Fundamentos de Arquitetura de Computadores

Prof. Marcos Quinet
Universidade Federal Fluminense – UFF
Instituto de Ciência e Tecnologia - ICT

Representação de Memória

- Toda a informação com a qual um sistema computacional trabalha está, em algum nível, armazenada em um sistema de memória, guardando os dados em caráter temporário ou “permanente”

- Como já estudamos, a menor quantidade de informação que podemos armazenar é em um dígito binário, ou BIT. Utilizando um bit estamos limitados a representar somente dois dados. Precisamos definir agrupamentos de bits para representarmos mais símbolos ou valores

Representação de Memória

- Para representarmos uma letra ou número através de bits convencionou-se montar um agrupamento de 8 bits, chamado de byte

- Inicialmente, dos 8 bits, 7 eram usados para combinações, permitindo 128 símbolos diferentes, e 1 bit usado para controle de paridade. Hoje os sistemas são mais confiáveis e os 8 bits podem ser utilizados para combinações, permitindo 256 símbolos diferentes

- Um byte pode armazenar um caractere, que é a unidade básica de armazenamento na maioria dos sistemas computacionais

Representação de Memória

- Vários padrões de codificação de caracteres utilizando bits foram criados, o que foi adotado pela maioria dos fabricantes foi o ASCII – *American Standard Code for Information Interchange*

- Alguns dos padrões existentes baseiam-se no padrão ASCII, fazendo adaptações para símbolos particulares de uma linguagem, como ‘~’, ‘^’ e ‘ç’

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Representação de Memória

- Atualmente o padrão Unicode é mais empregado, contando com diferentes versões (as mais populares são o UTF-8 e UTF-16)

- A versão 11.0, de junho de 2018 inclui 137.439 caracteres, suportando 146 linguagens distintas e até emojis

Representação de Memória

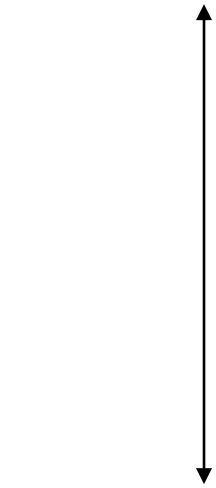
- O sistema de medida adotado para memória utiliza como base 2 por estarmos lidando com o sistema binário. Desta forma, 2 elevado a décima potência (2^{10} = 1024) de uma unidade equivale a uma unidade no próximo nível da escala de valores
  - 1024 bytes = 1 Kilobyte (KB)
  - 1024 kilobytes = 1 Megabyte (MB)
  - 1024 megabytes = 1 Gigabyte (GB)
  - 1024 gigabytes = 1 Terabyte (TB)
  - 1024 terabytes = 1 Petabyte (PB)
  - 1024 petabytes = 1 Exabyte (EB)
  - 1024 exabytes = 1 Zetabyte (ZB)
  - 1024 zetabytes = Yottabyte (YB)
  - 1024 yottabytes = 1 Brontobyte (BB)
  - 1024 brontobytes = 1 Geopbyte (GPB)

Tipos de Memória

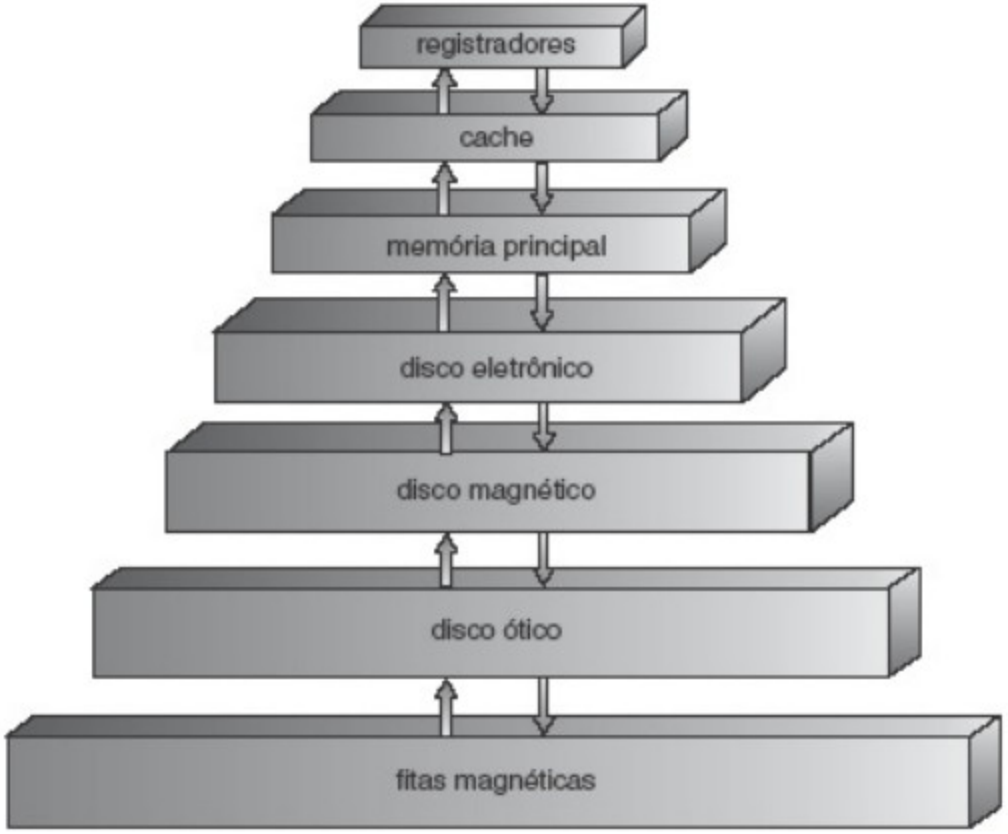
- Os componentes nos quais podemos armazenar informações em um sistema computacional, chamados de memória do computador podem formar em uma das seguintes classes:
  - Memória principal;
  - Memória secundária;
  - Memória cache;
  - Memória virtual.

Hierarquia de Memória

Alto custo
Baixo tempo de acesso
Baixa capacidade



Baixo custo
Alto tempo de acesso
Capacidade elevada



Hierarquia de Memória

- As características levadas em consideração para a utilização de um tipo de memória em um sistema computacional são o custo, a capacidade de armazenamento e a velocidade de acesso à informação

- A medida que descemos na hierarquia de memória, observamos que:
  - O custo por bit diminui;
  - A capacidade aumenta;
  - O tempo de acesso aumenta;
  - A frequência de acesso à memória pelo processador diminui.

Registradores

- Registradores são dispositivos de memória presentes no interior da UCP, construídos utilizando semicondutores (por isso são elementos de armazenamento volátil)

- Sua função é a de armazenar dados que serão utilizados pela UCP para o processamento da instrução atual ou de uma das próximas; os resultados de um processamento também precisam ser guardados temporariamente na UCP para serem novamente utilizados pela ULA ou para serem transferidos a um dispositivo de memória externo à UCP

- Registradores estão no topo da pirâmide, portanto possuem baixa capacidade de armazenamento (normalmente, o tamanho da palavra do processador), alta velocidade e alto custo. Sua utilização será estudada com mais detalhes durante o estudo da UCP

Comunicação entre Memórias

- Na maioria dos sistemas de computação, a memória principal é considerada rápida, mas não existe em grande quantidade, e a memória secundária é barata e existe em grande quantidade, mas muito lenta se comparada com as demais. Deve-se analisar as necessidades de tipo de memória do sistema, para que este não seja desnecessariamente caro ou insuportavelmente lento

- Na prática, as memórias são organizadas de modo que as mais rápidas (e consequentemente, mais caras e de menor capacidade de armazenamento) são localizadas mais próximas do processador

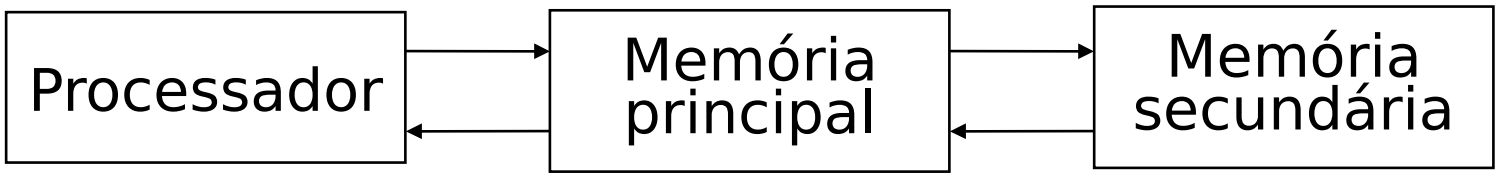
Comunicação entre Memórias

- São armazenados nas memórias de alta velocidade os dados e instruções que o processador vai utilizar com mais frequência. Em memórias mais lentas, com grande capacidade de armazenamento podem ser usadas para guardar dados e instruções que não serão necessários naquele momento

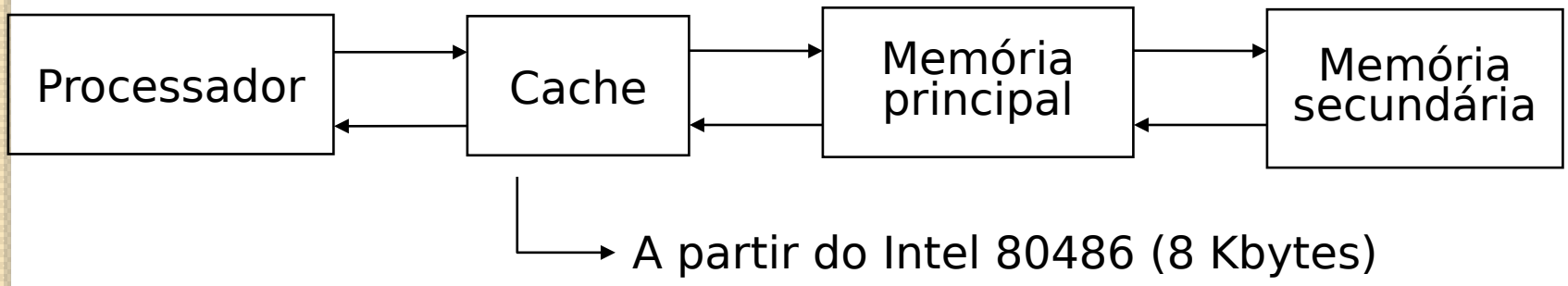
- As memórias são organizadas hierarquicamente de forma a obtermos um sistema com desempenho (velocidade) próximo ao da memória mas rápida e custo por bit próximo ao da memória de menor custo

Comunicação entre Memórias

- Memória em 2 níveis:



- Memória em 3 níveis:



Interação entre Processador e Memória Principal

- Apesar de nosso componente básico da memória principal ser uma célula de bit, em uma referência a memória, o processador não realiza o acesso ao conteúdo de uma única célula de bit, mas sim a um conjunto de células de bits

- Na maioria dos sistemas uma locação de memória é formada por 8 células de bit, logo uma locação de memória é conhecida como “byte de memória”

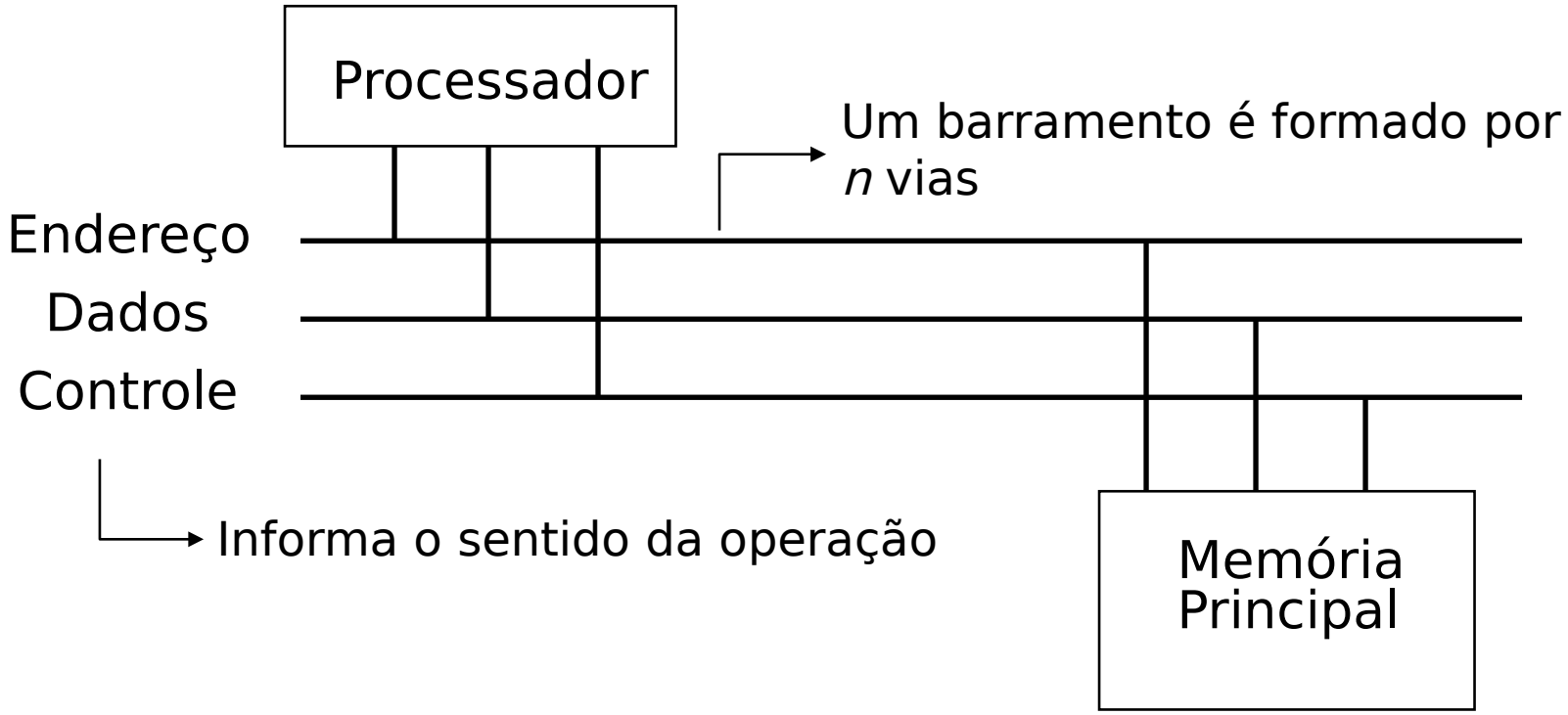
Interação entre Processador e Memória Principal

- Em um acesso, o processador deve fornecer uma identificação da locação onde será feito o acesso. Esta identificação é somente um número, chamado endereço de memória

- De posse deste endereço, a memória principal seleciona a locação correspondente e fornece ao processador a informação ali contida, no caso de um acesso de leitura; em um acesso de escrita, a memória principal armazena na locação indicada pelo endereço o dado fornecido pelo processador

Interação entre Processador e Memória Principal

- A memória principal e o processador são interligados através de três barramentos distintos.



Interação entre Processador e Memória Principal

- O barramento de endereço é a via através da qual o processador transmite para a memória principal o endereço da locação onde será feito o acesso

- O barramento de dados é a via através da qual o conteúdo da locação é transferida entre o processador e a memória principal

- Pelo barramento de controle trafegam diversos sinais, através dos quais o processador controla o acesso a memória, indicando, por exemplo, se o acesso é de leitura ou de escrita

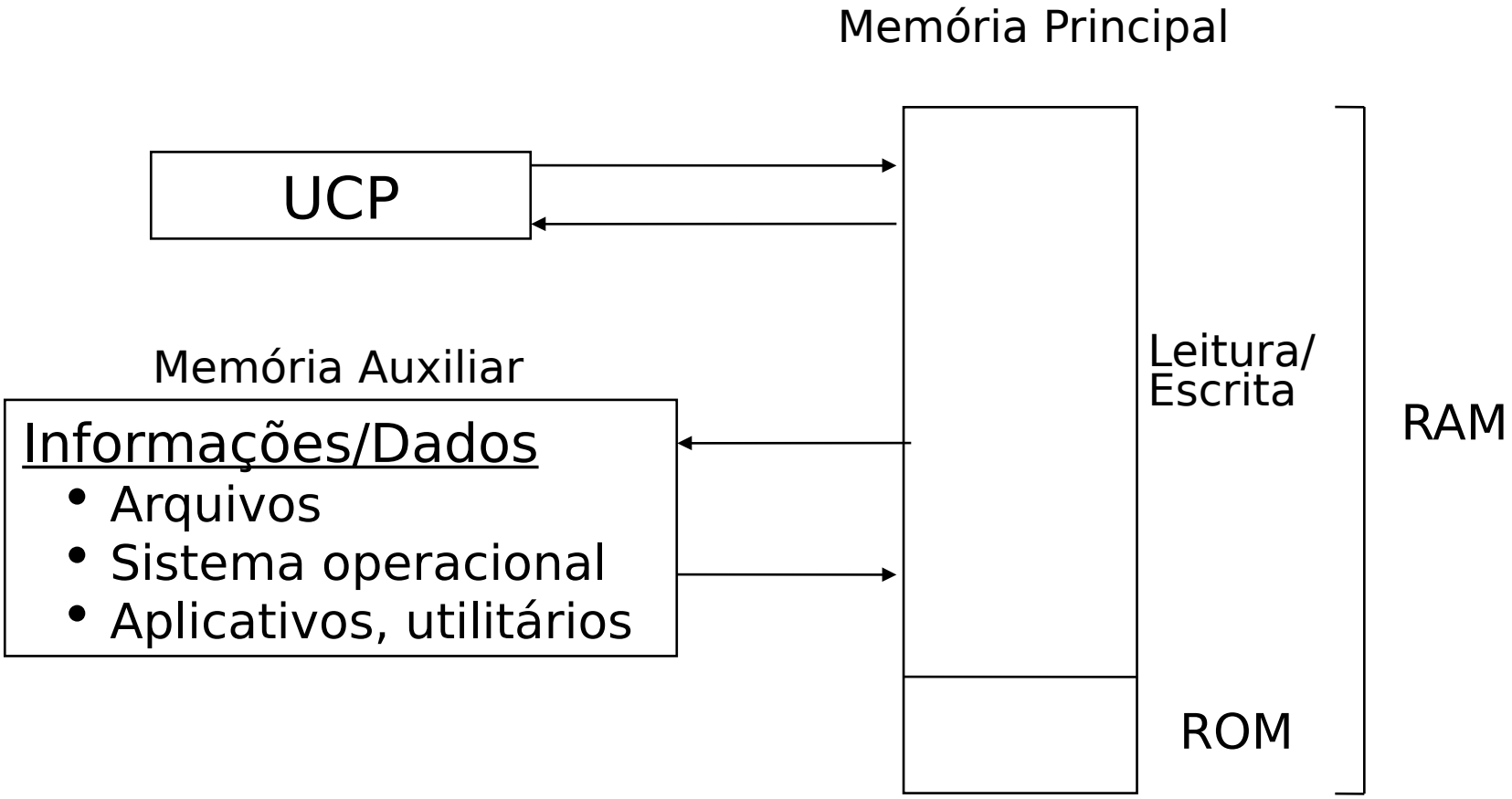
Interação entre Processador e Memória Principal

- O número de vias no barramento de dados é o fator determinante para o desempenho do processamento do sistema

- O fator limitante para a quantidade de memória que pode ser utilizada pelo sistema é o número de vias do barramento de endereço
  - Um sistema com barramento de *n* bits poderia endereçar até 2^n locações distintas de memória, porém, isso não significa que o sistema possua essa capacidade de memória principal instalada; por exemplo, sistemas baseados em processadores 80386 ou 80486 em geral possuíam no máximo 16 Mbytes de memória principal

- No caso de programas que utilizem uma capacidade de memória maior do que a memória principal existente no sistema é utilizado o mecanismo de memória virtual.

