Aula 6

Professores:

Lúcia M. A. Drummond

Simone de Lima Martins

Conteúdo:

Representação de Dados

- Tipos de dados
- Tipo caractere
- Tipo numérico

Fundação CECIERJ
Consórcio cederj

Representação de dados

- Instruções de máquina realizam operações sobre dados
- Dados e instruções são armazenados internamente como uma seqüência de bits
- As classes de dados abordadas nesta aula:
  - Caracteres
  - Números

Representação de dados

- Existem diversas formas de representação de dados que são utilizadas e compreendidas pelo hardware dos sistemas
- Cada dado é representado internamente por um número de bits
- A quantidade de bits utilizada afeta o tamanho e capacidade de inúmeros componentes do sistema
  - Tamanho do barramento de dados
  - Capacidade dos registradores
  - Capacidade da UAL

Tipos de dados

- O programador deve definir para o sistema como cada dado será manipulado
- Cada dado declarado no programa deve possuir um tipo associado
- Exemplo de declaração de tipo em Pascal:
  - VAR QUANT: INTEGER;
  - VAR QUANT: REAL;
- O tipo associado indica como os bits devem ser organizados para representar o dado e como devem ser realizadas as operações sobre o dado

Tipos de dados

Exemplo

- Somar 103 e 258 representados como valores inteiros

```
  0 1
  1 0 3
+ 2 5 8
-------
  3 6 1
```

Somar Voltar

- Somar 103 e 258 representados em notação científica

$103 = 0{,}103 \times 10^{+3}$ $258 = 0{,}258 \times 10^{+3}$

```
  0  0 1
  0 , 1 0 3
+ 0 , 2 5 8
-----------
  0 , 3 6 1
```

Resultado: $0{,}361 \times 10^{+3}$

Somar Voltar

Tipos de dados

- Tipos primitivos mais utilizados
  - Tipo caractere
  - Tipo numérico
- PASCAL
  - CHAR: caractere
  - INTEGER e REAL: numérico

Tipo caractere

- Como representar todos os caracteres alfabéticos (maiúsculos e minúsculos), algarismos decimais, sinais de pontuação utilizando somente dois símbolos (0 e 1) ?
- Cada caractere é representado por uma seqüência distinta de bits
- Caracteres são codificados para o correspondente grupo de bits de acordo com o código utilizado

Tipo caractere

- Códigos
  - BCD (Binary Coded Decimal)
    - Grupo de 6 bits/caractere
    - Permite a codificação de até 64 caracteres
  - EBCDIC (Extended Binary Coded Decimal Interchange Code)
    - Grupo de 8 bits/caractere
    - Permite a codificação de até 256 caracteres
  - ASCII (American Standard Code for Information Exchange)
    - Grupo de 7 bits/caractere mais um bit de paridade
    - Permite a codificação de até 128 caracteres
    - Versões estendidas utilizam 8 bits
  - UNICODE
    - Grupo de 16 bits/caractere
    - Permite a codificação de até 65.536 caracteres

Utilizando a tabela ASCII

Decimal → / Hex →												
32 20	(space)	48 30	0	64 40	@	80 50	P	96 60	`	112 70	p
33 21	!	49 31	1	65 41	A	81 51	Q	97 61	a	113 71	q
34 22	"	50 32	2	66 42	B	82 52	R	98 62	b	114 72	r
35 23	#	51 33	3	67 43	C	83 53	S	99 63	c	115 73	s
36 24	$	52 34	4	68 44	D	84 54	T	100 64	d	116 74	t
37 25	%	53 35	5	69 45	E	85 55	U	101 65	e	117 75	u
38 26	&	54 36	6	70 46	F	86 56	V	102 66	f	118 76	v
39 27	'	55 37	7	71 47	G	87 57	W	103 67	g	119 77	w
40 28	(	56 38	8	72 48	H	88 58	X	104 68	h	120 78	x
41 29	)	57 39	9	73 49	I	89 59	Y	105 69	i	121 79	y
42 2A	*	58 3A	:	74 4A	J	90 5A	Z	106 6A	j	122 7A	z
43 2B	+	59 3B	;	75 4B	K	91 5B	[	107 6B	k	123 7B	{
44 2C	,	60 3C	<	76 4C	L	92 5C	\	108 6C	l	124 7C	|
45 2D	—	61 3D	=	77 4D	M	93 5D	]	109 6D	m	125 7D	}
46 2E	.	62 3E	>	78 4E	N	94 5E	^	110 6E	n	126 7E	~
47 2F	/	63 3F	?	79 4F	O	95 5F	_	111 6F	o	127 7F	DEL

