Universidade Federal De Campina Grande Departamento De Engenharia Elétrica
Laboratório De Arquitetura De Sistemas Digitais – LASD
Prof. Rafael B. C. Lima

Universidade Federal de Campina Grande

Aluno: _____

Matrícula: _____ Data: _____

Sprint 7 – Instruções de desvio – CPU MIPS

Descrição geral do problema: Modifique o circuito de avanço do PC para incluir as instruções de desvio condicional (BEQ) e incondicional (J).

Requisitos mínimos:

Abra o projeto da Sprint6 e edite-o para incluir as funcionalidades dessa sprint. **Obs: "File > Open Project" e NÃO "File > Open".**

1. A fim de completar a versão v0.3 da CPU, modifique o circuito de avanço do PC para possibilitar desvios condicionais, em função de um teste de igualdade (BEQ – Branch if equal) e incondicionais (J – Jump).
   - Instancie 2 novos muxes *MuxPCSrc* e *MuxJump*;
   - Inclua mais um somador para o imediato na instrução BEQ. *Adder Branch*
   - Conecte os sinais de controle *Jump* e *Branch* nos respectivos muxes.
   - A sugestão de montagem está representada na Figura 1.

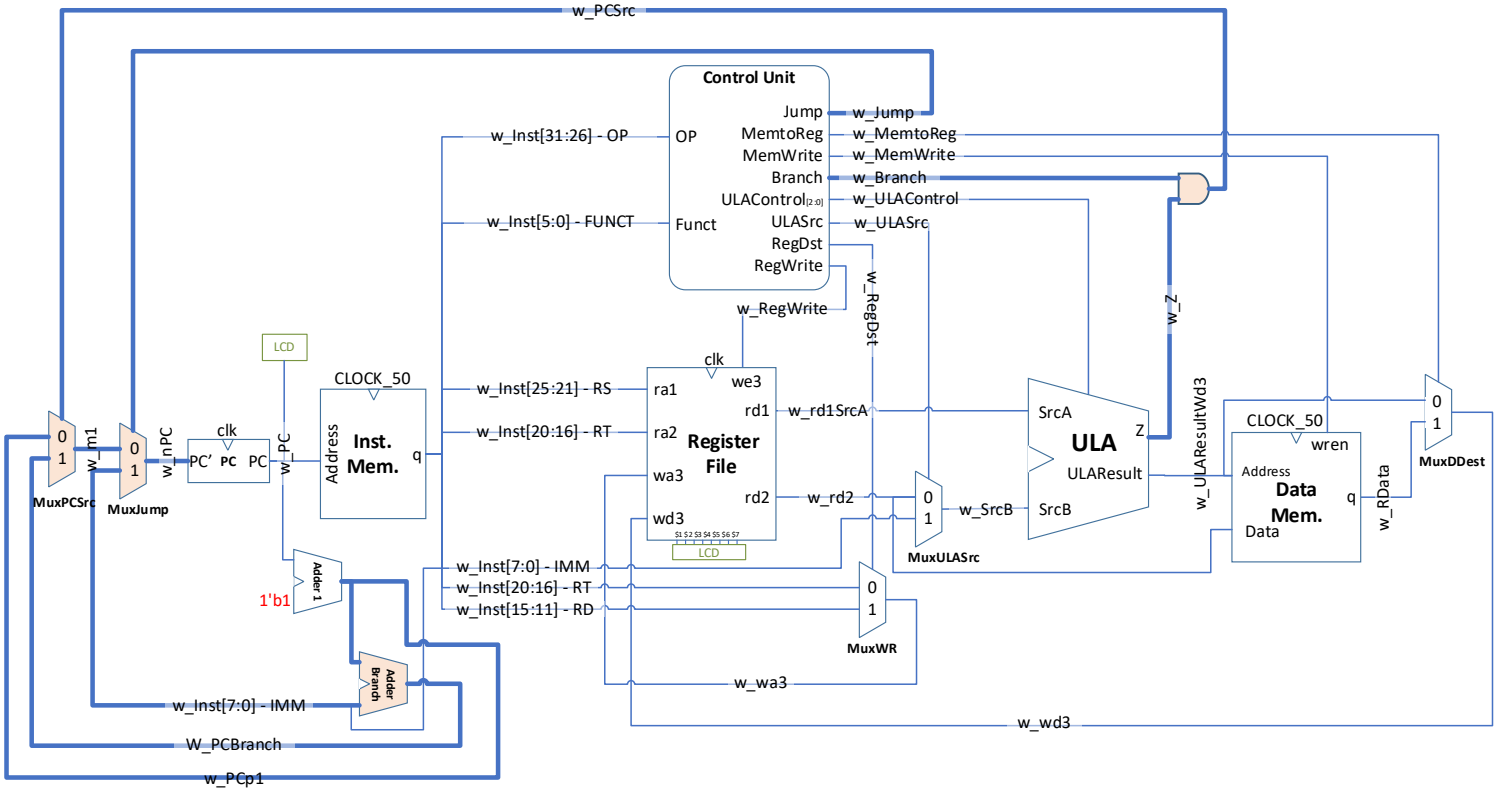

Figura 1 – CPU V0.3 e memórias

Nome	Tamanho
w_PCSrc	1 bit
w_Jump	1 bit
w_Branch	1 bit
w_Z	1 bit

Nome	Tamanho
w_m1	8 bits
w_nPC	8 bits
w_PCBranch	8 bits

Tabela 1 – novos fios, utilizados na montagem

2. Na sprint anterior, foi implementada uma nova memória de instruções que era inicializada por meio de um arquivo *.mif*. Esse arquivo continha o conjunto de instruções, em código de máquina, referentes ao programa a ser executado. Para facilitar a criação de novos programas, utilize o software **MIPS_Assembler** para converter seus códigos de assembly para código de máquina.
    - Baixe o executável nesse LINK e recorte-o para a pasta local do seu projeto.
    - Digite o código, em assembly, do programa a ser executado e clique em "converter". Será gerado um arquivo .mif na mesma pasta que o executável se encontra. Se o arquivo .mif tiver o mesmo nome que você apontou ao criar a memória ROM de instruções, basta compilar o projeto novamente no Quartus II, que o seu novo programa já estará "carregado"! Para mudar o nome do arquivo .mif vá em *Arquivo>Configuração do Mif*
    - Essa ferramenta é muito simples, somente suporta as 13 instruções presentes na Figura 2. Para mais detalhes, acesse o menu de Ajuda. Usar os registradores no formato $x, espaço simples e virgula. Todas as constantes já estão em hexa.

