

MC542

Organização de Computadores Teoria e Prática

2007 - 2011

Prof. Paulo Cesar Centoducatte

ducatte@ic.unicamp.br

www.ic.unicamp.br/~ducatte

MC542

Organização de Computadores Teoria e Prática

Referências:

- David M. Harris & Sarah L. Harris, Digital Design and Computer Architecture - **DDCA**
- Stephen Brown & Zvonko Vranesic, Fundamentals of Digital Logic (with VHDL design) - **FDL**
- David A. Patterson & John L. Hennessy, Computer Organization and Design (the hardware/software interface) - **COD**

MC542

Introdução

Abstração, Sistemas Numéricos

“DDCA” - (Capítulo 1)

“FDL” - (Capítulo 5)

Abstração, Sistemas Numéricos, Tecnologia

Sumário

- **Objetivos**
- **Abstração**
 - Abstração Digital
- **Binário**
- **Representação de Números (Revisão)**
 - Posicional
 - Inteiros sem Sinal
 - » Decimal
 - » Binário
 - » Hexadecimal e Octal
 - » Conversão entre bases
 - » Valores e Intervalos
 - Bits, Bytes, Nibbles...
 - Soma de Números Inteiros e Overflow

Abstração, Sistemas Numéricos, Tecnologia

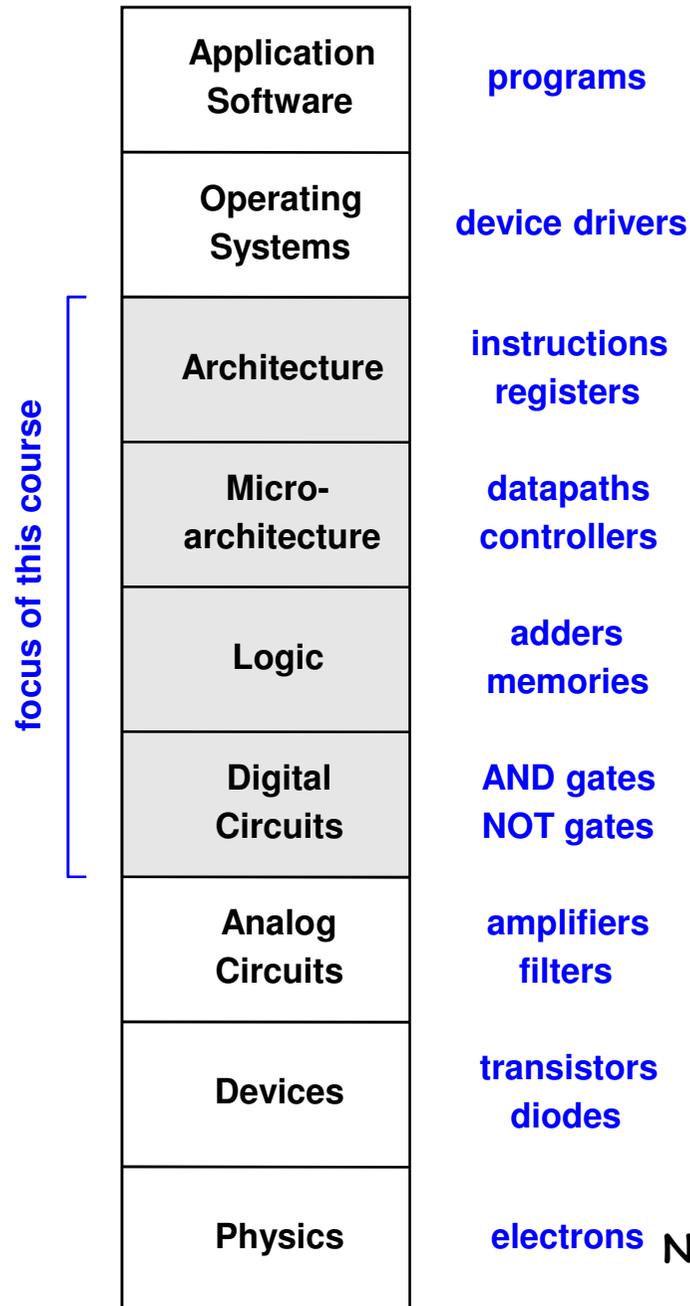
Sumário

- **Representação de Números (cont.)**
 - **Representação de Números Negativos**
 - » Sinal e Magnitude
 - » Complemento de 1
 - » Complemento de 2
 - **Adição e Subtração**
 - » Sinal e Magnitude
 - » Complemento de 1
 - » Complemento de 2
 - » Overflow
- **Representações de Números Reais**
 - Fixo
 - Ponto-Flutuante
- **BCD**

Objetivos

- **Considerações:**
 - Familiaridade com eletricidade básica
 - Experiência com programação
- **Objetivos do Curso**
 - Aprender os princípios de projetos digitais
 - Aprender como um computador funciona
 - Projetar um microprocessador

Abstração



Um sistema pode ser visto com níveis de detalhes diferentes.

