Linguagens de Montagem

Revisão dos Sistemas de Numeração

Aula 00

Edmar André Bellorini

Ano letivo 2021

Objetivos desta revisão

- Computadores e o mundo binário
- Dados são armazenado no formato binário
- Dados são interpretados em diversos formatos:
  - Binária
  - Decimal
  - Hexadecimal
  - Octal

Um bom domínio dos sistemas de numeração Binário e Hexadecimal leva a uma menor dificuldade na disciplina de Linguagens de Montagem.

Representação Binária

- Forma mais simples de representação dos números Naturais em máquinas digitais
- Inteiros Não-Sinalizados
- É dado por:

$$v = \sum_{n-1}^{0} 2^i * a_i$$

sendo:
$i = 0 \rightarrow$ o bit menos significativo da string binária
$v \rightarrow$ o valor quantitativo do número Natural

Representação Binária - Exemplo 01 com 8 bits de largura

- $0100\ 0110^{LSb}$

i	7	6	5	4	3	2	1	0
Bit	0	1	0	0	0	1	1	0

considerando $v = \sum_{n-1}^{0} 2^i * a_i$, temos:

$2^7 * 0 + 2^6 * 1 + 2^5 * 0 + 2^4 * 0 + 2^3 * 0 + 2^2 * 1 + 2^1 * 1 + 2^0 * 0$

$0 + 64 + 0 + 0 + 0 + 4 + 2 + 0 = 70$

Representação Binária - Exemplo 02 com 8 bits de largura

- $1100\ 0110^{LSb}$

i	7	6	5	4	3	2	1	0
Bit	1	1	0	0	0	1	1	0

considerando $\sum_{n-1}^{0} 2^i * a_i$, temos:

$$2^7 * 1 + 2^6 * 1 + 2^5 * 0 + 2^4 * 0 + 2^3 * 0 + 2^2 * 1 + 2^1 * 1 + 2^0 * 0$$

$$128 + 64 + 0 + 0 + 0 + 4 + 2 + 0 = 198$$

Observações

- Em teoria, as strings Binárias podem ser consideradas virtualmente infinitas
- Porém, as máquinas se limitam em tamanhos 2^i
  - com $i = 2, 3, 4$ e 5 em arquiteturas de 32 bits
  - com $i = 2, 3, 4, 5$ e 6 em arquiteturas de 64 bits
    * existem palavras com tamanhos diferentes nas instruções vetoriais

Operação de Adição Binária

- Dado por:

$$\begin{array}{rr} & c_i \\ & b_i \\ + & a_i \\ \hline c_{i+1} & s_i \end{array}$$

- onde:
  - $c_i \rightarrow$ bit carry-in, isto é, carry-out da operação dos bits $_{i-1}$
  - b_i e $a_i \rightarrow$ bits das palavras a e b na posição i
  - $s_i \rightarrow$ bit solução para posição i
  - $c_{i+1} \rightarrow$ bit carry-out da posição i

Regras de Adição Binária

- Dadas por:

Situação	bit c_i	bit b_i	bit a_i	bit s_i	bit c_{i+1}
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

Adição Binária - Exemplo 01 (slide a)

- $12_d + 28_d$

Adição Binária - Exemplo 01 (slide b)

- Conversão:
  - $b = 12_d \rightarrow 0000\ 1100_b$
  - $a = 28_d \rightarrow 0001\ 1100_b$
- Adição:

	7	6	5	4	3	2	1	0	i
			1	1	1				c_i
	0	0	0	0	1	1	0	0	b_i
+	0	0	0	1	1	1	0	0	a_i
	0	0	1	0	1	0	0	0	s_i

- Resposta:
  $2^7 * 0 + 2^6 * 0 + 2^5 * 1 + 2^4 * 0 + 2^3 * 1 + 2^2 * 0 + 2^1 * 0 + 2^0 * 0$
  $0 + 0 + 32 + 0 + 8 + 0 + 0 + 0 = 40_d$

Adição Binária - Exemplo 02 (slide a)

- $254_d + 3_d$

Adição Binária - Exemplo 02 (slide b)

- Conversão:
  - $b = 254_d \rightarrow 1111\ 1110_b$
  - $a = 3_d \rightarrow 0000\ 0011_b$
- Adição:

?	7	6	5	4	3	2	1	0	i
1	1	1	1	1	1	1			c_i
	1	1	1	1	1	1	1	0	b_i
+	0	0	0	0	0	0	1	1	a_i
1	0	0	0	0	0	0	0	1	s_i

- Resposta:
  $2^7 * 0 + 2^6 * 0 + 2^5 * 0 + 2^4 * 0 + 2^3 * 0 + 2^2 * 0 + 2^1 * 0 + 2^0 * 1$
  $0 + 0 + 0 + 0 + 0 + 0 + 0 + 1 = 1_d$

Adição Binária - Exemplo 02 (slide c)

- $254_d + 3_d = 1_d$?
  - Quantidade de bits limitado
  - não é possível representar a solução correta
    $257_d = 1\ 0000\ 0001_b$
  - Falha na representação
  - Flag especial chamada **CF** (*CarryFlag*)
    - CF será usada durante as aulas com instruções aritméticas de inteiros

Representação em Complemento de Dois (C2)

- Forma de representação dos números Inteiros
- Sinalizados
- Dado por:

$$v = -2^{n-1} * a_{n-1} \sum_{n-2}^{0} 2^i * a_i$$

sendo:
$i = 0 \rightarrow$ o bit menos significativo da string binária
$v \rightarrow$ o valor quantitativo do número Inteiro

Representação em C2 - Exemplo 01 com 8 bits de largura

- $0100\ 0110^{LSb}$

i	7	6	5	4	3	2	1	0
Bit	0	1	0	0	0	1	1	0

considerando $v = -2^{n-1} * a_{n-1} \sum_{n-2}^{0} 2^i * a_i$, temos:

$$-2^7 * 0 + 2^6 * 1 + 2^5 * 0 + 2^4 * 0 + 2^3 * 0 + 2^2 * 1 + 2^1 * 1 + 2^0 * 0$$
$$-0 + 64 + 0 + 0 + 0 + 4 + 2 + 0 = 70$$

Representação em C2 - Exemplo 02 com 8 bits de largura

- $1100\ 0110^{LSb}$

i	7	6	5	4	3	2	1	0
Bit	1	1	0	0	0	1	1	0

considerando $v = -2^{n-1} * a_{n-1} \sum_{n-2}^{0} 2^i * a_i$, temos:

$$-2^7 * 1 + 2^6 * 1 + 2^5 * 0 + 2^4 * 0 + 2^3 * 0 + 2^2 * 1 + 2^1 * 1 + 2^0 * 0$$
$$-128 + 64 + 0 + 0 + 0 + 4 + 2 + 0 = -58$$

Operação de Adição em C2

- Segue todas as regras da Adição Binária
- Dado por:

$$\begin{array}{rr} & c_i \\ & b_i \\ + & a_i \\ \hline c_{i+1} & s_i \end{array}$$

- onde:
  - $c_i \rightarrow$ bit carry-in, isto é, carry-out da operação dos bits $_{i-1}$
  - b_i e $a_i \rightarrow$ bits das palavras a e b na posição i
  - $s_i \rightarrow$ bit solução para posição i
  - $c_{i+1} \rightarrow$ bit carry-out da posição i

Adição em C2 - Exemplo 01 (slide a)

- $12_d + 28_d$

Adição em C2 - Exemplo 01 (slide b)

- Conversão:
  - $b = 12_d \rightarrow 0000\ 1100_b$
  - $a = 28_d \rightarrow 0001\ 1100_b$
- Adição:

	7	6	5	4	3	2	1	0	i
			1	1	1				c_i
	0	0	0	0	1	1	0	0	b_i
+	0	0	0	1	1	1	0	0	a_i
	0	0	1	0	1	0	0	0	s_i

- Resposta:
  $-2^7 * 0 + 2^6 * 0 + 2^5 * 1 + 2^4 * 0 + 2^3 * 1 + 2^2 * 0 + 2^1 * 0 + 2^0 * 0$
  $-0 + 0 + 32 + 0 + 8 + 0 + 0 + 0 = 40_d$

Adição em C2 - Exemplo 02 (slide a)

