



Centro de Ciência e Tecnologia
Laboratório de Ciências Matemáticas
Ciência da Computação

Arquitetura de Computadores

Aula 05

Memórias

2 Organização de sistemas de computadores

Objetivo:

2.2 Memória primária

2.3 Memória secundária

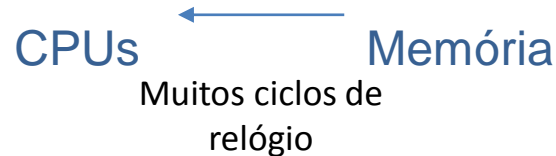
2.2.5 Memória cache

CPUs mais velozes:

- É possível colocar cada vez mais circuitos em um chip
- Paralelismo e operação superescalar

Memórias

- Aumento da capacidade do chip
- Não há aumento da velocidade



PROBLEMA ECONÔMICO, NÃO TECNOLÓGICO

2.2.5 Memória cache

Memórias velozes:

- Precisam estar localizadas no chip da CPU (**passar pelo barramento** é uma operação muito **lenta**)

Ideal (SONHO):

- ❖ Ter uma grande quantidade de memória rápida a preço baixo

Opções:

- ❖ Ter uma pequena quantidade de memória rápida
- ❖ Ter uma grande quantidade de memória lenta

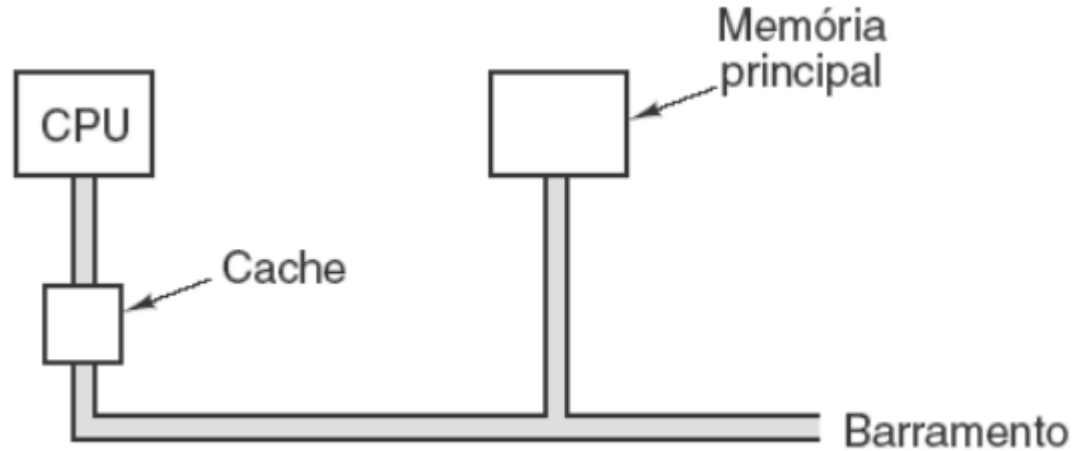
Solução:

- ❖ Combinar uma pequena quantidade de memória rápida com uma grande quantidade de memória lenta
- ❖ Obter (quase) a **velocidade** da memória rápida, a **capacidade** da memória lenta a um **preço** moderado



2.2.5 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.



Ideia básica: 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 a memória principal.

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




O sucesso ou o fracasso depende da fração das palavras que estão na cache

2.2.5 Memória cache

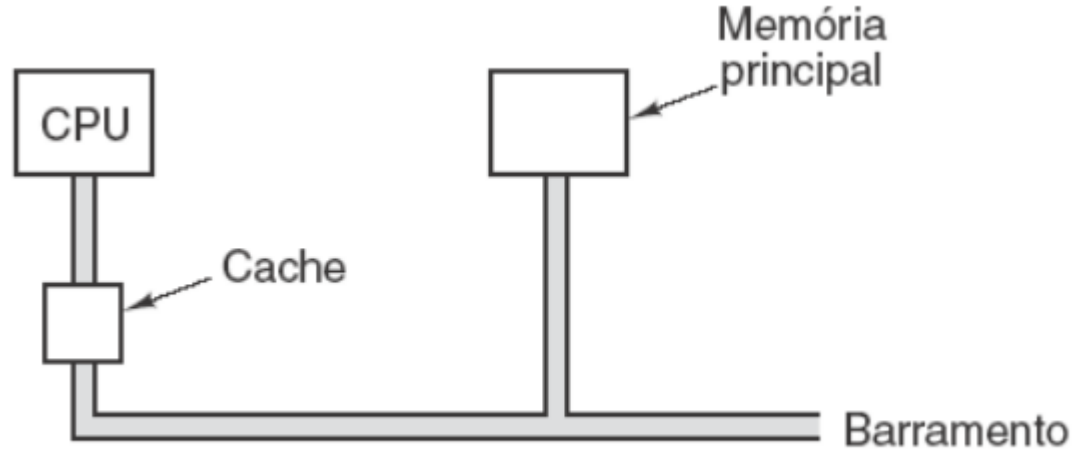
Características que sabemos por observação:

- Programas não acessam suas memórias em total aleatoriedade.
- Se uma dada referência à memória for para o endereço A , é provável que a próxima referência à memória estará na vizinhança geral de A .
- Ex.: no próprio programa, as instruções são buscadas em localizações consecutivas da memória. (exceto em desvios, e de chamadas de procedimentos.)
- Grande parte do tempo de execução de um programa é gasto em laços, nos quais um número limitado de instruções é executado repetidas vezes.

 **Princípio da localidade:** é a observação de que referências à memória feitas em qualquer intervalo de tempo curto tendem a usar penas uma pequena fração da memória, e forma a base de todos os sistemas de cache.

