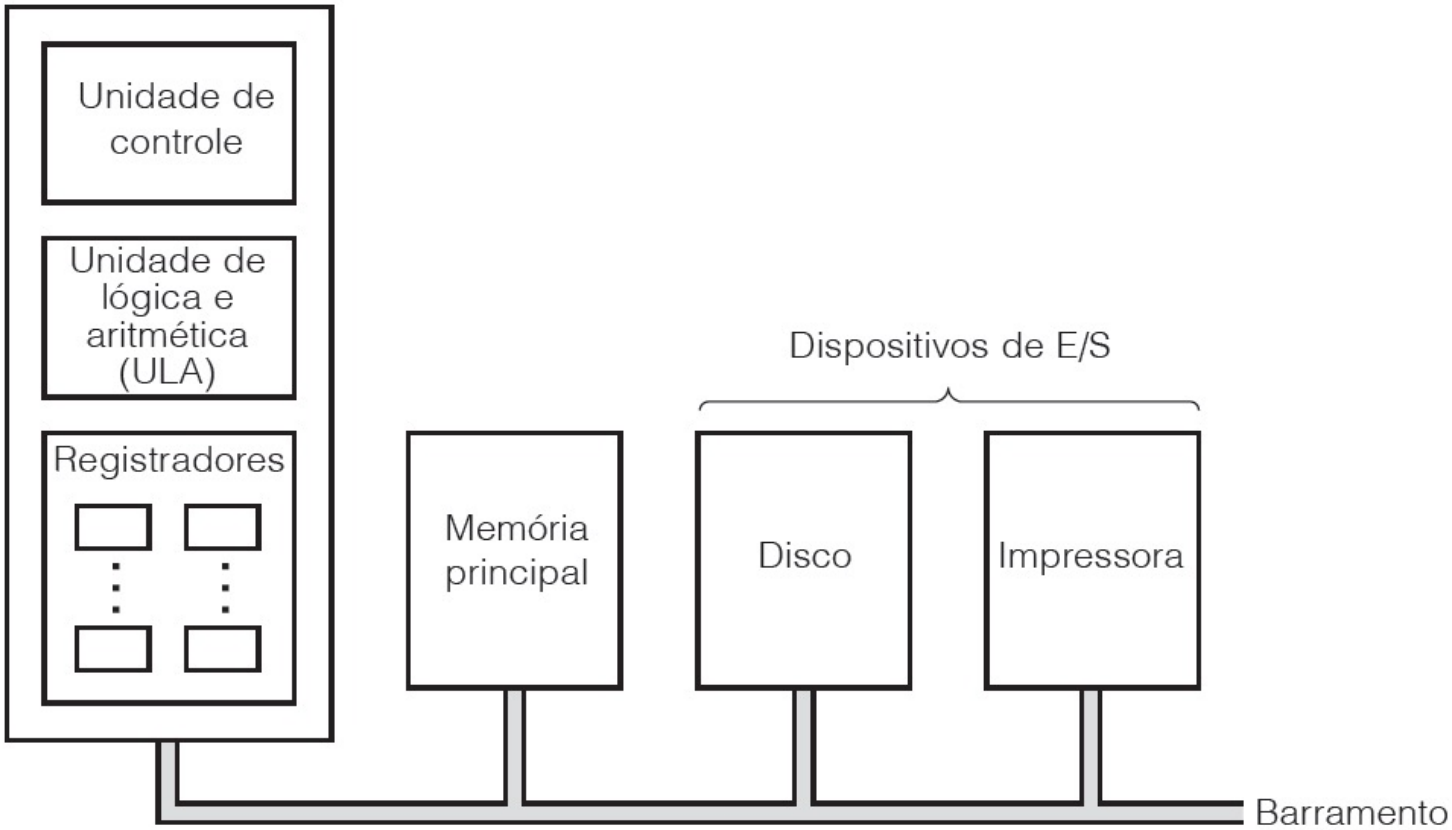
Organização de sistemas de computadores

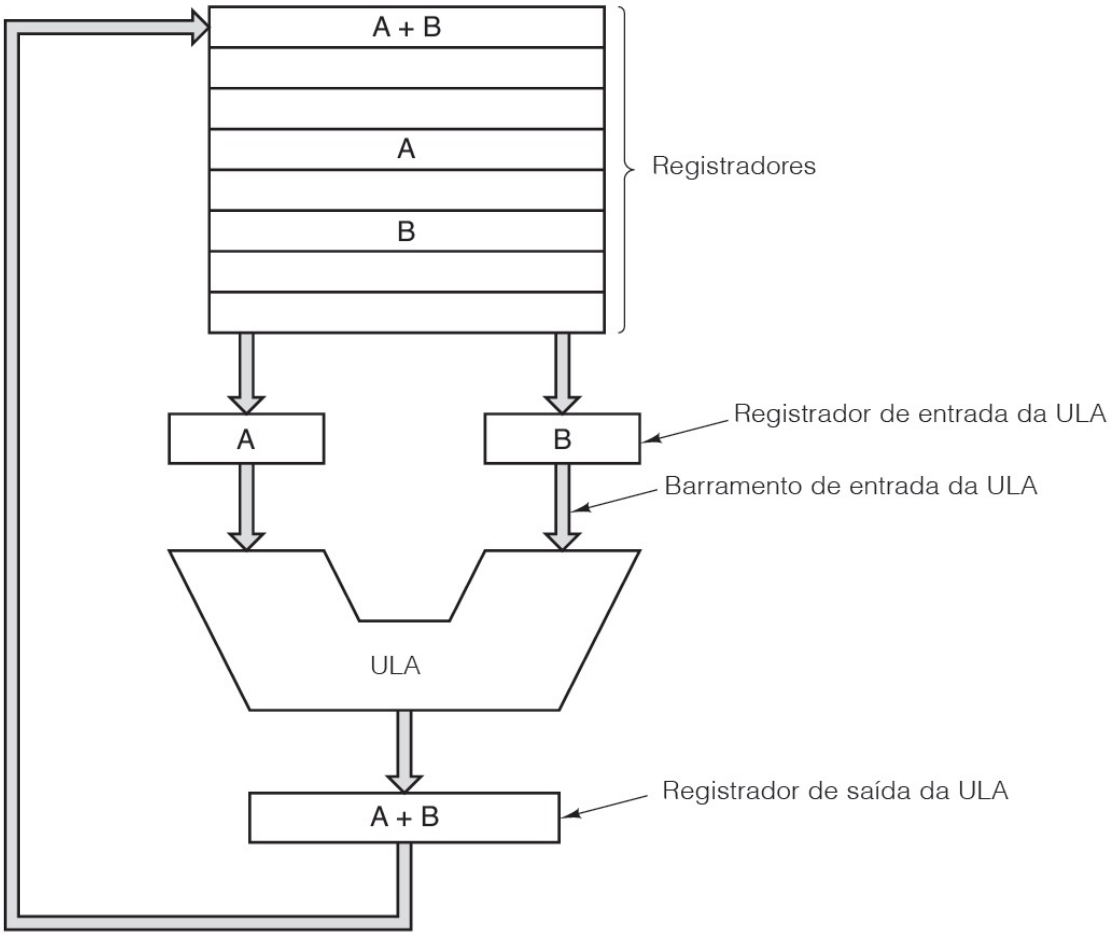
Processadores

- A CPU (Central Processing Unit – unidade central de processamento) é o “cérebro” do computador.



Organização da CPU

- O caminho de dados de uma típica máquina de von Neumann.



Execução de instrução

- A CPU executa cada instrução em uma série de pequenas etapas. Em termos simples, as etapas são as seguintes:

1. Trazer a próxima instrução da memória até o registrador de instrução.

2. Alterar o contador de programa para que aponte para a próxima instrução.

3. Determinar o tipo de instrução trazida.

4. Se a instrução usar uma palavra na memória, determinar onde essa palavra está.

Execução de instrução

5. Trazer a palavra para dentro de um registrador da CPU, se necessário.

6. Executar a instrução.

7. Voltar à etapa 1 para iniciar a execução da instrução seguinte.

- A figura a seguir mostra esse programa informal reescrito como um método Java (isto é, um procedimento) denominado *interpret*.

Execução de instrução

```
public class Interp {

    static int PC;                                  // contador de programa contém endereço da próxima instr
    static int AC;                                  // o acumulador, um registrador para efetuar aritmética
    static int instr;                               // um registrador para conter a instrução corrente
    static int instr_type;                          // o tipo da instrução (opcode)
    static int data_loc;                            // o endereço dos dados, ou –1 se nenhum
    static int data;                                // mantém o operando corrente
    static boolean run_bit = true;                  // um bit que pode ser desligado para parar a máquina

    public static void interpret(int memory[ ], int starting_ address) {
        // Esse procedimento interpreta programas para uma máquina simples com instruções que têm
        // um operando na memória. A máquina tem um registrador AC (acumulador), usado para
        // aritmética. A instrução ADD soma um inteiro na memória do AC, por exemplo.
        // O interpretador continua funcionando até o bit de funcionamento ser desligado pela instrução HALT.
        // O estado de um processo que roda nessa máquina consiste em memória, o
        // contador de programa, bit de funcionamento e AC. Os parâmetros de entrada consistem
        / na imagem da memória e no endereço inicial.

    PC = starting_address;
    while (run_bit) {
        instr = memory[PC];                         // busca a próxima instrução e armazena em instr
        PC = PC + 1;                                // incrementa contador de programa
        instr_type = get_instr_type(instr);         // determina tipo da instrução
        data_loc = find_data(instr, instr_type);    // localiza dados (–1 se nenhum)
        if (data_loc >= 0)                          // se data_loc é –1, não há nenhum operando
           data = memory[data_loc];                 // busca os dados
        execute(instr_type, data);                  // executa instrução
        }
    }
    private static int get_instr_type(int addr) { ... }
    private static int find_data(int instr, int type) { ... }
    private static void execute(int type, int data) { ... }
}
```

Execução de instrução

- Computadores simples com instruções interpretadas tinham benefícios, entre os quais os mais importantes eram:

1. A capacidade de corrigir em campo instruções executadas incorretamente ou até compensar deficiências de projeto no hardware básico.

2. A oportunidade de acrescentar novas instruções a um custo mínimo, mesmo após a entrega da máquina.

3. Projeto estruturado que permitia desenvolvimento, teste e documentação eficientes de instruções complexas.

RISC *versus* CISC

- Em 1980, um grupo em Berkeley, liderado por David Patterson e Carlo Séquin, começou a projetar chips para CPUs VLSI que não usavam interpretação.

- Eles cunharam o termo RISC para esse conceito e deram ao seu chip de CPU o nome RISC I CPU, seguido logo depois pelo RISC II.

- A característica que chamou a atenção de todos era o número relativamente pequeno de instruções disponíveis, em geral cerca de 50.

- Número muito menor do que os *mainframes* da IBM.

RISC *versus* CISC

- Mesmo que uma máquina RISC precisasse de quatro ou cinco instruções para fazer o que uma CISC fazia com uma só, se as instruções RISC fossem dez vezes mais rápidas, o RISC vencia.

- A Intel conseguiu empregar as mesmas ideias mesmo em uma arquitetura CISC.

- Mesmo que essa abordagem híbrida não seja tão rápida quanto um projeto RISC puro, ela resulta em desempenho global competitivo e ainda permite que softwares antigos sejam executados sem modificação.

Princípios de projeto para computadores modernos

- Há um conjunto de princípios de projeto, às vezes denominados princípios de projeto RISC, que os arquitetos de CPUs de uso geral se esforçam por seguir:

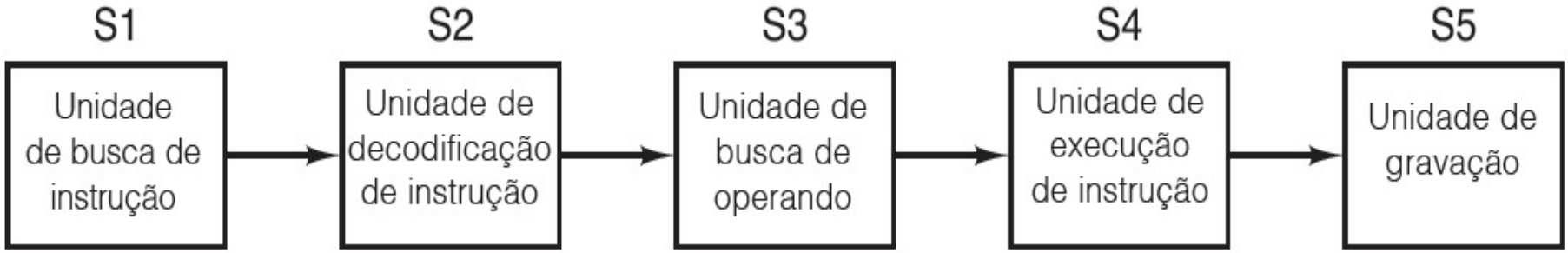
  - Todas as instruções são executadas diretamente por hardware.
  - É preciso maximizar a taxa de execução das instruções.
  - Instruções devem ser fáceis de decodificar.
  - Somente LOAD e STORE devem referenciar a memória.
  - É preciso providenciar muitos registradores.

Paralelismo no nível de instrução

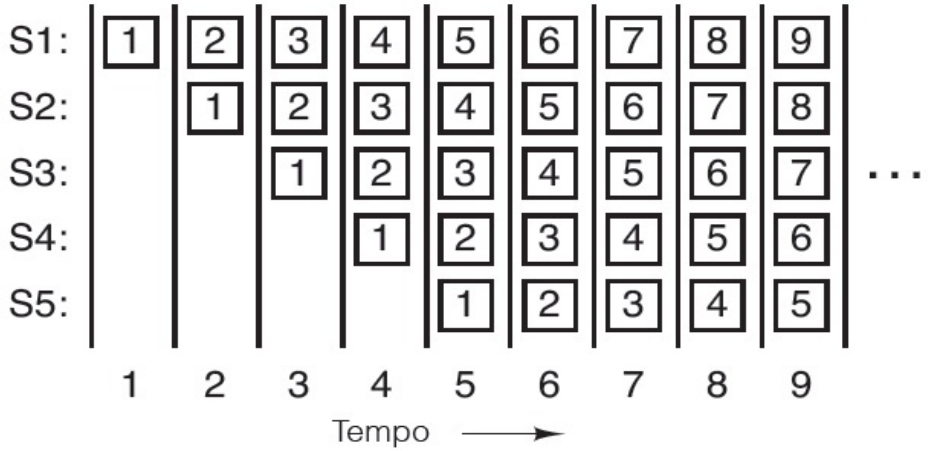
- O paralelismo tem duas formas gerais:
- **No nível de instrução**
  - O paralelismo é explorado dentro de instruções individuais para obter da máquina mais instruções por segundo.
- **No nível de processador**
  - Várias CPUs trabalham juntas no mesmo problema. Cada abordagem tem seus próprios méritos.

Pipelining (paralelismo)

- *Pipeline* de cinco estágios.



- Estado de cada estágio como uma função do tempo.