Abstração é usada para esconder detalhes quando eles não são importantes.

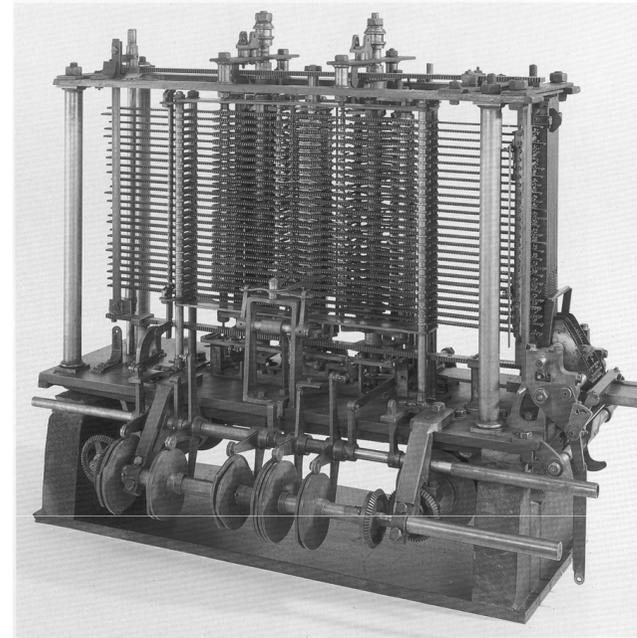
Níveis de abstração para um sistema computacional

Abstração Digital

- A maioria das variáveis físicas são contínuas, exemplo:
 - Tensão em um fio
 - Freqüência
 - Posição de um objeto em um plano
- Usando abstração digital, não se considera todos os valores possíveis e sim somente um conjunto discreto de valores

Analytical Engine

- Projetado por Charles Babbage (1834 - 1871)
- Considerado como o primeiro computador digital
- Representava valores discretos (0-9)



Binário

- Considera somente dois valores discretos:
 - 1's e 0's
 - 1: TRUE, HIGH
 - 0: FALSE, LOW
- "1" e "0" podem ser representados por um nível de voltagem específica ou outra grandeza física
- Circuitos digitais, em geral usam um nível de voltagem específico para representar o "1" e o "0"
- *Bit: Binary digit*

Representação de Números Posicional

- Decimal - Inteiros sem Sinal

$$D = d_{n-1} d_{n-2} \dots d_1 d_0$$

$$V(D) = d_{n-1} \times 10^{n-1} + d_{n-2} \times 10^{n-2} + \dots + d_1 \times 10^1 + d_0 \times 10^0$$

$$\begin{array}{cccc} 1000\text{'s} & 100\text{'s} & 10\text{'s} & 1\text{'s} \\ 5 & 3 & 7 & 4 \\ \hline 5374_{10} & = 5 \times 10^3 & + 3 \times 10^2 & + 7 \times 10^1 & + 4 \times 10^0 \\ & \text{milhares} & \text{centenas} & \text{dezenas} & \text{unidades} \end{array}$$

Representação de Números Posicional

- Binário - Inteiros sem sinal

$$B = b_{n-1} b_{n-2} \dots b_1 b_0$$

$$V(B) = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_1 \times 2^1 + b_0 \times 2^0$$

$$= \sum_{i=0}^{n-1} b_i \times 2^i$$

8's 4's 2's 1's

$$1\ 1\ 0\ 1_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

Representação Posicional Conversão entre Decimal e Binário

$$V(B) = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_1 \times 2^1 + b_0 \times 2^0$$

$$V(B) = b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \dots + b_1 \times 2^1 + b_0$$

$$\frac{V(B)}{2} = b_{n-1} \times 2^{n-2} + b_{n-2} \times 2^{n-3} + \dots + b_1 + \frac{b_0}{2}$$

Conversão de Decimal para Binário: Divisão sucessiva por 2

Exemplo

Convert $(857)_{10}$

			Remainder			
857	÷	2	=	428	1	LSB
428	÷	2	=	214	0	
214	÷	2	=	107	0	
107	÷	2	=	53	1	
53	÷	2	=	26	1	
26	÷	2	=	13	0	
13	÷	2	=	6	1	
6	÷	2	=	3	0	
3	÷	2	=	1	1	
1	÷	2	=	0	1	MSB

