

Organização Básica de computadores e linguagem de montagem

Prof. Edson Borin

2º Semestre de 2015

Representação de Informações no Computador

Representação de Informações

- Como representar informações em um computador
 - Números inteiros?
 - Texto?
 - Registros?
 - Vetores?

Representação de Informações

- Informações são representadas através de dígitos binários, ou *BITs* (**B**inary **di**gi**T**s).
- Dígitos 0 e 1
- Quantos estados (ou números) distintos podemos representar com 3 dígitos da base binária?

Representação de Informações

- Informações são representadas através de dígitos binários, ou *BITs* (**B**inary *digi***T**s).
- Dígitos 0 e 1
- Quantos estados (ou números) distintos podemos representar com 3 dígitos da base binária?
 - 8 estados se utilizarmos notação posicional
 - 4 estados se utilizarmos notação não posicional
 - 1: 001, 010, 100 (um *bit* 1 e dois *bits* 0)
 - 2: 110, 101, 011 (um *bit* 0 e dois *bits* 1)
 - 3: 000 (três *bits* 0)
 - 4: 111 (três *bits* 1)

Representação de Informações

- Notação posicional: valor do dígito depende da sua posição.
- Exemplo: Número decimal 132
 - Valor do dígito 2 = 2
 - Valor do dígito 3 = 30
 - Valor do dígito 1 = 100

Representação de Informações

- Notação posicional: valor do dígito depende da sua posição.
- Exemplo: Número decimal 132
 - Valor do dígito 2 = 2
 - Valor do dígito 3 = 30
 - Valor do dígito 1 = 100
- “Informações no computador são representadas através de números, codificados na base binária com notação posicional”

Bases numéricas

- A quantidade de dígitos distintos define a base numérica. Exemplos
 - Base 2, ou binária \Rightarrow 2 dígitos distintos: 0 e 1
 - Base 8, ou octal \Rightarrow 8 dígitos distintos: 0, 1, ..., 7
 - Base 10, ou decimal \Rightarrow 10 dígitos distintos: 0, ..., 9
 - ...
- Quais são os dígitos utilizados na base 16?

Bases numéricas

- A quantidade de dígitos distintos define a base numérica. Exemplos
 - Base 2, ou binária \Rightarrow 2 dígitos distintos: 0 e 1
 - Base 8, ou octal \Rightarrow 8 dígitos distintos: 0, 1, ..., 7
 - Base 10, ou decimal \Rightarrow 10 dígitos distintos: 0, ..., 9
 - ...
- Quais são os dígitos utilizados na base 16?
 - Dígitos da base hexadecimal: 0, 1, ..., 9, A, B, C, D, E, F

Bases numéricas

- Qual é a base dos números abaixo?
 - FE03
 - 8230
 - 9210
 - 1001

Bases numéricas

- Qual é a base dos números abaixo?
 - FE03
 - 8230
 - 9210
 - 1001
- Para distinguir temos que anotar o número com a base. Exemplos:
 - $FE03_{16}$
 - 1001_{10}
 - 1001_2

Bases numéricas

- Qual é o valor de cada dígito nos números abaixo?
 - 9210_{10}
 - 1001_2

Bases numéricas

- Qual é o valor de cada dígito nos números abaixo?
 - 9210_{10}
 - 1001_2
- O valor de um dígito ***d*** em um número na base ***t*** é dado por:
 - ***d* × *t* posição**
- Onde a posição é dada pela seguinte convenção:

Dígitos

0 0 0 0 9 2 | 0

Posição

7 6 5 4 3 2 | 0

Bases numéricas

- Qual é o valor de cada número abaixo em decimal?
 - 1001_2
 - FF_{16}

Bases numéricas

- Qual é o valor de cada número abaixo em decimal?
 - 1001_2
 - FF_{16}
- O valor de um número na base **t** com **n** dígitos é o somatório dos valores dos dígitos:

$$N_{10} = \sum_{i=0}^{n-1} \mathbf{d}_i \times \mathbf{t}^i$$

- onde **d_i** é o dígito na posição *i*.

Bases numéricas

- Qual é o valor de cada número abaixo em decimal?
 - $1001_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 9_{10}$
 - $FF_{16} = F \times 16^1 + F \times 16^0 = 15 \times 16 + 15 \times 1 = 255_{10}$
- O valor de um número na base **t** com **n** dígitos é o somatório dos valores dos dígitos:

$$N_{10} = \sum_{i=0}^{n-1} \mathbf{d}_i \times \mathbf{t}^i$$

- onde **d_i** é o dígito na posição *i*.