Pipelining (paralelismo)

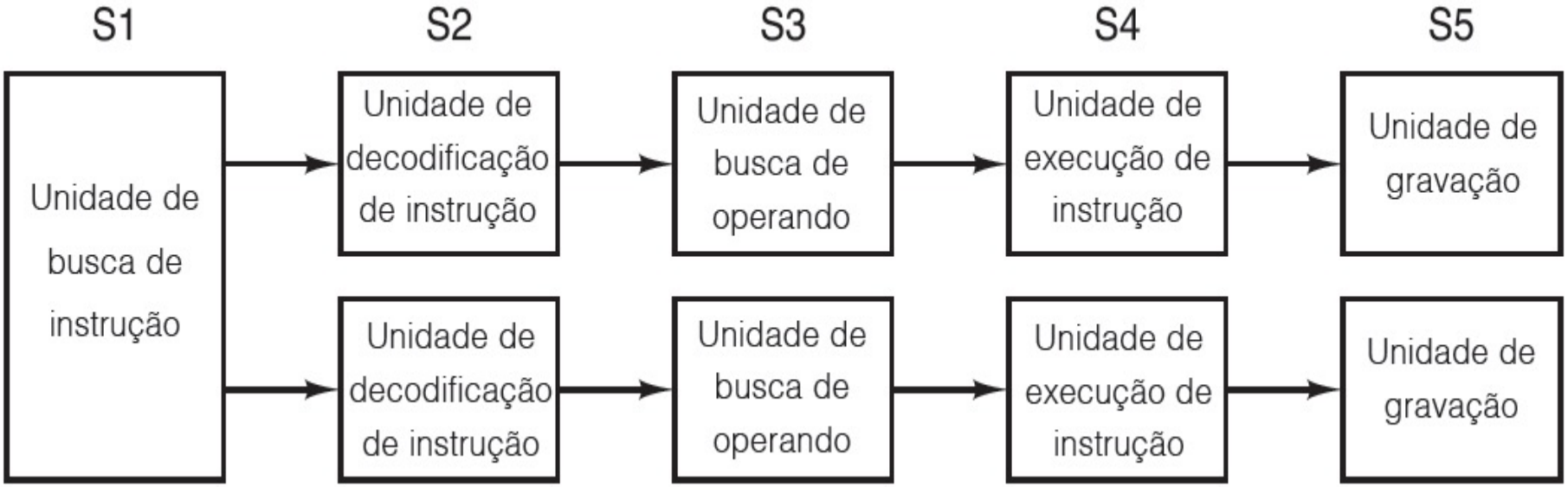
- O *pipelining* permite um compromisso entre **latência** (o tempo que demora para executar uma instrução) e **largura de banda de processador** (quantos MIPS a CPU tem).

- Com um tempo de ciclo de T ns e n estágios no *pipeline*, a latência é nT ns porque cada instrução passa por n estágios, cada um dos quais demora T ns.

- Visto que uma instrução é concluída a cada ciclo de *clock* e que há $10^9/T$ ciclos de *clock* por segundo, o número de instruções executadas por segundo é $10^9/T$.

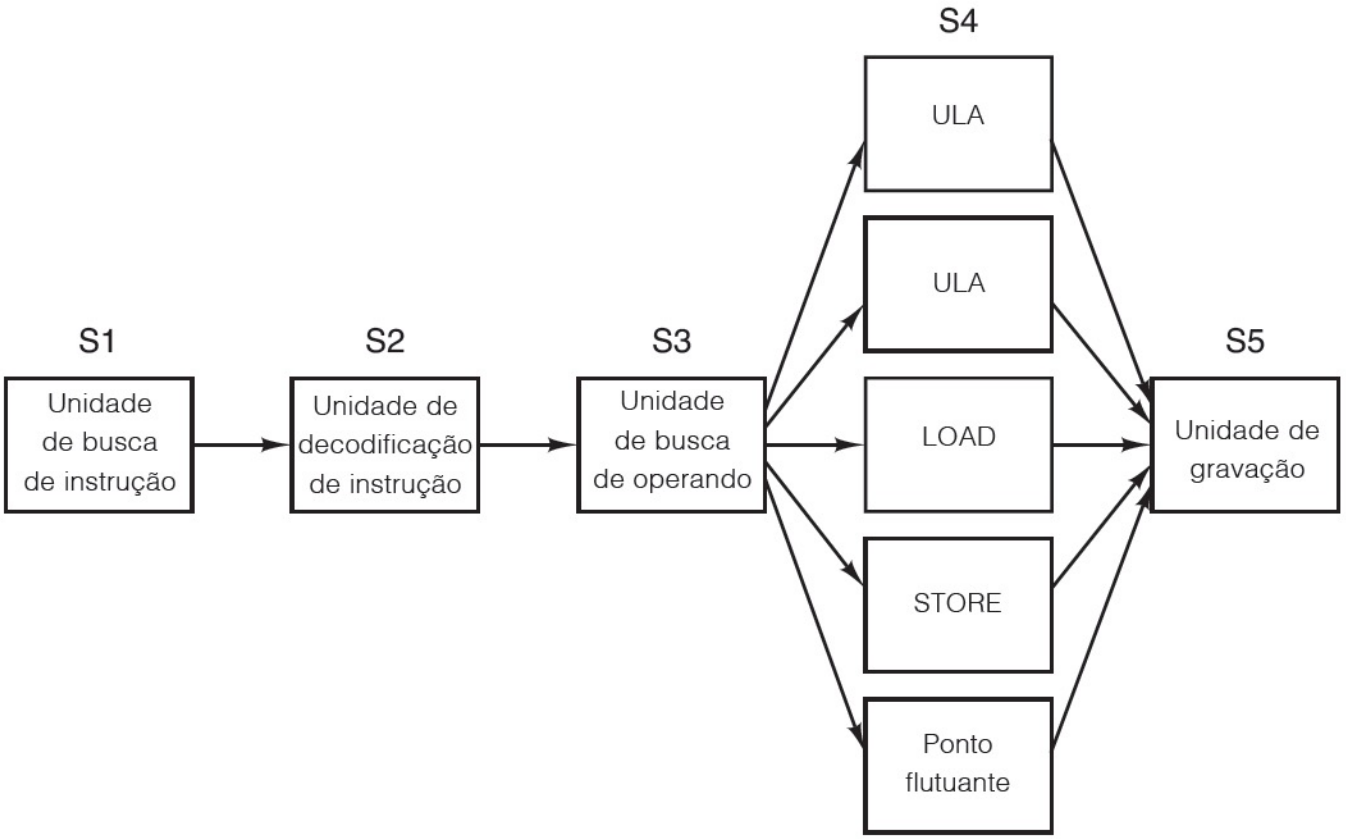
Pipelining (paralelismo)

- *Pipelines* duplos de cinco estágios com uma unidade de busca de instrução em comum.



Pipelining (paralelismo)

- A ideia básica é ter apenas um único *pipeline*, mas lhe dar várias unidades funcionais:



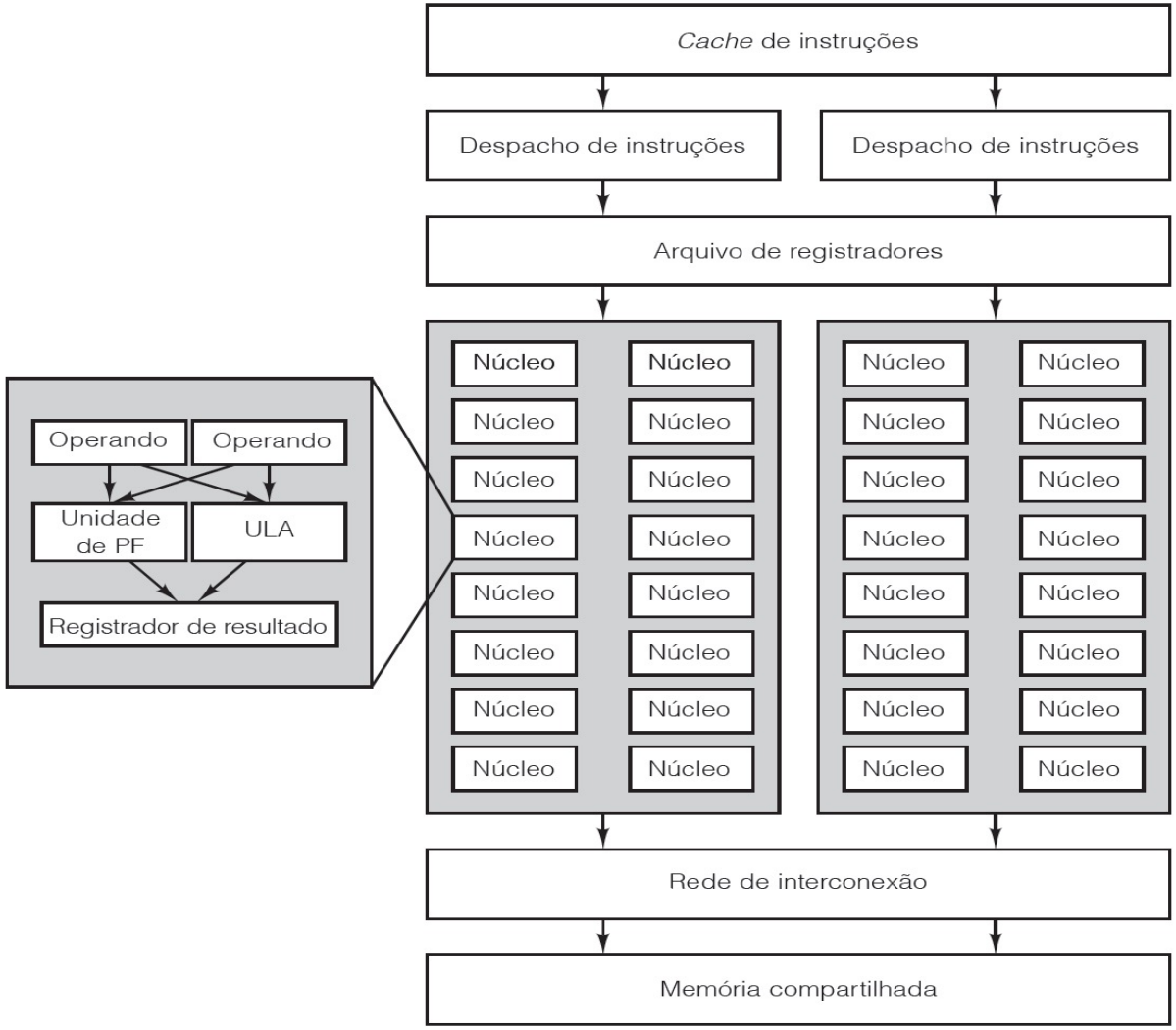
Paralelismo no nível do processador

- Um **processador SIMD** consiste em um grande número de processadores idênticos que efetuam a mesma sequência de instruções sobre diferentes conjuntos de dados.

- As modernas unidades de processamento de gráficos (GPUs) contam bastante com o processamento SIMD para fornecer poder computacional maciço com poucos transistores.

- A figura a seguir mostra o processador SIMD no núcleo da GPU Fermi da Nvidia.

Paralelismo no nível do processador

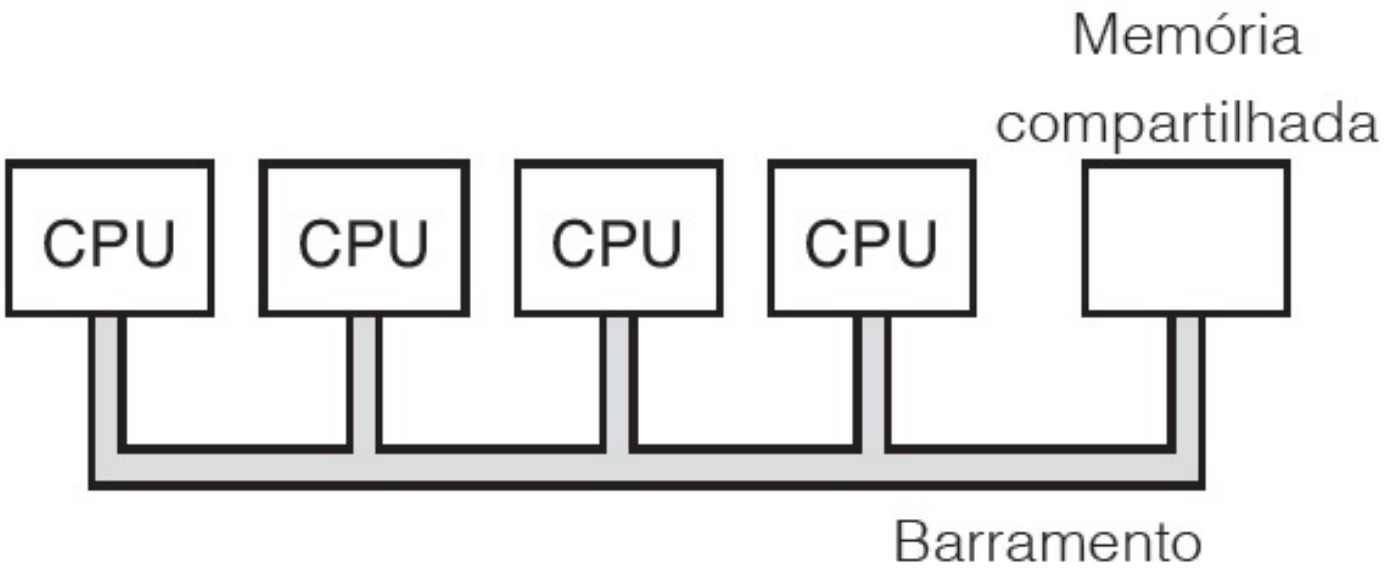


Paralelismo no nível do processador

- Para um programador, um **processador vetorial** se parece muito com um processador SIMD.
- Ele é muito eficiente para executar uma sequência de operações em pares de elementos de dados.
- Porém, todas as operações de adição são efetuadas em uma única unidade funcional, de alto grau de paralelismo.
- Nosso primeiro sistema paralelo com CPUs totalmente desenvolvidas é o **multiprocessador**, um sistema com mais de uma CPU que compartilha uma memória em comum.

