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

Aluno: _____

Matrícula: _____ Data: _____

Sprint 8 – Desvios – Processador RISC-V

Descrição geral do problema: Modifique o processador da sprint anterior para dar suporte a instrução de desvio BEQ. Atualize o *datapath* e a unidade de controle.

Requisitos mínimos:

Abra o projeto da Sprint7 e edite-o para incluir as funcionalidades dessa sprint. **Obs: "File > Open Project" e NÃO "File > Open".**

1. Atualize a unidade de controle para gerar os sinais relativos à instrução BEQ. Segue na Tabela 1 a lógica do novo decodificador.

		ENTRADAS			SAÍDAS						
	Instr	**OP**	**Funct3**	**Funct7**	**RegWrite**	**ImmSrc**	**ULASrc**	**ULAControl**	**MemWrite**	**ResultSrc**	**Branch**
R	ADD	0110011	000	0000000	1	xx	0	000	0	0	0
	SUB	0110011	000	0100000	1	xx	0	001	0	0	0
	AND	0110011	111	0000000	1	xx	0	010	0	0	0
	OR	0110011	110	0000000	1	xx	0	011	0	0	0
	SLT	0110011	010	0000000	1	xx	0	101	0	0	0
I	ADDi	0010011	000	xxxxxxx	1	00	1	000	0	0	0
	LW	0000011	010	xxxxxxx	1	00	1	000	0	1	0
S	SW	0100011	010	xxxxxxx	0	01	1	000	1	x	0
B	BEQ	1100011	000	xxxxxxx	0	10	0	001	0	x	1

Tabela 1 – Tabela do decodificador da Unidade de Controle

Utilize o testbench fornecido (UC_SP8_TB_32.sv e test_vector_SP8.txt) para simular seu módulo da Unidade de Controle no ambiente https://edaplayground.com/ . Certifique-se que todos os testes rodaram sem falhas ("Passou"), antes de prosseguir para a próxima etapa. Alguns exemplos, podem ser encontrados na seguinte videoaula sobre Testbenches no EDAplayground: https://www.youtube.com/watch?v=VsP6zHarUSM .

Lembrando que agora serão suportadas instruções do tipo R (add, sub, and, or e slt), I (addi e lw), S (sw) e B (beq)

	31:25	24:20	19:15	14:12	11:7	6:0
Tipo R	$Funct7_{6:0}$	$Rs2_{4:0}$	$Rs1_{4:0}$	$Funct3_{2:0}$	$Rd_{4:0}$	$Op_{6:0}$
Tipo I	$Imm_{11:0}$		$Rs1_{4:0}$	$Funct3_{2:0}$	$Rd_{4:0}$	$Op_{6:0}$
Tipo S	$Imm_{11:5}$	$Rs2_{4:0}$	$Rs1_{4:0}$	$Funct3_{2:0}$	$Imm_{4:0}$	$Op_{6:0}$
Tipo B	$Imm_{12,10:5}$	$Rs2_{4:0}$	$Rs1_{4:0}$	$Funct3_{2:0}$	$Imm_{4:1,11}$	$Op_{6:0}$

Tabela 2 – Regra de formação do código de máquina das instruções RISC-V

2. Atualize o conteúdo da memória de instruções, com o código de máquina do programa definido na Tabela 3. *Dica: utilize o RARs para converter o assembly em código de máquina.*

```
addi x1, x0, 7          #inicializa x1 com 7
addi x3, x0, 3          #inicializa x3 com 3

init:
addi x2, x0, -1         #inicializa o contador x2 com -1

incremento:
addi x2, x2, 1          #incrementa x2
slt x7, x2, x3          #atualiza a saída x7
beq x1, x2, init        #se a condição de parada for atingida, reinicia a rotina
beq x0, x0, incremento  #caso contrário, volta para incrementar x2
```

Alguma ideia de um possível uso para esse código?

Tabela 3 – Programa de testes dessa sprint

3. A fim de completar a próxima versão da CPU v0.3, todos os módulos desenvolvidos até agora devem ser **instanciados** e **conectados** conforme o circuito da Figura 2.

