Arquitectura de Computadores

Representação Numérica

Docente: Pedro Sobral
http://www.ufp.pt/~pmsobral

Pensamento...

"There are only 10 kinds of people in the world: those who understand binary and those who don't."

Números Decimais: Base 10

Dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9

Exemplo:

3271 =

$$(3x10^3) + (2x10^2) + (7x10^1) + (1x10^0)$$

Números: notação posicional

- Número na base B ⇒ B símbolos por dígito:
  - Base 10 (Decimal): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    Base 2 (Binária): 0, 1
- Representação Numérica:
  - $d_{31}d_{30} \ldots d_1d_0$ é um número de 32 dígitos
  - valor = $d_{31} \times B^{31} + d_{30} \times B^{30} + \ldots + d_1 \times B^1 + d_0 \times B^0$
- Binário: 0,1 ("bits" – "binary digits)
  - 0b11010 $= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$
    $= 16 + 8 + 2$
    $= 26$
  - Aqui 5 digitos binários foram convertidos em 2 decimais
  - Será que há uma base em que a conversão para binário é simples?

Números em Hexadecimal: Base 16

- Hexadecimal:
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
  - Dígitos comuns + 6 do alfabeto
  - Em C, escrevem-se 0x… (ex, 0xFAB5)
- Conversão: Binário⇔Hex.
  - 1 dígito hex. representa 16 valores decimais
  - 4 dígitos binários representam 16 valores decimais

  ⇒1 dígito hex. substitui 4 bits
- Dois símbolos Hex. Representam um "byte"
- Exemplo:
  - 1010 1100 0011 (binário) = 0x_____ ?

Decimal vs. Hexadecimal vs. Binário

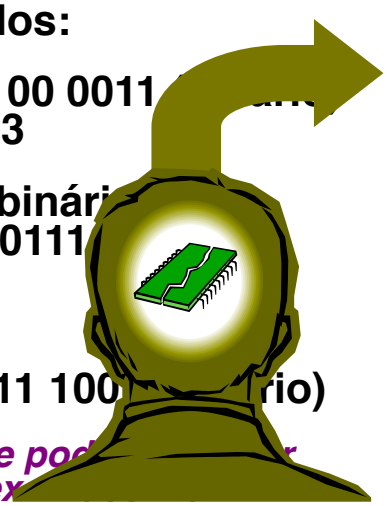
Exemplos:

1010 1100 0011
= 0xAC3

10111 (binári
= 0001 0111
= 0x17

0x3F9
= 11 1111 100

Como se pod
entre hex



00	0	0000
01	1	0001
02	2	0010
03	3	0011
04	4	0100
05	5	0101
06	6	0110
07	7	0111
08	8	1000
09	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

MEMORIZAR!

O que fazer com as representações dos números?

- O que fazemos com decimal!
  - Somar
  - Subtrair
  - Multiplicar
  - Dividir
  - Comparar
- Exemplo: 23 + 13 = 36 ; 23 – 13 = 10
  - …é tão simples adicionar números em binário que se podem construir circuítos para o fazer!
  - Subtracção: Idêntica à que se faz em decimal
  - Comparação: Como dizer que X>Y ?

Adição e Subtracção

$23_{10}=10111_2$

$13_{10}=1101_2$

11111	1
10111	10111
+ 1101	- 01101
100100	01010

- E se fôr 53+25 e 53-25?
- $(53)_{10}=(?)_{hex}$, $(25)_{10}=(?)_{hex}$
- Multiplicação e divisão na aula prática....
- ***Números Negativos !?***

Que base usar?

- Decimal: Ideal para os humanos especialmente para a aritmética…
- Hex: Se os humanos estiverem a olhar para uma longa linha de "bits" é preferível convertê-la para hex. É olhar para símbolos que representam 4 bits.
  - Terrível para a aritmética no papel…
- Binário: o que os computadores usam; vão aprender como os computadores executam: +, -, *, /
  - Representações:

    $32_{dec} == 32_{10} == 0x20 == 100000_2 == 0b100000$

Grande ideia: Bits podem representar tudo!!