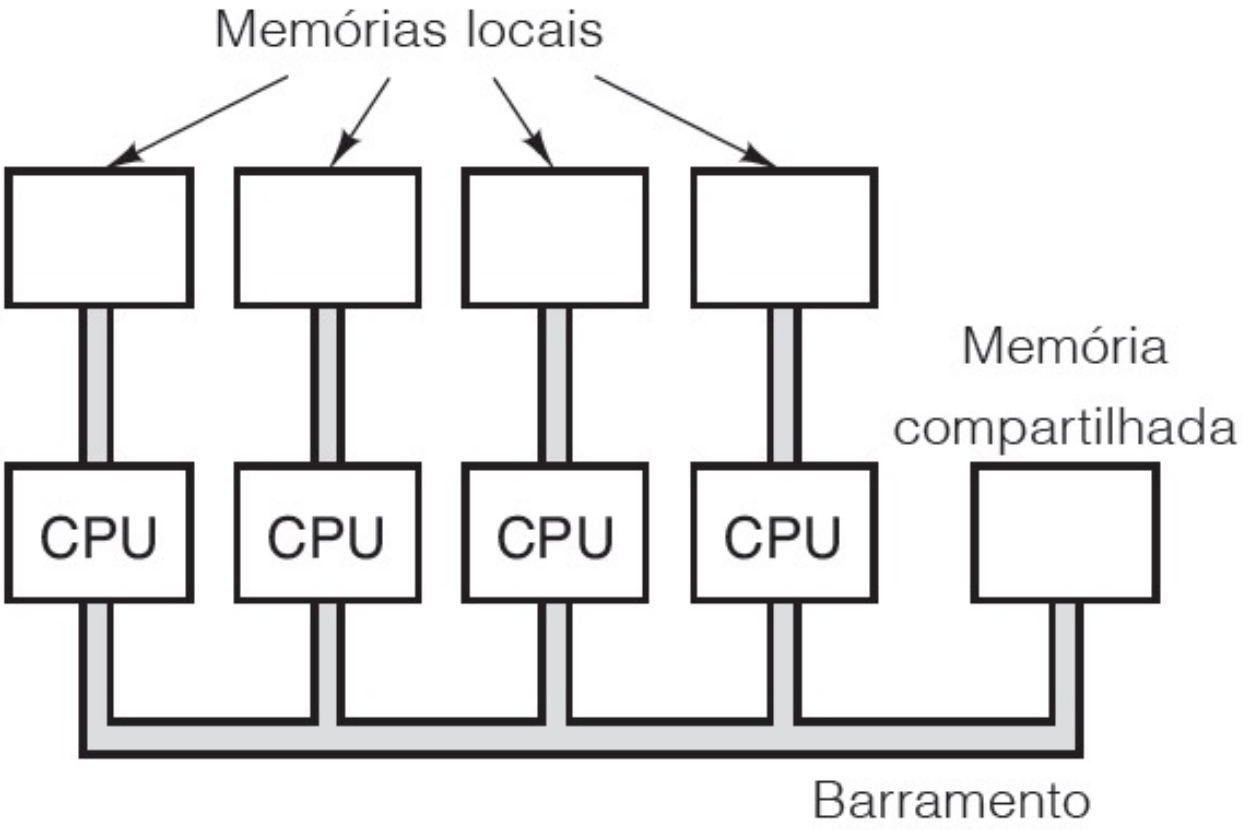
Paralelismo no nível do processador

- Multiprocessador de barramento único.



Paralelismo no nível do processador

- Multicomputador com memórias locais.



Paralelismo no nível do processador

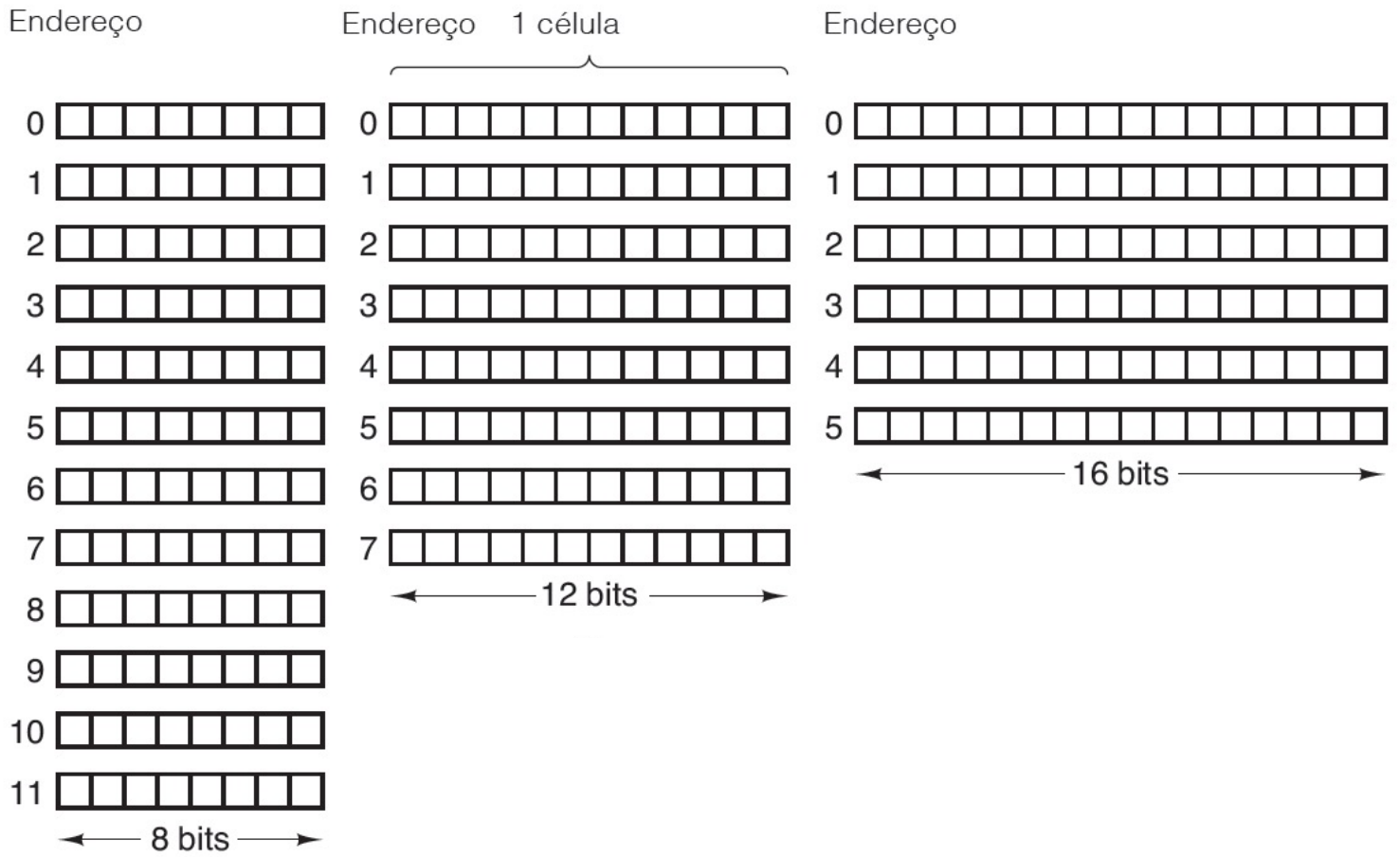
- Costuma-se dizer que as CPUs de um **multicomputador** são fracamente acopladas, para contrastá-las com as CPUs fortemente acopladas de um multiprocessador.
- As CPUs de um multicomputador se comunicam enviando mensagens umas às outras, mais ou menos como enviar e-mails, porém, com muito mais rapidez.
- Multiprocessadores são mais fáceis de programar.
- Multicomputadores são mais fáceis de construir.

Memória primária

- A **memória** é a parte do computador onde são armazenados programas e dados.
- A unidade básica de memória é dígito binário, denominado **bit**. Um bit pode conter um 0 ou um 1.
- Memórias consistem em uma quantidade de **células** (ou **locais**), cada uma das quais podendo armazenar uma informação.
- Cada célula tem um número, denominado seu **endereço**, pelo qual os programas podem se referir a ela.

Memória primária

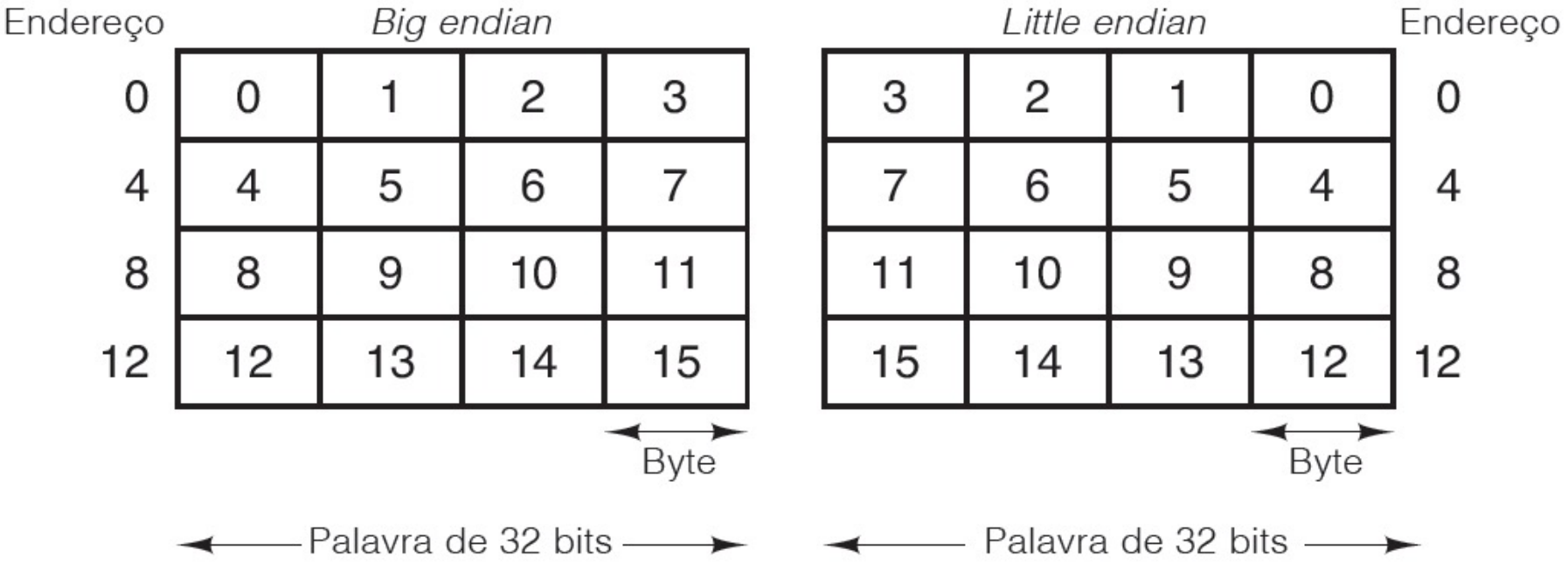
- Três maneiras de organizar uma memória de 96 bits.



Memória primária

- Os **bytes** em uma palavra podem ser numerados da esquerda para a direita ou da direita para a esquerda.

- Memória *big endian* e memória *little endian*.



Códigos de correção de erro

- Memórias de computador podem cometer erros de vez em quando devido a picos de tensão na linha elétrica, raios cósmicos ou outras causas.
- As propriedades de detecção de erro e correção de erro de um código dependem de sua distância de Hamming.
- Para detectar d erros de único bit, você precisa de um código de distância $d + 1$.
- Para corrigir erros de único bit, você precisa de um código de distância $2d + 1$.

Códigos de correção de erro

- Número de bits de verificação para um código que pode corrigir um erro único.

Tamanho da palavra	Bits de verificação	Tamanho total	Acréscimo percentual
8	4	12	50
16	5	21	31
32	6	38	19
64	7	71	11
128	8	136	6
256	9	265	4
512	10	522	2

Códigos de correção de erro

- A figura a seguir mostra a construção de um código de Hamming para a palavra de memória de 16 bits 1111000010101110.
- A palavra de código de 21 bits é 001011110000101101110.
- Para ver como funciona a correção de erros, considere o que aconteceria se o bit 5 fosse invertido por uma sobrecarga elétrica na linha de força.
- A nova palavra de código seria 001001110000101101110 em vez de 001011110000101101110.

Códigos de correção de erro

- Os 5 bits de paridade serão verificados com os seguintes resultados:

  - Bit de paridade 1 incorreto (1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21 contêm cinco 1s).

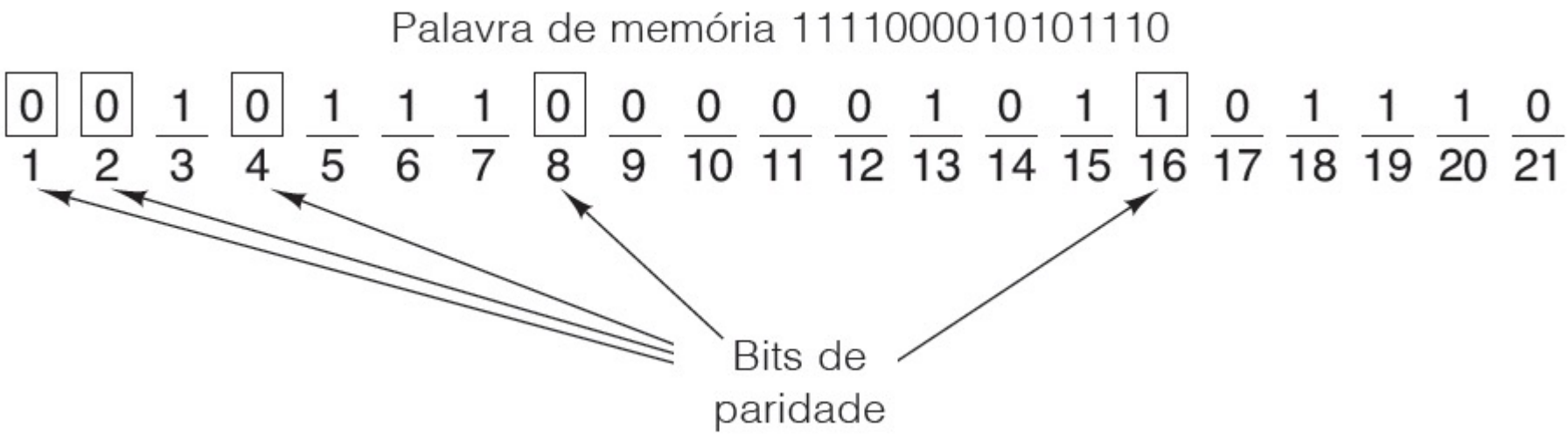
  - Bit de paridade 2 correto (2, 3, 6, 7, 10, 11, 14, 15, 18, 19 contêm seis 1s).

  - Bit de paridade 4 incorreto (4, 5, 6, 7, 12, 13, 14, 15, 20, 21 contêm cinco 1s).

  - Bit de paridade 8 correto (8, 9, 10, 11, 12, 13, 14, 15 contêm dois 1s).

  - Bit de paridade 16 correto (16, 17, 18, 19, 20, 21 contêm quatro 1s).

