RELATÓRIO – RENAHN E RAFAELLA

Programação assembly ARM e extensão de instrução

de processador ARM monociclo

O presente trabalho consistiu basicamente em três principais etapas, sendo:

1) Estudo das instruções já suportadas pelo processador ARM monociclo (ADD, SUB, ORR, AND, LDR, STR E B), assim como, do processamento das mesmas;

2) Desenvolvimento de um programa assembly para calcular uma progressão geométrica com razão 2, numa sequência de 10 números;

3) Estudo e implementação das seguintes instruções no processador ARM: TST, CMP E LSL;

Das três etapas realizadas neste trabalho, a etapa 3 foi a que requereu maior tempo de dedicação uma vez que foi a que apresentou as maiores dificuldades na execução do projeto. Portanto, será aqui relatado os problemas apresentados tais como as soluções encontradas para os mesmos, descrevendo as modificações realizadas ao longo do código de implementação.

Tendo em vista que o processador ARM já tem o suporte para as instruções básicas citadas na etapa 1, a implementação de TST E CMP não foi ao todo tão complexa, uma vez que a instrução TST utiliza a ADD, enquanto a CMP faz uso da SUB. A principal diferença entre ADD, SUB, ORR e AND para as implementas TST e CMP é que, enquanto as quatro primeiras realizam a instrução e jogam no registrador, as duas últimas realizam a instrução e atualizam as flags da ALU, escrevendo absolutamente nada no registrador. Para resolver esta diferença entra as instruções foi criada uma variável chamada de NoWrite, no módulo ARM do código, responsável por identificar essa atualização ou não das flags da ALU. No módulo decoder, essa variável recebe o valor de 1, no caso das instruções TST e CMP e depois é analisada no módulo condlogic.

Diferentemente das instruções TST e CMP, a última instrução implementada não faz uso de nenhuma outra já suportada pelo processador ARM. Apresentando, portanto, uma maior complexidade em sua implementação. A solução encontrada para tal foi realizar uma alteração nas entradas da ALU. Fazendo o estudo de uma ALU, sabe-se que para a leitura das quatro instruções básicas (ADD, SUB, ORR e AND), ela requer apenas dois bits de entrada para diferencias as operações entre si: 00 para ADD, 01 para SUB, 10 para AND e 11 para ORR. A alteração realizada foi na inserção de mais um bit de entrada na ALU responsável por identificar a instrução LSL.