

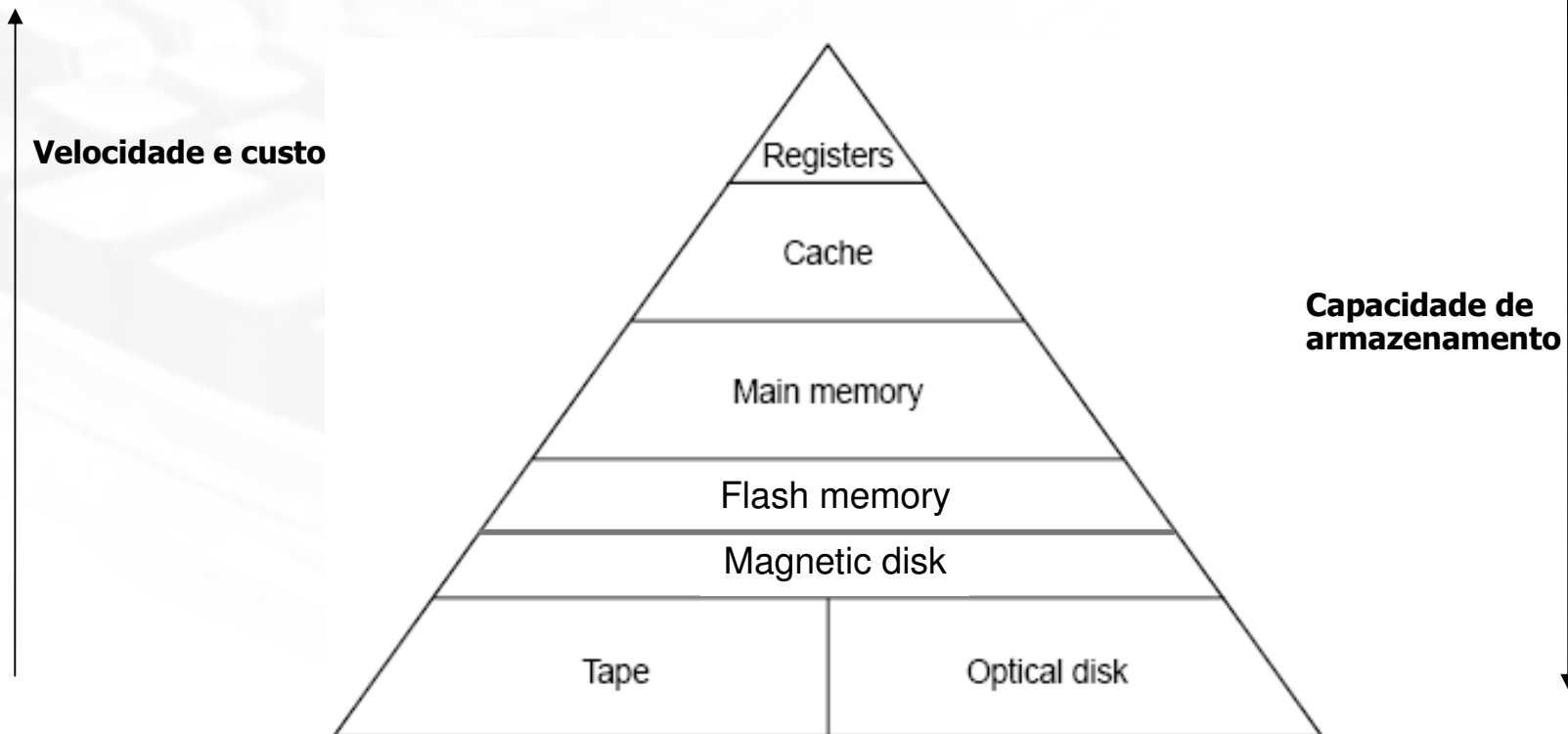


# Organização de Computadores

(Aula 4)

## Memória

# Hierarquia de Memória (1)



## Hierarquia de Memória (2)

- A memória *cache* é geralmente controlada por hardware
- A memória principal (RAM) e a secundária é que o usuário tem acesso.
- O sistema operacional através de um mecanismo de **Memória Virtual (Segmentação e/ou Paginação)** cria a “ilusão” ao usuário que a memória total é do tamanho da memória principal + memória secundária.
- A técnica de memória virtual realiza transferência de blocos de informação entre a memória primária e secundária automaticamente sem a intervenção do usuário comum.

# Registradores

- São dispositivos (elementos computacionais) capazes de receber dados, mantê-los armazenados por um curto período de tempo e transferi-los para outro dispositivo.
  - São, portanto, elementos de armazenamento temporário.
- Os registradores fazem parte da CPU.
- São extremamente rápidos e armazenam grupos reduzidos de bits.

