

PARTE I: CONCEITOS BÁSICOS DE COMPUTAÇÃO

Introdução à Computação



Sistemas de Numeração

Sistemas de Numeração

□ Principais sistemas de numeração:

□ Decimal

- 0, 1, ..., 9

□ Binário

- 0, 1

□ Octal

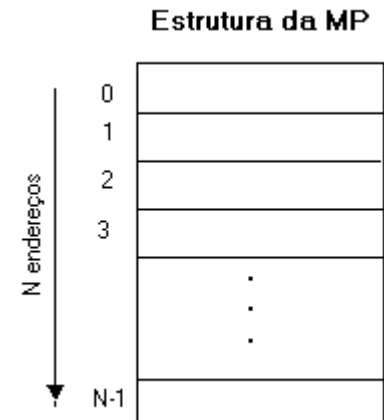
- 0, 1, ..., 7

□ Hexadecimal

- 0, 1, ..., 9, A, B, C, D, E, F

- As letras de A até F equivalem, em decimal, a 10, 11, 12, 13, 14 e 15, respectivamente

- O sistema octal foi muito utilizado em informática como uma alternativa mais compacta ao binário na programação em linguagem de máquina. Hoje, o sistema hexadecimal é mais utilizado como alternativa ao binário. Ambos permitem saber de forma simplificada o valor de cada byte da memória



Sistemas de Numeração

Comparação entre as bases 16, 10, 8 e 2			
0 _{hex}	0 _{dec}	0 _{oct}	0 0 0 0 _{bin}
1 _{hex}	1 _{dec}	1 _{oct}	0 0 0 1 _{bin}
2 _{hex}	2 _{dec}	2 _{oct}	0 0 1 0 _{bin}
3 _{hex}	3 _{dec}	3 _{oct}	0 0 1 1 _{bin}
4 _{hex}	4 _{dec}	4 _{oct}	0 1 0 0 _{bin}
5 _{hex}	5 _{dec}	5 _{oct}	0 1 0 1 _{bin}
6 _{hex}	6 _{dec}	6 _{oct}	0 1 1 0 _{bin}
7 _{hex}	7 _{dec}	7 _{oct}	0 1 1 1 _{bin}
8 _{hex}	8 _{dec}	10 _{oct}	1 0 0 0 _{bin}
9 _{hex}	9 _{dec}	11 _{oct}	1 0 0 1 _{bin}
A _{hex}	10 _{dec}	12 _{oct}	1 0 1 0 _{bin}
B _{hex}	11 _{dec}	13 _{oct}	1 0 1 1 _{bin}
C _{hex}	12 _{dec}	14 _{oct}	1 1 0 0 _{bin}
D _{hex}	13 _{dec}	15 _{oct}	1 1 0 1 _{bin}
E _{hex}	14 _{dec}	16 _{oct}	1 1 1 0 _{bin}
F _{hex}	15 _{dec}	17 _{oct}	1 1 1 1 _{bin}