2.2.5 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.



c = tempo de acesso médio

m = tempo de acesso a memória principal

k = quantidade que uma palavra é lida ou escrita em um curto intervalo de tempo (será preciso 1 referência à memória lenta e $K-1$ referência à memória rápida)

h = taxa de acerto

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

Taxa de falha da cache: $1-h$

$$\text{tempo de acesso médio} = c + (1 - h) m$$

2.2.5 Memória cache

Usando o princípio da localidade como guia, memórias principais e caches são divididas em blocos de tamanho fixo.

Linhas de cache: nos referimos a esses blocos dentro da cache.

Quando a busca na cache falha, toda a linha de cache é carregada da memória principal para a cache, e não apenas a palavra que se quer.

Ex.:

- Linha de cache de 64 bytes de tamanho
- Referência ao endereço de memória 260 puxará a linha que consiste nos bytes 256 a 319 para uma linha de cache

Esse tipo de operação é mais eficiente do que buscar palavras individuais porque é mais rápido buscar k palavras de uma só vez do que uma palavra k vezes.

2.2.5 Memória cache

Projeto de cache

É uma questão de importância cada vez maior para CPUs de alto desempenho.

Aspectos de projeto:

- Tamanho da cache
 - Quanto maior, melhor seu funcionamento, mas também maior o custo.
- Tamanho da linha de cache
 - Uma cache de 16 KB pode ser dividida em até 1024 linhas de 16 bytes, 2048 linhas de 8 bytes e outras combinações.
- Maneira de organização da cache
 - Como ela controla quais palavras da memória estão sendo mantidas no momento.
- As instruções de dados são mantidos na mesma cache ou em caches diferentes. (próximo slide)
- Número de caches.
 - Não é incomum ter chips com uma cache primária no chip, uma cache secundária fora do chip, mas no mesmo pacote do chip e uma terceira ainda mais distante.

2.2.5 Memória cache

Cache unificada

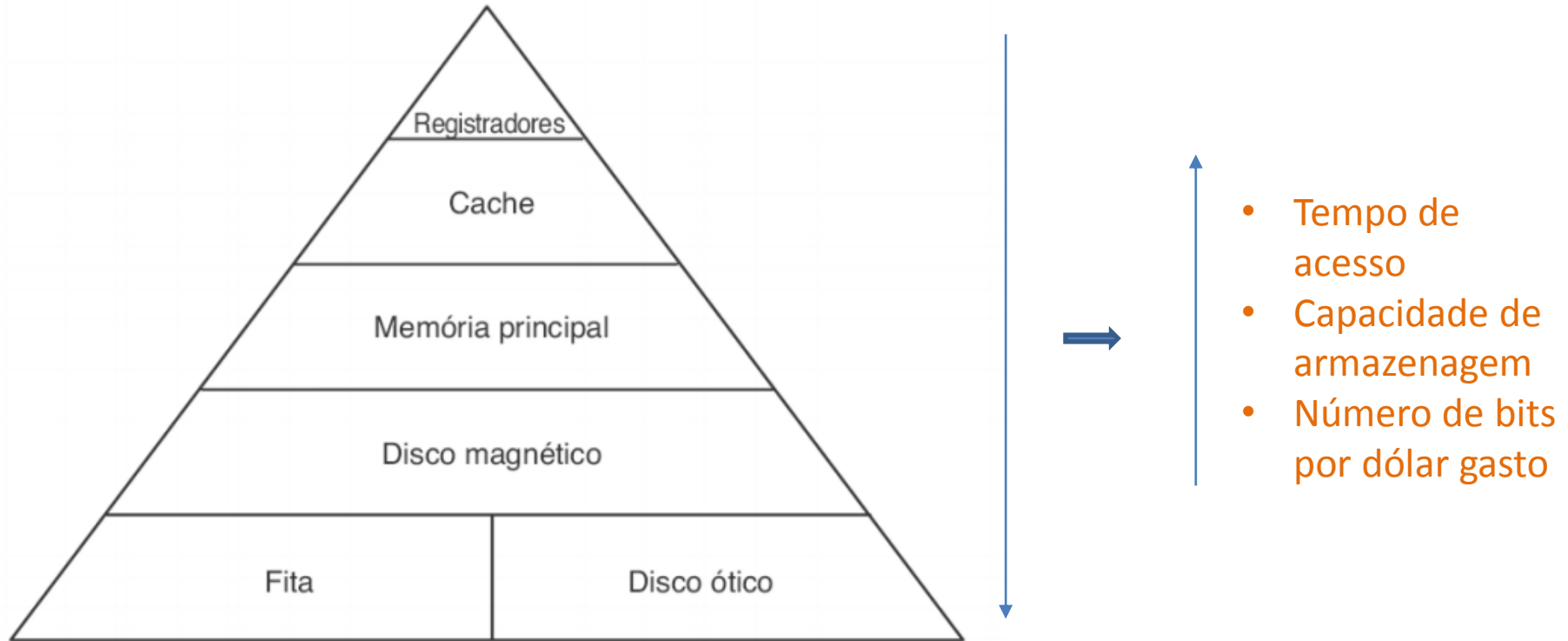
Instruções e dados usam a mesma cache, é um projeto mais simples e mantém automaticamente o equilíbrio entre buscas de instruções e de dados.

Cache dividida

Com instruções em uma cache e dados na outra. Esse projeto também é denominado arquitetura Harvard e essa referência volta ao passado até o computador Mark III de Howard Aiken, que tinha memórias diferentes para instruções e dados. A força que impele os projetistas nessa direção é a utilização muito difundida de CPUs com paralelismo.