Códigos de correção de erro



Memória *cache*

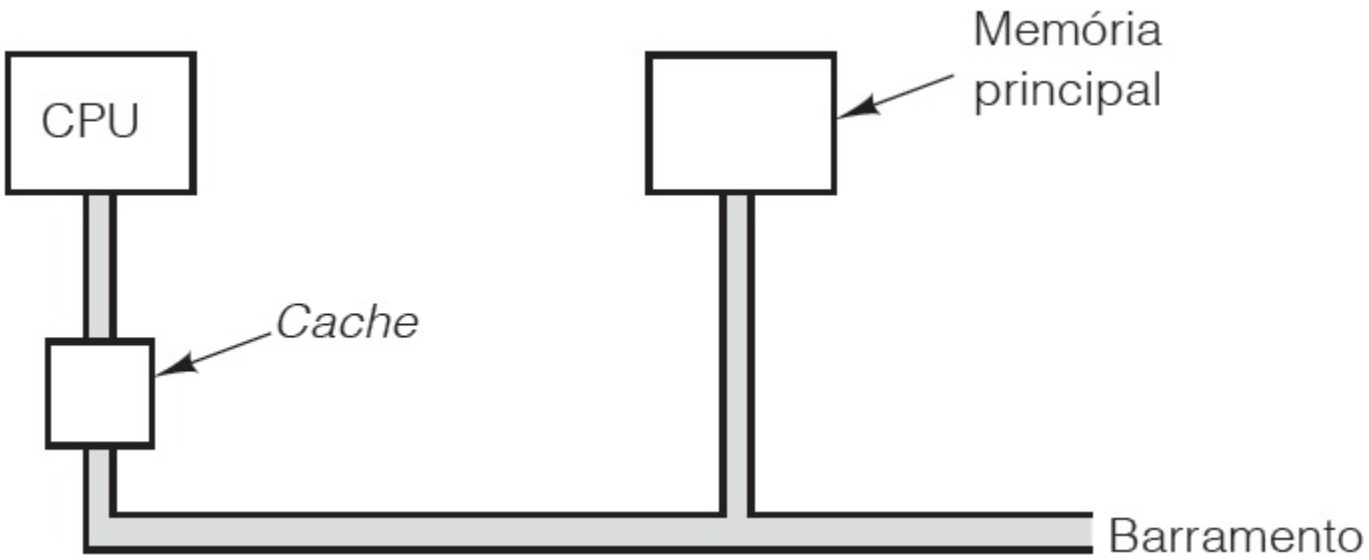
- A ideia básica de uma *cache* é simples: as palavras de memória usadas com mais frequência são mantidas na *cache*.

- Quando a CPU precisa de uma palavra, ela examina em primeiro lugar a *cache*. Somente se a palavra não estiver ali é que ela recorre à memória principal.

- Se uma fração substancial das palavras estiver na *cache*, o tempo médio de acesso pode ser muito reduzido.

Memória *cache*

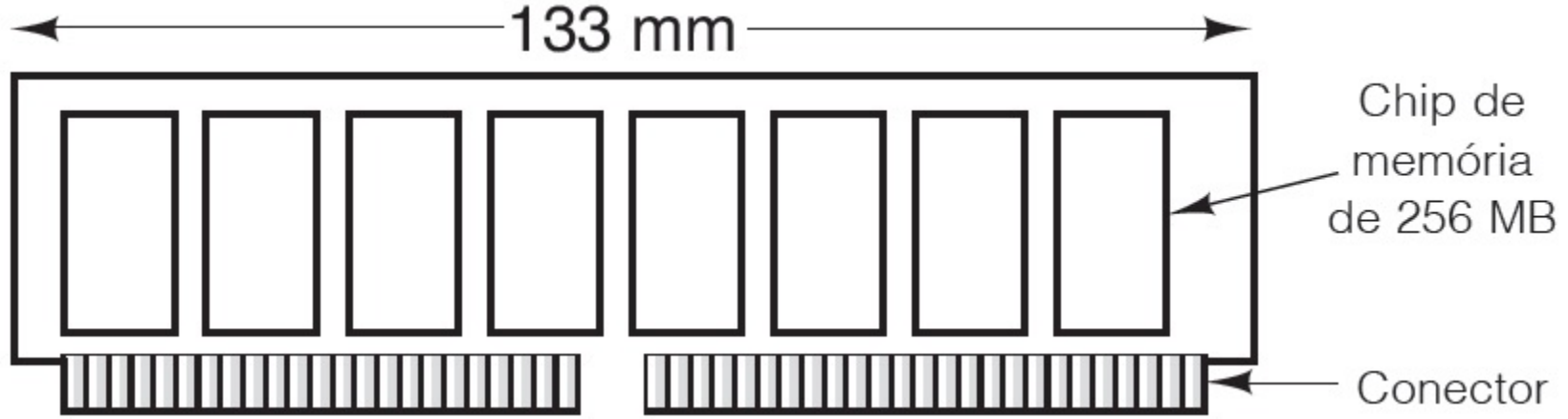
- A localização lógica da *cache* é entre a CPU e a memória principal.
- Em termos físicos, há diversos lugares em que ela poderia estar localizada.



Empacotamento e tipos de memória

- Uma configuração típica de DIMM poderia ter oito chips de dados com 256 MB cada. Então, o módulo inteiro conteria 2 GB.

- Muitos computadores têm espaço para quatro módulos, o que dá uma capacidade total de 8 GB se usarem módulos de 2 GB e mais, se usarem módulos maiores.



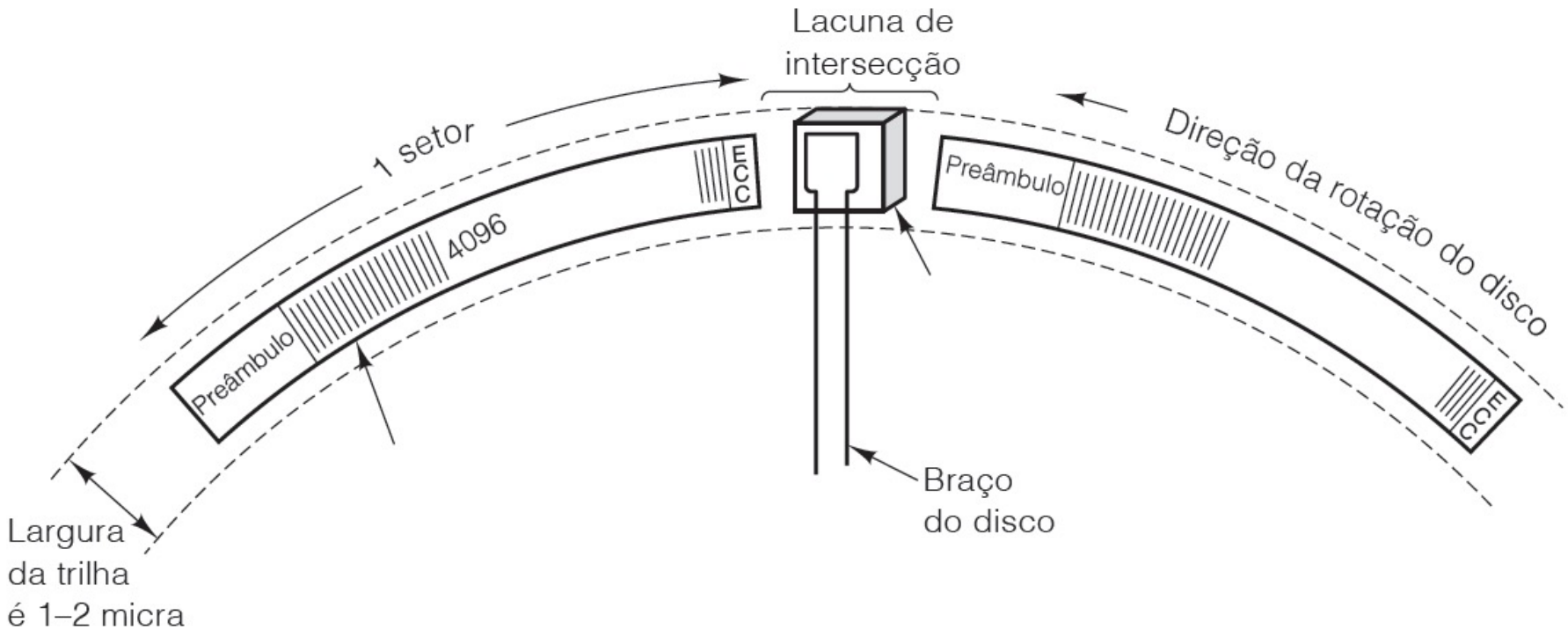
Memória secundária

- Seja qual for o tamanho da memória principal, ela sempre será muito pequena.

- A solução tradicional para armazenar
- grandes quantidades de dados é
- uma hierarquia de memória:

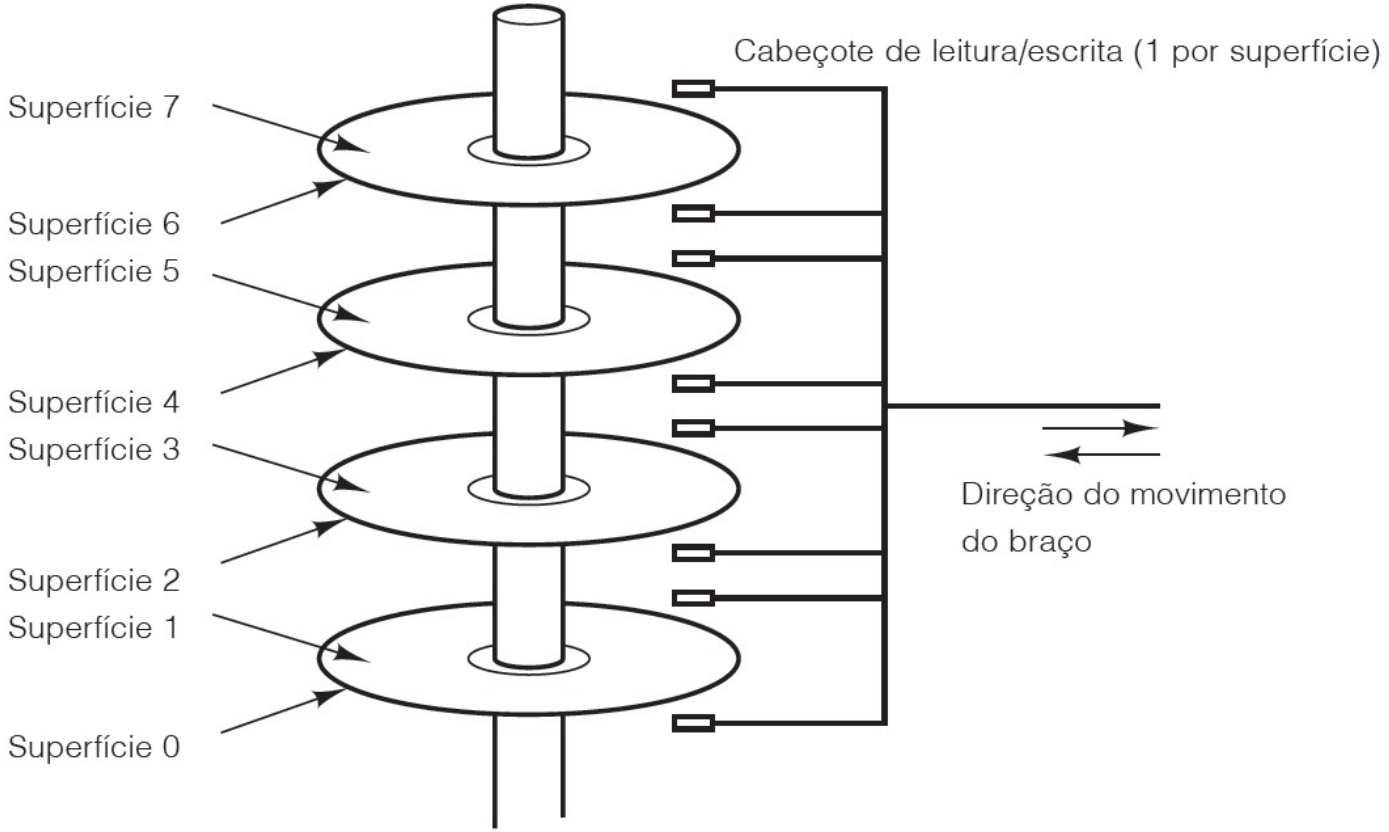
Memória secundária

- Um disco magnético é composto de um ou mais pratos de alumínio com um revestimento magnetizável.



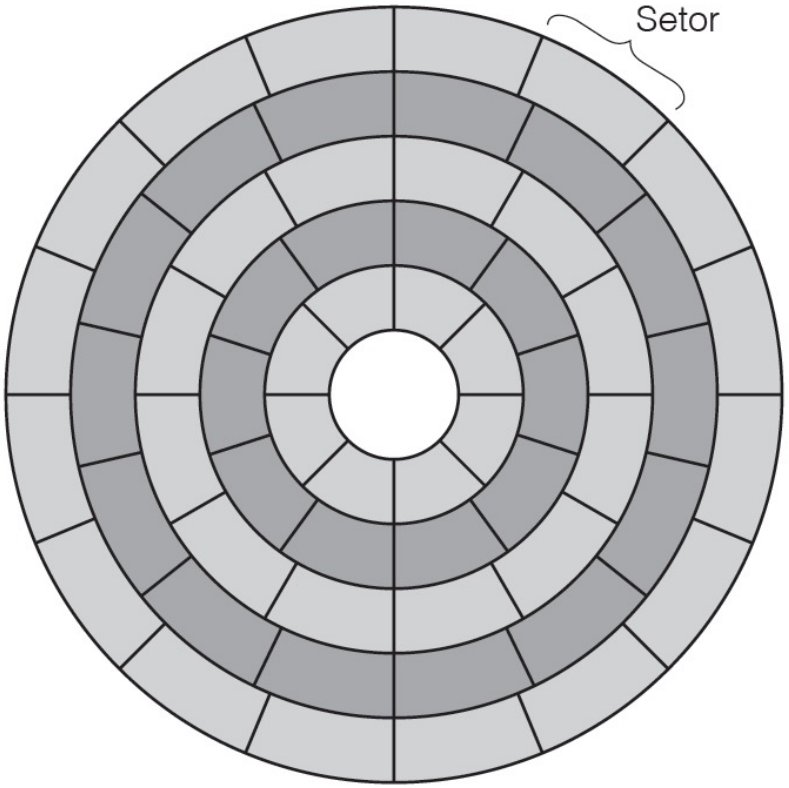
Memória secundária

- A maioria dos discos é composta de vários pratos empilhados na vertical:



Memória secundária

- Hoje, os cilindros são divididos em zonas e o número de setores por trilha aumenta de zona em zona partindo da trilha mais interna para a mais externa.



Memória secundária

- A tecnologia evoluiu rapidamente e passou do controlador em uma placa separada para o controlador integrado com os *drives*, começando com *drives* **IDE** em meados da década de 1980.

- Com o tempo, os *drives* IDE evoluíram para *drives* **EIDE**.

- Enquanto a tecnologia de disco continuava a melhorar, o padrão EIDE continuava a evoluir, mas, por alguma razão, o sucessor do EIDE foi denominado **ATA-3**.

- Na edição seguinte, o padrão recebeu o nome de **ATA PI-4**.

Memória secundária

- Em vez de aumentar o tamanho do conector do *drive*, esse padrão usa o que é chamado **ATA serial** para transferir 1 bit por vez por um conector de 7 pinos a velocidades que começam em 150 MB/s e que, com o tempo, espera-se que alcancem 1,5 GB/s.

- **Discos SCSI** têm uma interface diferente e taxas de transferência muito mais elevadas.

