Aula 4 - Arquitetura de Computadores

Paulino

05-03-2020

Representação de inteiros negativos com complemento de 2

- Negue (complemento de 1) cada bit da representação positiva com o númro de bits da palavra do processador
- Adicione 1, este é o número negativo em complemento de 2 (complemento de 1+1)

Exemplos:

decimal	8 bits	16 bits
-1	11111111	1111111111111111
-7	11111001	1111111111111001
-127	10000001	1111111110000001
-130	estouro	1111111101111110

Por que usar representação em complemento de 2?

- Observe que se você complementar um número duas vezes, você obtém o número original (i.e., -(-x) = x)
- A operação de complementar um número é fácil de ser executada pelo HW
- Adicionar um número e o complemento de 2 de outro dá o mesmo resultado que subtrair o primeiro do segundo $\Longrightarrow$ não precisa de HW para subtração

Exercícios

1. Calcule usando complemento de 2 com 8 bits:

a. 15 - 7
b. -15 + 7
c. -128 - 1
d. -128 - 128
e. 127 + 1

Soluções

$$
\begin{array}{cccccccccc}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & \\
 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & (15) \\
+ & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & (-7) \\
\hline
1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & (\ \ 8)
\end{array}
$$

$$
\begin{array}{cccccccccc}
 & & & & & 1 & 1 & 1 & & \\
 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & (-15) \\
+ & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & (7) \\
\hline
0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & (-8)
\end{array}
$$

$$
\begin{array}{cccccccccc}
1 & & & & & & & & & \\
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (-128) \\
+ & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & (-1) \\
\hline
1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & (est.)
\end{array}
$$

Sol. (cont.)

$$
\begin{array}{cccccccccc}
1 & & & & & & & & & \\
 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (-128) \\
+ & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (-128) \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (est.)
\end{array}
$$

$$
\begin{array}{cccccccccc}
 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & & \\
 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & (127) \\
+ & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & (1) \\
\hline
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & (est.)
\end{array}
$$

Observações importantes

- Para o complemento de 2 ter sentido sempre precisamos saber o tamanho da palavra
- A regra para saber se houve *estouro* precisa ser revista
  - Ao adicionar dois números com sinais opostos, não há *estouro*
  - Ao adicionar dois números com o mesmo sinal, há estouro se o sinal do resultado é diferente do sinal dos operandos
  - O vai-um do bit mais significativo não é usado para calcular o *estouro*, este bit é de sinal (**lembre-se**, o bit mais a esquerda é o **sinal**)
- O maior número positivo que podemos representar com n bits é $2^{n-1} - 1$
- O menor número negativo representável com n bits é -2^{n-1}
- Se usamos uma palavra com n bits para representar apenas inteiros não negativos, os números representados estão no intervalo $[0, 2^n - 1]$
- Ao representar números negativos com complemento de 2, os números estão no intervalo $[-2^{n-1}, 2^{n-1} - 1]$

Multiplicação

- Para simplificar, vamos exemplificar a multiplicação binária com números pequenos

	2	3
x	1	3
	6	9
2	3	=
2	9	9

				1	0	1	1	1
x					1	1	0	1
				1	0	1	1	1
			0	0	0	0	0	=
		1	0	1	1	1	=	=
	1	0	1	1	1	=	=	=
1	0	0	1	0	1	0	1	1

Propriedades da multiplicação

- A multiplicação de duas palavras de **n** bits pode resultar num número com **2n** bits.
- Para somar 2 números de n bits precisamos de n somadores completos.
- Para multiplicar 2 números de n bits, precisamos de $n.(n-1) = n^2 - n$ somadores completos.
- Os primeiros processadores (1ª e 2ª gerações) não tinham HW para a multiplicação, ela era executada em SW.
- Algumas arquiteturas mais simples, até hoje, não têm multiplicação no HW. A razão não é a complexidade do HW, mas o tempo de cálculo de uma multiplicação em relação ao tempo de cálculo de operações/instruções mais simples.

Divisão Binária

- De novo, ela é como a divisão decimal. Vide um exemplo na lousa.
- Embora a maioria dos processadores modernos tenha HW de divisão, existem frequentemente, alternativas por SW que apresentam desempenho não muito inferior.

Números reais

- Existem muitos números reais que não conseguimos representar com dígitos, por exemplo:
  - dízimas periódicas: $\frac{1}{3} = 0,333... = 0,\overline{333}$
  - números irracionais: $\pi = 3,14159...$
- A representação com dígitos desses números é infinita. De modo análogo, a representação de muitos números reais na forma binária é infinita. Na verdade, como a base 2 é menor do que a base 10, existem mais dízimas periódicas em binário do que em decimal.
- Por exemplo, a representação de 0,1 na base 2 fica: $0,1_{10} = 0,000\overline{1100} \approx 2^{-4} + 2^{-5} + 2^{-8} + 2^{-9} = 0,099609375$
- Para checar que efetivamente 0,1 é uma dízima podemos fazer o seguinte teste num computador:
  - Calcule: 0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1+0.1-1.0
  - Deveria dar 0, efetivamente, se fizer: 10 * 0.1 - 1.0 dá 0

Representação em Ponto Flutuante

- Para representar números reais, em computação usamos uma representação aproximada chamada de *ponto flutuante*
- Para entender esta representação, é útil entender como funciona a *notação científica*.
- A notação científica procura separar da representação do número a **ordem de grandeza** do **valor** na grandeza.
- Os números em notação científica normalizada são sempre escritos na forma: $d_0, d_1 d_2 d_3 ... x10^{expo}$ onde d_0 está entre 1 e 9, os outros dígitos estão entre 0 e 9
- Exemplos:
  - $2020 = 2,02.10^3$
  - $0{,}0004501 = 4,501.10^{-4}$
  - $3{,}14159 = 3,14159.10^0$
  - $0,1_{10} = 1,0.10^{-1} = 0,000110011001100_2... = (1,10011001100...)_2.2^{-4}$

Ponto Flutuante (norma IEEE 754)

- Observe que na base 2, o d_0 é sempre 1, por isso, na representação de ponto flutuante este 1 não é colocado.
- Existem diferentes tamanhos de ponto flutuante (PF), as mais usadas são: 32 bits e 64 bits. Em geral, o PF de 32 bits é o *tipo de dado* **float**, o PF de 64 bits é o **double**.
- No float, o primeiro bit, o mais a esquerda, é o bit de sinal, os 23 bits seguintes correspondem aos bits após a vírgula, os últimos 8 bits representam o expoente em complemento de 2.
- No double $\rightarrow$ *procure na Internet*
- A precisão da representação é:
  - float: 6 a 8 dígitos decimais dependendo do valor do expoente
  - double: 13 a 16 dígitos