## Memória Principal <sup>(1)</sup>

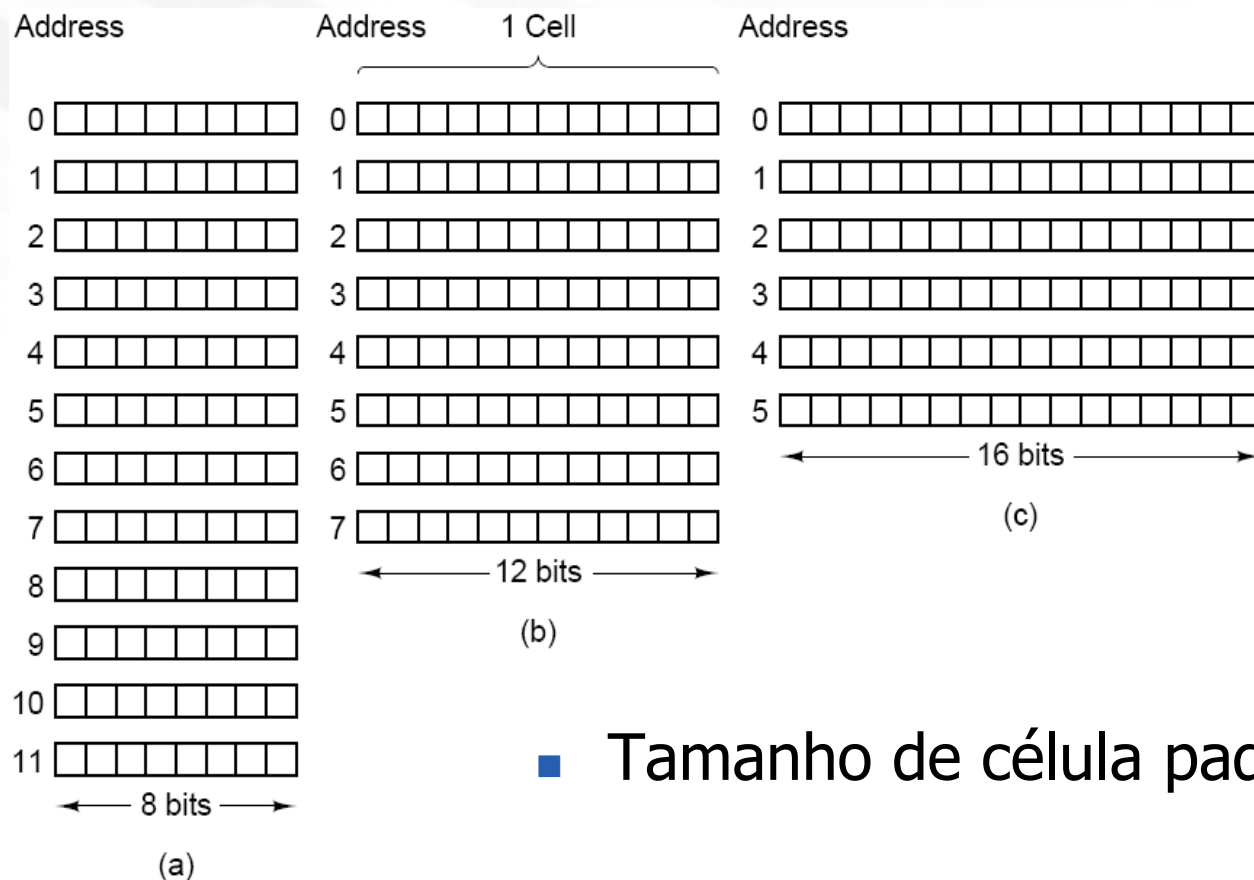
- A memória é a parte do computador em que os programas e os dados são armazenados.
- A memória principal (MP) armazena programas em execução e os dados utilizados por eles.
- Sem uma memória na qual processadores possam ler ou escrever informações, o conceito de computador digital com programa armazenado não pode ser implementado.
  - A CPU processa instruções que são obtidas na MP e os resultados são retornados à MP.

## Memória Principal (2)

- A unidade básica de memória é o **bit** (*binary digit*)
  - Abstração de valores 0 ou 1
  - Fisicamente é mais fácil distinguir entre dois valores distintos do que de mais valores. Tensão, corrente, ...
- A memória é formada por um conjunto de **células** (ou **posições**), cada uma das quais podendo guardar uma informação.
  - Todas as células de uma dada memória têm o mesmo número de bits
- Os números que identificam (referenciam) a posição da célula na memória são chamados de **Endereços**.
  - **A célula é a menor unidade endereçável da memória**
- Endereços são indexadores pelos quais os programas podem referenciar dados na memória.

## Memória Principal (3)

- A memória é formada por um conjunto de **células**
  - Todas as células de uma memória têm o mesmo nº de bits
- Cada célula tem um **endereço**

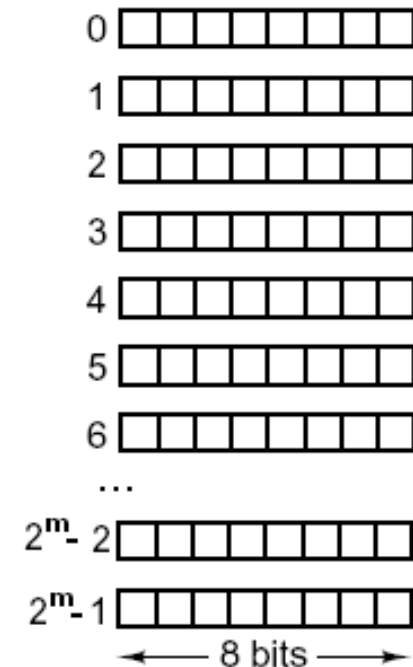


Como  
organizar uma  
memória de  
96 bits?

