Assignment – 2

Due: Wed. 31 August at 1:00 PM

Sample answers for questions

1. (a) How long does it take to execute one hundred million instructions from a program with the following instruction mix if the clock frequency is **1 GHz** for the implementation given in Figure 2?

   Instruction Mix: 30% loads, 10% stores, 50% ALU instructions, 10% conditional branches

   (b) Write the value of each control signal (for the implementation shown in Figure 2) in each cycle of executing instruction **and $8, $9, $10**

(a) Figure 2 indicates the multi-cycle implementation and the no. of cycles for each type of instructions are as follows:

Loads 5 cycles, stores 4 cycles, ALU instructions 4 cycles, branches 3 cycles

Given the instruction mix, the average cycles per instructions will be:
CPI = 0.3 * 5 + 0.1 * 4 + 0.5 * 4 + 0.1 * 3 = 4.2

CPU execution time = IC * CPI * T = $100 * 10^6 * 4.2 * 10^{-9} = 0.42$ Seconds

(b) ***and*** instruction takes 4 cycles to complete. The control signals for each cycles are:

Cycle 1:
MemRead = 1, IRWrite = 1, PCWrite = 1, ALUSrcA = 0, ALUSrcB = 01, ALUOp = 00, IoD = 0, PCSource = 00

Cycle 2:
ALUSrcA = 0, ALUSrcB = 11, ALUOp = 00

Cycle 3:
ALUSrcA = 1, ALUSrcB = 00,ALUOp = 10

Cycle 4:
RegWrite = 1, MemtoReg = 0, RegDst = 1

2. Describe the effect that a single stuck-at-0 fault (i.e. regardless of what it should be, the signal is always **0**) would have for the signals shown below, in single-cycle datapath shown in Figure 1. Which instructions if any, will not work correctly? Explain why?

   Consider each of the following faults separately:
   a. RegWrite = 0
   b. ALUop0 = 0
   c. ALUop1 = 0
   d. Branch = 0
   e. MemRead = 0
   f. MemWrite = 0

a. RegWrite = 0
Instructions which need to write the result in destination register will not operate properly. These are R-type instructions and LW instructions (from our subset of instructions R-type, LW, SW, beq, which were considered for implementation).

b. ALUop0 = 0
For beq instruction, ALUop0 must be 1 in order to use the ALU for subtract to compare the equality of its two operands. So, in this case, beq cannot work correctly.

c. ALUop1 = 0
For R-type instructions, ALUop1 must be 1 in order to use the ALU properly. So, in this case, R-type instructions cannot work correctly.

d. Branch = 0
For beq instruction, Branch control signal must be 1. So, in this case, beq cannot work correctly.

e. MemRead = 0
For LW instruction, MemRead control signal must be 1 in order to read a word from data memory. So, in this case, LW cannot work correctly.

f. MemWrite = 0
For SW instruction, MemWrite control signal must be 1 in order to write a word into data memory. So, in this case, SW cannot work correctly.

3. The critical path in a datapath is the path, which has the longest delay (and therefore it may affect the maximum clock frequency of the system). Assume that the logic blocks used to implement the single-cycle datapath (shown in Figure 1) have the following latencies in **pico seconds** (ps):

	I-Mem	Add	Mux	ALU	Regs	D-Mem	Sign-Extend	Shift-left-2	ALU ctrl
a)	400	100	30	120	200	350	20	0	50
b)	500	150	100	180	220	1000	90	20	55

What is the ***maximum clock frequency*** in each of the above cases?

Note: To get the full mark for this question you only need to calculate the time needed for LW instruction as it takes the longest time in this implementation. The numbers below were calculated using a diagram, which also includes the implementation of *j* instruction (as shown on page 7).

In order to determine the maximum clock frequency in our single cycle implementation, the time required to complete processing each type of instructions should be calculated.

