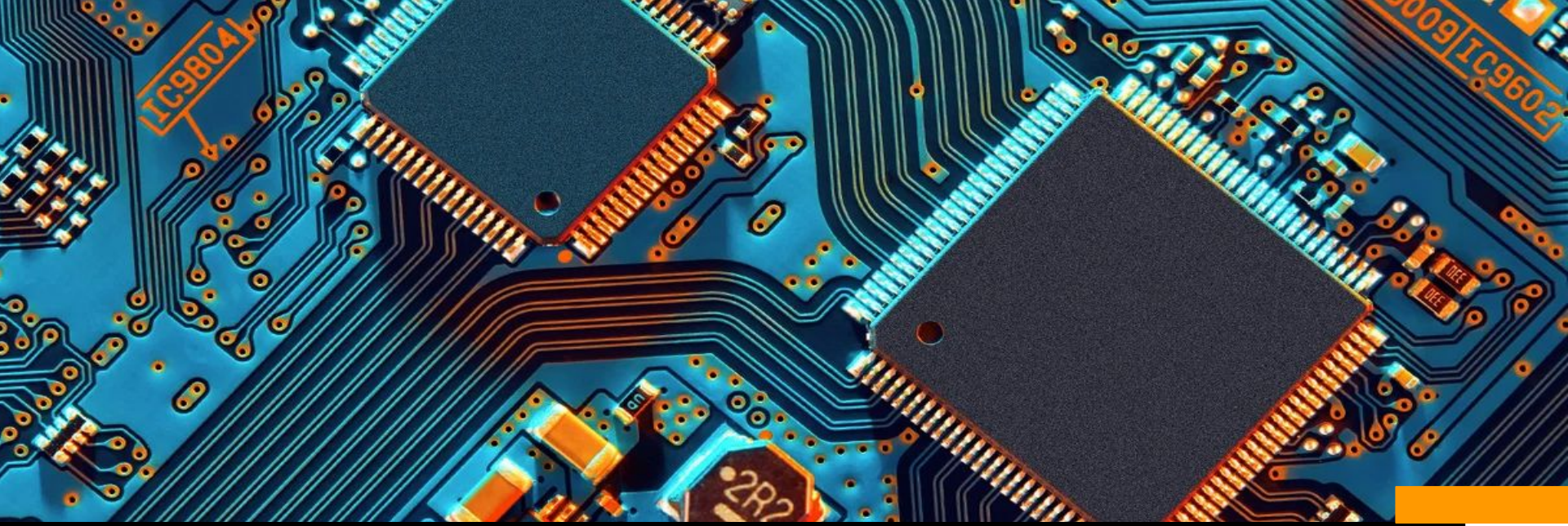


MC404AE - Organização Básica de Computadores e Ling. Montagem

Representação de Informações no Computador

Prof. Allan M. de Souza

UNICAMP

Representação de Informações

Como representar informações em um computador

- Números inteiros?
- Texto?
- Registros?
- Vetores?

Representação de Informações

Como representar informações em um computador?

- Informações são representadas através de dígitos binários, ou BITs (BInary digiTs).
- Dígitos 0 e 1
- Quantos estados (ou números) distintos podemos representar com 3 dígitos da base binária?

Representação de Informações

Quantos estados (ou números) distintos podemos representar com 3 dígitos da base binária?

- 8 estados se utilizarmos notação posicional
- 4 estados se utilizarmos notação não posicional
  - 1: 001, 010, 100 (um bit 1 e dois bits 0)
  - 2: 110, 101, 011 (um bit 0 e dois bits 1)
  - 3: 000 (três bits 0)
  - 4: 111 (três bits 1)

Representação de Informações

Notação posicional: valor do dígito depende da sua posição.

- **Exemplo:** Número decimal 132

  Valor do dígito 2 = 2

  Valor do dígito 3 = 30

  Valor do dígito 1 = 100

Representação de Informações

Notação posicional: valor do dígito depende da sua posição.

- **Exemplo:** Número decimal 132

  Valor do dígito 2 = 2

  Valor do dígito 3 = 30

  Valor do dígito 1 = 100

"Informações no computador são representadas através de números, codificados na base binária com notação posicional"

Representação de Informações

A quantidade de dígitos distintos define a base numérica.
Exemplos

- **Base 2**, ou binária => 2 dígitos distintos: 0 e 1
- **Base 8**, ou octal => 8 dígitos distintos: 0, 1, …, 7
- **Base 10**, ou decimal => 10 dígitos distintos: 0, …, 9
- …

Quais são os dígitos utilizados na base 16?