2.3 Memória secundária

2.3.1 Hierarquia das memórias



Hierarquia de memória de cinco níveis.

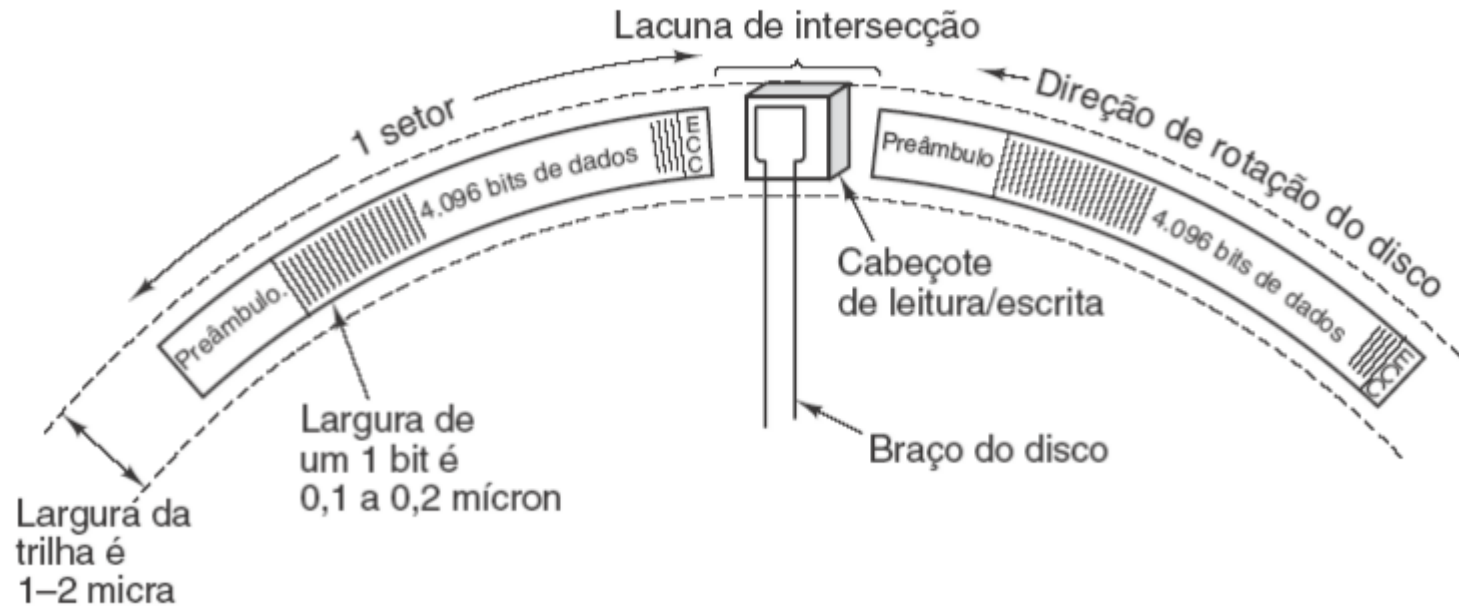
2.3 Memória secundária

Características

- ❖ Localização
 - Processador, interno, externo
- ❖ Capacidade
 - Tamanho de palavra; número de palavras
- ❖ Unidade de transferência
 - Palavra; bloco
- ❖ Método de acesso
 - Sequencial; direto (bloco sequencial); aleatório; associativo (por conteúdo)
- ❖ Desempenho
 - Tempo de acesso (op. R/W); tempo de ciclo;
 - Taxa de transferência (acesso aleatório; não aleatório)
- ❖ Tecnologia
 - Semicondutores; óptica; magneto-óptico
- ❖ Características físicas
 - Volátil/não volátil; apagável/não apagável
- ❖ Organização
 - Arranjo físico de bits para formar palavras

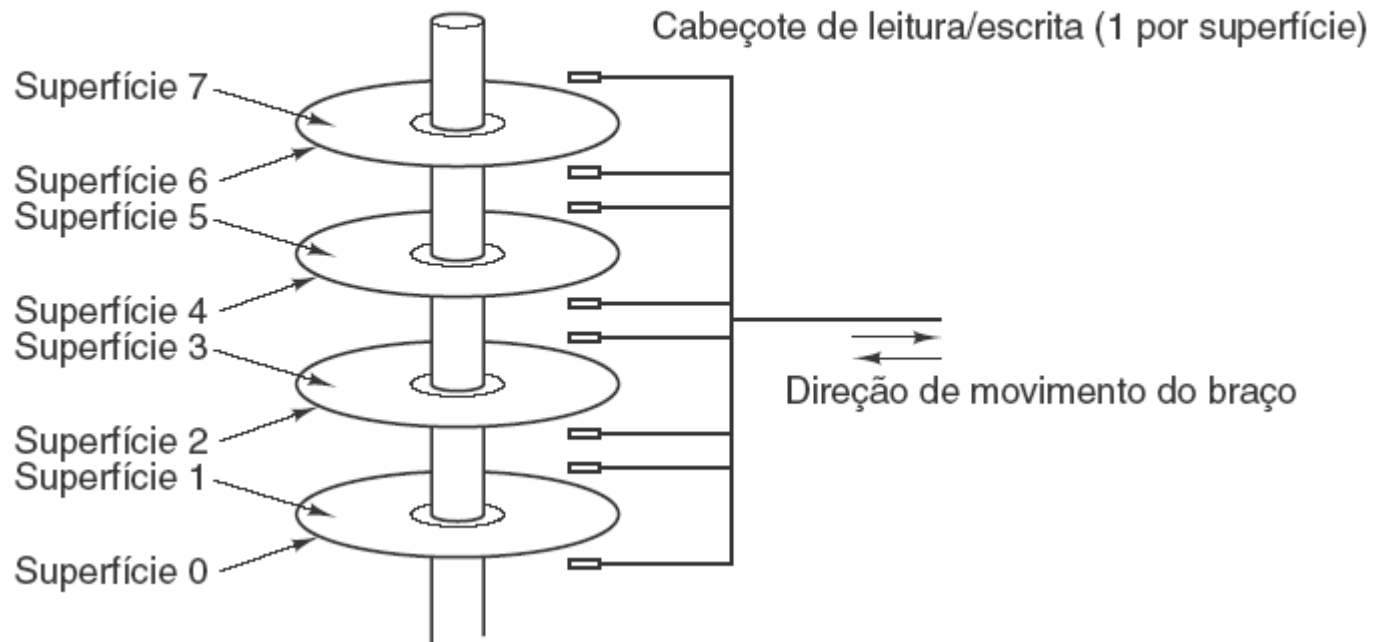
- ❖ Um disco é um **prato** circular construído de material não magnético, chamado de substrato, coberto por um material magnetizável. Tradicionalmente, o substrato tem sido alumínio ou um material de liga de alumínio.
- ❖ Mais recentemente, foram introduzidos substratos de vidro. O substrato de vidro apresenta diversos benefícios, incluindo os seguintes:
 - Melhoria na uniformidade da superfície do filme magnético, aumentando a confiabilidade do disco.
 - Redução significativa nos defeitos gerais da superfície, ajudando a diminuir os erros de leitura-gravação.
 - Capacidade de aceitar alturas de voo mais baixas (descritas mais adiante).
 - Melhor rigidez, para reduzir a dinâmica do disco.
 - Maior capacidade de suportar choque e danos.