- Tamanho de célula padrão: 8 bits ... 1 **byte**

## Memória Principal (4)

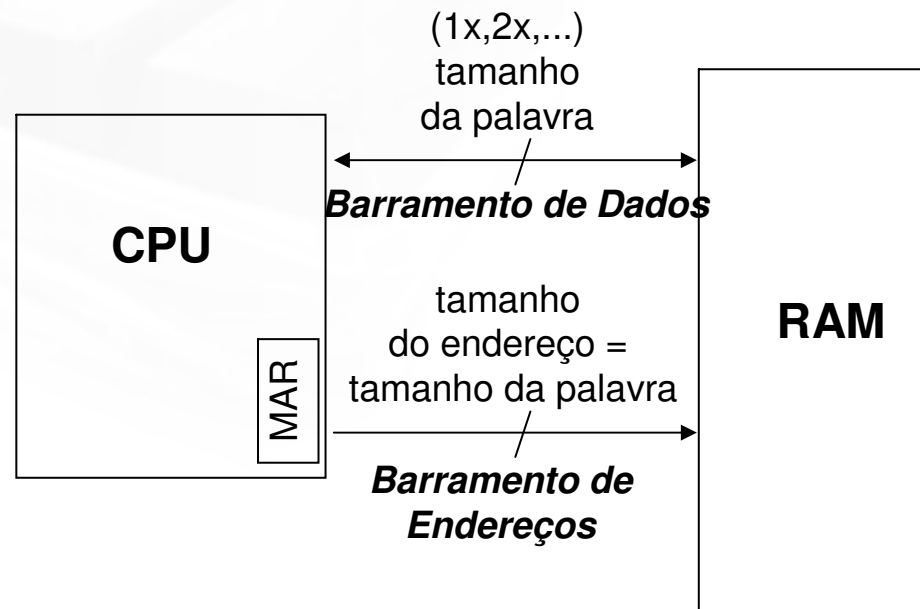
- Se a memória tiver  $n$  células, elas terão endereços de 0 a  $(n - 1)$ .
- Se uma célula tiver  $k$  bits, ela poderá armazenar qualquer uma das  $2^k$  combinações possíveis para os bits.
- Se um determinado endereço tem  $m$  bits, o número máximo de células endereçáveis é de  $2^m$ .





## Memória Principal (5)

- Os computadores modernos agrupam as células (ou bytes) em Palavras (*word*)
  - Ex: uma palavra de 32 bits tem 4 bytes (ou 4 células)
- Nesses computadores, a Palavra é a parte mínima de dados que podem ser transferidos de/para a memória (MP)
- A informação na palavra pode ser um dado ou uma instrução



## Memória Principal (4)

- **“Processadores de 32 bits” têm palavras de 32 bits**
  - **Os registradores são de 32 bits**
    - nº de bits do barramento de endereços em geral (mas não obrigatoriamente) é igual ao nº de bits dos registradores (ex: *Memory Address Register* - MAR)
  - **As instruções são (em geral) de 32 bits**
  - **Cada instrução deve tratar palavras de 32 bits**
    - **movimentar, somar subtrair... dados armazenados em registradores de 32 bits**

## Memória Principal (5)

- Os bytes de uma palavra podem ser ordenados na memória da esquerda para a direita OU da direita para a esquerda
- **Big Endian:** Os bytes mais significativos primeiro (Mac)
- **Little Endian:** Os bytes menos significativos primeiro (Intel)

00000000 00000000 00000100 00000001

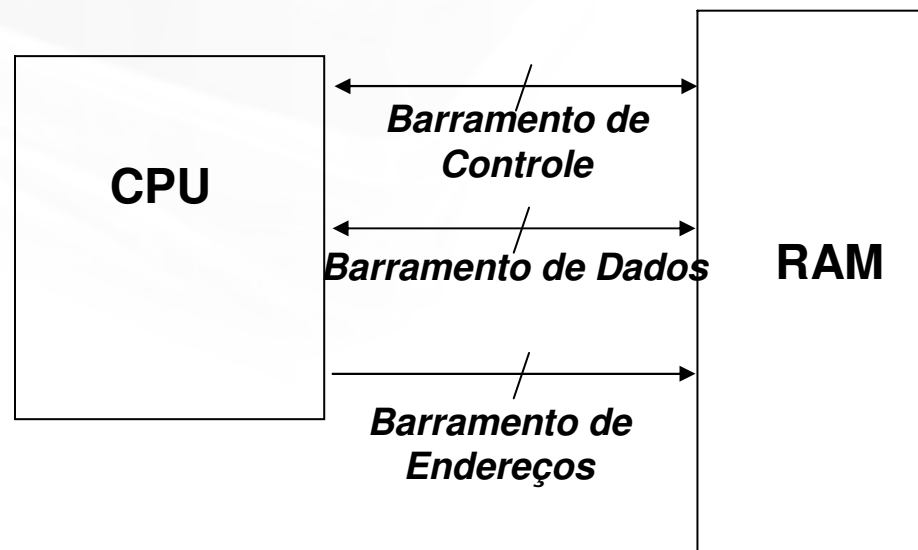
Address	Big-Endian representation of 1025	Little-Endian representation of 1025
00	00000000	00000001
01	00000000	00000100
02	00000100	00000000
03	00000001	00000000

## Memória Principal <sup>(6)</sup>

- O endereço identifica a palavra na memória, e é usado tanto para armazenar (*store*) como para carregar (*load*).
- **Em geral, registradores são usados para endereçar a memória**  
**tamanho do barramento de endereços = tamanho da palavra**
- No projeto de cada computador é definida a organização da memória (tamanho do barramento de endereços, tamanho da palavra e da célula de memória, etc.)
  - Exemplos de memória:
    - 64K ( $2^{16}$ ) X 8 bits (65 536 posições de 8 bits = 64 kilobytes)
    - 4 Giga ( $2^{32}$ ) X 8 bits (4 294 967 296 posições de 8 bits)
    - 1 Mega ( $2^{20}$ ) X 16 bits (1 048 576 posições de 16 bits)
  - Qual o máximo de memória endereçável de um computador de 64bits?
    - célula = 1 byte