Representação de Informações

A quantidade de dígitos distintos define a base numérica.
Exemplos

- **Base 2**, ou binária => 2 dígitos distintos: 0 e 1
- **Base 8**, ou octal => 8 dígitos distintos: 0, 1, …, 7
- **Base 10**, ou decimal => 10 dígitos distintos: 0, …, 9
- …

Quais são os dígitos utilizados na base 16?

- Dígitos da base hexadecimal: 0, 1, …, 9, A, B, C, D, E, F

Bases Numéricas

Bases numéricas

Qual é a base dos números abaixo?

- FE03
- 8230
- 9210
- 1001

Bases numéricas

Qual é a base dos números abaixo?

- FE03
- 8230
- 9210
- 1001

Para distinguir temos que anotar o número com a base.

- $FE03_{16}$
- 1001_{10}
- 1001_{2}

Bases numéricas

Qual é a base dos números abaixo?

- FE03
- 8230
- 9210
- 1001

Em linguagens de programação esta notação é geralmente realizado com prefixos

Para distinguir temos que anotar o número com a base.

- $FE03_{16}$
- 1001_{10}
- 1001_{2}

- **0x**$FE03_{16}$
- **01**001_{10}
- **0b**1001_{2}

Exemplo em C

Bases numéricas

Qual é o valor de cada dígito nos números abaixo?

- 9210_{10}
- 1001_{2}

Bases numéricas

Qual é o valor de cada dígito nos números abaixo?

- 9210_{10}
- 1001_{2}

O valor de um dígito **d** em um número na base **t** é dado por:

- $d * t^{posição}$

Onde a posição é dada pela seguinte convenção:

Dígitos	0	0	0	0	9	2	1 0
Posição	7	6	5	4	3	2	1 0

Bases numéricas

Qual é o valor de cada número abaixo em decimal?

- 1001_2
- FF_{16}

Bases numéricas

Qual é o valor de cada número abaixo em decimal?

- 1001_2
- FF_{16}

O valor de um número na base t com n dígitos é o somatório dos valores dos dígitos:

$$N_{10} = \sum_{i=0}^{n-1} d_i * t^i$$

onde $\mathbf{d_i}$ é o dígito na posição **i**

Bases numéricas

Qual é o valor de cada número abaixo em decimal?

- $1001_2 = 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 9^{10}$
- $FF_{16} = F*16^1 + F*16^0 = 15x16 + 15x1 = 255_{10}$

O valor de um número na base t com n dígitos é o somatório dos valores dos dígitos:

$$N_{10} = \sum_{i=0}^{n-1} d_i * t^i$$

onde $\mathbf{d_i}$ é o dígito na posição **i**

Conversão de Bases Numéricas

Conversão de bases numéricas

Como fazemos para converter um número na representação decimal para a representação binária?

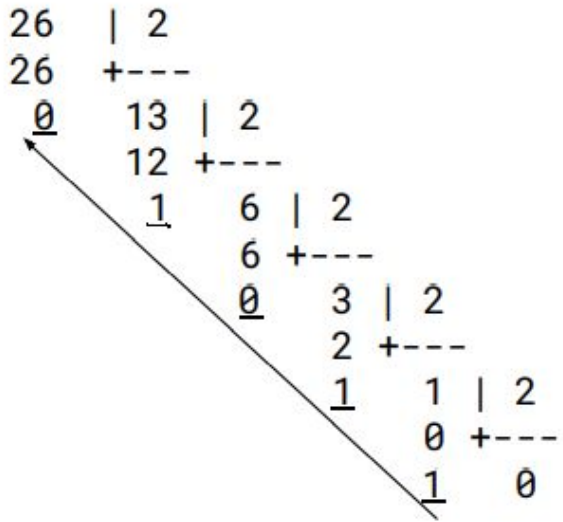
- **Por exemplo:** o número 26_{10}

```
26
```

Conversão de bases numéricas

Como fazemos para converter um número na representação decimal para a representação binária?