Conversão de bases numéricas

Tipo de conversão	Procedimento
Decimal => Binário	Divisões sucessivas por 2 até se obter zero no quociente. Leitura dos dígitos binários no resto de baixo para cima.
Binário => Decimal	Soma de potências de 2 cujo expoente é a posição do bit e cujo coeficiente é o próprio bit.
Hexadecimal => Binário	Expandir cada dígito hexa em quatro dígitos binários segundo seu valor.
Binário => Hexadecimal	Compactar cada quatro dígitos binários em um único dígito hexa segundo seu valor.
Decimal => Hexadecimal	Divisões sucessivas por 16 até se obter zero no quociente. Converter restos p/ dígitos hexadecimais. Leitura dos dígitos de baixo para cima.
Hexadecimal => Decimal	Soma de potências de 16 cujo expoente é a posição do dígito e cujo coeficiente é o valor do próprio dígito hexa.

Bases numéricas - Exercícios

- Qual o valor em binário dos seguintes números
 - 15_{16}
 - 139_{10}
- Qual o valor em hexadecimal dos seguintes números
 - 101001_2
 - 16_{10}
 - 240_{10}
 - 20_8

Números Sem Sinal

- Na representação sem sinal, todos os *bits* são utilizados como dígitos do número.
- Exemplo: Registradores com 3 bits podem representar 8 números distintos: 0 a 7

$$000 = 0_{10}$$

$$001 = 1_{10}$$

$$010 = 2_{10}$$

$$011 = 3_{10}$$

$$100 = 4_{10}$$

$$101 = 5_{10}$$

$$110 = 6_{10}$$

$$111 = 7_{10}$$

Números Com Sinal

- Três tipos de codificação mais conhecidas
 - Sinal e magnitude
 - Complemento de 1
 - Complemento de 2

Sinal e Magnitude

- Na representação “sinal e magnitude” o *bit* mais a esquerda (o mais significativo) representa o sinal do número e os outros *bits* representam a magnitude.
- Qual é o valor dos números abaixo na representação “sinal e magnitude” e sem sinal?
 - a) $0001\ 0101_2$
 - b) $1000\ 1010_2$

Sinal e Magnitude

- Na representação “sinal e magnitude” o *bit* mais a esquerda (o mais significativo) representa o sinal do número e os outros *bits* representam a magnitude.
- Qual é o valor dos números abaixo na representação “sinal e magnitude” e sem sinal?
 - a) $0001\ 0101_2$
 - b) $1000\ 1010_2$
- E estes números?
 - a) $0000\ 0000_2$
 - b) $1000\ 0000_2$

Sinal e Magnitude

Número	Sem sinal	Sinal e Mag.
000	0	0
001	1	1
010	2	2
011	3	3
100	4	-0
101	5	-1
110	6	-2
111	7	-3

Complemento de 1

- Na representação “complemento de 1” o *bit* mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

Número	Sem sinal	Sinal e Mag.	Comp. de 1
000	0	0	0
001	1	1	1
010	2	2	2
011	3	3	3
100	4	-0	?
101	5	-1	?
110	6	-2	?
111	7	-3	?

Complemento de 1

- Na representação “complemento de 1” o *bit* mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.

Número	Sem sinal	Sinal e Mag.	Comp. de 1
000	0	0	0
001	1	1	1
010	2	2	2
011	3	3	3
100	4	-0	-3
101	5	-1	-2
110	6	-2	-1
111	7	-3	-0

Complemento de 1

- Na representação “complemento de 1” o *bit* mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.
- Primeiro *bit* 0 \Rightarrow o número é positivo e o valor pode ser obtido da mesma maneira que na representação sem sinal
- Primeiro *bit* 1 \Rightarrow o número é negativo. Para descobrir a magnitude, basta inverter todos os *bits* e computar o valor na representação sem sinal.
- Qual é o valor de 10010_2 ?

