Laboratório 11

Este laboratório visa exercitar os conhecimentos sobre processadores. A plataforma utilizada é o processador m1ps (meu 1º processador simples), conforme o diagrama de blocos da Figura 1, e o hardware de monitoramento mM (m1ps Monitor), conforme descrito em aula.

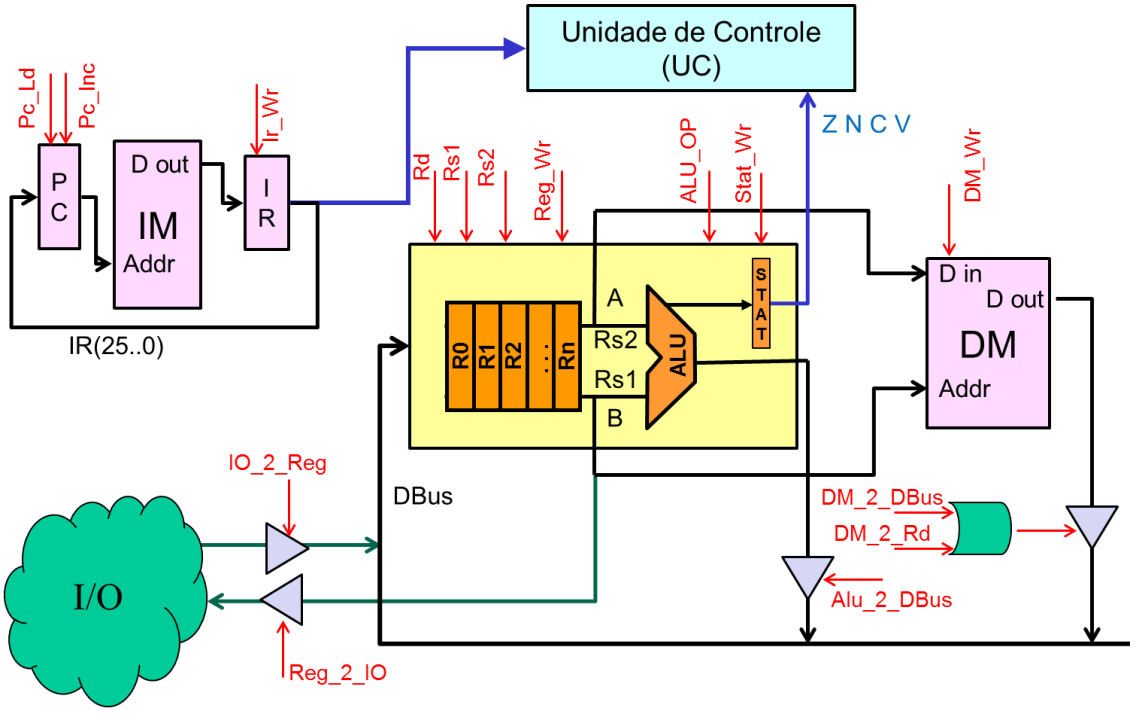

Figura 1 - Diagrama de Blocos do m1ps

São fornecidos códigos VHDL, completos ou não, de:

- m1ps *top-level* (código correto)
- módulos componentes do m1ps
- monitor mM (código correto)

Tudo é fornecido no arquivo ***m1ps.zip***. Dentro do arquivo compactado cada módulo está numa pasta. O *top-level* do processador está na pasta *Processor* com o nome de arquivo ***processor.bdf.*** Todos arquivos .vhd dos módulos estão nas pastas dos módulos e na pasta ***Packages***, exceto aqueles que serão pedidos para serem feitos.

A tarefa do aluno é completar, fazer alterações e extensões no processador.

Trabalho 1) Banco de registradores

Reutilizando o projeto de banco de registradores feito no Laboratório 7 (Questão 2), modifique-o para que ele siga as seguintes especificações do banco de registradores do processador m1ps:

1) Permita duas leituras simultâneas.
2) O registrador r0 não pode ter seu valor alterado, nunca.
3) Toda leitura endereçada ao registrador r0 retorna 0.

O *top-level* do banco de registradores pode ser visto no diagrama de blocos (componente BANK) do processor ou no arquivo de pacotes (***Packates/pack.vhd***). Após a modificação do VHDL copie seu ***.vhd*** para a pasta Packages com o nome ***bank.vhd***

Trabalho 2) Memória

Reutilizando o projeto da memória feito no Laboratório 10 (questão 1), modifique-o para que ele siga as especificações da Memória de Instrução (IM) e da Memória de Dados (DM) do processador m1ps. O *top-level* das memórias pode ser visto no diagrama de blocos (componentes I-Memory e D-Memory) do processador ou no arquivo ***pack.vhd*** da pasta Packages. Após a modificação do VHDL copie seu ***.vhd*** para a pasta Packages com o nome ***memory.vhd***

Note que as memórias IM e DM podem ser instanciadas por meio de um único componente, mas elas terão funções de diferentes: IM é uma memória apenas de leitura e DM de leitura e escrita.