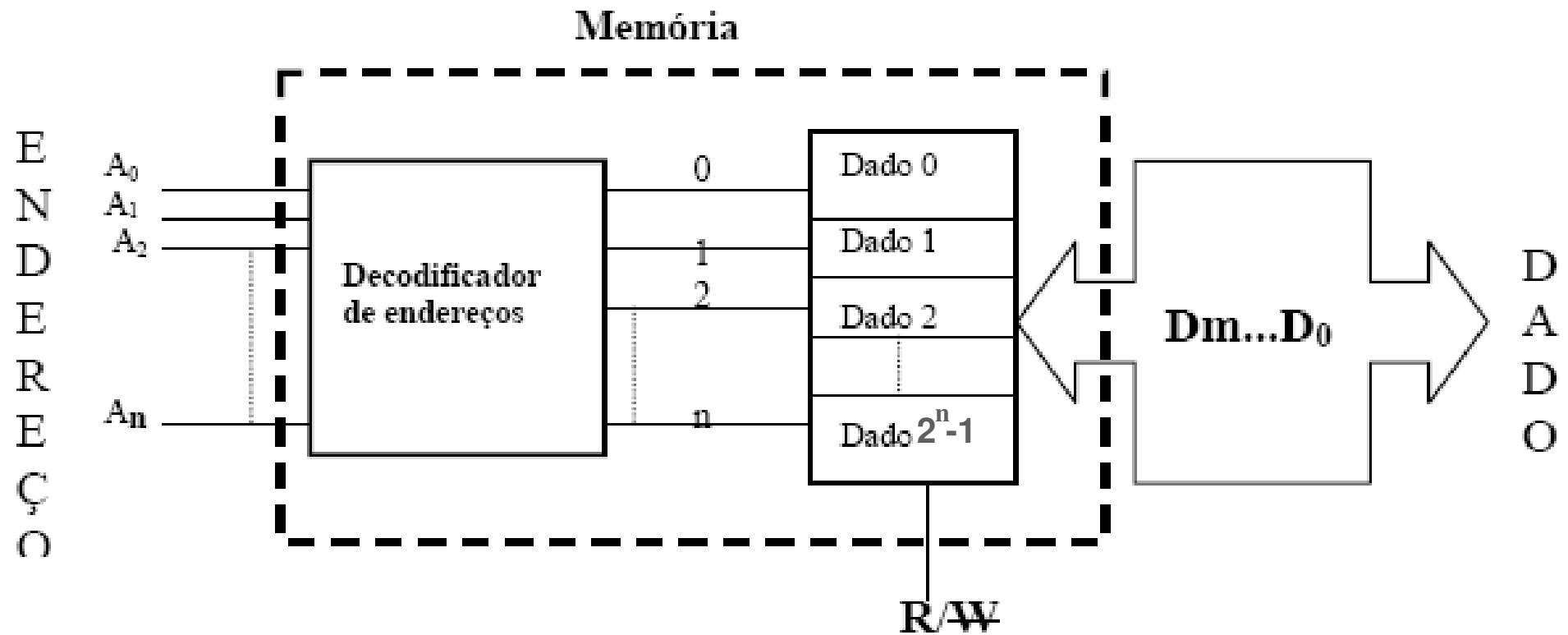
## Memória Principal (7)

- **Leitura:** Quando um valor é recuperado da memória, o conteúdo da palavra não é alterado. Apenas uma cópia será enviada pela memória.
- **Escrita:** A gravação de um novo conteúdo em uma palavra se dá com a destruição do conteúdo anterior.



## Memória Principal <sup>(8)</sup>

- Diagrama esquemático da organização da memória
  - Para simplificar: **1 palavra = 1 célula**



## Memória Principal <sup>(9)</sup>

- Uma memória com  $n$  linhas de endereços e  $m$  linhas de dados.
- As linhas de endereços ( $n$  bits) permitem endereçar  $2^n$  diferentes células de memória.
- O tamanho do dado, contido em cada célula de memória é de  $m$  bits.
- Cada código de endereço de  $n$  bits seleciona, através do decodificador de endereços, apenas uma única célula na memória.
- O sentido do fluxo de dados dependerá do **signal de controle R/W**.
  - Se  $R/W = 1$ , a operação é de READ, ou seja, leitura na memória e o dado sairá da memória
  - Se  $R/W = 0$ , a operação será de escrita (WRITE) e o dado entrará na memória.