- $-116_d + 2_d$

Adição em C2 - Exemplo 02 (slide b)

- Conversão:
  - $b = -116_d \rightarrow 1000\ 1100_b$
  - $a = 2_d \rightarrow 0000\ 0010_b$
- Adição:

	7	6	5	4	3	2	1	0	i
									c_i
	1	0	0	0	1	1	0	0	b_i
+	0	0	0	0	0	0	1	0	a_i
	1	0	0	0	1	1	1	0	s_i

- Resposta:
  $-2^7 * 1 + 2^6 * 0 + 2^5 * 0 + 2^4 * 0 + 2^3 * 1 + 2^2 * 1 + 2^1 * 1 + 2^0 * 0$
  $-128 + 0 + 0 + 0 + 8 + 4 + 2 + 0 = -114_d$

Adição em C2 - Exemplo 03 (slide a)

- $-116_d + -2_d$

Adição em C2 - Exemplo 03 (slide b)

- Conversão:
  - $b = -116_d \rightarrow 1000\ 1100_b$
  - $a = -2_d \rightarrow 1111\ 1110_b$
- Adição:

?	7	6	5	4	3	2	1	0	i
1	1	1	1	1	1				c_i
	1	0	0	0	1	1	0	0	b_i
+	1	1	1	1	1	1	1	0	a_i
1	1	0	0	0	1	0	1	0	s_i

- Resposta:
  $-2^7 * 1 + 2^6 * 0 + 2^5 * 0 + 2^4 * 0 + 2^3 * 1 + 2^2 * 0 + 2^1 * 1 + 2^0 * 0$
  $-128 + 0 + 0 + 0 + 8 + 0 + 2 + 0 = -118_d$

Adição em C2 - Exemplo 03 (slide c)

- $-116_d + -2_d = -118_d$
- e com CF $= 1$
  - CF é ignorada durante as operações sinalizadas
- Existe uma outra Flag que anuncia possíveis erros nas operações sinalizadas
  - *Overflow Flag*, carinhosamente chamada de **OF**
  - e neste exemplo, ela retornou 0 (zero)

Adição em C2 - Exemplo 04 (slide a)

- $-126_d + -4_d$

Adição em C2 - Exemplo 04 (slide b)

- Conversão:
  - $b = -126_d \rightarrow 1000\ 0010_b$
  - $a = -4_d \rightarrow 1111\ 1100_b$
- Adição:

?	7	6	5	4	3	2	1	0	i
1									c_i
	1	0	0	0	0	0	1	0	b_i
+	1	1	1	1	1	1	0	0	a_i
1	0	1	1	1	1	0	1	0	s_i

- Resposta:
  $-2^7 * 0 + 2^6 * 1 + 2^5 * 1 + 2^4 * 1 + 2^3 * 1 + 2^2 * 1 + 2^1 * 1 + 2^0 * 0$
  $-0 + 64 + 32 + 16 + 8 + 4 + 2 + 0 = 126_d$ (???)

Adição em C2 - Exemplo 04 (slide c)

- $-126_d + -4_d = 126_d$
- Adicionar 2 números negativos resultou em um número positivo!
  - OF = 1
- *Overflow Flag* sempre é ligada em **1** quando, ao somar-se 2 números de **mesmo** sinal, o resultado é de sinal **contrário**.
  - OF = 1
- *Overflow Flag* nunca ocorre em adição de números com sinais **diferentes**.
  - OF = 0

Carry vs Overflow

Flags	Valor	Representação	
		Binária	C2
CF	0	Ok	Desconsiderada
	1	Falha	Desconsiderada
OF	0	Desconsiderada	Ok
	1	Desconsiderada	Falha

Operação Complemento de 2

- Operação que altera o sinal de um número representado em Complemento de 2
- Dado por

$$\begin{array}{rr} & \overline{a_i} \\ + & 1_b \\ \hline & -a_i \end{array}$$

- onde:
  - $\overline{a_i} \rightarrow$ bits da palavra em representação C2, positivo ou negativo
  - $1_b \rightarrow$ valor numérico 1 em representação em C2
  - $-a_i \rightarrow$ bits solução da inversão de sinal de $\overline{a_i}$
    positivo passa a ser negativo
    negativo passa a ser positivo
    * problema com menor valor negativo!