Memória Principal



Memória Principal

- Da capacidade total da memória principal, existirá uma área reservada aonde estarão carregados os dados da memória ROM, que apesar de estarem armazenados em um circuito de memória não-volátil, ao iniciar-se o computador são carregados para uma parte da memória principal para que possam ser executados

- Memórias de somente leitura (ROM) também possuem a característica do tempo de acesso aleatório, ou seja, também podem ser referidas como RAM. A utilização dos termos RAM e ROM popularmente utilizado pelo comércio, não são empregados da forma mais correta possível

Memória ROM

- A memória principal é formada por módulos divididos em duas categorias: leitura e escrita (que é popularmente tratada como RAM) e ROM
- A memória ROM (*Read Only Memory* – memória somente de leitura) é previamente gravada, podendo apenas ser acessada, e não modificada; os dados nela armazenados mantém-se íntegros na ausência de energia elétrica; isso a caracteriza como uma memória não-volátil
- A memória ROM pode ser de três tipos: PROM (*Programmable Read Only Memory*), EPROM (*Erasable PROM*) ou EEPROM (*Electrically EPROM*)

Memória ROM

- A memória PROM permite apenas acessos para leitura dos dados nela gravados. As memórias EPROM e EEPROM podem ter seu conteúdo apagado através da exposição a raios ultravioleta e corrente elétrica, respectivamente

- Em um dos tipos de memória ROM são armazenadas as sub-rotinas utilizadas pelo sistema operacional para interação com o hardware do computador, conhecidas como *firmware*. O mais conhecido *firmware* utilizados em sistemas microcomputadores é o BIOS (Basic Input/Output System)

Memória RAM

- A memória RAM (*Random Access Memory*) do computador é utilizada para o processador efetuar acessos de leitura e escrita ao dispositivo na qual esteja implementada (por exemplo, bancos ou “pentes” de memória)
- As memórias do tipo RAM são utilizadas fundamentalmente para o armazenamento de instruções e dados de um programa em execução
- O acesso aleatório se refere ao fato de palavras individuais da memória poderem ser acessadas diretamente, utilizando uma lógica de endereçamento implementada em hardware

Memória RAM

- Memórias RAM são implementadas utilizando semicondutores organizados em pastilhas, cuja principal característica é de que operações de leitura e gravação possam ser realizadas rapidamente

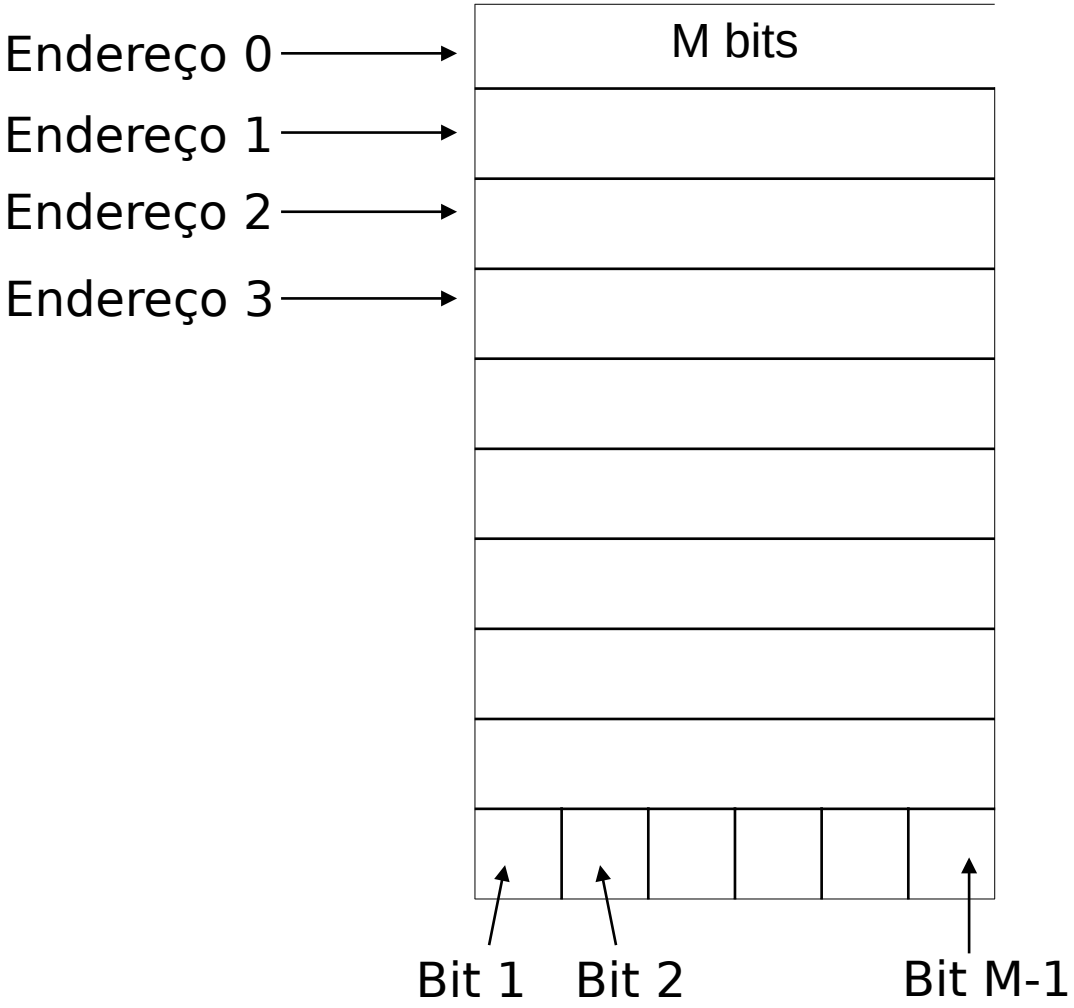
- As memórias RAM são voláteis, isto é, os dados nela armazenados só existem enquanto alimentados continuamente por corrente elétrica. No caso da ausência de corrente, todos os dados são imediatamente apagados de forma irrecuperável. A RAM só pode ser usada para o armazenamento temporário de dados

Memória RAM

- A característica do acesso aleatório (randômico) é considerada boa, pois o tempo de acesso é igual para todas as localidades de memória

- Em tecnologias de acesso sequencial, o tempo de acesso é proporcional à localidade na qual a informação se encontra
  - Por exemplo, se você está na primeira posição de memória de uma fita magnética, para acessar o dado que está na última posição, deve-se passar toda a fita até alcançar o final da mesma. Se em seguida quisermos acessar a segunda posição de memória, a fita deve ser rebobinada quase até o começo. Na média, o tempo de acesso sequencial é sempre pior do que o aleatório

Organização da Memória Principal



A memória principal é dividida em N células, todas formadas pela mesma quantidade de M bits.

Tipos de Memória RAM

- Dependendo dos materiais utilizados na fabricação da memória, existem dois tipos de RAM: estática (SRAM – *Static* RAM) ou dinâmica (DRAM – *Dynamic* RAM)
- SRAM e DRAM se diferem quanto à capacidade de armazenamento e ao tempo de acesso
- Uma DRAM é constituída por células que armazenam dados com a carga de capacitores, onde a presença ou ausência de carga em um capacitor representa um dígito binário 0 ou 1

DRAM e SRAM

- A DRAM requer uma regeneração periódica de carga (*refresh*) para manter os dados nela armazenados, pois um capacitor tem a tendência natural de se descarregar com o tempo
- Na memória SRAM os dígitos binários são armazenados utilizando configurações *flip-flop (*também chamado de *latch)* com portas lógicas, mantendo estes dados enquanto houver fornecimento de energia; um circuito *flip-flop* pode ser usado como uma memória de 1 bit

DRAM e SRAM

- Nas memórias DRAM a célula de bit é implementada por um circuito eletrônico que ocupa uma área de integração menor que a ocupada pelo circuito usado nas memória SRAM
- Como a área por célula bit é menor, para a mesma área total de integração o número total de células de bit em uma DRAM é maior, tornando-a mais densa

DRAM e SRAM

- As memórias SRAM possuem como vantagem sobre as DRAM um tempo de acesso reduzido, chegando a ter um tempo de acesso várias vezes menor

- Enquanto a capacidade de armazenamento das memórias DRAM cresce rapidamente, a diminuição do tempo de acesso é de cerca de 30% a cada dez anos

DRAM e SRAM

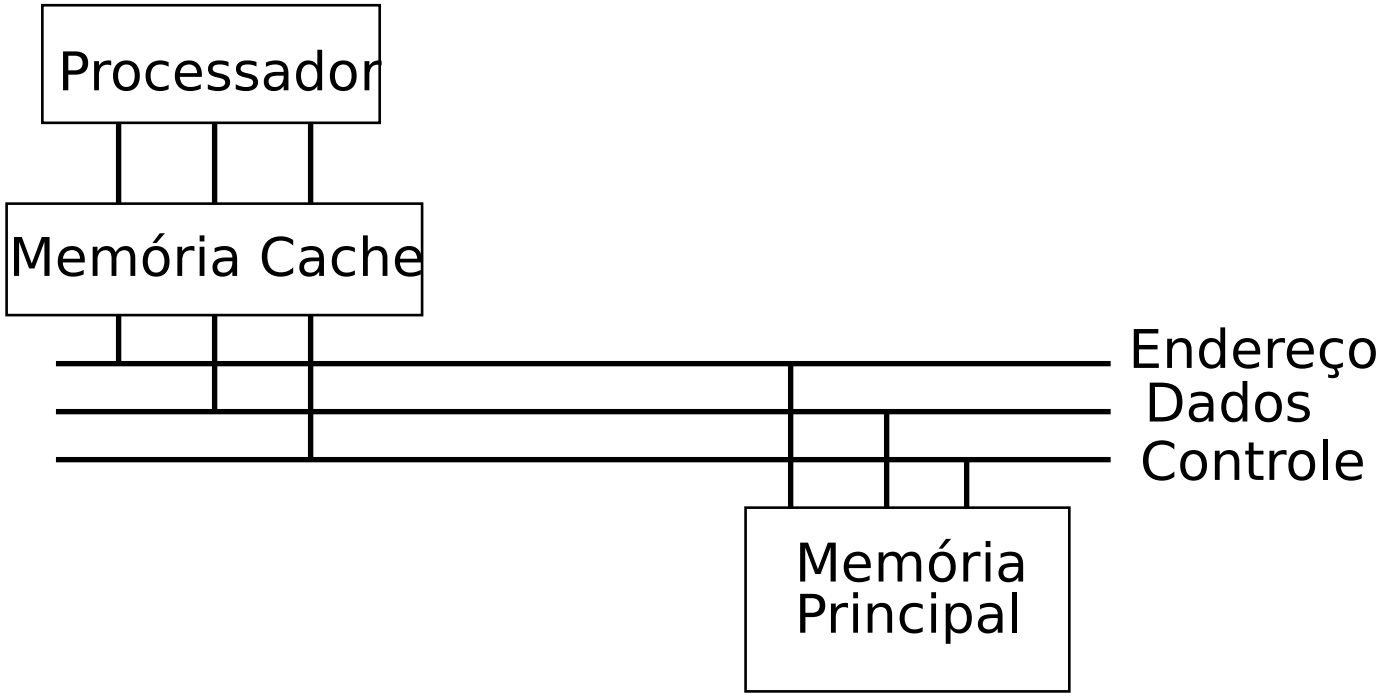
- Dispositivos DRAM são utilizados em sistemas de baixo e médio custo, permitindo dotar o sistema com uma capacidade de memória elevada, sem aumentar significativamente seu custo
- Por apresentarem um custo mais elevado, as memórias SRAM são utilizadas apenas para sistemas de alto desempenho ou para implementar pequenas porções de memória de funções específicas, como a cache

SRAM vs. DRAM

	Transistores por bit	Tempo de acesso	Custo	Aplicações
SRAM	**6**	**1x**	**100x**	**Memórias cache**
DRAM	**1**	**10x**	**1x**	**Memórias principais**

Memória Cache

- A memória cache é uma pequena porção de memória inserida entre o processador e a memória principal.



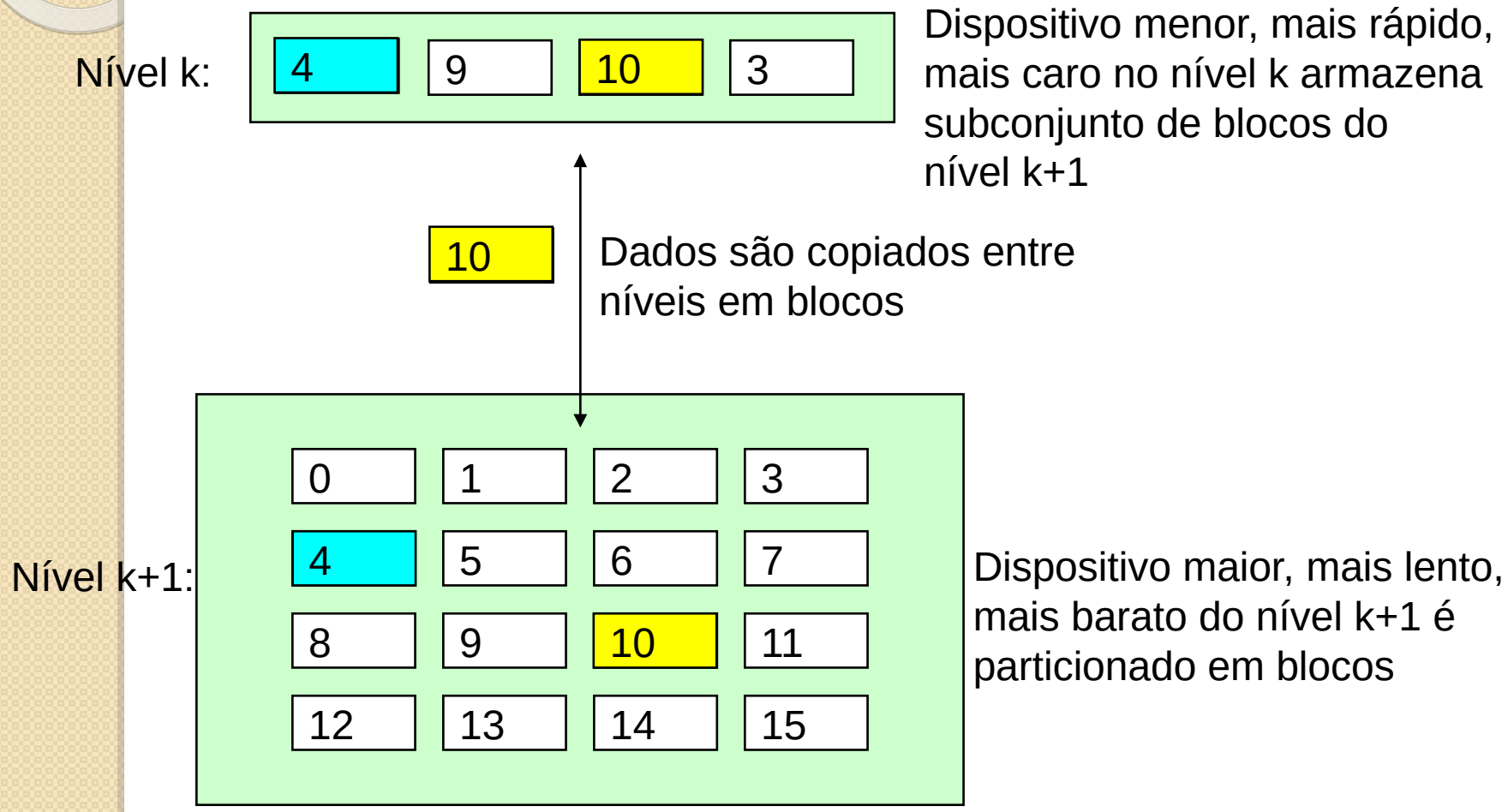
Memória Cache

- A memória cache é implementada com o mesmo tipo de circuito eletrônico usado nas células de bit de uma memória RAM estática (SRAM)

- Devido ao custo, a capacidade de armazenamento costuma ser da ordem de Kbytes. Nos sistemas mais modernos já encontramos caches com 1 Mbyte ou mais

- O tempo de acesso é o mais baixo dentre os tipos de memória, possibilitando que o processador acesse os dados ali armazenados em apenas um único ciclo de *clock*

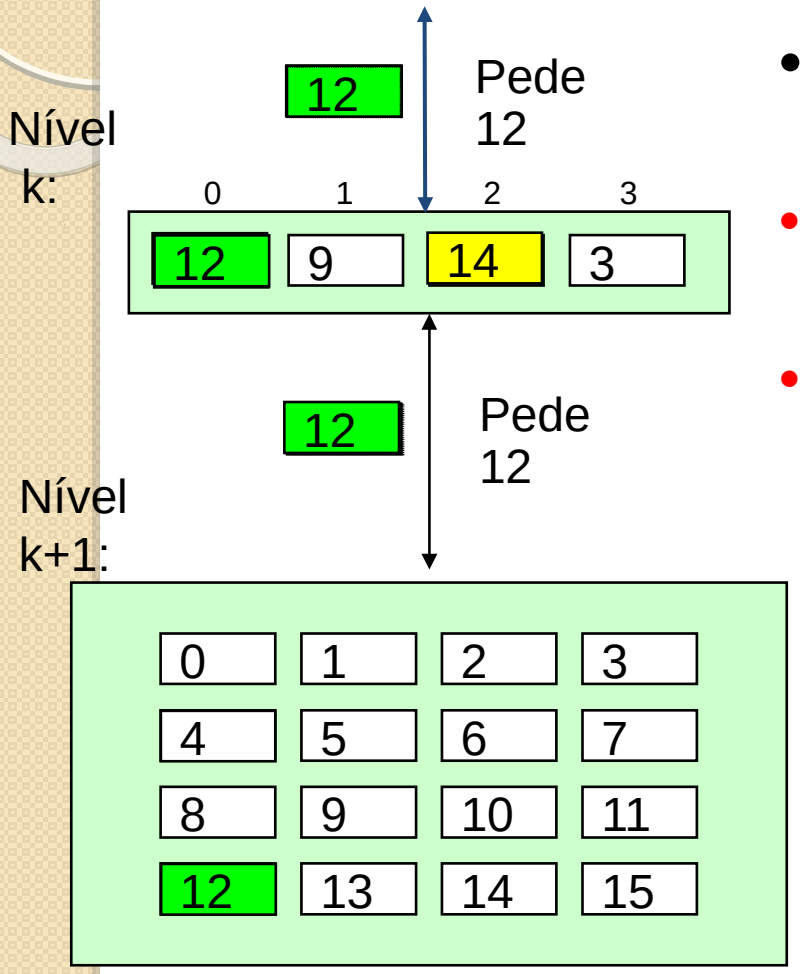
Caching na hierarquia de memórias



Memória Cache

- Neste modelo de organização, o processador direciona os acessos para a memória cache. Se o dado referenciado encontra-se na cache, o acesso é completado em apenas um ciclo de clock
- Caso o dado referenciado encontre-se na cache, dizemos que ocorreu um *cache hit*, caso contrário, ocorre um *cache miss*
- Quando ocorre um *cache miss*, um bloco contendo o dado referenciado, mais os dados armazenados em sua vizinhança, é copiado da memória principal para a memória cache.