## Tipos de Memória (1)

- **Memória Volátil:** É aquela cuja informação original é perdida se a energia for desligada.
  - **Memória Não-volátil:** É aquela que retém o padrão de bits original mesmo que a energia seja desligada.
- 
- **Memória Endereçada Seqüencialmente:** Para se obter a informação de um endereço, é necessário percorrer os endereços anteriores.
    - **Ex.: Fita Magnética**
  - **Memória de Acesso Randômico (RAM – *Random Access Memory*):**
    - A célula pode ser acessada sem ter que percorrer os endereços anteriores.
    - O tempo de acesso é praticamente o mesmo para todas as células
    - Volátil



## Tipos de Memória (2)

### ■ RAMs Estáticas (SRAM)

- Baseadas em *flip-flops*
- Conteúdo persiste enquanto circuito alimentado
- Mais rápidas (geralmente usadas como memória cache)

### ■ RAMs Dinâmicas (DRAM)

- Baseadas em capacitores
- Carga deve ser restaurada periodicamente, pois suas informações vão desaparecer após um certo intervalo de tempo
- Menores, consomem menos potência, mais baratas
- SDRAM (Synchronous Dynamic Random Access Memory)
  - **Módulo SDR (*Single Data Rate*)**: transfere um dado por pulso de clock
  - **Módulo DDR (*Double Data Rate*)** : tráfego é de dois dados por pulso de clock (transfere dados tanto na subida quanto na descida do sinal de clock)
    - **DDR-II**: melhorias no padrão

## Tipos de Memória (3)

### ■ **ROM (*Read-Only Memory*)**

- Simples: decodificador, linhas de saída e portas lógicas
- Aplicações de alto volume

### ■ **PROM (*Programmable ROM*)**

- Conteúdo escrito com um “queimador” de PROMs. Podem ser escritas com dispositivos especiais mas não podem mais ser apagadas.

### ■ **EPROM (*Erasable PROM*)**

- Podem ser apagadas e reutilizadas pelo uso de radiação ultravioleta.
- Antigos chips de BIOS (Basic Input/Output System) de PC

### ■ **EEPROM (*Electrically Erasable PROM*)**

- Conteúdo pode ser modificado eletricamente
- Pode ser lida um número ilimitado de vezes, mas só pode ser apagada e programada um número limitado de vezes (entre 100.000 e 1 milhão).
- Pelo menos 64 vezes menores que uma EPROM
- Armazenam a BIOS em PC atuais

# Tipos de Memória (4)



## ■ Memória Flash

- Como a EEPROM, mas que permite que múltiplos endereços (blocos) sejam apagados ou escritos numa só operação.
- Players MP3, celulares, câmeras digitais, ... "HD" p/ Laptops!?!

## ■ Encapsulamento de Memória Principal

### ■ SIMM – *Single Inline Memory Module*

- Uma linha (em apenas um lado) de conectores da unidade de memória para a placa de circuito impresso
- Adequadas p/ barramento de dados de 32 bits



SIMM (72 pinos)

### ■ DIMM – *Dual Inline Memory Module*

- Uma linha de conectores da unidade de memória para a placa de circuito impresso, em ambos os lados da placa (pente) de memória
- Muito usadas c/ Pentiums (64 bits no barramento de dados)



DIMM DDR 1GB 400 MHz (184-pinos)

## Códigos com Correção/Detecção de Erros <sup>(1)</sup>

- Os dados armazenados na memória dos computadores podem ocasionalmente serem alterados (modificados)
  - Oscilações de tensão de alimentação, etc.
- Para a prevenção desses erros, algumas memórias armazenam informações extras, usando **códigos** e **mecanismos** que permitam a detecção e/ou correção de erros.
- **Palavra de Código:**  
uma unidade de  $n$  bits =  $m$  bits de dados

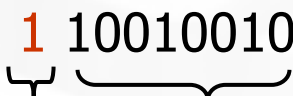
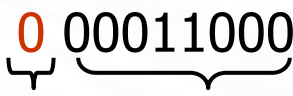
+

$r$  de redundância

{ Informação referente  
à detecção/ correção  
de erro

## Códigos com Correção/Detecção de Erros (2)

### EXEMPLO

- **Bit de Paridade:** bit configurado em 0 ou 1 para assegurar que o número total de bits 1 no campo de dados é par (ou ímpar).
  - Paridade refere-se ao número de bits '1' no número binário.
  - Ex:   
bit de paridade    dado
  -   
bit de paridade    dado

## Códigos com Correção/Detecção de Erros <sup>(3)</sup>

- **Distância de Hamming:** número de bits que diferem em duas palavras de um código qualquer.

10001001

10110001 XOR

00111000

- Distância de Hamming do código = menor distância de Hamming entre duas palavras
- Se um código possui distância de Hamming de **d**, ele é capaz de
  - detectar **d-1** erros
  - corrigir **mod [(d-1)/2]** erros

# Códigos com Correção/Detecção de Erros (4)

Tabela de palavras  
de código válidas  
(bit de paridade)

0	000
1	001
1	010
0	011
1	100
0	101
0	110
1	111

Distância de Hamming do código = 2



Código capaz de detectar **1** erro  
Capaz de corrigir **0** erros

Tabela de palavras  
de código válidas  
(código hipotético)