Result is $(1101011001)_2$

Exercícios

- Converter 10101_2 para decimal

- Converter 47_{10} para binário

Valores e Intervalos

- Considere um número decimal com N -dígitos
 - Representa 10^N possíveis valores
 - O Intervalo é: $[0, 10^N - 1]$
 - Exemplo,
 - » Um número decimal 3-dígitos representa $10^3 = 1000$ valores, com intervalo de $[0, 999]$
- Considere um número binário N -bit
 - Representa 2^N possíveis valores
 - O Intervalo é: $[0, 2^N - 1]$
 - » Exemplo, um número binário 3-bit $2^3 = 8$ valores, com intervalo de $[0, 7]$ (i.e., 000_2 a 111_2)

Números Hexadecimal

Base 16

Dígito Hexa	Equivalente Decimal	Equivalente Binário
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Números Octal

Base 8

Decimal	Binary	Octal	Hexadecimal
00	00000	00	00
01	00001	01	01
02	00010	02	02
03	00011	03	03
04	00100	04	04
05	00101	05	05
06	00110	06	06
07	00111	07	07
08	01000	10	08
09	01001	11	09
10	01010	12	0A
11	01011	13	0B
12	01100	14	0C
13	01101	15	0D
14	01110	16	0E
15	01111	17	0F
16	10000	20	10
17	10001	21	11
18	10010	22	12

Conversão Hexadecimal para Binário

- Converter $4AF_{16}$ ($0x4AF$) para binário

- Converter $0x4AF$ para decimal

Conversão Hexadecimal para Binário

- Converter $4AF_{16}$ (0x4AF) para binário

010010101111_2

- Converter 0x4AF para decimal

$$\begin{aligned} 010010101111_2 &= 1 + 2 + 4 + 8 + 32 + 128 + 1024 \\ &= 1199_{10} \end{aligned}$$

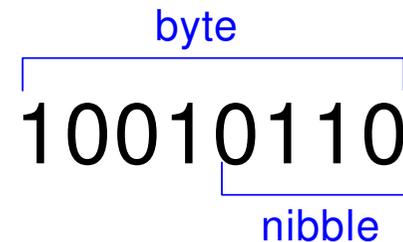
$$\begin{aligned} 0x4AF &= (15 \times 16^0) + (10 \times 16^1) + (4 \times 16^2) \\ &= 1199_{10} \end{aligned}$$

Bits, Bytes, Nibbles...

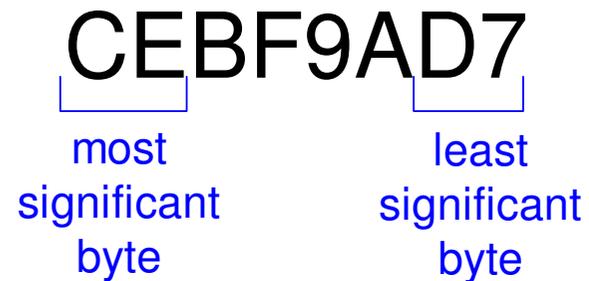
- **Bits**



- **Bytes & Nibbles**



- **Bytes**



Potências de 2

- $2^{10} = 1 \text{ kilo} \approx 1000 \text{ (1024)}$
- $2^{20} = 1 \text{ mega} \approx 1 \text{ milhão(1.048.576)}$
- $2^{30} = 1 \text{ giga} \approx 1 \text{ bilhão (1.073.741.824)}$

Estimando Potência de 2

- Qual o valor de 2^{22} ?

$$2^2 \times 2^{20} = 4 \text{ Mega}$$

- Quantos valores uma variável de 32-bit pode representar?

$$2^2 \times 2^{30} = 4 \text{ Giga}$$

Soma

- Decimal

$$\begin{array}{r} 11 \leftarrow \text{carries} \\ 3734 \\ + 5168 \\ \hline 8902 \end{array}$$

- Binária

$$\begin{array}{r} 11 \leftarrow \text{carries} \\ 1011 \\ + 0011 \\ \hline 1110 \end{array}$$

