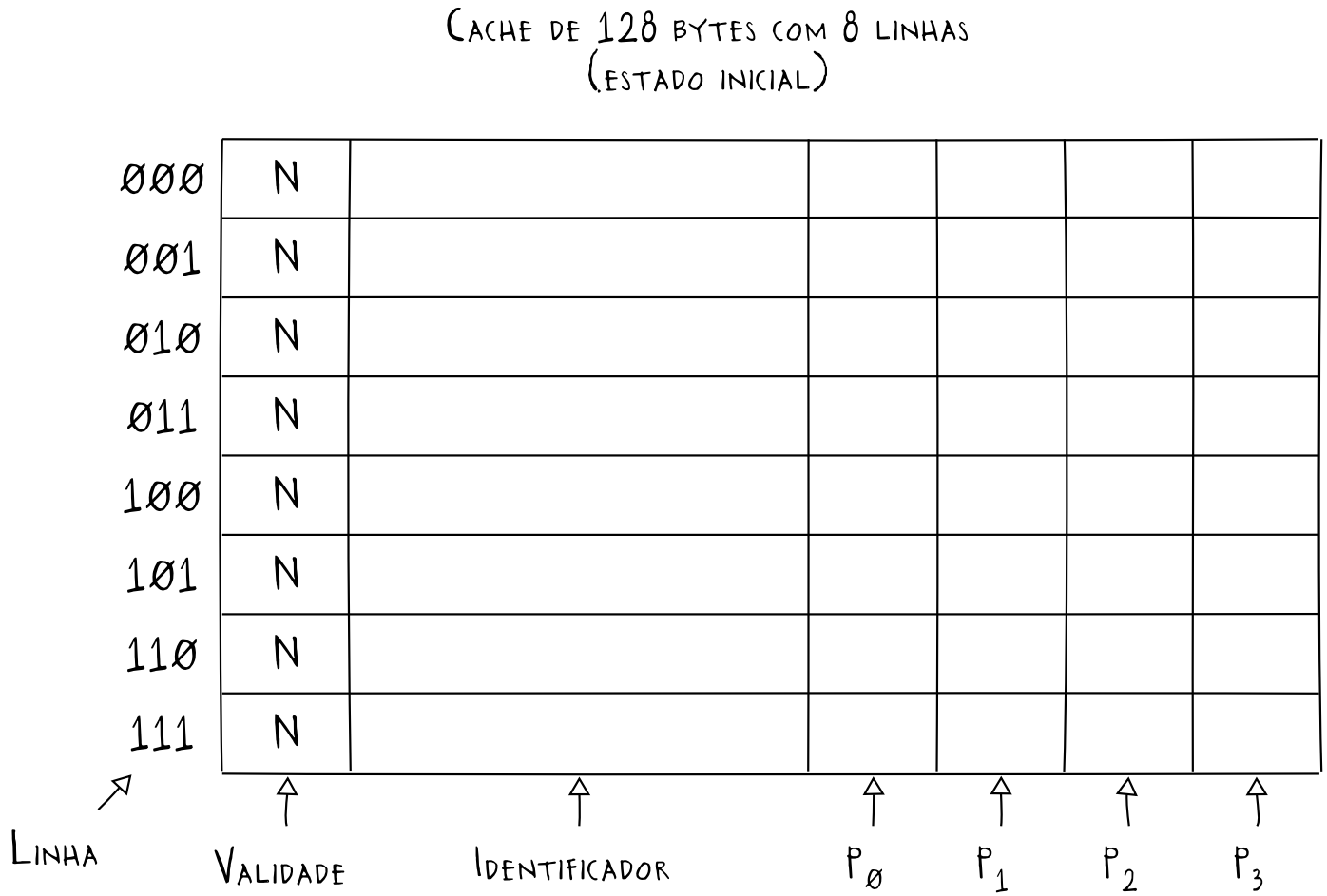
Memória cache

- 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



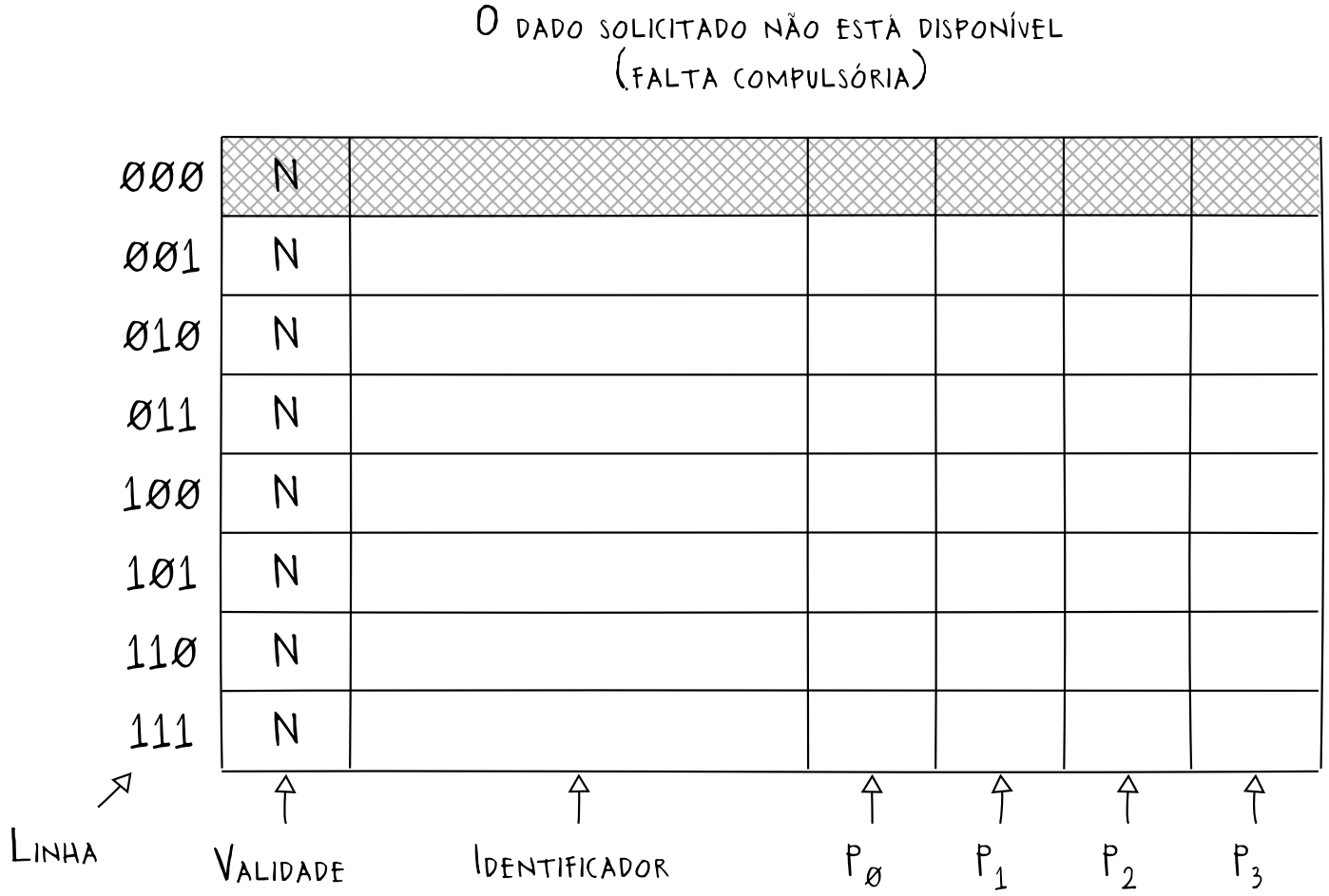
Memória cache

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



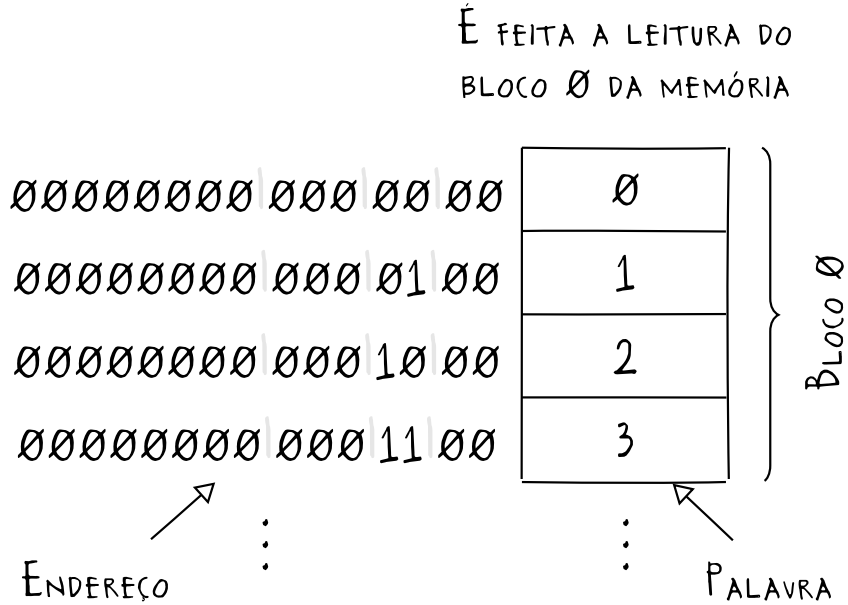
Memória cache

- Mapeamento direto
  - Solicitação do endereço 00000000|000|00|00
  - Identificador = 00000000, Linha = 000 e Palavra = 00



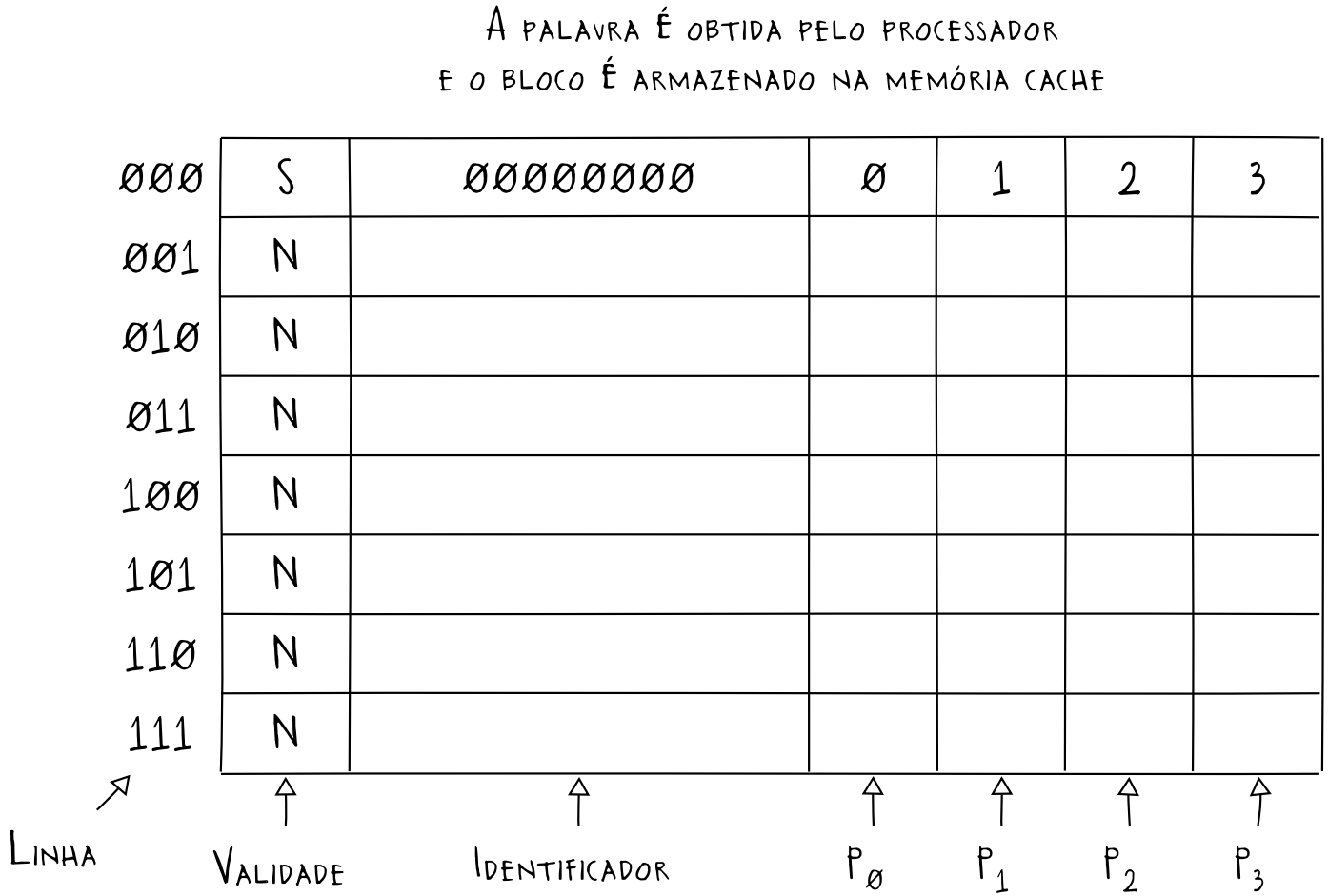
Memória cache

- Mapeamento direto
  - Solicitação do endereço 00000000|000|00|00
  - Identificador = 00000000, Linha = 000 e Palavra = 00



