Birla Institute of Technology and Science – Pilani, Hyderabad/Goa Campus
Second Semester 2018-19
CS F342: Computer Architecture Comprehensive Exam (PART B, Closed Book-Solutions)
Note: Answer all the questions in the same sequence in the main answer sheet.

---

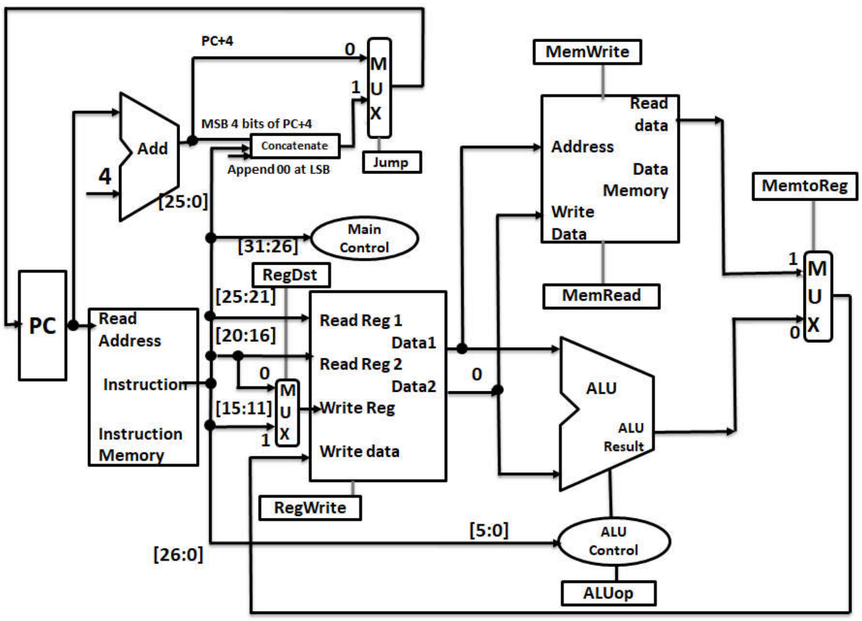
1. A modified MIPS ISA has support only for following instructions:
Load and Store instructions without the offset address
Example:
lw \$s0, (\$t0) ; loads the register \$s0 with a value from memory location whose address is stored in \$t0 register.
sw \$s0, (\$t0) ; Stores the contents of register \$s0 to a memory location whose address is stored in \$t0 register.
Arithmetic and Logical Instructions (Add, Sub, Or, And)
Example:
add \$s0, \$s1, \$s2 ; adds the contents of \$s1 and \$s2 register and stores the result in \$s0 register.
Unconditional Jump Instructions
Example:
J Label ; Jumps unconditionally and uses Pseudo-direct addressing (Same as in MIPS) to calculate the jump address
The major components available for implementing above ISA are Instruction Memory with delay of 200ps; Data Memory with delay of 200ps; ALU with delay of 100ps; Register File with delay of 50ps for read and delay of 50ps for write; Assume the delays of Multiplexers, Adders, PC register as 10ps;
(Assume that instruction format for the above specifications remains the same as MIPS ISA except for load/store instructions without the offset address. For load/store instructions without offset address the last 16-bits of the instruction code are not used)

(i) Draw a neat labelled, optimized, datapath using as less resources as possible, for single cycle implementation of modified MIPS processor, for the above specifications. Also label all the control signals in the datapath.
(Full marks will be given only for optimized datapath and not necessarily for the correct datapath.)

(ii) List the values of all control signals for all the 7 instruction types (lw, sw, add, sub, or, and, J) in tabular form. You don't have to specify the value of control signal for ALU, instead mention the operation done by the ALU.