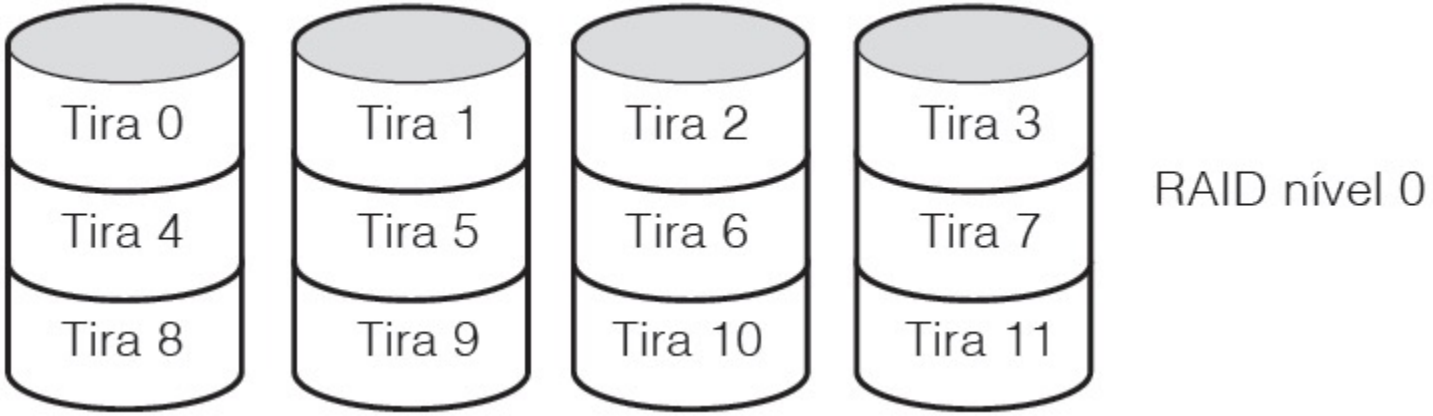
- O SCSI é um barramento ao qual podem ser conectados um controlador SCSI e até sete dispositivos.

- O cabo mais comum para SCSI de 8 bits tem 50 fios.

Memória secundária

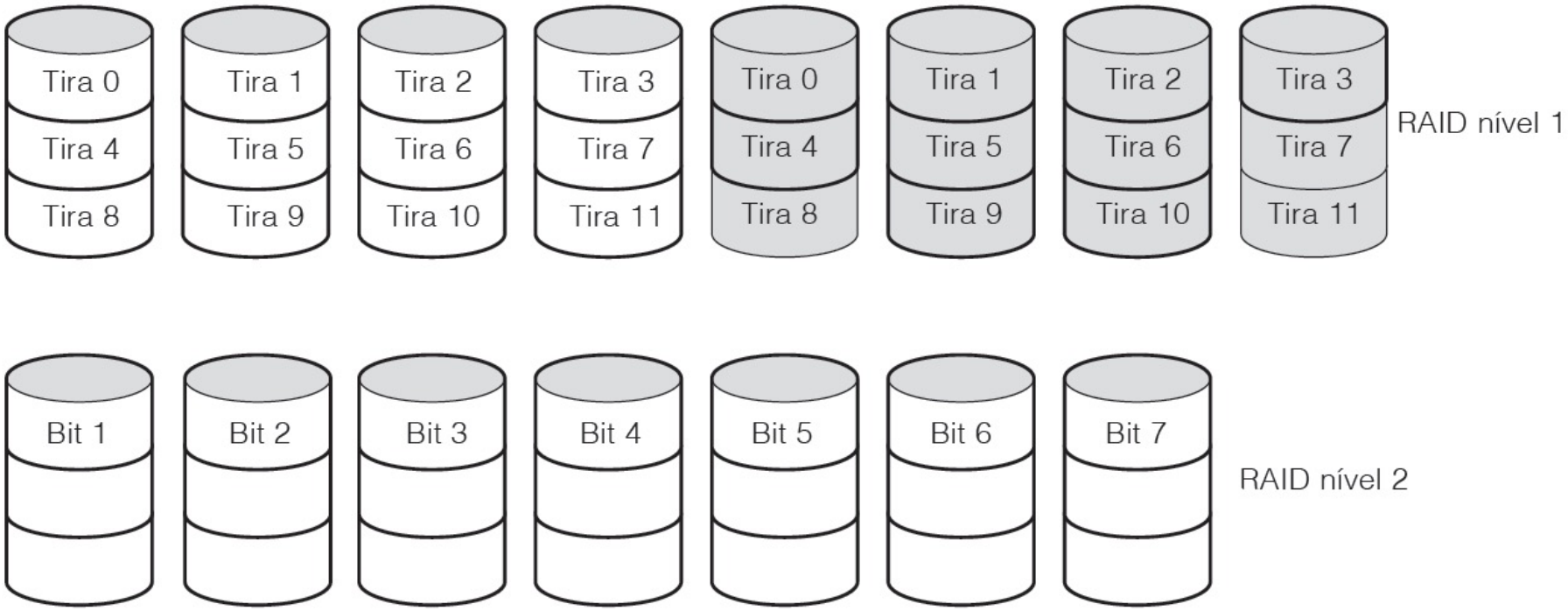
- Um **RAID** deveria parecer um SLED para o sistema operacional, mas ter melhor desempenho e melhor confiabilidade.

- RAIDs níveis 0 a 5. Os drives de backup e paridade estão sombreados.

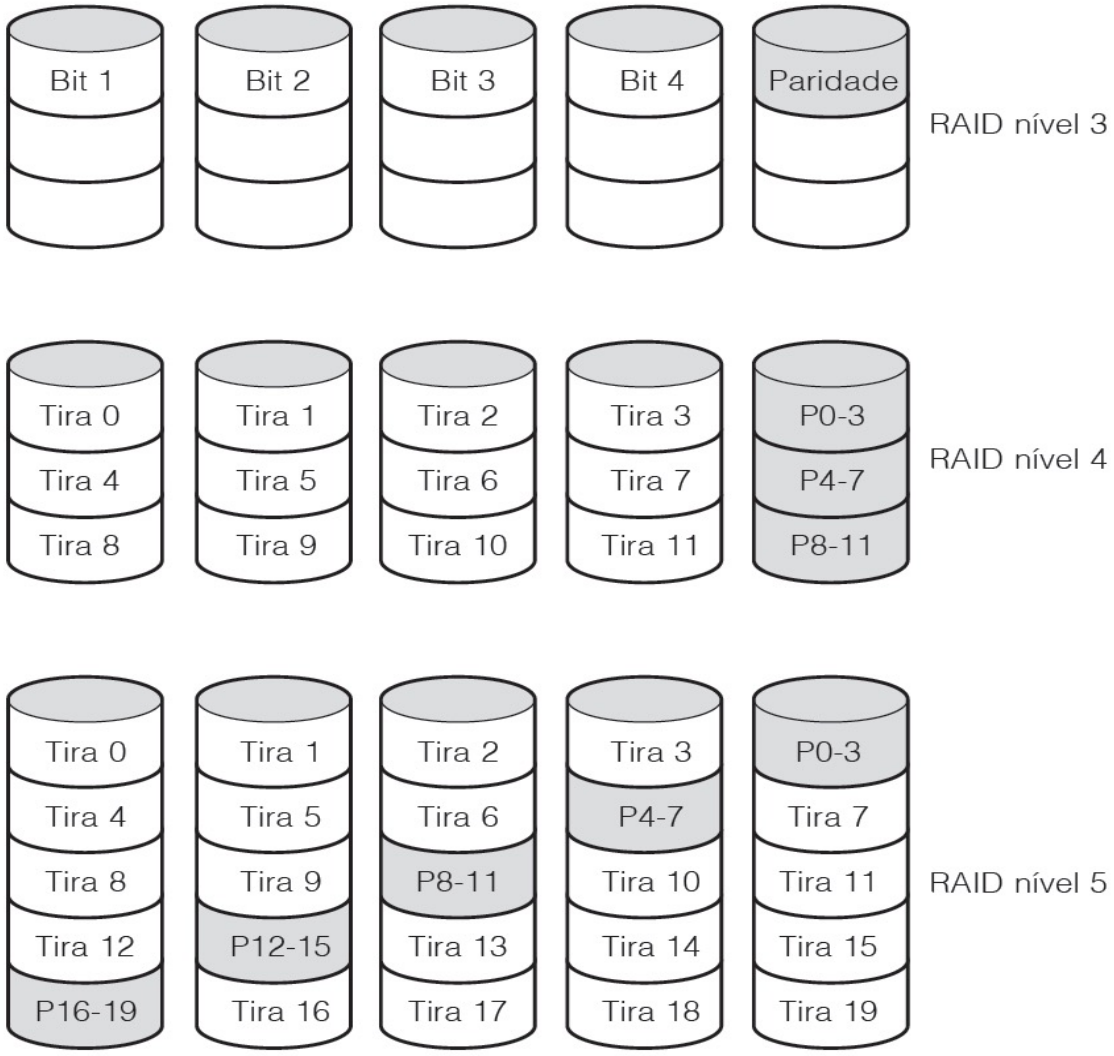


RAID nível 0

Memória secundária



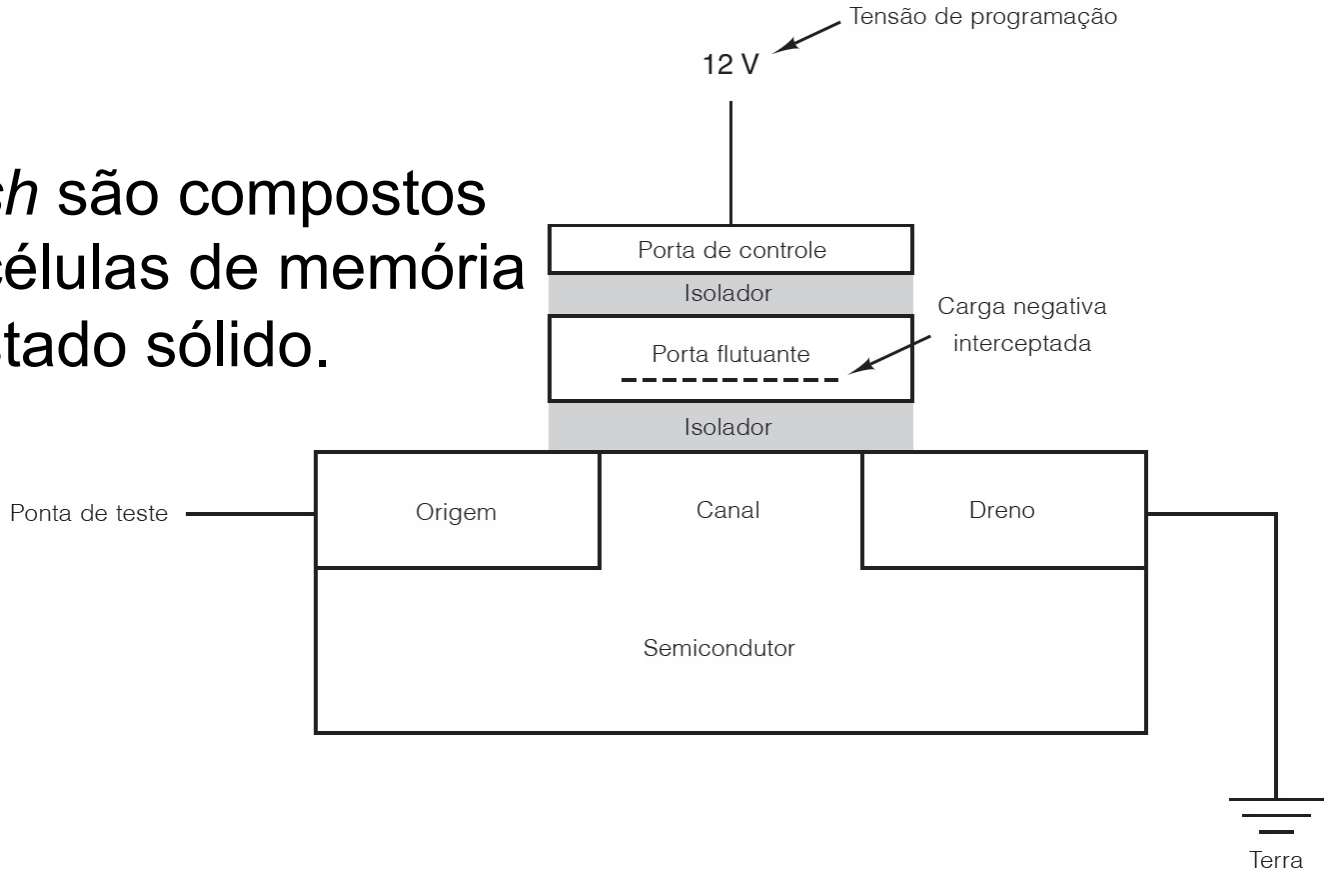
Memória secundária



Memória secundária

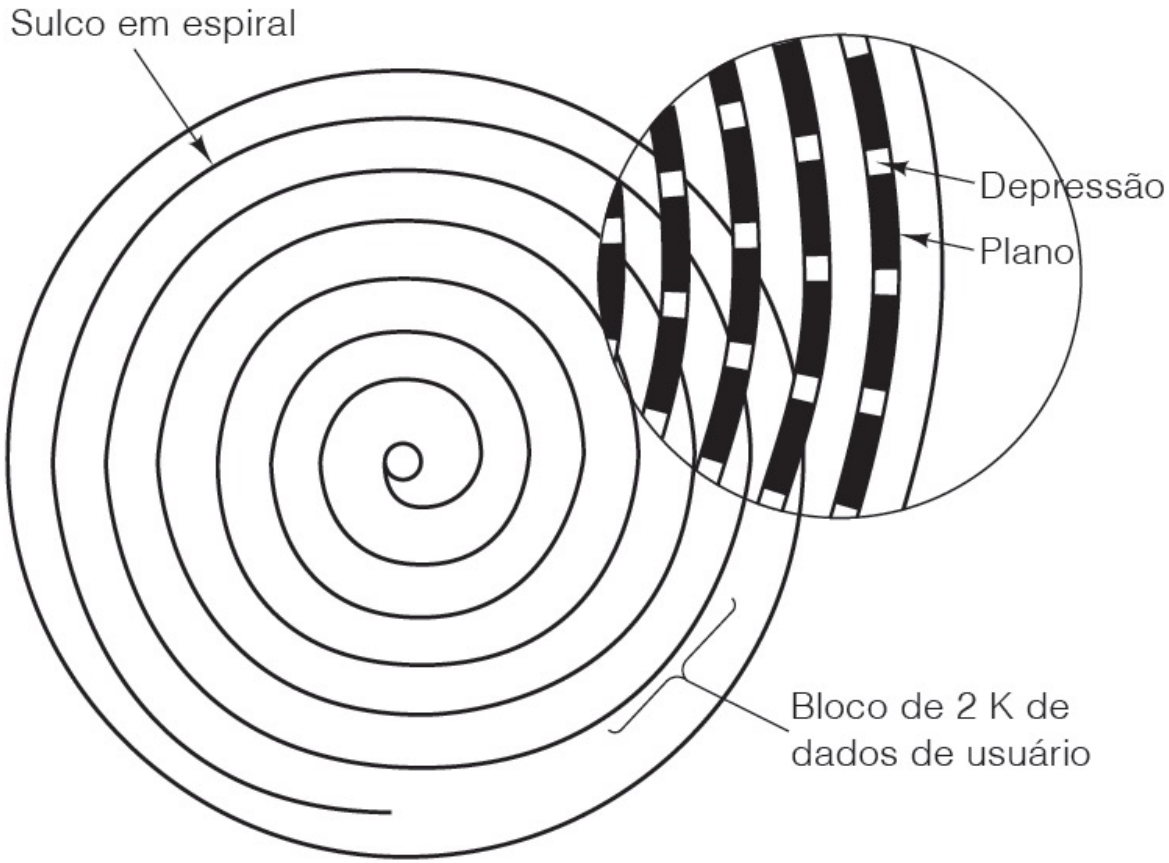
- Discos feitos de memória *flash* não volátil, geralmente denominados **discos em estado sólido**.