Conversão Base X – Base 10

Números Inteiros

$$\text{num}_{10} = a_n x^n + a_{n-1} x^{n-1} + \dots + a_0 x^0$$

□ Binário – Decimal: 1011_2

$$1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$$

$$1 * 8 + 0 * 4 + 1 * 2 + 1 * 1 = 11_{10}$$

□ Octal – Decimal: 271_8

$$2 * 8^2 + 7 * 8^1 + 1 * 8^0$$

$$2 * 64 + 7 * 8 + 1 * 1 = 185_{10}$$

□ Hexadecimal – Decimal: $2AB3_{16}$

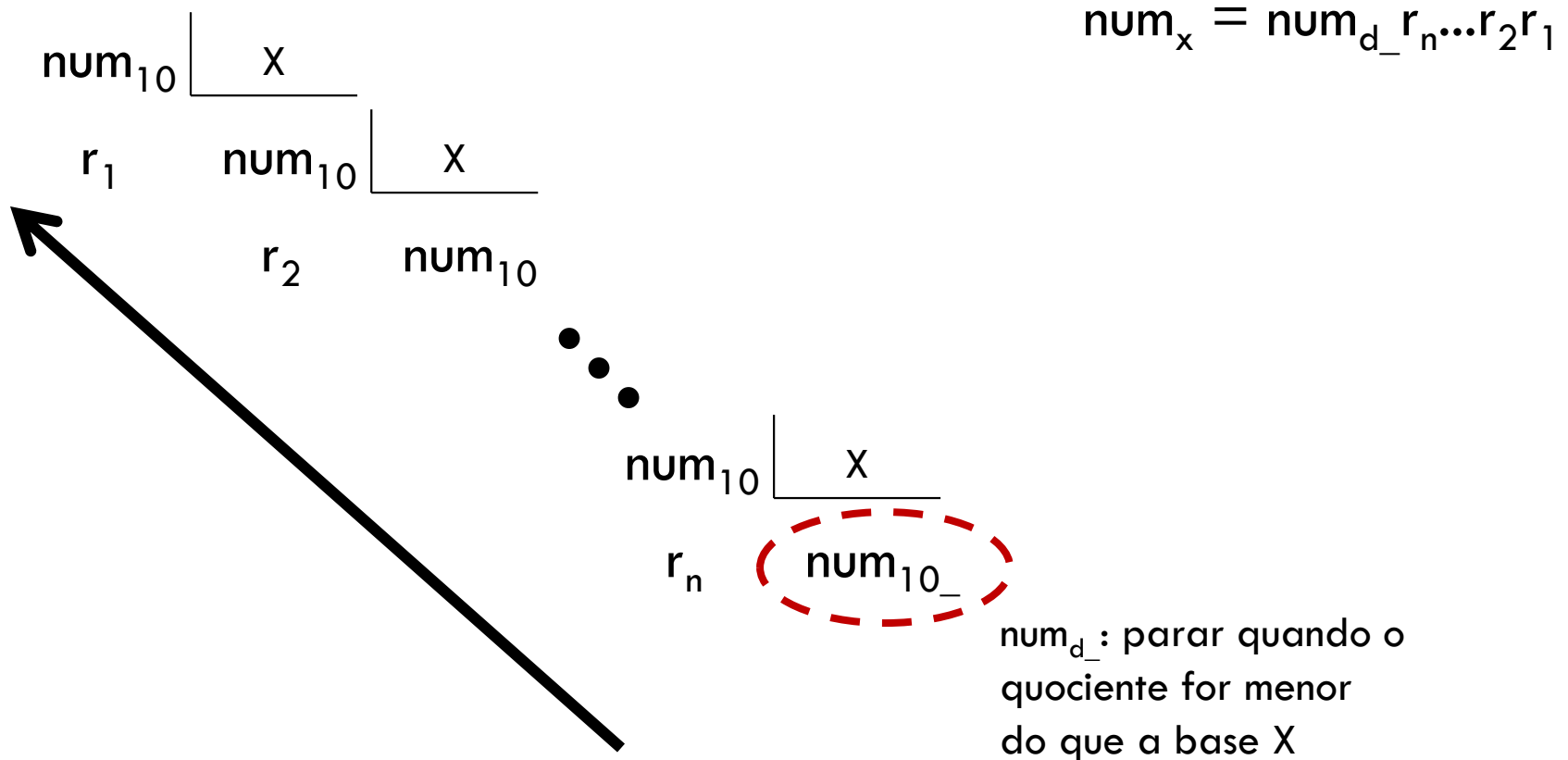
$$2 * 16^3 + A * 16^2 + B * 16^1 + 3 * 16^0$$

