Laboratório 10

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.

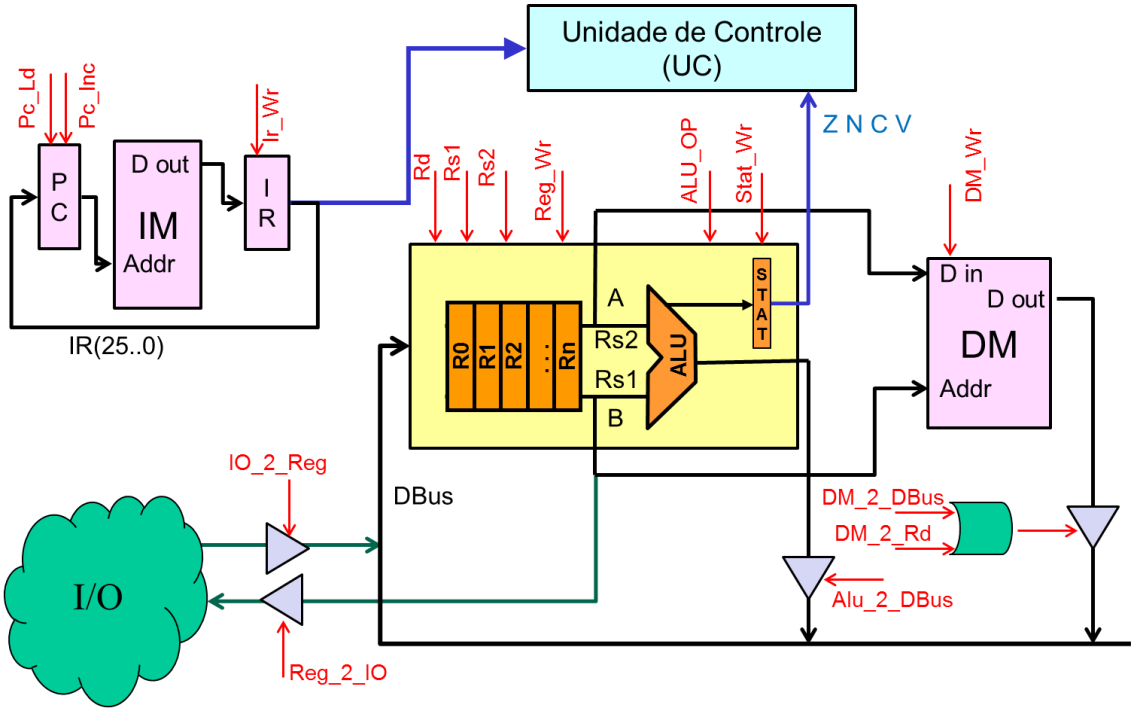

Figura 1 - Diagrama de Blocos do m1ps

Faça o download do arquivo *lab10_m1ps_v2018.1.qar* do Moodle. Ele é um arquivo de projeto do Quartus arquivado e contém o código-fonte - incompleto - do processador, o código do componente *m1ps monitor*, programas que computam a sequência de Fibonacci e arquivos auxiliares de simulação, configuração e acompanhamento.

Após restaurar o projeto no Quartus (*Project > Restore Archived Project*), o diretório alvo terá a seguinte estrutura:

- **mM**: diretório contendo os códigos VHDL do componente *m1ps monitor*;
- **Processor**: diretório contendo o código-fonte do processador;
  - *fibonacci*.mif*: imagens de memória com os programas compilados;
- **CPU.sdc**: arquivo de restrições para análise de tempo;
- **CPU_sim.vwf**: arquivo de entrada para simulação do processador;
- **fibonacci*.m1ps**: fontes Assembly dos programas;

Versão 28/04/2018.

