Calculadora binária de sinal magnitude

Felipe A. Chatalov, RA: 118992

Departamento de Informática – Universidade Estadual de Maringá (UEM)
Maringá – PR – Brazil

ra118992@uem.br

1. Introdução

Neste relatório será apresentado e explicado em detalhes o funcionamento das operações de soma, subtração, divisão e multiplicação utilizando o sinal magnitude como modo de representar números em binário. Além de uma breve explicação sobre como foi realizado as funções e operações.

2. Objetivos

Este trabalho tem como objetivo, mostrar o funcionamento e execução passo a passo de contas matemáticas básicas (soma, subtração, divisão e multiplicação) de números inteiros, com 16 bits de capacidade, utilizando do sinal magnitude como forma de representar os números binários e usando o mesmo para realizar todas as operações.

3. Representação em sinal magnitude

A representação em sinal magnitude funciona da seguinte maneira: reservamos o primeiro bit a esquerda (mais significativo) para ser usado como bit de sinal (0 para positivo e 1 para negativo) e deixamos os outros bits como bits para o próprio número em si, seguindo a ordem convencional de números binários, por exemplo: em um número de 8 bits queremos representar o inteiro 13 e o -15, em sinal magnitude temos que 13 = 00001101 e -15 = 10001111.

3.1 Operação de soma

Para a operação de soma ser feita utilizando sinal magnitude precisamos separar o sinal do número, realizar a operação no número e após isso, identificar qual sinal ficará o resultado e por fim, juntar o sinal com o resultado.

Figura 1. Tabela verdade da operação XOR

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Tendo em mente a tabela verdade da operação XOR, a conta segue a seguinte: pegamos o último bit de cada número e fazemos XOR para saber o resultado, caso ambos bits estejam em 1, temos um carryout, ele é um bit extra que segue para a próxima operação, logo temos, o penúltimo bit de ambos os números mais o carryout da operação passada, que chega como carryin, assim faremos (n1 XOR n2 XOR carryin), caso 2 ou mais bits estejam ativos geramos um novo carryout para a próxima operação, isto segue até o 2 bit mais à esquerda do número, já que o primeiro se trata do sinal, logo nao entra na operação. Por fim temos que calcular o sinal que a soma resultará no final, para isso verificamos o primeiro bit de ambos os números, caso eles sejam diferentes, ou seja, um positivo e o outro negativo, fazemos o jogo de sinal e vemos que na verdade estamos trabalhando com uma subtração, logo não se encaixa no escopo deste algoritmo. Caso os 2 bits sejam iguais, ambos positivos ou negativos, basta colocar como resultado o mesmo sinal que algum deles, por padrão passamos o sinal do primeiro número para o resultado.

3.2 Operação de subtração

Para a operação de subtração ser feita utilizando sinal magnitude utilizamos da mesma porta lógica XOR apresentada na operação da soma, e sua tabela verdade na Figura 1. Porém, diferente da soma, que recebe o carryin caso 2 ou mais bits estejam ativos, na operação de subtração o carryin da operação depende da seguinte fórmula: carryin recebe 1 se n1 - (n2 + carryin) $<$ 0, caso contrário, carryin recebe 0. Lembrando que na primeira iteração, no bit mais à direita, o carryin começa com valor 0. Para o bit resultante fazemos como na soma e apenas passamos os 2 bits + carryin para a porta lógica XOR.

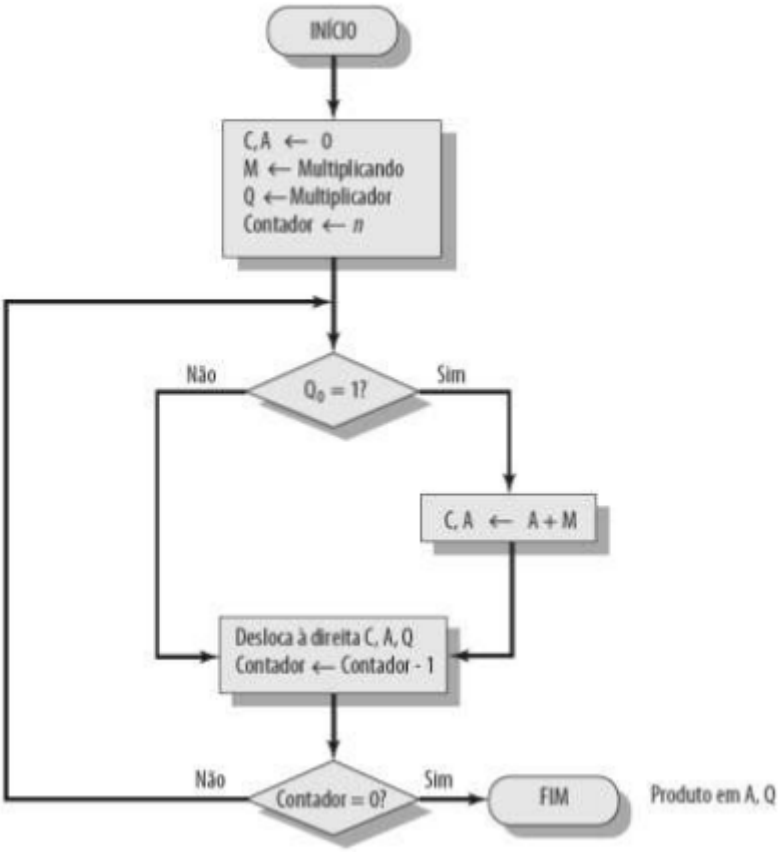
Do mesmo jeito que a operação de soma, precisamos calcular o sinal separado da conta do número resultante. Para calcular o sinal caso ambos sejam diferentes entre si, estamos na verdade falando de uma soma, logo, a conta não entra no escopo deste algoritmo e passa os bits para a operação de soma calcular o resultado final. Caso os bits sejam iguais temos então uma operação de subtração genuína, então precisamos encontrar qual dos 2 números passados é maior em valor absoluto(ignorando o sinal) do que o outro, já que a operação de soma depende de o maior número absoluto vir primeiro e logo após o menor número absoluto, após aplicá los de maneira correta verificamos o sinal, caso os números passados já estejam em ordem(maior primeiro e menor em seguida) o sinal do resultado acaba sendo o sinal do primeiro número. Caso precise inverter os números então o sinal acaba ficando como o inverso do primeiro número. Após isso, juntamos o sinal com o resultado da operação e temos o resultado final da operação de subtração.