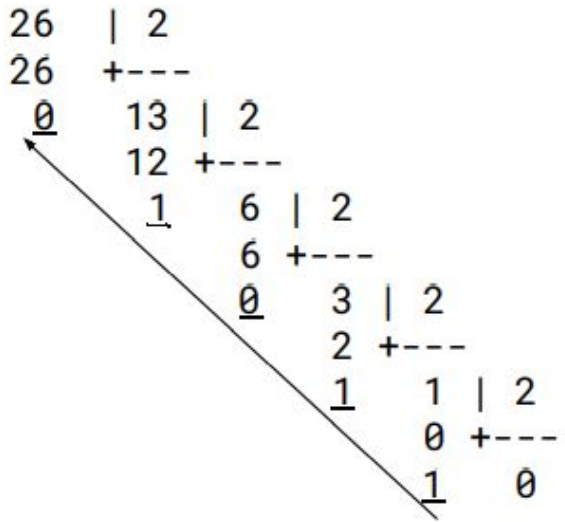
- **Por exemplo:** o número 26_{10}



Conversão de bases numéricas

Como fazemos para converter um número na representação decimal para a representação binária?

- **Por exemplo:** o número 26_{10}



E na base hexadecimal?

Conversão de bases numéricas

Tipo de conversão	Procedimento
Decimal => Binário	Divisões sucessivas por 2 até se obter zero no quociente. Leitura dos dígitos binários no resto de baixo para cima
Binário => Decimal	Soma de potências de 2 cujo expoente é a posição do bit e cujo coeficiente é o próprio bit.
Hexadecimal => Binário	Expandir cada dígito hexa em quatro dígitos binários segundo seu valor
Binário => Hexadecimal	Compactar cada quatro dígitos binários em um único dígito hexa segundo seu valor.
Decimal => Hexadecimal	Divisões sucessivas por 16 até se obter zero no quociente. Converter restos p/ dígitos hexadecimais. Leitura dos dígitos de baixo para cima
Hexadecimal => Decimal	Soma de potências de 16 cujo expoente é a posição do dígito e cujo coeficiente é o valor do próprio dígito hexa.

Bases numéricas - Exercícios

Qual o valor em binário dos seguintes números

- 151_{8}
- 139_{10}

Qual o valor em hexadecimal dos seguintes números

- 101001_{2}
- 16_{10}
- 240_{10}
- 20_{8}

Bases numéricas - Exercícios

Qual o valor em binário dos seguintes números

- $151_{6} = 1000011_{2}$
- $139_{10} = 10001011_{2}$

Qual o valor em hexadecimal dos seguintes números

- $101001_{2} = 29_{16}$
- $16_{10} = 10_{16}$
- $240_{10} = F0_{16}$
- $20_{8} = 10_{16}$

Números com Sinal

Números Sem Sinal

Na representação sem sinal, todos os bits são utilizados como dígitos do número.

- Registradores com 3 bits podem representar 8 números distintos: 0 a 7
  - $000_2 = 0_{10}$
  - $001_2 = 1_{10}$
  - $010_2 = 2_{10}$
  - $011_2 = 3_{10}$
  - $100_2 = 4_{10}$
  - $101_2 = 5_{10}$
  - $110_2 = 6_{10}$
  - $111_2 = 7_{10}$

Números Com Sina

Três tipos de codificação mais conhecidas

- Sinal e magnitude
- Complemento de 1
- Complemento de 2

Sinal e Magnitude

Na representação “**sinal e magnitude**” o bit mais à esquerda (o mais significativo) representa o sinal do número e os outros bits representam a magnitude.

Qual é o valor dos números abaixo na representação “**sinal e magnitude**” e sem sinal?

- $0001\ 0101_2$
- $1000\ 1010_2$

Sinal e Magnitude

Na representação “**sinal e magnitude**” o bit mais à esquerda (o mais significativo) representa o sinal do número e os outros bits representam a magnitude.

Qual é o valor dos números abaixo na representação “**sinal e magnitude**” e sem sinal?

- **0001 0101**$_2$
- **1000 1010**$_2$

Sinal e Magnitude

Na representação “**sinal e magnitude**” o bit mais à esquerda (o mais significativo) representa o sinal do número e os outros bits representam a magnitude.

Qual é o valor dos números abaixo na representação “**sinal e magnitude**” e sem sinal?

- $0001\ 0101_2$
- $1000\ 1010_2$

E estes números?