VAR N: INTEGER;

56 41 52 20 4E 3A 49 4E 54 45 47 45 52 3B

N:= 5;

4E 3A 3D 35 3B

Fundação CECIERJ
Consórcio cederj

Tipo numérico

- Forma mais eficiente de representar números é utilizar representação binária
- Deve-se levar em consideração:
  - A representação do sinal de um número
  - A representação da vírgula (ou ponto) que separa a parte inteira da parte fracionária de um número não-inteiro
  - A quantidade limite de algarismos possível de ser processada pela UAL de um processador

Representação em ponto fixo

- Consiste na determinação de uma posição fixa para a vírgula (ou ponto)
- Todos os dados representados em ponto fixo possuem a mesma quantidade de algarismos inteiros e fracionários
  - 1101,101 1110,001 0011,110
- As posições mais adotadas para a vírgula são:
  - Na extremidade esquerda do número (número é totalmente fracionário)
  - Na extremidade direita do número (número é inteiro)

Representação em ponto fixo

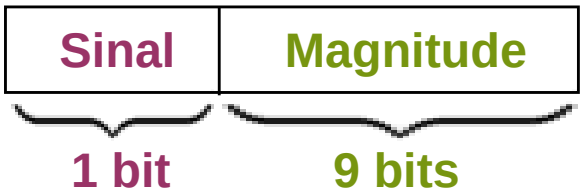
- Na maioria dos processadores atuais, esta representação é utilizada para representar números inteiros
- Como representar números inteiros sem sinal ?
  - Converte para a base 2
  - Considere que podemos utilizar 5 bits para representar inteiros sem sinal
  - A representação do número $(24)_{10}$ será 11000

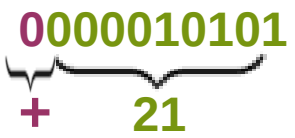
Representação em ponto fixo

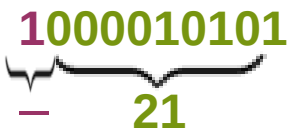
- Como representar números inteiros com sinal ?
  - Sinal e magnitude
  - Complemento a 2

Representação em ponto fixo

- Sinal e magnitude
  - A representação em sinal magnitude de um número com n bits é obtida utilizando-se o bit mais à esquerda para indicar o sinal e os n-1 bits para indicar a magnitude do número
    - **Bit de sinal 0 indica número positivo e 1, negativo**

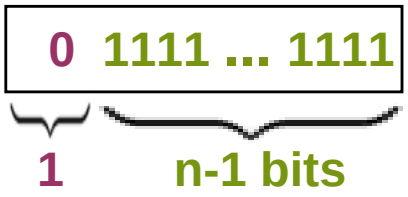






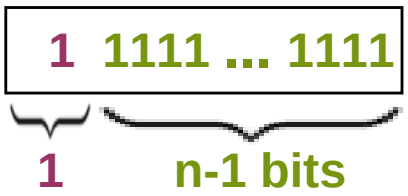
Representação em ponto fixo

- O maior número que pode ser representado em sinal e magnitude utilizando-se n bits



$+(2^{n-1}- 1)$

- O menor número que pode ser representado em sinal e magnitude utilizando-se n bits



$-(2^{n-1}- 1)$

- Faixa de valores - $(2^{n-1}- 1)$ a $+ (2^{n-1}- 1)$

Representação em ponto fixo

- Características da representação sinal e magnitude
  - Duas representações para o número 0
    - +0 (00000...000) e -0 (10000...000)
    - Mais complicado de testar se um valor é igual a 0
  - A mesma quantidade de números positivos e negativos é representada