3.3 Operação de divisão

Para a implementação da operação de divisão, utilizamos um algoritmo simples para isso, que consiste em executar subtrações no dividendo enquanto o dividendo for maior que o divisor em magnitude, a cada subtração incrementamos o valor do quociente que será o resultado final da nossa divisão e o resto ficará guardado no próprio dividendo que será retornado no final junto com o quociente. Após executar o algoritmo de divisão precisamos calcular qual o sinal do número final, tanto do quociente quanto do resto, para isso faremos o seguinte, o sinal do quociente ao final é simples, positivo caso o sinal dos 2 operandos sejam iguais ou negativo caso os 2 operandos tenham sinais opostos. Para o sinal do resto da divisão, ele será igual ao sinal do dividendo, ou 1 operando.

3.4 Operação de multiplicação

Figura 2. Fluxograma para algoritmo de multiplicação



Já para a implementação da operação de multiplicação utilizamos o algoritmo da Figura 2 acima, que, por meio de deslocamentos e somas chega ao resultado com menos iterações que fazer várias somas subsequentes. O algoritmo funciona da seguinte forma: o contador recebe o número de bits dos operandos, C e A recebem 0 na primeira iteração, então verificamos o último bit, mais à direita do multiplicador, caso seja igual a 1 C e A recebe a soma de A + M, caso aconteça overflow ele fica guardado em C, enquanto A recebe a soma, após isso, deslocamos C A e Q um bit à direita, para isso consideramos que C A e Q são 1 vetor de bits, ou seja, o número que sai de A entra em Q e incrementamos o contador, caso Q0 seja 0 apenas deslocamos e incrementamos 1 do contador, repetimos o mesmo processo até que contador seja igual a 0. Ao final teremos o produto em A e Q juntos, ou seja, o produto final tem o 2*N bits, sendo N o número de bits dos operandos iniciais. Agora que temos o resultado basta verificar o sinal que, assim como na divisão, basta verificar a igualdade dos sinais dos operandos, multiplicando e multiplicador, caso sejam iguais, o resultado sai positivo, caso sejam diferentes, temos um resultado negativo.

4. Decisões de projeto para implementação

Para o projeto de uma calculadora com funções de adição, subtração, divisão e multiplicação foi utilizado a linguagem Python versão 3.8. Para a representação de números binário foi utilizado Sinal Magnitude e strings em Python para representar os bits, por exemplo, 0010 se refere a 2 e 1011 se refere a -3. As funções de operação recebem o número como um string de bits desde de o começo e todas as operações e verificações seguintes são feitas utilizando bit a bit dos operandos. Também foi criado funções para transformar de binário para decimal e vice-versa, visando transformar o input do usuário para binário e também para mostrar a resposta na tela de maneira mais apresentável. Os números utilizados possuem valor máximo e mínimo de +32767 a -32767 pois foram feitos com 16 bits, sendo 1 deles para sinal.

Já para o uso do programa, foi feito de modo em que ao executar o programa, o usuário digite uma operação para realizar, neste caso, se trata de uma soma, subtração, multiplicação ou divisão com apenas 2 operandos, por exemplo: ‘4 + -2’ ou ‘-2 * -8’, como mostrado, caso o usuário deseje usar um número negativo basta colocar ‘-’ diretamente antes do número desejado, após isso o programa irá mostrar o passo a passo da operação e retornar na tela o resultado. Após a conclusão da operação o programa irá perguntar a próxima conta que o usuário deseja realizar, caso ele queira sair do programa basta digitar ‘quit’ que o programa finaliza.

5. Conclusão

Com a conclusão deste trabalho foi possível aprender mais sobre a forma de representação numeral sinal magnitude e também compreender melhor o funcionamento de algoritmos de soma, subtração, multiplicação e divisão usados com esta mesma representação numérica.

6. Referências

Stallings, William; Arquitetura e organização de computadores; 8a edição, São Paulo;

Pearson Pratice Hall, 2010