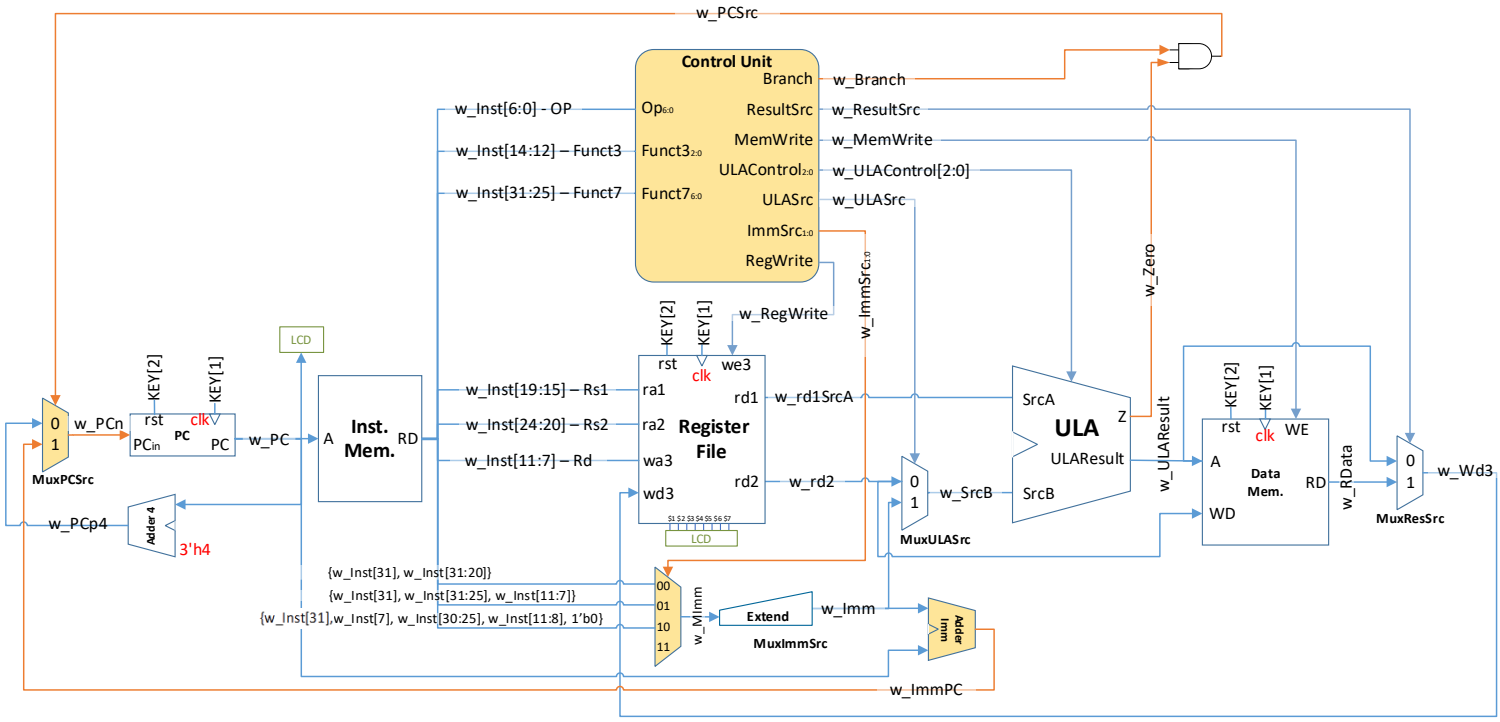

Figura 2 – Processador V0.3

Perceba que o módulo do extensor sofrerá uma modificação pois os imediatos do tipo I e S tem 12bits, já o tipo B tem 13. Pode-se internalizar o MuxImmSrc no módulo *Extend* e expandir os sinais com comprimentos diferentes ou padronizar em 13bits e estender 19bits, conforme a Figura 2.

Nome	**Tamanho**
w_PCSrc	1 bit
w_Zero	1 bit
w_Branch	1 bit

w_ImmSrc	2 bits
w_ImmPC	32 bits
w_PCn	32 bits

Tabela 4 – novos fios, utilizados na montagem

4. Após o debug preliminar, aumente o clock para 2Hz.

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
SLT $X, $Y, $Z	Menor que	$X = 1 se $Y < $Z e 0 c.c.
LB $X, i($Y)	Carregar da memória	$X ← end[$Y+ i]
SB $X, i($Y)	Armazenar na memória	End[$Y+ i] ← $X
ADDi $X, $Y, i	Adicionar Imediato	$X = $Y + i
BEQ $X, $Y, i	Desviar se igual	Se $X == $Y, PC = PC + i

Tabela 6 –Conjunto de instruções RISC-V suportadas pela CPU do LASD

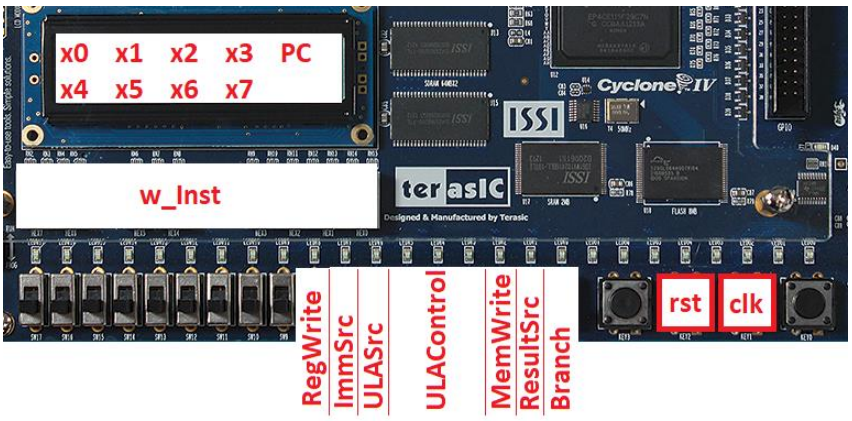

Figura 3 – Placa Altera DE2

5. Rode o programa da Tabela 1 e diga qual o conteúdo dos registradores e da memória de dados, ao finalizá-lo:

   Registradores:

Registrador	x0	x1	x2	x3	x4	x5	x6	x7
Dado								

Desafio (Valendo +0,5 na média geral)

- Adicione suporte para a instrução JALR. Isso possibilitará chamar e retornar das sub-rotinas;
- Teste seu projeto, escrevendo uma sub-rotina de delay, com duração de 500ms (Assuma que o clock da CPU é 100Hz);
- Utilize sua sub-rotina para criar uma onda quadrada de aproximadamente 1Hz no registrador $6;

op	funct3	funct7	Type	Instruction	Description	Operation
1100111 (103)	000	-	I	jalr rd, rs1, imm	jump and link register	PC = rs1 + SignExt(imm), rd = PC + 4