- Characteres?
  - 26 letras ⇒ 5 bits ($2^5 = 32$)
  - Maiúsculas, minúsculas, pontuação, ⇒ 7 bits (em 8) ("ASCII")
  - Código normalizado que cobre todas as linguagens ⇒ 8,16,32 bits ("Unicode")
    `www.unicode.com`

- Valores Lógicos?
  - 0 ⇒ Falso, 1 ⇒ Verdade
- cores ? Ex: *Red (00)* *Green (01)* *Blue (11)*
- localizações / Endereços? comandos?
- MEMORIZAR: N bits ⇒ quando muito 2^N coisas

Como representar números negativos ?

- Até agora, números sem sinal
- Solução óbvia: sinal definido pelo bit mais à esquerda!
  - 0 ⇒ +, 1 ⇒ -
  - Os restante bits representam o valor absoluto…
- Representação: "sinal e magnitude"
- MIPS usa inteiros de 32-bit. Portanto $+1_{10}$

0000 0000 0000 0000 0000 0000 0000 0001

- E - 1_{10} ficaria:

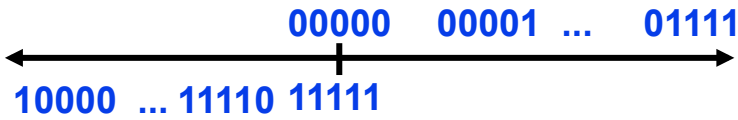
1000 0000 0000 0000 0000 0000 0000 0001

Problemas do sinal e magnitude?

- Circuíto aritmético complicado…
  - Passos especiais para operar números de sinais contrários…
- Também DOIS zeros!
  - 0x00000000 = $+0_{10}$
  - 0x80000000 = -0_{10}
  - Qual o significado de 2 zeros em programação?
- Sendo assim a representação em sinal e magnitude foi abandonada.

Outra tentativa: complementar os bits

- Exemplo: $7_{10} = 00111_2$ $-7_{10} = 11000_2$
- Chamada Complemento para Um
- Nota: números positivos começam por 0s, e os negativos por 1s.



- O que é -00000 ? Resposta: 11111
- Quantos números positivos em N bits?
- E negativos?

Problemas do Complemento para um

- A aritmética é ainda complexa.
- Continuamos com dois zeros…
  - `0x00000000` = $+0_{10}$
  - `0xFFFFFFFF` = -0_{10}
- Embora usada durante algum tempo em alguns computadores esta solução foi abandonada porque uma outra era melhor!

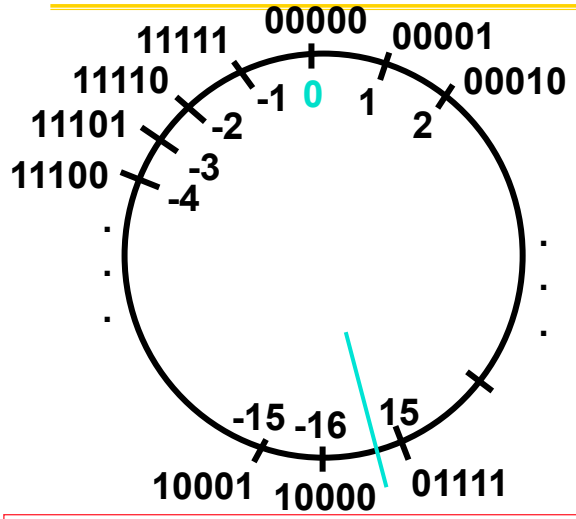
Norma para a representação de negativos

- Tal como no sinal e magnitude, se o nº começa por 0s ⇒ positivo, começa por 1s ⇒ negativo
  - 000000...xxx é ≥ 0, 111111...xxx é < 0
  - Excepto que 1…1111 é -1, e não -0

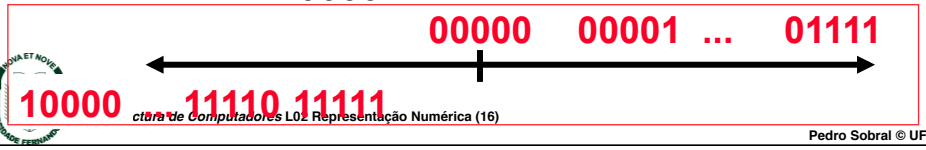
° Esta representação chama-se:

- Complemento para Dois

Complemento para 2,"N bits": N = 5



- 2^{N-1} não-negativos
- 2^{N-1} negativos
- Um zero
- Quantos positivos?