Soma Binária: Exemplos

- Some os seguintes números:

$$\begin{array}{r} 1001 \\ + 0101 \\ \hline \end{array}$$

$$\begin{array}{r} 1011 \\ + 0110 \\ \hline \end{array}$$

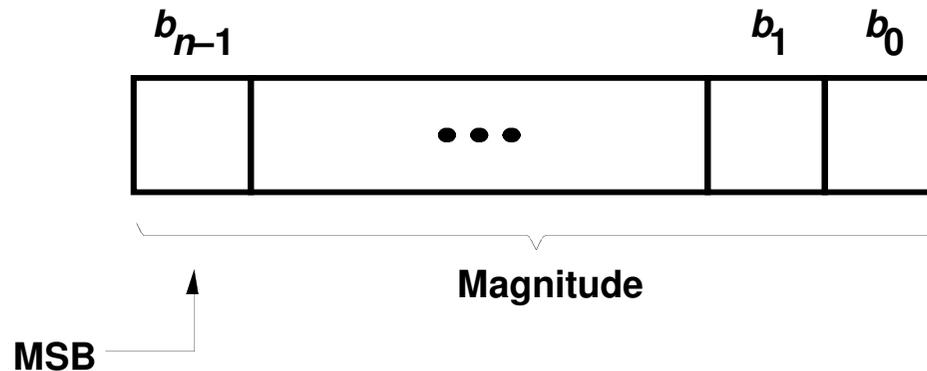
Overflow

- Sistemas Digitais operam com um número fixo de bits
- A Adição tem **overflow** quando o resultado não pode ser representado com o número de bits disponíveis
- Exemplo: somar 13 e 5 usando números de 4-bit

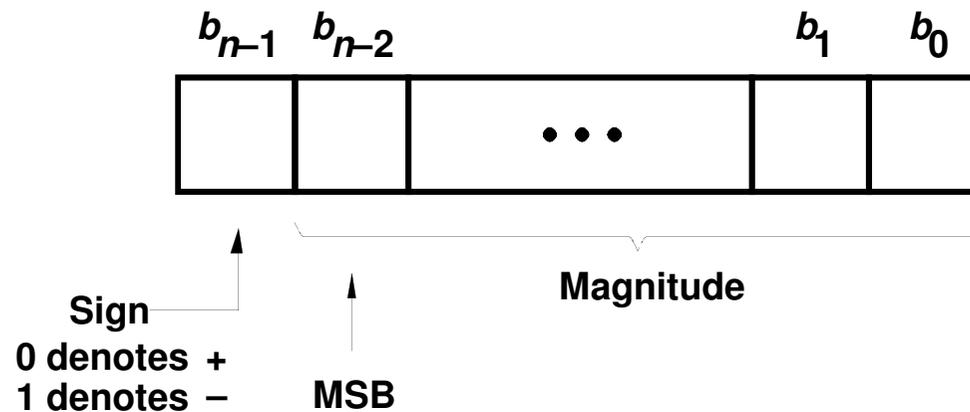
$$\begin{array}{r} 11\ 1 \\ 1101 \\ +\ 0101 \\ \hline 10010 \end{array}$$

Representação de Números Negativos

Sinal e Magnitude



Número sem Sinal



Número com Sinal

Representação de Números Negativos

- **Sinal e Magnitude**

- 1 bit de sinal, N-1 bits de magnitude
- O bit de sinal é o mais significativo (mais a esquerda)
 - » Número negativo: 1
 - » Número positivo: 0

- Exemplo, representação de ± 5 com 4-bit:

$$-5 = 1101_2$$

$$+5 = 0101_2$$

- Intervalo de um número N-bit sinal/magnitude:

$$[-(2^{N-1}-1), 2^{N-1}-1]$$

Representação de Números Negativos

Complemento de 1

Em complemento de "Um" o número negativo K , com n -bits, é obtido subtraindo seu positivo P de $2^n - 1$

$$K = (2^n - 1) - P$$

Exemplo: se $n = 4$
então:

$$K = (2^4 - 1) - P$$

$$K = (16 - 1) - P$$

$$K = (1111)_2 - P$$

$$K = -7 \quad \rightarrow \quad P = 7$$

$$7 = (0111)_2$$

$$-7 = (1111)_2 - (0111)_2$$

$$-7 = (1000)_2$$

Representação de Números Negativos

Complemento de 2

Em complemento de "Dois" o número negativo K , com n -bits, é obtido subtraindo seu positivo P de 2^n

$$K = 2^n - P \quad \longrightarrow \quad K = (2^n - 1) + 1 - P$$

$$K = (2^n - 1) - P + 1$$

Exemplo: se $n = 4$
então:

$$K = 2^4 - P$$

$$K = 16 - P$$

$$K = (10000)_2 - P$$

$$K = -7 \quad \rightarrow \quad P = 7$$

$$7 = (0111)_2$$

$$-7 = (10000)_2 - (0111)_2$$

$$-7 = (1001)_2$$

Representação de Números Negativos

- Complemento de 2
 - Regra Prática

$$K = 2^n - P \quad \longrightarrow \quad K = (2^n - 1) + 1 - P$$
$$K = (2^n - 1) - P + 1$$