- $0000\ 0000_2$
- $1000\ 0000_2$

Sinal e Magnitude

Número binário	Sem sinal	Sinal e Mag.
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-0
101	5	-1
110	6	-2
111	7	-3

Complemento de 1

Na representação “**complemento de 1**” o bit mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

- Primeiro bit 0 => o número é positivo e o valor pode ser obtido da mesma maneira que na representação sem sinal
- Primeiro bit 1 => o número é negativo. Para descobrir a magnitude, basta inverter todos os bits e computar o valor na representação sem sinal.

Qual é o valor de 10010_2 ?

Complemento de 1

Na representação “**complemento de 1**” o bit mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

- Primeiro bit 0 => o número é positivo e o valor pode ser obtido da mesma maneira que na representação sem sinal
- Primeiro bit 1 => o número é negativo. Para descobrir a magnitude, basta inverter todos os bits e computar o valor na representação sem sinal.

Qual é o valor de 10010_2 ?

$Magnitude(10010_2) = 01101_2 = 13$

Portanto:; $10010_2 = -13$

Complemento de 1

Na representação “complemento de 1” o bit mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

Número binário	Sem sinal	Sinal e Mag.	Complemento de 1
000	0	0	0
001	1	1	1
010	2	2	2
011	3	3	3
100	4	-0	???
101	5	-1	???
110	6	-2	???
111	7	-3	???

Complemento de 1

Na representação “complemento de 1” o bit mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

Número binário	Sem sinal	Sinal e Mag.	Complemento de 1
000	0	0	0
001	1	1	1
010	2	2	2
011	3	3	3
100	4	-0	-3
101	5	-1	-2
110	6	-2	-1
111	7	-3	-0

Complemento de 2

Na representação “complemento de 2” o bit mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

- Primeiro bit 0 => o número é positivo e o valor pode ser obtido da mesma maneira que na representação sem sinal
- Primeiro bit 1 => o número é negativo. Para descobrir a magnitude, devemos **inverter** todos os bits, **somar 1**, e então computar o valor na representação sem sinal.

Qual é o valor de 10010_2?

Complemento de 2

Na representação “complemento de 2” o bit mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

- Primeiro bit 0 => o número é positivo e o valor pode ser obtido da mesma maneira que na representação sem sinal
- Primeiro bit 1 => o número é negativo. Para descobrir a magnitude, devemos **inverter** todos os bits, **somar 1**, e então computar o valor na representação sem sinal.

Qual é o valor de 10010_2?

$Magnitude(10010_2) = 01101_2 + 1_2 = 14$

Portanto: $10010_2 = -14$

Complemento de 2

Na representação “complemento de 2” o bit mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

Número binário	Sem sinal	Sinal e Mag.	Comp. de 1	Comp. de 2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	???
101	5	-1	-2	???
110	6	-2	-1	???
111	7	-3	-0	???

Complemento de 2

Na representação “complemento de 2” o bit mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

Número binário	Sem sinal	Sinal e Mag.	Comp. de 1	Comp. de 2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

Complemento de 2

Na representação “complemento de 2” o bit mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

A representação Complemento de 2 é a mais utilizada

000	0
001	1
010	2
011	3
100	-4
101	-3
110	-2
111	-1

Representação de Números

Número binário	Sem sinal	Sinal e Mag.	Comp. de 1	Comp. de 2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1
Maior	7	3	3	3
Menor	0	-3	-3	-4

Representação de Números

Número binário	Sem sinal	Sinal e Mag.	Comp. de 1	Comp. de 2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1
Maior	7	3	3	3
Menor	0	-3	-3	-4
Maior	$2^n - 1$	$2^{n-1} - 1$	$2^{n-1} - 1$	$2^{n-1} - 1$
Menor	**0**	$-(2^{n-1} - 1)$	$-(2^{n-1} - 1)$	$-(2^{n-1})$

Números no Computador

"Informações no computador são representadas através de números, codificados na base binária com notação posicional"

Quantos bits o computador usa para codificar cada número?

Números no Computador

"Informações no computador são representadas através de números, codificados na base binária com notação posicional"

Quantos bits o computador usa para codificar cada número?

- Computadores modernos codificam números com palavras de 8, 16, 32, 64 ou mais bits.
- Geralmente é uma potência de 2.

Uma arquitetura de 32 bits é uma arquitetura que é capaz de armazenar e realizar operações aritméticas em números com até 32 bits.

