MC542

Organização de Computadores
Teoria e Prática

2007

Prof. Paulo Cesar Centoducatte

ducatte@ic.unicamp.br

www.ic.unicamp.br/~ducatte

MC542
1.1

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
1.2

MC542

Introdução

Abstração, Sistemas Numéricos

"DDCA" - (Capítulo 1)
"FDL" - (Capítulo 5)

MC542
1.3

Abstração, Sistemas Numéricos, Tecnologia
Sumário

- Objetivos
- Abstração
  - Abstração Digital
- Binário
- Representação de Números
  - Posicional
  - Inteiros sem Sinal
    - Decimal
    - Binário
    - Hexadecimal e Octal
    - Conversão entre bases
    - Valores e Intervalos
  - Bits, Bytes, Nibbles…
  - Soma de Números Inteiros e Overrflow

MC542
1.4

Abstração, Sistemas Numéricos, Tecnologia
Sumário

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

MC542
1.5

Objetivos

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

MC542
1.6

Abstração

	Application Software	programs
	Operating Systems	device drivers
focus of this course	Architecture	instructions registers
focus of this course	Micro-architecture	datapaths controllers
focus of this course	Logic	adders memories
focus of this course	Digital Circuits	AND gates NOT gates
	Analog Circuits	amplifiers filters
	Devices	transistors diodes
	Physics	electrons

Um sistema pode ser visto com níveis de detalhes diferentes.

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

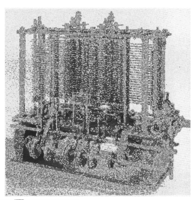
Níveis de abstração para um sistema computacional

Abstração Digital

- A maioria das variáveis físicas são continuas, 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 específica para representar o 1 e o 0
- *Bit*: *B*inary dig*it*

Representação de Números Posicional

- Decimal – Inteiros sem Sinal

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

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

1000's 100's 10's 1's

$$5374_{10} = 5 \times 10^3 + 3 \times 10^2 + 7 \times 10^1 + 4 \times 10^0$$

milhares centenas dezenas unidades

Representação de Números Posicional

- Binário – Inteiros sem sinal

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

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

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

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

$$1101_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 13_{10}$$

Representação Posicional
Conversão entre Decimal e Binário

$$V(B) = b_{n-1}x2^{n-1} + b_{n-2}xb^{n-2} + \ldots + b_1x2^1 + b_0x2^0$$

$$V(B) = b_{n-1}x2^{n-1} + b_{n-2}xb^{n-2} + \ldots + b_1x2^1 + b_0$$

$$\frac{V(B)}{2} = b_{n-1}x2^{n-2} + b_{n-2}xb^{n-3} + \ldots + b_1 + \frac{b_0}{2}$$

Conversão de Decimal para Binário: Divisão sucessiva por 2

Exemplo

Convert $(857)_{10}$

			Remainder	
$857 \div 2$	=	428	1	LSB
$428 \div 2$	=	214	0	
$214 \div 2$	=	107	0	
$107 \div 2$	=	53	1	
$53 \div 2$	=	26	1	
$26 \div 2$	=	13	0	
$13 \div 2$	=	6	1	
$6 \div 2$	=	3	0	
$3 \div 2$	=	1	1	
$1 \div 2$	=	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 *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

  $010010101111_2 = 1 + 2 + 4 + 8 + 32 + 128 + 1024 = 1199_{10}$

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

Bits, Bytes, Nibbles…

- Bits

  10010110

  most significant bit — least significant bit

- Bytes & Nibbles

  byte

  10010110

  nibble

- Bytes

  CEBF9AD7

  most significant byte — least significant byte

Potências de 2

- 2^{10} = 1 kilo ≈ 1000 (1024)

- 2^{20} = 1 mega ≈ 1 milhão(1.048.576)

- 2^{30} = 1 giga ≈ 1 bilhão (1.073.741.824)

Estimando Potência de 2

- Qual o valor de 2^{22}?

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

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

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

Soma

- Decimal

```
   11  ← carries
   3734
 + 5168
 ------
   8902
```

- Binária

```
   11  ← carries
   1011
 + 0011
 ------
   1110
```

Soma Binária: Exemplos

- Some os seguintes números:

```
   1001            1011
+  0101         +  0110
```

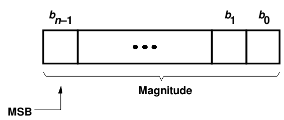

Overflow

- 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

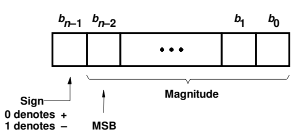
```
  11 1
   1101
+  0101
  10010
```


Representação de Números Negativos

Sinal e Magnitude



Número sem Sinal



Número com Sinal

Representação de Números Negativos

- **Sinal e Magnitude**
  - 1 bit de signal, N-1 bits de magnitude
  - O bit de sinal é o mais significativo (mais a esquerda)
    - » Número negativo: 1
    - » Número possitivo: 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 subtraíndo 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 -> 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 subtraíndo seu positivo P de 2^n

