


Trabalho Prático - Simulador NEANDER

Este trabalho está dividido em 3 partes e terão suas entregas realizadas em grupo em tempos distintos para cada uma dessas partes.

Parte 1: Analisar o programa escrito com mnemônicos do Neander sob as condições de testes estabelecidas explicando a ação que o programa executa e qual é o algoritmo utilizado para tal ação. Fazer uso de texto e fluxogramas para tais explicações.

Memória de Programa	
PC	End.
0x00	0x20
0x01	0x2A
0x02	0x10
0x03	0x82
0x04	0x20
0x05	0x80
0x06	0xA0
0x07	0x27
0x08	0x20
0x09	0x81
0x0A	0xA0
0x0B	0x27
0x0C	0x20
0x0D	0x80
0x0E	0x10
0x0F	0x83
0x10	0x20
0x11	0x81
0x12	0x60
0x13	0x30
0x14	0x28
0x15	0x10
0x16	0x29
0x17	0x20
0x18	0x83
0x19	0x30
0x1A	0x29
0x1B	0x90
0x1C	0x27
0x1D	0x10
0x1E	0x83
0x1F	0x20
0x20	0x82
0x21	0x30
0x22	0x28
0x23	0x10
0x24	0x82
0x25	0x80
0x26	0x17
0x27	0xF0
0x28	0x01
0x29	0x00
0x2A	0x00

Memória de Dados	
PC	End.
0x80	0x0B
0x81	0x01
0x82	0x0B
0x83	0x00



Estado da memória após o término do programa

Parte 2: Escrever um programa para o simulador Neander que implemente as operações aritméticas básicas de adição e subtração sobre números inteiros positivos, representados como inteiros com sinal (representados na faixa entre -128 e +127, considerando o zero um número positivo). Estas operações devem ser utilizadas para calcular o valor da expressão:

$$x = (a + b) - c$$

Para a definição das variáveis devem ser obrigatoriamente utilizadas as seguintes posições de memória:

- Endereço 0x80: variável a ;
- Endereço 0x81: variável b ;
- Endereço 0x82: variável c ;
- Endereço 0x83: sinal da variável x (0 se + e 1 se -);
- Endereço 0x84: variável x .

A operação deve ser obrigatoriamente realizada com números naturais dentro do espaço de representação citado anteriormente. Dessa forma todas as variáveis de entrada e a de resultado representam números inteiros de 1 *byte*.

Dicas:

1. O Neander não possui operação de subtração. Entretanto, uma subtração pode ser transformada em uma soma através do complemento do subtraendo
2. Faça teste de verificação se as variáveis de entrada são realmente números reais (inteiros e > 0). Caso alguma entrada viole essa regra o programa deve ser encerrado (entrar em modo *halt*).

Parte 3: Com base na funcionalidade do programa pedido para ser realizado na Parte 2 realize as seguintes ações:

1. Crie o fluxograma que define o algoritmo implementado pelo grupo;
2. Identifique os pontos de otimização que podem ser implementados com reaproveitamento de *software*;
3. Implemente os reaproveitamentos de *software*;
4. Apresente o novo fluxograma;
5. Analise se esta ação promoveu a otimização no nível esperado.