Note that the "add" for PC+4 is done at the same time as accessing I-Mem. So, we should notice from the given timing delays the critical path is not the path to update the PC as the memory access times are much longer than the time required for other components. Also, no specific timing is given for the control unit so we should assume that the value of each control signal is available (i.e. has its stable value of 0 or 1 depending on the type of instruction) at the right time needed for the specific component.
The following timing calculation is done only for the longest path for each instruction type in our single-cycle implementation (as shown in Figure on page 7) .

a) **LW:** I-Mem + Regs + Mux + ALU + D-Mem + Mux + Mux+ Regs = 400 + 200 + 30 + 120 + 350 + 30 + 30 + 200 = 1360 ps
SW: I-Mem + Regs + Mux + ALU + D-Mem = 400 + 200 + 30 + 120 + 350 = 1100 ps
R-Type: I-Mem + Regs + Mux + ALU + Mux + Mux+ Regs = 400 + 200 + 30 + 120 + 30 + 30 + 200 = 1010 ps
BEQ: I-Mem + Regs + Mux + ALU + Mux + Mux = 400 + 200 + 30 + 120 + 30 + 30 = 810 ps
J: I-Mem + Shift-left-2 + Mux = 400 + 0 + 30 = 430 ps

So, the max clock frequency is determined by LW which is 1/1360 = 736 MHz

b) **LW:** I-Mem + Regs + Mux + ALU + D-Mem + Mux + Mux+ Regs = 500 + 220 + 100 + 180 + 1000 + 100 + 100 + 220 = 2420 ps
SW: I-Mem + Regs + Mux + ALU + D-Mem = 500 + 220 + 100 + 180 + 1000 = 2000 ps
R-Type: I-Mem + Regs + Mux + ALU + Mux + Mux+ Regs = 500 + 220 + 100 + 180 + 100 + 100 + 220 = 1420 ps
BEQ: I-Mem + Regs + Mux + ALU + Mux + Mux = 500 + 220 + 100 + 180 + 100 + 100 = 1200 ps
J: I-Mem + Shift-left-2 + Mux = 500 + 20 + 100 = 620 ps

So, the max clock frequency is determined by LW which is 1/2420 = 414 MHz

4. Explain how you can extend the datapath and control of the given implementation (shown in Fig. 1 on page 3) to include instruction ***jr rs***.

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

The contents of register ***rs*** should be used as the target address to update the PC to fetch instruction in the next clock cycle. So, one way is to extend the 2-input multiplexer (which is connected to the input of PC) to become a 3-input multiplexer where the third input comes from the output of register file (Data Read 1) or to add a new 2-input multiplexer as shown in the diagram. An additional control signal should be generated (which is shown as ***jr*** in the diagram) to select the proper input of the new multiplexer when the current instruction is ***jr rs***.

Note: Changes are shown in blue and red colours on page 5 (Fig. 1X).

5. An exception will occur when one of the following instructions is executed on the multi-cycle datapath.

   (a) Explain which instruction generates an exception and why?
   (b) Explain what happens when the exception occurs and how the problem can be fixed.

```
…
lui     $19, 0x7fff
ori     $19, $19, 0xffff
add     $24, $19, $19
lui     $20, 0x0200
ori     $20, $20, 0x8000
sw      $24, 0($20)
…
```

(a) An overflow exception occurs when instruction ***add $24, $19, $19*** is executed (as 0x7fffffff is the biggest positive number which can be represented as 32-bit signed value and adding that to another number will generate overflow exception).
(b) When the exeptin occurs, the address of this instruction will be copied into **EPC** control register (in co-processor 0) and after the last clock cycle of this instruction the control is transferred to the **exception handler address** (0x80000080 or 0x80000180 depending on the implementation) in the kernel. (The effect of this instruction should be canceled so the **exception handler code** can generate error message or channging EPC to skip this instruction or replacing that with addu instrution. *The way this problem can be fixed in general is implementation dependent*).

