Chapter 2: Instructions: Language of the Computer

2.4. Signed and Unsigned Numbers

Arquitetura e Organização de Computadores

Introdução: bases numéricas

- Base de um sistema numérico:
  - Quantidade de algarismos disponíveis para representar todos os números
    - Hexadecimal: 16 algarismos
      - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
    - Decimal: 10 algarismos
      - 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
    - Octal: 8 algarismos
      - 0, 1, 2, 3, 4, 5, 6, 7
    - Binário: 2 algarismos
      - 0, 1
- **Questão 01**: Computadores trabalham com base 2. Por quê?

Introdução: bases numéricas

- Bit menos significativo:
  - É o mais à “direita”, cujo expoente é 0 (zero)
- Bit mais significativo:
  - É o mais à “esquerda”, cujo expoente é i
- Em geral, em qualquer base, o valor do i-ésimo algarismo:
  - d_i x base^i
    - i aumenta da direita para esquerda, começando com 0 (zero)
- Em um número com **n** algarismos e base **b**:
  - Total de números: b^n
  - Amplitude: de 0 até b^n - 1

$$x = \sum_{i=n-1}^{0} x_i \times b^i = x_{n-1} \times b^{n-1} + x_{n-2} \times b^{n-2} + \cdots + x_1 \times b^1 + x_0 \times b^0$$

Introdução: bases numéricas

- Quando trabalhamos com bases diferentes, costumamos colocar um número subscrito que indica a base. Por exemplo:
  - 1011_2
  - 1011_8
  - 1011_{10}
  - 1011_{16}

- **Questão 02**: Calcule o valor em decimal dos números acima.

- Transformação entre bases:
  - Algumas podem ser feitas diretamente (não todas)
  - Na dúvida, transforme para base decimal e, só depois, para a base desejada

Introdução: bases numéricas

- Em RISC-V, com palavras de 32 bits, temos:

$$
\begin{aligned}
&00000000\ 00000000\ 00000000\ 00000000_{two} = 0_{ten}\\
&00000000\ 00000000\ 00000000\ 00000001_{two} = 1_{ten}\\
&00000000\ 00000000\ 00000000\ 00000010_{two} = 2_{ten}\\
&\ldots \qquad\qquad\qquad \ldots\\
&11111111\ 11111111\ 11111111\ 11111101_{two} = 4{,}294{,}967{,}293_{ten}\\
&11111111\ 11111111\ 11111111\ 11111110_{two} = 4{,}294{,}967{,}294_{ten}\\
&11111111\ 11111111\ 11111111\ 11111111_{two} = 4{,}294{,}967{,}295_{ten}
\end{aligned}
$$

Introdução: E os números negativos?

- A representação numérica em base binária que vimos até agora é perfeita para números positivos e para o zero. Mas e os números negativos?

- **Questão 03**: como os números negativos podem ser representados em base binária?

Introdução: E os números negativos?

- Solução inicial: usar um bit **extra** para representar o sinal
  - Esse bit extra é chamado de SINAL-MAGNITUDE
- Problemas com o bit de sinal:
  - Onde colocar?
    - 1100110
    - 1001101
  - Hardware para realizar somas é mais complicado, pois uma soma pode resultar em números positivos ou negativos
  - Teremos 2 representações para o número 0 (zero), um zero positivo e um zero negativo:
    - 10000000
    - 00000001

Introdução: E os números negativos?

- Solução escolhida: **complemento de dois**
  - **Não usar** um bit extra
  - Usar o bit mais significativo, chamado de SINAL:
    - Se for **0**: o número é positivo
    - Se for **1**: o número é negativo

$$
\begin{aligned}
00000000\ 00000000\ 00000000\ 00000000_{two} &= 0_{ten}\\
00000000\ 00000000\ 00000000\ 00000001_{two} &= 1_{ten}\\
00000000\ 00000000\ 00000000\ 00000010_{two} &= 2_{ten}\\
\ldots \qquad\qquad\qquad\qquad\qquad\qquad\quad & \ldots\\
01111111\ 11111111\ 11111111\ 11111101_{two} &= 2{,}147{,}483{,}645_{ten}\\
01111111\ 11111111\ 11111111\ 11111110_{two} &= 2{,}147{,}483{,}646_{ten}\\
01111111\ 11111111\ 11111111\ 11111111_{two} &= 2{,}147{,}483{,}647_{ten}\\
10000000\ 00000000\ 00000000\ 00000000_{two} &= -\ 2{,}147{,}483{,}648_{ten}\\
10000000\ 00000000\ 00000000\ 00000001_{two} &= -\ 2{,}147{,}483{,}647_{ten}\\
10000000\ 00000000\ 00000000\ 00000010_{two} &= -\ 2{,}147{,}483{,}646_{ten}\\
\ldots \qquad\qquad\qquad\qquad\qquad\qquad\quad & \ldots\\
11111111\ 11111111\ 11111111\ 11111101two &= -\ 3_{ten}\\
11111111\ 11111111\ 11111111\ 11111110_{two} &= -\ 2_{ten}\\
11111111\ 11111111\ 11111111\ 11111111_{two} &= -\ 1_{ten}
\end{aligned}
$$

Introdução: E os números negativos?

