Circuitos Lógicos
Aula 5

Aula passada

- Sistemas numéricos
- Metodo de conversão
- Conversão entre sistemas
- Números fracionários

Aula de hoje

- Conversão fracionária
- Método da multiplicação
- Código BCD
- Código ASCII
- Exercício da lista

Representação Fracionária

- Qualquer valor fracionário (F < 1) pode ser representado na base 10 como
- $F = A_{-1} \times 10^{-1} + A_{-2} \times 10^{-2} + \ldots$
- Sequência para representar F pode ser finita ou infinita
- Ex. 3/5 ?   1/3 ?

- Generalização: qualquer valor fracionário (F < 1) na pode ser representado em qualquer base como
- $F = A_{-1} \times B^{-1} + A_{-2} \times B^{-2} + \ldots$
- Sequência finita ou infinita depende da base
- Ex. 1/3 na base 3? 3/5 na base 2?

Representação Fracionária

- Computador (e nós também) usa número finito de dígitos para representar valores
- Logo, representação de um valor pode não ser exata, se sequência for infinita
- Ex: valor 1/3 na base 10 usando 5 dígitos. Qual é o erro cometido?
  - Erro = valor real - valor representado
  - E = 1/3 - 0.33333 = 0.000003333...
- Ex. 1/10 na base 2 usando 5 dígitos
  - E = 1/10 – 0.00011_2 = 1/10 - 0.09375 = 0.00625

Método para Conversão Fracionária

B $\longrightarrow$ **10**

- Como converter um número fracionário de uma base B qualquer para a base 10?
- Fácil! Escrever o valor do número como soma de potências
- Ex. $0.AB1_{16}$? 0.56_7? 0.0101_2 ?
- $F = A_{-1} \times B^{-1} + A_{-2} \times B^{-2} + \ldots$

Método para Conversão Fracionária

10 $\longrightarrow$ **B**

- Como converter um número fracionário da base 10 para uma base B?
- Ideias?
- O que acontece quando multiplicamos um número fracionário por sua base?
- Ex. 0.3214 * 10 = 3.214

Primeiro dígito depois do .

- Como obter o segundo dígito? E o terceiro?

Multiplicações Sucessivas

- Ex. 0.4896

```
.4896 * 10 = 4.896; dígito ------------------------------> 4
 ^            ^
              .896 * 10 = 8.96; dígito ---------------------> 8
               ^          ^
                           .96 * 10 = 9.6; dígito --------> 9
                            ^         ^
                                       .6 * 10 = 6.0; ----> 6
                                        ^        ^
```

- Multiplicar por 10; parte inteira nos dá o próximo dígito fracionário;
- Usar parte fracionária na próxima multiplicação; parar quando valor for zero

Multiplicações Sucessivas

- Mesmo princípio funciona para qualquer base B
- Multiplicador é a base B, pois sempre teremos um dígito inteiro entre 0 e B-1
- Ex. 1/10 = 0.1

```
.1 * 2 = 0.2; dígito ---------------------------------------------> 0
         ^
          .2 * 2 = 0.4; dígito -----------------------------------> 0
                   ^
                    .4 * 2 = 0.8; dígito -------------------------> 0
                             ^
                              .8 * 2 = 1.6; ----------------------> 1
                                       ^
                                        .6 * 2 = 1.2; ------------> 1
                                                 ^
                                                  .2 * 2 = 0.4 ---> 0
                                                           ^
```

Formalizando o Método

- **Vamos mostrar que o método funciona**
- Seja F um valor fracionário qualquer, então
  - $F = A_{-1} \times 2^{-1} + A_{-2} \times 2^{-2} + \ldots$
- Como obtivemos os dígitos A_{-1}, A_{-2}, ... ?
- Multiplicando os dois lados por 2 temos
- $2F = A_{-1} + A_{-2} \times 2^{-1} + A_{-3} \times 2^{-2} + \ldots$
- Substraindo A_{-1} e multiplicando os dois lados por 2 temos
- $2(2F – A_{-1}) = A_{-2} + A_{-3} \times 2^{-1} + \ldots$
- E assim por diante. Ao final do metodo temos
- $2(\ldots 2(2F - A_{-1}) - A_{-2})\ldots) - A_{-k} = 0$
- Desenvolvendo, temos
- $2^{k} F - 2^{k-1} A_{-1} - 2^{k-2} - A_{-2} - \ldots - A_{-k} =0$
- Divindo tudo por 2^{k} e passando para o outro lado, obtemos F

Código BCD

- Código: representação de símbolos com números (binários)
  - Ex. letras do alfabeto
- BCD: Binary Coded Decimal
- Ideia: Representar dígitos decimais com um número binário
- Quantos dígitos binários tem o código?
- 4 bits são necessários, pois temos 10 símbolos (0,1,...,9)

Decimal:	0	1	2	3	4	5	6	7	8	9
BCD:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Código BCD

- Cada dígito decimal é codificado independentemente
- Ex. 183 em BCD?
- Ex. 0001 0010 0111 em decimal?
- Diferente da representação do número em binário
- Razão para este código?
- Vantagens: tamanho fixo, fácil conversão, facilita cálculos em decimal
- Desvantagens: mais bits, maior complexidade no circuito para implementar operações matemáticas
- Ainda muito usado em circuitos digitais

Código ASCII

- Código alphanumérico: representa letras maiúsculas e minúsculas, números, caracteres de pontuação, alguns sinais frequentes, etc.
- ASCII (American Standard Code for Information Interexchange) é o mais famoso e mais usado código alphanumérico
- 7 bits = 128 símbolos (suficiente?)
- Usado para transmitir informação entre computador e periféricos
  - disco, teclado, impressora, outro computador (ssh), etc

Código ASCII - Exemplo

- Código ASCII da frase “Opa!”
- O = 79 = 01001111_2
- p = 112 = 01110000_2
- a = 97 = 01100001_2
- ! = 33 = 00100001_2
- Apersar de ter 7 bits, computador usa 8 bits para armazenar o código
  - memória funciona em múltiplos de bytes

Outros Códigos

- Existem muitos outros códigos alphanuméricos
- Representar letras e caracteres em outros idiomas
- Representar símbolos matemáticos
- UTF: família de códigos para *unicode*
  - UTF-8 vem sendo muito usado (8 bits)
- Muitas variações do ASCII (ISO-xxxx)
- Até hoje ainda sofremos com uma falta de padrão universal
  - Ex. Texto no browser aparece errado, etc.