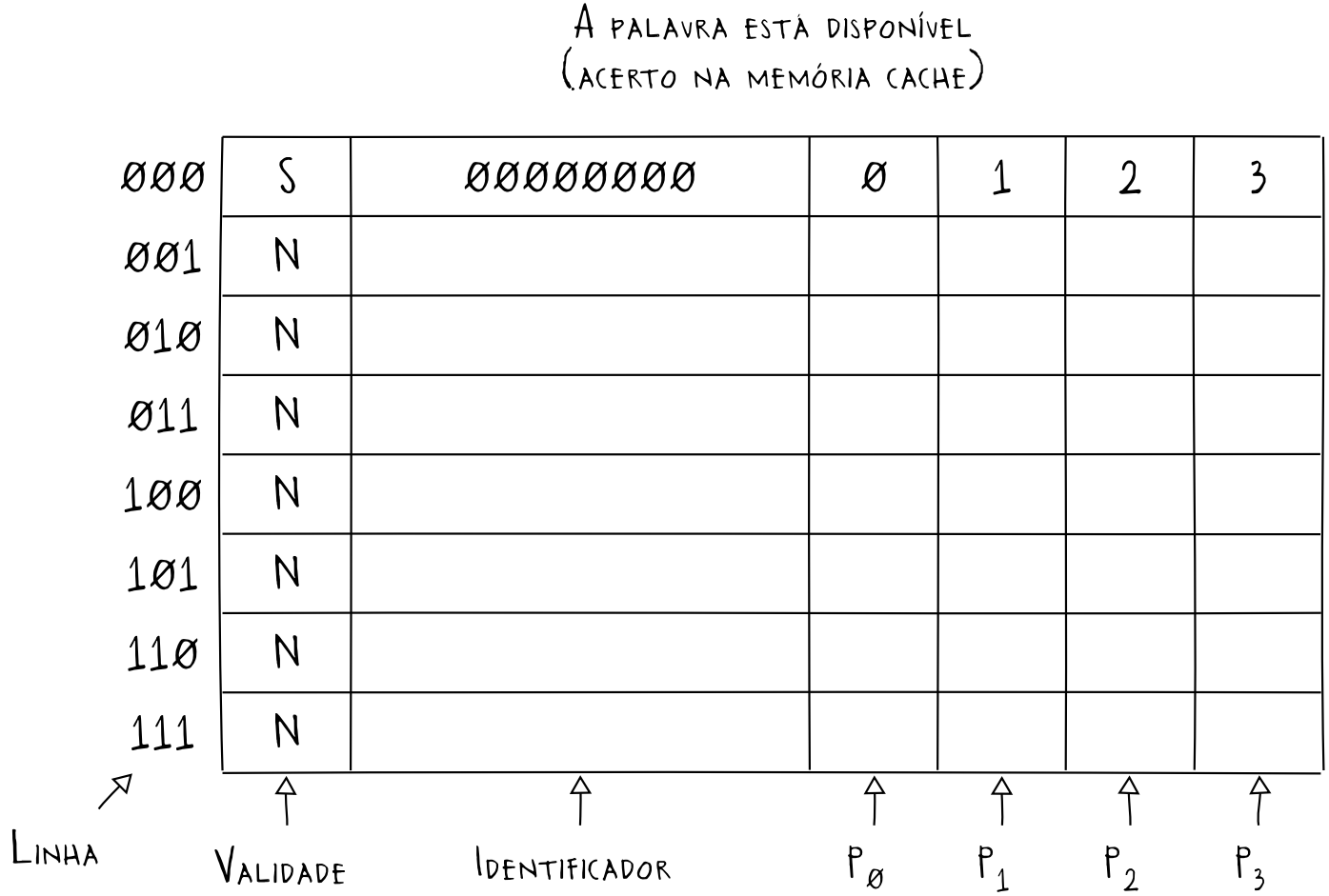
Memória cache

- Mapeamento direto
  - Solicitação do endereço 00000000|000|00|00
  - Identificador = 00000000, Linha = 000 e Palavra = 00



Memória cache

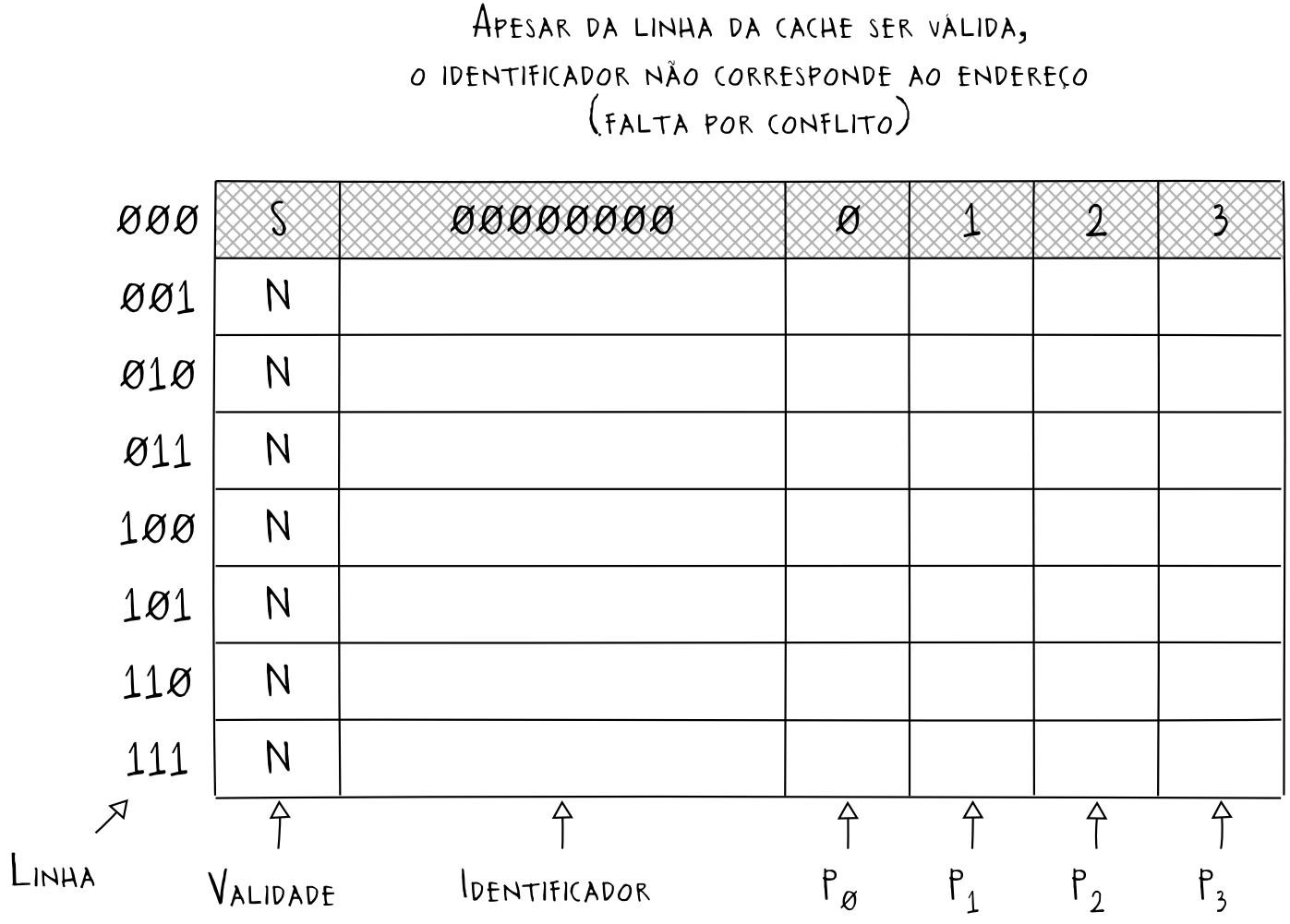
- Mapeamento direto
  - Solicitação do endereço 00000000|000|11|00
  - Identificador = 00000000, Linha = 000 e Palavra = 11



LINHA	VALIDADE	IDENTIFICADOR	P0	P1	P2	P3
000	S	00000000	0	1	2	3
001	N					
010	N					
011	N					
100	N					
101	N					
110	N					
111	N					

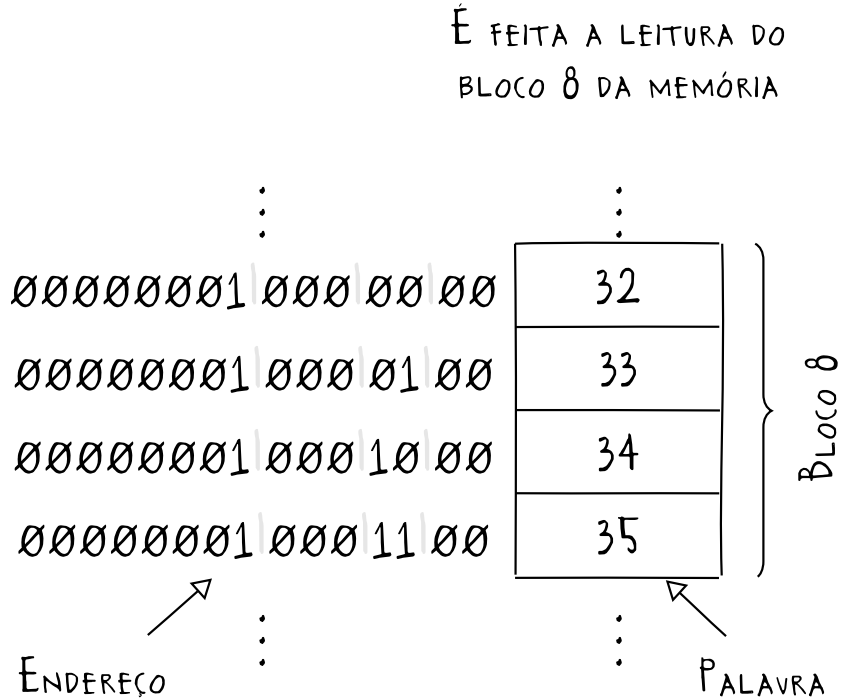
Memória cache

- Mapeamento direto
  - Solicitação do endereço 00000001|000|01|00
  - Identificador = 00000001, Linha = 000 e Palavra = 01



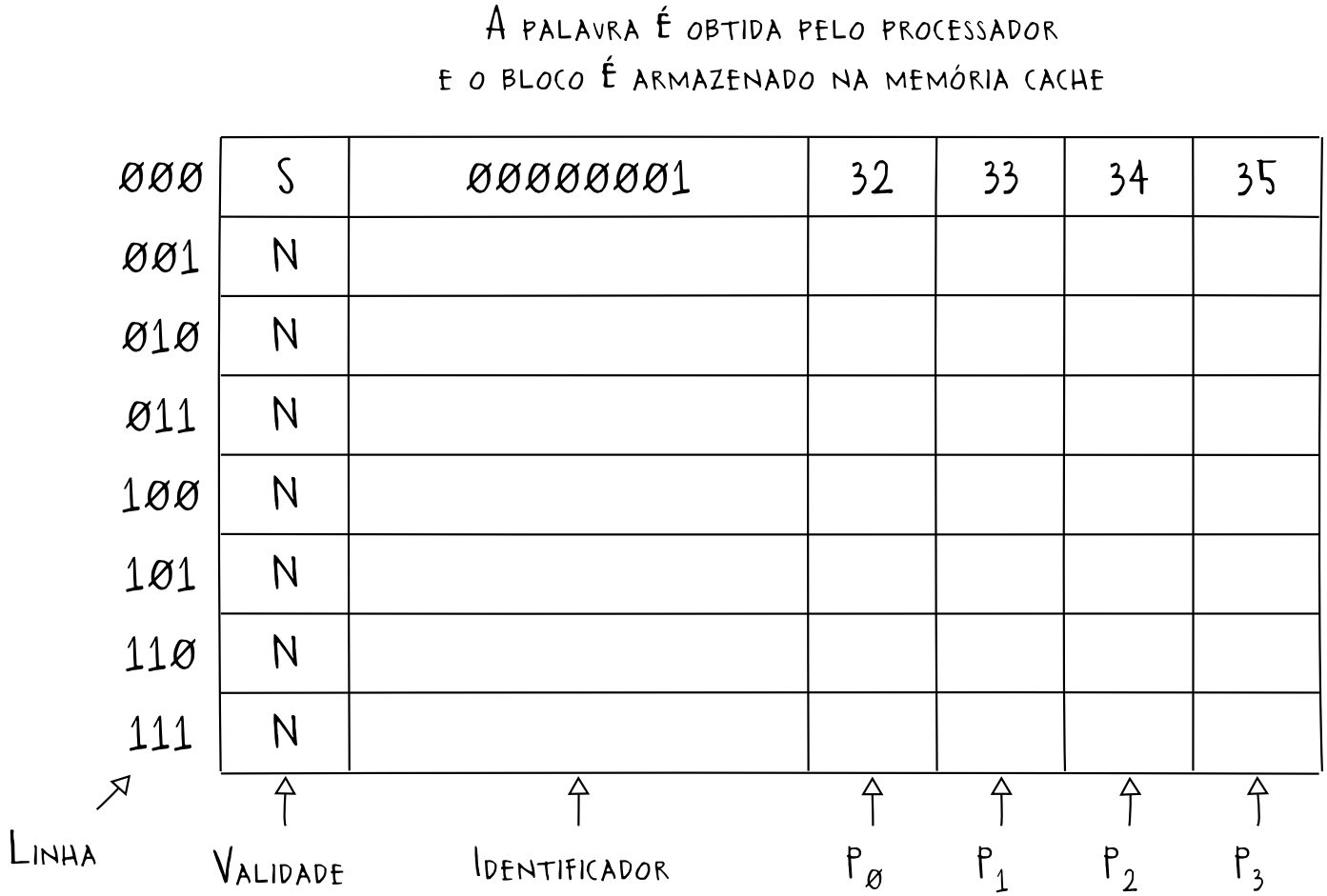
Memória cache

- Mapeamento direto
  - Solicitação do endereço 00000001|000|01|00
  - Identificador = 00000001, Linha = 000 e Palavra = 01