2.3.2 Discos magnéticos



Porção de uma trilha de disco. São ilustrados dois setores

2.3.2 Discos magnéticos



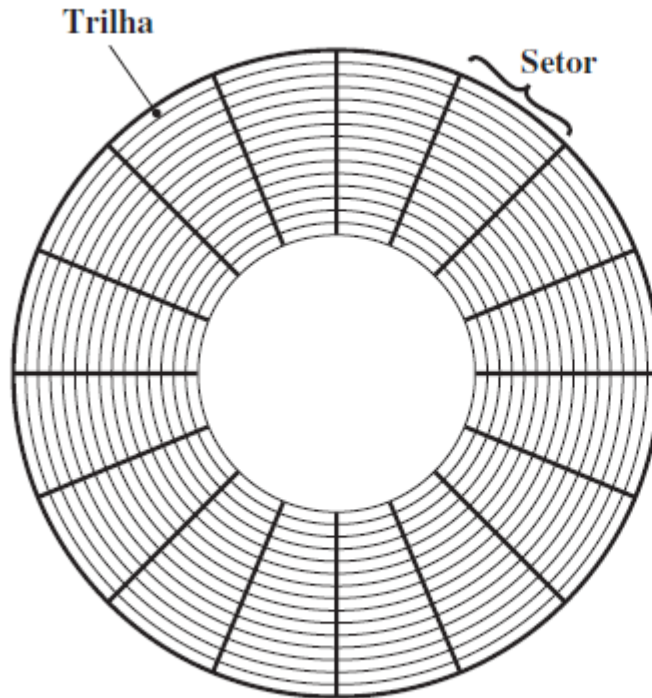
Um disco com quatro pratos.

2.3.2 Discos magnéticos

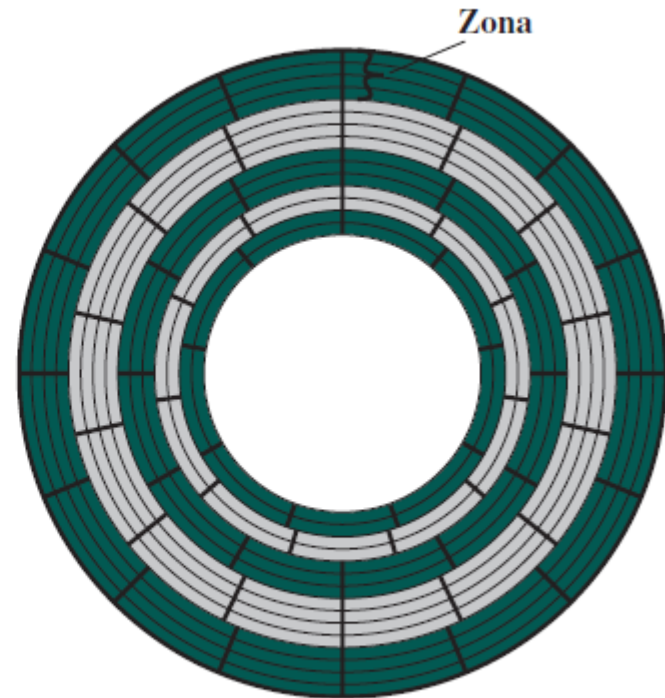


2.3 DISCO MAGNÉTICO

Comparação de métodos de layout de disco.

