Assignment – 2

Due: Thu. 7 September at 1:00 PM

Submit a printed copy of your answers to Engineering Student Services Stall, with a cover sheet (created on Canvas).

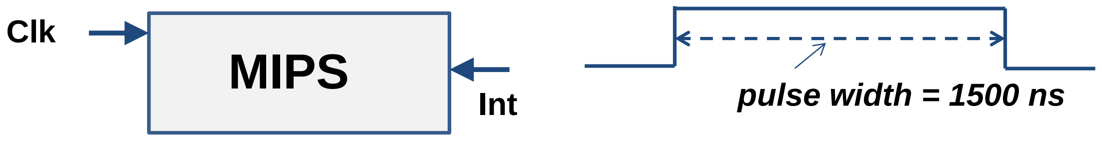
1. As shown below, input ***Int*** of the MIPS processor (with multi-cycle implementation) is connected to an external device, which generates single pulses with variable lengths. **An external interrupt** is generated when the rising edge or falling edge of the pulse occurs. Also, a programmable counter is available which is incremented by 1 in every clock cycle. Two memory addresses are used to access this counter, which are indicated as ***count_prog*** and ***count_read***. Writing 0x00000000 in ***count_prog*** will initialize this counter to 0 and then it starts counting from the next clock cycle. Writing 0xFFFFFFFF in ***count_prog*** will stop the counter and then the count value (which indicates the number of clock cycles since it has been initialized) can be read from ***count_read***. (Note that ***count_pog*** and ***count_read*** are symbols which represent two memory addresses.)

   Write the required assembly code to measure the pulse width, (which is indicated as a number of 10 ns cycles). For example, if the pulse width is 4250 ns, your code should write 425 in memory location indicated as **pulse_width**. (The minimum pulse width is assumed to be 1000 ns so up to 100 ns inaccuracy in the measurement will be acceptable if it happens).

   Also assume that the rising edge happens first and then the interrupt is generated. When the falling edge occurs an interrupt will be generated again.

   The processor's clock frequency is 100 MHz. (Assume that the user program is an infinite loop while waiting for the interrupts).

   (Note that, in this multi-cycle implementation, load instructions take 5 clock cycles, control transfer instructions take 3 clock cycles and other instructions take 4 clock cycles to complete).



Other pins of the processor and also memory and programmable counter are not shown in this Figure.

2. Different implementations of the ARM instruction set architecture are used in mobile devices. The purpose of this question is to get an experience about a different processor ISA. In this exercise, you are given some ARM assembly codes as indicated below:

i		LDR	r0, #1024	; load base address of table
		LDR	r1, #200	; initialize loop counter
		EOR	r3, r3, r3	; clear r3 (using r3 XOR r3)
	L1:	LDR	r4, [r0]	; get first addition operand
		ADD	r3, r3, r4	; add to r2
		ADD	r0, r0, #4	; increment to next table element
		SUBS	r1, r1, #1	; decrement loop counter
		BNE	L1	; if loop counter is not 0, go to L1
ii		ROR	r1, r2, #4	; r1 = $r2_{3:0}$ concatenated with $r2_{31:4}$

Answer the following questions for both ***i*** and ***ii*** separately:

(a) For the above ARM assembly code write an equivalent MIPS assembly code.

(b) Assuming that the average CPI of the MIPS assembly code is the same as the average CPI of the ARM assembly code, and the MIPS processor has a clock frequency that is 1.5 times faster than the ARM processor. Determine how much faster (or slower) is this ARM processor compared to the given MIPS processor.

3. The CPI for each group of instructions for processors M1 and M2 (which have multi-cycle implementations) are given as follows:

M1: Loads/Stores (4 cycles), ALU operations (3 cycles), branch and jump instructions (3 cycles)

M2: Loads (5 cycles), Stores (4 cycles), ALU operations (4 cycles), branch and jump instructions (3 cycles)

M3: is a five-stage pipelined implementation.

Consider an application which has 10^{10} MIPS instructions with the following instruction mix: Loads 20%, Stores 10%, Conditional Branches 10%, Jumps 5% and R-type instructions 55%.
The CPU clock frequency for M1, M2 and M3 are 0.8 GHz, 1 GHz and 1.25 GHz respectively.

(a) Compute the time to execute these instructions on processor M1.

(b) Compute the time to execute these instructions on processor M2.