Representação em ponto fixo

- Aritmética em sinal e magnitude
  - Algoritmo de soma:
    - Verificam-se os sinais dos números e efetua-se uma comparação entre eles
    - Se ambos possuem o mesmo sinal, somam-se as magnitudes e o sinal do resultado é o mesmo das parcelas
    - Se os números possuem sinais diferentes:
      - identifica-se a maior das magnitudes e registra-se o seu sinal
      - subtrai-se a magnitude menor da maior
      - o sinal do resultado é igual ao sinal de maior magnitude

Representação em ponto fixo

- Soma utilizando-se representação sinal e magnitude e 6 bits

		1 1 0
	+1 0	0 0 1 0 1 0
+	+1 5	0 0 1 1 1 1
	+2 5	0 1 0 0 1

Somar Voltar

	+1 5	0 0 1 1 1 1
+	- 4	1 0 0 1 0 0
	+1 1	0 0 1 0 1 1

Somar Voltar

Fundação CECIERJ Consórcio cederj

Representação em ponto fixo

- Aritmética em sinal e magnitude
  - Algoritmo de subtração:
    - **Troca-se o sinal do subtraendo**
    - **Executa algoritmo de soma**
    - **Exemplo: -18 - (+12)= -18 + (-12)**

		0 0 0 0
	- 1 8	1 1 0 0 1 0
+	- 1 2	1 0 1 1 0 0
	- 3 0	1 1 1 1 1 0

Somar Voltar

Representação em ponto fixo

- Complemento a 2
  - Números positivos são representados como na representação em sinal e magnitude
  - Números negativos
    - Invertem-se os bits da representação positiva
    - Soma-se 1
  - Números em complemento a 2 com 8 bits

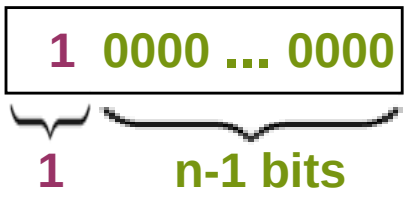
+3 = 00000011
+2 = 00000010
+1 = 00000001
0 = 00000000
-1 = inv(00000001) + 1 = 11111110 + 1 = 11111111
-2 = 11111110
-3 = 11111101

Representação em ponto fixo

- Inteiro com sinal codificado por um vetor com *w* bits $[x_{w-1}, x_{w-2}, ..., x_0]$ utilizando representação complemento a 2
  - $N = -x_{w-1}\, 2^{w-1} + \sum_{i=0}^{w-2} x_i \times 2^i$
  - $0101 = -0 \times 2^3 + 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 = +5$
  - $1101 = -1 \times 2^3 + 1 \times 2^0 + 0 \times 2^1 + 1 \times 2^2 = -3$
- Bit mais significativo é o bit de sinal: 1 para negativos, 0 para positivos

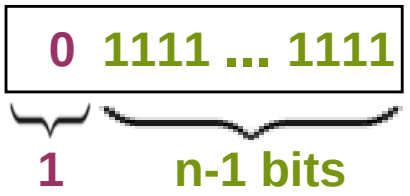
Representação em ponto fixo

- Intervalo de valores a ser representado em complemento a 2:
  - Menor número



$-(2^{n-1})$

  - Maior número



$+(2^{n-1}- 1)$

  - Faixa de valores: $-(2^{n-1})$ a $+(2^{n-1}- 1)$

Representação em ponto fixo

- Vantagens da utilização de representação complemento a 2:
  - uma única representação para o zero
  - necessita apenas um circuito somador para operações de soma e subtração
  - operações simples para encontrar a representação de números negativos (inverte e soma 1)

- Assimetria na quantidade de números representados
  - -2^{n-1} a $+(2^{n-1}-1)$

Representação em ponto fixo

- Extensão de sinal em complemento a 2
  - Dado um inteiro *x* representado com sinal utilizando-se *w* bits, deseja-se representá-lo com *k+w* bits
  - Faça *k* cópias do bit de sinal
  - $X' = \underbrace{x_{w-1}, \ldots, x_{w-1}}_{\text{K cópias de BMS}}, x_{w-1}, x_{w-2}, \ldots, x_0$
  - 0011 ⟶ 00000011
  - 1100 ⟶ 11111100