- Os discos *flash* são compostos
- de muitas células de memória
- *flash* em estado sólido.



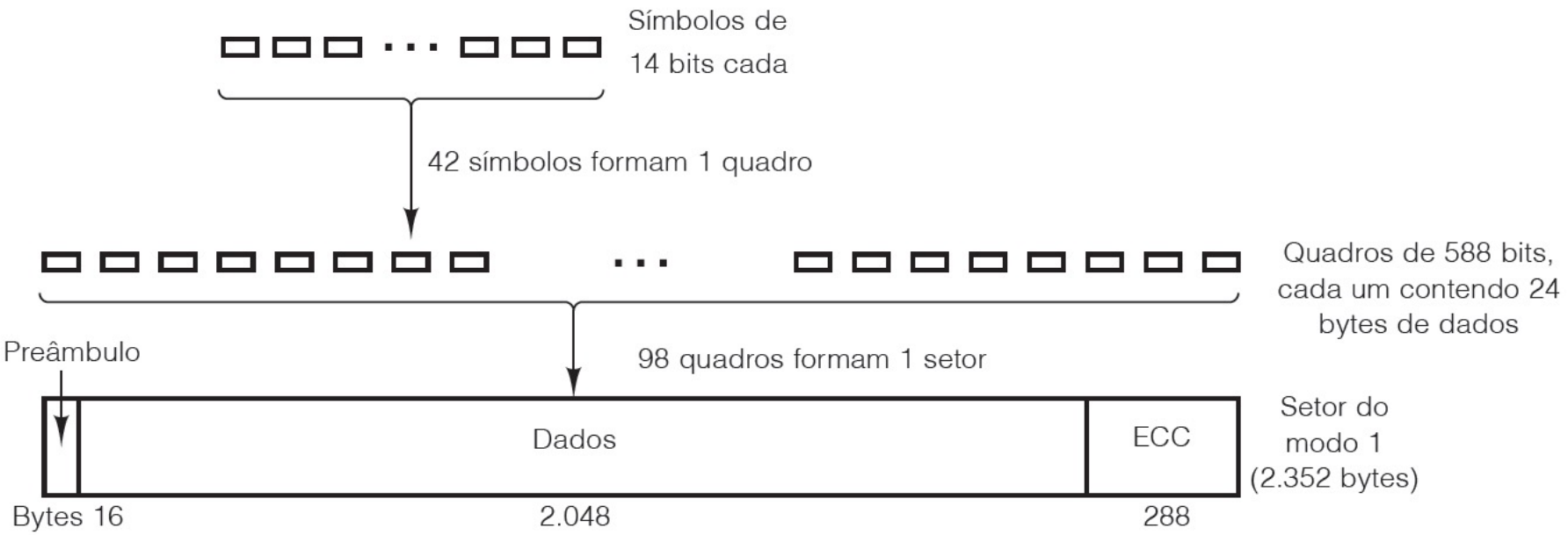
Memória secundária

- O formato básico de um **CD-ROM** consiste em codificar cada byte em um símbolo de 14 bits.



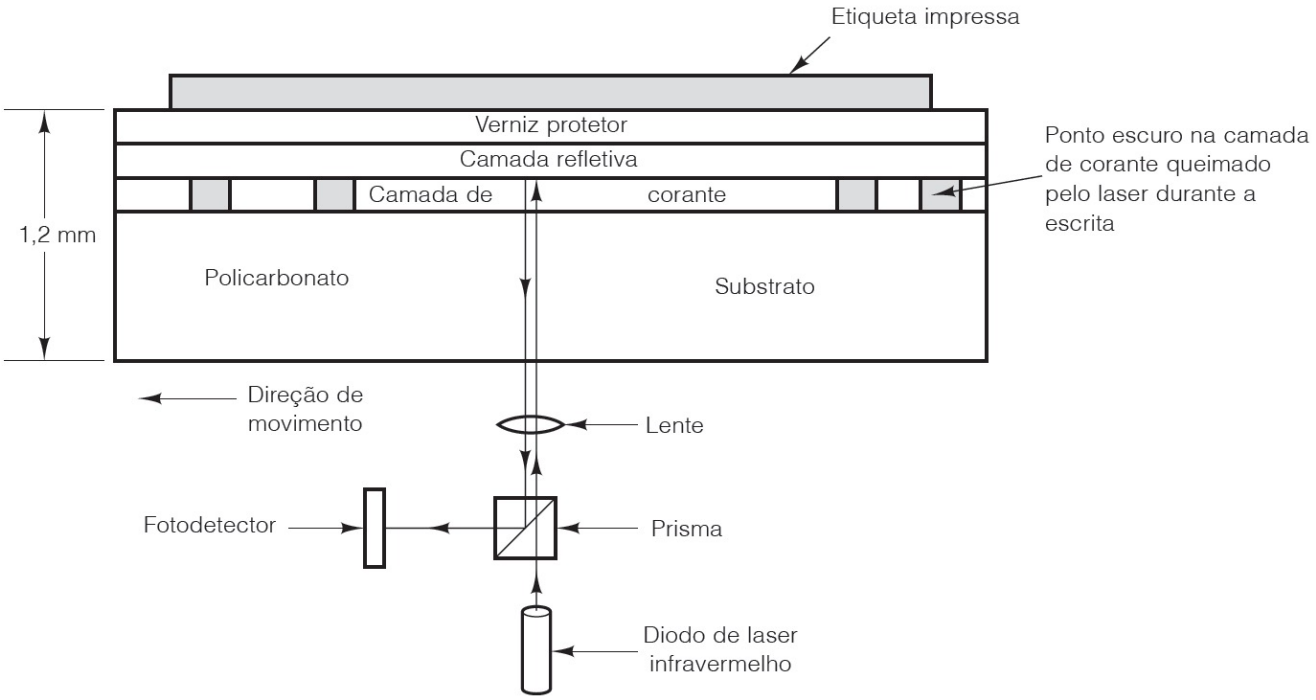
Memória secundária

- *Layout* lógico de dados em um CD-ROM.



Memória secundária

- Nos **CD-Rs** as diferentes refletividades das depressões e dos planos têm de ser simuladas. Isso é feito com a adição de uma camada de corante entre o policarbonato e a superfície refletiva.

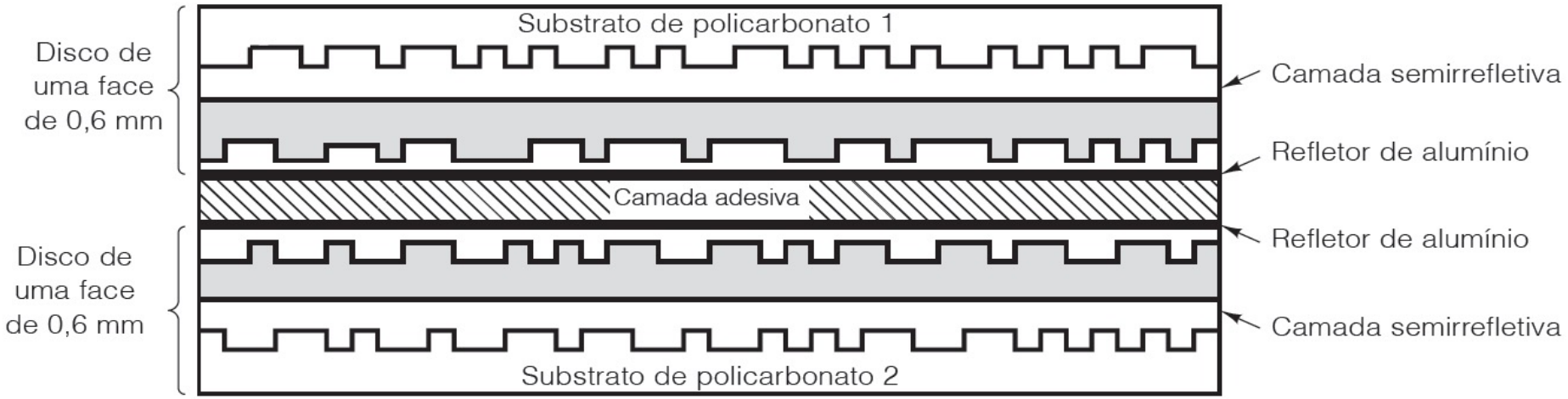


Memória secundária

- Uma tecnologia disponível agora é o **CD-RW** (CDs regraváveis), que usa um meio do mesmo tamanho do CD-R.
- Contudo, o CD-RW usa uma liga de prata, índio, antimônio e telúrio para a camada de gravação.
- Essa liga tem dois estados estáveis: cristalino e amorfo, com diferentes refletividades.
- Uma combinação de tecnologia e demanda por três indústrias imensamente ricas e poderosas resultou no **DVD**.

Memória secundária

- Disco de DVD de dupla face, dupla camada.

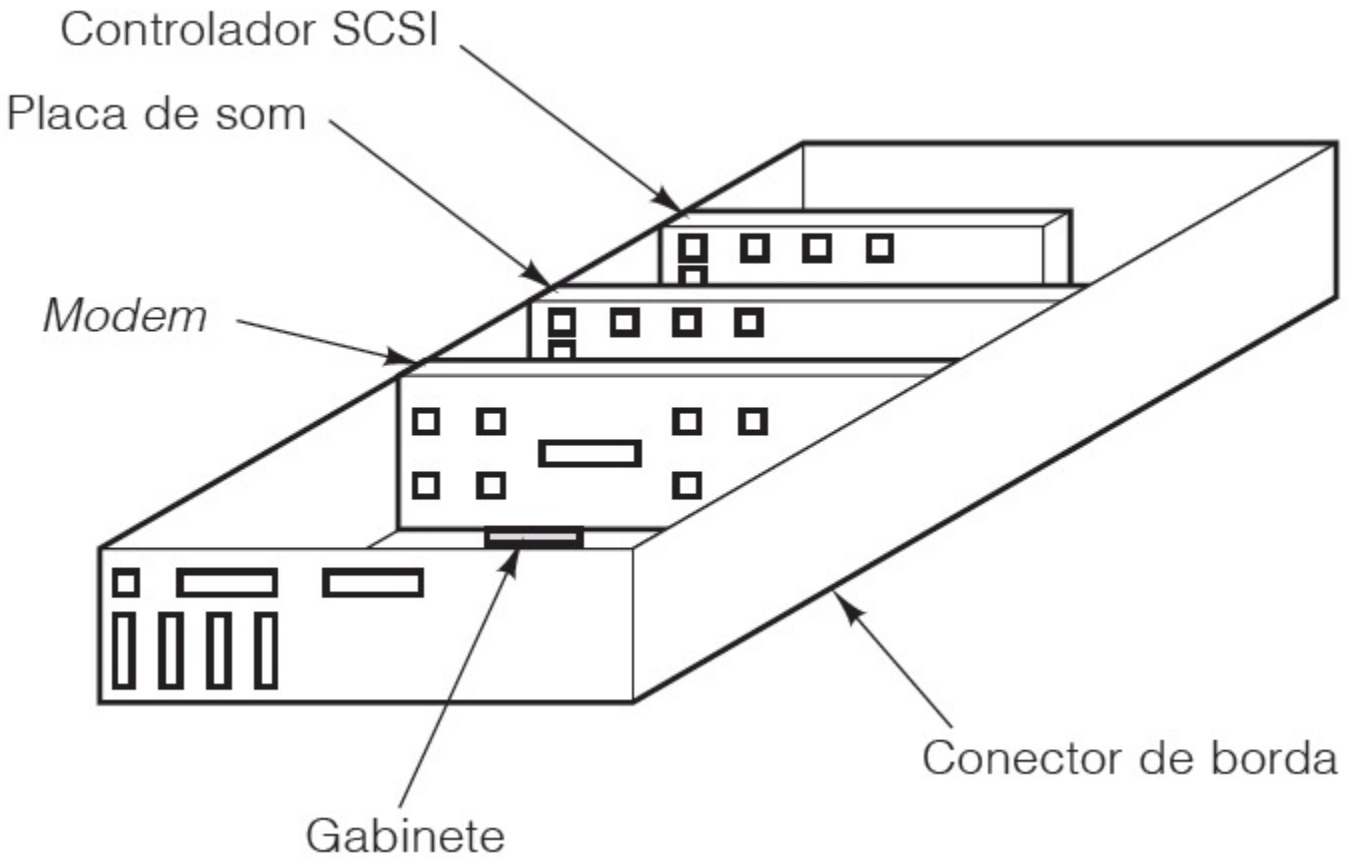


- O DVD mal acabara de ser lançado e seu sucessor já ameaçava torná-lo obsoleto.

- O **Blu-ray** (raio azul), assim chamado porque usa um laser azul, em vez do vermelho usado por DVDs.