Memória cache

- Mapeamento direto
  - Solicitação do endereço 00000001|000|01|00
  - Identificador = 00000001, Linha = 000 e Palavra = 01



Memória 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

Linha	Validade	Identificador	P_0	P_1	P_2	P_3
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

Memória cache

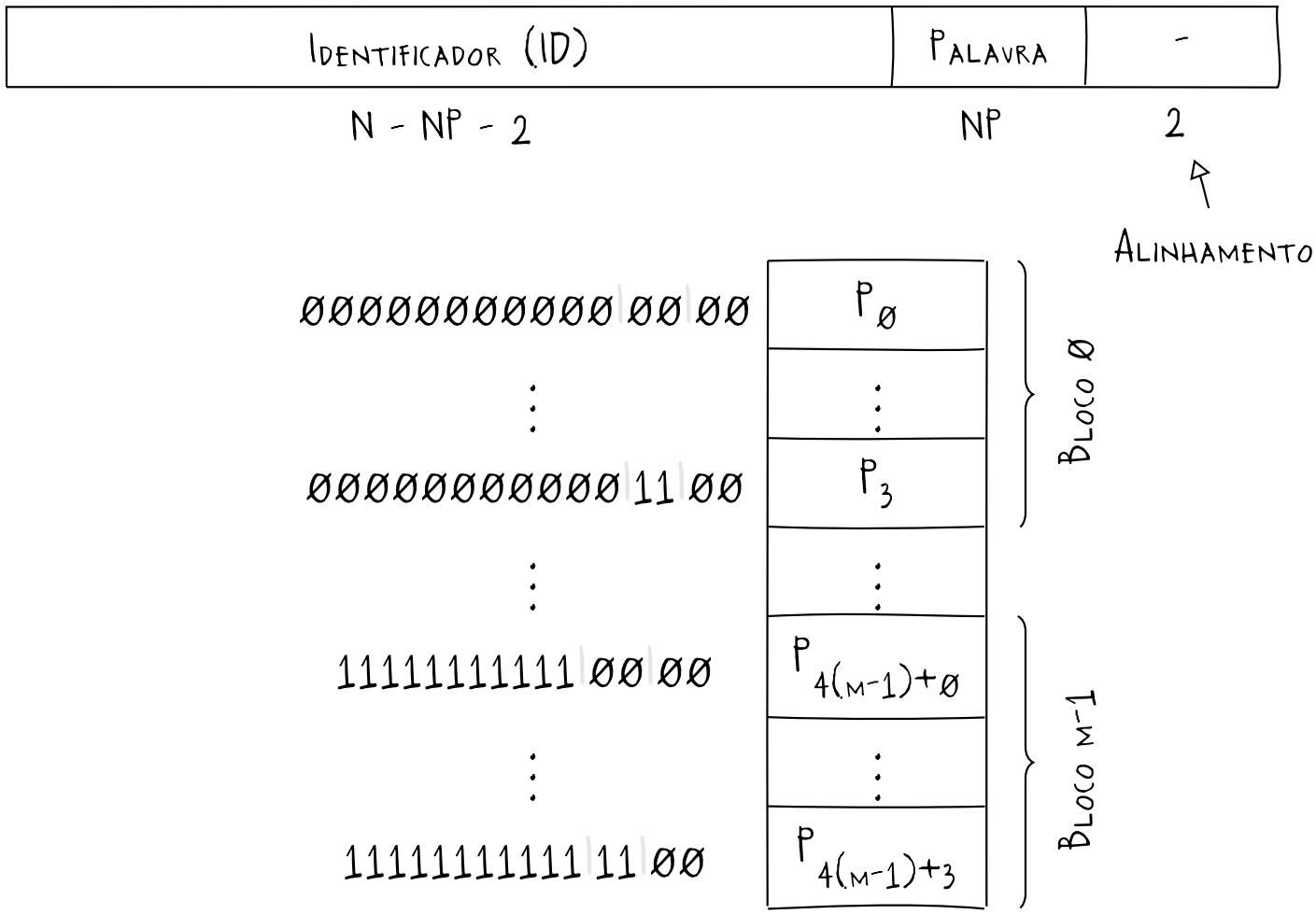
- Mapeamento direto
  - ✓ Implementação de baixo custo e simples
  - ✓ A indexação depende somente da seleção de bits do endereço de memória solicitado

Memória 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

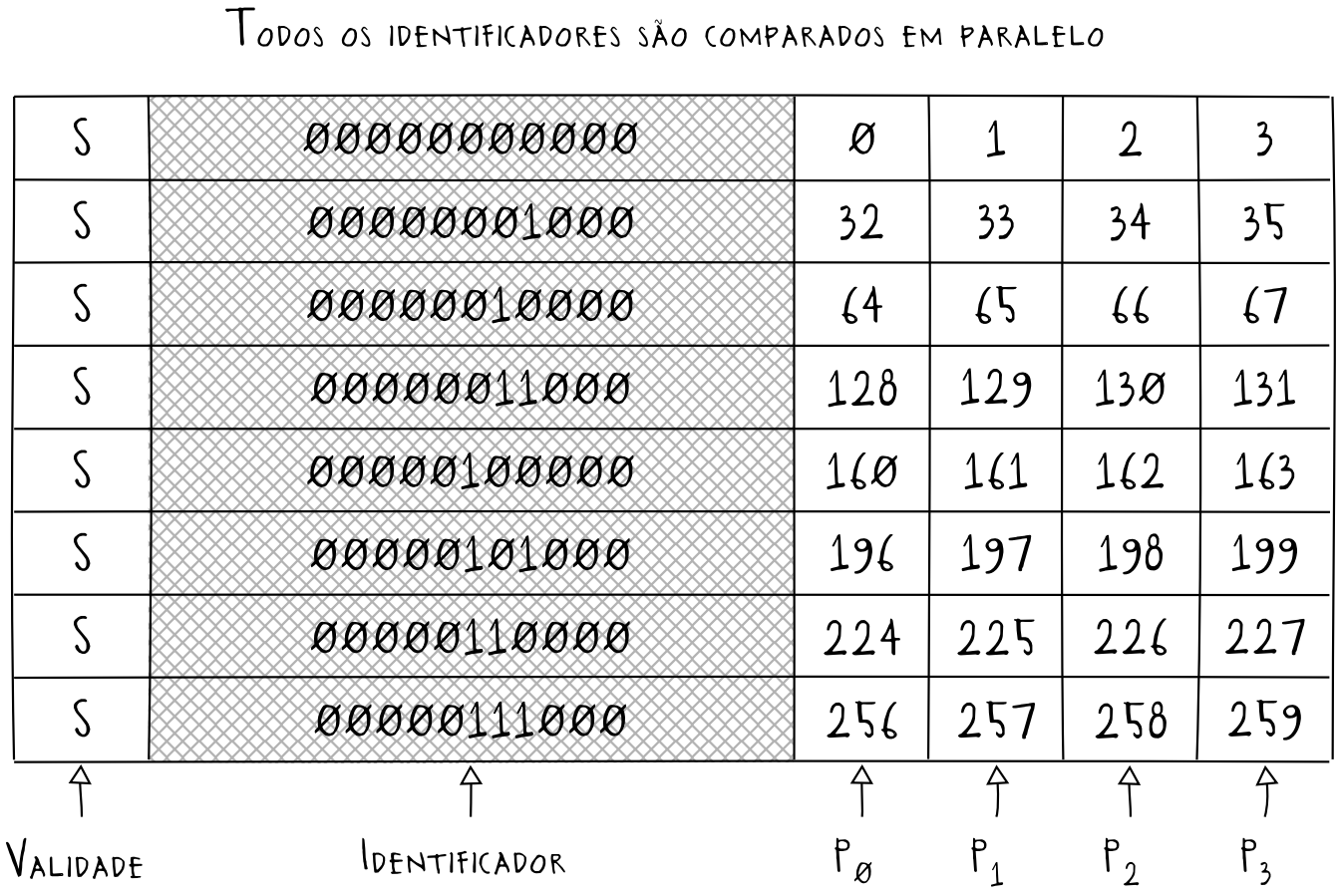
Memória cache

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



Memória cache

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



Memória cache

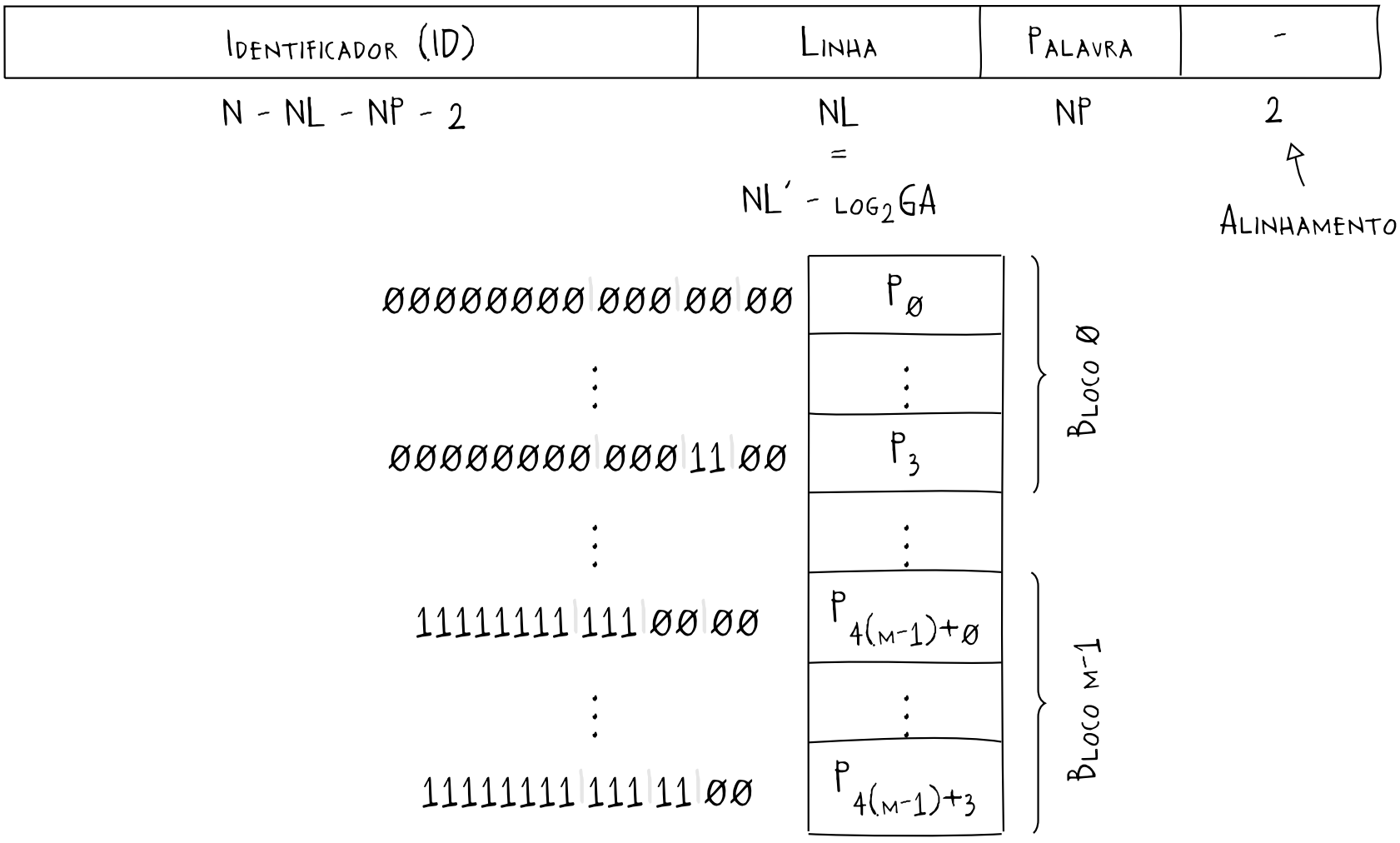
- Mapeamento totalmente associativo
  - ✓ Não existe conflito de endereçamento, porque um bloco pode ser associado a qualquer linha da cache