Representação em ponto fixo

- Adição em complemento a 2
  - Soma em binário normal
  - Monitora o bit de sinal para saber se houve estouro (overflow)
- Subtração em complemento a 2
  - Obtém a representação em complemento a 2 do subtraendo e soma ao minuendo
    - a - b = a + (-b)
  - Somente necessita de circuitos de soma e de inversão
  - Monitora o bit de sinal para saber se houve estouro (overflow)

Representação em ponto fixo

- Adição em complemento a 2
  - +11 + (+20)

		0 0 0 0 0
	+1 1	0 0 1 0 1 1
+	+2 0	0 1 0 1 0 0
	+31	0 1 1 1 1 1

Somar Voltar

  - -25 + (-12)

		0 0 1 0 0
	-2 5	1 0 0 1 1 1
+	-1 2	1 1 0 1 0 0
	-37	0 1 1 0 1 1

Somar Voltar

Representação em ponto fixo

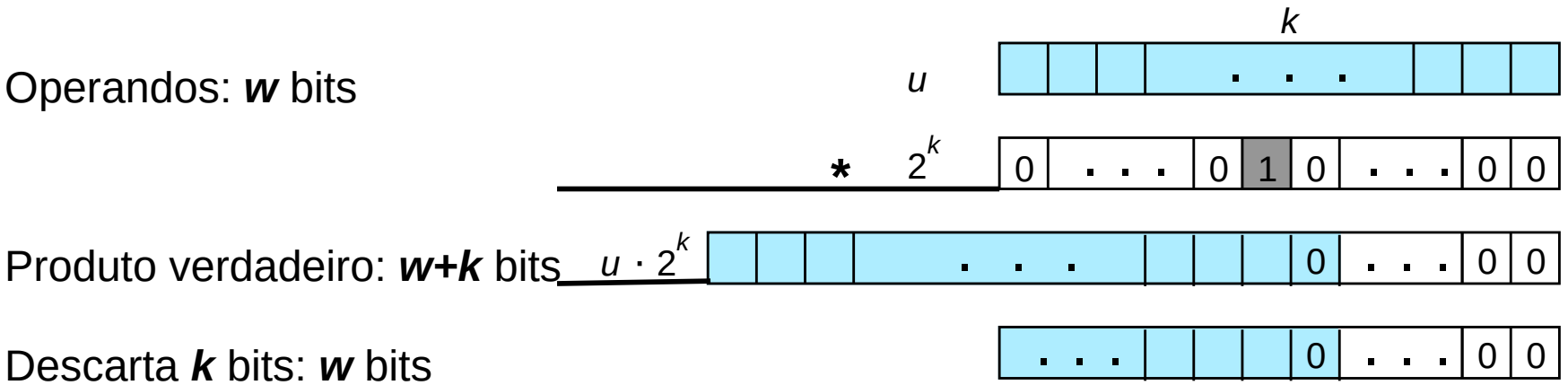
- Subtração em complemento a 2
  - +11 - (+21) = +11 + (-21)

				0 1 0 1 1
+1 1	0 0 1 0 1 1			0 0 1 0 1 1
+ - 2 1	0 1 0 1 0 1	1 0 1 0 1 0 + 1	=	1 0 1 0 1 1
- 1 0				1 1 0 1 1 0