$$2 * 16^3 + 10 * 16^2 + 11 * 16^1 + 3 * 16^0$$

$$2 * 4.096 + 10 * 256 + 11 * 16 + 3 * 1 = 10.931_{10}$$

Conversão Base 10 – Base X

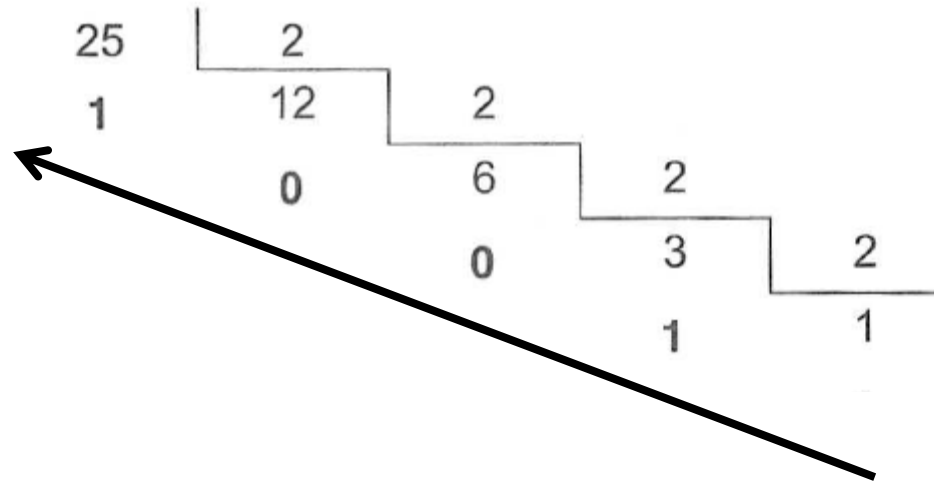
Números Inteiros



Conversão Base 10 – Base X

Números Inteiros

- $(25)_{10}$ para a base 2:

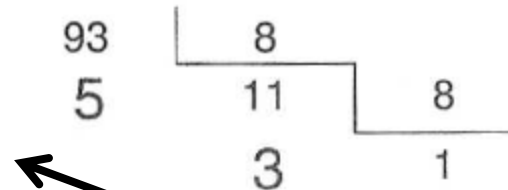


Então, $(25)_{10} = (11001)_2$

Conversão Base 10 – Base X

Números Inteiros

- $(93)_{10}$ para a base 8:

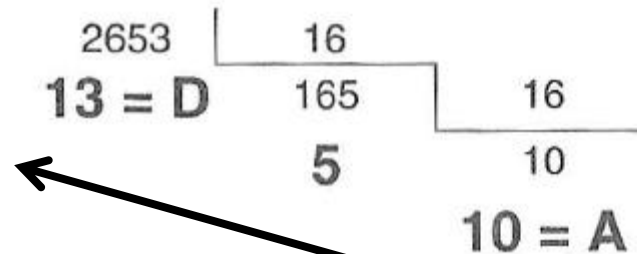


Então, $(93)_{10} = (135)_8$

Conversão Base 10 – Base X

Números Inteiros

- $(2653)_{10}$ para a base 16:



Então, $(2653)_{10} = (A5D)_{16}$

Conversão Base X – Base 10

Números Fracionários

$$\text{num}_{10} = a_n x^n + \dots + a_0 x^0, a_{-1} x^{-1} + \dots + a_{-n} x^{-n}$$

□ Binário – Decimal: $101,101_2$

$$1 * 2^2 + 0 * 2^1 + 1 * 2^0, 1 * 2^{-1} + 0 * 2^{-2} + 1 * 2^{-3}$$

$$1 * 4 + 0 * 2 + 1 * 1, 1 * 0.5 + 0 * 0.25 + 1 * 0.125$$

$$4 + 0 + 1, 0.5 + 0 + 0.125 = 5,625_{10}$$

□ Octal – Decimal: $0,307_8$

$$0 * 8^0, 3 * 8^{-1} + 0 * 8^{-2} + 7 * 8^{-3} = 0,388671875_{10}$$

□ Hexadecimal – Decimal: $0,2B_{16}$

$$0 * 16^0, 2 * 16^{-1} + 11 * 16^{-2} = 0,16796875_{10}$$

Conversão Base 10 – Base X

Números Fracionários

- Parte Inteira: $\text{num}_x = \text{num}_d r_n \dots r_2 r_1$
- Parte Fracionária
 - ▣ Realizada mediante multiplicações sucessivas por X, separando-se as partes inteiras dos resultados, até que se obtenha o valor zero para a parte fracionária dos mesmos
 - ▣ Caso o zero nunca seja alcançado, tem-se uma dízima periódica
 - ▣ As partes inteiras obtidas são os dígitos da conversão

Conversão Base 10 – Base X

Números Fracionários

$(0.5625)_{10}$ para a base 8:

$$8 * 0.5625 = 4 + 0.5$$

$$8 * 0.5 = 4 + 0.0$$

$$(0.5625)_{10} = (0,44)_8$$

$(0.169189453125)_{10}$ para a base 16:

$$16 * 0.169189453125 = 2 + 0.70703125$$

$$16 * 0.70703125 = 11 + 0.3125$$

$$16 * 0.3125 = 5 + 0.0$$

$$(0.169189453125)_{10} = (0,2B5)_{16}$$

Conversão Base 10 – Base X

Números Fracionários

$(0.3)_{10}$ para a base 2:

$$2 * 0.3 = 0 + 0.6$$

$$2 * 0.6 = 1 + 0.2$$

$$2 * 0.2 = 0 + 0.4$$

$$2 * 0.4 = 0 + 0.8$$

$$2 * 0.8 = 1 + 0.6$$

$$2 * 0.6 = 1 + 0.2$$

Dízima periódica

$$(0.3)_{10} = (0,0 \underbrace{1001}_{1001} \underbrace{1001}_{1001} \dots)_2$$

Aritmética Binária

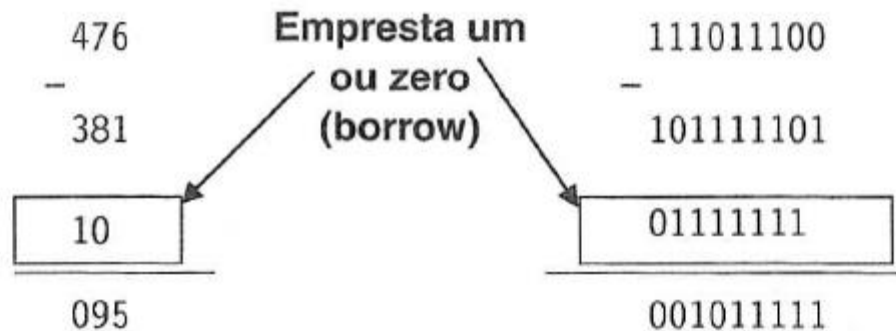
Operandos		a + b		a - b		a * b
a	b	soma	carry	subtração	borrow	produto
0	0	0	0	0	0	0
0	1	1	0	1	1	0
1	0	1	0	1	0	0
1	1	0	1	0	0	1

Operação	Sistema decimal	Sistema binário
Soma	<div> <div>101</div> <div>718</div> <div>+</div> <div>476</div> <hr/> <div>1194</div> </div>	<div> <div>1111011100</div> <div>1011001110</div> <div>+</div> <div>1110111100</div> <hr/> <div>10010101010</div> </div>
	<p>Vai um ou zero (carry)</p>	

Aritmética Binária

Operandos		a + b		a - b		a * b
a	b	soma	carry	subtração	borrow	produto
0	0	0	0	0	0	0
0	1	1	0	1	1	0
1	0	1	0	1	0	0
1	1	0	1	0	0	1

Subtração



Exercícios

- Transformar os seguintes números da base 10 para as bases 2, 8 e 16
 - 184_{10}
 - 1632_{10}
 - $113,5_{10}$
- Transformar os seguintes números das bases 2, 8 e 16 para a base 10
 - $A53_{16}$
 - 756_8
 - $1011100,101_2$



Tipos de Dados

Tipos de Dados



- Tipos de Dados Primitivos
- Demais Tipos
 - ▣ Tipo Cadeia de Caracteres (Strings)
 - ▣ Tipo Array

Tipos de Dados Primitivos

- Tipos de dados não-definidos em termos de outros tipos são chamados tipos de dados primitivos
- Praticamente todas as linguagens de programação oferecem um conjunto de tipos de dados primitivos
- Dividem-se em:
 - ▣ Numéricos
 - Inteiro
 - Ponto-flutuante
 - ▣ Booleano/Lógico
 - ▣ Caractere

Tipos de Dados Primitivos

Representação Inteiro

- Representado em um computador por uma palavra, com um dos bits, tipicamente da extrema esquerda, representando o sinal
- Um inteiro pode ser armazenado de três formas:
 - ▣ Notação de sinal-magnitude
 - ▣ Complemento de um
 - ▣ Complemento de dois
 - Utilizado pela maioria dos computadores
 - Conveniente para a adição e subtração
 - Mesmas regras para soma e subtração

Tipos de Dados Primitivos

Representação Inteiro

□ Complemento de dois

- ▣ O número negativo é formado tomando-se o complemento lógico da versão positiva e adicionando-se um

0 = + / 1 = -

+10 = 0 0001010

-10 = 1º passo (complemento)

1 1110101

2º passo (adição 1)

11110101

1

11110110 (se houver, o último carry é desprezado)

Faixa de Representação

$-2^{N-1} \leq X \leq +2^{N-1} - 1$, N=bits

8 bits: $-128 \leq X \leq +127$

$2^8 = 256$ representações e 256 valores

Tipos de Dados Primitivos

Representação Inteiro

□ Complemento de dois

Vantagens:

1. Mesmas regras para soma e subtração
2. Única representação para o número 0

00000000 (+0)