Complemento de 2

Números de 32 bits em Complemento de 2:

$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = 0_{10}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = +1_{10}$
$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = +2_{10}$
...
maxint
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = +2{,}147{,}483{,}646_{10}$
$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = +2{,}147{,}483{,}647_{10}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = -2{,}147{,}483{,}648_{10}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = -2{,}147{,}483{,}647_{10}$
$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = -2{,}147{,}483{,}646_{10}$
minint
...
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2 = -3_{10}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = -2_{10}$
$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = -1_{10}$

Aritmética Binária: Soma e Subtração

Como no ensino fundamental: (vai-um/vem-um)

```
  0111  (7)      0111 (7)      0110 (6)
+ 0110  (6)    - 0110 (6)    - 0101 (5)
------------  ------------  ------------
  1101 (13)      0001 (1)      0001 (1)
```

Aritmética Binária: Soma e Subtração

Como no ensino fundamental: (vai-um/vem-um)

```
  0111  (7)      0111 (7)      0110 (6)
+ 0110  (6)    - 0110 (6)    - 0101 (5)
  1101 (13)      0001 (1)      0001 (1)
```

Subtração em complemento de 2 pode ser feita com uma soma (A - B = A + (-B)).

- Ex: 7 - 6 = 7 + (-6)

```
  0111 (+7)
+ 1010 (-6)
  0001 (=1)
```

Aritmética Binária: Overflow

Overflow: quando o resultado é maior (menor) do que a palavra do computador pode representar.

- **Exemplo:** Ocorre overflow na operação abaixo?

```
  0111 (7)
+ 0001 (1)
------
  1000
```

Aritmética Binária: Overflow

Overflow: quando o resultado é maior (menor) do que a palavra do computador pode representar.

- **Exemplo:** Ocorre overflow na operação abaixo?

```
  0111  (7)
+ 0001  (1)
------
  1000
```

Na representação sem sinal não ocorre *overflow.* Note que 7 + 1 = 8

Na representação Complemento de 2 ocorre *overflow.* Note que 7 + 1 = -8

Aritmética Binária: Detecção de Overflow

- Não ocorre overflow quando adicionamos um número positivo a um número negativo

- Não ocorre overflow quando os sinais dos números são os mesmos na subtração

- Ocorre overflow quando os valores afetam o sinal:
  - Somar dois números positivos resulta em um número negativo
  - Somar dois números negativos resulta em um número positivo
  - Subtrair um número negativo de um positivo resulta em um negativo
  - Subtrair um número positivo de um negativo resulta em um positivo

Aritmética Binária: Detecção de Overflow

Exercício: Compute o resultado da operação abaixo e verifique se houve overflow

4 + 5 em uma representação com números sinalizados de **8 bits**

$$\begin{array}{r} (4) \\ +\ (5) \\ \hline \end{array}$$

Aritmética Binária: Detecção de Overflow

Exercício: Compute o resultado da operação abaixo e verifique se houve overflow

4 + 5 em uma representação com números sinalizados de **4 bits**

$$\begin{array}{r} (4) \\ +\ (5) \\ \hline \end{array}$$

Representação de Caracteres

Representação de Caracteres

Cada caractere é associado a um número distinto. Existem diversos padrões.

Exemplo: Padrão ASCII - American Standard Code for Information -- Usa 7 bits, (128 caracteres distintos)

64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I

96	´
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i

48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9

[redacted]

Cada caractere é associado a um número distinto.

- ASCII usa 7 bits
- Um texto é armazenado como uma cadeia de caracteres!
  - Posições consecutivas da memória!

[redacted]

Cada caractere é associado a um número distinto.

- ASCII usa 7 bits
- Um texto é armazenado como uma cadeia de caracteres!
  - Posições consecutivas da memória!