- Cuidados ao usar **complemento de dois**
  - O menor número "positivo" é zero:
    - 00000000 00000000 00000000 00000000
  - O maior número positivo ($2^{31} - 1$) é 2.147.483.647:
    - 01111111 11111111 11111111 11111111
  - O menor número negativo (-2^{31}) é -2.147.483.648:
    - 10000000 00000000 00000000 00000000
  - O maior número negativo (-2^{0}) é -1:
    - 11111111 11111111 11111111 11111111
  - Um dos números negativos não tem correspondente positivo:
    - -2.147.483.648
- É complicado mas, por incrível que pareça, isso simplifica o hardware
- Amplitude: (-2^{n-1}) até $(2^{n-1} - 1)$

Introdução: E os números negativos?

- Como saber qual número estamos querendo dizer quando usamos complemento de dois?
  - Igual antes, mas usando base negativa no bit mais significativo, e bases positivos no resto:

$$x = (x_{n-1} \times -b^{n-1}) + \left(\sum_{i=n-2}^{0} x_i \times b^i \right)$$
$$= (x_{n-1} \times -b^{n-1}) + (x_{n-2} \times b^{n-2} + x_{n-3} \times b^{n-3} + \cdots + x_1 \times b^1 + x_0 \times b^0)$$

- **Questão 04**: que números são esses? (é para ralar mesmo...)
  - $10000000\ 11111111\ 10101010\ 01010101_2$
  - $11111111\ 11111111\ 11111111\ 11111100_2$
  - $10101010\ 10101010\ 10101010\ 10101010_2$

Introdução: E os números negativos?

- Números sem sinal:
  - Unsigned
- Números com sinal:
  - Signed

- O RISC-V tem duas instruções para tratar essas diferenças:
  - **`lbu`** (load byte unsigned)
    - Preenche à esquerda com zeros
  - **`lb`** (load byte)
    - Mantém tudo corretamente com o sinal

- Endereçamento de memória não tem números negativos

Introdução: E os números negativos?

- Algumas linguagens de programação são explícitas (C):

Data Types	Memory Size	Range
char	1 byte	−128 to 127
signed char	1 byte	−128 to 127
unsigned char	1 byte	0 to 255
short	2 byte	−32,768 to 32,767
signed short	2 byte	−32,768 to 32,767
unsigned short	2 byte	0 to 65,535
int	2 byte	−32,768 to 32,767
signed int	2 byte	−32,768 to 32,767
unsigned int	2 byte	0 to 65,535
short int	2 byte	−32,768 to 32,767
signed short int	2 byte	−32,768 to 32,767
unsigned short int	2 byte	0 to 65,535
long int	4 byte	-2,147,483,648 to 2,147,483,647
signed long int	4 byte	-2,147,483,648 to 2,147,483,647
unsigned long int	4 byte	0 to 4,294,967,295
float	4 byte	
double	8 byte	
long double	10 byte	

Introdução: E os números negativos?

- **Questão 05**: represente os seguintes números em complemento de dois, com palavras binárias de 8 bits
  - -12_{10}
  - -25_{10}
  - -77_{10}
  - -120_{10}

- Tem algum macete?
  - Claro, tem 2! Já vamos ver, primeiro você tem que roer o osso...

Introdução: E os números negativos?

- "1º Macete" para trabalhar com complemento de dois:
  - Considere uma palavra de 4 bits, com sinal:
    - Considere um número qualquer: 0111_2
    - Faça a **inversão** de todos bits: 1000_2
    - Faça a soma desses números:

$$\begin{array}{rr} & 0111_2 \\ + & 1000_2 \\ \hline = & 1111_2 \end{array}$$

    - Agora perceba: $\mathbf{1111_2 = -1_{10}}$! Então:

$$x + \overline{x} = -1$$

$$x + \overline{x} + 1 = 0$$

$$\overline{x} + 1 = -x$$

Introdução: E os números negativos?

- **Questão 06**: Transforme 2_{10} em 2_2, usando palavra de 32 bits. Depois ache a representação em complemento de dois do número -2_{10}. Ao final, faça uma verificação de que tudo está correndo, achando a representação binária (em 32 bits) do número $-(-2_{10})$.

Introdução: E os números negativos?

- "2º Macete" para trabalhar com complemento de dois:
  - Como estender um número com n bits para um maior?
    - Basta pegar o bit de sinal e replicar até preencher!
    - Exemplo: -2_{10}

      Palavra com 16 bits: 11111111 11111110
      Palavra com 32 bits: _____ _____ 11111111 11111110
      Palavra com 32 bits: 11111111 11111111 11111111 11111110

Você entendeu?

- **Questão 07**: Calcule o seguinte:

  a) Qual é o valor decimal deste número em complemento de dois (palavra com 64 bits)?

  ```
  11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111111 11111000
  ```

  b) Se o número binário acima fosse um número unsigned de 64 bits, qual seu valor decimal?

Observações

- Existem outras formas de representar números negativos, mas não são muito usadas e não vamos ver
  - Complemento de um
  - Notação enviesada

- Até agora só vimos a representação de números inteiros. Números de ponto flutuante (com vírgula) são outros quinhentos...

- Se você estiver com dúvidas em transformações de bases, treine! É importante que, pelo menos, você saiba transformar entre binário, decimal e hexadecimal.

Hora de Esfriar a Cabeça!