11111111

1

1 00000000 (carry desprezado) (-0)

Qual o número em decimal?
(considerando 5 bits)

0 1101 = $01101_2 = 13_{10}$

1 1001 = $00111_2 = -7_{10}$

Copia-se da direita para a esquerda até o primeiro um, inclusive, e depois complementa

Tipos de Dados Primitivos


Representação Inteiro

□ Complemento de dois

▣ Aritmética em complemento de dois

- $10 + 5$: o resultado deve ser $+15$;

Ora, observando a Tabela 1.10, $\text{comp-2}(+10) = 01010$ e $\text{comp-2}(+5) = 00101$.
Então,

Carrys: 

Os dois mais à esquerda são iguais

$$\begin{array}{r} 01010 \\ + 00101 \\ \hline 01111 \end{array} = \text{comp-2}(+15) \rightarrow \text{Resultado correto}$$

Em todos os exemplos a seguir, considere uma palavra de 5 bits: $[-16, +15]$

Tipos de Dados Primitivos


Representação Inteiro

□ Complemento de dois

▣ Aritmética em complemento de dois

$$6_{10} = 00110_2$$

- $5 - 6 = 5 + (-6)$: o resultado deve ser -1 ;
Ora, observando a Tabela 1.10, $\text{comp-2}(+5) = 00101$ e $\text{comp-2}(-6) = 11010$.
Então,

Carrys: 

Os dois mais à esquerda são iguais

$$\begin{array}{r} 00101 \\ + 11010 \\ \hline \end{array}$$

$$11111 = \text{comp-2}(-1) \rightarrow \text{Resultado correto}$$

Tipos de Dados Primitivos

Representação Inteiro

□ Complemento de dois

▣ Aritmética em complemento de dois

$$10_{10} = 01010_2$$

- $15 - 10 = 15 + (-10)$: o resultado deve ser +5;
Ora, observando a Tabela 1.10, $\text{comp-2}(+15) = 01111$ e $\text{comp-2}(-10) = 10110$.
Então,

Carrys: \longrightarrow 1 1 1 1 0

Os dois mais
à esquerda
são iguais

$$\begin{array}{r} 01111 \\ + 10110 \\ \hline 100101 \end{array}$$

Como o número de bits do resultado deve ser 5, elimina-se o bit mais à esquerda, resultando

$$00101 = \text{comp-2}(+5) \rightarrow \text{Resultado correto}$$

Tipos de Dados Primitivos

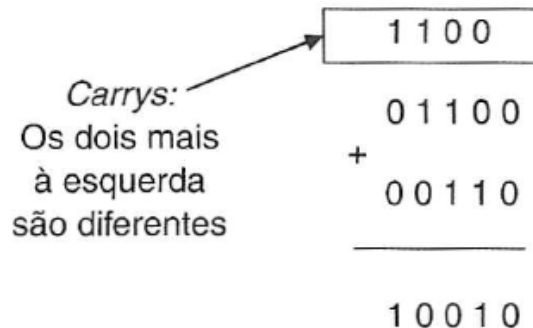
Representação Inteiro

□ Complemento de dois

▣ Aritmética em complemento de dois

- $12 + 6$: o resultado deve ser $+18$, que está fora do intervalo de números representáveis em complemento de 2 de 5 bits, ou seja $[-16, +15]$; não se pode esperar que a operação produza resultados corretos.

Observando a Tabela 1.10, $\text{comp-2}(+12) = 01100$ e $\text{comp-2}(+6) = 00110$. Então,

Carrys: 

$$\begin{array}{r} 01100 \\ + 00110 \\ \hline 10010 \end{array}$$

Os dois mais à esquerda são diferentes

$10010 = \text{comp-2}(-14) \rightarrow$ Resultado incorreto, como esperado

Tipos de Dados Primitivos

Representação Inteiro

□ Complemento de dois

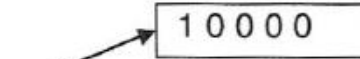
▣ Aritmética em complemento de dois

$$9_{10} = 01001_2$$

$$8_{10} = 01000_2$$

- $-9 - 8 = (-9) + (-8)$: o resultado deve ser -17 , que também está fora do intervalo de números representáveis em complemento de 2 de 5 bits; não se pode esperar que a operação produza resultados corretos.