- **fibonacci*.xls**: planilhas para acompanhamendo de simulação e execução, contendo cada instrução executada, sua representação em binário e hexadecimal, e os números de Fibonacci esperados.
- **m1ps.qpf**: arquivo de projeto do Quartus;
- **mM_toplevel.qsf**: configurações do projeto - já contém os assignments de pinos para gravação na placa.

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

Tarefa 1) Banco de registradores

Crie um banco de registradores para o m1ps, tomando como base o projeto feito no Laboratório 6 (Questão 2). Modifique-o para que ele siga as seguintes especificações :

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 processador (*Processor/CPU.bdf*) ou no arquivo de pacotes (*Processor/pack.vhd*). A entidade principal de seu banco de registradores deve ser adicionada ao projeto em um arquivo **<bank.vhd>** no diretório **Processor**.

Entrega: o arquivo <bank.vhd> e outros arquivos auxiliares utilizados devem fazer parte da entrega final.

Tarefa 2) Memória

Baseando-se no projeto da memória feito no Laboratório 9 (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 (*Processor/CPU.bdf*) ou no arquivo *Processor/pack.vhd*. A entidade principal de sua memória deve ser adicionada ao projeto em um arquivo **<memory.vhd>** no diretório **Processor.**

Entrega: o arquivo <memory.vhd> e outros arquivos auxiliares utilizados devem fazer parte da entrega final.

Tarefa 3) Execução do código Fibonacci – 1

Altere a entidade top-level do projeto para a entidade *CPU* <Processor/CPU.bdf> para simular o comportamento do processador. Lembre-se que, para simulação, não é necessário executar o fluxo inteiro de compilação do Quartus, apenas *Analysis & Synthesis* é o suficiente.

Utilize o arquivo *CPU_sim.vwf* para simulação e certifique-se de que as instruções estão sendo executadas corretamente. Você pode basear-se no

código-fonte Assembly <fibonacci.m1ps> e na planilha de acompanhamento <fibonacci.xls> na sua verificação.

Dica: essa simulação pode levar alguns minutos para ser completada. Você deveria ver os números de Fibonacci sendo exibidos no sinal *IO_OUT* a partir de, aproximadamente, metade do tempo de simulação.

Entrega: adicione no arquivo de entrega final um arquivo <fibonacci.png> com uma screenshot de simulação em que seja possível identificar a exibição de, pelo menos, dois números de Fibonacci consecutivos no sinal *IO_OUT*.

Tarefa 4) Análise de desempenho

Mantendo a entidade top-level do projeto como a entidade *CPU* <Processor/CPU.bdf>, utilize o analisador de tempo do Quartus (*TimeQuest*) para descobrir a frequência máxima de *clock* para o processador. Quais são (ou Qual é) o(s) componente(s) mais lento(s), isto é, que fazem parte do caminho crítico da via de dados?

Entrega: Elabore um arquivo **<analysis.txt>** contendo a frequência máxima encontrada e os componentes no caminho crítico. Adicione o arquivo na entrega final.

Dica: utilize os relatórios *Report Fmax Summary* e *Report Timing*. No relatório *Report Timing*, utilize o sinal de clock nos campos *From clock* e *To clock* - não é necessário preencher nenhum outro campo.

Para crédito extra: Proponha, brevemente, descrevendo no arquivo **<analysis.txt>** o que poderia ser alterado na via de dados do processador para melhorar o seu desempenho e quais as possíveis alterações na máquina de controle. A sua proposta não precisa ser completa - ou seja, não precisa explicar todos os detalhes de implementação - e nem precisa ser implementada, mas precisa demonstrar um bom entendimento do funcionamento do processador. Note que não existe apenas uma resposta “correta” para essa proposta. Não serão aceitas sugestões que envolvam apenas a utilização de componentes mais rápidos, ou seja, supondo que a memória seja o caminho crítico, uma proposta como “utilizar memórias mais rápidas” não é o suficiente.

Tarefa 5) Execução do código Fibonacci – 2 (com mM)

Defina novamente a entidade *mM_toplevel* como top level do projeto. Compile e grave na placa.