Conceitos gerais de caching



- Programa precisa do objeto d, que está armazenado em algum bloco b
- Acerto de cache (cache hit)
  - Programa encontra b na cache nível k. E.x., bloco 14.
- Falta de cache (cache miss)
  - b não está no nível k, então a cache do nível k precisa pegar o bloco do nível k+1. Ex., bloco 12.
  - Se a cache do nível k está cheia, então algum bloco corrente terá que sair para dar lugar ao novo. Quem será a vítima?
    - Política de colocação: onde o novo bloco deve ir ? Ex., b mod 4
    - Política de substituição: qual bloco deve sair ? Ex., LRU

Localidade de Referência

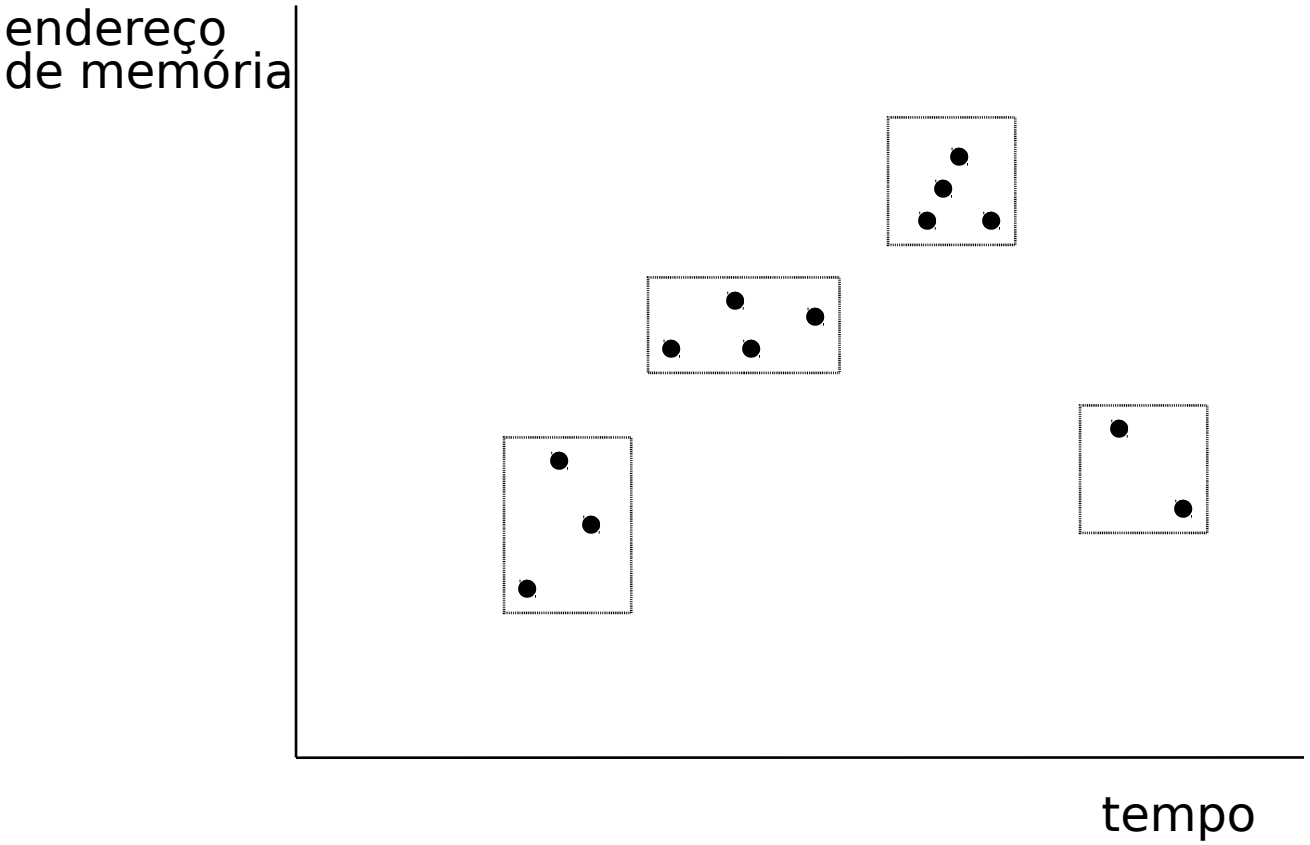
- Somente após a transferência deste bloco de dados para a memória cache o processador consegue completar o acesso. Quando ocorre um *cache miss*, o acesso consome vários ciclos de clock para ser completado

- Na prática, observa-se que entre 90% e 98% dos acessos resultam em um *cache hit*. Isso acontece devido a uma propriedade exibida por vários tipos de programa, chamada propriedade da localidade de referência

Localidade de Referência

- Segundo esta propriedade, os acessos à memória que acontecem ao longo da execução de um programa são uniformemente distribuídos através da memória, mas tendem a se concentrar em pequenas regiões da memória durante um certo intervalo de tempo

Localidade de Referência



Localidade de Referência

- Cada ponto no gráfico representa a ocorrência de um acesso à memória. A localidade de referência mostra-se no fato que, durante alguns intervalos de tempo, são feitos acessos a locações de memória em endereços próximos
- O princípio da localidade diz que quando é feito um acesso a uma locação de memória, existe uma grande probabilidade que acessos a locações em endereços vizinhos sejam feitos em endereços próximos. Na memória cache, se estes acessos de fato acontecerem, ocorrerá um *cache hit* em todos os acessos, que serão computados em um único ciclo de *clock*

Acesso à Cache

- A memória cache armazena os dados que se encontram em regiões da memória principal onde se verifica uma concentração de acessos

- Observe que a cache tem a capacidade de se adaptar a mudanças nas regiões de concentração de acesso que ocorrem ao longo da execução de um programa. Uma nova região com concentração de acessos torna-se ativa quando é feito um acesso a um dado que se encontra distante dos dados que foram acessados recentemente; como esse dado não está na cache, ocorrerá um *cache miss*

Acesso à Cache

- Devido ao *cache miss*, serão copiados para a memória cache não somente o dado referenciado, mas todos os dados na vizinhança, que pertencem a nova região onde se concentrarão os próximos acessos, que resultarão em *cache hits*

- Com a utilização de memória cache, apenas uma pequena parcela dos acessos realizados sofrem com o alto tempo de acesso da memória principal. Assim, as memórias cache são extremamente importantes para obter-se um melhor desempenho

Acesso à Cache

- Normalmente, a frequência de *cache hit* aumenta à medida que o tamanho da memória cache aumenta. Porém, na prática observamos que a partir de um certo ponto o número de *cache hits* não aumenta significativamente com o aumento da capacidade da memória cache

- A partir deste ponto não compensa aumentar o tamanho da cache, pois o aumento no custo do processador não traz um benefício proporcional

Acesso à Cache

- Para minimizar os *cache hits* residuais (que podem chegar a cerca de 10%), alguns sistemas utilizam duas memórias cache, uma primária (L1) e outra secundária (L2), em uma estrutura chamada de memória cache em dois níveis

- A cache primária é integrada ao processador, possui um tamanho pequeno e apresenta um tempo de acesso que permite acessos em um único ciclo de *clock*

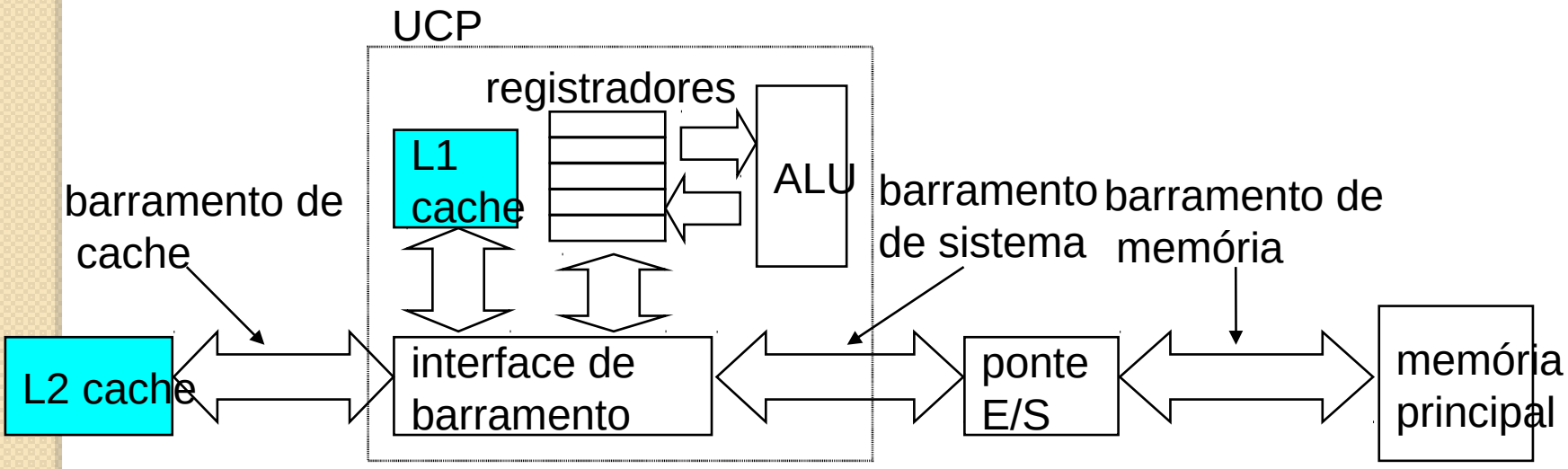
Cache primária e secundária

- Os acessos a cache primária que resultam em miss são direcionados para a cache secundária. Apesar de um pouco mais lenta que a cache primária, ainda é significativamente mais rápida que a memória principal

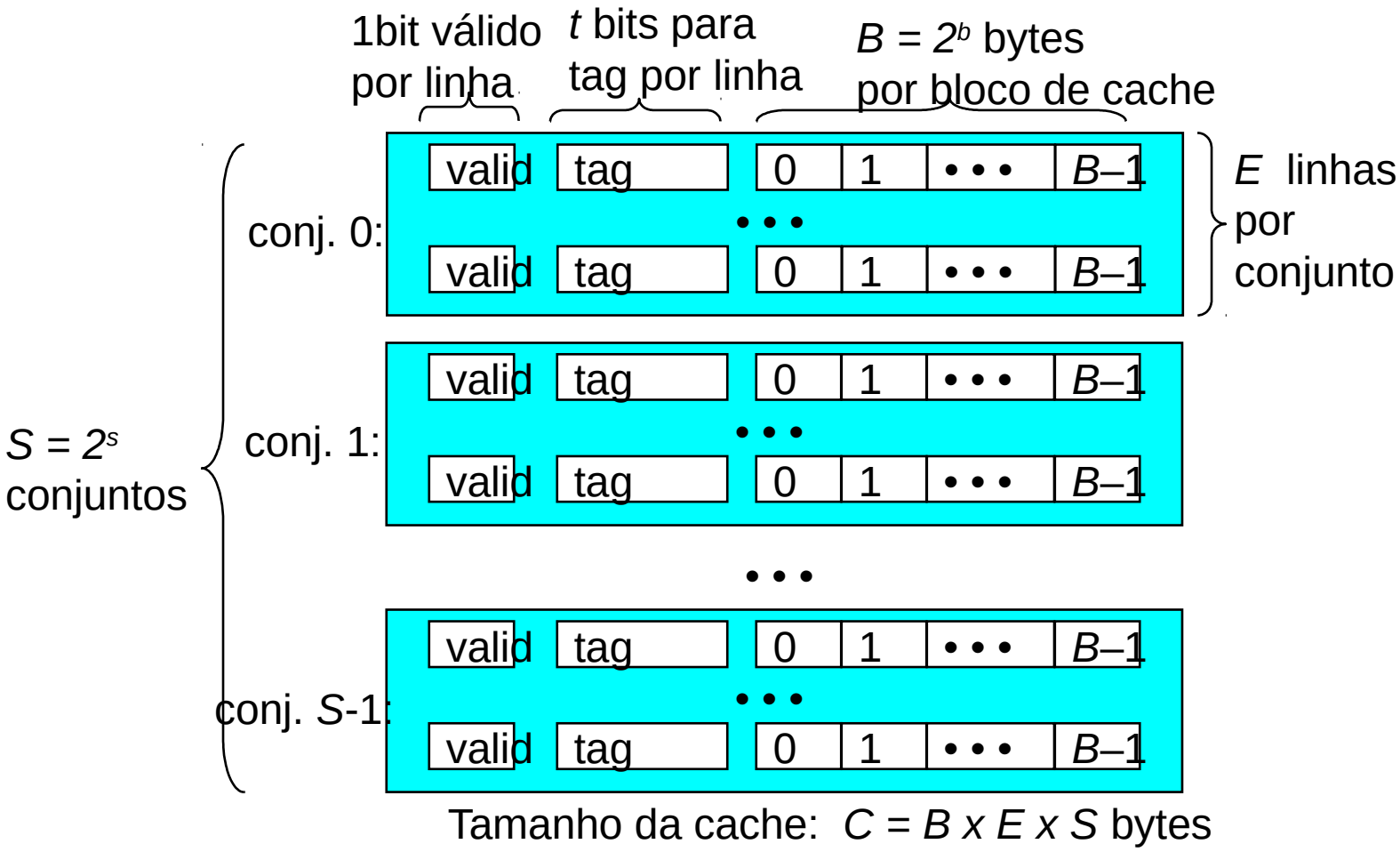
- Desta forma, os *misses* da cache primária apresentam uma penalidade menor quando capturados pela cache secundária do que se fossem direcionados diretamente para a memória principal

Organização de memórias cache

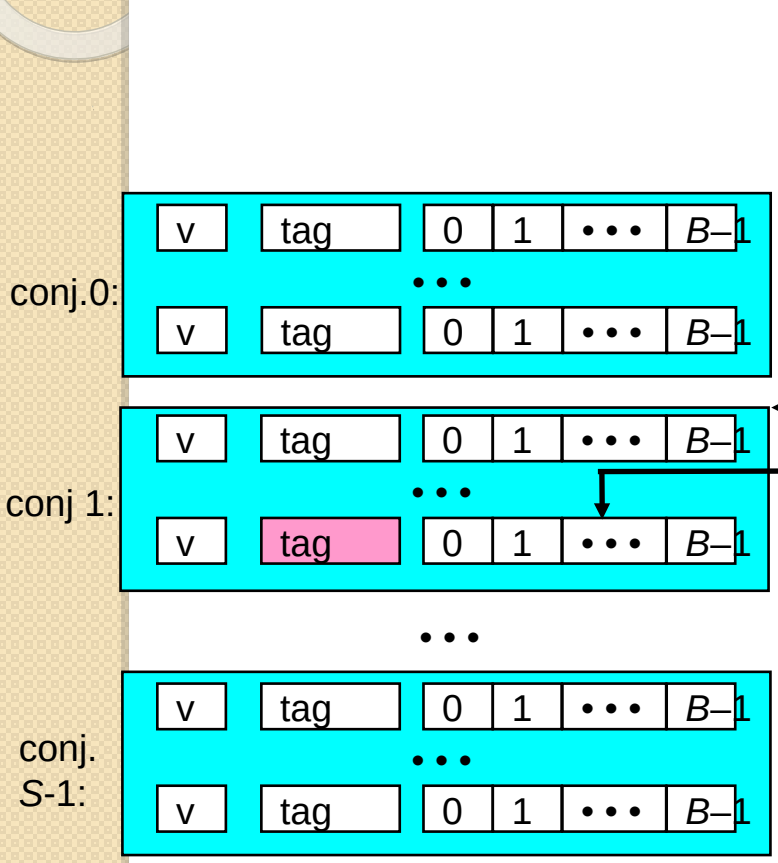
- A UCP procura por dados em L1, depois em L2 e finalmente na memória principal

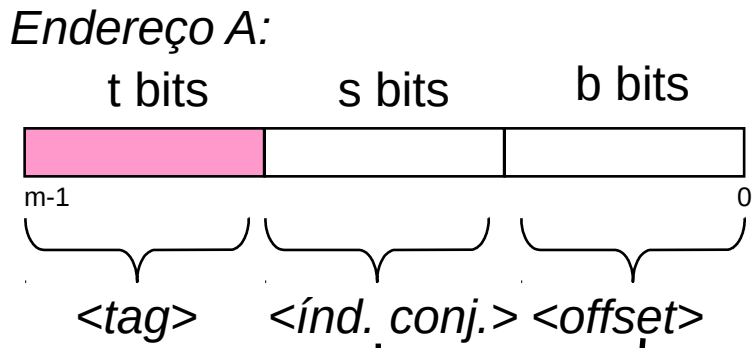


Organização geral da memória cache



Endereçamento de cache



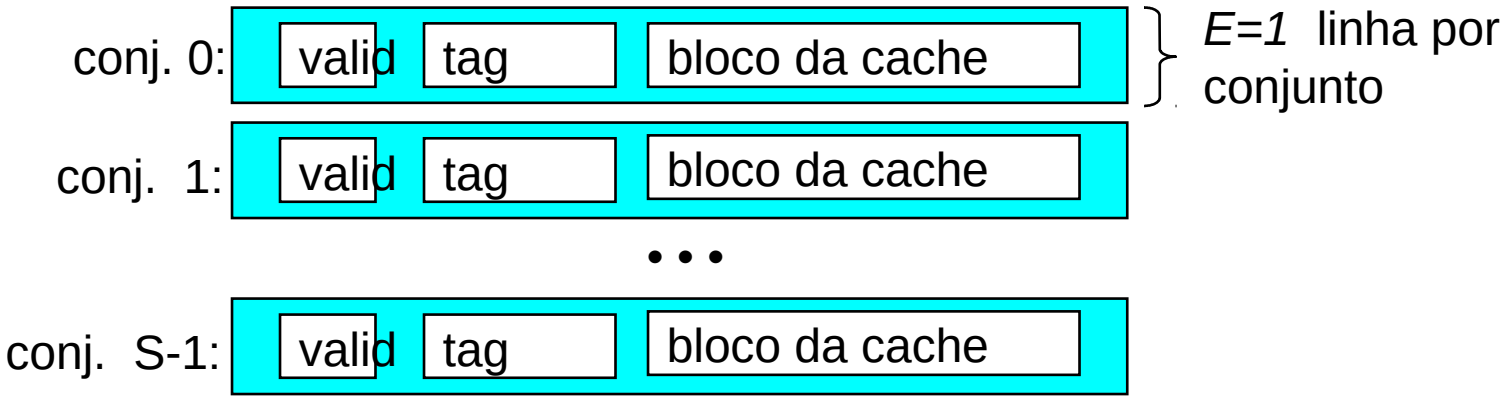


A palavra no endereço A está na cache se os bits de tag em uma das linhas válidas no conjunto <ind. conj.> são idênticos aos bits do campo <tag>.