Operação Complemento de 2 - Exemplos

- Exemplo 1: Operação Complemento de 2 em 42_d
- Exemplo 2: Operação Complemento de 2 em -42_d
- Exemplo 3: Operação Complemento de 2 em 127_d
- Exemplo 4: Operação Complemento de 2 em -128_d

Operação Complemento de 2 - Exemplo 01

- Conversão:
  - $a = 42_d \rightarrow 0010\ 1010_b$
  - $\overline{0010\ 1010} \rightarrow 1101\ 0101$
- Adição:

?	7	6	5	4	3	2	1	0	i
							1		c_i
	1	1	0	1	0	1	0	1	$\overline{a_i}$
	0	0	0	0	0	0	0	1	1_b
0	1	1	0	1	0	1	1	0	$-a_i$

- Resposta:
  $-2^7 * 1 + 2^6 * 1 + 2^5 * 0 + 2^4 * 1 + 2^3 * 0 + 2^2 * 1 + 2^1 * 1 + 2^0 * 0$
  $-128 + 64 + 0 + 16 + 0 + 4 + 2 + 0 = -42_d$
  (sinal alterado com sucesso)

Operação Complemento de 2 - Exemplo 02

- Conversão:
  - $a = -42_d \rightarrow 1101\ 0110$
  - $\overline{1101\ 0110} \rightarrow 0010\ 1001_b$
- Adição:

?	7	6	5	4	3	2	1	0	i
							1		c_i
	0	0	1	0	1	0	0	1	$\overline{a_i}$
	0	0	0	0	0	0	0	1	1_b
0	0	0	1	0	1	0	1	0	s_i

- Resposta:
  $-2^7 * 0 + 2^6 * 0 + 2^5 * 1 + 2^4 * 0 + 2^3 * 1 + 2^2 * 0 + 2^1 * 1 + 2^0 * 0$
  $-0 + 0 + 32 + 0 + 8 + 0 + 2 + 0 = 42_d$
  (sinal alterado com sucesso)

Operação Complemento de 2 - Exemplo 03

- Conversão:
  - $a = 127_d \rightarrow 0111\ 1111_b$
  - $\overline{0111\ 1111} \rightarrow 1000\ 0000$
- Adição:

?	7	6	5	4	3	2	1	0	i
									c_i
	1	0	0	0	0	0	0	0	$\overline{a_i}$
	0	0	0	0	0	0	0	1	1_b
0	1	0	0	0	0	0	0	1	$-a_i$

- Resposta:
  $-2^7 * 1 + 2^6 * 0 + 2^5 * 0 + 2^4 * 0 + 2^3 * 0 + 2^2 * 0 + 2^1 * 1 + 2^0 * 0$
  $-128 + 0 + 0 + 0 + 0 + 0 + 0 + 1 = -127_d$
  (sinal alterado com sucesso)

Operação Complemento de 2 - Exemplo 04 - problema

- Conversão:
  - $b = -128_d \rightarrow 1000\ 0000$
  - $\overline{1000\ 0000} \rightarrow 0111\ 1111_b$
- Adição:

?	7	6	5	4	3	2	1	0	i
	1	1	1	1	1	1	1		c_i
	0	1	1	1	1	1	1	1	$\overline{a_i}$
	0	0	0	0	0	0	0	1	1_b
0	1	0	0	0	0	0	0	0	s_i

- Resposta:
  $-2^7 * 1 + 2^6 * 0 + 2^5 * 0 + 2^4 * 0 + 2^3 * 0 + 2^2 * 0 + 2^1 * 0 + 2^0 * 0$
  $-128 + 0 + 0 + 0 + 0 + 0 + 0 + 0 = -128_d$
  (sinal não foi alterado com sucesso)

Representação em Hexadecimal (Hex)

- É uma representação que compacta a visualização de uma string binária
- A string binária é dividida em *nibbles*
  - 1 *nibble* = 4 bits = 16 valores (0 até F)
- Transcrição de um valor *nibble* para um dígito hexadecimal