(iii) Find the clock cycle time for the optimized datapath, assuming that the control unit is not part of the critical path. **[15+8+2 = 25M]**

(i)Datapath (* Concatenation of bit lines does not require computational blocks hence delay is 0)



(ii)

Control Signals→ Instructions	RegDst	RegWrite	ALU Operation	MemRead	MemWrite	MemtoReg	Jump
lw	0	1	Not Used	1	0	1	0
sw	X	0	Not Used	0	1	X	0
add	1	1	addition	0	0	0	0
sub	1	1	subtraction	0	0	0	0
or	1	1	OR	0	0	0	0
and	1	1	AND	0	0	0	0
j	X	0	Not Used	0	0	X	1

(iii) Clock cycle time = 10ps (PC Reg) + 200ps (Instruction Memory) + 50ps (Regfile Read) + 200ps (Data Memory*) + 10ps (Mux MemtoReg) + 50ps (Regfile Write) = 520ps

(Note that the RegDst Mux does not contribute to critical path delay)

(ALU has delay of 100ps which is less than delay of Data Memory. Since the Data Memory and ALU are in parallel, worst case delay for this stage is 200ps)

2. General implementation of forwarding in MIPS processor (discussed in class) allows forwarding from EX/MEM register and MEM/WB register to input of EX stage only. But some cases forwarding might be required from MEM/WB register to the input MEM stage. An example below shows this case

lw $s0, 0($t0)

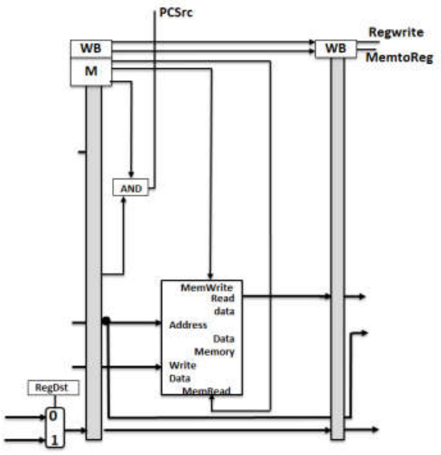
sw $s0, 0($t1)

A stall can be avoided in the above case by allowing forwarding from MEM/WB register to the input MEM stage.

(i) Determine the conditions (expression) under which forwarding should be allowed from MEM/WB register to input of MEM stage.

(ii) Also redraw, neatly, the MEM stage datapath and show the modification to be done in the MEM stage (shown in figure) of pipelined processor to allow forwarding from MEM/WB register to input of MEM stage. Highlight the new signals added if any and briefly explain their functionality.

(Assume that the stall condition for **lw** followed by data dependent instruction is modified to allow this forwarding) **[6+9=15M]**



Solutions:

(i) Forward from MEM/WB register to input of MEM stage.

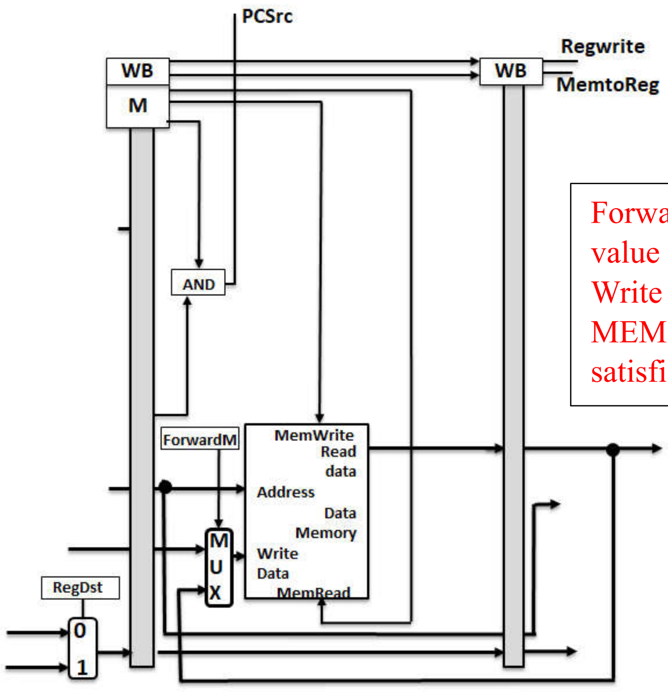
If **(MEM/WB.MemRead = =1)** //to check for load instruction, alternatively (MEM/WB.MemtoReg==1 and MEM/WB.Regwrite= =1)

and (EX/MEM.MemWrite= =1) //to check if load instruction is followed by store

and (MEM/WB.RegisterRd = =EX/MEM.RegisterRd) //to check if lw and sw Rt fields match

and (MEM/WB.RegisterRd != $Zero)

(ii)



ForwardM signal passes the updated value from MEM/WB register to the Write Data port if the conditions for MEM to MEM forwarding are satisfied