O conteúdo da palavra começa no deslocamento de <offset> bytes a partir do início do bloco

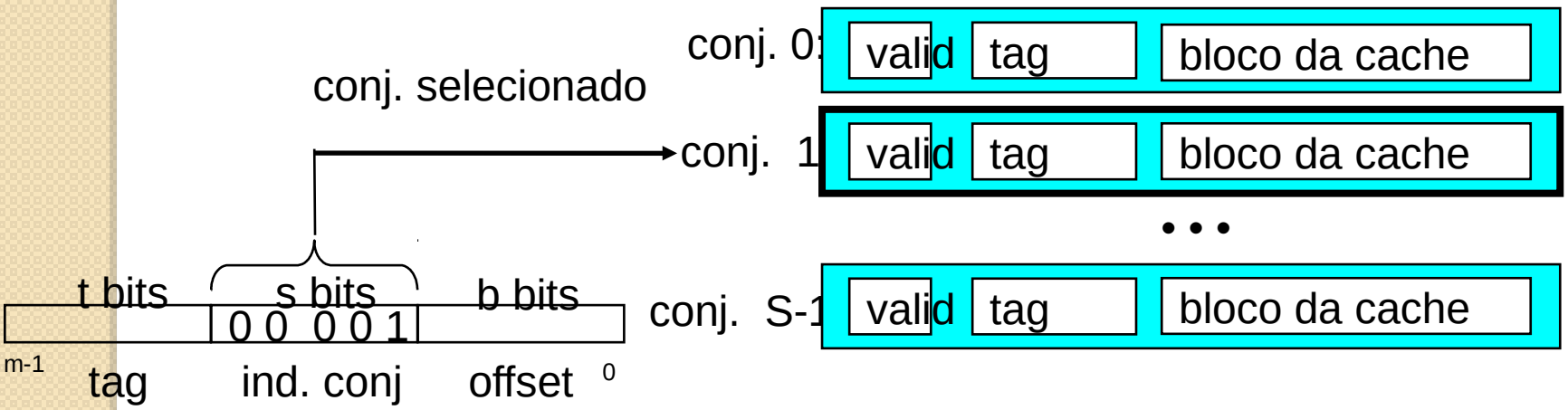
Cache mapeada diretamente

- Forma mais simples.
- Caracterizada por ter uma linha por conjunto.



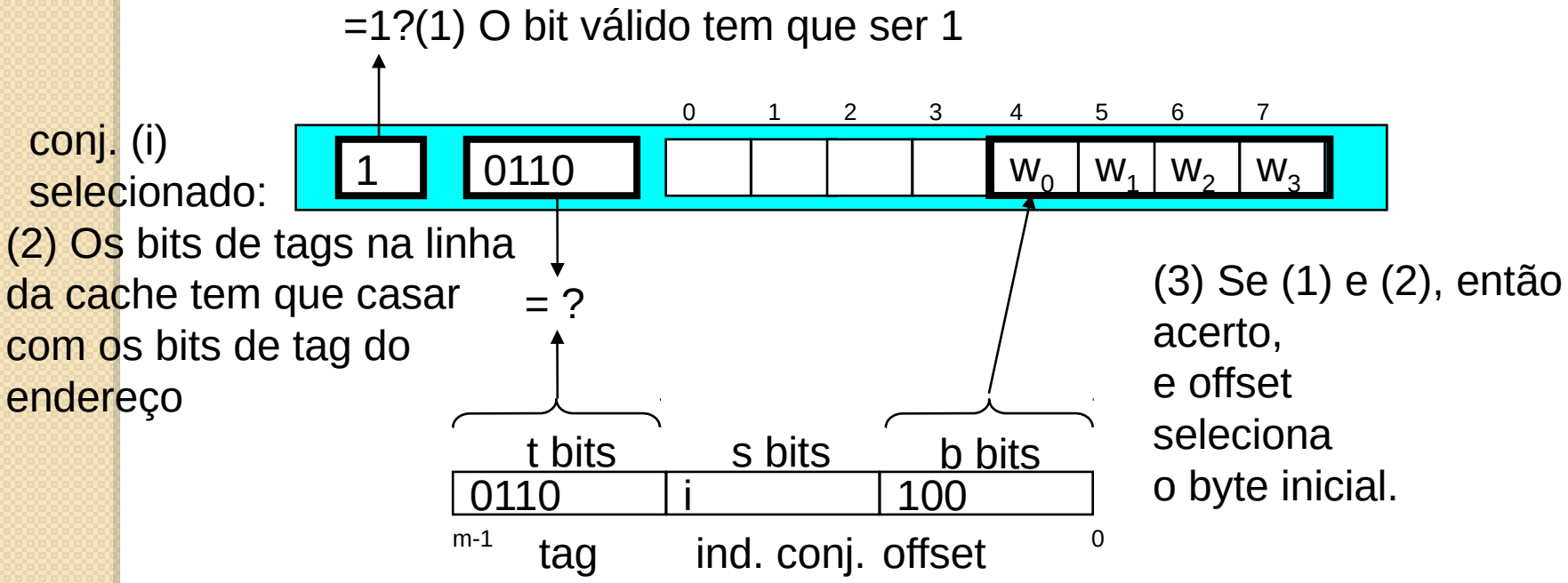
Accesso a caches mapeadas diretamente

- Seleção do conjunto feita pelos bits de índice de conjunto.



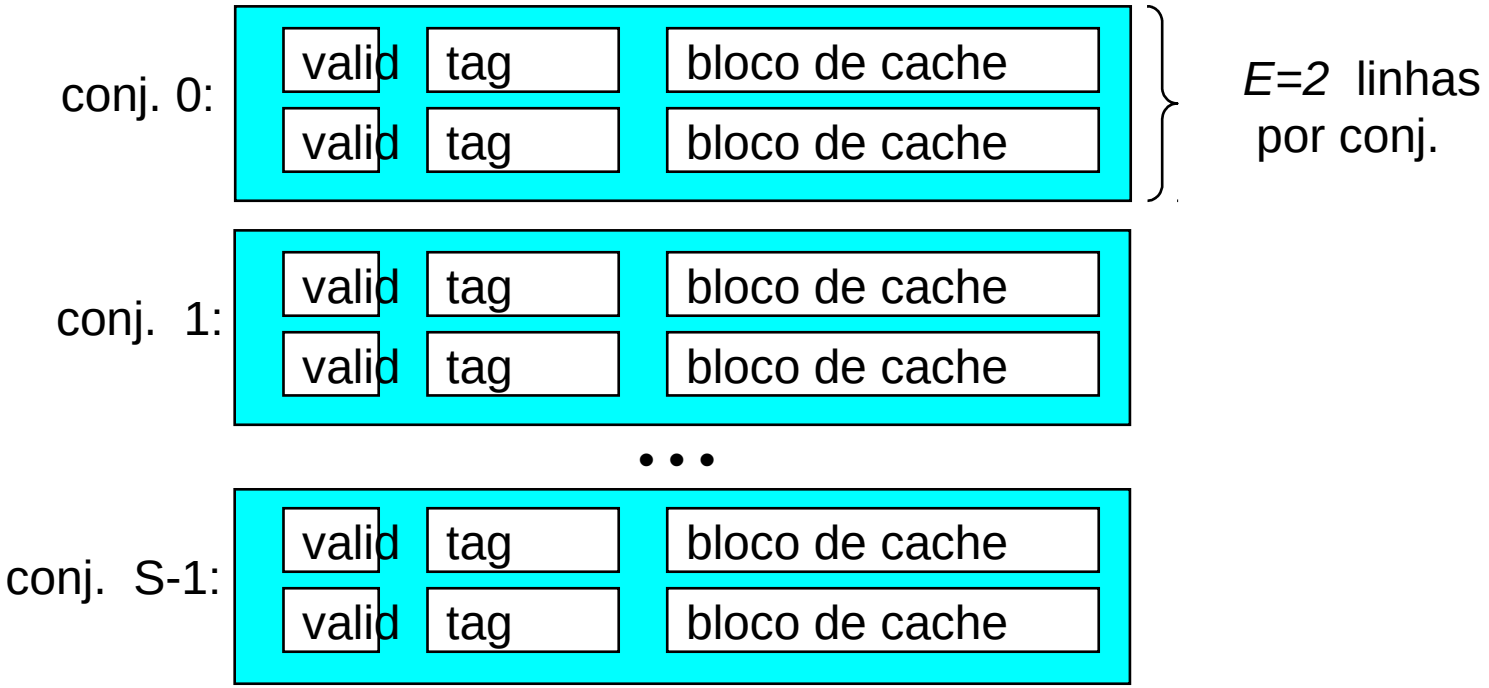
Accesso a caches mapeadas diretamente

- Encontra uma linha válida no conjunto selecionado com campo *tag* idêntico a bits de *tag* e extrai a palavra desejada.



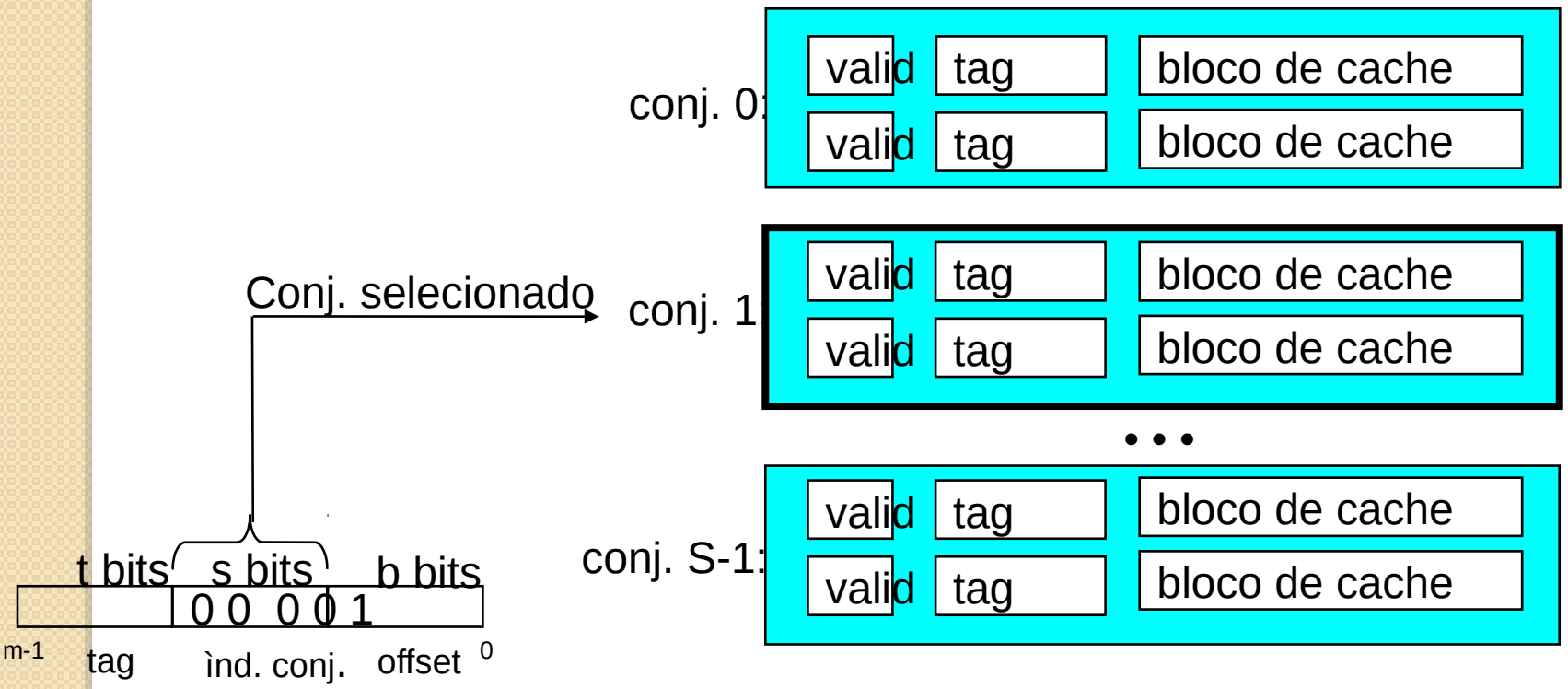
Caches associativas por conjunto

- Caracterizadas por mais de uma linha no conjunto.



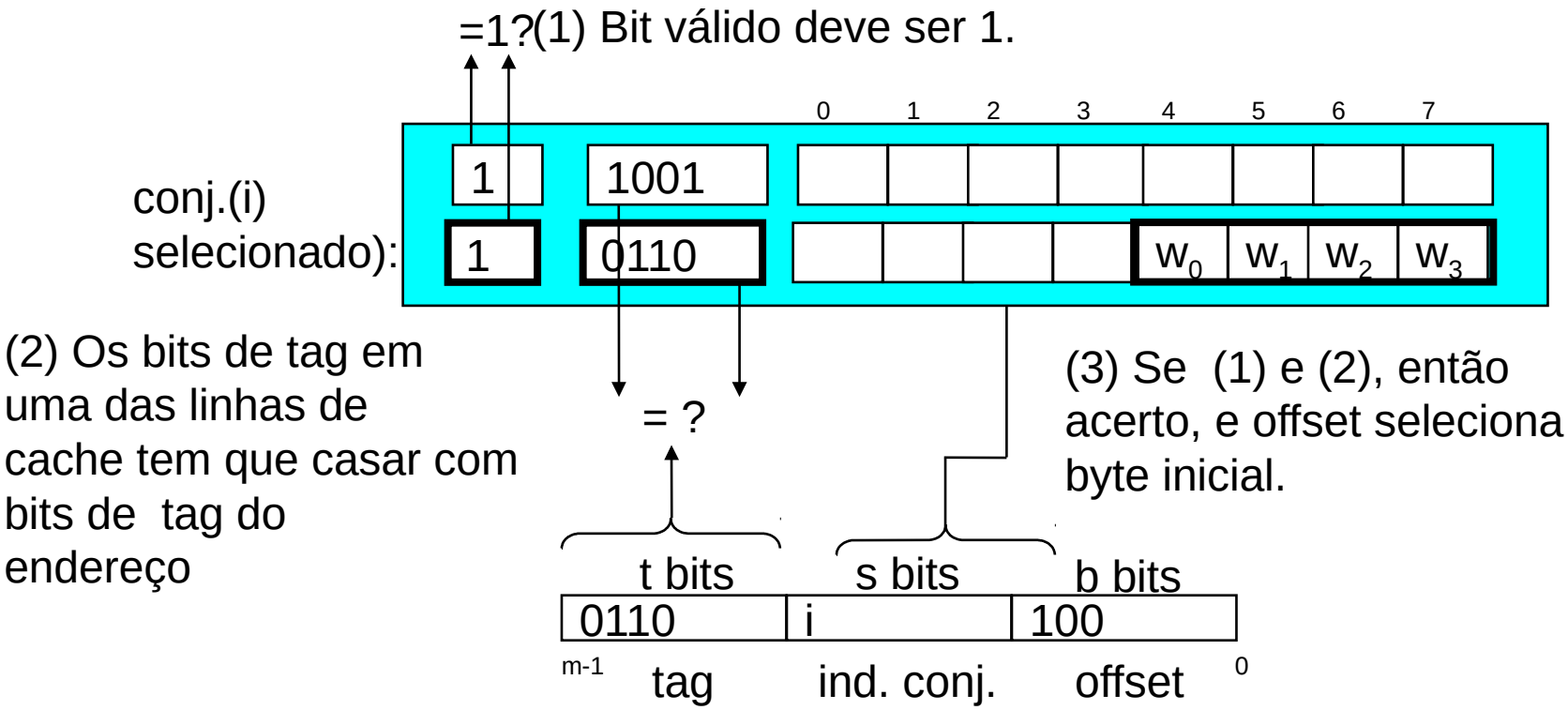
Acesso a caches associativas por conjuntos

- Seleção do conjunto é igual à memória mapeada diretamente.



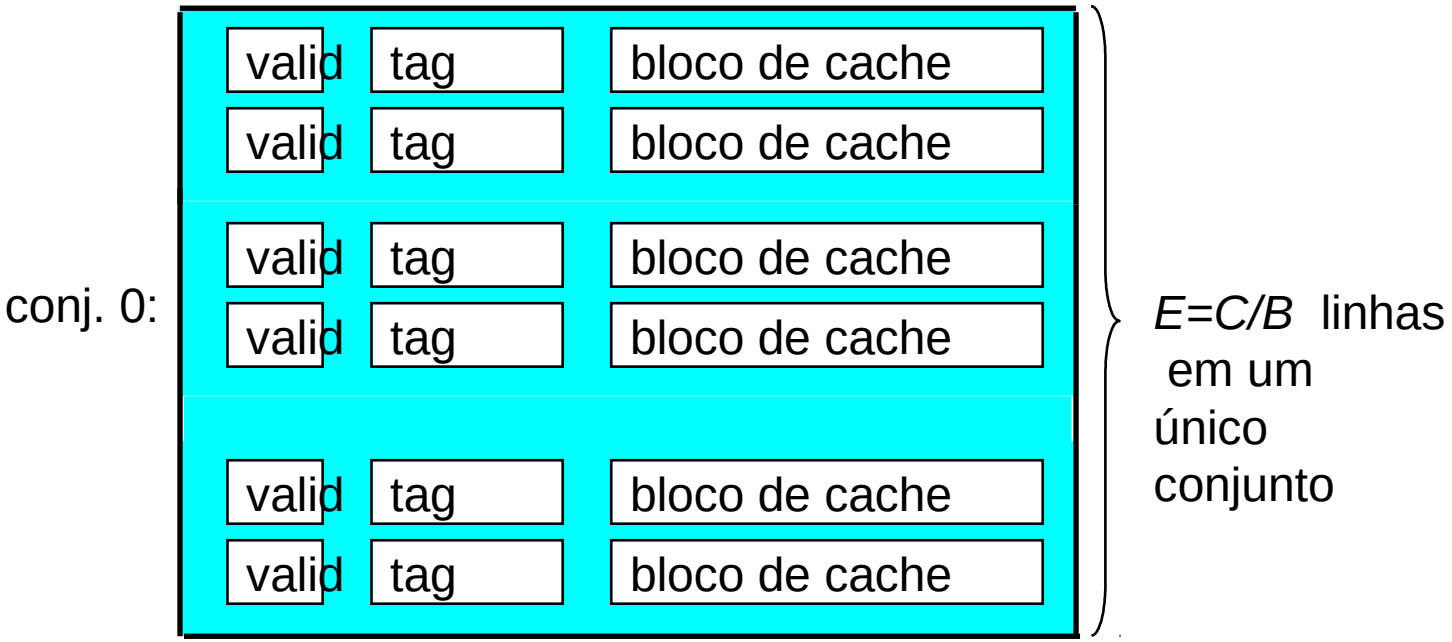
Acesso a caches associativas por conjunto

- Compara o tag de cada linha válida do conjunto selecionado.