Complemento para 2: N=32

0000 ... 0000 0000 0000 0000_2 =	0_{10}
0000 ... 0000 0000 0000 0001_2 =	1_{10}
0000 ... 0000 0000 0000 0010_2 =	2_{10}
. . .	
0111 ... 1111 1111 1111 1101_2 =	$2,147,483,645_{10}$
0111 ... 1111 1111 1111 1110_2 =	$2,147,483,646_{10}$
0111 ... 1111 1111 1111 1111_2 =	$2,147,483,647_{10}$
1000 ... 0000 0000 0000 0000_2 =	$-2,147,483,648_{10}$
1000 ... 0000 0000 0000 0001_2 =	$-2,147,483,647_{10}$
1000 ... 0000 0000 0000 0010_2 =	$-2,147,483,646_{10}$
. . .	
1111 ... 1111 1111 1111 1101_2 =	-3_{10}
1111 ... 1111 1111 1111 1110_2 =	-2_{10}
1111 ... 1111 1111 1111 1111_2 =	-1_{10}

- Um zero; 1º bit chamado bit de sinal
- 1 negativo "extra" :não há $2,147,483,648_{10}$

Fórmula: Complemento para 2

- Pode representar números positivos e negativos considerando os bits e as potências de 2.

$$d_{31} \times -(2^{31}) + d_{30} \times 2^{30} + \ldots + d_2 \times 2^2 + d_1 \times 2^1 + d_0 \times 2^0$$

- Example: 1101_2

$= 1 \times -(2^3) + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$

$= -2^3 + 2^2 + 0 + 2^0$

$= -8 + 4 + 0 + 1$

$= -8 + 5$

$= -3_{10}$

Como encontrar o complemento para 2?

- Trocar todos os 0s para 1s e 1s para 0s (inverter ou complementar), depois somar 1 ao resultado.
- Prova: Soma de um número e o seu complemento (para um) tem que ser $111...111_2$

  Contudo, $111...111_2 = -1_{10}$

  Seja x' $\Rightarrow$ o complemento para um de x

  Então $x + x' = -1 \Rightarrow x + x' + 1 = 0 \Rightarrow x' + 1 = -x$

- **Exemplo: -3 para +3 de volta a -3**

  x: $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2$

  x': $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0010_2$

  +1: $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0011_2$

  ()': $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_2$

  +1: $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1101_2$

Como encontrar o complemento para 2?

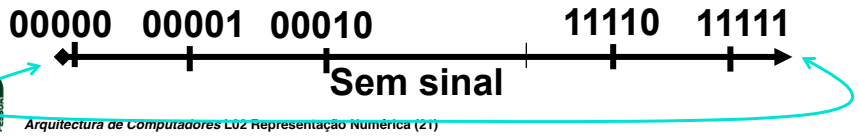
- Converter a rep. em complemento para 2 de n bits para mais do que n bits.
- Simplesmente **replicar** o bit mais significativo (de sinal) do menor para completar os bits em falta
  - compl. para 2: Positivos- infinitos 0s (à esq.)
  - compl. para 2: Negativos-infinitos 1s (à esq.)
  - A representação em binário esconde esses bits; a extensão do sinal restaura alguns…
  - 16-bit -4_{10} para 32-bit:

    $1111\ 1111\ 1111\ 1100_2$

    $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1100_2$

E se os números forem muito grandes?

- A linha bits é apenas uma representação do número. Falando estritamente, tem o nome de "numerais".
- Números têm um quantidade ∞ de digitos
  - Onde quase todos são iguais (00…0 or 11…1) excepto para alguns bits mais à direita
  - Normalmente não mostramos os bits mais à esq.
- Se o resultado de uma soma (ou -, *, /) não puder ser representado pelos bits existentes no HW…diz-se que ocorreu overflow.



Conclusão...

- Representamos "coisas" no computador como padrões de bits: N bits ⇒ 2^N
- Decimal para os humanos, binário para computadores, hex para escrever binário mais facilmente.
- Complemento para 1 - abandonado
- Complemento para 2 universal em computação: não é possível evitar, portanto tem que ser entendido!
- Overflow: números ∞; computadores finitos, erros!
- http://en.wikipedia.org/wiki/Binary_numeral_system