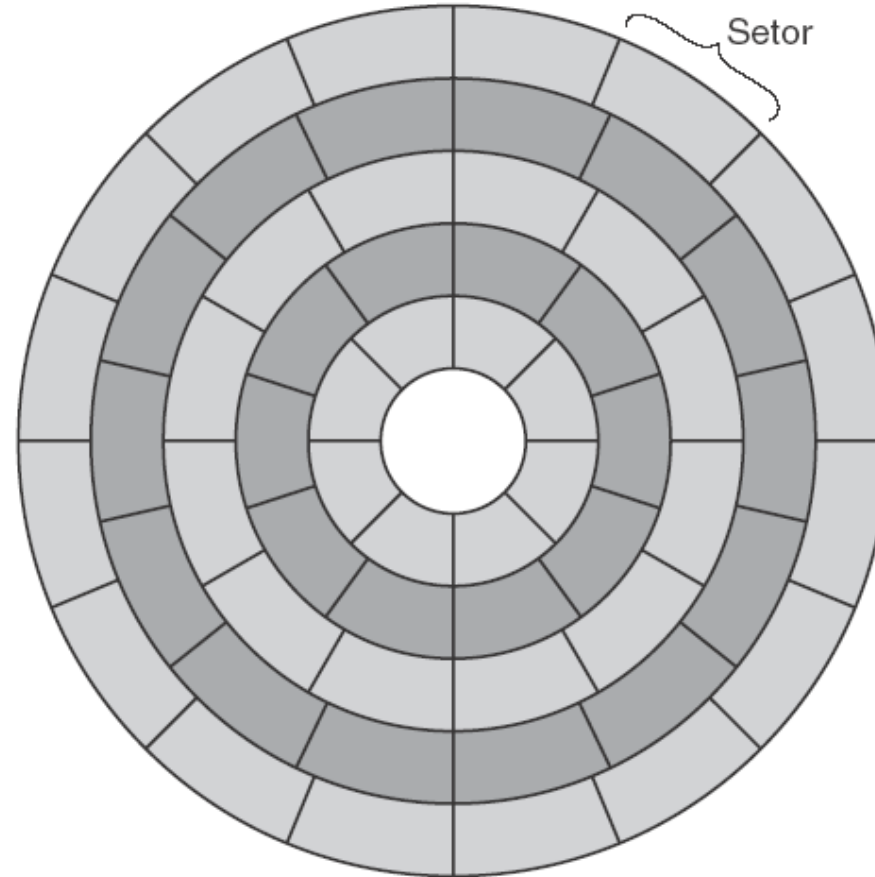


(a) Velocidade angular constante



(b) Gravação em múltiplas zonas

2.3.2 Discos magnéticos

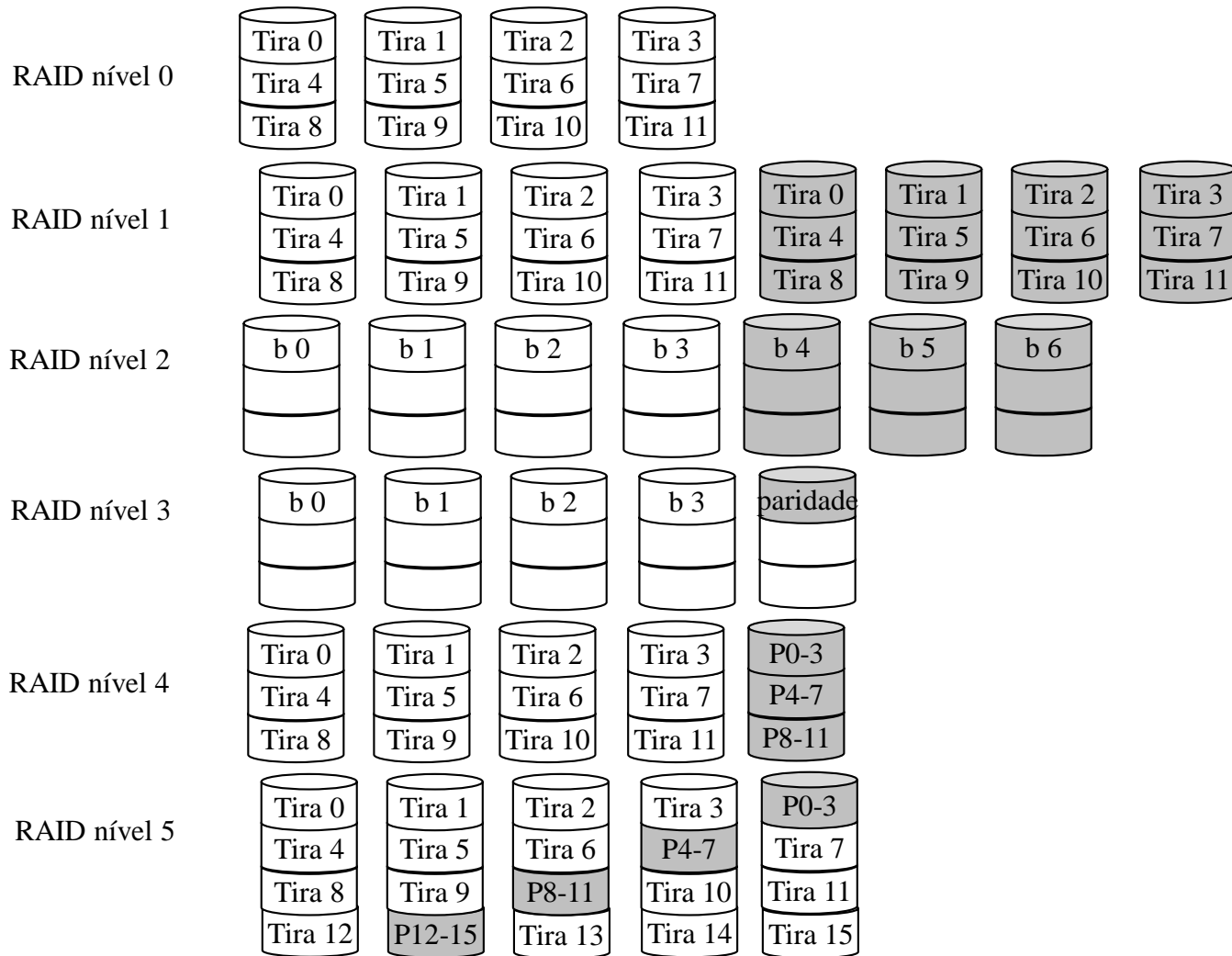


Um disco com cinco zonas. Cada zona tem muitas trilhas.

