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: _____

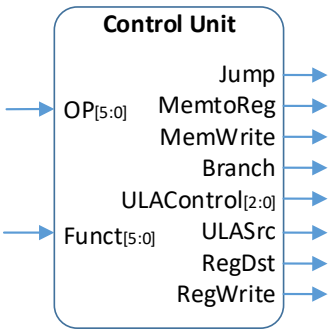
Sprint 5 – Unidade de Controle – CPU MIPS

Descrição geral do problema: Implemente a Unidade de Controle, uma memória ROM de instruções e o registrador PC. Esses novos blocos possibilitarão rodar 6 tipos de instruções em assembly de MIPS.

Requisitos mínimos:

Abra o projeto da Sprint4 e edite-o para incluir as funcionalidades dessa sprint. **Obs: "File > Open Project" e NÃO "File > Open".**

1. Faça a descrição de hardware, em Verilog, da Unidade de Controle indicada na figura 1. Esse módulo gera os sinais de controle para cada uma das instruções suportadas pela CPU.



- Esse circuito é um decodificador COMBINACIONAL.
- A relação lógica entre as entradas e saídas é definida pela Tabela 1
- Dica: É possível utilizar 1 ou 2 "cases" na implementação do módulo (Lembrar do "default")

Figura 1 – Unidade de Controle.

	ENTRADAS		SAÍDAS							
Instr	**OP**	**Funct**	**RegWrite**	**RegDst**	**ULASrc**	**ULAControl**	**Branch**	**MemWrite**	**MemtoReg**	**Jump**
ADD	000000	100000	1	1	0	010	0	0	0	0
SUB	000000	100010	1	1	0	110	0	0	0	0
AND	000000	100100	1	1	0	000	0	0	0	0
OR	000000	100101	1	1	0	001	0	0	0	0
SLT	000000	101010	1	1	0	111	0	0	0	0
LW	100011	xxxxxx	1	0	1	010	0	0	1	0
SW	101011	xxxxxx	0	x	1	010	0	1	x	0
BEQ	000100	xxxxxx	0	x	0	110	1	0	x	0
ADDi	001000	xxxxxx	1	0	1	010	0	0	0	0
J	000010	xxxxxx	0	x	x	xxx	x	0	x	1

Tabela 1 – Tabela do decodificador da Unidade de Controle

A arquitetura MIPS prevê 3 tipos de instruções **R**, **I** e **J**. A regra de formação do código de máquina de cada tipo de instrução está ilustrada na Tabela 2. Perceba que cada uma das instruções pode ser unicamente identificada pela combinação dos campos **OP** e **funct**. As instruções do tipo **R** tem **OP** igual a zero e são diferenciadas pelo campo **funct**. Já instruções do tipo **I** e **J**, dependem somente do **OP**.

Tipo R	OP[5:0]	rs[4:0]	rt[4:0]	rd[4:0]	shamt[4:0]	funct[5:0]
Tipo I	OP[5:0]	rs[4:0]	rt[4:0]	imm[15:0]		
Tipo J	OP[5:0]	address[25:0]				

Tabela 2 – Regra de formação do código de máquina das instruções MIPS

2. Faça a descrição de hardware, em Verilog, do Registrador PC (Program Counter). Tal componente é um registrador de 8 bits com uma entrada para carregamento paralelo (*PCin)* e uma saída paralela (PC).

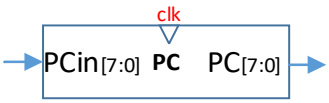


Figura 2 – PC.

- Esse circuito é SEQUENCIAL. Depende da borda de subida do clock.
- A cada clock, o novo *PCin* é carregado no registrador e disponibilizado na saída *PC*

3. O último elemento a ser implementado é a memória de instruções. Nessa sprint, deverá ser implementada uma memória ROM de instruções puramente combinacional, com até 256 posições de 32 bits. A sequência de instruções (programa) que será executada pela CPU, é armazenada na memória de instruções.