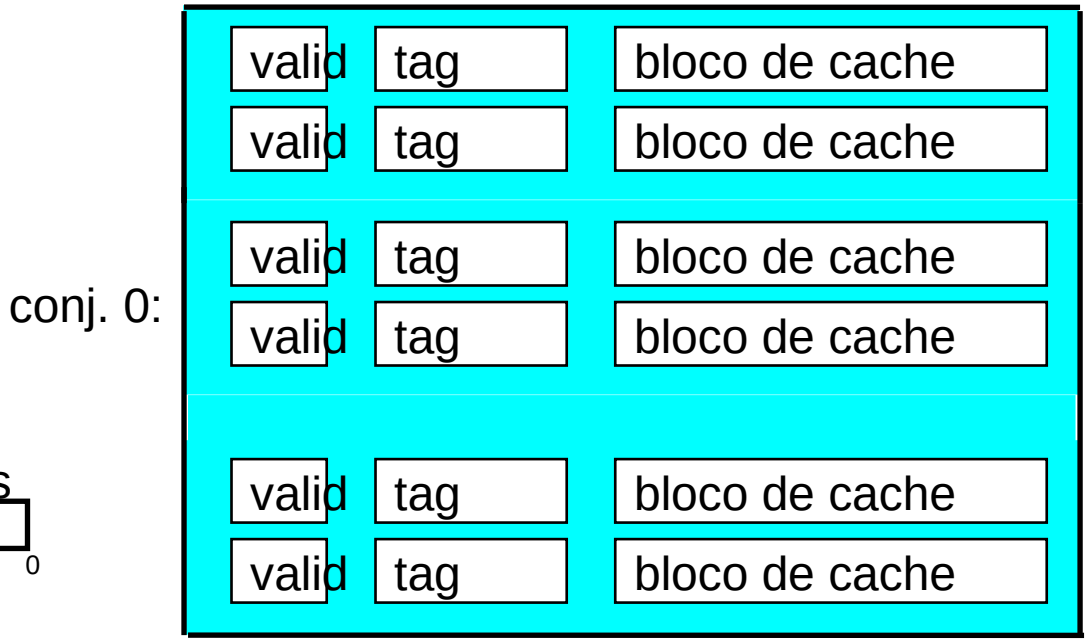
Caches totalmente associativas

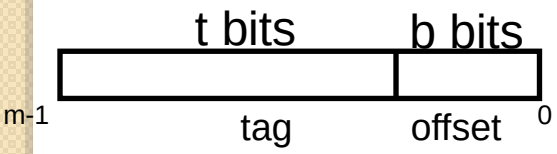
- Caracterizadas por um único conjunto conter todas as linhas.



Acesso a caches totalmente associativas

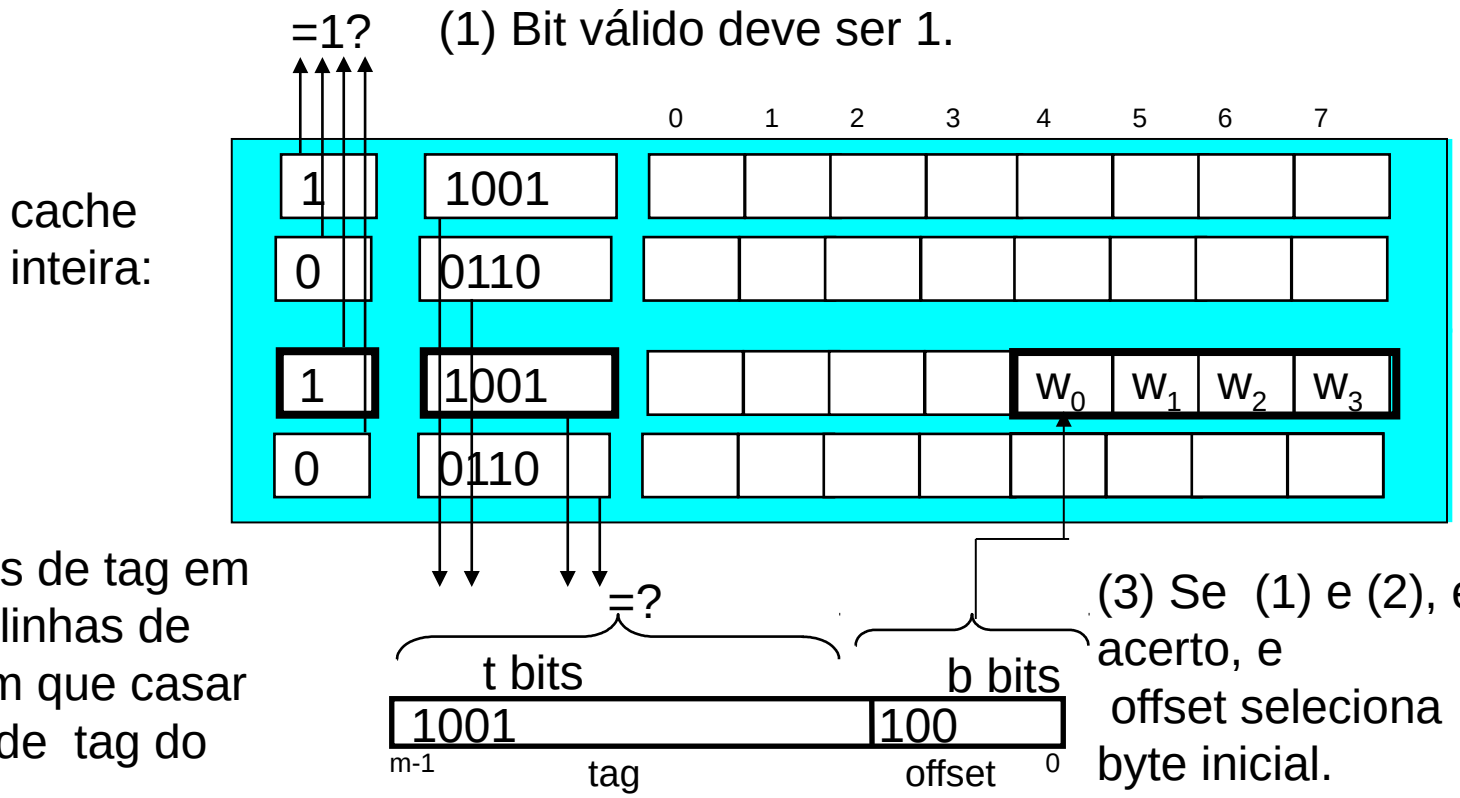
- Conjunto 0 é sempre selecionado.





Acesso a caches totalmente associativas

- Compara tag do endereço com campo tag da cache



(2) Os bits de tag em uma das linhas de cache tem que casar com bits de tag do endereço

(3) Se (1) e (2), então acerto, e offset seleciona byte inicial.

Algoritmo de substituição de dados

- Consiste em determinar qual o bloco da memória cache deve ser retirado para ceder lugar a outro por ocasião de uma falta
- No mapeamento direto não há opção!
- Nos demais mapeamentos ...
  - LRU - *Least Recently Used*
  - FIFO - *First In First Out*
  - LFU - *Least Frequently Used*
  - Escolha aleatória

Política de escrita

- A escrita é sempre realizada na cache pela CPU. Quando deve ser realizada na memória principal?
- Problema: vários processos em várias CPU ou dispositivos de E/S podem acessar um mesmo bloco na MP.
  - valores diferentes para um mesmo dado!
- Políticas de escrita:
  - *write through*
  - *write back*
  - *write once*

Política de escrita

- Escrita em ambas (*write through*)
  - Sempre que se escreve na cache, escreve-se na memória principal;
  - Pode haver queda no desempenho.
- Escrita somente no retorno (*write back*)
  - Escreve apenas quando o bloco é substituído: há bit de alteração;
  - A memória principal pode ficar desatualizada.
- Escrita uma vez (*write once*)
  - É utilizada quando há múltiplas CPUs;
  - A CPU que primeiro alterar o bloco autaliza-o na memória local, avisando às demais;
  - As demais CPUs não utilizam mais o dado da cache;
  - A atualização final ocorre na substituição.

Memória Virtual

- Como já vimos anteriormente, a memória principal disponível em um computador é em geral bem menor do que o tamanho máximo de memória permitido pelo processador (por exemplo, em uma arquitetura de 32 bits podemos endereçar até 4 GB de memória principal).

- O mecanismo de memória virtual foi criado para permitir a execução de programas que necessitem de mais memória do que a quantidade instalada no sistema.

Fundamentos de Memória Virtual

- Memória virtual – separação da memória lógica do usuário da memória física
  - somente uma parte do programa precisa estar na memória para execução;
  - espaço de endereçamento lógico pode ser muito maior que espaço de endereçamento físico;
  - permite o compartilhamento de espaços de endereços por vários processos;
  - compartilhamento possibilita criação mais eficiente de processos.
- Memória virtual pode ser implementada por:
  - Paginação sob demanda;
  - Segmentação sob demanda.

Memória Virtual

- Em um sistema de memória virtual, o endereço de memória gerado pelo programa é um endereço virtual, sendo diferente do endereço real usado para o acesso a memória principal

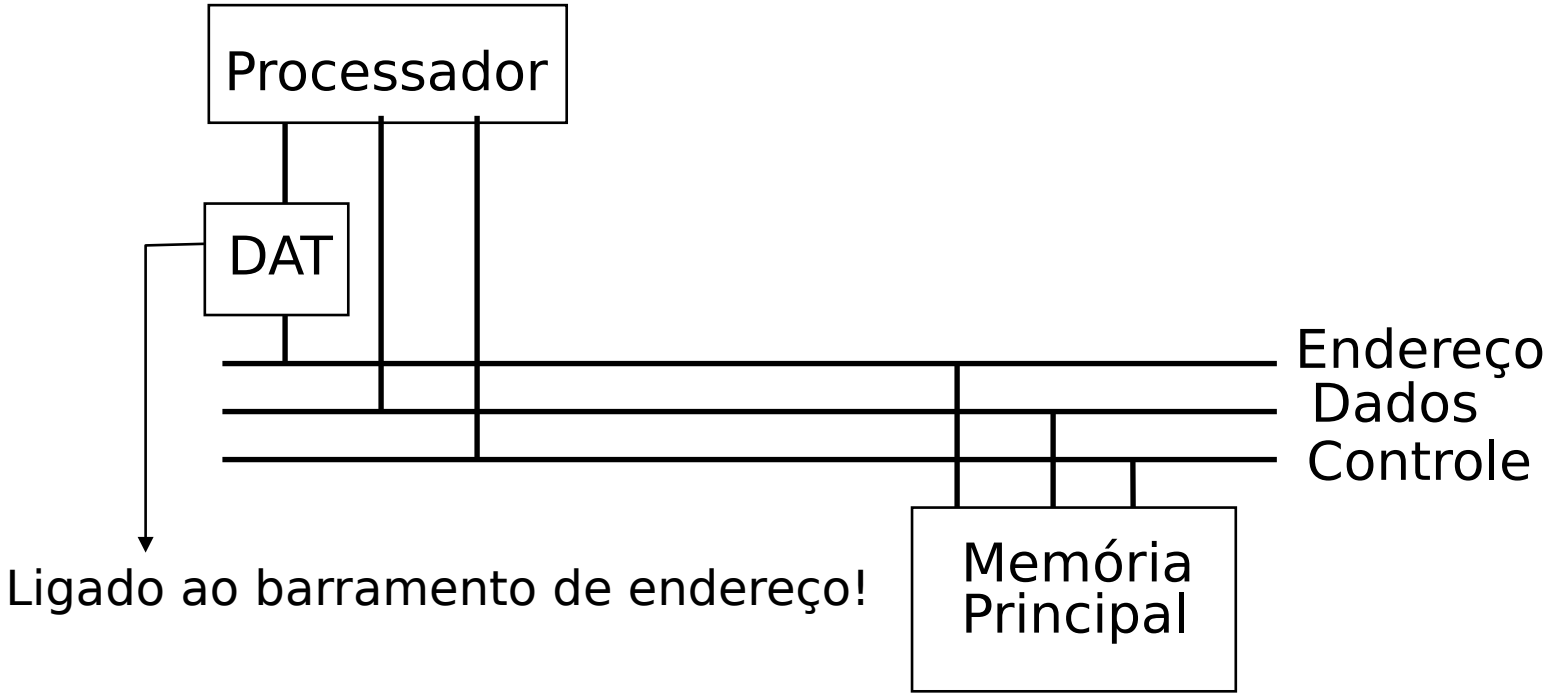
- Os possíveis endereços virtuais que podem ser gerados pelo programa formam o espaço de endereçamento virtual, enquanto a faixa de endereços na memória principal constitui o espaço de endereçamento real

Memória Virtual

- Embora sob o ponto de vista do programa as instruções e dados estejam armazenados no espaço de endereçamento virtual, na realidade eles continuam armazenados na memória principal, representada pelo espaço de endereçamento real

- Esta diferença entre endereçamento virtual e real utiliza um mecanismo que faz a correspondência entre o endereço virtual gerado pelo programa e o endereço real que será usado para acessar a memória principal, o gerenciador de memória virtual, ou DAT (*Dynamic Address Translator*)

Memória Virtual



Sub-sistema de memória

Memória Virtual

- Como a memória virtual pode ser muito maior do que a memória principal presente, pode acontecer que o bloco de dados referenciado pelo programa não esteja presente na memória principal no momento que é referenciado

- Os blocos de dados e instruções de um programa em execução ficam armazenados na memória secundária, que é uma unidade de disco do sistema

Memória Virtual

- Na ausência do bloco referenciado pelo programa no momento que o DAT realiza o mapeamento, este é carregado da memória secundária para a memória principal

- A operação de movimentação de blocos (*swapping*) envolvem as seguintes questões:
  - Quando movimentar os blocos?
  - Onde colocar um novo bloco na memória principal?
  - Qual o tamanho ideal do bloco?

Memória Virtual – Segmentação e Paginação

- Blocos de memória podem ser organizados de duas formas: em segmentos ou páginas
- O segmento pode ser formado por um bloco de informações logicamente relacionadas (por exemplo, os dados armazenados em uma matriz); os segmentos podem ter tamanho variável
- Uma página é um bloco de dados de tamanho fixo e não segue nenhuma coesão lógica, isto é, os dados podem não ter relação entre si

Segmentação e Paginação

- O DAT possui uma pequena memória interna chamada TLB (*Translation Lookaside Buffer*), que armazena os pares de endereços virtual/principal acessados recentemente, como uma memória cache

- No acesso a uma posição de memória virtual, se o dado requerido não possui um registro relacionando com a memória principal, o bloco de dados também não se encontra na memória principal. Essa ausência de bloco é denominada falha de página ou falha de segmento

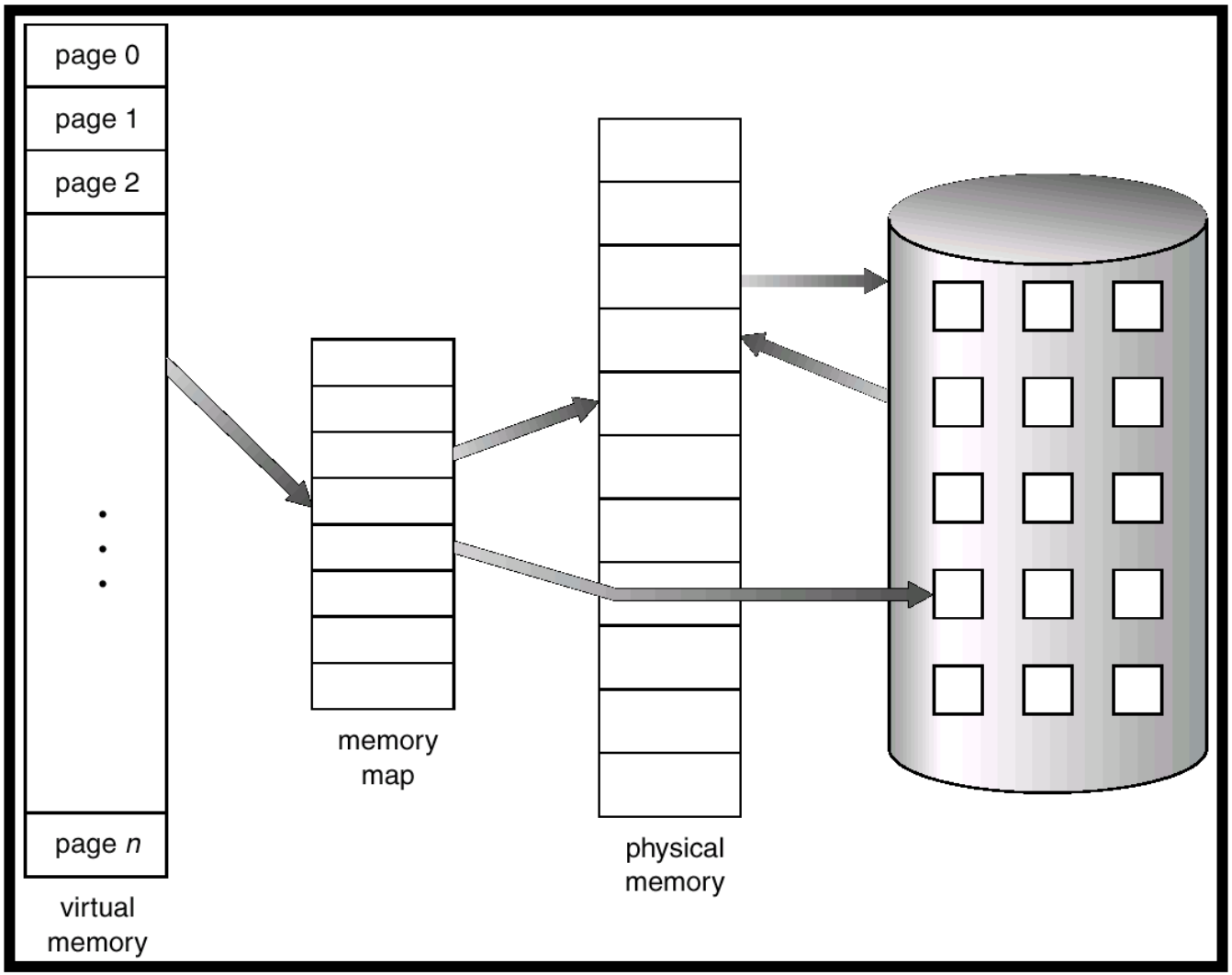
Controle de Memória Virtual

- Ao ocorrer uma falha o DAT gera uma interrupção, transferindo o controle do processador para o sistema operacional, para que este transfira o bloco de dados referenciado da memória secundária para a memória principal
- Após a transferência do bloco de dados ser concluída, o controle do processador retorna ao programa em execução, exatamente para o ponto onde foi interrompido, agora com os dados referenciados presentes na memória principal