2.3.6 RAID

- ⌘ Projeto de discos paralelos (Patterson et al. 88): Redundant Array of Inexpensive Disk - Arranjo redundante de discos baratos independentes)
- ⌘ Oposição Política de SLED (Single Large Expensive Disk)
- ⌘ 6 níveis em uso comum (esquemas)
- ⌘ Controlado por controlador RAID
- ⌘ Alguns por controlador RAID SCSI que aparenta um único Disco
- ⌘ Conjunto de discos vistos pelo SO como um único disco
- ⌘ Dados distribuídos nos discos físicos
- ⌘ Podem usar redundância para armazenar informação de paridade

2.3.6 RAID



2.3.6 RAID 0

- ⌘ Não redundante
- ⌘ Tiras de dados por todos os discos
- ⌘ Tiras compostas por k setores

- ☒ tira 0: 0 a $k-1$

- ☒ tira 1: k a $2k-1$.. etc.

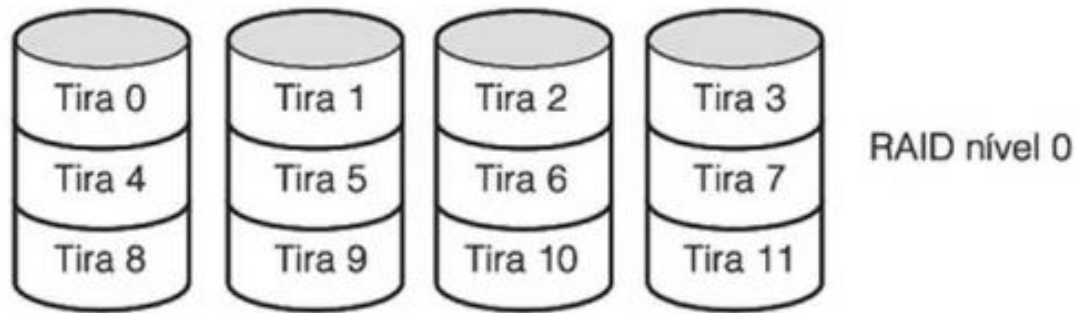
- ⌘ Tiras consecutivas em alternância circular (**striping**)

- ⌘ Funciona melhor com requisições grandes.

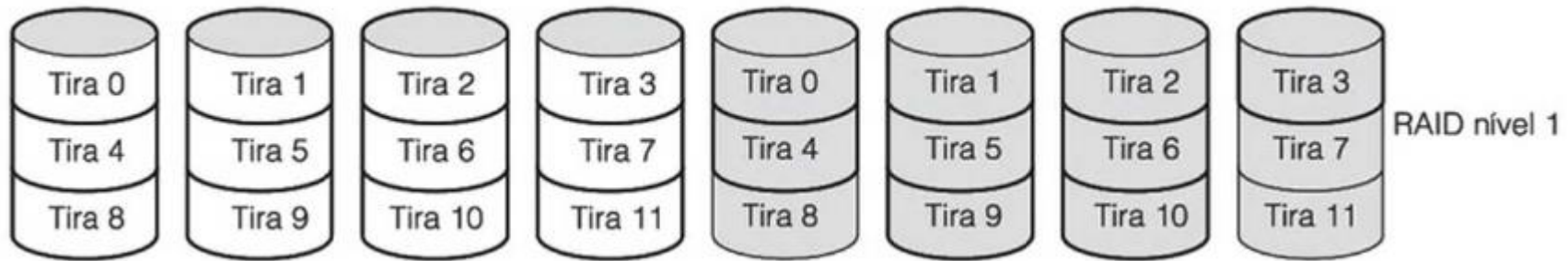
- ⌘ Desvantagem

- ☒ Falhas de um disco é multiplicado por 4... por tanto, confiabilidade bem pior que SLED

- ☒ Requisição de SO de informações para um único setor degrada o sistema

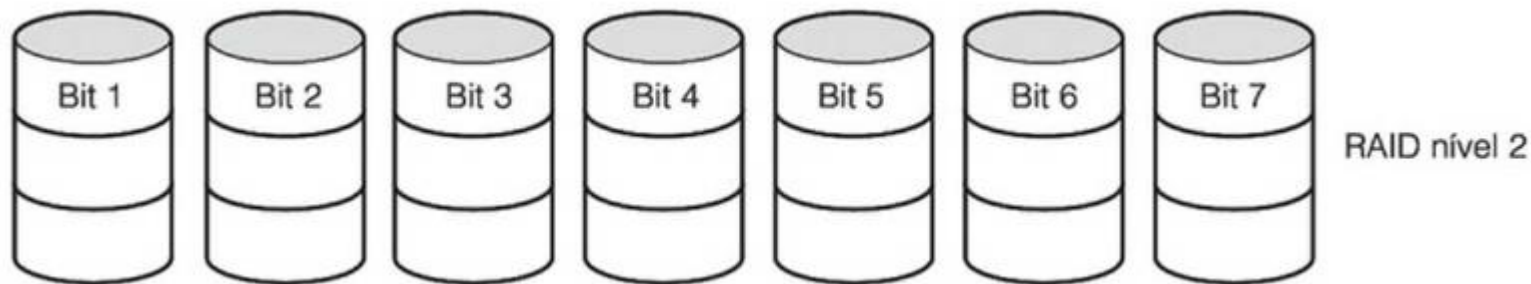


2.3.6 RAID 1



- ⌘ Redundante (RAID verdadeiro)
- ⌘ Duplica todos os discos(há 4 discos primários e 4 discos de backup)
- ⌘ Leitura: qualquer das duas cópias podem ser usadas (desempenho duas vezes melhor)
- ⌘ Escrita em ambos
- ⌘ Recuperação dos dados simples
 - ☒ Instalação de um novo drive, e copiar todo o drive de backup para ele
 - ☒ Requisição de SO de informações para um único setor degrada o sistema