Memória cache

- Mapeamento totalmente associativo
  - ✓ Não existe conflito de endereçamento, porque um bloco pode ser associado a qualquer linha da cache
  - ✗ Alto custo de comparação paralela de todos os identificadores armazenados nas linhas

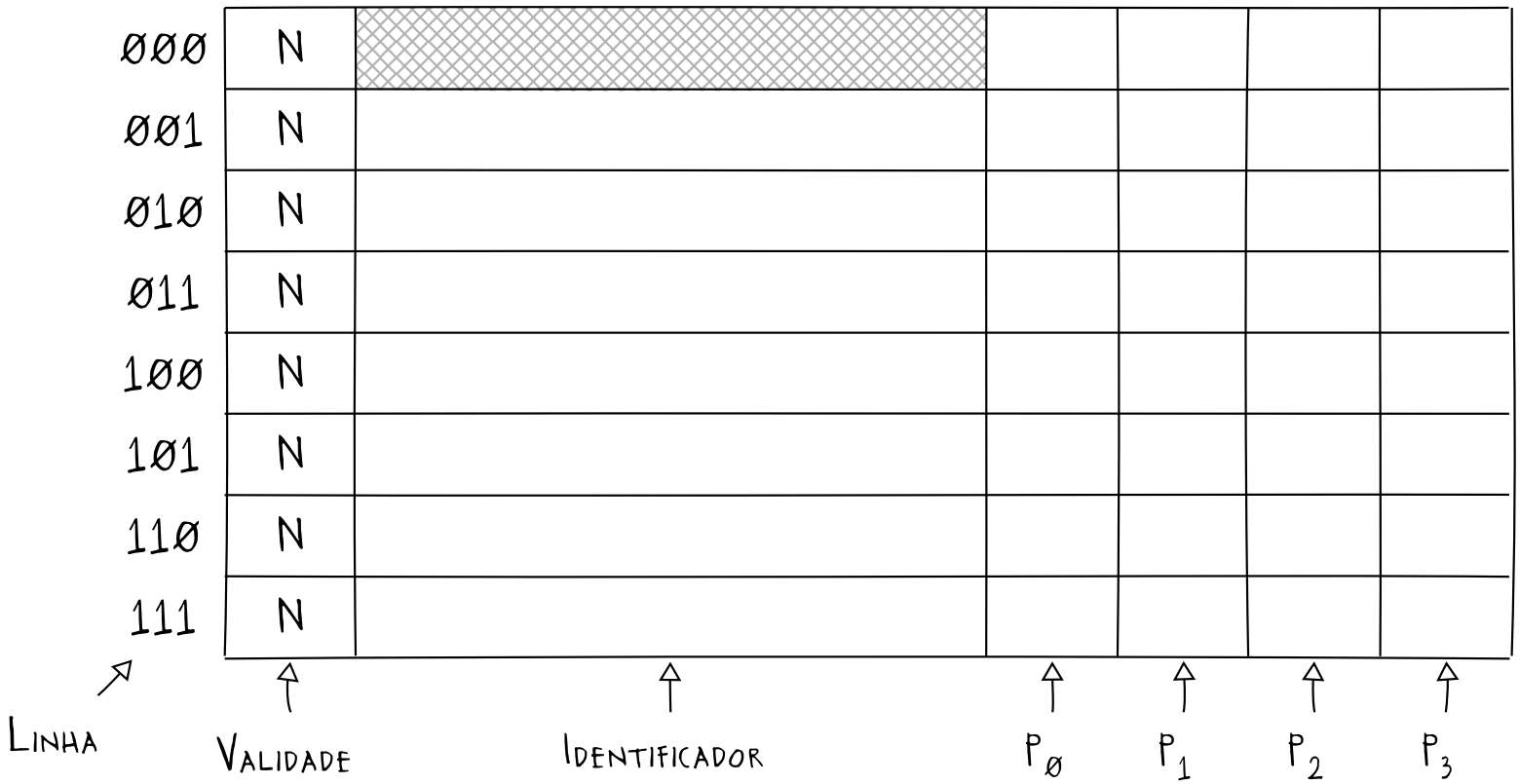
Memória cache

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



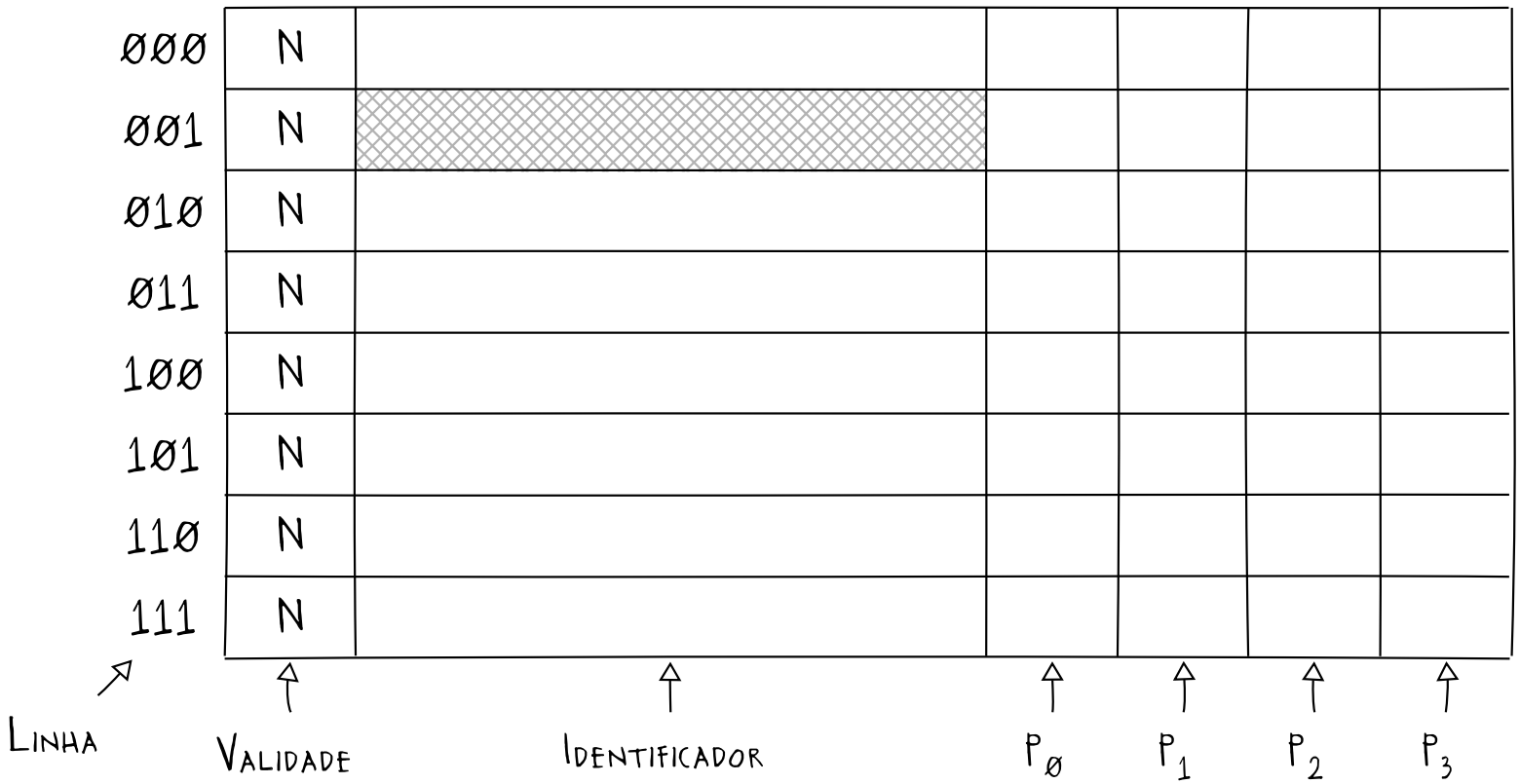
Memória cache

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



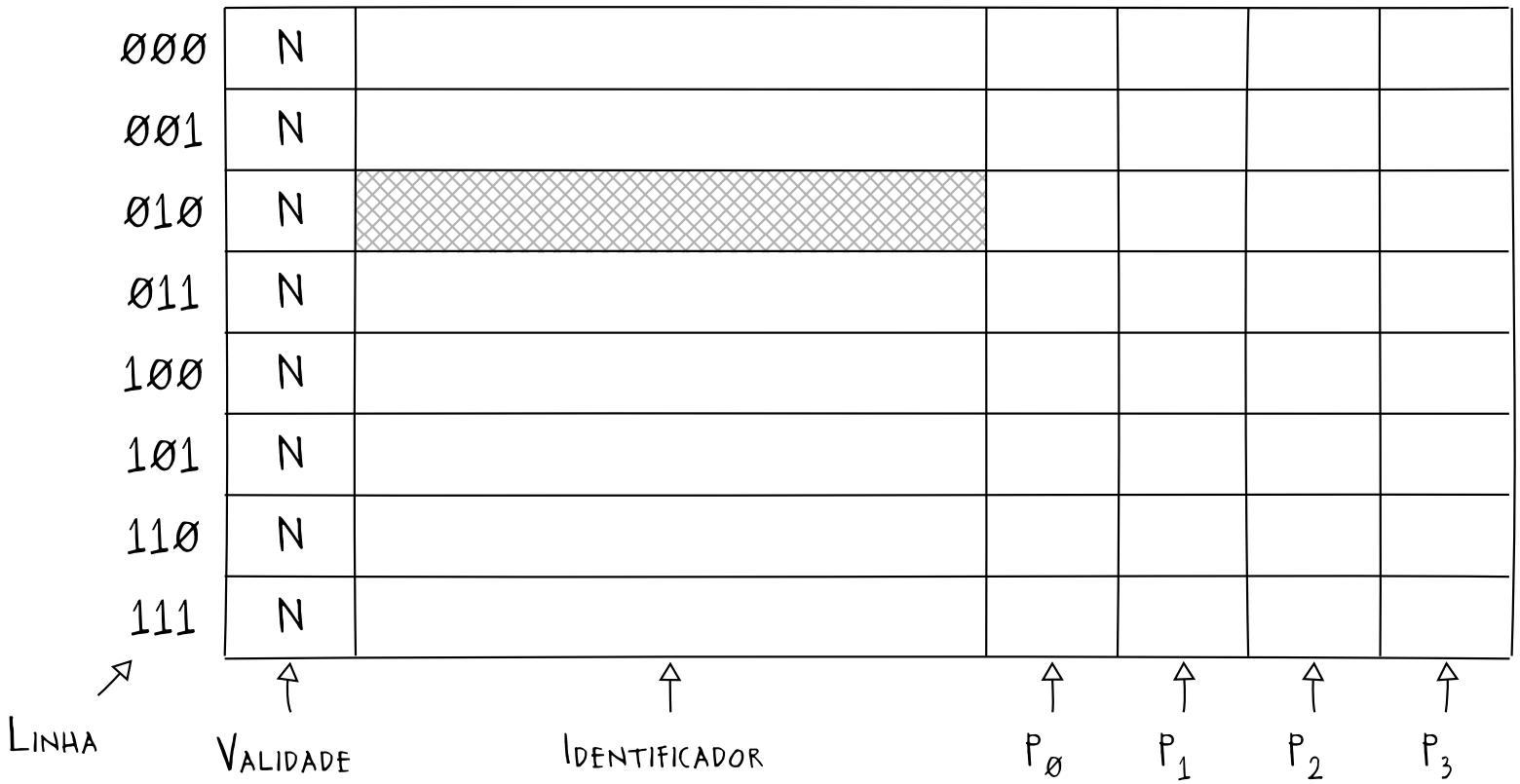
Memória cache

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



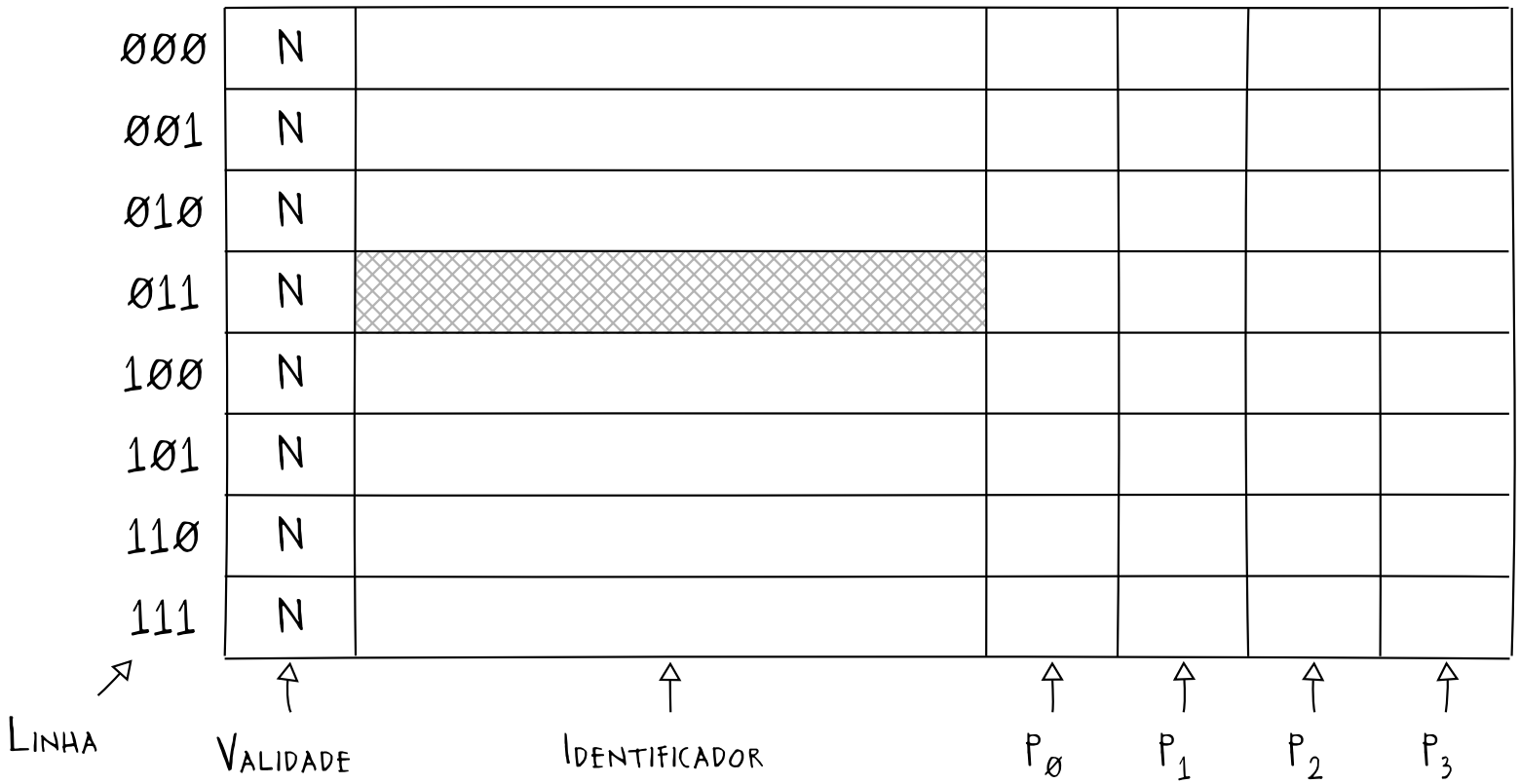
Memória cache

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



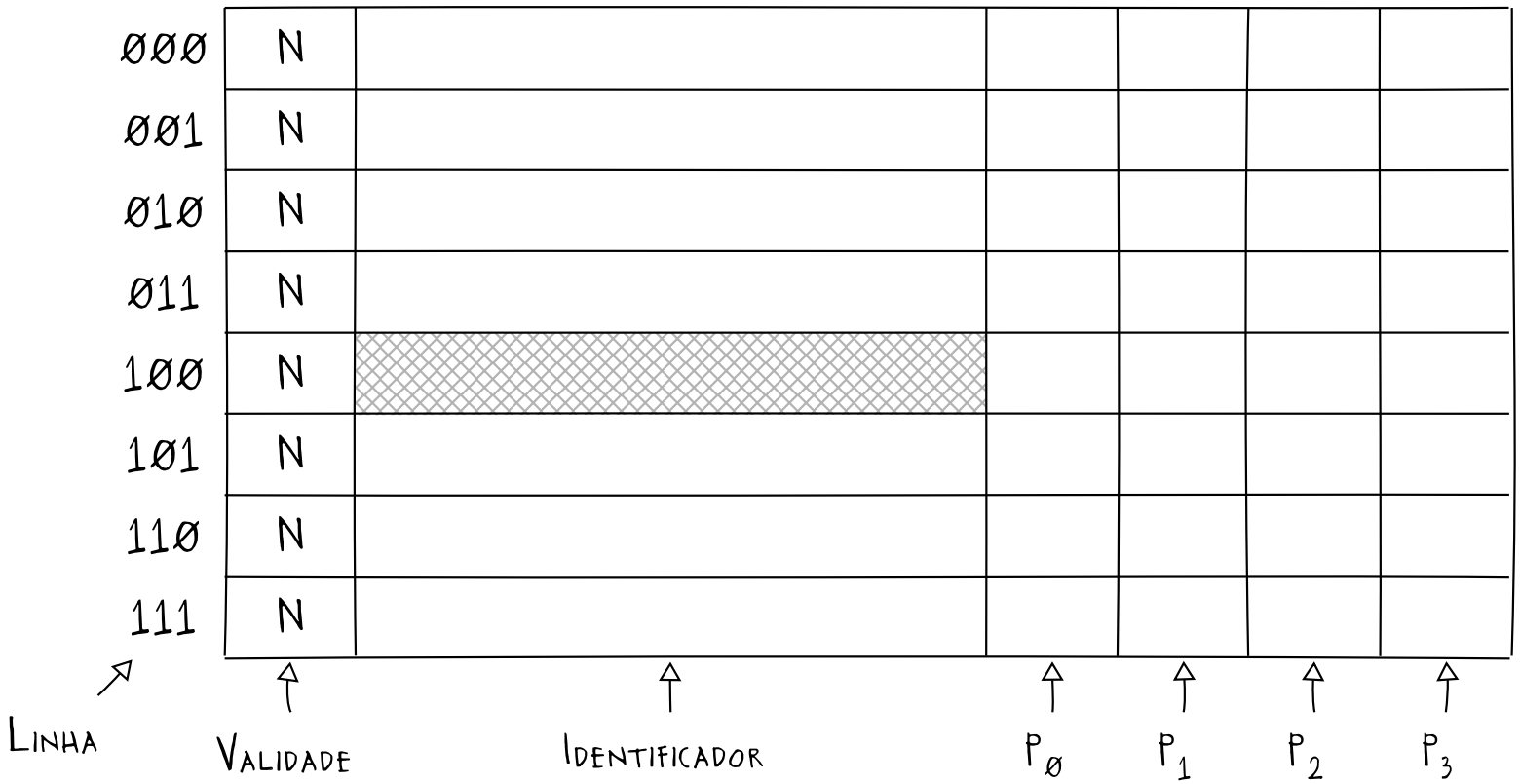
Memória cache

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



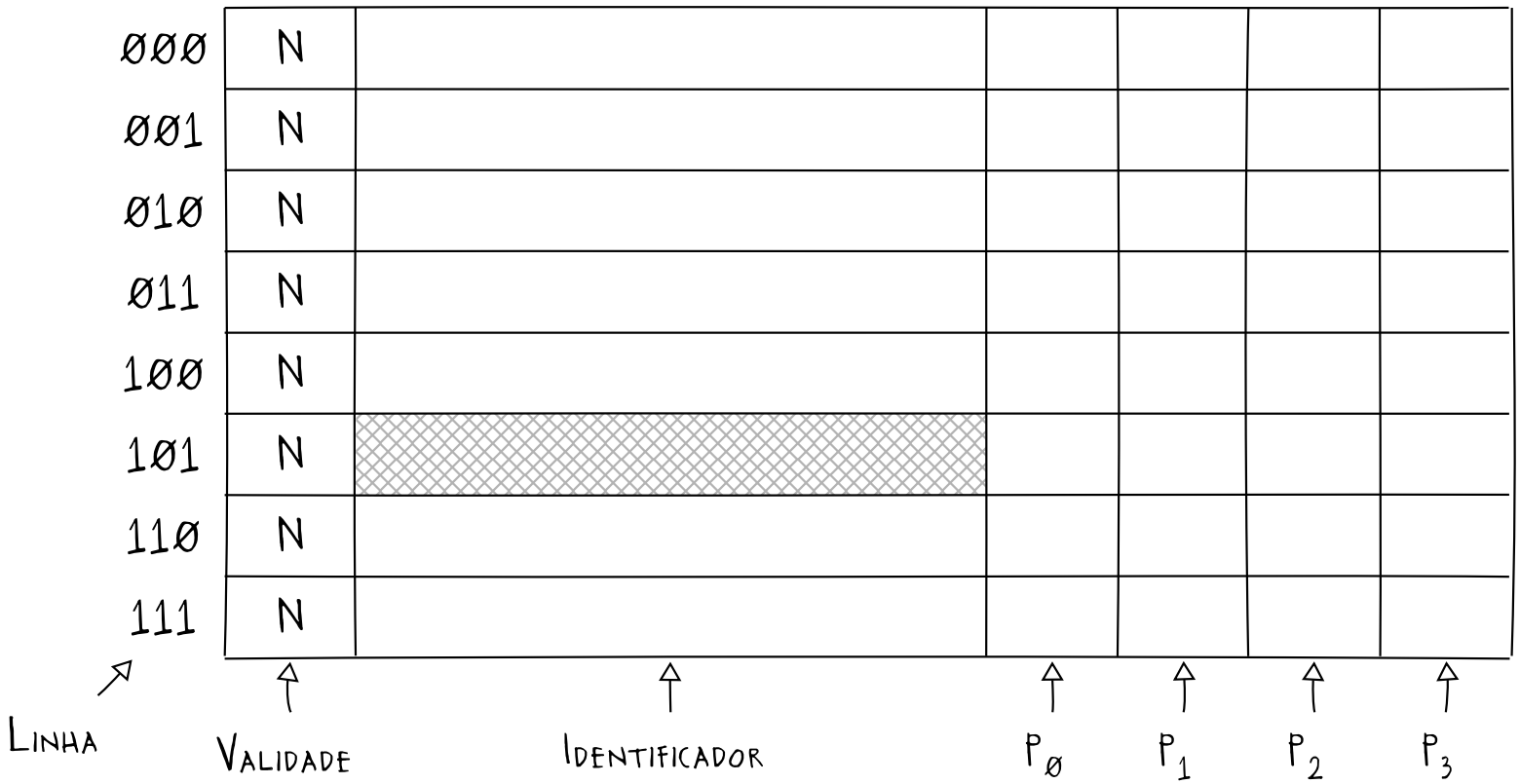
Memória cache

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



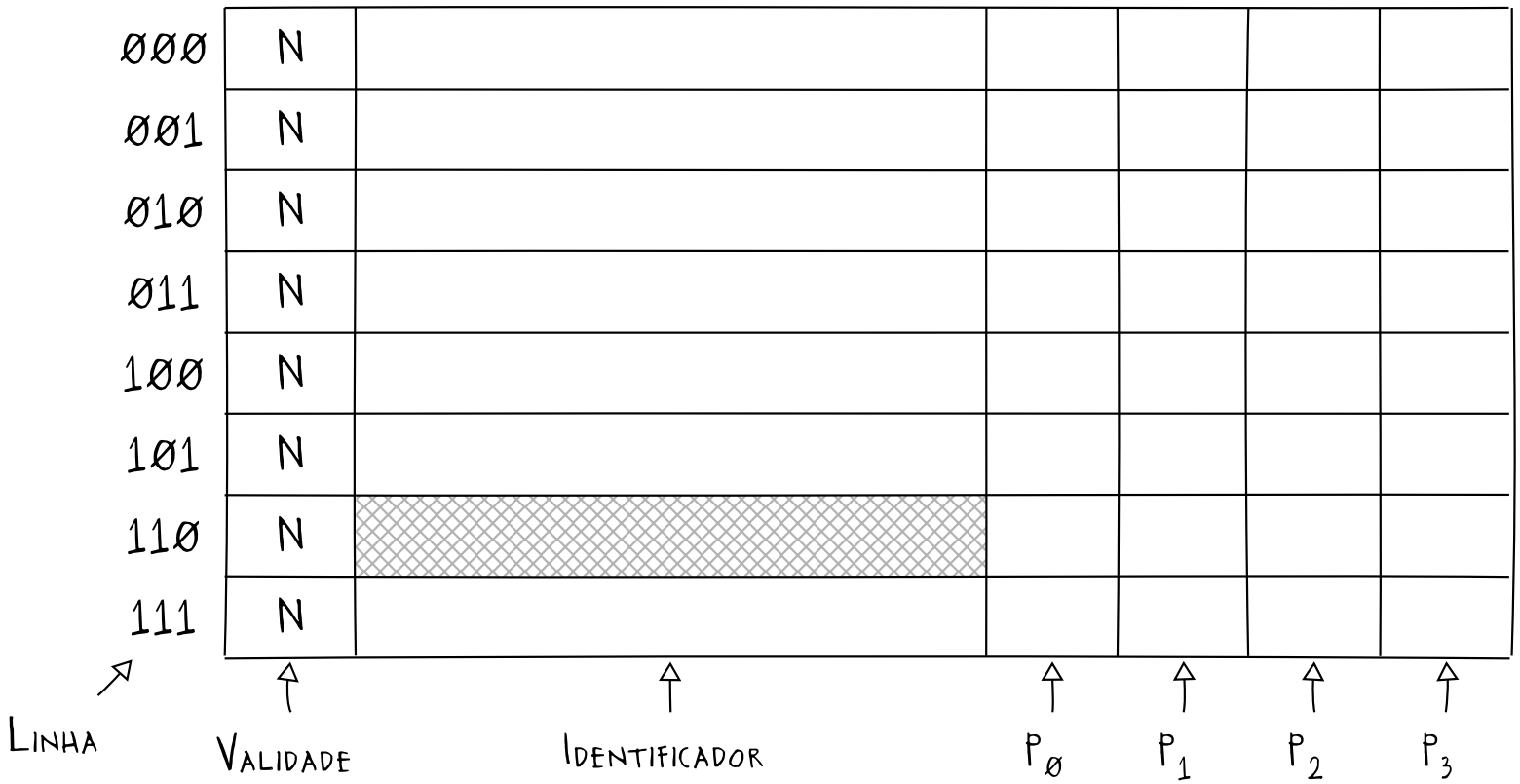
Memória cache

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



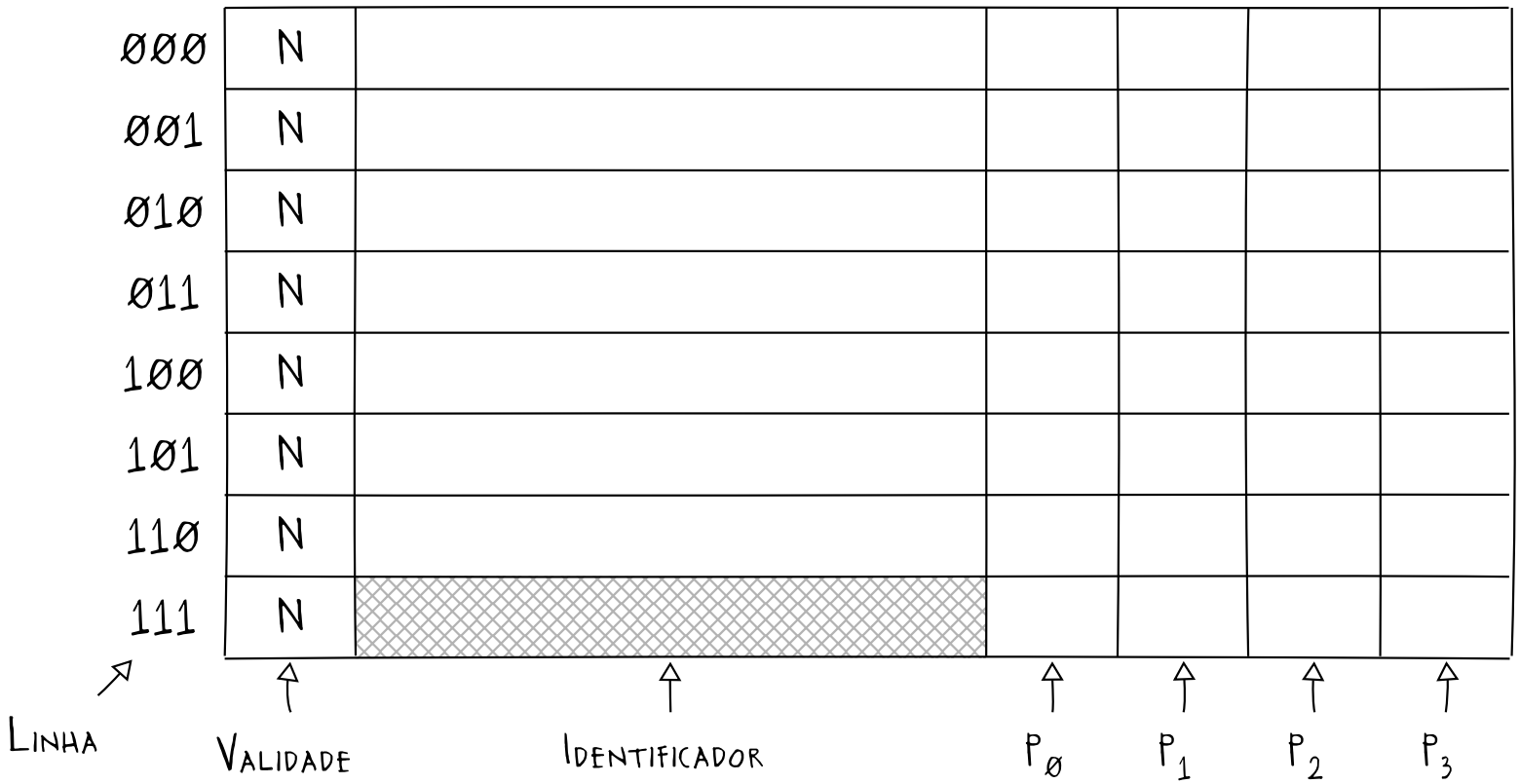
Memória cache

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



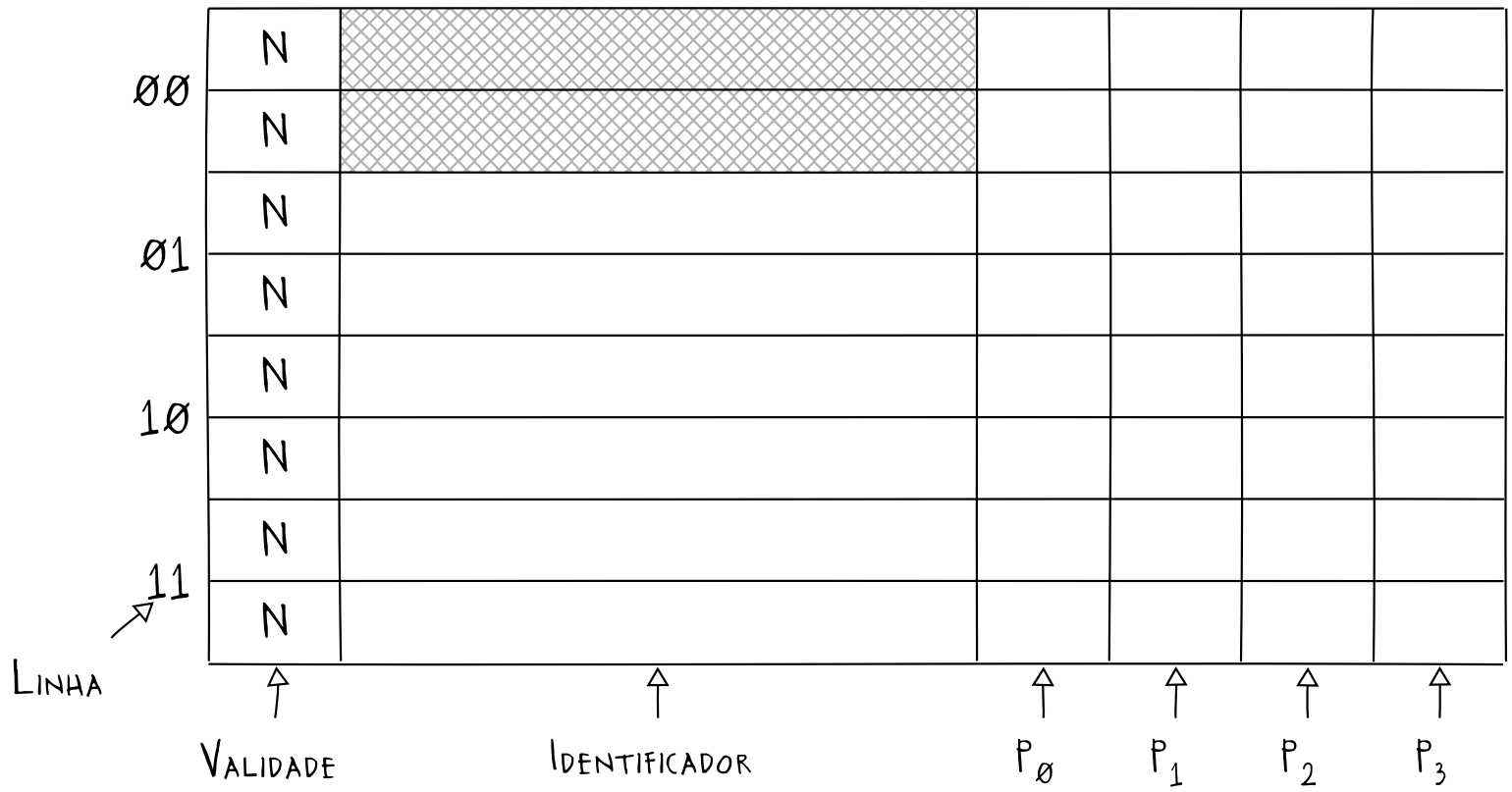
Memória cache

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



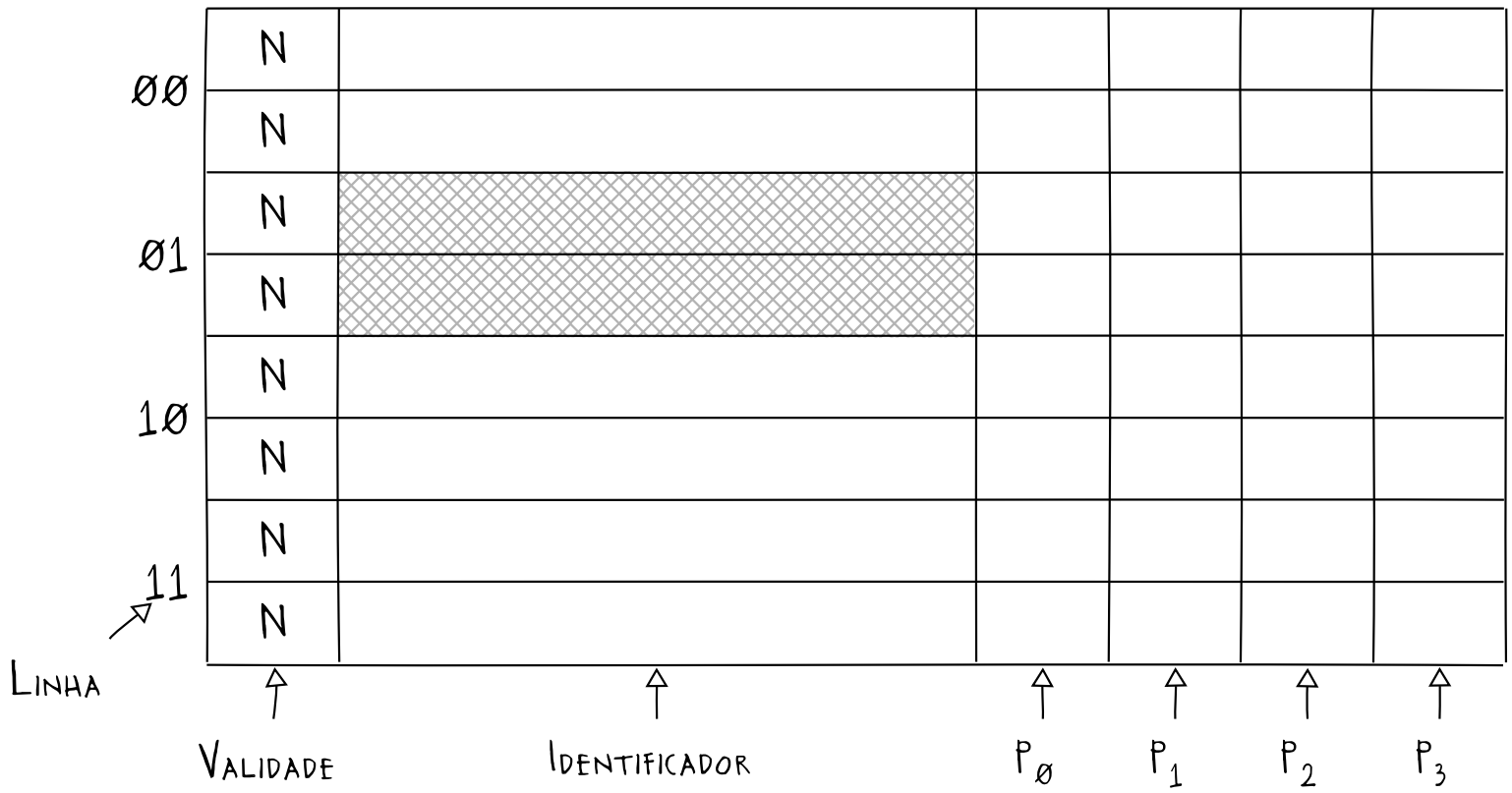
Memória cache

- Mapeamento associativo por conjunto