Somar Voltar

Representação em ponto fixo

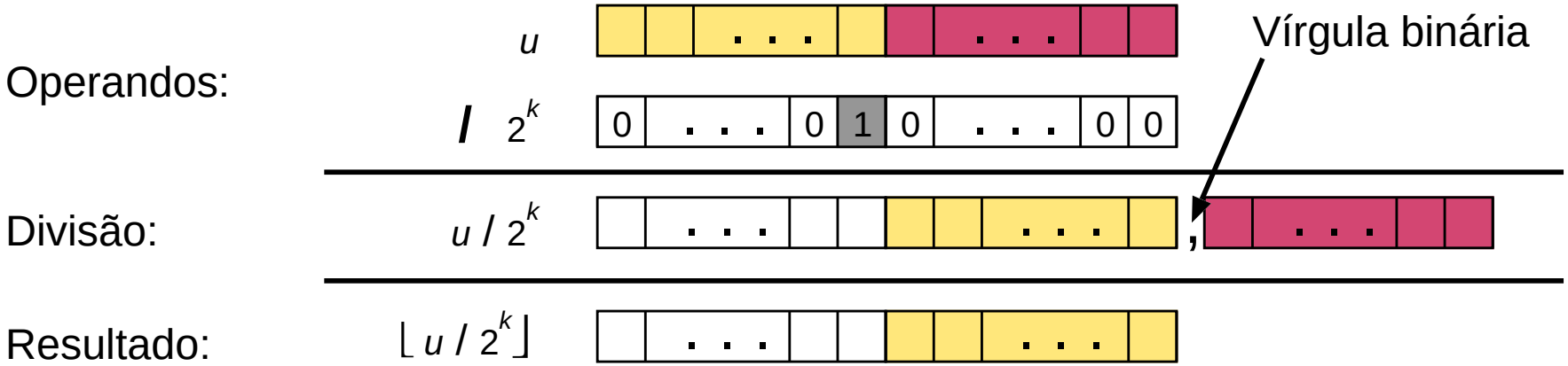
- Operação de multiplicação por potência de 2
  - u << k fornece $u * 2^k$
  - Para números com e sem sinal



  - Exemplos
    - u << 3 == u * 8
    - u << 5 - u << 3 == u * 24
    - Maioria das máquinas desloca e soma mais rápido que multiplica
      - Compilador gera o código automaticamente

Representação em ponto fixo

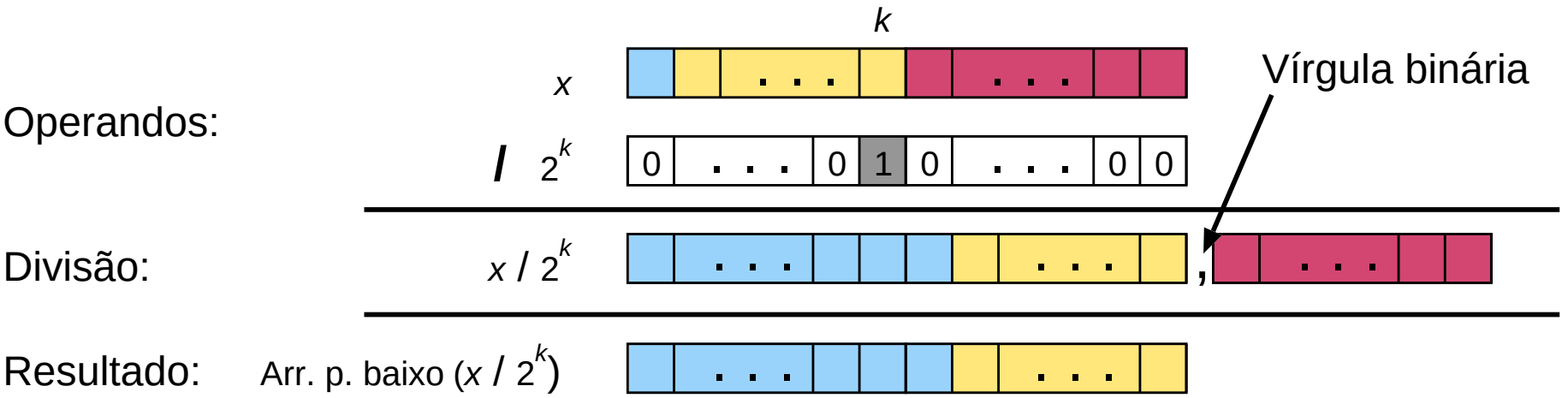
- Quociente da divisão de número sem sinal por potência de 2
  - u >> k fornece $\lfloor u / 2^k \rfloor$
  - Utiliza deslocamento lógico