$$K = 11\dots 11 - (p_{n-1} \dots p_0) + 1$$

$$K = (\overline{p_{n-1} \dots p_0}) + 1$$

Representação de Números Negativos

- Complemento de 2

- O mesmo que sem sinal porém o **most significant bit (msb)** tem valor -2^{N-1}
- Maior número positivo de 4-bit: 0111_2 (7_{10})
- Maior número negativo de 4-bit: 1000_2 ($-2^3 = -8_{10}$)
- O **most significant bit** também indica o sinal (1 = negativo, 0 = positivo)
- Intervalo de um número de N -bit:

$$[-2^{N-1}, 2^{N-1}-1]$$

Representação de Números Negativos

$b_3b_2b_1b_0$	Sign and magnitude	1's complement	2's complement
0111	+7	+7	+7
0110	+6	+6	+6
0101	+5	+5	+5
0100	+4	+4	+4
0011	+3	+3	+3
0010	+2	+2	+2
0001	+1	+1	+1
0000	+0	+0	+0
1000	-0	-7	-8
1001	-1	-6	-7
1010	-2	-5	-6
1011	-3	-4	-5
1100	-4	-3	-4
1101	-5	-2	-3
1110	-6	-1	-2
1111	-7	-0	-1

Adição e Subtração Sinal e Magnitude

- Exemplo: $-5 + 5$:

$$\begin{array}{r} 1101 \\ + 0101 \\ \hline 10010 \end{array}$$

- Duas representações para o 0 (± 0):

$$\begin{array}{r} 1000 \\ 0000 \end{array}$$

Adição e Subtração Complemento de 1

$\begin{array}{r} (+5) \\ + (+2) \\ \hline (+7) \end{array}$	$\begin{array}{r} 0101 \\ + 0010 \\ \hline 0111 \end{array}$	$\begin{array}{r} (-5) \\ + (+2) \\ \hline (-3) \end{array}$	$\begin{array}{r} 1010 \\ + 0010 \\ \hline 1100 \end{array}$
--	--	--	--

$\begin{array}{r} (+5) \\ + (-2) \\ \hline (+3) \end{array}$	$\begin{array}{r} 0101 \\ + 1101 \\ \hline 10010 \\ \boxed{1} \\ \boxed{0011} \end{array}$	$\begin{array}{r} (-5) \\ + (-2) \\ \hline (-7) \end{array}$	$\begin{array}{r} 1010 \\ + 1101 \\ \hline 10111 \\ \boxed{1} \\ \boxed{1000} \end{array}$
--	--	--	--

Adição e Subtração Complemento de 2

$$\begin{array}{r} (+5) \quad 0101 \\ + (+2) \quad +0010 \\ \hline (+7) \quad 0111 \end{array}$$

$$\begin{array}{r} (-5) \quad 1011 \\ + (+2) \quad +0010 \\ \hline (-3) \quad 1101 \end{array}$$

$$\begin{array}{r} (+5) \quad 0101 \\ + (-2) \quad +1110 \\ \hline (+3) \quad 10011 \end{array}$$


ignore

$$\begin{array}{r} (-5) \quad 1011 \\ + (-2) \quad +1110 \\ \hline (-7) \quad 11001 \end{array}$$


ignore

Subtração em Complemento de 2

$$\begin{array}{r} (+5) \quad 0101 \\ - (+2) \quad \underline{-0010} \\ \hline (+3) \end{array} \quad \Rightarrow \quad \begin{array}{r} 0101 \\ + 1110 \\ \hline 10011 \end{array}$$

