Universidade Federal da Paraíba
Departamento de Engenhara Elétrica

Profissional II

Projeto de um processador de ciclo único

Professor: José Maurício Neto (mauricio@cear.ufpb.br)

Temas:

- Projeto e implementação de um processador

Parte 1:

Assembler é um programa que traduz código assembly em código de máquina (binário). Na primeira parte desse projeto o objetivo é construir um mini-assembler chamado (mmasm) para um certo conjunto de instruções (Especificado na Parte 2);

Os seguintes requisitos devem ser considerados:
- O mmasm deve receber o programa (código assembly) através de um arquivo de entrada usado como argumento:./mmasm prog.asm
- O programa (código assembly) deve ser considerado sintaticamente correto
- O mmasm deve aceitar instruções do tipo R-Type, I-Type, e J-Type
- O programa não deverá conter símbolos (labels), tudo será tratado como endereço
- Considerar a primeira instrução no endereço 0x00

Parte 2:

O projeto consiste na implementação de uma CPU que lê um programa codificado de forma binária (conjunto de instruções e dados) na memória, e executa essas instruções. Cada palavra da memória deve conter pelo menos 4 bits que codificam a instrução a ser executada e 6 bits de dados (utilizados nas operações aritméticas), na forma "iiiidddddd". O registradores devem ter no mínimo 8 bits. A CPU deve ser capaz de executar, no mínimo, as seguintes instruções:

- ADD - Soma o valor de um registrador ao valor de outro registrador (ADD regA,regB)
- SUB - Subtrai o valor de um registrador ao valor de outro registrador (SUB regA,regB)
- ADDi - Soma o valor do dado ao valor de um registrador (ADDi regA,VALOR)
- SUBi - Subtrai o valor do dado ao valor de um registrador (SUBi regA,VALOR)
- MUL2 - Multiplica o valor de um registrador por 2
- DIV2 - Divide o valor de um registrador por 2
- CLR - Zera o conteúdo de um registrador
- RST - Zera o conteúdo de todos os registradores e reinicia a execução da primeira instrução na memória
- MOV – Copia o conteúdo de um registrador em outro registrador (MOV regA,regB)
- JMP – Altera a seqüência de execução das instruções definindo o endereço da próxima instrução a ser executada (JMP dddddd)
- OUT – envia um dado ao hardware de saída. (A saída será exibida num conjunto de displays de 7 segmentos)
- LOAD
- STORE

O projeto deve ser elaborado utilizando-se blocos lógicos (flip-flops, portas lógicas e memórias) ou por meio de linguagem verilog. Considere uma Memória de Instrução de, pelo menos, 50 instruções e uma memória de dados de, pelo menos 50 posições. O barramento de saída do processador, fisicamente, será conectado no FPGA da Altera ao conjunto de displays de 7

segmentos e deve usar um conversor de bcd pra display de 7seg de modo a exibir o dado do barramento.

Entregáveis:

- - Pacote com o projeto do mmasm, o projeto da CPU e um relatório
- - O relatório deve conter as seguintes coisas:
    - Descrição da sua solução para mmmas (os detalhes de implementação, algoritmos, lógica de funcionamento) bem como da implementação da CPU
    - Formatação das instruções assembly no arquivo .asm de entrada que o seu mmasm reconhece
    - Quatro exemplos de programas assembly que o seu mmasm consegue entender
    - Diagrama lógico de cada bloco lógico da CPU (Use software apropriado para construção do diagrama, Ex.: Quartus, Proteus, Digital Works...)
    - Tabela da verdade de cada bloco, ou diagramas de tempo do funcionamento de cada bloco.
    - Diagrama lógico completo da implementação da CPU (Use software apropriado para construção do diagrama, Ex.: Quartus, Proteus, Digital Works...)
    - Justificativa para a implementação realizada

Sugestões

Implemente o Assembler levando em consideração a arquitetura a ser implementada na parte 2. Leve em consideração o número de instruções que o CPU será capaz de implementar e utilize a codificação binária que melhor se adéque a arquitetura usada.