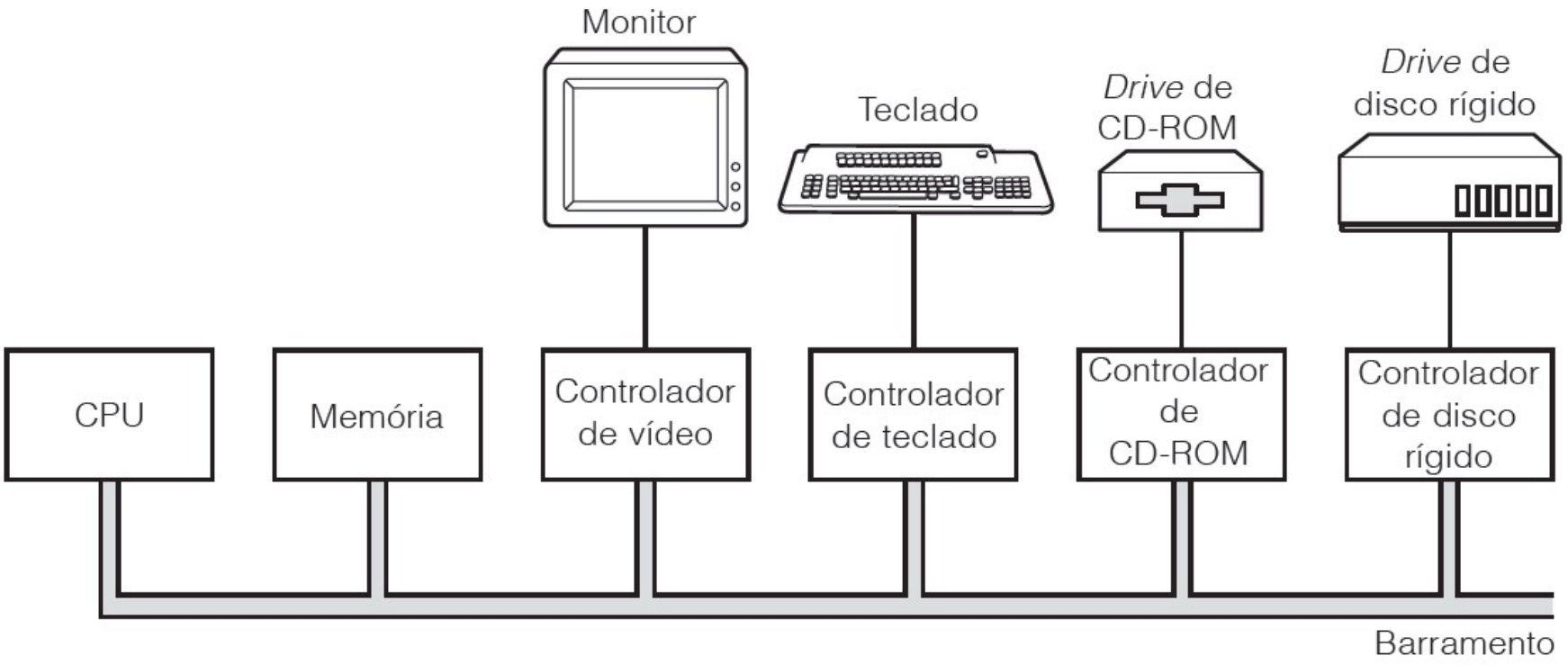
Entrada/Saída

- A maioria dos computadores pessoais e estações de trabalho tem uma estrutura semelhante à mostrada abaixo:



Entrada/Saída

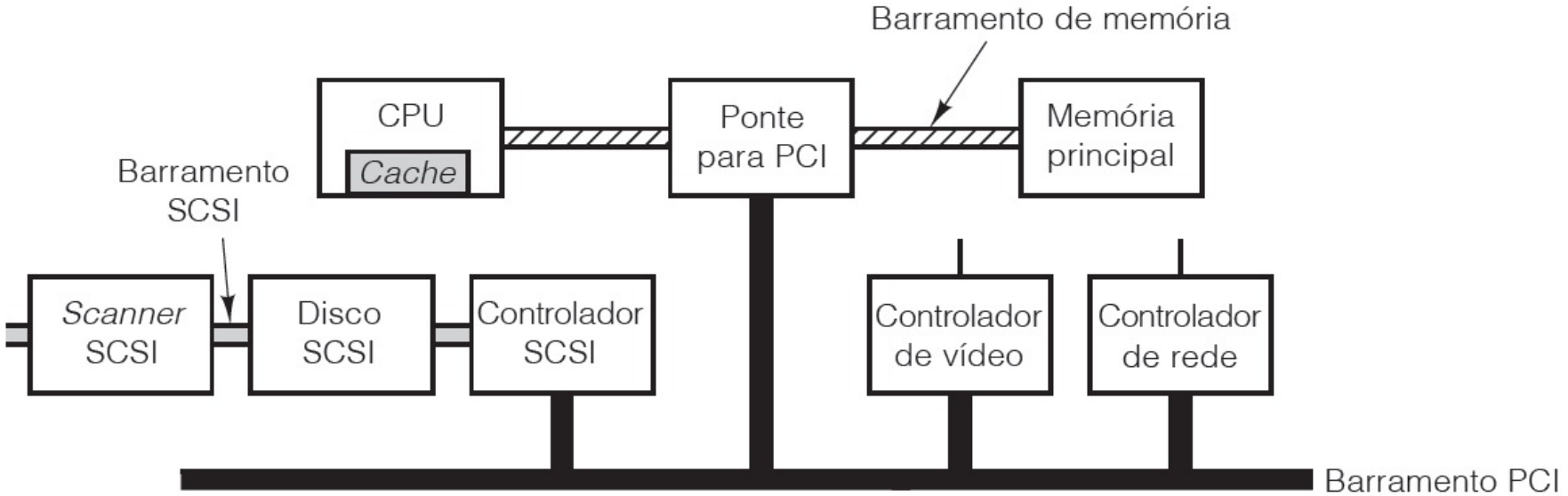
- A estrutura lógica de um computador pessoal simples pode ser vista abaixo:



Entrada/Saída

- A função de um controlador é controlar seu dispositivo de E/S e manipular para ele o acesso ao barramento.

- O mais popular deles é o **barramento PCI** - pode ser usado em muitas configurações, mas a figura abaixo ilustra uma configuração típica.



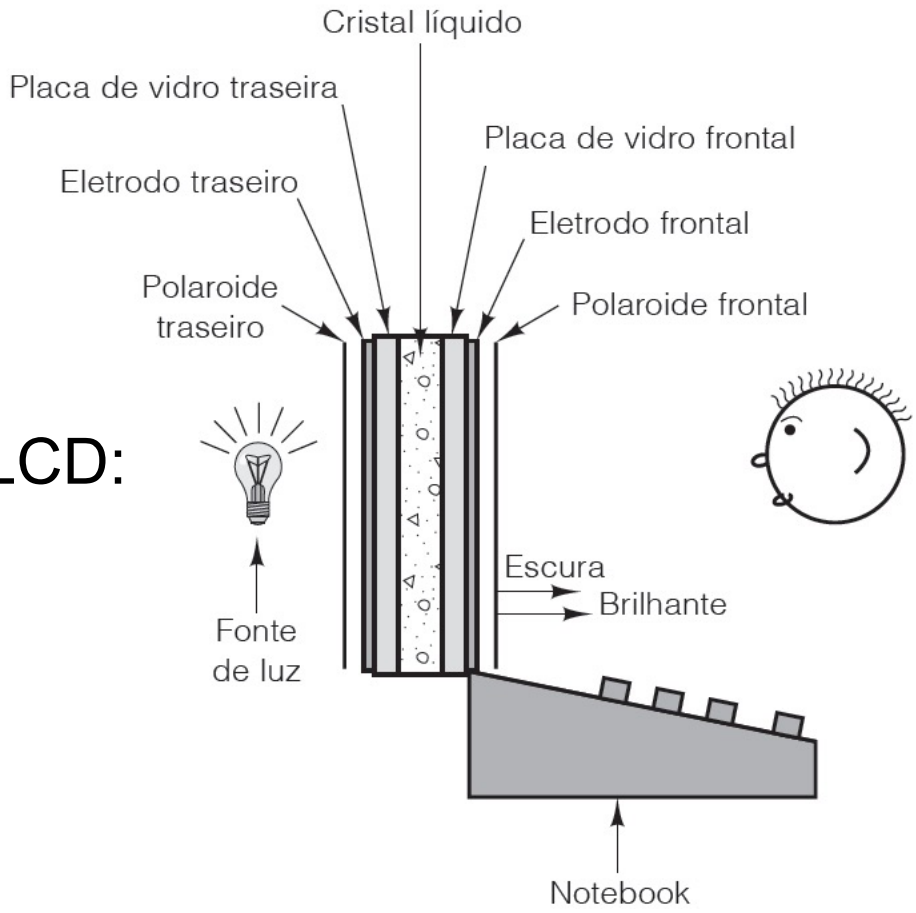
Entrada/Saída

- Há muitos tipos de dispositivos de E/S disponíveis.
- Terminais de computador consistem em duas partes: um teclado e um monitor.
- Dispositivos de toque (*touch screens*) podem ser encontrados em duas categorias: opacos e transparentes.
- Um dispositivo sensível ao toque opaco é o *touchpad* de um notebook.
- Um dispositivo transparente típico é a tela de um smartphone ou tablet.

Entrada/Saída

- A mais comum tecnologia de monitor de tela plana é o **LCD**.

- Construção de uma tela de LCD:

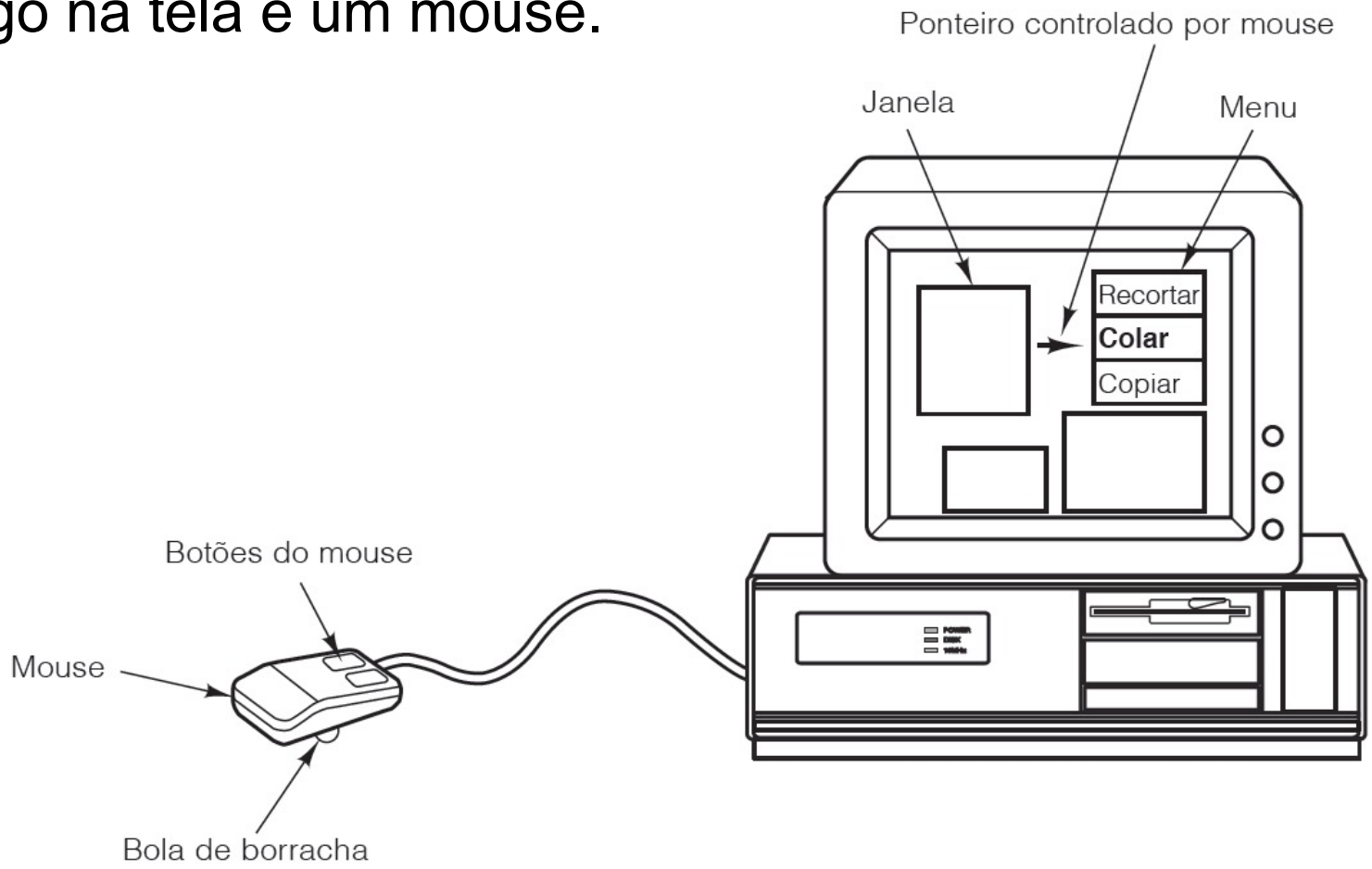


Entrada/Saída

- Quase todos os monitores são renovados de 60 a 100 vezes por segundo por uma memória especial, denominada **RAM de vídeo**.
- Essa memória tem um ou mais mapas de bits que representam a imagem da tela.
- Em uma tela com, por exemplo, 1.920 × 1.080 elementos de imagem, denominados pixels, uma RAM de vídeo conteria 1.920 × 1.080 valores, um para cada pixel.
- Ela poderia conter muitos desses mapas de bits, para permitir a passagem rápida de uma imagem para outra.

Entrada/Saída

- O meio mais comum de permitir que usuários apontem algo na tela é um mouse.



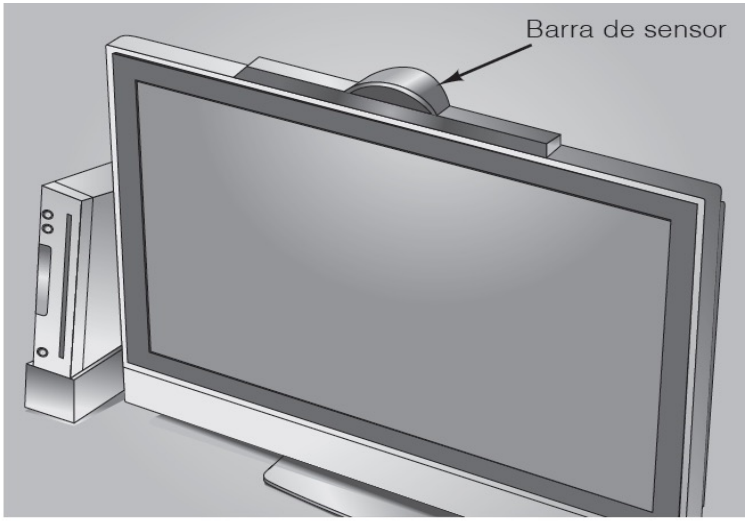
Entrada/Saída

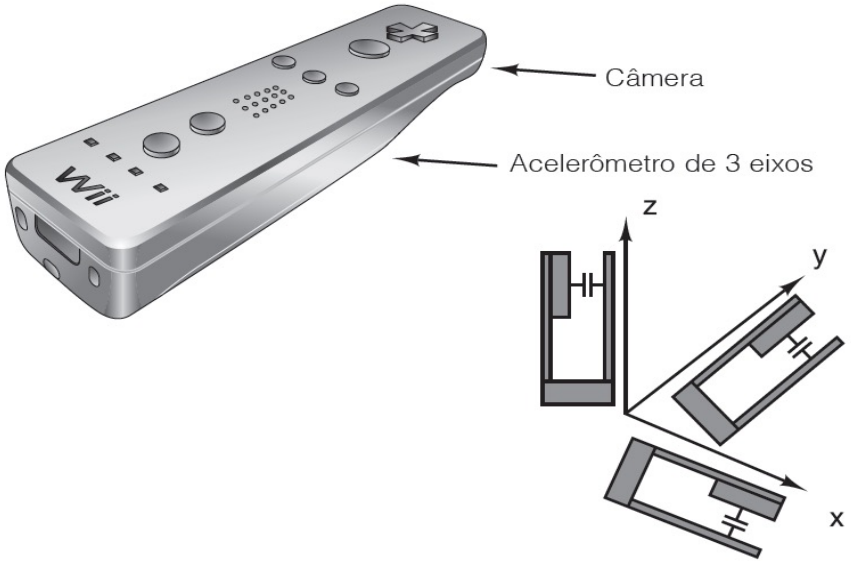
- Lançado em 2006 com o console de jogos Nintendo Wii, o **controlador Wiimote** contém botões tradicionais para jogos e mais uma capacidade de sensibilidade dupla ao movimento.