2.3.6 RAID 2



⌘ Trabalha por palavra, possivelmente até por bytes

⌘ Vantagens

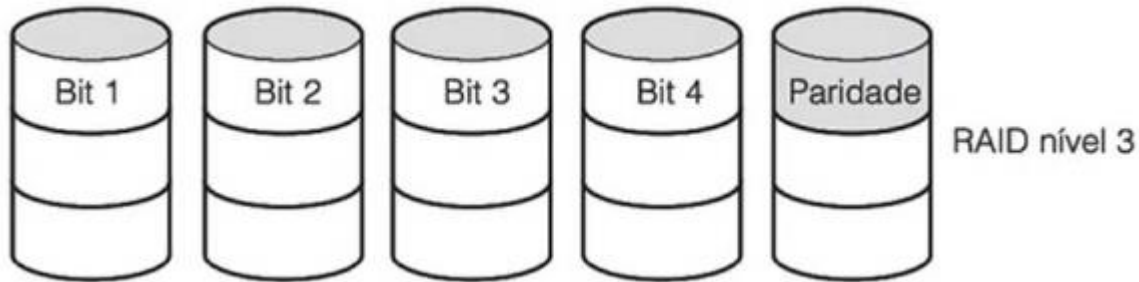
☑ Perder um drive não causa problema (Hamming Code pode detectar e corrigir o erro)

⌘ Desvantagens

☑ Rotação de todos dos drives devem ser sincronizadas

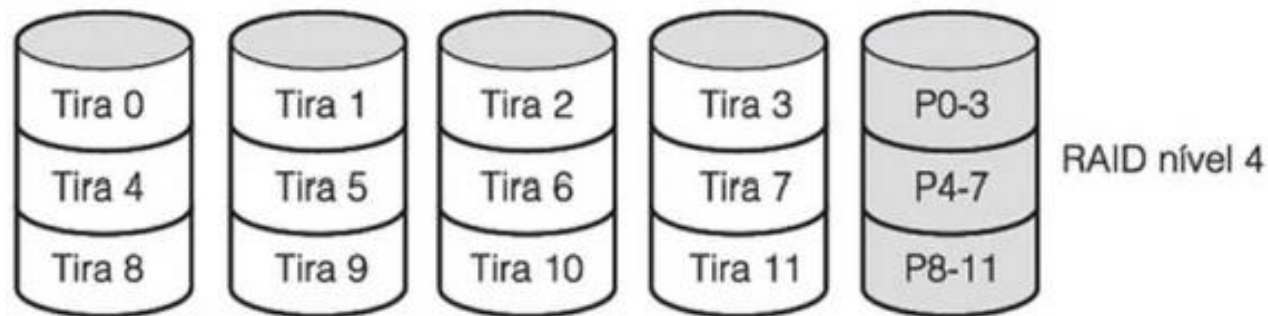
☑ Exige muito do controlador, uma vez que ele deve efetuar uma soma de verificação de Hamming a cada tempo de bit

2.3.6 RAID 3



- ⌘ Similar ao RAID 2
- ⌘ Apenas um disco redundante
- ⌘ Um único bit de paridade é computado para cada palavra de dados e escrito em um drive de paridade
- ⌘ Drive em exata sincronia
- ⌘ Correção de erro
 - ☒ Erros aleatórios não detectados: somente detecção, sem correção
 - ☒ Falha de drive: prevê correção total de erros de 1 bit, uma vez que a posição do bit defeituoso é conhecida

2.3.6 RAID 4



- ⌘ Trabalha com tiras

- ⌘ Não requer drives sincronizados

- ⌘ Como RAID 0, com paridade tira por tira escrita em um drive extra

- ⌘ Vantagem

- ☒ Se um drive falhar, os bytes perdidos podem ser recalculados com base no drive de paridade