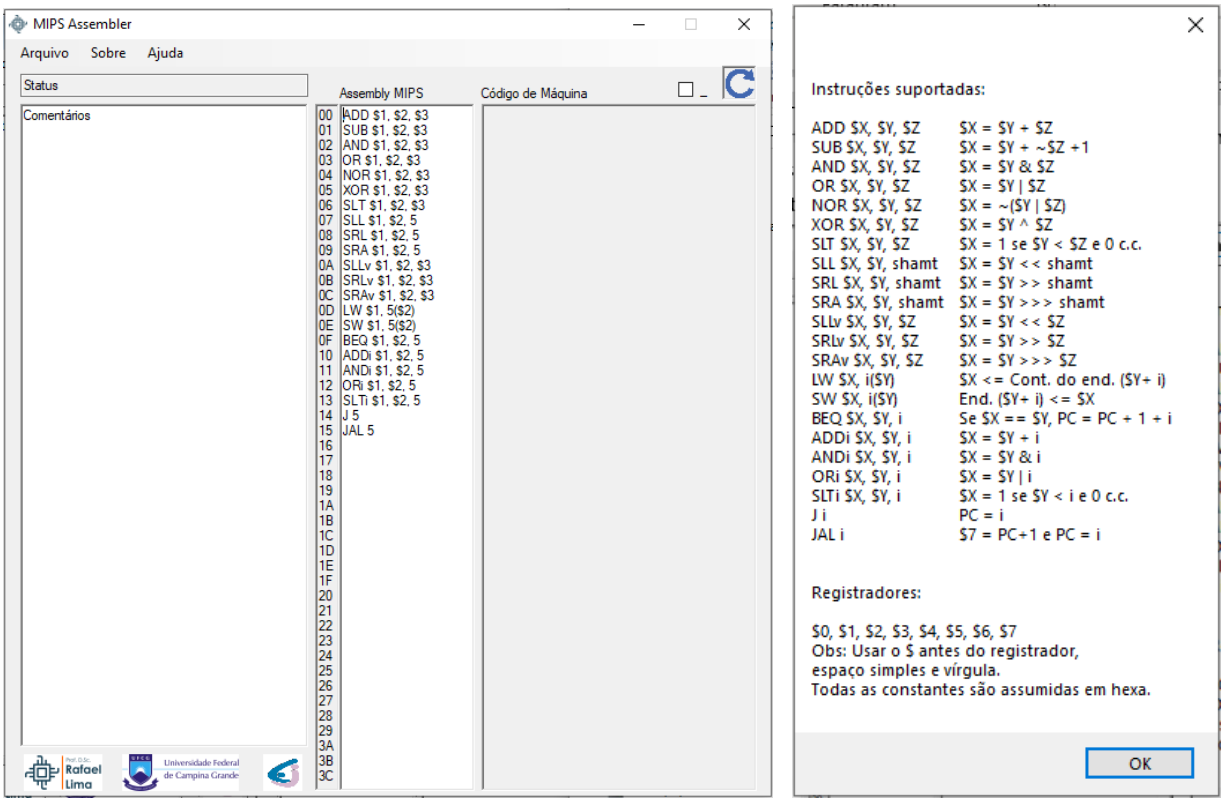

Figura 2 – MIPS_AssemblerV1_4.exe

Relembrando o conjunto de instruções suportadas pela CPU

Instrução	Descrição	Algoritmo
ADD $X, $Y, $Z	Adicionar	$X = $Y + $Z
SUB $X, $Y, $Z	Subtrair	$X = $Y - $Z
AND $X, $Y, $Z	AND Bit a bit	$X = $Y & $Z
OR $X, $Y, $Z	OR Bit a bit	$X = $Y | $Z
NOR $X, $Y, $Z	NOR Bit a bit	$X = ~($Y | $Z)
SLT $X, $Y, $Z	Menor que	$X = 1 se $Y < $Z e 0 c.c.
LW $X, i($Y)	Carregar da memória	$X <= Cont. do end. ($Y+ i)
SW $X, i($Y)	Armazenar na memória	End. ($Y+ i) <= $X
BEQ $X, $Y, i	Desviar se igual	Se $X == $Y, PC = PC + 1 + i
ADDi $X, $Y, i	Adicionar Imediato	$X = $Y + i
J i	Desvio incondicional	PC = i

Tabela 2 –Conjunto de instruções MIPS suportadas pela CPU do LASD

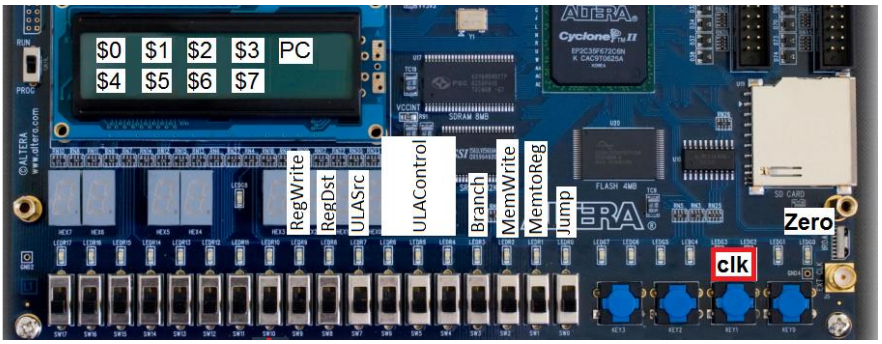

Figura 3 – Placa

3. Aumente o clock do seu processador para 2Hz.
4. Rode o programa da Tabela 3 e diga qual o conteúdo dos registradores ao finalizá-lo:

$0:_0_, $1:_7_, $2:___, $3:___, $4:___, $5:___, $6:___, $7:___

Posição	Assembly
00	ADDi $1, $0, 7
01	ADDi $3, $0, 3
02	ADDi $2, $0, FF
03	ADDi $2, $2, 1
04	SLT $7, $2, $3