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

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

Exemplo: se n = 4
então:

$K = 2^4 - P$
$K = 16 - P$
$K = (10000)_2 - P$

K = -7 -> P = 7
$7 = (0111)_2$
$-7 = (10000)_2 - (0111)_2$
$-7 = (1001)_2$

Representação de Números Negativos

- **Complemento de 2**
  - Regra Prática

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

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

$K = 11\ldots11 - (p_{n-1} \ldots p_0) + 1$

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

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:**

```
  1101
+ 0101
 10010
```

- **Duas representações para o 0 (± 0):**

```
1000
0000
```


Adição e Subtração
Complemento de 1

```
 (+5)     0101       (−5)     1010
+(+2)   +0010       +(+2)   +0010
 (+7)     0111       (-3)     1100

 (+5)     0101       (−5)     1010
+(−2)   +1101       +(−2)   +1101
 (+3)    10010       (−7)    10111
             1                   1
          0011                1000
```


Adição e Subtração
Complemento de 2

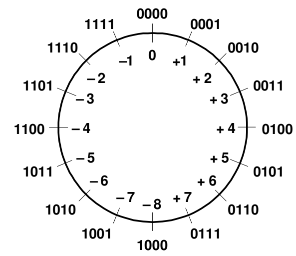
```
 (+5)     0101       (−5)     1011
+(+2)   +0010       +(+2)   +0010
 (+7)     0111       (−3)     1101

 (+5)     0101       (−5)     1011
+(−2)   +1110       +(−2)   +1110
 (+3)    10011       (−7)    11001
         ignore              ignore
```

Subtração em Complemento de 2

(+5) − (+2) = (+3)	0101 − 0010	⟹	0101 + 1110 = 10011 (ignore)
(−5) − (+2) = (−7)	1011 − 0010	⟹	1011 + 1110 = 11001 (ignore)
(+5) − (−2) = (+7)	0101 − 1110	⟹	0101 + 0010 = 0111
(−5) − (−2) = (−3)	1011 − 1110	⟹	1011 + 0010 = 1101

Complemento de 2



Overflow em Complemento de 2

Quando há overflow?
Como detectar se houve overflow?

(+7) + (+2) = (+9)	0111 + 0010 = 1001; c_4 = 0, c_3 = 1	(−7) + (+2) = (−5)	1001 + 0010 = 1011; c_4 = 0, c_3 = 0
(+7) + (−2) = (+5)	0111 + 1110 = 10101; c_4 = 1, c_3 = 1	(−7) + (−2) = (−9)	1001 + 1110 = 10111; c_4 = 1, c_3 = 0

Extensão de N para M bits

- Um valor pode ter sua representação extendida 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:

	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Unsigned									0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Two's Complement	1000	1001	1010	1011	1100	1101	1110	1111	0000	0001	0010	0011	0100	0101	0110	0111								
Sign/Magnitude		1111	1110	1101	1100	1011	1010	1001	0000 1000	0001	0010	0011	0100	0101	0110	0111								

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):

$\pm$ Mantissa x 10^E

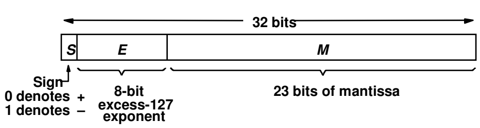
Mantissa = xxx.yyyyyy

- Se Mantissa possui somente 1 dígito a esquerda do ponto decimal -> forma padronizada
  e se diferente de zero -> normalizado

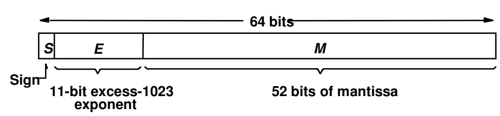
- Padrão IEEE-754
  - Normalizado
  - Bit Escondido

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

IEEE-754 de Precisão Simples



IEEE-754 de Precisão Dupla



IEEE-754
Valores Representados

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

MC542
1.49

IEEE-754: Exemplo

$-0{,}75_{10} = -0{,}11_2$

Normalizando ➔ $1{,}1 \times 2^{-1}$

31	30 … 23	22 … 0
1	0 1 1 1 1 1 1 0	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

MC542
1.50

IEEE-754: Exemplo

Qual o decimal correspondente ?

31	30 … 23	22 … 0
1	1 0 0 0 0 0 0 1	1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

$N = -(1+0.25) \times 2^{(129-127)} = -1{,}25 \times 4 = -5{,}0$

MC542
1.51

IEEE-754: Exemplo

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

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

Biased exponent = 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

MC542
1.52

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

MC542
1.53

Adição Usando BCD

X	0 1 1 1	7
+ Y	+ 0 1 0 1	+ 5
Z	1 1 0 0	12
	+ 0 1 1 0	
carry ⟶	1 0 0 1 0	
	S = 2	

X	1 0 0 0	8
+ Y	+ 1 0 0 1	+ 9
Z	1 0 0 0 1	17
	+ 0 1 1 0	
carry ⟶	1 0 1 1 1	
	S = 7	

MC542
1.54

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

MC542
1.55