Observando a Tabela 1.10, $\text{comp-2}(-9) = 10111$ e $\text{comp-2}(-8) = 11000$. Então,

Carrys: 

	1 0 0 0 0
1 0 1 1 1	
+	
1 1 0 0 0	
<hr/>	
1 0 1 1 1 1	

Os dois mais à esquerda são diferentes

Como o número de bits do resultado deve ser 5, elimina-se o bit mais à esquerda, resultando

$$0 1 1 1 1 = \text{comp-2}(+15) \rightarrow \text{Resultado incorreto, como esperado}$$

Tipos de Dados Primitivos

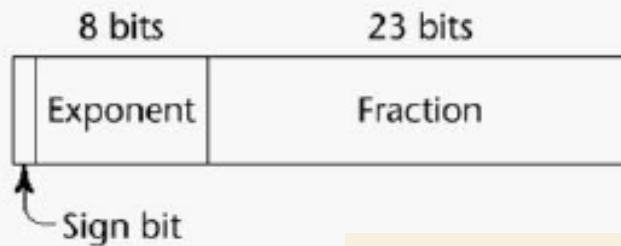
Representação Ponto-flutuante

- ❑ Modelam os números reais, mas são aproximações
- ❑ São representados como frações e expoentes (notação científica)
- ❑ Linguagens para fins científicos
- ❑ Suportam pelo menos dois tipos de ponto-flutuante (float e double)

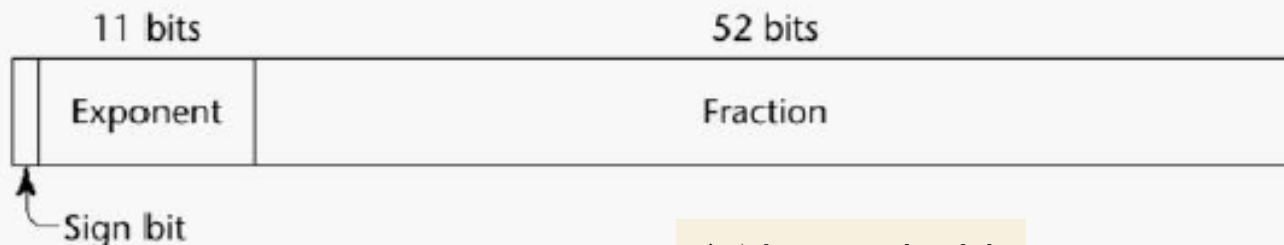
Tipos de Dados Primitivos

Representação Ponto-flutuante

- Formatos de Representação de Pontos Flutuantes:
(a) Precisão Simples, (b) Precisão Estendida



(a) 32 bits = float



(b) 64 bits = double

IEEE Floating-Point Standard 754

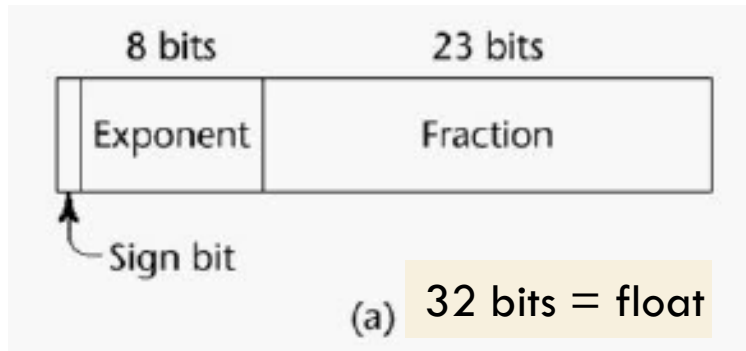
Tipos de Dados Primitivos

Representação Ponto-flutuante

$$57,683 = 0,57683 \times 10^2$$

$\pm 0, M \times B^{\pm e}$ (**B** = base; **e** = expoente; **M** = mantissa)

$B = 10; e = 2; M = 57683$



+	+2	57683
---	----	-------

$$+0,57683 \times 10^{+2}$$

Se a base fosse 10!