	Divisão	Calculada	Hexa	Binário
x	15213	15213	3B 6D	00111011 01101101
x >> 1	7606,5	7606	1D B6	**0**0011101 10110110
x >> 4	950,8125	950	03 B6	**0000**0011 10110110
x >> 8	59,4257813	59	00 3B	**00000000** 00111011

Representação em ponto fixo

- Quociente da divisão de número com sinal por potência de 2
  - x >> k fornece $\lfloor x / 2^k \rfloor$
  - Utiliza deslocamento aritmético
  - Arredonda para direção errada quando u < 0



Representação em ponto fixo

- Quociente da divisão de número com sinal por potência de 2

	Divisão	Calculado	Hexa	Binário
y	-15213	-15213	C4 93	11000100 10010011
y >> 1	-7606,5	-7607	E2 49	**1**1100010 01001001
y >> 4	-950,8125	-951	FC 49	**1111**1100 01001001
y >> 8	-59,4257813	-60	FF C4	**11111111** 11000100

Representação em ponto flutuante

- Como representar números fracionários ?
  - Vírgula poderia ficar fixa em algum lugar
  - Limite no número de bits para parte inteira e fracionária

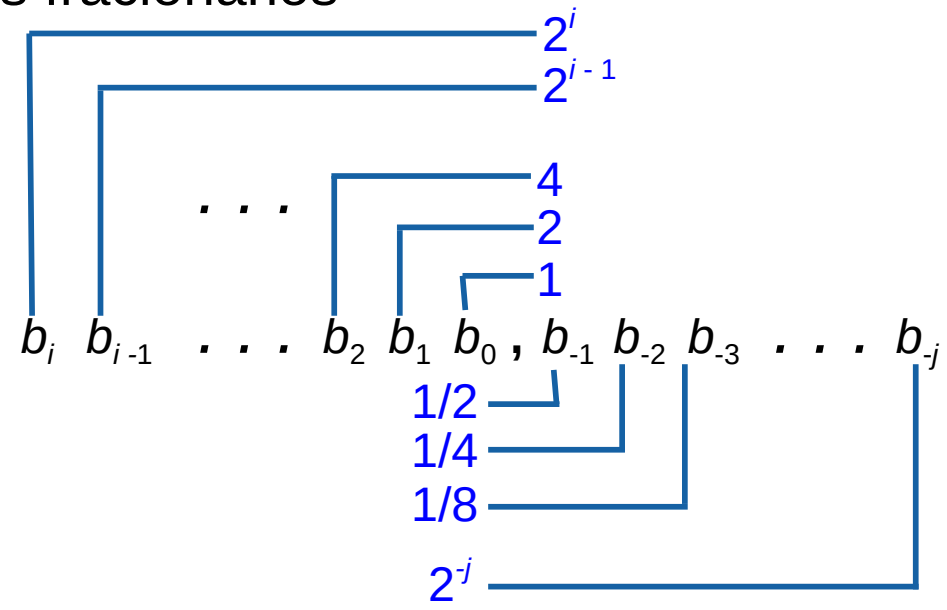
- Notação científica normalizada na base 10
  - $1.000.000.000 = 1,0 \times 10^{9}$
  - $0,000000000017 = 1,7 \times 10^{-11}$

Representação em ponto flutuante

- Representação IEEE Standard 754
  - Estabelecido em 1985 como padrão uniforme para aritmética em ponto flutuante
  - A maioria das CPUs suporta este padrão
- Foi projetado para criar um padrão com facilidades para operações numéricas

Representação em ponto flutuante

- Números binários fracionários



- Representação
  - **Bits à direita da "vírgula binária" representam potências fracionárias de 2**
  - **Representa o número:** $\sum_{k=-j}^{i} b_k \cdot 2^k$

Representação em ponto flutuante

- Representação IEEE 754
  - Forma numérica
    - $-1^{s}\ M\ \ 2^{E}$
      - Bit de sinal *s* determina se o número é negativo ou positivo
      - O significando *M* é um valor fracionário na faixa [1.0,2.0).
      - O expoente *E* é um número inteiro com sinal
- Codificação