00	000
01	011
10	110
11	101

Distância de Hamming do código = 3



Código capaz de detectar **2** erros  
Capaz de corrigir **1** erro

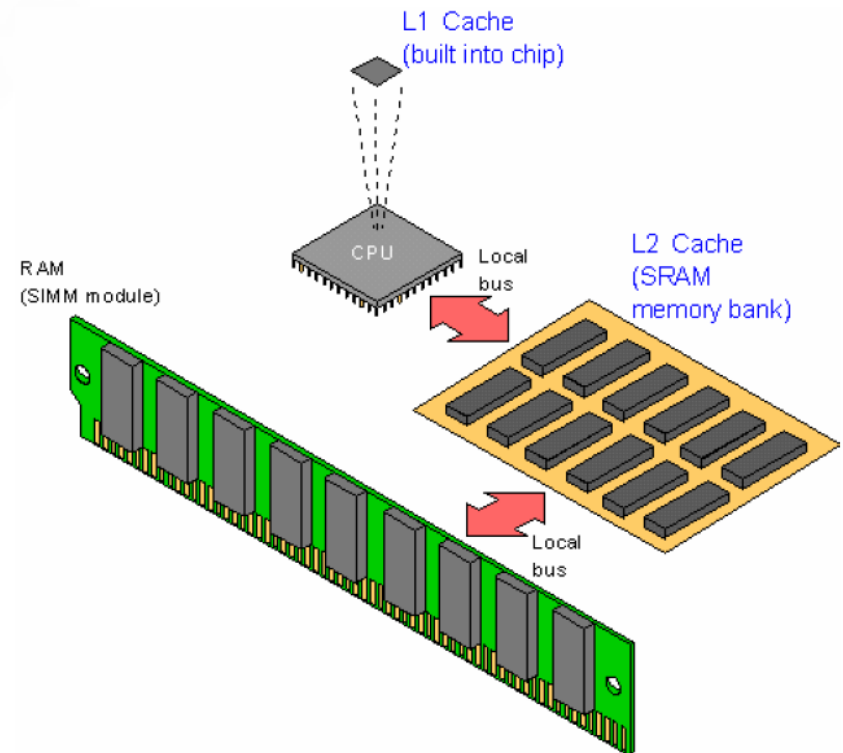
# Memória *Cache* <sup>(1)</sup>

- Princípio básico
  - Na execução de um programa de computador, muitas das referências são a um pequeno conjunto de posições de memória.
    - Muitos comandos presentes em *loops*.
    - Manipulação de matrizes
- **Cache** é um dispositivo interno a um sistema que serve de intermediário entre uma CPU e o dispositivo principal de armazenamento (MP).
- A idéia principal é que o acesso a MP pode ser demorado e vale a pena armazenar as informações mais procuradas em meio mais rápido.
- **Memória *Cache*:** memória pequena (capacidade de armazenamento) e rápida
  - Contém os dados e/ou instruções mais recentemente referenciados pelo processador.



## Memória *Cache* (2)

- Quando um processador precisar de uma palavra de memória, ele primeiro busca essa palavra na *cache*.
- Somente no caso de ela não estar armazenada na *cache* é que a busca se dará na memória principal.
- Se uma parte substancial dos acessos for satisfeita pela *cache*, o **tempo médio** de acesso a uma palavra em memória será pequeno, próximo ao tempo de acesso à *cache*.
- Em alguns computadores podem existir diversos níveis de *cache*
  - Ex: nível 1 é implementado dentro do chip ; nível 2 implementado na placa-mãe



# Memória *Cache* <sup>(3)</sup>

## ■ Princípio da Localidade

### ■ Localidade *Temporal*

- Uma posição de memória referenciada recentemente tem boas chances de ser referenciada novamente em um futuro próximo
- Iterações e recursividade

### ■ Localidade *Espacial*

- Uma posição de memória vizinha de uma posição referenciada recentemente tem boas chances de também ser referenciada
- Dados tendem a ser armazenados em posições contíguas

### ■ Como explorar o princípio de localidade?

#### ■ Localidade Temporal

- Mantenha palavras mais recentemente acessados na cache

#### ■ Localidade Espacial

- A leitura na memória, geralmente, é feita com mais de uma palavra ao mesmo tempo (blocos ou linhas de cache)
- Mova **blocos** de palavras contíguas para a cache

# Memória *Cache* (4)

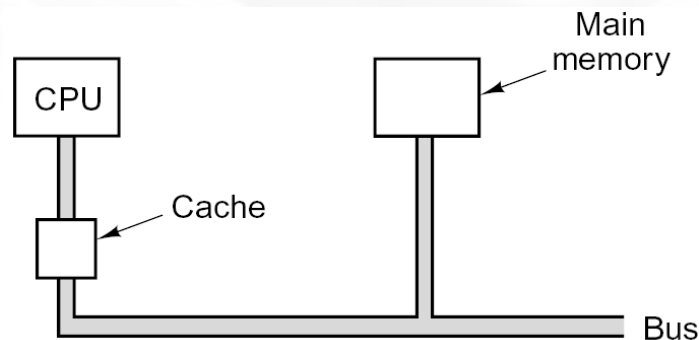
## ■ Princípio da Localidade (cont.)

### ■ Bloco ou Linhas de Cache

- Menor unidade de informação transferida entre os níveis da hierarquia de memória
- Um bloco é transferido num único acesso entre a memória principal e a cache, através de um largo barramento de dados

### Exemplo

Se uma cache tiver uma linha com 64 bytes, uma referência ao endereço 260 que não está na cache (falha) vai trazer da memória principal p/ a cache a linha composta pelos bytes 256 a 319:



**Posicionamento lógico da memória *cache***

Memória Principal

0	—	63
64	—	127
128	—	191
192	—	255
256	—	319

Bloco {

64 bytes

## Memória *Cache* <sup>(5)</sup>

- Se determinada palavra for lida ou escrita  $k$  vezes em um curto intervalo de tempo, o processador precisará
  - referenciar uma **única** vez a memória principal (“lenta”)
  - e  $k - 1$  vezes a memória rápida.