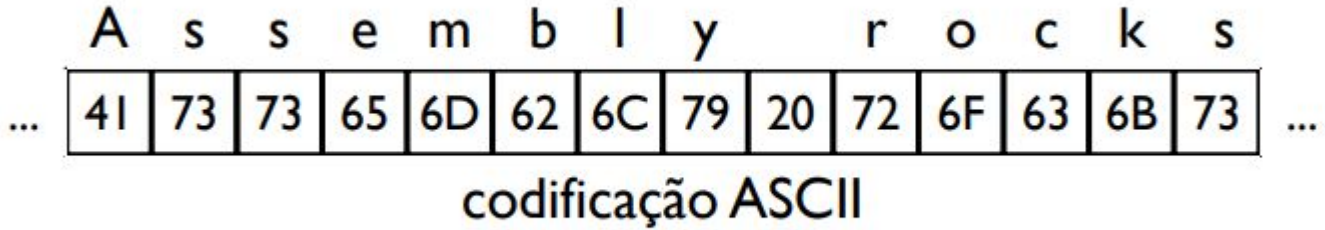


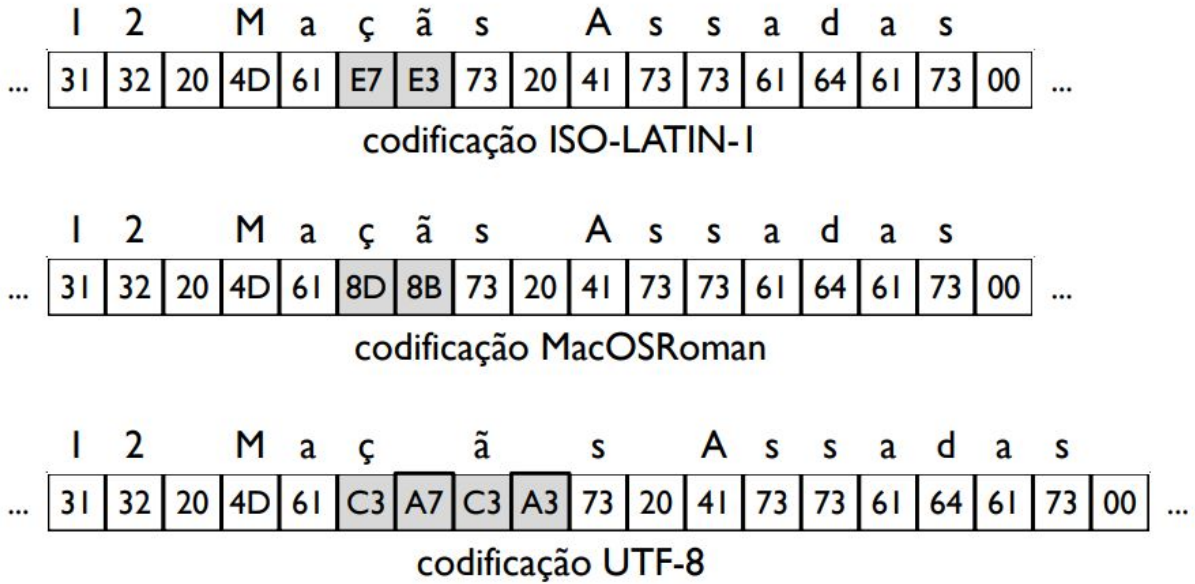
Tabela ASCII

DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII	DEC	HEX	ASCII
0	00	NUL	32	20		64	40	@	96	60	‘
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	“	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	‘	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C	\	124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

Representação de Caracteres

Representação de Cadeias de Caracteres (strings) na memória do computador:

- “Maçãs Assadas”



Organização de Dados na Memória

Organização de dados na memória

Caracteres na memória do computador

A grande maioria das memórias de computadores atuais possuem palavras (unidades de armazenamento endereçáveis) de 1 byte (8 bits).

- No endereço 0 cabe um dado de 1 byte, no endereço 1 cabe um dado de 1 byte e assim por diante.

Organização de dados na memória

Caracteres na memória do computador

A grande maioria das memórias de computadores atuais possuem palavras (unidades de armazenamento endereçáveis) de 1 byte (8 bits).

- No endereço 0 cabe um dado de 1 byte, no endereço 1 cabe um dado de 1 byte e assim por diante.

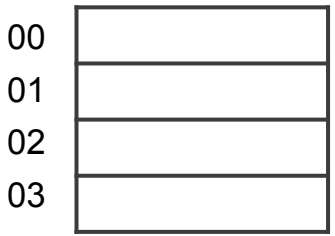
Quando armazenamos números de 7 bits em **1 byte** nós desperdiçamos bits da memória. Por outro lado, esta abordagem facilita a leitura dos dados pois cada palavra de memória possui um **único caractere** e cada caractere está armazenado em uma única palavra de memória.