↑
ignore

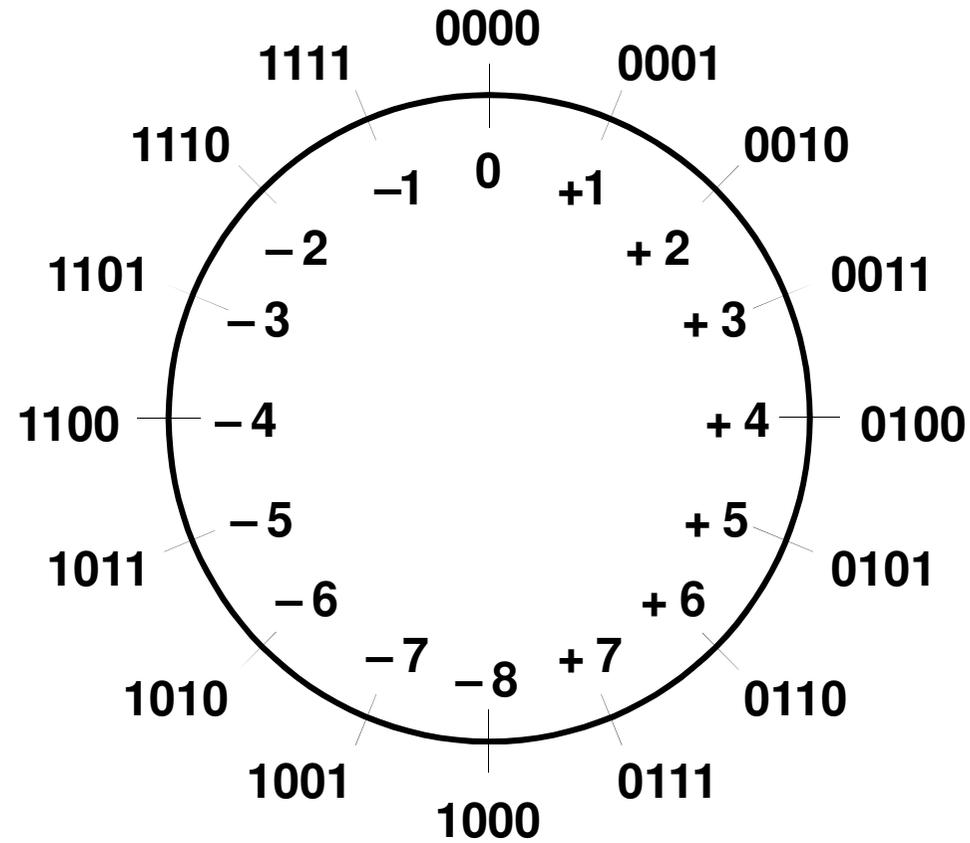
$$\begin{array}{r} (-5) \quad 1011 \\ - (+2) \quad \underline{-0010} \\ \hline (-7) \end{array} \quad \Rightarrow \quad \begin{array}{r} 1011 \\ + 1110 \\ \hline 11001 \end{array}$$

↑
ignore

$$\begin{array}{r} (+5) \quad 0101 \\ - (-2) \quad \underline{-1110} \\ \hline (+7) \end{array} \quad \Rightarrow \quad \begin{array}{r} 0101 \\ + 0010 \\ \hline 0111 \end{array}$$

$$\begin{array}{r} (-5) \quad 1011 \\ - (-2) \quad \underline{-1110} \\ \hline (-3) \end{array} \quad \Rightarrow \quad \begin{array}{r} 1011 \\ + 0010 \\ \hline 1101 \end{array}$$

Complemento de 2



Overflow em Complemento de 2

Quando há overflow?

Como detectar se houve overflow?

$$\begin{array}{r} (+7) \quad 0111 \\ + (+2) \quad +0010 \\ \hline (+9) \quad 1001 \\ c_4 = 0 \\ c_3 = 1 \end{array}$$

$$\begin{array}{r} (-7) \quad 1001 \\ + (+2) \quad +0010 \\ \hline (-5) \quad 1011 \\ c_4 = 0 \\ c_3 = 0 \end{array}$$

$$\begin{array}{r} (+7) \quad 0111 \\ + (-2) \quad +1110 \\ \hline (+5) \quad 10101 \\ c_4 = 1 \\ c_3 = 1 \end{array}$$

$$\begin{array}{r} (-7) \quad 1001 \\ + (-2) \quad +1110 \\ \hline (-9) \quad 10111 \\ c_4 = 1 \\ c_3 = 0 \end{array}$$

Extensão de N para M bits

- Um valor pode ter sua representação estendida de N bits para M bits (com $M > N$) usando:
 - Sign-extension
 - Zero-extension

Sign-extension

- O bit de sinal é copiado para os bits mais significativos.
- O valor do número é mantido o mesmo.
- Exemplo 1:
 - Representação de 3 com 4-bit = 0011
 - Representação sign-extended de 3 com 8-bit: 00000011
- Exemplo 2:
 - Representação de -5 com 4-bit = 1011
 - Representação sign-extended de -5 com 8-bit: 11111011