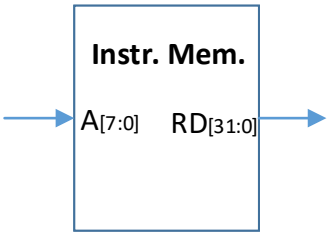


Figura 3 – Instruction Memory.

- Esse circuito é COMBINACIONAL
- Essa memória será somente de LEITURA. Dessa forma poderá ser implementada como um decodificador. Um *case,* por exemplo*,* (Lembrar do "default").
- O conteúdo da memória (linhas do case) será definido pela sequência de instruções do programa a ser executado. Lembrando que as instruções devem estar em código de máquina. Nessa sprint, deverá ser executado o programa da Tabela 3.

Endereço	Assembly	Código de máquina
00	ADDi $1, $0, 3	32'b_001000_00000_00001_00000_00000_000011
01	ADDi $2, $0, 9	32'b_001000_00000_00010_00000_00000_001001
02	ADD $2, $1, $2	32'b_000000_00001_00010_00010_00000_100000
03	AND $3, $1, $2	32'b_000000_00001_00010_00011_00000_100100
04	OR $4, $1, $2	32'b_000000_00001_00010_00100_00000_100101

Tabela 3 –programa teste

4. A fim de completar a primeira versão da nossa CPU v0.1, todos os módulos desenvolvidos até agora devem ser **instanciados** e **conectados** conforme o circuito da Figura 4.
   - Os fios devem ser criados com uma nomenclatura lógica para facilitar o entendimento geral do circuito e facilitar o debug.
   - Essa CPU v0.1 será capaz de rodar as instruções ADD, SUB, AND, OR, SLT e ADDi

Ao término da última sprint, a CPU será capaz de rodar as 10 instruções da Tabela

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
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 4 –Conjunto de instruções MIPS suportadas pela CPU do LASD