Complemento de 1

- Na representação “complemento de 1” o *bit* mais à esquerda também indica o sinal, entretanto a magnitude é representada de maneira diferente.
- Primeiro *bit* 0 \Rightarrow o número é positivo e o valor pode ser obtido da mesma maneira que na representação sem sinal
- Primeiro *bit* 1 \Rightarrow o número é negativo. Para descobrir o valor, basta inverter todos os *bits* e computar o valor na representação sem sinal.
- Qual é o valor de 10010_2 ?
 - $10010_2 \Rightarrow 01101_2 = 13_{10}$. Logo: $10010_2 = -13_{10}$

Complemento de 2

- Na representação “complemento de 2” o *bit* mais à esquerda indica o sinal, entretanto a magnitude é representada de maneira diferente.

Número	Sem sinal	Sinal e Mag.	Comp. de 1	Comp. de 2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	?
101	5	-1	-2	?
110	6	-2	-1	?
111	7	-3	-0	?

Complemento de 2

- Na representação “complemento de 2” o *bit* mais à esquerda indica o sinal, entretanto a magnitude é representada de maneira diferente.

Número	Sem sinal	Sinal e Mag.	Comp. de 1	Comp. de 2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

Representação de Números

Número	Sem sinal	Sinal e Mag.	Comp. de 1	Comp. de 2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

- Representações “Sinal e Mag.” e “Comp. de 1” possuem dois zeros: 0 e -0
- A representação “complemento de 2” é a mais utilizada.

Representação de Números

Número	Sem sinal	Sinal e Mag.	Comp. de 1	Comp. de 2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

Maior	7	3	3	3
Menor	0	-3	-3	-4

Representação de Números

Número	Sem sinal	Sinal e Mag.	Comp. de 1	Comp. de 2
000	0	0	0	0
001	1	1	1	1
010	2	2	2	2
011	3	3	3	3
100	4	-0	-3	-4
101	5	-1	-2	-3
110	6	-2	-1	-2
111	7	-3	-0	-1

Maior	7	3	3	3
Menor	0	-3	-3	-4

Maior	$2^n - 1$	$2^{n-1} - 1$	$2^{n-1} - 1$	$2^{n-1} - 1$
Menor	0	$-(2^{n-1} - 1)$	$-(2^{n-1} - 1)$	$-(2^{n-1})$

Números no Computador

- “Informações no computador são representadas através de números, codificados na base binária com notação posicional”
- Quantos *bits* o computador usa para codificar cada número?

Números no Computador

- “Informações no computador são representadas através de números, codificados na base binária com notação posicional”
- Quantos *bits* o computador usa para codificar cada número?
 - O IAS utiliza 40 *bits*!
 - Palavras da memória possuem 40 *bits*
 - Registradores da ULA possuem 40 *bits*.

Números no Computador

- Computadores modernos codificam números com palavras de 8, 16, 32, 64 ou mais *bits*.
- Geralmente é uma potência de 2.
- Uma arquitetura de 32 *bits* é uma arquitetura que é capaz de armazenar e realizar operações aritméticas em números com até 32 *bits*.

Complemento de 2

- Números sinalizados de 32 *bits* em Complemento de 2:

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = 0_{10}$$

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = +1_{10}$$

$$0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = +2_{10}$$

...

$$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = +2,147,483,646_{10}$$

$$0111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = +2,147,483,647_{10}$$

$$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000_2 = -2,147,483,648_{10}$$

$$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0001_2 = -2,147,483,647_{10}$$

$$1000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2 = -2,147,483,646_{10}$$

...

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2 = -3_{10}$$

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1110_2 = -2_{10}$$

$$1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = -1_{10}$$

maxint



minint



Aritmética Binária: Soma e Subtração

- Como no ensino fundamental: (vai-um/vem-um)

0111 (7)	0111 (7)	0110 (6)
+ 0110 (6)	-0110 (6)	-0101 (5)
<hr/>		<hr/>
1101 (13)	0001 (1)	0001 (1)

- Subtração em complemento de 2 pode ser feita com uma soma ($A - B = A + (-B)$).
- Ex: $7 - 6 = 7 + (-6)$

$$\begin{array}{r} 0111 (+7) \\ + 1010 (-6) \\ \hline 0001 (=1) \end{array}$$

Aritmética Binária: *Overflow*

- *Overflow*: quando o resultado é maior (menor) do que a palavra do computador pode representar.
- Exemplo: Ocorre *overflow* na operação abaixo?

$$\begin{array}{r} 0111 \text{ (7)} \\ + 0001 \text{ (1)} \\ \hline 1000 \end{array}$$