  - Grau de associatividade 2



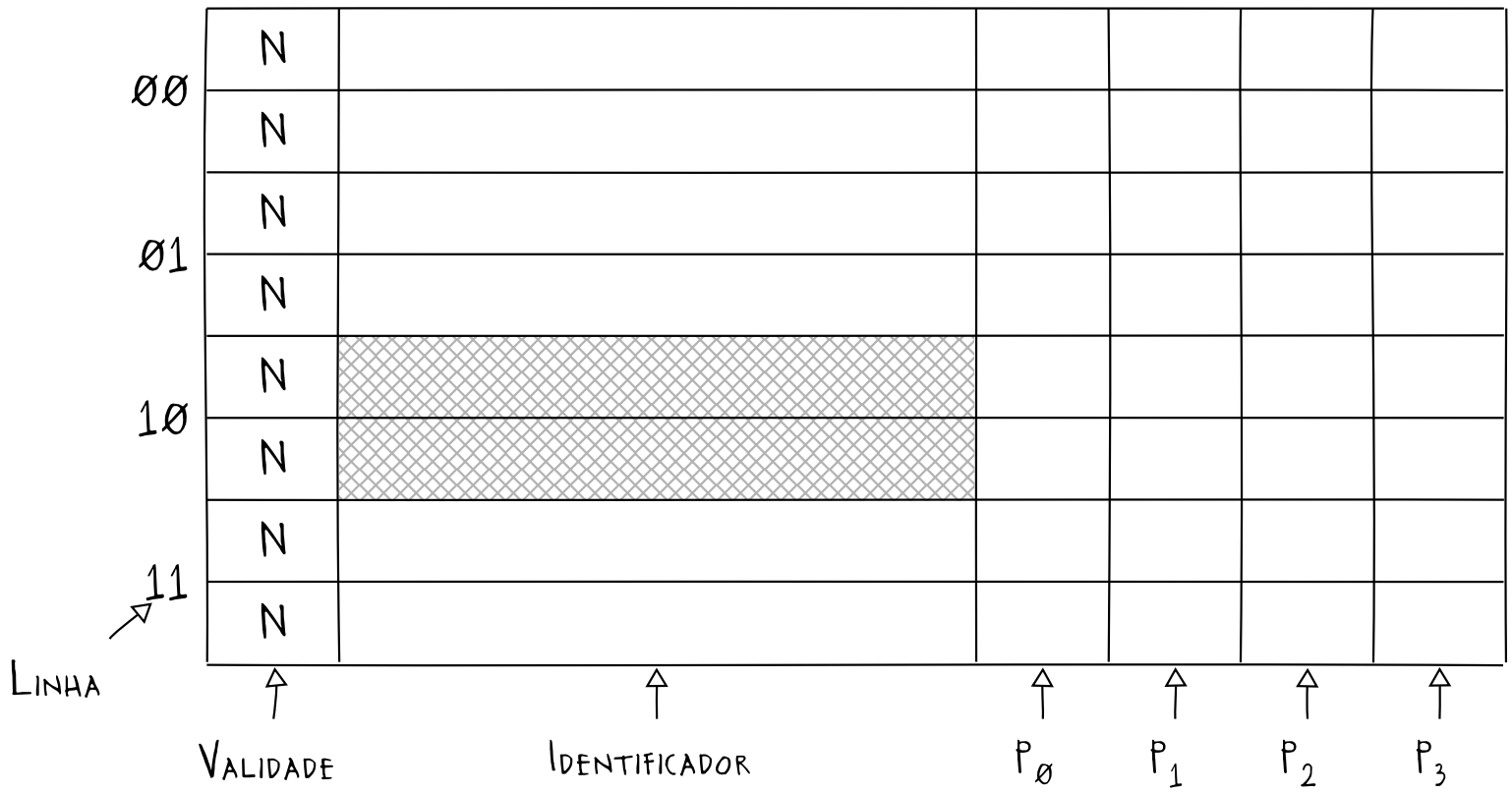
Memória cache

- Mapeamento associativo por conjunto
  - Grau de associatividade 2



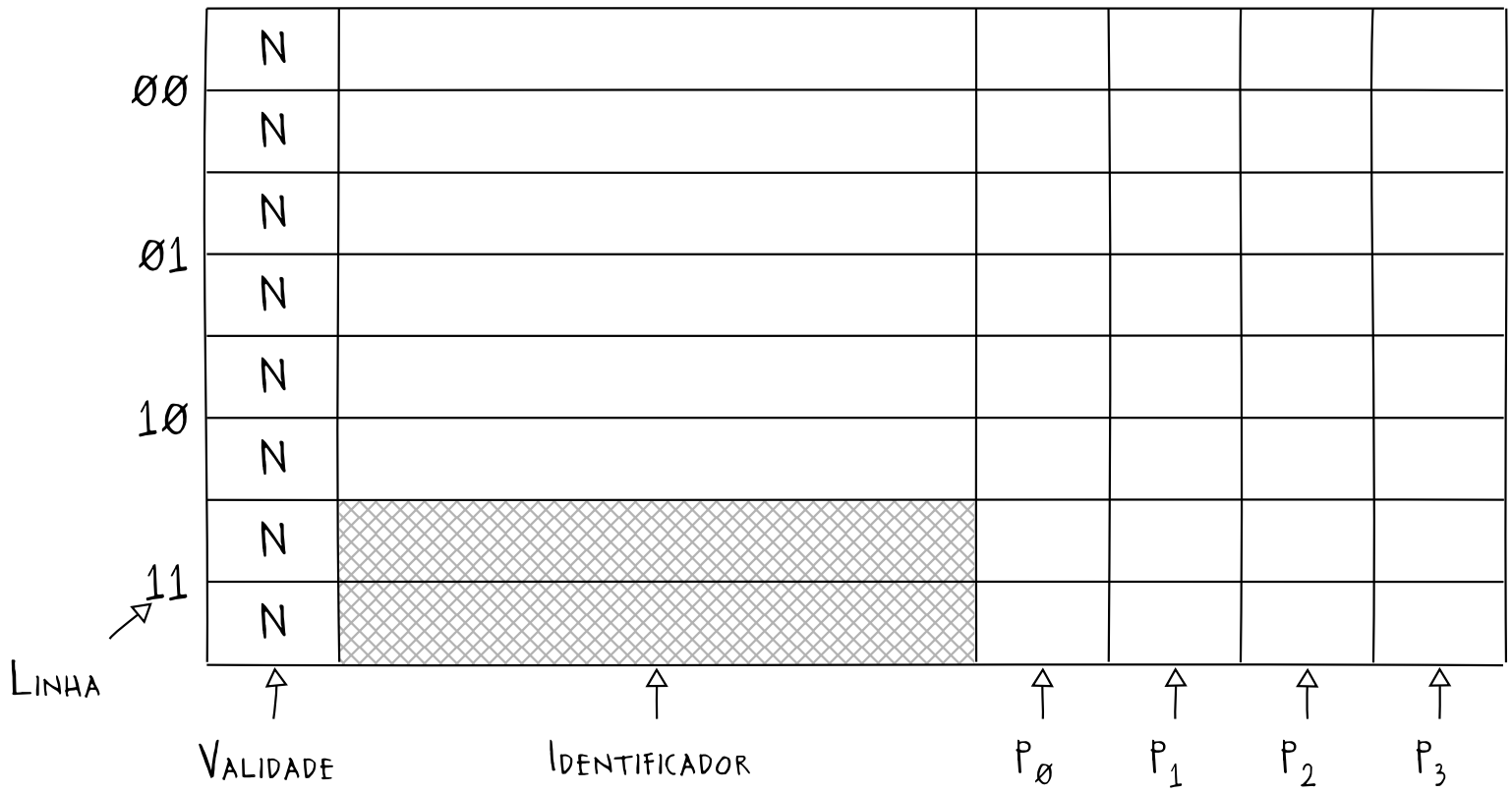
Memória cache

- Mapeamento associativo por conjunto
  - Grau de associatividade 2



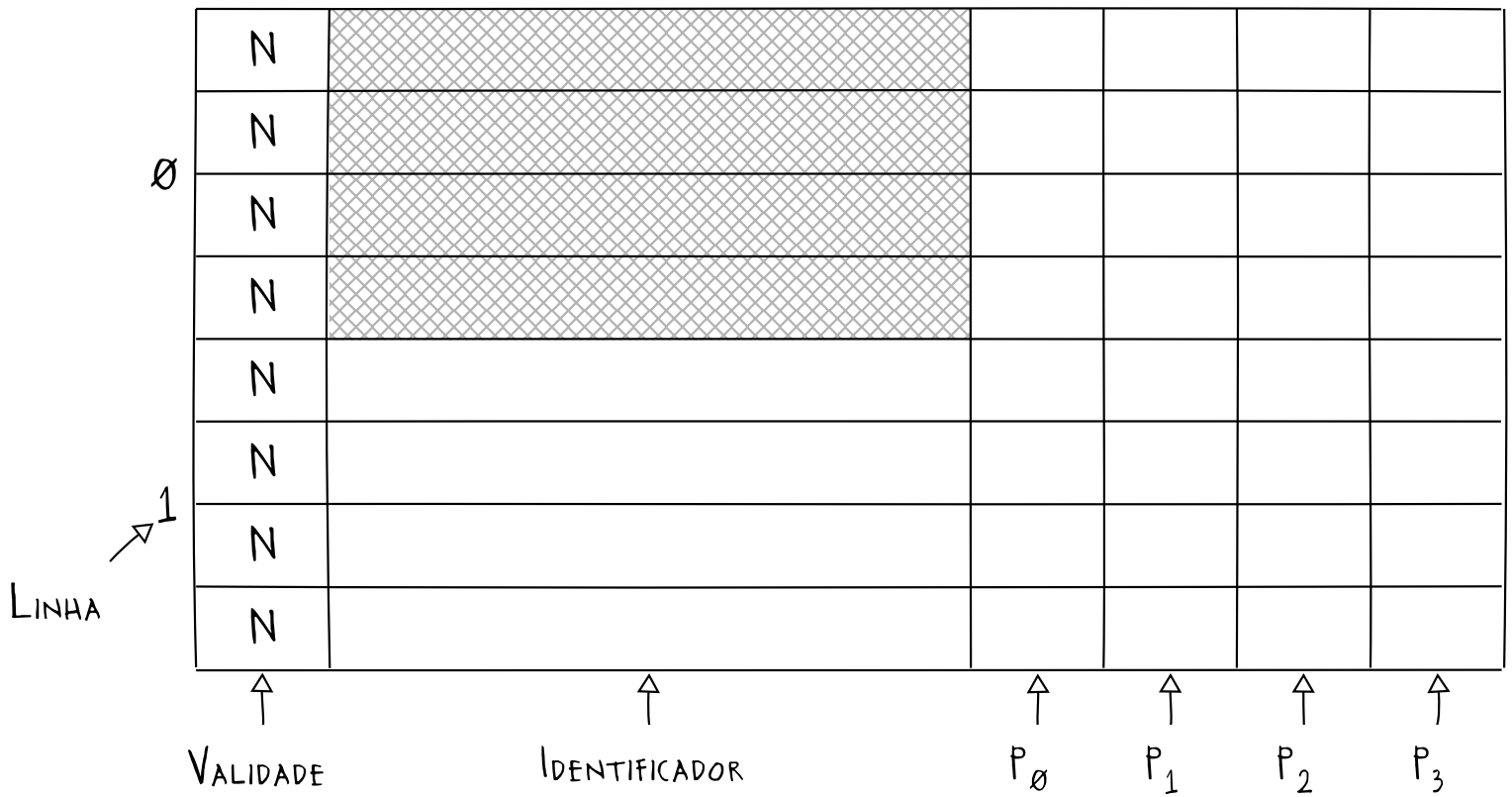
Memória cache

- Mapeamento associativo por conjunto
  - Grau de associatividade 2



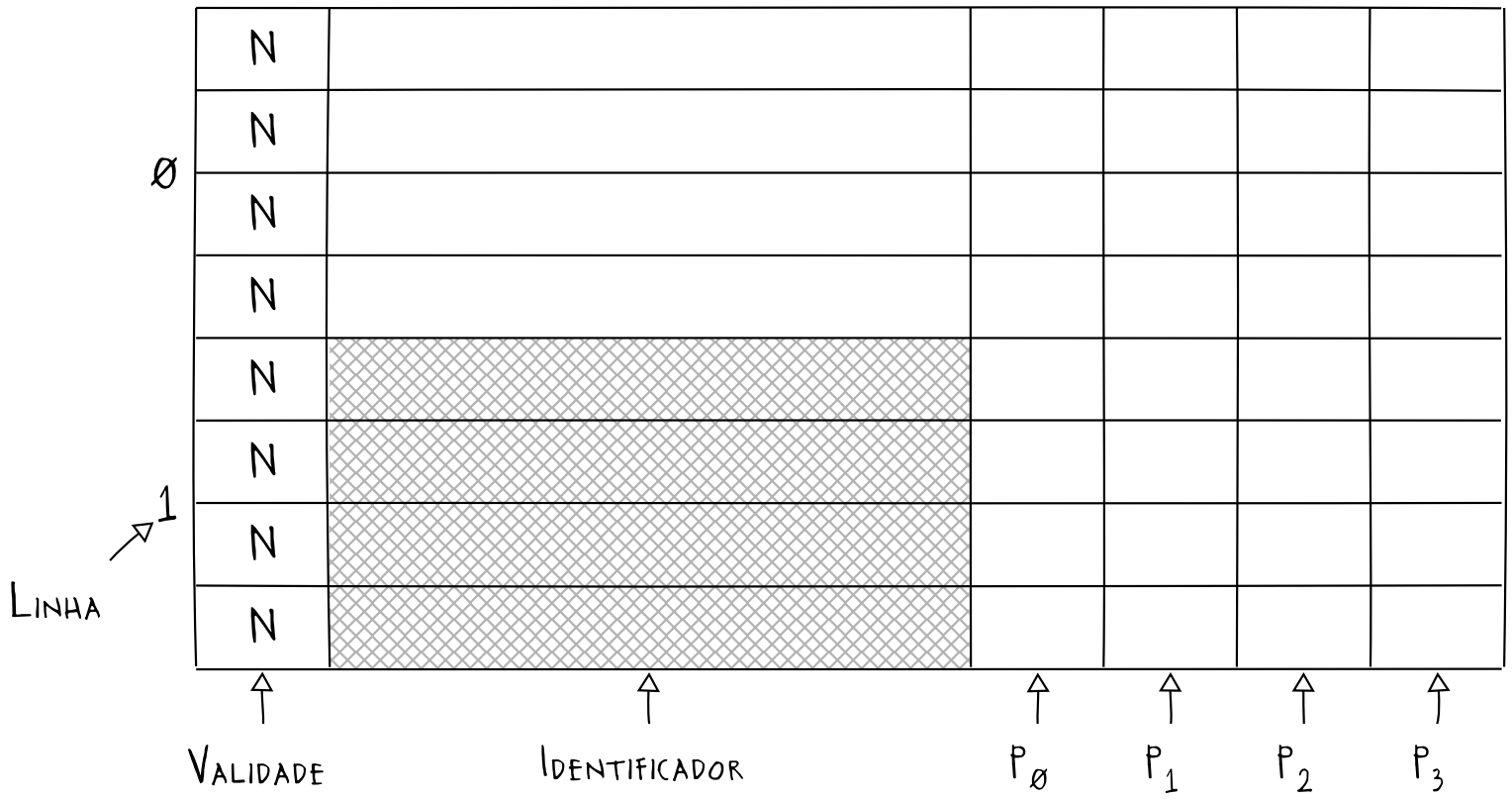
Memória cache

- Mapeamento associativo por conjunto
  - Grau de associatividade 4



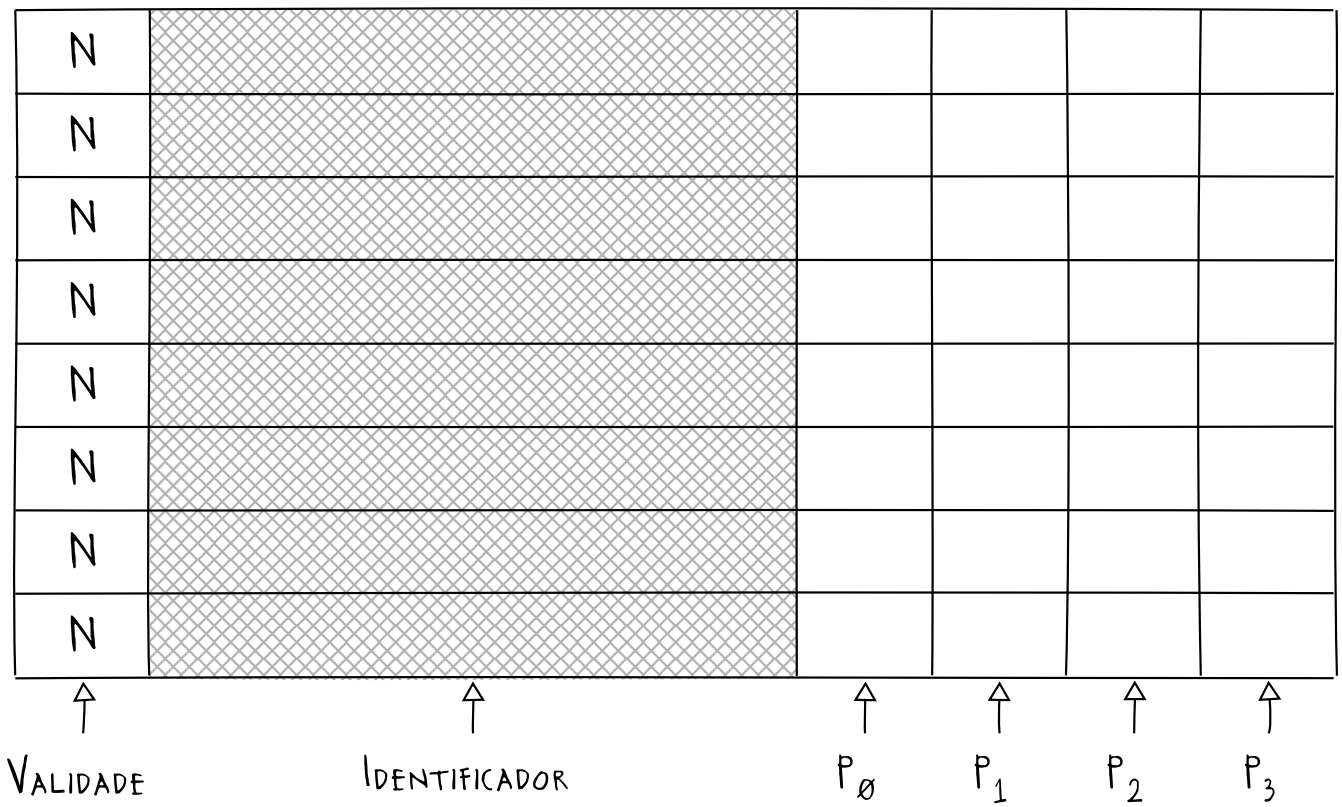
Memória cache

- Mapeamento associativo por conjunto
  - Grau de associatividade 4



Memória cache

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



Memória cache

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

Memória 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

Memória 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 bloco armazenado na cache
    - *First-In First-Out* (FIFO): o mais antigo sempre é substituído, independente das referências realizadas
    - *Last Frequently Used* (LFU): o que é menos frequentemente acessado pelo processador é substituído na cache
    - *Last Recently Used* (LRU): é feita substituição do bloco que foi referenciado a mais tempo pelo processador
    - Randômico: qualquer bloco é aleatoriamente substituído

