Instituto Federal Sudeste de Minas Gerais – Campus Barbacena
Curso de Tecnologia em Sistemas para Internet
Disciplina: Lógica de Programação

Prof.: Wender Magno Cota

Exercício Prático

Como computadores são elementos binários, a forma mais eficiente de representar números deve ser a binária, isto é, converter o número diretamente de decimal para o seu correspondente valor binário.
Ao trabalhar em computação com valores binários deve-se leva em consideração três fatores que podem acarretar inconvenientes no projeto e na utilização da máquina.

- a representação do sinal do número
- a representação da vírgula (ou ponto) que separa a parte inteira da parte fracionária de um número não inteiro
- a quantidade limite de algarismos possível de ser processada pelo processador.

O primeiro problema é resolvido com o acréscimo de mais um bit na representação do número. Este bit indica o sinal do número. Por convenção adota-se 0 para valor positivo e 1 para valor negativo.

O segundo problema reside na forma de representação de números fracionários. Isso devido a dificuldade de se representar a vírgula internamente. A solução para este problema pode ser encontrada na escolha entre dois métodos de representação e de realização de operações aritméticas:

- representação em ponto fixo
- representação em ponto flutuante

Na matemática real a quantidade de números é infinita. No entanto, computadores são máquinas de tamanho finito, podendo assim, representar uma quantidade finita de números.

Tipos de representação de inteiros

- Sinal e magnitude
- Complemento a base – 1
- Complemento a base

Sinal e Magnitude
A representação de números com n algarismos(n bits) é obtida usando-se 1 bit, normalmente o mais à esquerda, para representar o sinal do número, e os n-1 bits restantes para indicarem a magnitude. A magnitude possui a mesma representação tanto para números positivos como para números negativos.

Faixa de representação
$-(2^{n-1} – 1)$ a $+(2^{n-1} -1)$
Exemplo usando-se 10 bits

0000110110 = 54

0000110110= -54

Desvantagens:

a) Duas representações para o zero
b) Necessidade de dois circuitos somadores

Aritmética em complemento de 2

Devido as vantagens, principalmente por necessitar de um único circuito somador, a representação em complemento a 2, os sistemas de computação a empregam de forma generalizada. O conceito de complemento é válido para qualquer base de numeração(B). Há dois tipos de complementos, sendo else, complemento à base e complemento à base menos 1. Considerando a base 2 temos o complemento a 2 e o complemento a 1.

Complemento à base

Em operações aritméticas, o complemento à base de um número N é o valor que falta para se obter B^n, ou seja:

complemento à base de N= B^n – N, onde n é a quantidade de algaristmos utilizados na operação e N é o valor numérico.

Exemplo:

Considerando-se a base 2 e usando 5 algarismos(bits) temos

N= 110 Complemento de 2 de N = 2^5 – N= 100000 – 110 = 11010

Faixa de representação

-2^{n-1} a $+(2^{n-1} -1)$

Aritmética em complemento 2

A aritmética em complemento a 2 é necessário apenas um componente para somar dois números e um componente que realize a operação de complementação. O algoritmo básico refere-se, então, à soma dos números, considerando-se que os números negativos estejam representados em complemento a 2; ele acusa, também, overflow se o resultado ultrapassar a quantidade de bits representáveis.

Algoritmo de adição em complemento a 2

a) Somar os dois números, bit a bit, inclusive o bit de sinal
b) Desprezar o último vai 1(para fora do número), se houver.
c) Se, simultaneamente, ocorrer vai 1 para o bit de sinal e vai 1 para fora do número, ou se ambos não ocorrerem, o resultado está correto.
d)Se ocorrer apenas um dos vai 1, o resultado está incorreto. Ocorreu overflow. O overflow somente pode ocorrer se ambos os números tiverem o mesmo sinal e, nesse, caso, se o sinal do resultado for oposto ao dos números.

Algoritmo de subtração em complemento a 2

a) Complementar a 2 o subtraendo, independentemente se é um valor positivo ou negativo.
Somar ambos os números, utilizando o algoritmo da adição.

Exemplos: Números representados por 4 bits

a) Somar os números 1100 com 1101. Os números são negativos e estão representados em complento a 2. Para sabermos seu valor decimal precisamos convertê-los para a representação em sinal e magnitude. Neste caso, teremos que trocar o valor dos bits da magnitude e somar 1 ao resultado, ou seja

1100, amagnitude é 100. invertendo os bits temos 011. Somando-se 1 temos 100
1100 = -4
1101= -3

```
   11        vai 1 para fora do número é desprezado
    1100
    1101
    1001  +
```

b) Somar 13 + 15 usando 6 bits para representação
13 = 001101
15 = 001111 +
011100 = 28

c) 23 + 20
23 = 010111
20 = 010100 +
111001

Ocorreu overflow

Complemento à base -1

Complemento à base menos de N é
$(2^n - N) - 1$, onde n é a quantidade de algarismos do número.

Exemplo com 4 bits
Complemento de um de 01110 = 10001

O objetivo exercício prático é implementar um somador binário usando a representação de complemento de 2. Seu programa deve ler vários conjuntos de 2 números na base 10 e representar cada um em complemento de dois usando 16 bits. Solicitar ao usuário que escolha operações(soma ou subtração) a serem para cada par de valor e apresentar o reultado em binário e em decimal.
Obs.

a) Detectar overflow;
b) As operações de soma e subtração devem ser realizadas com os números em binário(representados em complement de dois).
c) O program deve ser devidamente modularizado(uso correto de funções).
d) Usar apenas funções das bibliotecas C padrão ANSI.
e) O programa deve compilar sem nenhuma warning.

Data de Entrega: 11/07/2021