$c$ : tempo de acesso à *cache*

$m$ : tempo de acesso à memória principal

$h$ : **Taxa de acertos** – fração de todas as referências que puderam ser satisfeitas pela *cache*.

$$h = (k - 1)/k$$

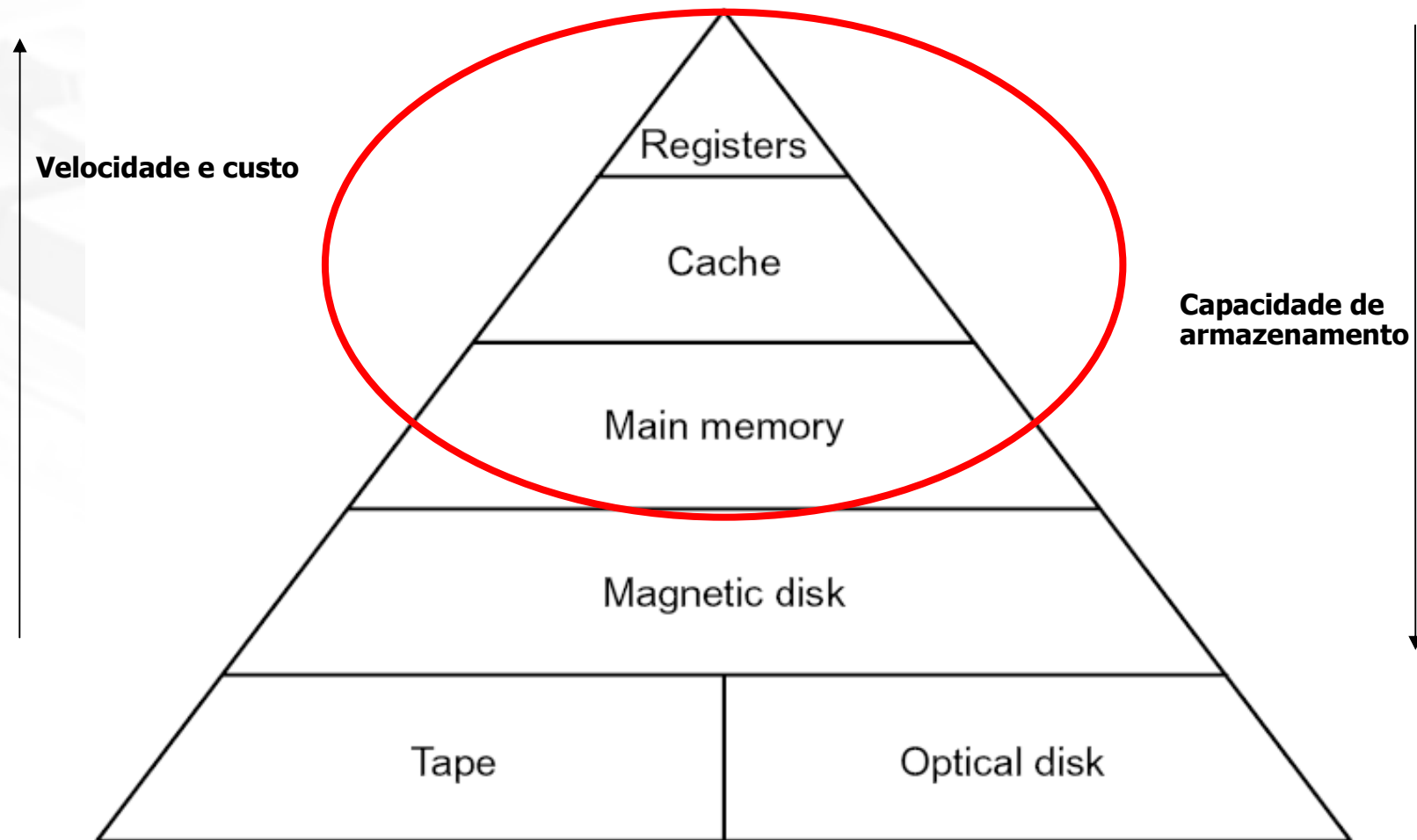
$$\text{Taxa de Falhas} = 1 - h$$

- Tempo médio de acesso =  $c + (1 - h)m$

## Memória *Cache* <sup>(6)</sup>

- **Acerto (*hit*)**
  - Posição acessada está na *cache*
- **Erro (*miss*)**
  - Posição acessada ausente da *cache*
  - Buscada da memória principal
- ***Cache Unificada*** (instruções e dados juntos) x ***Cache Dividida*** (uma *cache* para cada)
- Questões a discutir:
  - Qual o melhor tamanho da *cache*?
  - Qual o tamanho da **linha da *cache***?
    - Em geral, uma linha maior aproveita melhor a localidade espacial
    - MAS linha maior significa maior tempo para preencher a linha

# Hierarquia de Memória



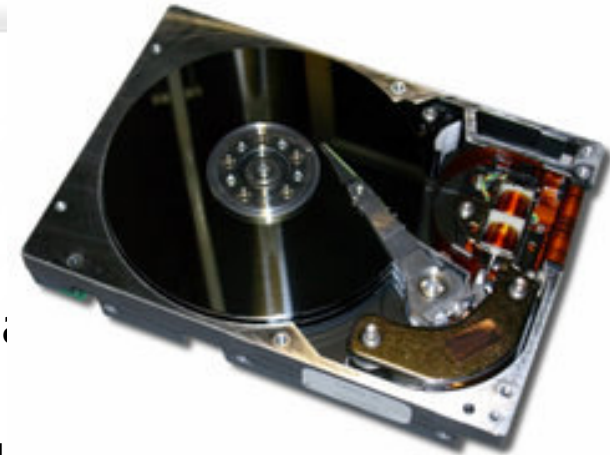
## Memória Secundária <sup>(1)</sup>

- Memórias que não podem ser endereçadas diretamente, isto é, a informação precisa ser carregada em memória primária antes de poder ser tratada pelo processador.
- Necessário pois o conteúdo da MP é apagado quando o computador é desligado. Desta forma, tem-se um meio de executar novamente programas e carregar arquivos contendo os dados da próxima vez em que o computador for ligado.
- São geralmente **não-voláteis**, permitindo guardar os dados permanentemente.
  - Discos Rígidos (HDs)
  - CDs
  - DVDs
  - Disquetes, etc.