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 dos primeiros endereços da memória memória;

- SW[7..0] estão conectados aos bits menos significativos do sinal IO_IN do processador;
- Os visores de sete segmentos mostram os bits menos significativos do sinal IO_OUT do processador, convertidos para hexadecimal;
- KEY3 executa um *reset*;
- KEY2, quando pressionado, envia um clock de 20 Hz para o processador;
- KEY2, quando pressionado, envia um clock de 4 Hz para o processador;
- KEY0 é um clock manual.

Baseando-se no código Assembly <fibonacci.m1ps> e na planilha de acompanhamento <fibonacci.xls>, execute o código no processador (você terá que usar os switches da placa para enviar os valores de entrada) e confira o correto funcionamento.

Esta configuração poderá ser avaliada na demonstração.

Tarefa 6) Implementação da instrução ADDI

Modifique o processador para implementar a instrução ***ADDI***, conforme apresentado nos slides da Aula. Será necessário alterar a via de dados e a unidade de controle.

Formato da instrução ADDI:

Op(31..26)	Rd(25..21)	Rs1(20..16)	Imediato(15..0)	Obs
001 100				Rd ← Rs1 + Imed
6	5	5	16	

Entrega: Elabore um arquivo **<addi.txt>** descrevendo, brevemente, a sua estratégia de projeto para a nova instrução: o que foi alterado no processador e por quê. Adicione o arquivo **<addi.txt>** e todos os arquivos modificados ou criados no arquivo de entrega final.

Tarefa 7) Execução do código Fibonacci – 3 (ADDI)

a) Tomando como base o que foi feito na Tarefa 3, altere, na entidade *CPU* <Processor/CPU.bdf> o arquivo de imagem da memória de instruções para <fibonacci_addi.mif> e re-execute a simulação do processador. Utilize o código-fonte Assembly <fibonacci_addi.m1ps> e a planilha de acompanhamento <fibonacci_addi.xls> na sua verificação.

Entrega: adicione no arquivo de entrega final um arquivo **<fibonacci_addi.png>** com uma screenshot de simulação em que seja possível identificar a execução de, pelo menos, uma instrução ADDI.

b) Tomando como base a Tarefa 5, re-execute o programa de Fibonacci na placa utilizando a instrução ADDI.

Esta configuração poderá ser avaliada na demonstração.

Tarefa 8) 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.

Entrega: Elabore um arquivo **<halt.txt>** descrevendo, brevemente, a sua estratégia de projeto para a nova instrução: o que foi alterado no processador e por quê. Adicione o arquivo **<halt.txt>** e todos os arquivos modificados ou criados no arquivo de entrega final.

Tarefa 9) Execução do código Fibonacci – 3 (ADDI)

c) Tomando como base o que foi feito na Tarefa 3, altere, na entidade *CPU* <Processor/CPU.bdf> o arquivo de imagem da memória de instruções para <fibonacci_halt.mif> e re-execute a simulação do processador. Utilize o código-fonte Assembly <fibonacci_halt.m1ps> e a planilha de acompanhamento <fibonacci_halt.xls> na sua verificação.

Entrega: adicione no arquivo de entrega final um arquivo **<fibonacci_halt.png>** com uma screenshot de simulação em que seja possível identificar a execução da instrução HALT.

d) Tomando como base a Tarefa 5, re-execute o programa de Fibonacci na placa utilizando a instrução HALT.

Esta configuração poderá ser avaliada na demonstração.

Tarefa 10) Entrega final

Crie e envie no Moodle um arquivo compactado **<m1ps.zip>** contendo:

- A descrição completa do processador (diretório Processor), com as últimas versões de todos os arquivos modificados neste Laboratório;
- Os arquivos de screenshot das Tarefas 3, 7 e 9;
- Os arquivos de texto das Tarefas 4, 6 e 8.