  - MSB é o bit de sinal
  - O campo exp codifica *E*
  - O campo frac codifica *M*

Representação em ponto flutuante

- Representação IEEE 754
  - Tamanhos
    - **Precisão simples: 8 bits para exp, 23 bits para frac**
      - 32 bits no total
    - **Precisão dupla: 11 bits para exp, 52 bits para frac**
      - 64 bits no total
- Precisão estendida: 15 bits para exp, 63 bits para frac
  - Somente em máquinas compatíveis com Intel
  - 80 bits no total
    - » 1 bit não utilizado

Representação em ponto flutuante

- Valores numéricos normalizados
  - exp ≠ 000...0 e exp ≠ 111...1
  - Representação em excesso de *n* para expoente

    $E = Exp - Bias$

    - *Exp* : inteiro sem sinal representado por exp
    - *Bias* : Valor de *n*
      - » Precisão simples: 127 (*Exp*: 1...254, *E*: -126...127)
      - » Precisão dupla: 1023 (*Exp*: 1...2046, *E*: -1022...1023)
      - » Em geral: *Bias* = $2^{e-1} - 1$, onde e é o número de bits do expoente
  - Significando codificado com 1 implícito antes da vírgula

    $M = 1,xxx...x_2$

    - xxx...x: bits do campo frac
    - Mínimo quando 000...0 (*M* = 1,0)
    - Máximo quando 111...1 (*M* = 2,0 - ε)

Representação em ponto flutuante

- Valor

  Float F = 15213.0;

  $15213_{10} = 11101101101101_2 = 1{,}1101101101101_2 \times 2^{13}$

- Significando

  M = $1{,}\underline{1101101101101}_2$

  frac = $\underline{1101101101101}0000000000_2$

- Expoente

  E = 13

  Bias = 127

  Exp = 140 =10001100_2

Representação em ponto flutuante

Hexa: 4 6 6 D B 4 0 0

Binário: 0100 0110 0110 1101 1011 0100 0000 0000

Representação em ponto flutuante

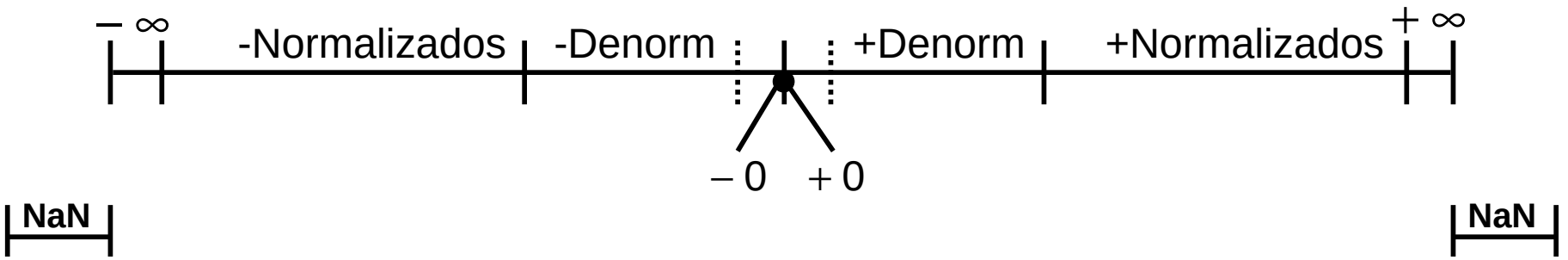
- Valores denormalizados
  - exp = 000...0
  - Valor
    - Valor do expoente ***E*** = *-Bias* + 1
    - Valor do significando ***M*** = 0, xxx...x_2
      - xxx...x: bits do campo frac
  - Casos
    - exp = 000...0, frac = 000...0
      - Representa o valor 0
      - Existem duas representações: +0 and -0
    - exp = 000...0, frac ≠ 000...0
      - Números muito perto de 0,0