Organização de dados na memória

Números na memória do computador

- Como fazemos para armazenar um número de 32 bits em uma memória endereçada a byte?

- Exemplo: Número de 32 bits (4 bytes)
  - $1025_{10} = 00000000\ 00000000\ 00000100\ 00000001_2$



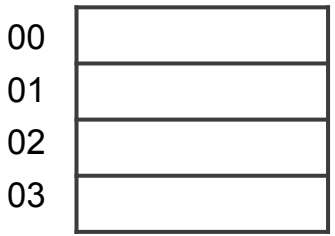
Organização de dados na memória

Números na memória do computador

- Como fazemos para armazenar um número de 32 bits em uma memória endereçada a byte?

  Depende do Endianness

- Exemplo: Número de 32 bits (4 bytes)
  - $1025_{10} = 00000000\ 00000000\ 00000100\ 00000001_2$



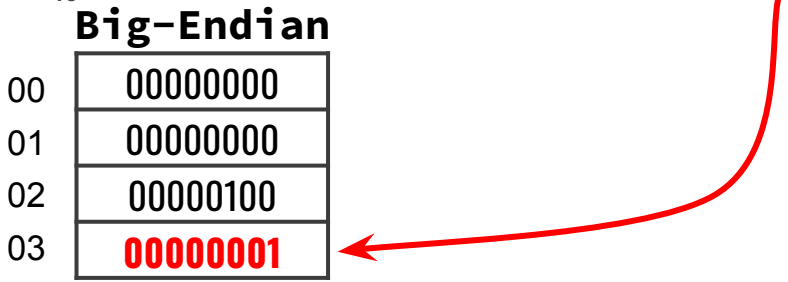
Organização de dados na memória

Números na memória do computador

- Como fazemos para armazenar um número de 32 bits em uma memória endereçada a byte?

  Depende do Endianness

- Exemplo: Número de 32 bits (4 bytes)
  - 1025_{10} = 00000000 00000000 00000100 **00000001**$_2$



Big-Endian: Byte menos significativo é armazenado no **maior** endereço

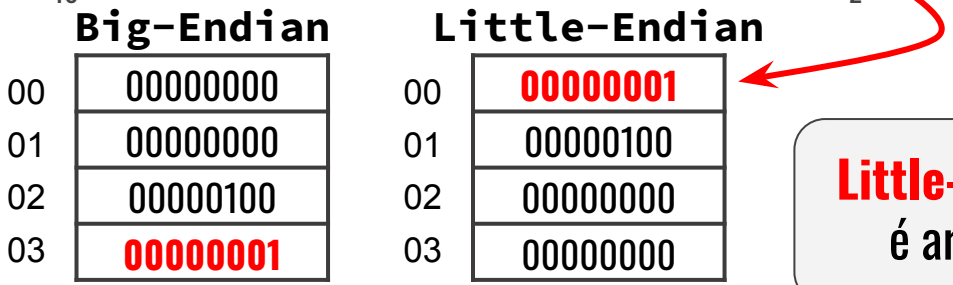
Organização de dados na memória

Números na memória do computador

- Como fazemos para armazenar um número de 32 bits em uma memória endereçada a byte?

  Depende do Endianness

- Exemplo: Número de 32 bits (4 bytes)
  - 1025_{10} = 00000000 00000000 00000100 **00000001**$_2$



Little-Endian: Byte menos significativo é armazenado no **menor** endereço

Organização de dados na memória

Números na memória do computador

- Como fazemos para armazenar um número de 32 bits em uma memória endereçada a byte?

  Depende do Endianness

- Exemplo: Número de 32 bits (4 bytes)
  - 1025_{10} = 00000000 00000000 00000100 **00000001**$_2$

	Big-Endian
00	00000000
01	00000000
02	00000100
03	**00000001**

	Little-Endian
00	**00000001**
01	00000100
02	00000000
03	00000000

Network Endian?

Organização de dados na memória

Vetores na memória

Como fazemos para armazenar um vetor de dados em uma memória endereçada a byte?

Resposta:

- Os elementos do vetor são armazenados de forma consecutiva na memória.

Organização de dados na memória

Vetores na memória

Os elementos de um vetor são armazenados de forma consecutiva na memória.