Controle de Memória Virtual

- Com a utilização da memória virtual, não é necessário que todas as instruções e dados de um programa permaneçam na memória principal durante a execução do programa, blocos são transferidos da memória secundária para a principal a medida que forem referenciados
- É importante ressaltarmos que todo o mecanismo de memória virtual funciona de forma totalmente transparente para o programa, e mais importante, para o programador que o desenvolveu

Memória virtual maior que memória física

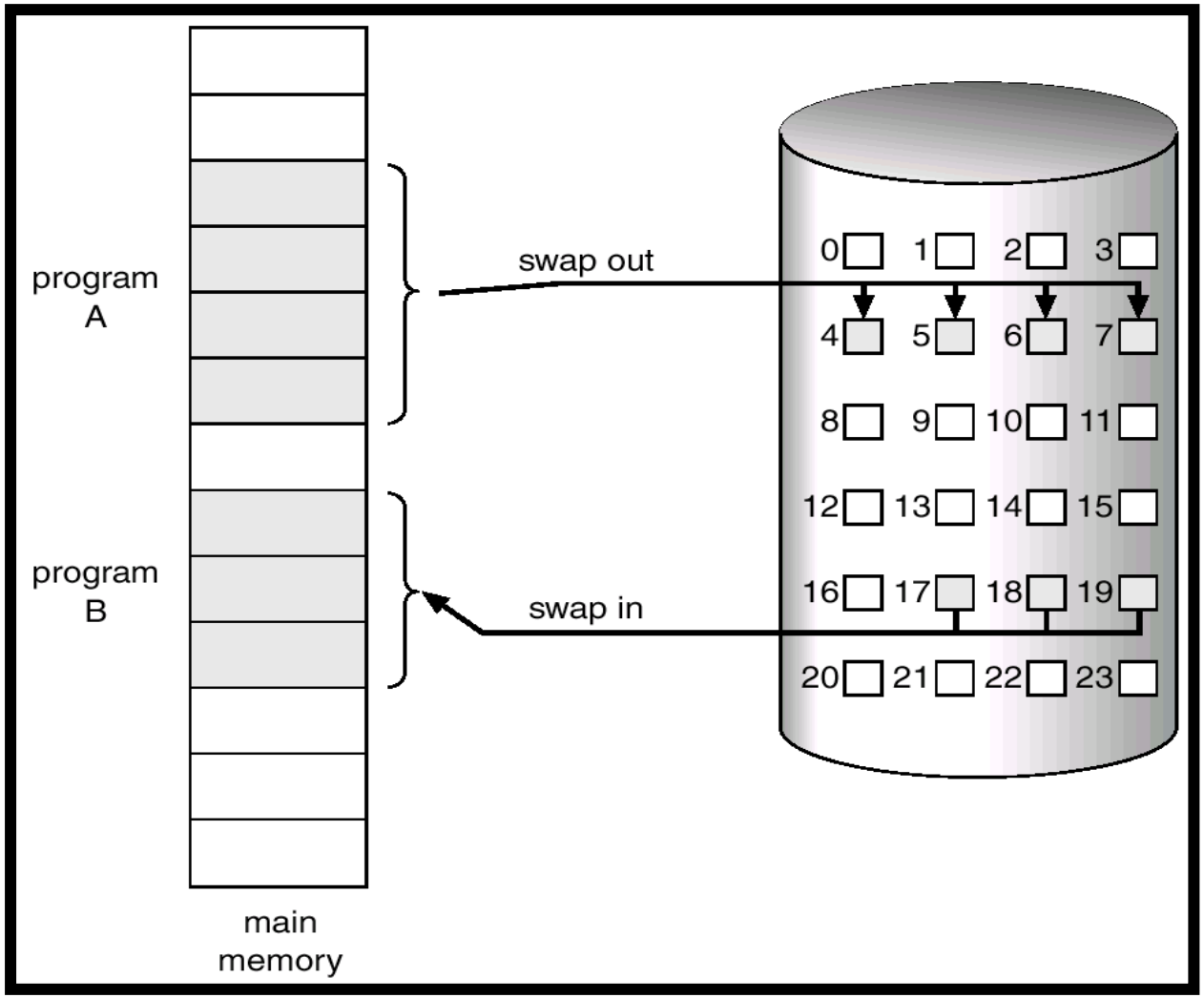


Paginação sob demanda

- Uma página é colocada na memória somente quando se necessita dela:
  - Menos E/S para carregamento e *swap* de programas;
  - Menos memória para os processos;
  - Resposta mais rápida;
  - Maior número de usuários.

- Se necessita de uma página quando uma referência é feita a ela:
  - referência inválida ⇒ aborta;
  - Não está na memória ⇒ traz para memória.

Transferência de memória paginada para espaço de disco contínuo



Bit válido-inválido

- Cada entrada da tabela de páginas possui um bit válido-inválido associado, representando: 1 ⇒ está na memória, 0 ⇒ não está na memória

- Inicialmente o bit válido–inválido é inicializado com o valor 0 em todas as entradas da tabela.

Bit válido-inválido

- Exemplo:

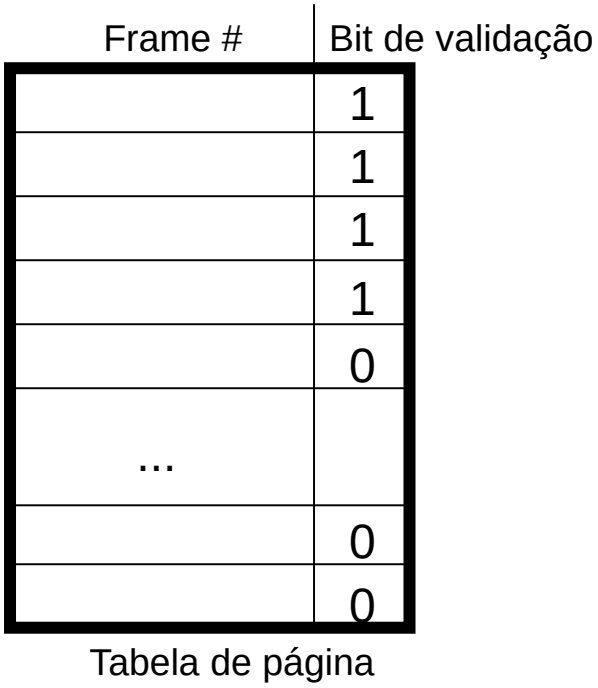
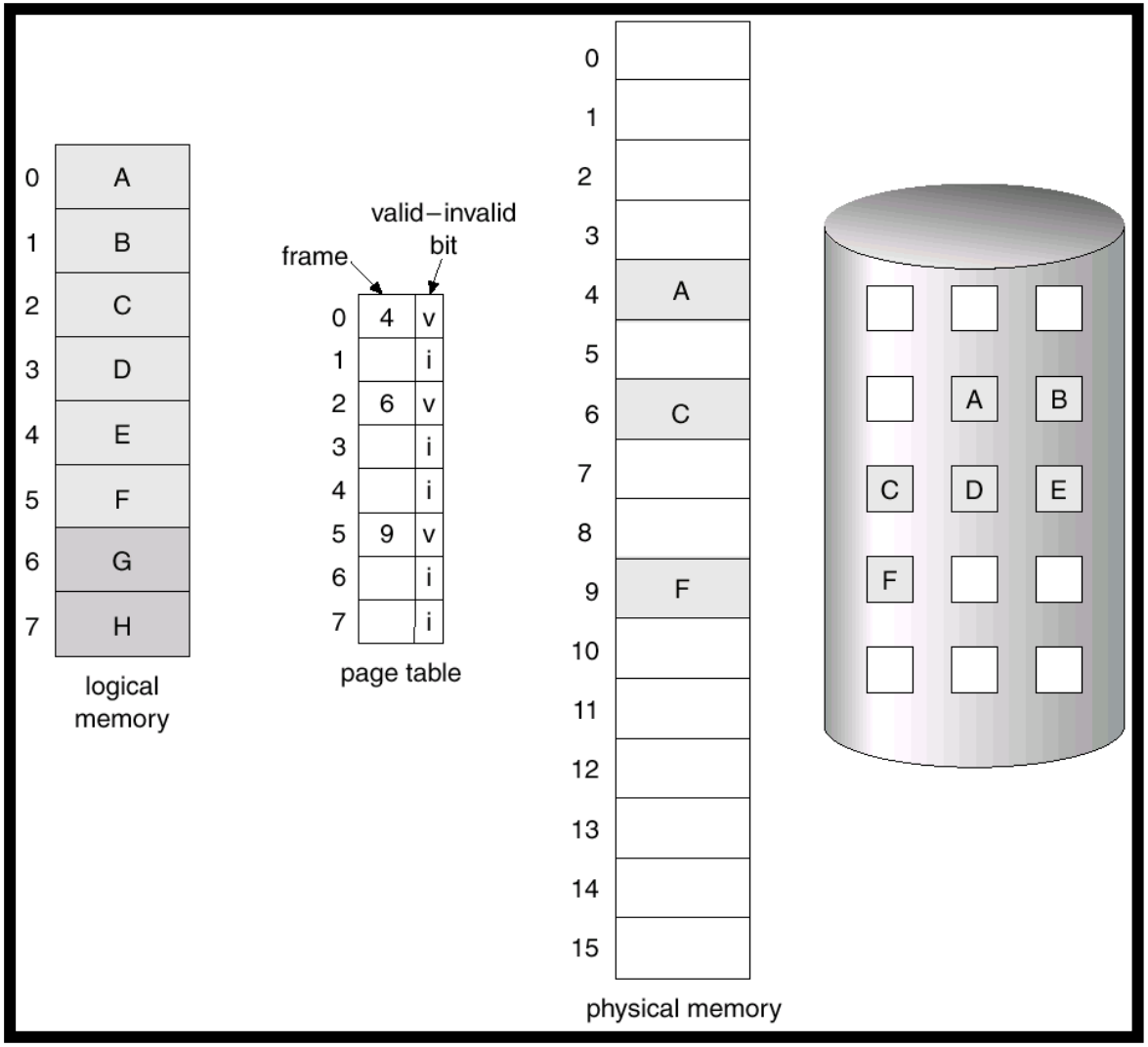


Tabela de página

- Bit válido-inválido igual a 0 quando estiver sendo realizada uma tradução de endereços ⇒ falta de página (*page fault*).

Tabela de páginas quando algumas páginas não estão na memória principal

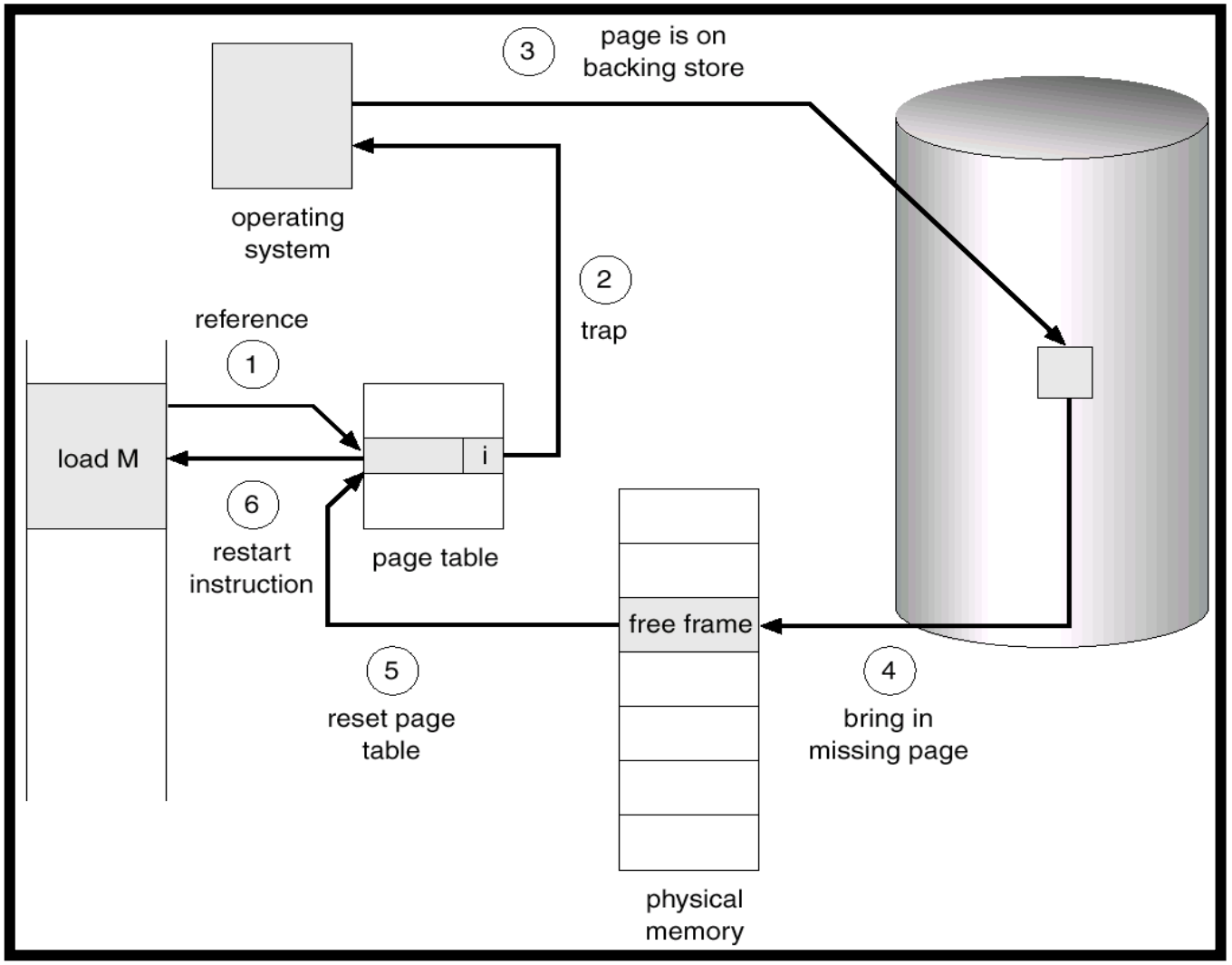


Falta de página

- A primeira referência a uma página sempre irá gerar uma exceção ao sistema operacional indicando uma falta de página

- O sistema operacional consulta uma tabela interna para verificar se a referência é inválida (processo será abortado) ou se a página não está na memória. Em seguida:
  - Obtém um frame livre;
  - Traz a página do disco para o frame;
  - Atualiza bit válido nas entradas das tabelas.

Tratando uma falta de página



O que acontece se não existir frame livre ?

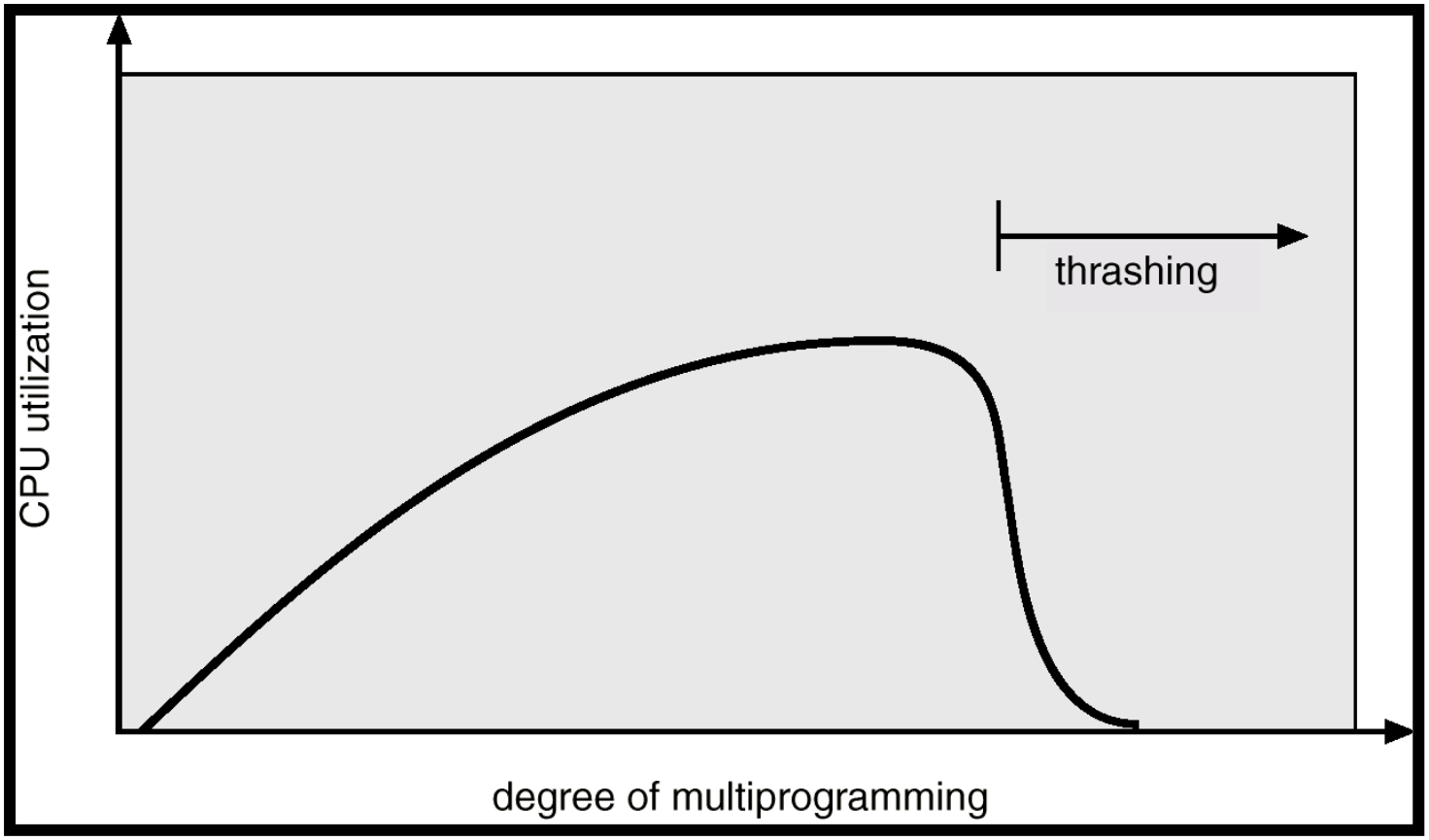
- Substituição de página– encontra uma página que está na memória mas não está sendo utilizada e a coloca no disco
  - O algoritmo utilizado deve fornecer menor número de faltas de páginas
- A mesma página pode ser trazida para a memória várias vezes

Thrashing

- Se um processo não possui páginas suficientes para ser executado na memória, a taxa de falta de páginas pode se tornar muito alta
  - baixa utilização de UCP;
  - sistema operacional acha que precisa aumentar o nível de multiprogramação;
  - permite que outro processo entre para ser executado causando mais faltas de página.