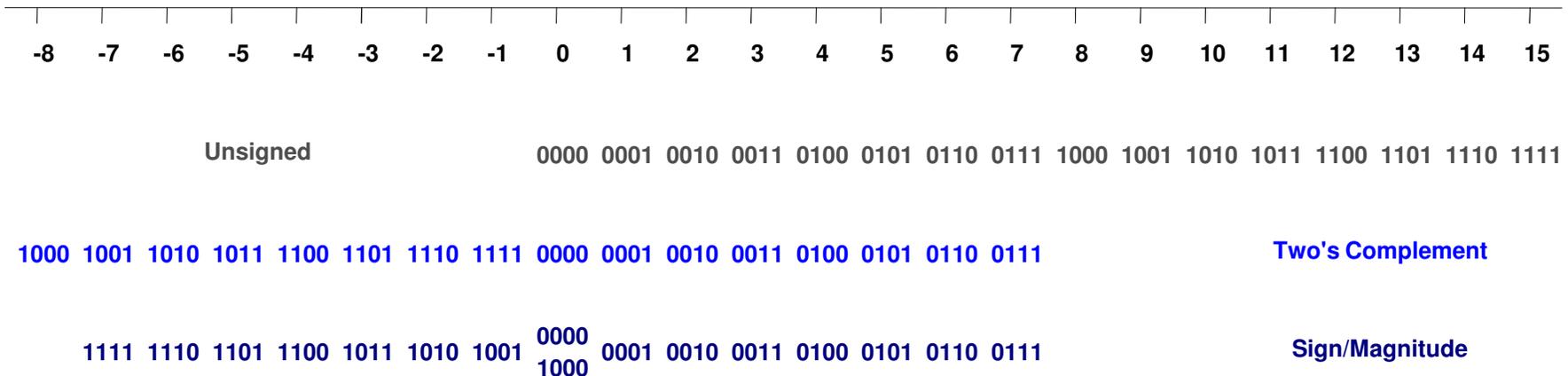
Zero-Extension

- Zeros são copiados nos bits mais significativos.
- O valor do número pode ser alterado.
- Exemplo 1:
 - Valor em 4-bit = 0011
 - Valor zero-extended com 8-bit: 00000011
- Exemplo 2:
 - Valor em 4-bit = 1011
 - Valor zero-extended com 8-bit: 00001011

Comparação

Number System	Range
Unsigned	$[0, 2^N-1]$
Sign/Magnitude	$[-(2^{N-1}-1), 2^{N-1}-1]$
Two's Complement	$[-2^{N-1}, 2^{N-1}-1]$

Representação em 4-bit:



Representações de Números Reais

- Ponto Fixo

- Exemplo: 6.75 com 4 bits para inteiros e 4 bits para a fração

6 0.75
0110.1100
01101100 ←

$$2^2 + 2^1 + 2^{-1} + 2^{-2} = 6.75$$

OBS.: O ponto binário não faz parte da notação e é implícito

Representações de Números Reais

- Represente -6.5_{10} usando uma representação binária de 8 bits (4 inteiro e 4 fração).

6.5  0110.1000

- Sinal/magnitude:

11101000

- Complemento de 2:

Inverte os bits: 10010111
Soma 1 ao **lsb**: + 1
 10011000

Representações de Números Reais

- Ponto Flutuante (Notação Científica):