05	BEQ $1, $2, 1
06	J 3
07	J 2

loop que faz $7 ser 1 por 3 iterações e 0 em uma, e reseta

PC + 2

Alguma ideia de um possível uso para esse código?

Tabela 3 –programa teste

Desafio (Valendo +0,1 na média geral)

- Escreva um programa que carregue uma constante qualquer de 4 bits ($X_3X_2X_1X_0$) no registrador $1;
- Em seguida, separe os quatro dígitos, em binário;
- Armazene cada bit em um registrador **$4 = X_3, $5 = X_2, $6 = X_1** e **$7 = X_0**;
- Teste sua lógica com a constante 4'h9 (4'b1001). **$1 = 09**, **$4 = 01**, **$5 = 00**, **$6 = 00**, **$7 = 01**.

BONUS >>> Quem fizer o código mais otimizado ganhará uma pontuação extra!

- O código com a menor quantidade de instruções (economia de memória de programa) ganhará +1,0 na média geral!
- O código que necessitar da menor quantidade de clocks para chegar no resultado final (maior velocidade) ganhará +1,0 na média geral!

Em caso de empate, ganhará quem usar menos registradores. Se o empate persistir, a data de entrega será o fator decisivo. Lembrando que seu programa tem que rodar na CPU da Figura 1.

Submeta seu desafio nesse LINK.