- *Thrashing*: um processo está ocupado realizando a troca de páginas entre a memória e o disco

Thrashing



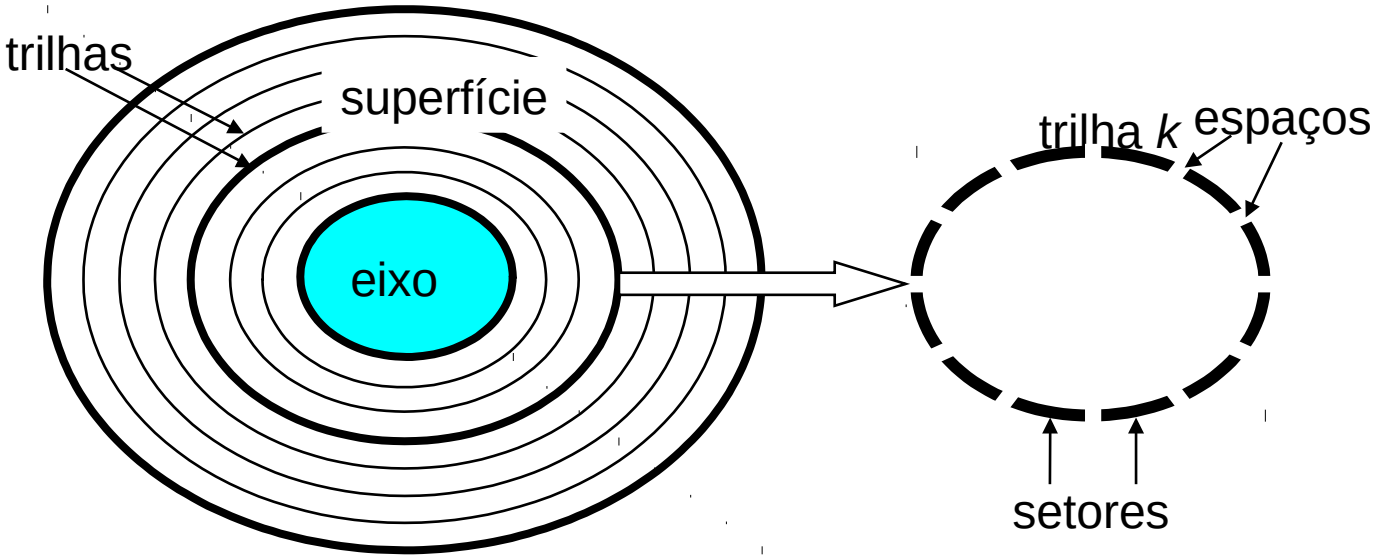
Memória Secundária

- Dispositivos de memória secundária são necessários a qualquer sistema de computação, pois nenhum computador fica ininterruptamente ligado, ou seja, os dados e instruções não podem ficar armazenados indefinidamente na memória principal

- É necessário um meio de armazenamento mais ‘permanente’ dos dados. Estes meios podem ser através de mídias magnéticas (disquetes, HD’s) ou dispositivos de armazenamento óptico (CDs, DVDs)

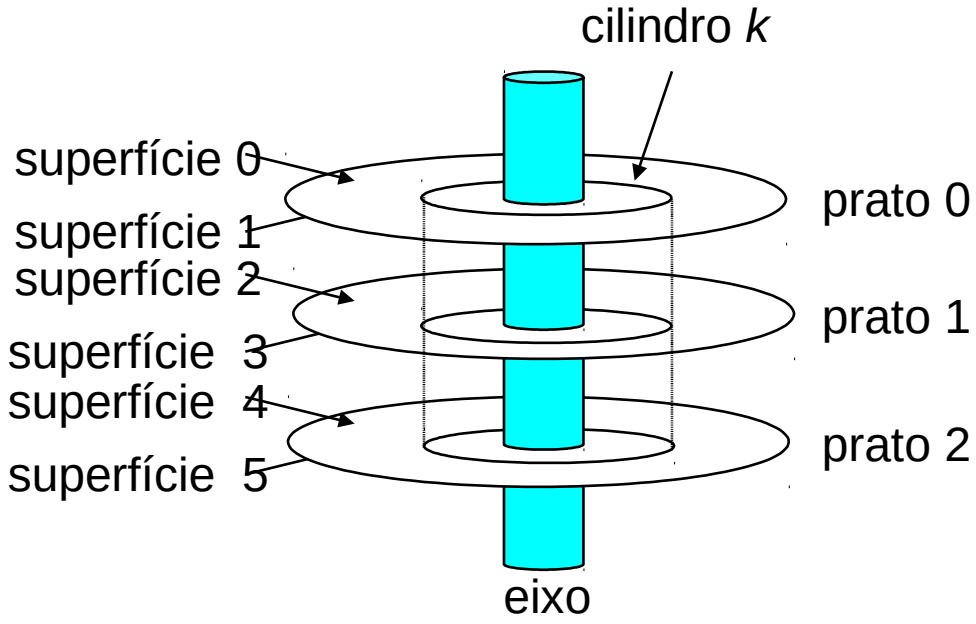
Geometria dos discos

- Um disco consiste de pratos, cada qual com duas superfícies.
- Cada superfície consiste de anéis concêntricos denominados trilhas.
- Cada trilha consiste de setores separados por espaços.



Discos com múltiplos pratos

- Trilhas alinhadas formam um cilindro.



Capacidade do disco

- Capacidade: número máximo de bits que podem ser armazenados expresso em gigabytes (1 GB = 10^9).
- Fatores que determinam a capacidade:
  - Densidade de gravação (bits/in): número de bits que podem ser gravados em 1 polegada de uma trilha.
  - Densidade de trilha(trilhas/in): número de trilhas que podem existir em um segmento radial.
  - Densidade de armazenamento (bits/in^2): produto da densidade de gravação com densidade de trilha.
  -
- Disco modernos particionam as trilhas em conjuntos disjuntos denominados zonas de armazenamento.
  - Cada trilha em uma zona possui o mesmo número de setores, determinado pela circunferência da trilha mais interna.
  - Cada zona possui um número diferente de setores/trilha.

Calculando capacidade de disco

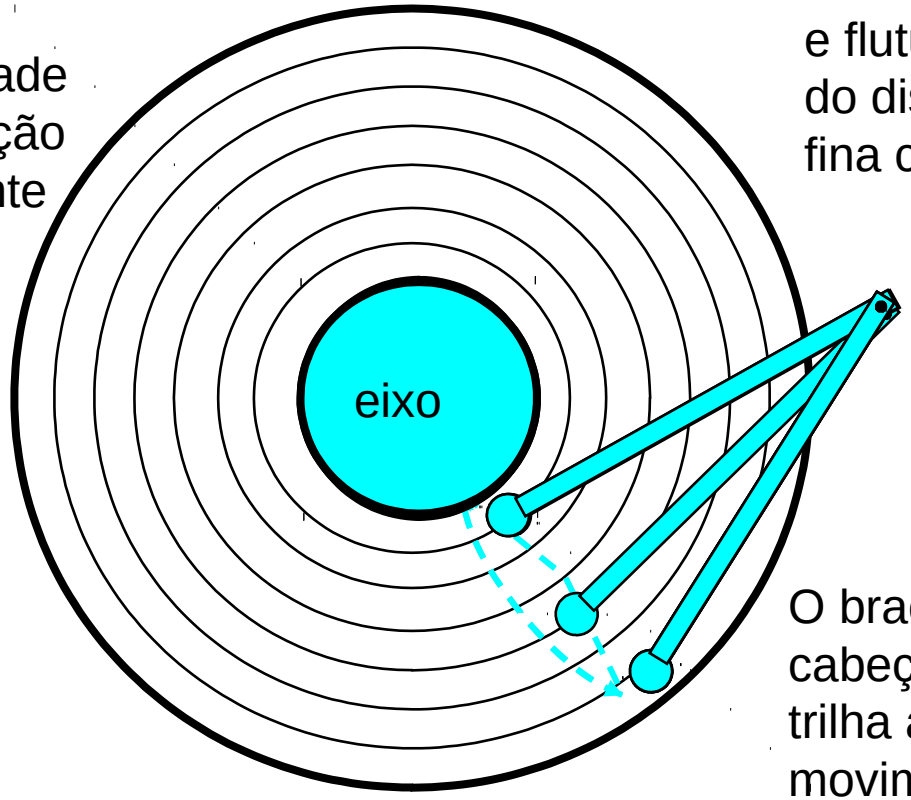
- Capacidade = (# bytes/setor) x (méd. # setores/trilha) x(# trilhas/superfície) x (# superfícies/prato) x (# pratos/disco)
- Exemplo:
  - 512 bytes/setor
  - 300 setores/trilha (em média)
  - 20.000 trilhas/superfície
  - 2 superfícies/prato
  - 5 pratos/disco

- Capacidade = 512 x 300 x 20000 x 2 x 5
  = 30.720.000.000
  = 30,72 GB

Operação do disco

A superfície do disco gira com uma velocidade de rotação constante

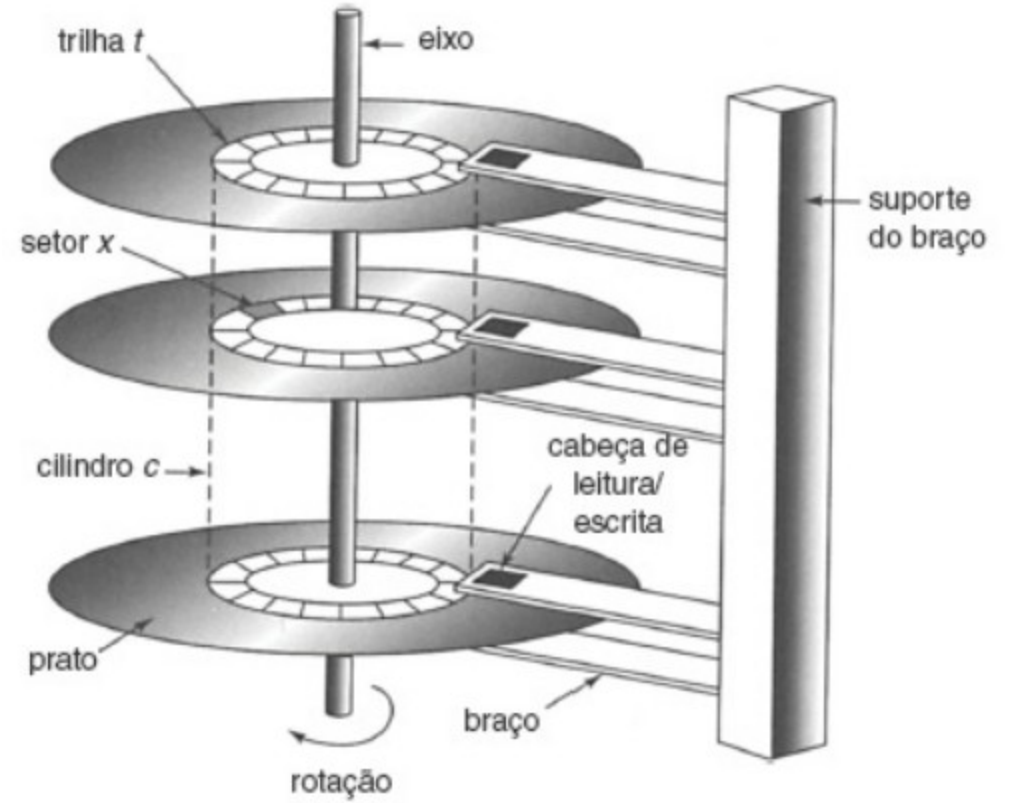
A cabeça de leitura/escrita está ligada ao final do braço e flutua sobre a superfície do disco em cima de uma fina camada de ar



O braço pode posicionar a cabeça sobre qualquer trilha através de um movimento radial

Operação de disco com múltiplos discos

Cabeças de leitura/escrita se movem em conjunto de cilindro para cilindro



Tempo de acesso ao disco

- Tempo médio de acesso a um setor desejado:
  - Tacesso = Tmed procura + Tmed rotação + Tmed transferência
- Tempo de procura (Tmed procura)
  - Tempo para posicionar as cabeças no clindro que contém o setor desejado
  - Típico Tmed procura = 9 ms
- Latência rotacional (Tmed rotação)
  - Tempo de espera para que o primeiro bit do setor passe pela cabeça
  - Tmed rotação = 1/2 x 1/RPMs x 60 seg/1 min
- Tempo de transferência (Tmed transferência)
  - Tempo para ler os bits do setor
  - Tmed transferência = 1/RPM x 1/(med # setores/trilha) x 60 segs/1 min.

Exemplo

- Dados:
  - Velocidade de rotação= 7.200 RPM
  - Tempo médio de procura = 9 ms.
  - Med # setores/trilha = 400.
- Teremos:
  - Tmed rotação = 1/2 x (60 segs/7200 RPM) x 1000 ms/seg = 4 ms.
  - Tmed transferência = 60/7200 RPM x 1/400 segs/trilha x 1000 ms/seg = 0.02 ms
  - Tacesso = 9 ms + 4 ms + 0.02 ms
- Pontos importantes:
  - Tempo de acesso dominado pelo tempo de procura e latência rotacional.
  - Tempo de acesso da SRAM é 4 ns/doubleword, DRAM por volta de 60 ns
    - Disco é aprox. 40.000 vezes mais devagar que SRAM, e 2.500 vezes mais devagar que DRAM.

Blocos lógicos

- Discos modernos apresentam uma visão abstrata mais simples da geometria complexa de setores:
  - O conjunto de setores disponíveis é modelado como uma sequência de blocos lógicos de tamanho **fixo**
- O mapeamento entre os blocos lógicos e físicos é realizado pelo *hardware/firmware* dos controladores de disco
- Permite que o controlador separe cilindros sobressalentes para cada zona

Organização do Disco – Setor de Boot

- Nem toda a área disponível no disco pode ser utilizada para a gravação de arquivos de dados, é necessário reservar certas porções para operações necessárias a qualquer sistema, como o setor de boot e tabelas de alocação de arquivos

- A BIOS (*Basic Input/Output System*) lê um setor específico do disco rígido, chamado se setor de boot mestre MBR (*Master Boot Record*), que armazena a tabela de partição, que indica qual partição do disco será usada para o boot

- O setor de boot é o primeiro setor de área reservada, e armazena o bloco de parâmetros da BIOS que descreve a partição do disco, e ainda, pode conter um programa que inicia a carga do sistema operacional, chamado de *bootstrap loader*

Organização do disco - endereçamento

- Para que o sistema operacional seja capaz de recuperar dados rapidamente de um sistema de memória de armazenamento secundário, é necessária a utilização de um sistema de endereçamento, denominado genericamente de formatação

- A formatação organiza trilhas e setores do disco em regiões onde os dados são, de fato, gravados; o tamanho destas regiões varia segundo o processo de formatação utilizado

File Allocation Table - FAT

- Toda a área de arquivos é dividida em *clusters*. Arquivos são alocados nessa área um *cluster* de cada vez, mas não necessariamente em *clusters* adjacentes. Utilizamos uma tabela de alocação de arquivos (FAT- *file allocation table*) para encadear todos os *clusters* de um arquivo

- Para cada *cluster* na área de arquivos existe uma entrada na FAT, contendo um ponteiro, que nada mais é do que um endereço de *cluster*. Desta forma, um arquivo é representado por uma cadeia de entradas na FAT, cada entrada apontando para a próxima entrada na cadeia

Organização do Disco – Clusters

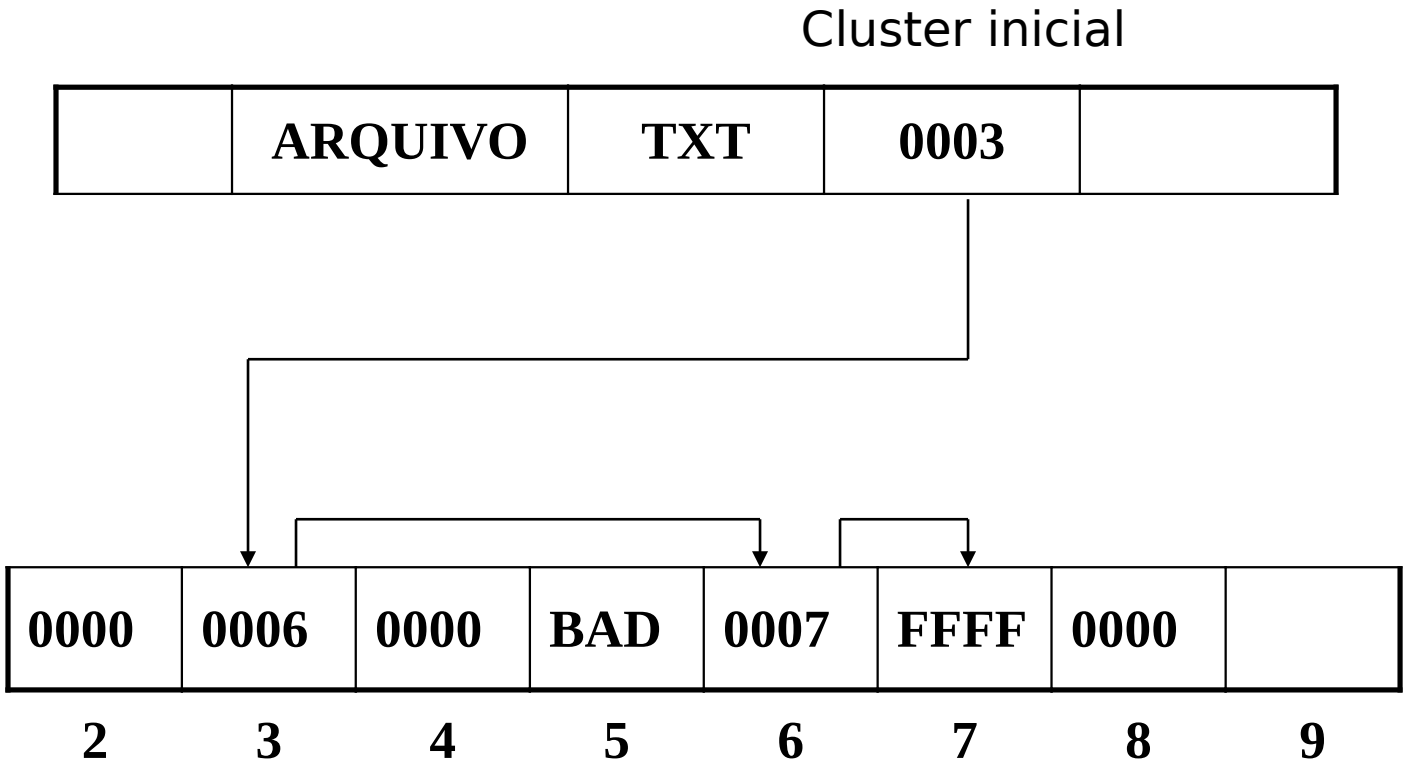
- *Cluster* é a menor unidade de área alocada pelos sistemas operacionais que trabalham com o padrão FAT nas regiões de armazenamento de arquivos. Um cluster pode ser formado por um ou mais setores

- Sempre que o sistema operacional necessita alocar uma área na região de dados, em operações como a criação de um arquivo ou um diretório, sempre é utilizado um número inteiro de *clusters*

File Allocation Table - FAT

- Dependendo da quantidade de bits utilizada no endereçamento, podemos indexar uma diferente quantidade de *clusters*
- O tamanho mínimo já adotado para um *cluster* é de 512 bytes. Em um sistema FAT-16, temos 16 bits disponíveis para o endereçamento de memória, ou seja, podemos endereçar 2^{16} = 65.536 posições diferentes. Se cada *cluster* tem 512 bytes, então 512 x 65.536 = 33.554.432 bytes, ou 32 MB
- Para endereçar uma maior quantidade de espaço em disco é necessário utilizar *cluster* de maior capacidade e um sistema de endereçamento com mais bits, como o FAT-32.