Memória cache

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

Memória 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

Memória 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

Memória 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?

Memória 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

Memória 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

Memória 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 no barramento

Memória 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

Memória 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

Memória 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 no barramento, uma vez que todas as operações são realizadas na cache

Memória 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 no barramento, uma vez que todas as operações são realizadas na cache

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

Memória 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

Memória 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

Memória cache

- Avaliação 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 de acesso

Memória cache

- Avaliação 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 de acesso
  - 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)

Memória cache

- Avaliação 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 de acesso
  - 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

Memória cache

- Avaliação 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

$$\textit{Latência média} = TA + TF \times P$$

Memória cache

- Avaliação 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%

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

Memória cache

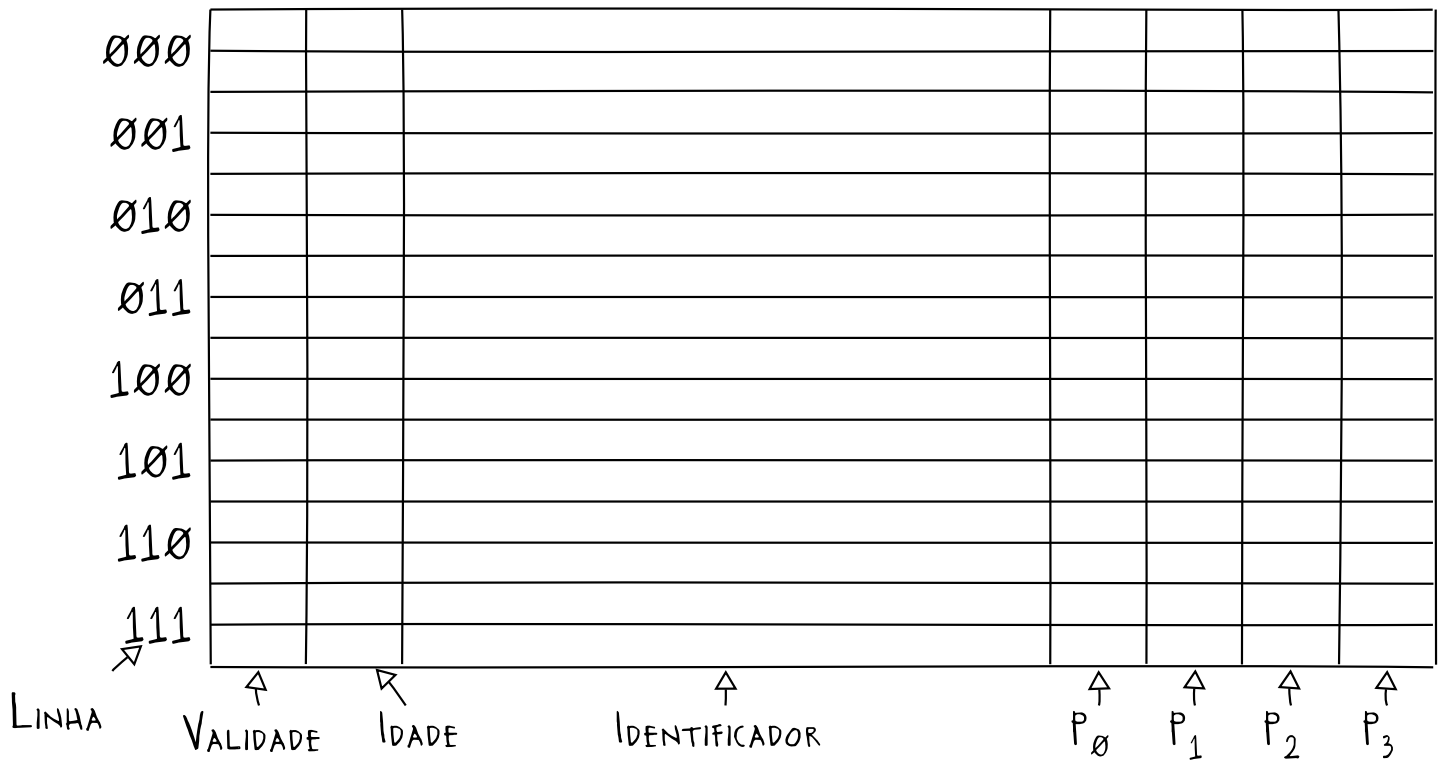
- Avaliação quantitativa de desempenho
  - Frequência de operação de 2 GHz
  - Tempo de acerto de 3 ciclos (cache)
  - Penalidade com tempo de 10 ciclos (memória)
  - Taxa de falta de 5%

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

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 antes da execução das instruções, com as taxas de acerto de das caches D e I no final da execução

```
#cache_mem:irm    0x80000000          line=0,valid={0,0},age={0,0},id={0x000000,0x000000}
0x80000000:jal    zero,0x00050        pc=0x800000a0,zero=0x80000004
#cache_mem:irm    0x800000a0          line=2,valid={0,0},age={0,0},id={0x000000,0x000000}
0x800000a0:auipc  sp,0x00008          sp=0x800000a0+0x00008000=0x800080a0
#cache_mem:irh    0x800000a4          line=2,age=0,id=0x1000001,block[0]={0x00008117,0xf6010113,0xf99ff0ef,0x608000ef}
0x800000a4:addi   sp,sp,0xf60         sp=0x800080a0+0xffffff60=0x80008000