- Todas as interações com o Wiimote são enviadas em tempo real ao console de jogos, usando um rádio Bluetooth interno.

- Os sensores de movimento no Wiimote permitem que ele sinta seu próprio movimento nas três dimensões e mais; quando apontado para a televisão, ele oferece uma capacidade minuciosa para apontar.

Entrada/Saída





Entrada/Saída

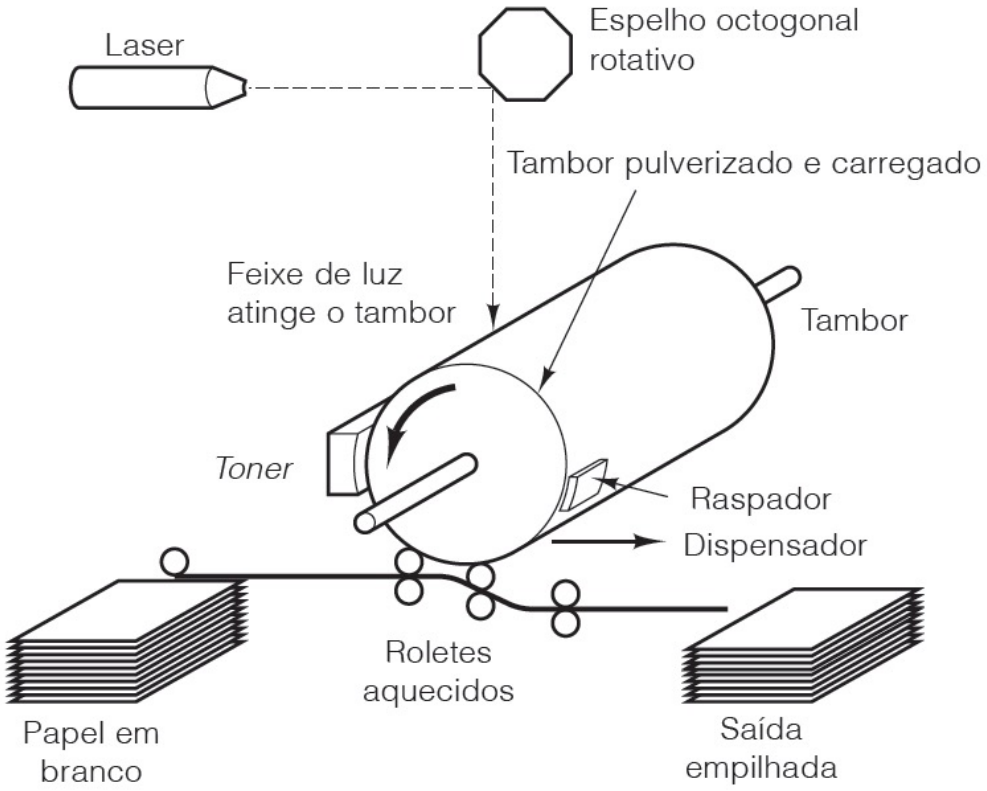
- O **Microsoft Kinect** usa apenas a visão do computador para determinar as interações do usuário com o console de jogos.

- Ele funciona sentindo a posição do usuário na sala, mais a orientação e o movimento de seu corpo.

- Os jogos são controlados por movimentos predeterminados de suas mãos, braços e qualquer outra coisa que os projetistas do jogo acreditarem que você deva mexer a fim de controlar seu jogo.

Entrada/Saída

- Talvez o desenvolvimento mais interessante da impressão desde que Johann Gutenberg inventou o tipo móvel no século XV é a **impressora a laser**.



Entrada/Saída

- Para impressão doméstica de baixo custo, as impressoras a jato de tinta são as favoritas.
- A cabeça de impressão móvel, que mantém os cartuchos de tinta, é varrida horizontalmente pelo papel por uma correia, enquanto a tinta é espirrada por minúsculos esguichos.
- As gotículas de tinta têm um volume de mais ou menos 1 picolitro, de modo que 100 milhões delas formam uma única gota d'água.

Entrada/Saída

- Uma variante da impressora a jato de tinta é a **impressora de tinta sólida**.
- Esse tipo de impressora aceita quatro blocos sólidos de uma tinta especial à base de cera, que são derretidos e passam para reservatórios de tinta quente.
- Outro tipo de impressora em cores é a **impressora a cera**.
- Ela tem uma larga fita encerada em quatro cores, segmentada em faixas do tamanho de páginas.

Entrada/Saída

- Ainda outro tipo de impressora em cores é a **impressora por sublimação de corante**, ou de tinta.
- Uma base contendo os corantes CMYK passa sobre um cabeçote de impressão térmico que contém milhares de elementos de aquecimento programáveis.
- As tintas são vaporizadas instantaneamente e absorvidas por um papel especial que está próximo.
- Por fim, a **impressora térmica**, que contém uma pequena cabeça de impressão com alguma quantidade de minúsculas agulhas que podem ser aquecidas.

Entrada/Saída

- *Modems* modernos funcionam a taxas de dados na faixa de 56 kbps, normalmente a taxas muito mais baixas.
- Eles usam uma combinação de técnicas para enviar múltiplos bits por baud, modulando a amplitude, a frequência e a fase.
- Quase todos eles são *full-duplex*, o que quer dizer que podem transmitir em ambas as direções ao mesmo tempo.
- *Modems* que só podem transmitir em uma direção por vez são denominados *half-duplex*. Linhas que só podem transmitir em uma direção são linhas *simplex*.

Entrada/Saída

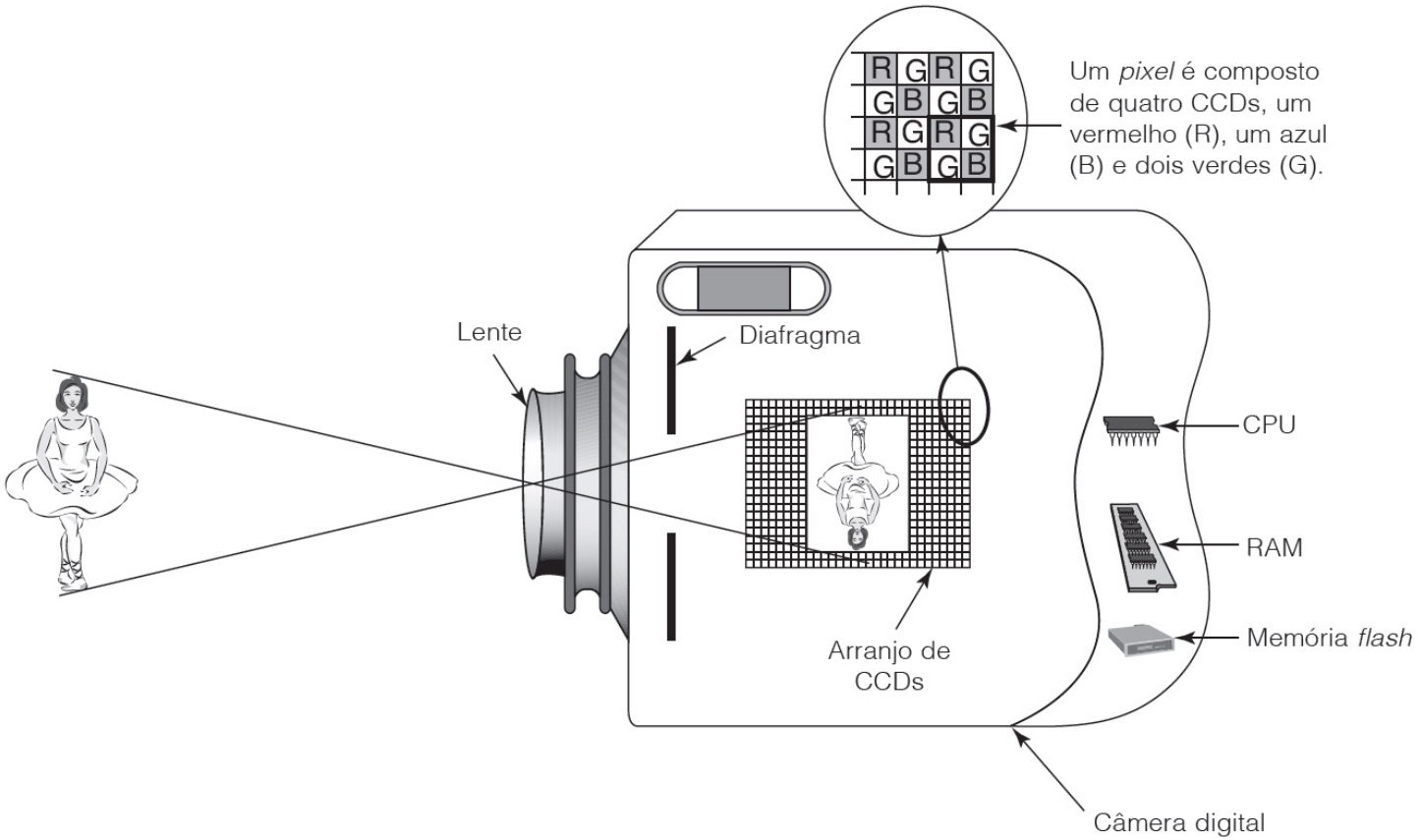
- Em cada cidade, a operadora por cabo tem uma central e uma grande quantidade de caixas cheias de dispositivos eletrônicos denominados **terminais de distribuição** (*headends*) distribuídos por todo o seu território.
- Estes estão conectados à central por cabos de alta largura de banda ou de fibra ótica.
- Cada terminal tem um ou mais cabos que passam por centenas de casas e escritórios.
- Cada cliente da provedora por cabo está ligado ao cabo que passa por sua casa ou escritório.

Entrada/Saída

- Assim, centenas de usuários compartilham o mesmo cabo até o terminal.
- O acesso à Internet requer um *modem* por cabo, um dispositivo que tem duas interfaces: uma com o computador e outra com a rede a cabo.
- A interface computador-*modem* a cabo é direta.
- Em geral, é Ethernet, exatamente como na ADSL.
- No futuro, o *modem* inteiro poderá se resumir a uma pequena placa inserida no computador.

Entrada/Saída

- Uma utilização cada vez mais popular de computadores é a fotografia digital.



Entrada/Saída

- Para transferir os caracteres para o computador, um número é designado a cada um, por exemplo, a = 1, b = 2, ..., z = 26, + = 27, – = 28.
- O mapeamento de caracteres para números inteiros é denominado **código de caracteres**.
- Um código de ampla utilização é denominado **ASCII**.
- Cada caractere ASCII tem 7 bits, o que permite 128 caracteres no total.
- Cada caractere ASCII é armazenado em um byte separado.

Entrada/Saída

- O conjunto de caracteres ASCII.

Hexa	Nome	Significado	Hexa	Nome	Significado
0	NUL	Null	10	DLE	Data Link Escape
1	SOH	Start Of Heading	11	DC1	Device Control 1
2	STX	Start Of TeXt	12	DC2	Device Control 2
3	ETX	End Of TeXt	13	DC3	Device Control 3
4	EOT	End Of Transmission	14	DC4	Device Control 4
5	ENQ	Enquiry	15	NAK	Negative AcKnowledgement
6	ACK	ACKnowledgement	16	SYN	SYNchronous idle
7	BEL	BELl	17	ETB	End of Transmission Block
8	BS	BackSpace	18	CAN	CANcel
9	HT	Horizontal Tab	19	EM	End of Medium
A	LF	Line Feed	1A	SUB	SUBstitute
B	VT	Vertical Tab	1B	ESC	ESCape
C	FF	Form Feed	1C	FS	File Separator
D	CR	Carriage Return	1D	GS	Group Separator
E	SO	Shift Out	1E	RS	Record Separator
F	SI	Shift In	1F	US	Unit Separator

Entrada/Saída

- O conjunto de caracteres ASCII.

Hexa	Car	Hexa	Car	Hexa	Car	Hexa	Car	Hexa	Car	Hexa	Car
20	Espaço	30	0	40	@	50	P	60	'	70	p
21	!	31	1	41	A	51	Q	61	a	71	q
22	"	32	2	42	B	52	R	62	b	72	r
23	#	33	3	43	C	53	S	63	c	73	s
24	$	34	4	44	D	54	T	64	d	74	t
25	%	35	5	45	E	55	U	65	e	75	u
26	&	36	6	46	F	56	V	66	f	76	v
27	'	37	7	47	G	57	W	67	g	77	w
28	(	38	8	48	H	58	X	68	h	78	x
29	)	39	9	49	I	59	Y	69	i	79	y

Entrada/Saída

- O conjunto de caracteres ASCII.

Hexa	Car	Hexa	Car	Hexa	Car	Hexa	Car	Hexa	Car	Hexa	Car
2A	*	3A	:	4A	J	5A	Z	6A	j	7A	z
2B	+	3B	;	4B	K	5B	[	6B	k	7B	{
2C	,	3C	<	4C	L	5C	\	6C	l	7C	|
2D	-	3D	=	4D	M	5D	]	6D	m	7D	}
2E	.	3E	>	4E	N	5E	^	6E	n	7E	~
2F	/	3F	?	4F	O	5F	_	6F	o	7F	DEL

Entrada/Saída

- Um grupo de empresas de computadores formou um consórcio para criar um novo sistema, denominado **Unicode**.

- Agora, o Unicode é suportado por algumas linguagens de programação (por exemplo, Java), alguns sistemas operacionais (por exemplo, Windows) e muitas aplicações.

- A ideia que fundamenta o Unicode é designar a cada caractere e símbolo um valor único de 16 bits, denominado **ponto de código**.

Entrada/Saída

- Por conseguinte, outro esquema de codificação foi desenvolvido e denominado **Formato de Transformação UTF-8 UCS**, que é Unicode na essência.

- Códigos UTF-8 têm tamanho variável, de 1 a 4 bytes, e podem codificar cerca de dois bilhões de caracteres.

- Ele é o conjunto de caracteres dominante em uso na Web.

- Uma das propriedades interessantes do UTF-8 é que os códigos de 0 a 127 são os caracteres ASCII, permitindo que sejam expressos em 1 byte (contra os 2 bytes do Unicode).

Entrada/Saída

- O esquema de codificação UTF-8.

Bits	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	0ddddddd					
11	110ddddd	10dddddd				
16	1110dddd	10dddddd	10dddddd			
21	11110ddd	10dddddd	10dddddd	10dddddd		
26	111110dd	10dddddd	10dddddd	10dddddd	10dddddd	
31	1111110d	10dddddd	10dddddd	10dddddd	10dddddd	10dddddd