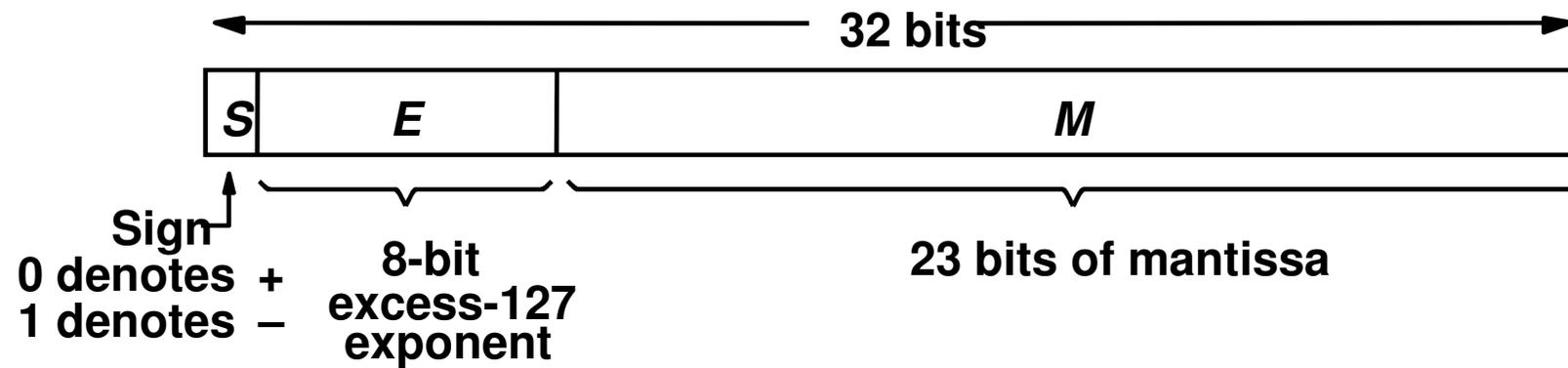
$$\pm \text{Mantissa} \times 10^E$$

$$\text{Mantissa} = \text{xxx.yyyyyy}$$

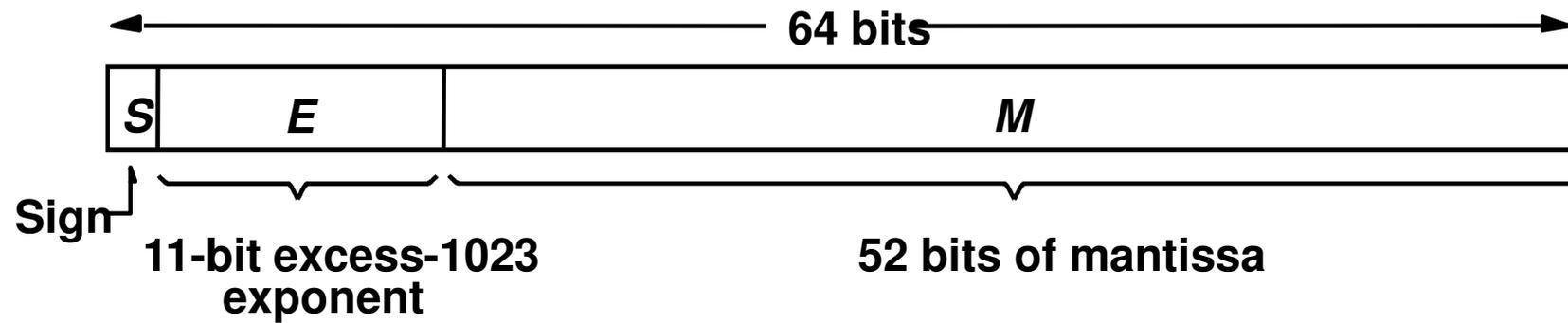
- Se Mantissa possui somente 1 dígito a esquerda do ponto decimal -> **forma padronizada**
e se diferente de zero -> **normalizado**
- Padrão IEEE-754
 - Normalizado
 - Bit Escondido

$$(-1)^s \times (1 + \text{Fração}) \times 2^E$$

IEEE-754 de Precisão Simples



IEEE-754 de Precisão Dupla



IEEE-754

Valores Representados

Exponent	Fraction	Represents
$e = 0 = E_{\text{mim}} - 1$	$f = 0$	± 0
$e = 0 = E_{\text{mim}} - 1$	$f \neq 0$	$0.f \times 2^{E_{\text{mim}}}$
$E_{\text{mim}} \leq e \leq E_{\text{max}}$		$1.f \times 2^e$
$e = E_{\text{max}} + 1$	$f = 0$	$\pm \infty$
$e = E_{\text{max}} + 1$	$f \neq 0$	NaN

IEEE-754: Exemplo

- Represente o valor 228_{10} usando a representação um floating point de 32-bit

$$228_{10} = 11100100_2 = 1.11001 \times 2^7$$

$$\text{Biased exponent} = \text{bias} + 7$$

$$127 + 7 = 134 = 0x10000110_2$$

1 bit	8 bits	23 bits
0	10000110	110 0100 0000 0000 0000 0000
Sign	Biased Exponent	Fraction

BCD

Binary-Coded-Decimal

Decimal digit	BCD code
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Adição Usando BCD

$$\begin{array}{r}
 X \quad \quad 0111 \quad \quad 7 \\
 + Y \quad + 0101 \quad + 5 \\
 \hline
 Z \quad \quad 1100 \quad 12 \\
 \quad \quad + 0110 \\
 \hline
 \text{carry} \rightarrow 10010 \\
 \quad \quad \underbrace{\hspace{2em}} \\
 \quad \quad S = 2
 \end{array}$$

$$\begin{array}{r}
 X \quad \quad 1000 \quad \quad 8 \\
 + Y \quad + 1001 \quad + 9 \\
 \hline
 Z \quad \quad 10001 \quad 17 \\
 \quad \quad + 0110 \\
 \hline
 \text{carry} \rightarrow 10111 \\
 \quad \quad \underbrace{\hspace{2em}} \\
 \quad \quad S = 7
 \end{array}$$

“No mundo há 10 tipos de pessoas: as que sabem contar em binário e as que não sabem”