Trabalho de Verilog 5

Instruções

- O exercícios devem ser implementadas usando a linguagem Verilog. Todos os códigos devem ser simulados no ModelSim.
- A solução deverá ser salva em um arquivo no formato V. Cada equipe vai enviar dois arquivos Equipe-X e Equipe-TB-X onde X é o número da equipe (ex: Equipe-1.v, Equipe-TB-1.v, etc). **Arquivos com nomes em formatos diferentes destes serão ignorados**.
- Entrega: **até às 23:55h do dia 06/07/2020** (não serão aceitas listas entregues após este horário).
- Os arquivos devem ser entregues pelo classroom.
- Cada equipe deve resolver a sua lista de **forma individual**.
- **CUIDADO COM CÓPIAS!** Cópias não serão toleradas! Será usada uma ferramenta para detecção de cópias. Será feita a comparação com soluções disponíveis na internet e com os exercícios entregues pelas demais equipes. Caso seja detectada alguma fraude as questões serão anuladas.
- Este trabalho vale 10 pontos e tem peso 45% na nota da terceira unidade.

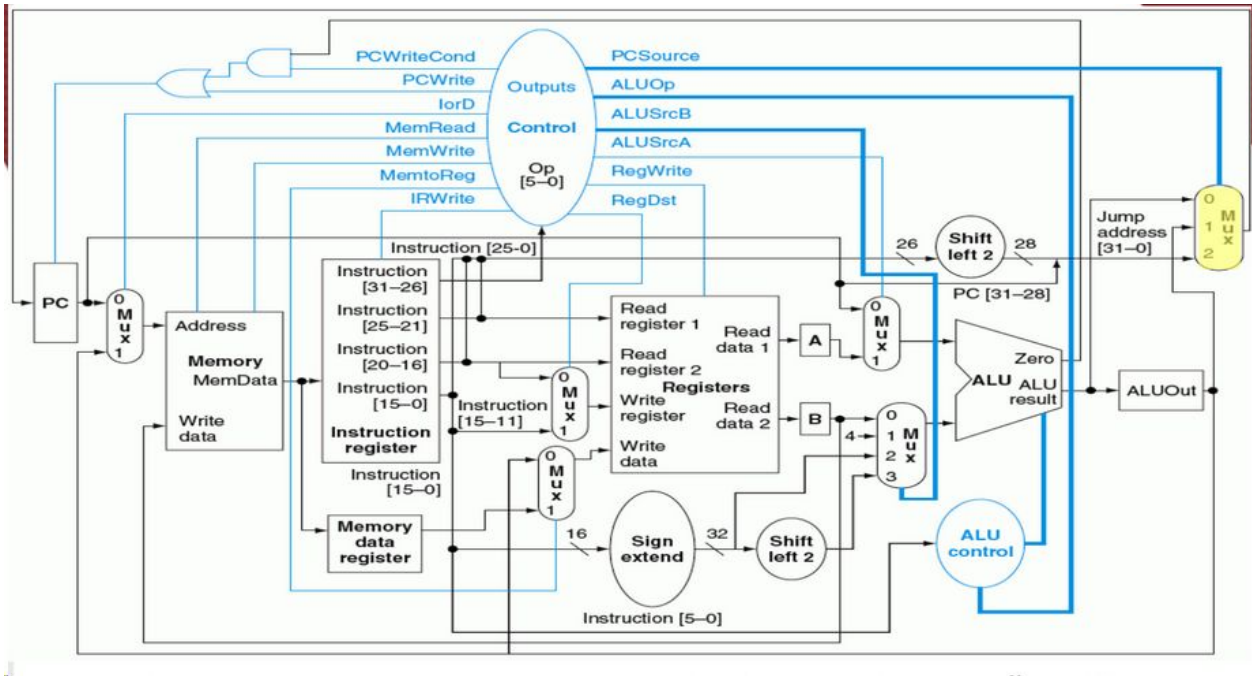
1. Projete a unidade de controle de um MIPS multiciclo. Considere como referência os diagramas de estados elaborados no trabalho de multiciclo realizado na segunda unidade. Considere que a unidade de controle deverá ter os seguintes sinais:

Nome	**Descriçao**	**Tamanho (bits)**
IorD (saída)	Seleciona que componente fornece o endereço para acesso a memória: 0 - PC fornece endereço para a memória 1 - ALUOut fornece endereço para a memória	1 bit
MemRead (saída)	Quando em 1 habilita leitura da memória.	1 bit
MemWrite (saída)	Quando em 1 habilita escrita na memória.	1 bit

MemToReg (saída)	Seleciona a origem do dado a ser escrito em registrador 0 - Valor escrito na entrada “Write data” dos registradores vem da ALUOut 1 - Valor escrito na entrada “Write data” vem de MDR	1 bit
IRWrite (saída)	Quando em 1 habilita a escrita no registrador de instrução	1 bit
RegDst (saída)	Seleciona o registrador a ser escrito: 0 - Número do registrador destino para escrita vem de rd 1 - Número do registrador destino para escrita vem de rt	1 bit
RegWrite (saída)	Quando em 1 habilita escrita no banco de registradores	1 bit
ALUSrcA (saída)	Seleciona o operando A da ULA: 0 - 1° operando da ALU vem do PC 1 - 1° operando da ALU vem do registrador A	1 bit
ALUSrcB (saída)	Seleciona o operando B da ULA 00 - 2° Operando da ALU vem de B 01 - 2° Operando da ALU é a constante 4 10 - 2° Operando é o sinal estendido, 16 bits menos significantes de IR 11 - 2° Operando é o sinal estendido, 16 bits menos significantes de IR, deslocados de 2 bits para a esquerda	2 bits
ALUOp (saída)	Seleciona a operação a ser realizada pela ULA: 00 - ALU faz soma 01 - ALU Faz subtração	2 bits
PCSource (saída)	Seleciona a a origem do valor a ser escrito no PC: 00 - Saída de ALU (PC + 4) é enviado ao PC para escrita	2 bits

	01 - ALUOut (endereço destino do branch) é enviado ao PC para escrita 10 - Endereço destino do jump (26 bits menos significativos do IR deslocados de 2 bits para a esquerda concatenados com PC+4[31:28]) é enviado ao PC para escrita	
PCWrite (saída)	Quando 1 habilita a escrita no PC	1 bit
PCWriteCond (saída)	Quando 1 habilita a escrita no PC se o sinal Zero da ULA for igual 1.	1 bit
Clk (entrada)	Sinal de clock proveniente do testbench	1 bit
reset (entrada)	Sinal de reset proveniente do test bench. Quando em 1 este faz com que a máquina de estados da unidade de controle retorne ao estado inicial.	1 bit
Opcode (entrada)	Opcode proveniente do registrador de instrução.	6 bits
Funct (entrada)	Código de função proveniente do registrador de instrução. Este valor deverá ser considerado sempre que o Opcode for igual a 0x00	6 bits

O diagrama abaixo mostra a conexão dos sinais acima descritos:



Implemente a maquina de estados de forma que haja suporte as seguintes instruções:

- Instrução SW (opcode = 0x2B)
- Instrução LW (opcode = 0x23)
- Instrução J (opcode = 0x02)
- Instrução ADD (opcode = 0x00 e funct = 0x20)
- Instrução SUB (opcode = 0x00 e funct = 0x22)
- Instrução ADDI (opcode = 0x8)

2. Implemente um test bench para validar o correto acionamento dos sinais de controle da unidade de controle implementado na questão anterior.
    a. O test bench deverá gerar um sinal de clock
    b. Gere um sinal de reset no início do teste para garantir que a máquina de estados estará num estado conhecido.
    c. Os opcodes e códigos de funct devem ser gerados pelo TestBench.