Representação em ponto flutuante

- Valores especiais
  - exp = 111...1
  - Casos
    - exp = 111...1, frac = 000...0
      - Representa o valor ∞ (infinito)
      - Operações em que ocorrem overflows
      - Positivo e negativo
      - Ex., 1,0/0,0 = –1,0/–0,0 = +∞, 1,0/–0,0 = – ∞
    - exp = 111...1, frac ≠ 000...0
      - Not-a-Number (NaN)
      - Representa o caso quando não se pode determinar um valor numérico
      - Ex., sqrt(-1), ∞ – ∞

Representação em ponto flutuante

- Visualização da codificação de números reais em ponto flutuante



Representação em ponto flutuante

- Exemplos de valores

Descrição	exp	frac	Valor numérico
Zero	00...00	00...00	0,0
Menor Pos. Denorm.	00...00	00...01	$2^{-\{23,52\}} \times 2^{-\{126,1022\}}$
Simples	$\approx 1,4 \text{ X } 10^{-45}$		
Dupla	$\approx 4,9 \text{ X } 10^{-324}$		
Maior Pos. Denorm.	00...00	11...11	$(1,0 - \varepsilon) \times 2^{-\{126,1022\}}$
Simples	$\approx 1,18 \text{ X } 10^{-38}$		
Dupla	$\approx 2,2 \text{ X } 10^{-308}$		
Menor Pos. Norm.	00...01	00...00	$1,0 \times 2^{-\{126,1022\}}$
Maior que o maior denormalizado			
Um	01...11	00...00	1,0
Maior Pos. Norm.	11...10	11...11	$(2,0 - \varepsilon) \times 2^{\{127,1023\}}$
Simples	$\approx 3,4 \times 10^{38}$		
Dupla	$\approx 1,8 \times 10^{308}$		

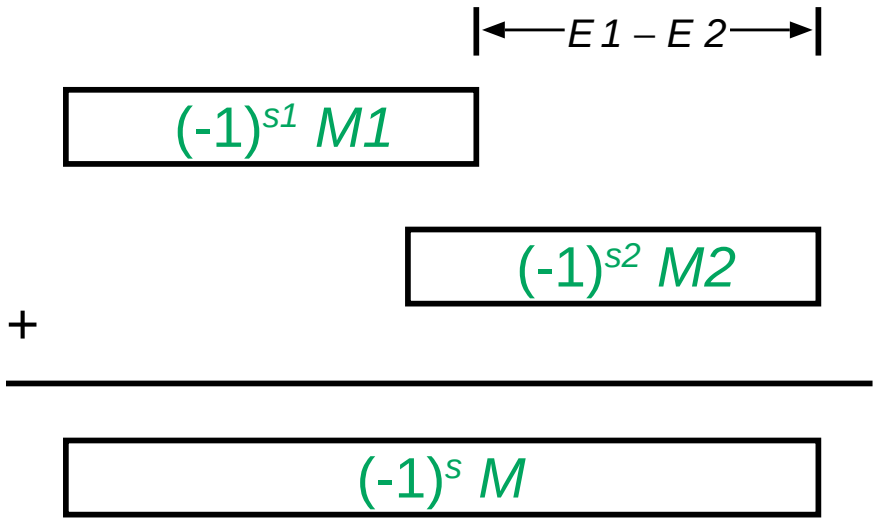
Representação em ponto flutuante

- Soma em ponto flutuante

  $(-1)^{s1}\ M1\ \ 2^{E1}$

  $(-1)^{s2}\ M2\ \ 2^{E2}$

  – Assuma ***E1*** > ***E2***



- Resultado exato

  $(-1)^{s}\ M\ \ 2^{E}$

  – Sinal ***s***, significando ***M***:

    - Resultado de alinhamento com sinal & soma

  – Expoente ***E***: ***E1***

- Ajuste

  – Se ***M*** ≥ ***2***, desloca ***M*** para a direita, incrementa ***E***

  – Se ***M*** < ***1***, desloca ***M*** para a esquerda ***k*** posições, decrementa ***E*** de ***k***

  – Overflow se ***E*** fora da faixa

  – Arredonda ***M*** para ajuste ao campo frac

Exercícios

- Capítulo 7 do livro texto
  - 4, 5, 9, 12, 13, 15 e 27