Aritmética Binária: Detecção de *Overflow*

- Não existe *overflow* quando adicionamos um número positivo e um número negativo
- Não existe *overflow* quando os sinais dos números são os mesmos na subtração
- Ocorre *overflow* quando os valores afetam o sinal:
 - Somando dois números positivos dá um número negativo
 - Somando dois números negativos dá um número positivo
 - Subtrai um número negativo de um positivo e dá negativo
 - Subtrai um número positivo de um negativo e dá positivo

Aritmética Binária: Detecção de *Overflow*

- Exercício: Compute o resultado das operações abaixo e verifique se houve *overflow*
 - a) $4 + 5$ em uma representação com números sinalizados de 8 *bits*
 - b) $4 + 5$ em uma representação com números sinalizados de 4 *bits*

Representação de Caracteres

- Cada caractere é associado a um número distinto.
- Existem diversos padrões.
- Exemplo: Padrão ASCII - American Standard Code for Information -- Usa *7 bits*, (128 caracteres distintos) – Exemplo:

64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I

96	'
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i

48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9

Representação de Caracteres

- Cada caractere é associado a um número distinto.
- ASCII Usa *7 bits*
- Um texto é armazenado como uma cadeia de caracteres!
- Posições consecutivas da memória

Representação de Caracteres

- Cada caractere é associado a um número distinto.
- ASCII Usa *7 bits*
- Um texto é armazenado como uma cadeia de caracteres!
- Posições consecutivas da memória

A	s	s	e	m	b	l	y		r	o	c	k	s		
...	41	73	73	65	6D	62	6C	79	20	72	6F	63	6B	73	...

codificação ASCII

Representação de Caracteres

Representação de Cadeias de Caracteres (*strings*) na memória do computador:

Exemplo: “Maças Assadas”

	I	2		M	a	ç	ã	s		A	s	s	a	d	a	s		
...	31	32	20	4D	61	E7	E3	73	20	41	73	73	61	64	61	73	00	...

codificação ISO-LATIN-1

	I	2		M	a	ç	ã	s		A	s	s	a	d	a	s		
...	31	32	20	4D	61	8D	8B	73	20	41	73	73	61	64	61	73	00	...

codificação MacOSRoman

	I	2		M	a	ç		ã		s		A	s	s	a	d	a	s		
...	31	32	20	4D	61	C3	A7	C3	A3	73	20	41	73	73	61	64	61	73	00	...

codificação UTF-8

Representação de Caracteres

- No IAS, as palavras da memória possuíam 40 *bits*!
- A grande maioria das memórias de computadores atuais possuem unidades de armazenamento endereçáveis de um *byte* (8 *bits*).
- No endereço 0 cabe um dado de 1 *byte*, no endereço 1 cabe um dado de 1 *byte* e assim por diante.

Representação de Caracteres

- No IAS, as palavras da memória possuíam 40 *bits*!
- A grande maioria das memórias de computadores atuais possuem unidades de armazenamento endereçáveis de um *byte* (8 *bits*).
 - No endereço 0 cabe um dado de 1 *byte*, no endereço 1 cabe um dado de 1 *byte* e assim por diante.
- Quando armazenamos números de 7 *bits* em 1 *byte* nós desperdiçamos *bits* da memória.

Representação de Números na Memória

- Como fazemos para armazenar um número de 32 *bits* em uma memória endereçada a *byte*? Ou seja, em uma memória onde as unidades de armazenamento possuem 1 *byte*.
- Exemplo: Número de 32 *bits* (4 *bytes*)
 $1025_{10} = 00000000 \ 00000000 \ 00000100 \ 00000001_2$

Representação de Números na Memória

- Como fazemos para armazenar um número de 32 *bits* em uma memória endereçada a *byte*? Ou seja, em uma memória onde as unidades de armazenamento possuem 1 byte.
- Resposta: Depende do *Endianness*.
- Exemplo: Número de 32 *bits* (4 *bytes*)

$$1025_{10} = 00000000 \ 00000000 \ 00000100 \ 00000001_2$$

Endereço	<i>Big-Endian</i>	<i>Little-Endian</i>
00	00000000	00000001
01	00000000	00000100
02	00000100	00000000
03	00000001	00000000

Outras Referências

- Capítulo 8 do livro do Stallings.
- Capítulo 2.4 do livro do Patterson e Hennessy
- Capítulo 1 da apostila do professor Anido:

<http://www.ic.unicamp.br/~ranido/mc404/faiska/cap1.pdf>