#cache_mem:irh    0x800000a8          line=2,age=0,id=0x1000001,block[0]={0x00008117,0xf6010113,0xf99ff0ef,0x608000ef}
0x800000a8:jal    ra,0xfffcc          pc=0x80000040,ra=0x800000ac
...
#cache_mem:irm    0x80000050          line=5,valid={0,0},age={0,0},id={0x000000,0x000000}
#cache_mem:dwm    0x80000874          line=7,valid={0,0},age={0,0},id={0x000000,0x000000}
0x80000050:sw     zero,0x000(a0)      mem[0x80000874]=0x00000000
...
#cache_mem:irh    0x80000554          line=5,age=0,id=0x100000a,block[0]={0x22b1a283,0x2251a5a3,0x29a1a303,0x2861ad23}
#cache_mem:dwh    0x8000090f          line=0,age=0,id=0x1000012,block[1]={0x00000000,0x00000000,0x00000000,0x00000000}
0x80000554:sw     t0,0x22b(gp)        mem[0x8000090f]=0x00000000
...
#cache_mem:irm    0x80000090          line=1,valid={1,1},age={51,19},id={0x100000c,0x100000d}
0x80000090:slli   zero,zero,31        zero=0x00000000<<31=0x00000000
#cache_mem:irh    0x80000094          line=1,age=0,id=0x1000001,block[0]={0x01f01013,0x00100073,0x40705013,0x00008067}
#cache_mem:irh    0x80000090          line=1,age=0,id=0x1000001,block[0]={0x01f01013,0x00100073,0x40705013,0x00008067}
#cache_mem:irh    0x80000098          line=1,age=0,id=0x1000001,block[0]={0x01f01013,0x00100073,0x40705013,0x00008067}
0x80000094:ebreak
#cache_mem:dstats                     hit=0.9872
#cache_mem:istats                     hit=0.9741
```