Birla Institute of Technology and Science Pilani
CS F342 Computer Architecture
Second Semester 2017-18
Multi-Cycle Processor Design

Weightage: 12.5% **Marks:** 25 marks
Date of Announcement: 11th March 2018 **Date of Submission:** 07th April 2018

Project Statement

The project is about design, implementation and testing of a 16-bit multi-cycle RISC processor. Each instruction is of 16-bit (2 bytes) and passes through **five cycles** IF, ID, EX, MEMR and WB similar to the MIPS processor discussed in class. However, certain instructions can be completed in less than five cycles also. But, the longest instruction shall take five cycles.

Cycle-1 IF: Instruction Fetch

Cycle-2 ID: Instruction Decode

Cycle-3 EX: Execute

Cycle-4 MEM: Memory operation (data memory)

Cycle-5 WB: Write Back

The processor can be visualized as the following sub modules.

Instruction Memory

Data Memory

Register File

Arithmetic and Logic Unit (ALU)

Control unit.

Instruction Memory: It has a size of 16kB. One word (16-bits, 2 Bytes) can be read from this at a time. Use suitable control signals for this module.

Data Memory: It has a size of 16kB. One word (16-bits, 2 Bytes) can be read from this at a time. Use suitable control signals for this module.

Register File: There a total of 16 registers (R0 to R15), each 16-bit. A few registers are hard coded to constants. Example R0 is hardcoded to zero.

Arithmetic Logic Unit (ALU): It is a 16-bit digital circuit capable of doing arithmetic and logic operations. The list of operations is listed in the instruction set.

Control Unit: This is a finite state machine (FSM). Design the FSM to activate the suitable elements on the data path.

Instruction Set:

1. **Addition:** Register, Immediate
2. **Subtraction:** Register, Immediate
3. **Shift Logical:** Left, right
4. **Load/ Store:** Immediate, Register
5. **Jump:** Immediate, Register
6. **Branch:** Equal, Not-equal
7. **Logical:** Nand , Or

Addition: (3 cycles)
Register addressing: RD = RS1 + RS2

Opcode	Destination	Source-1	Source-2
1000	Reg. No. 4-bit	Reg. No. 4-bit	Reg. No. 4-bit

Immediate addressing: RD = RD + Sign-Extended-Immediate Data

Opcode	Destination	Immediate data
1001	Reg. No. 4-bit	8-bit data

Example: If the opcode is 1001001110010000 then it implies, R3 = R3 + FF90H

Immediate addressing: RD = RD + Immediate Data (upper byte filled by zeros)

Opcode	Destination	Immediate data
1010	Reg. No. 4-bit	8-bit data

Example: If the opcode is 1010001110010000 then it implies, R3 = R3 + 0090H

Subtraction: (3 cycles)
Register addressing: RD = RS1 - RS2

Opcode	Destination	Source-1	Source-2
1100	Reg. No. 4-bit	Reg. No. 4-bit	Reg. No. 4-bit

Immediate addressing: RD = RD - sign-Extended-Immediate Data

Opcode	Destination	Immediate data
1101	Reg. No. 4-bit	8-bit data

Immediate addressing: RD = RD - Immediate Data (upper byte filled by zeros)

Opcode	Destination	Immediate data
1110	Reg. No. 4-bit	8-bit data

Shift Operations (3 cycles)
Shift Left Logical: RD = shift logical left (fill zeros) RD by the immediate number specified

Opcode	Destination	Immediate Data	Func. field
0000	Reg. No. 4-bit	Reg. No. 4-bit	**0001**

Shift Right Logical: RD = shift logical right (fill zeros) RD by the immediate number specified

Opcode	Destination	Immediate Data	Func. field
0000	Reg. No. 4-bit	Reg. No. 4-bit	**0010**

Shift Arithmetic Right: RD = shift right (copy MSB) RD by the immediate number specified

Opcode	Destination	Immediate Data	Func. field
0000	Reg. No. 4-bit	Reg. No. 4-bit	**0011**

Logical (3 cycles)
Logical Nand register addressing: RD = RS1 ***nand*** RS2

Opcode	Destination	Source-1	Source-2
1011	Reg. No. 4-bit	Reg. No. 4-bit	Reg. No. 4-bit

Logical or register addressing: RD = RS1 ***or*** RS2

Opcode	Destination	Source-1	Source-2
1111	Reg. No. 4-bit	Reg. No. 4-bit	Reg. No. 4-bit

Logical Nand Immediate: RD = RD nand (sign extended immediate data)

Opcode	Destination	Immediate data
0111	Reg. No. 4-bit	8-bit data

Logical or Immediate: RD = RD or (sign extended immediate data)

Opcode	Destination	Immediate data
0110	Reg. No. 4-bit	8-bit data

Branch Instructions (3 cycles)
Branch Equal: branch to the address in register RT when RA and RB are equal

Opcode	Target Add. (RT)	RA	RB
0100	Reg. No. 4-bit	Reg. No. 4-bit	Reg. No. 4-bit

Branch Not Equal: branch to the address in register RT when RA and RB are unequal

Opcode	Target Add. (RT)	RA	RB
0101	Reg. No. 4-bit	Reg. No. 4-bit	Reg. No. 4-bit

Jump (3 cycles) Target address to be calculated in 3rd cycle using the ALU.
Jump to PC + sign extended immediate data

Opcode	Immediate address
0011	12-bit address PC relative

Example: If the value of PC at which JMP instruction is located is 1200H and the opcode is 3200H then after the jump, PC will point to 1202H (incremented in 1st cycle)+0200H (sign extended immediate data).

Load Word (5 cycles)
load RD from data memory, whose location is pointed out by RP + sign- extended immediate data left shifted by one bit. RD can be one among R12, R13, R14 and R15 and hence the upper 2-bits (11) of the register id are not specified. RP (pointer) can be one among R8, R9, R10 and R11 only, hence upper two bits (10) are not specified.

Opcode	RD	RP	Immediate
0001	Reg. No. 2-bit	Reg. No. 2-bit	8-bit

Example: if the opcode is 1025H, it is a load instruction.
R12 is to be loaded from data memory.
Address is given by: R8 + 0050H

Store Word (4 cycles)

Store contents of RD to data memory location pointed out by RP + sign- extended immediate data padded with a LSB 0. RD can be one among R12, R13, R14 and R15 and hence the upper 2-bits (11) of the register id are not specified. RP (pointer) can be one among R8, R9, R10 and R11 only, hence upper two bits (10) are not specified.

Opcode	RD	RP	Immediate
0010	Reg. No. 2-bit	Reg. No. 2-bit	8-bit

Example: if the opcode is 2CA5H, it is a store instruction.
R15 is to be copied into data memory location whose address is.
Address is given by: R8 + FF4AH (1111 1111 0100 1010)

****** The End ******