Tipos de Dados Primitivos

Representação Ponto-flutuante

407,375

$$(407)_{10} = (110010111)_2$$

$$(0,375)_{10} = (0,011)_2$$

$$(407,375)_{10} = (110010111,011)_2$$

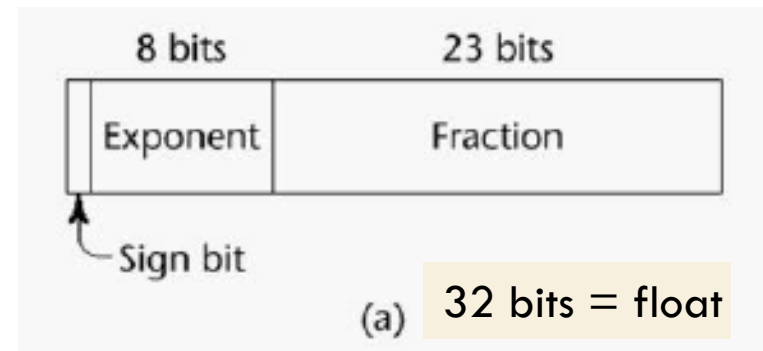
$$(407,375)_{10} = 110010111,011 \times 2^0$$

$$(407,375)_{10} = 0,110010111011 \times 2^{9(1001)}$$

0	00001001	110010111011000000000000
---	----------	--------------------------

A base implícita é 2!

$$\pm 0,M \times B^{\pm e}$$



Tipos de Dados Primitivos

Representação Booleano/Lógico

- Só permite dois valores: verdadeiro ou falso (true or false)
- Pode ser implementado num bit, mas geralmente é implementado num byte
 - ▣ Um bit é difícil de ser acessado com eficiência
 - ▣ Byte = menor célula de memória eficientemente endereçável

Tipos de Dados Primitivos

Representação Caractere

- Armazenados como codificações numéricas
- O código mais usado: ASCII (baseado no inglês)
 - ▣ Codifica 128 caracteres diferentes ($2^7 = 128$; 7 bits + 1 paridade; 0..127)
 - ▣ ASCII estendida: 8 bits = $2^8 = 256$ caracteres (0..255)
- Globalização: Unicode (UTF-8, UTF-16 e UTF-32)
 - ▣ Inclui caracteres da maioria das linguagens naturais
 - ▣ Usado em Java, C#, JavaScript, etc.

<http://www.douglaspasqua.com/slides/unicode.pdf>

Tipos de Dados Primitivos

Representação Caractere

Tabela ASCII

dec.	hex.	octal	ASCII	mnem.	dec.	hex.	octal	ASCII	dec.	hex.	octal	ASCII	dec.	hex.	octal	ASCII
0	00	000	^@	NUL	32	20	040		64	40	100	@	96	60	140	`
1	01	001	^A	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	02	002	^B	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	03	003	^C	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	04	004	^D	EOT	36	24	044	\$	68	44	104	D	100	64	144	d
5	05	005	^E	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	06	006	^F	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	07	007	^G	BELL	39	27	047	'	71	47	107	G	103	67	147	g
8	08	010	^H	BS	40	28	050	(72	48	110	H	104	68	150	h
9	09	011	^I	HTAB	41	29	051)	73	49	111	I	105	69	151	i
10	0A	012	^J	LF	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	0B	013	^K	VTAB	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	0C	014	^L	FF	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	0D	015	^M	CR	45	2D	055	-	77	4D	115	M	109	6D	155	m
14	0E	016	^N	SO	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	0F	017	^O	SI	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	^P	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	^Q	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	^R	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	^S	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	^T	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	^U	NACK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	^V	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	^W	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	^X	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	^Y	EN	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	^Z	SUB	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	^[ESC	59	3B	073	;	91	5B	133	[123	7B	173	{
28	1C	034	^\	FS	60	3C	074	<	92	5C	134	\	124	7C	174	
29	1D	035	^]	GS	61	3D	075	=	93	5D	135]	125	7D	175	}
30	1E	036	^^	RS	62	3E	076	>	94	5E	136	^	126	7E	176	~
31	1F	037	^_	US	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

Exercícios

- Os códigos a seguir são números binários inteiros escritos em complemento de 2, cada um com diferente número de bits. Dizer o valor em decimal com sinal de cada um deles.
 - ▣ 11110
 - ▣ 11
 - ▣ 010101010
- Realizar as operações a seguir em complemento de 2 e dizer se o resultado é correto. Considere uma representação de 8 bits.
 - ▣ $75 + 60 = 135$
 - ▣ $35 + (-70) = -35$
 - ▣ $(-70) + (-70) = -140$
 - ▣ $15 + 35 = 50$



Referências

Referências

- ❑ **Mokarzel, F. C. Introdução à ciência da computação. Elsevier, 2008.**
- ❑ Sebesta, R. W. Conceitos de linguagens de programação. Bookman, 2003.