## Memória Secundária (2)

- **Disco Rígido (HD - *Hard Disk*)**

- É a parte do computador onde são armazenadas informações que "não se apagam" (**Aquivos**)
- Caracterizado como memória física, não-volátil.
- O disco rígido é um sistema lacrado contendo **discos de metal recobertos por material magnético** onde os dados são gravados através de **cabeças**, e revestido externamente por uma proteção metálica que é presa ao gabinete do computador por parafusos.
- Também é chamado **Winchester**. (Rifles Winchester 30-30)
- **Nos Sistemas Operacionais mais recentes, o disco rígido é também utilizado para expandir a memória principal, através de mecanismos de Memória Virtual.**





## Memória Secundária <sup>(6)</sup>

### ■ Setor de *Boot* – Disco Rígido

- Nele é registrado qual **sistema operacional** está instalado no computador, com qual **sistema de arquivos** o disco foi formatado e quais arquivos devem ser lidos para inicializar o computador.
- Também é conhecido como **Trilha MBR** ou **Trilha 0**.
- **MBR (Master Boot Record)** - Registro de Inicialização Mestre
  - Contém a tabela de partição do disco que dará boot.
  - O MBR é lido pela **BIOS (*Basic Input Output System*)**, que interpreta a tabela de partição e em seguida carrega um programa chamado ***bootstrap***, que é o responsável pelo carregamento do Sistema Operacional.

# Memória Secundária (7)

## ■ Controlador de Disco

- **Controlador**(a) é um dispositivo de hardware que realiza a interface entre o exterior de um dispositivo e o seu funcionamento interno.
- Processador envia o endereço físico de dados para um HD e o controlador traduz esse endereço e aciona os dispositivos mecânicos específicos do disco para que os dados possam ser lidos e enviados.

## ■ Padrões

- **IDE** - *Integrated Drive Electronics*
- **(E)IDE** - *(Extended) Integrated Drive Electronics*
- **ATAPI** - *Advanced Technology Attachment Packet Interface*
- **SCSI** (***S**mall **C**omputer **S**ystem **I**nterface*).
  - Permite conexão de uma larga gama de periféricos (HDs, CD-ROMs, impressoras, scanners, etc.)

## Referências

- Andrew S. Tanenbaum, Organização Estruturada de Computadores, 5ª edição, Prentice-Hall do Brasil, 2007. Capítulo 2
- Lúcia Helena M. Pacheco, **Visão Geral de Organização Estruturada de Computadores e Linguagem de Montagem**. Universidade Federal de Santa Catarina. Centro Tecnológico, Departamento de Informática e de Estatística.
- Artigo sobre correção e detecção de erros
  - <http://www.dcc.ufla.br/infocomp/artigos/v1.1/cce.pdf>
- <http://www.wikipedia.org>

### Tabela ASCII

### Caracteres normais

Binário	Decimal	Hex	Gráfico	Binário	Decimal	Hex	Gráfico	Binário	Decimal	Hex	Gráfico
0010 0000	32	20	(vazio) (□)	0100 0000	64	40	@	0110 0000	96	60	`
0010 0001	33	21	!	0100 0001	65	41	A	0110 0001	97	61	a
0010 0010	34	22	"	0100 0010	66	42	B	0110 0010	98	62	b
0010 0011	35	23	#	0100 0011	67	43	C	0110 0011	99	63	c
0010 0100	36	24	\$	0100 0100	68	44	D	0110 0100	100	64	d
0010 0101	37	25	%	0100 0101	69	45	E	0110 0101	101	65	e
0010 0110	38	26	&	0100 0110	70	46	F	0110 0110	102	66	f
0010 0111	39	27	'	0100 0111	71	47	G	0110 0111	103	67	g
0010 1000	40	28	(	0100 1000	72	48	H	0110 1000	104	68	h
0010 1001	41	29	)	0100 1001	73	49	I	0110 1001	105	69	i
0010 1010	42	2A	*	0100 1010	74	4A	J	0110 1010	106	6A	j
0010 1011	43	2B	+	0100 1011	75	4B	K	0110 1011	107	6B	k
0010 1100	44	2C	,	0100 1100	76	4C	L	0110 1100	108	6C	l
0010 1101	45	2D	-	0100 1101	77	4D	M	0110 1101	109	6D	m
0010 1110	46	2E	.	0100 1110	78	4E	N	0110 1110	110	6E	n
0010 1111	47	2F	/	0100 1111	79	4F	O	0110 1111	111	6F	o
0011 0000	48	30	0	0101 0000	80	50	P	0111 0000	112	70	p
0011 0001	49	31	1	0101 0001	81	51	Q	0111 0001	113	71	q
0011 0010	50	32	2	0101 0010	82	52	R	0111 0010	114	72	r