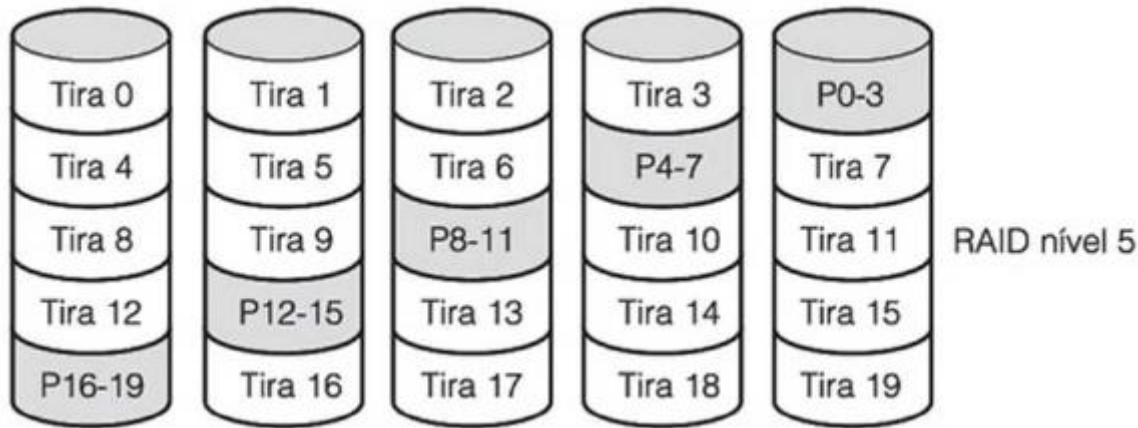
- ⌘ Desvantagens

- ☒ Desempenho medíocre para pequenas atualizações

- ☒ Um setor alterado, necessário ler todos os drives para recalcular a paridade

- ☒ Pequenas atualizações requer duas leituras e duas escritas

2.3.6 RAID 5



- ⌘ Similar RIAD 4
- ⌘ Distribui os bits de paridade uniformemente para todos os drives (alternância circular)
- ⌘ Numa falha de drive, a reconstrução é um processo complexo

2.3.6 RAID

Categoria	Nível	Descrição	Taxa de requisição de E/S	Taxa de transferência de dados	Aplicação típica
Intercalação de dados (striping)	0	Não-redundante	Tiras grandes: excelente	Tiras pequenas: excelente	Alto desempenho para dados não-críticos
Espelhamento	1	Espelha	Bom/razoável	Razoável/razoável	Unids de disco de sistemas; arquivos críticos
Acesso paralelo	2	Redundante (Hamming)	Pobre	Excelente	
	3	Paridade de bits intercalada	Pobre	Excelente	Grandes requisições de E/S; CAD
Acesso independente	4	Paridade de bloco intercalada	Ex/raz.	Raz/Pob.	
	5	Paridade de bloco intercalada e distribuida	Exc/Raz.	Raz/Pob.	Busca de dados/ grande volumen de leituras
	6	Paridade de bloco dupla intercalada e distribuida	Exc/pob.	Raz/Pob.	Grande disponibilidade de dados

2.3.7 Códigos de caracteres

O conjunto mínimo contém:

- 26 letras maiúsculas
- 26 letras minúsculas
- Algarismos de 0 a 9
- Conjunto de símbolos especiais (sinal de menos, vírgula, retorno, etc.)

Códigos de caracteres: mapeamento de caracteres para números inteiros

Padrões: ASCII e UNICODE

2.3.7 Códigos de caracteres

ASCII (American Standard Code for Information Interchange – código padrão americano para troca de informações)

- Contém 7 bits
- Total de 128 caracteres

Caracteres de controle, não impressos

Hex	Nome	Significado	Hex	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

Conjunto de caracteres ASCII: caracteres 0 – 31.

2.3.7 Códigos de caracteres

ASCII

Caracteres de
impressão

Hex	Car.	Hex	Car.	Hex	Car.	Hex	Car.	Hex	Car.	Hex	Car.
20	(Space)	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
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

Conjunto de caracteres ASCII: caracteres 32 – 127.

2.3.7 Códigos de caracteres

UNICODE

- Francês precisa de acento, alemão de sinais diacríticos, línguas europeias têm certas letras que não se encontram no ASCII...
- Novo sistema é criado: UNICODE
- Padrão Internacional
- Designar a cada caractere e símbolo um valor único de **16** bits, denominado **ponto de código**
- Símbolos de 16 bits tem 65.536 pontos de código
- Todos os idiomas do mundo usam cerca de 200.00 símbolos

2.3.7 Códigos de caracteres

UNICODE

- Adivinha só: 65.536 pontos de código não foram suficientes para satisfazer a todos, de modo que, em **1996, 16 planos adicionais de 16 bits cada foram acrescentados**, expandindo o número total de caracteres para **1.114.112**.
- Embora melhor que o ASCII, o Unicode por fim esgotou os pontos de código e também requer 16 bits por caractere para representar o texto ASCII puro, o que é um desperdício. Por conseguinte, outro esquema de codificação foi desenvolvido para resolver essas questões. Ele é denominado Formato de Transformação **UTF-8 UCS**, em que UCS significa Universal Character Set (**conjunto de caracteres universal**), que é Unicode na essência.

2.3.7 Códigos de caracteres

UNICODE

- 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.
- Os códigos de 0 a 127 são os caracteres ASCII, permitindo que sejam expressos em 1 byte (contra os 2 bytes do Unicode)
- Para caracteres que não são ASCII, o bit de alta ordem do primeiro byte é definido como 1, indicando que virão 1 ou mais bytes adicionais.

2.3.7 Códigos de caracteres

UNICODE

- Seis formatos diferentes são usados, conforme ilustra a Figura. Os bits marcados com “d” são bits de dados.

Bits	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	Oddddddd					
11	110dddddd	10ddddddd				
16	1110dddd	10ddddddd	10ddddddd			
21	1111Oddd	10ddddddd	10ddddddd	10ddddddd		
26	111110dd	10ddddddd	10ddddddd	10ddddddd	10ddddddd	
31	111111Od	10ddddddd	10ddddddd	10ddddddd	10ddddddd	10ddddddd

2.3.7 Códigos de caracteres

UNICODE

O UTF-8 tem uma série de vantagens em relação ao Unicode e outros esquemas.

- Se um programa ou documento utiliza apenas caracteres que estão no conjunto ASCII, cada um pode ser representado em 8 bits.
- O primeiro byte de cada caractere UTF-8 determina exclusivamente o número de bytes deste.
- Os bytes de continuação em um caractere UTF-8 sempre começam com 10, enquanto o byte inicial nunca começa assim, tornando o código autossincronizável. Em particular, no caso de um erro de comunicação ou memória, sempre é possível prosseguir e achar o início do próximo caractere (supondo que ele não tenha sido danificado).