Supondo que cada elemento ocupe TAM bytes, e o vetor se inicie no endereço BASE, então o i-ésimo elemento é armazenado nos bytes associados aos endereços

BASE + i * TAM a BASE + (i+1) * TAM - 1.

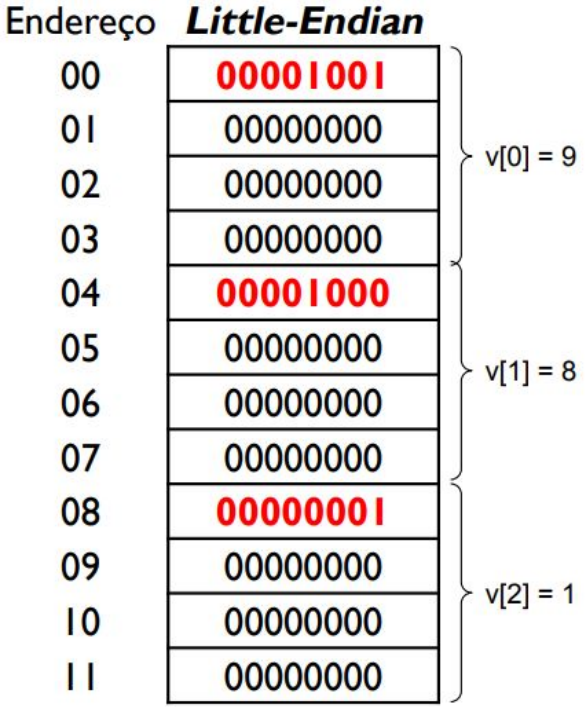
- O primeiro elemento (i=0) será armazenado nos bytes associados aos endereços BASE a BASE+(TAM-1)
- O décimo elemento (i=9) será armazenado nos bytes associados aos endereços 9xBASE a 9xBASE+(TAM-1)

Organização de dados na memória

Vetores na memória

```
int v[3] = {9, 8, 1};
```

- Supondo que o vetor v seja alocado no endereço 0



Organização de dados na memória

Registros na memória

Como fazemos para armazenar registros (structs) em uma memória endereçada a byte?

Resposta:

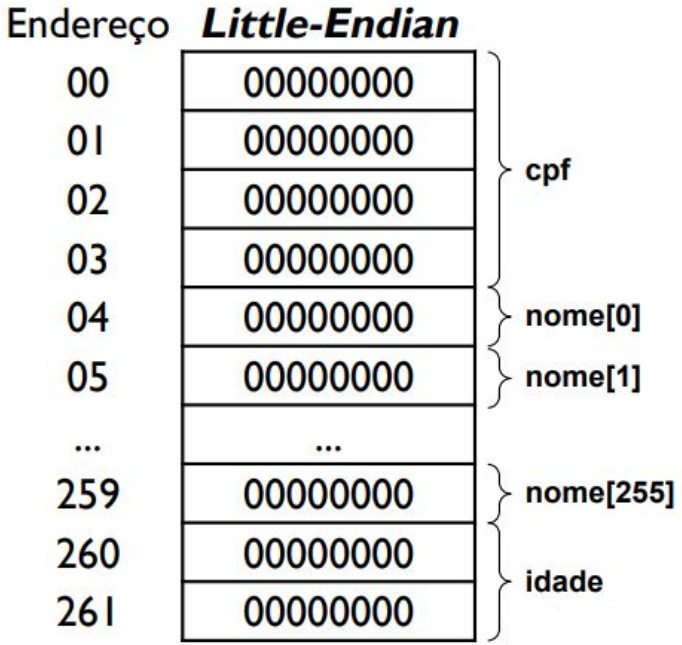
- Os campos dos registros são armazenados de forma consecutiva na memória.

Organização de dados na memória

Registros na memória

```
struct id {
    int cpf;
    char nome[256];
    short idade;
} fulano;
```

- Supondo que o registro fulano seja armazenado no (a partir do) endereço zero de memória.



Organização de dados na memória

Matrizes na memória

Como fazemos para armazenar uma matriz de dados em uma memória endereçada a byte?

Depende da linguagem de programação:

- Em 'C': As linhas da matriz são armazenadas de forma consecutiva na memória (uma linha por vez)
  - Organização conhecida como "row-major order"
- Em Fortran: As colunas da matriz são armazenadas de forma consecutiva na memória
  - Organização conhecida como "column-major order"