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

Exemplo de Representação em Hexadecimal

- $0000\ 1100_b$
  $0C_h$
- $0001\ 1100_b$
  $1C_h$
- $1000\ 1100_b$
  $8C_h$
- $0000\ 0001\ 1111\ 1101\ 1011\ 0110\ 1110\ 1010\ 1011\ 0101_b$
  $01FDB6EAB5_h$

Representação em Octal (Oct)

- É uma representação que compacta a visualização de uma string binária
- A string binária é dividida a cada 3 bits
- Assemelha-se à estrutura da Representação Hexadecimal
  - 3 bits = 8 valores (0 até 7)
- Transcrição de um valor de 3 bits para um dígito octal

dec	3b	oct
0	000	0
1	001	1
2	010	2
3	011	3

dec	3b	oct
4	100	4
5	101	5
6	110	6
7	111	7

Considerações Finais

- Como diferenciar um 1 binários de um 1 decimal ou 1 hexadecimal?
  - Binário: sufixo $_b$ ou $_2$
    $0110\ 0110_b$ ou $0110\ 0110_2$
  - Decimal: sufixo $_d$ ou $_{10}$
    77_d ou 77_{10}
  - Hexadecimal: sufixo $_h$ ou $_{16}$ ou prefixo $0x$
    $01FDB6EAB5_h$ ou $01FDB6EAB5_{16}$ ou $0x01FDB6EAB5$
  - Octal: sufixo $_o$ ou $_8$
    05347_o ou 05347_8
- Operação de Negação é diferente da Operação de Complemento
  - Não confundir Representação em Complemento de 2 com Operação de Complemento de 2

Atividades para praticar (slide a)

- Converta os valores em base decimal para base binária de 8 bits de palavra:
  (respostas estão entre parênteses)
  1. 13_d (R: 0000 1101)
  2. 43_d (R: 0010 1011)
  3. 130_d (R: 1000 0010)
- Converta os valores binários abaixo para base decimal:
  (respostas estão entre parênteses)
  1. $0001\ 1001_b$ (R: 25_b)
  2. $0110\ 0000_b$ (R: 96_d)
  3. $1100\ 0010_b$ (R: 194_d)

Atividades para praticar (slide b)

- Adicione os seguintes valores em base binária de 8 bits de palavra: (respostas estão entre parênteses)
  1. $13_d + 43_d$ (R: 0000 1101 + 0010 1011 = 0011 1000)
  2. $13_d + 181_d$ (R: 0000 1101 + 1011 0101 = 1100 0010)
  3. $127_d + 201_d$ (R: 0111 1111 + 1100 1001 = ????)
     (não é possível representar o valor 328_d em binário com palavra de 8 bits)

Atividades para praticar (slide c)

- Converta os valores em base decimal para representação em Complemento de 2 de 8 bits de palavra:
  (respostas estão entre parênteses)
  1. 97_d (R: 0110 0001_{c2})
  2. 126_d (R: 0111 1110_{c2})
  3. -95_d (R: 1010 0001_{c2})
- Converta os valores em representação em Complemento de 2 abaixo para base decimal:
  (respostas estão entre parênteses)
  1. 1101 1100_{c2} (R: -36_d)
  2. 0011 1100_{c2} (R: 60_d)
  3. 1000 0011_{c2} (R: -125_d)
  4. 0110 0110_{c2} (R: 102_d)

Atividades para praticar (slide d)

- Adicione os seguintes valores em representação em Complemento de 2 com palavras de 8bits:
  (respostas estão entre parênteses)
  1. $13_d + 43_d$ (R: 0000 1101 + 0010 1011 = 0011 1000)
  2. $-13_d + -43d$ (R: 1111 0011 + 1101 0101 = 1100 1000)
  3. $126_d + -95d$ (R: 0111 1110 + 1010 0001 = 0001 1111)
  4. $126_d + 126d$ (R: 0111 1110 + 0111 1110 =????)
     (uhm, talvez algo como um overflow deve ter ocorrido!)

Fim do Documento

Dúvidas?

Próxima aula:

- Aula 01: Introdução à Linguagem de Máquina
  - Diferença entre linguagens (alto nível, baixo nível e de máquina)
  - Arquitetura de Von Neumann e Estrutura de memória
  - Introdução à unidade de processamento
  - Montador NASM e Sintaxes
  - “Hello World“