Trabalho 3) Execução do código Fibonacci – 1

Crie um projeto na pasta ***Processor*** usando os arquivos ***processor.bdf*** e ***processor.vwf***, compile, simule para testar os Trabalhos 1 e 2.

Você pode ver o código fonte em assembly no arquivo ***fibonacci.m1ps*** dentro da pasta ***m1ps***. Para acompanhar a simulação você pode usar o arquivo ***m1psfibonacci.xls***.

É possível simular seu próprio código assembly. Basta escrevê-lo, submetê-lo ao montador em http://mc613.caiohoffman.org/ copiar a saída para um arquivo texto .mif. Instanciar a memória IM com seu arquivo e simular.

Trabalho 4) Análise de desempenho

Descubra a frequência máxima de *clock* para o processador (via analisador de *timing* do Quartus). Quais são (ou Qual é) o(s) componente(s) mais lento(s), isto é, fazem parte do caminho crítico da via de dados?

Trabalho 5) IO para as instruções IN e OUT

Altere o arquivo ***mM_toplevel.vhd*** na pasta ***mM_toplevel*** para que a saída IO_OUT do processador seja mostrada em hexadecimal nos quatro displays de sete segmentos. Note que não serão usados todos os bits.

Ainda no mesmo arquivo, altere-o para que a entrada IO_IN do processador receba 32 bits, nos quais os nove menos significativos serão os valores das *switches keys* 9 à 1. Importante: não use a SW0.

Trabalho 6) Execução do código Fibonacci – 2 (com mM)

Configure corretamente o projeto ***mM_toplevel*** (isto é, corrija a localização da pasta de pacotes, associe os pinos para as entradas e saídas feitas no Trabalho 4) compile e execute na placa DE1. Veja se o programa funciona como esperado.

Utilização do mM:

- Conecte a placa DE1 a um monitor.
- SW0 serve para trocar a tela de visualização. Uma tela apresenta os valores dos registradores e os sinais internos da via de dados do processador. A outra apresenta o estado da memória. Note que por limitações não são mostradas todas as palavras da memória, apenas as primeiras 208.
- KEY3 executa um *reset*.
- KEY1 executa instrução por instrução.
- KEY0 executa *clock* por *clock*.

Obs.: Importe o arquivo DE1_pin_assignments.csv para facilitar a associação dos pinos.

Trabalho 7) Implementação da instrução HALT

Implemente a instrução HALT no processador que tem a seguinte especificação: ao executar esta instrução, a máquina de estados fica com o estado preso no ciclo de execução e somente via sinal externo de ***reset*** o processador reinicia a execução com PC=0.

Utilize as seguintes especificações:

- ***Opcode*** da instrução => 011000.
- Instrução => 0x60000000.

Altere o código *assembly* fornecido (***fibonacci.m1ps***) e insira uma instrução HALT. Use o montador e atualize o arquivo ***fibonacci.mif*** da pasta ***Processor*** e ***Package***. Reexecute o Trabalho 6.

Importante:

- É necessário alterar a máquina de estados (componente UC) para implementar a instrução.
- Após fazer as modificações no módulos necessários, lembre-se atualizá-los na pasta ***Package***.

Trabalho 8) Extensão do ISA para ADDI

Reprojete o processador modificando o arquivo ***processor.bdf*** para implementar a instrução ***ADDI***, conforme apresentado nos slides da Aula. Será necessário alterar a via de dados e a unidade de controle. Após as modificações nos módulos necessários, lembre-se atualizá-los na pasta ***Package***.

Trabalho 9) Execução do código Fibonacci – 3 (Final)

Altere o código *assembly* ***fibonacci.m1ps*** de forma que ele use a instrução ***ADDI*** ao invés da instrução ***IN*** para iniciação dos registradores. Use o montador e atualize o arquivo ***fibonacci.mif*** da pasta ***Processor*** e ***Package***. Reexecute os Trabalhos 3 e 6.

Obs.: Nos testes, utilize valores imediatos positivos e negativos.

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

Entregar os .vhd gerados para os itens 1, 2 7 e 8.