Exemplo de FAT



Organização do Disco – Clusters

- Dependendo do tamanho atribuído aos *clusters*, podemos observar duas situações que levam o sistema a apresentar um desempenho abaixo do possível:
  - ***Clusters* pequenos**: grande fragmentação de arquivos (fragmentação externa);
  - ***Clusters* grandes**: espaço ocioso em disco, pois os arquivos ocuparão pouco espaço em cada cluster (fragmentação interna).

Fragmentação

- Arquivos sofrem alterações quando uma certa quantidade de bits que os compõem são alterados, eliminados ou atualizados

- Nem sempre os espaços disponíveis podem ser reutilizados de imediato, fazendo com que a memória fique cheia de “buracos”

- Existem várias formas de alocarmos memória para a gravação de novas informações, as de maior utilização por seu desempenho e velocidade são:

Fragmentação

- A fragmentação interna ocorre quando a área mínima de gravação que definimos para arquivos não é totalmente utilizada

- A fragmentação externa ocorre quando temos muitas “sobras” de espaços, que são muito pequenas para armazenar qualquer informação

- O problema de fragmentação externa pode ser resolvido através da operação de desfragmentação, mas é computacionalmente muito custosa

NTFS

- Para discos de maior capacidade, procuramos reduzir o tamanho do cluster, para evitar a fragmentação interna, e a consequente perda de espaço de armazenamento

- O NTFS (*New Technology File System*) foi desenvolvido pela Microsoft para ser utilizado pelo Windows NT para ser um sistema de arquivos mais flexível, confiável, adaptável, seguro e veloz para operações com arquivos, como leitura, escrita e busca

- O acesso é direto ao setor físico, de 512 bytes, independente do tamanho do disco

- Com partições NTFS é ainda possível armazenar um maior volume de dados, suporte a nomes longos de arquivos e facilidades para operações de rede

Partições

- Discos podem ser divididos logicamente em espaços que chamamos de partições, com o objetivo de realizar um melhor gerenciamento de uma grande quantidade de dados
- Cada partição contém seu próprio setor de boot, o que quer dizer que cada uma pode ter seu próprio sistema operacional
- No primeiro setor físico do disco rígido fica armazenado o MBR (*Master Boot Record*), que é uma tabela de partição, onde ficam armazenadas as informações sobre as partições.

Partições

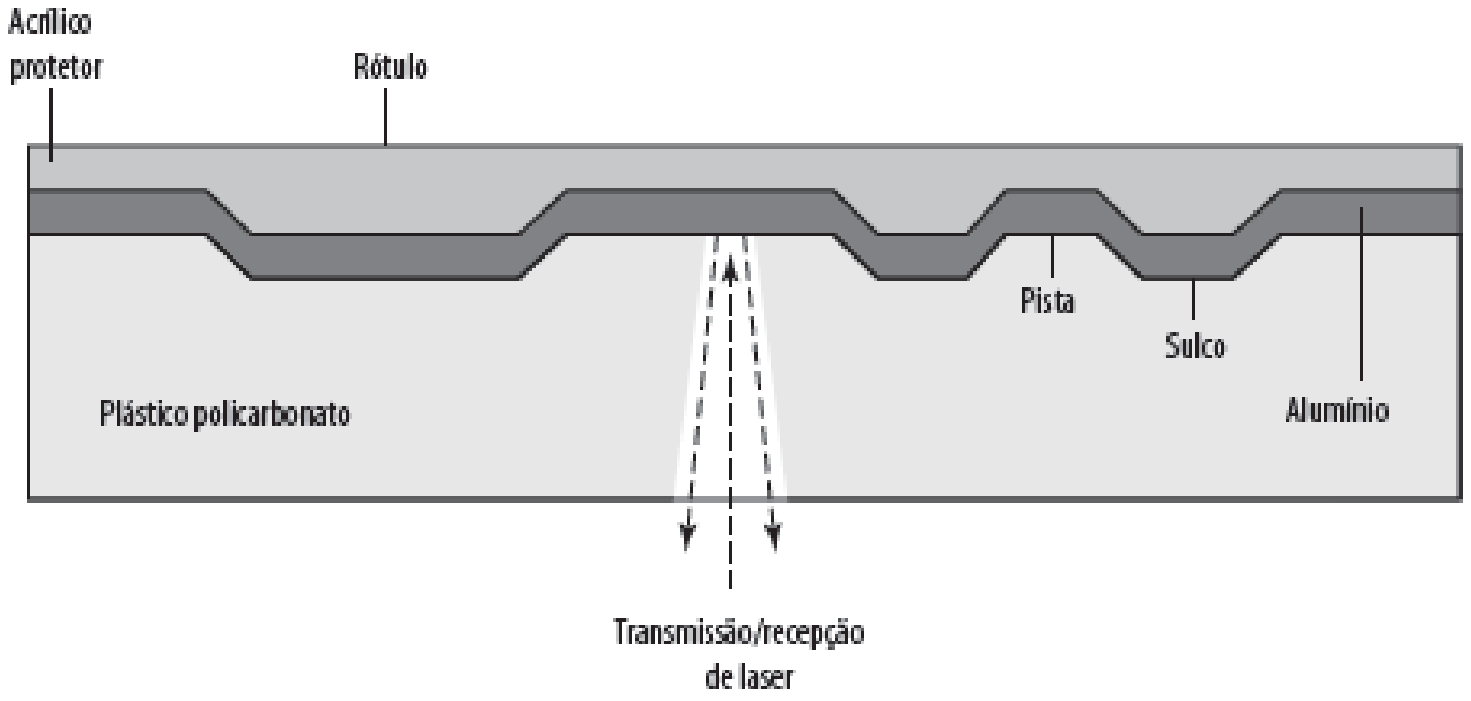
- Junto ao MBR fica armazenado um programa de pré-boot, que manipula a tabela de partição, determinando de qual partição será realizada a inicialização (*boot*) do computador. Somente uma partição pode ser considerada ativa por vez

- Partições estendidas não contém em seu primeiro setor lógico um setor de boot, mas uma tabela de particionamento secundária, contendo apenas o tamanho do disco atual e o endereço que indica o próximo disco, caso este exista

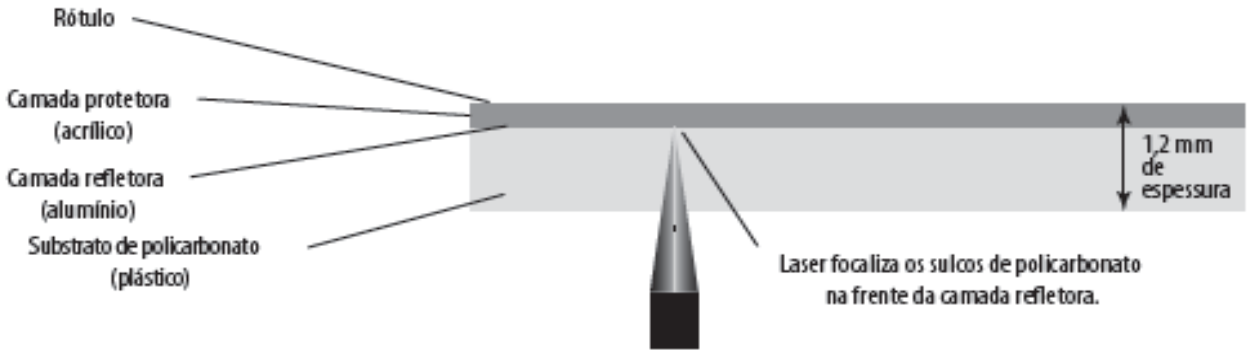
CD-ROM de armazenamento óptico

- Principais características:
  - Originalmente para áudio;
  - 650 MB gerando mais de 70 minutos de áudio;
  - Policarbonato com cobertura altamente reflexiva, normalmente alumínio;
  - Dados armazenados como sulcos, na forma de uma espiral, que abrange toda a estrutura física do disco;
  - Lidos pela reflexão do laser;
  - Densidade de empacotamento constante;
  - Velocidade linear constante.

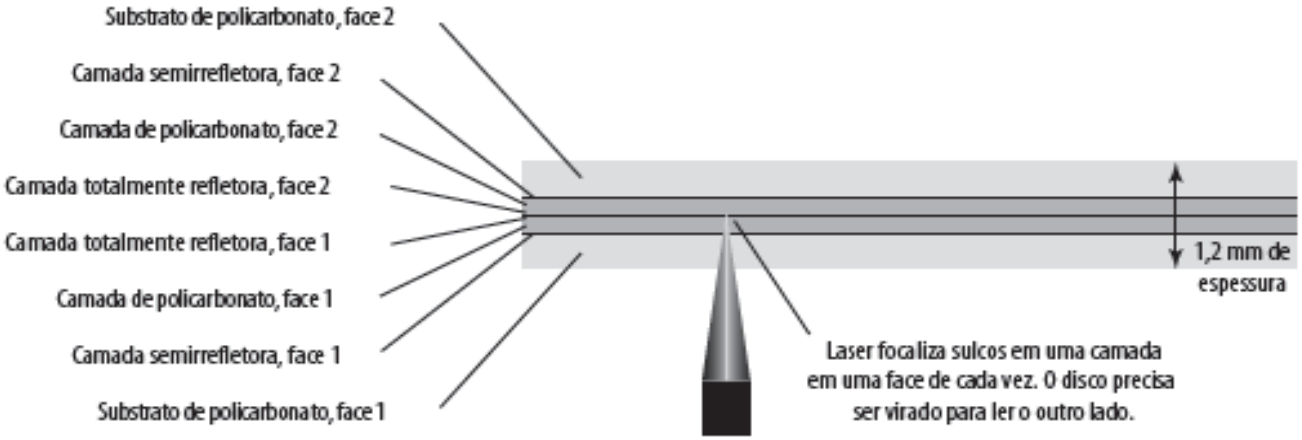
Estrutura física



CD e DVD



(a) CD-ROM — Capacidade de 682 MB

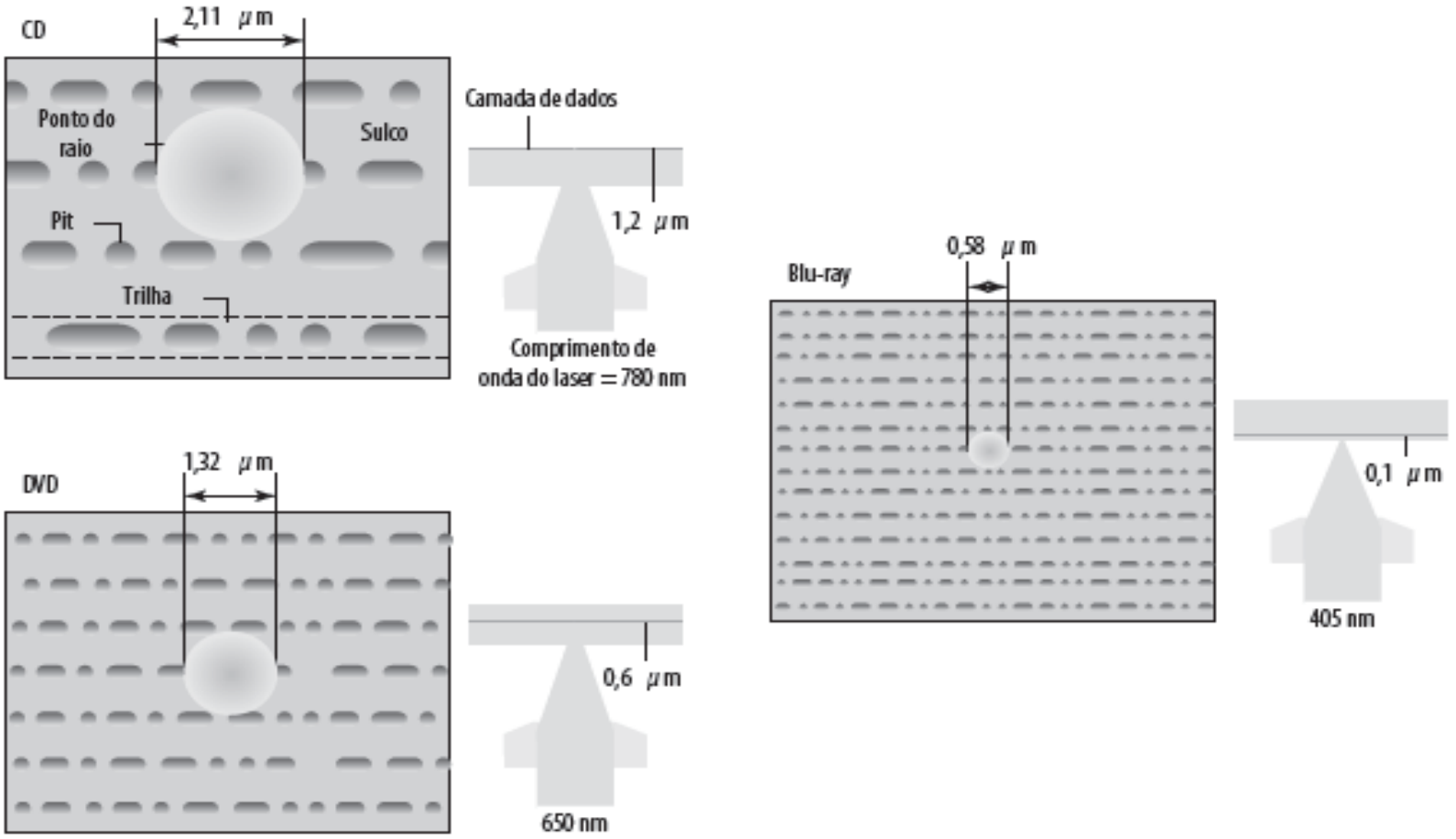


(b) DVD-ROM, dupla face, dupla camada — Capacidade de 17 GB

Discos ópticos de alta definição

- Projetados para vídeos de alta definição.
- Capacidade muito mais alta que DVD.
  - Laser com comprimento de onda mais curto.
    - Faixa do azul violeta.
  - Sulcos menores.
- HD-DVD:
  - 15 GB de único lado, única camada.
- Blu-ray:
  - Camada de dados mais próxima do laser.
    - Foco mais estreito, menos distorção, sulcos menores.
  - 25 GB em única camada.
  - Disponível para apenas leitura (BD-ROM), regravável uma vez (BR-R) e re-regravável (BR-RE).

Características da memória óptica



Fita magnética

- Acesso serial;
- Lenta;
- Muito barata;
- Ainda muito utilizada para backup e arquivamento;
- Forma mais moderna: unidades de fita Linear Tape Open (LTO)
  - Desenvolvida no final da década de 1990.
  - Alternativa de fonte aberto para os diversos sistemas de fita patenteados.

Unidades de fita *Linear Tape Open* (LTO)

	LTO-1	LTO-2	LTO-3	LTO-4	LTO-5	LTO-6	LTO-7	LTO-8	*Type M-8*[Note 1]	LTO-9	LTO-10	LTO-11	LTO-12
Release date	2000[5]	2003	2005	2007	2010[6]	Dec. 2012[7]	Dec. 2015[8][9][10]	Dec. 2017		*TBA*	*TBA*	*TBA*	*TBA*
Native/raw data capacity	100 GB	200 GB	400 GB	800 GB	1.5 TB[11]	2.5 TB[12]	6.0 TB[10][13]	12 TB[14]	9 TB	24 TB[11][15]	48 TB[11]	96 TB[11]	192 TB[11]
compressed capacity	200 GB	400 GB	800 GB	1.6 TB	3.0 TB	6.25 TB	15 TB	30 TB	22.5 TB	60 TB	112.5 TB	240 TB	480 TB
Max uncompressed speed (MB/s)[13][Note 2]	20	40	80	120	140	160	300[16]	360	300	*708*	*1,100*	*TBA*	*TBA*
Max compressed speed (MB/s)	40	80	160	240	280	400	750	900	750	*1,770*	*2,750*	*TBA*	*TBA*
Time to write a full tape at max uncompressed speed(hh:mm)	1:25	1:25	1:25	1:50	3:10	4:35	5:55	9:15	8:20	*TBA*	*TBA*	*TBA*	*TBA*
Compression capable?	Yes, "2:1"					Yes, "2.5:1"				Planned, "2.5:1"[15][17]			
WORM capable?	No		Yes							Planned			
Encryption capable?	No			Yes						Planned			
Max. number of partitions	1 (no partitioning)				2	4				Planned			

1. ^ Previously unused LTO-7 tape, not an independent generation, part of LTO-8 generation. See: Compatibility
2. ^ Maximum uncompressed speeds valid for full height drives. Half height drives may not attain the same speed. Check manufacturer's specifications.