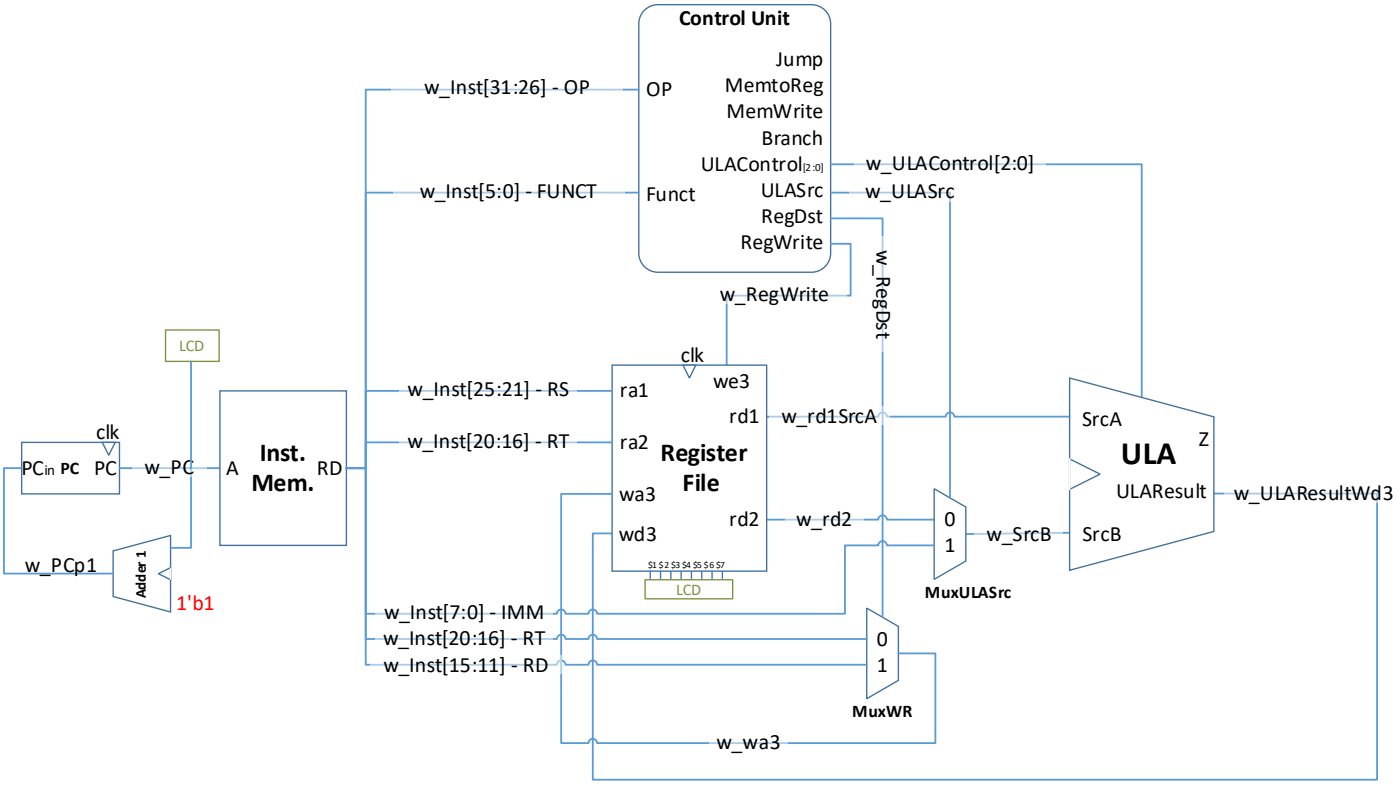


Figura 4 – CPU v0.1

Os fios utilizados na montagem da Figura 4, estão resumidos na Tabela 5.

Nome	Tamanho
w_ULASrc	1 bit
w_RegWrite	1 bit
w_RegDst	1 bit
w_ULAControl	3 bits
w_wa3	3 bits
w_PCp1	8 bits

w_PC	8 bits
w_rd1SrcA	8 bits
w_rd2	8 bits
w_SrcB	8 bits
w_ULAResultWd3	8 bits
w_Inst	32 bits

Tabela 5 –fios utilizados na montagem

5. Ligações auxiliares para Debug:
   - Faça uma pequena alteração no módulo RegisterFile. Crie 8 saídas auxiliares para visualizar externamente os valores de cada um dos registradores. Ao instanciar o módulo, faça as seguintes ligações: .S0(w_d0x0), .S1(w_d0x1), .S2(w_d0x2), .S3(w_d0x3), .S4(w_d1x0), .S5(w_d1x1), .S6(w_d1x2), .S7(w_d1x3). Nesse caso, o LCD deverá ficar conforme a Figura 5.
   - Mostre também o PC, na posição w_d0x4 do LCD. (assign)
   - Visualize os 8 sinais de controle gerados pelo módulo "Control Unit" nos LEDs vermelhos. LEDR[9:0]. (assign)

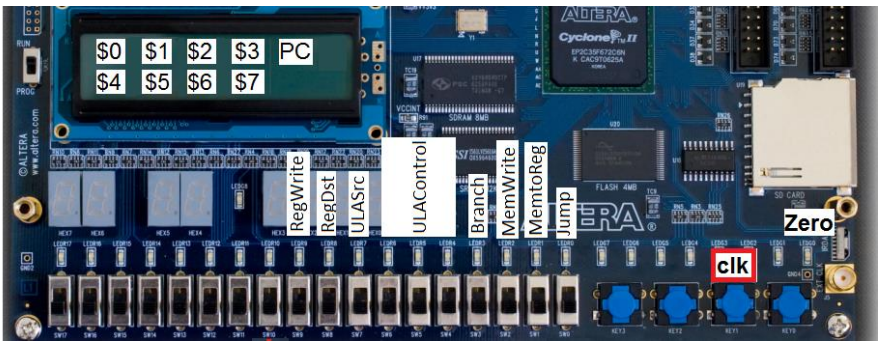


Figura 5 – Placa

6. O circuito proposto tem quantas entradas externas? Rode o programa da Tabela 3 e diga qual o conteúdo dos registradores ao finalizá-lo:

$0:___, $1:___, $2:___, $3:___, $4:___

Desafio (Valendo +0,1 na média geral)

- Escreva e rode na CPU, um novo programa para testar a instrução SLT. Ambos os casos da condição, verdadeiro e falso.