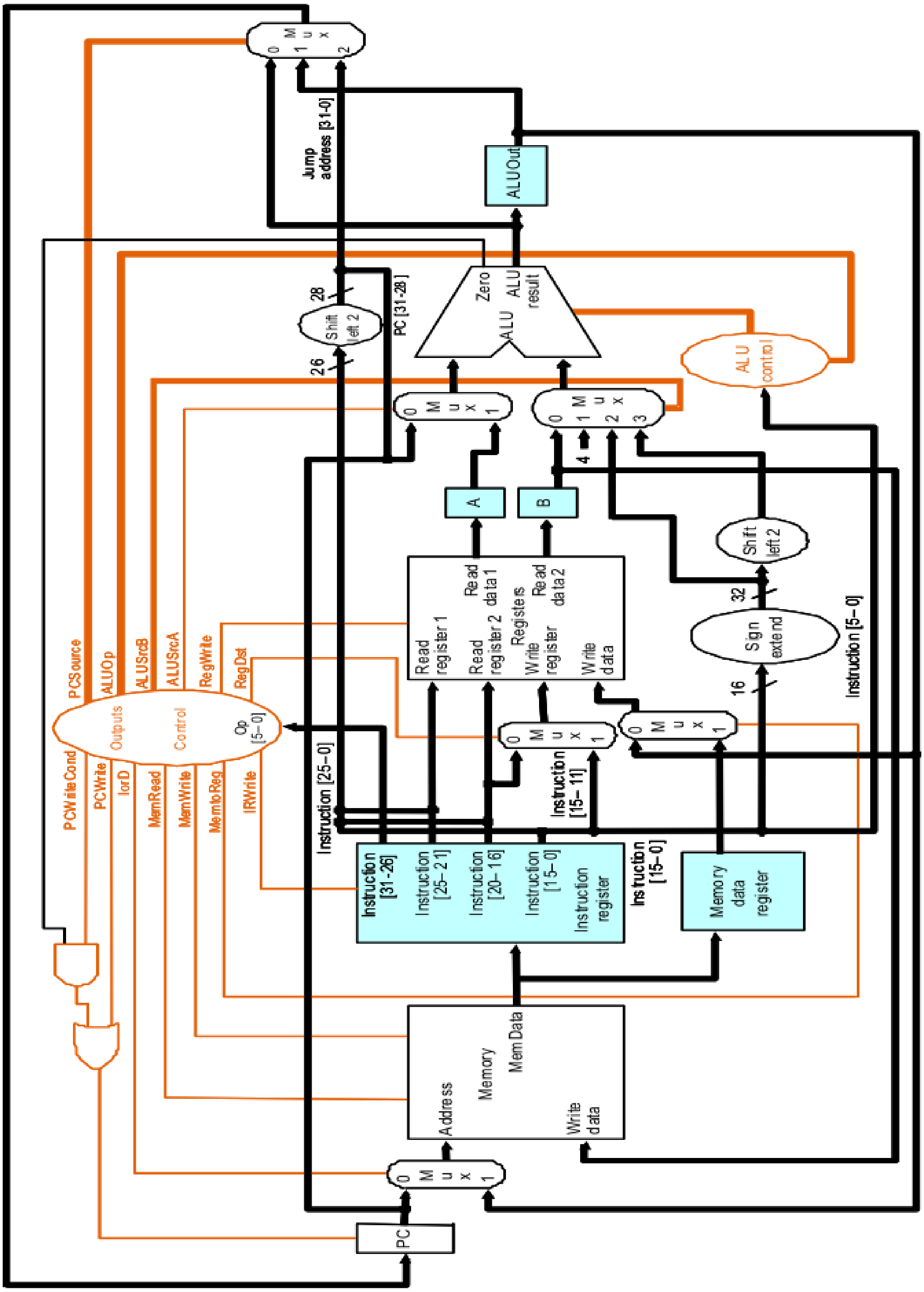
(c) Compute the time to execute these instructions on processor M3.

4. Write the MIPS instructions which are part of an interrupt service routine to enable further interrupts.

5. Explain how you can extend the datapath and control of the given implementation (shown in Fig. 1 on the page 4) to include instruction ***jal***.

   (Make modifications on the diagram if necessary and/or explain the required changes).

Fig 1 – Multi-cycle implementation of Datapath


PC
Mux 0 1
Address
Memory
MemData
Write data
Instruction [31-26]
Instruction [25-21]
Instruction [20-16]
Instruction [15-0]
Instruction register
Memory data register
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
A
B
Sign extend
16
32
Shift left 2
Shift left 2
26
28
PC [31-28]
Jump address [31-0]
ALU
Zero
ALU result
ALUOut
ALU control
4
Instruction [25-0]
Instruction [15-11]
Instruction [15-0]
Instruction [5-0]
Control
Outputs
Op [5-0]
PCSource
ALUOp
ALUSrcB
ALUSrcA
RegWrite
RegDst
PCWriteCond
PCWrite
IorD
MemRead
MemWrite
MemtoReg
IRWrite