Fig 1X – **Modified** Single-cycle implementation of Datapath

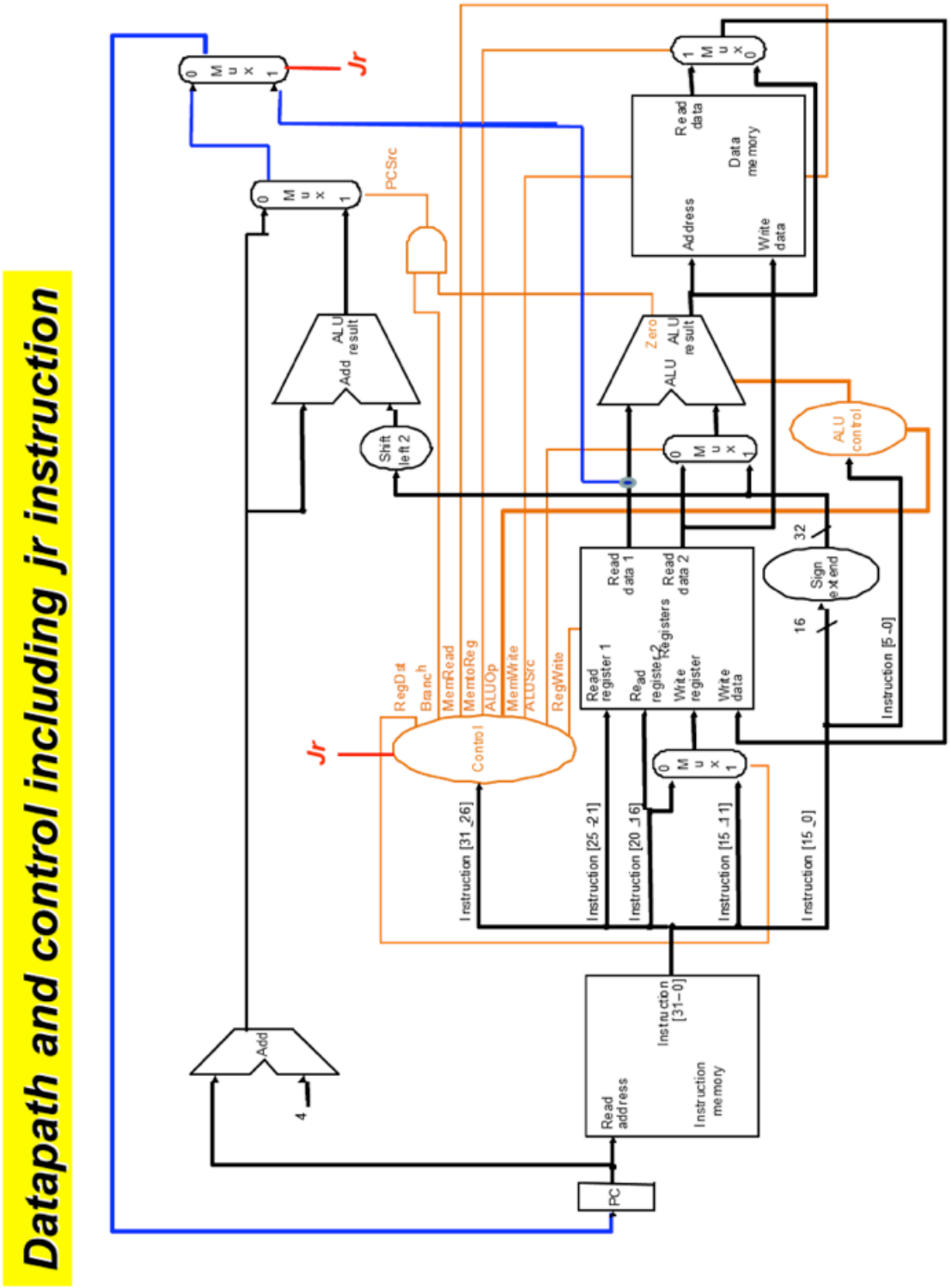
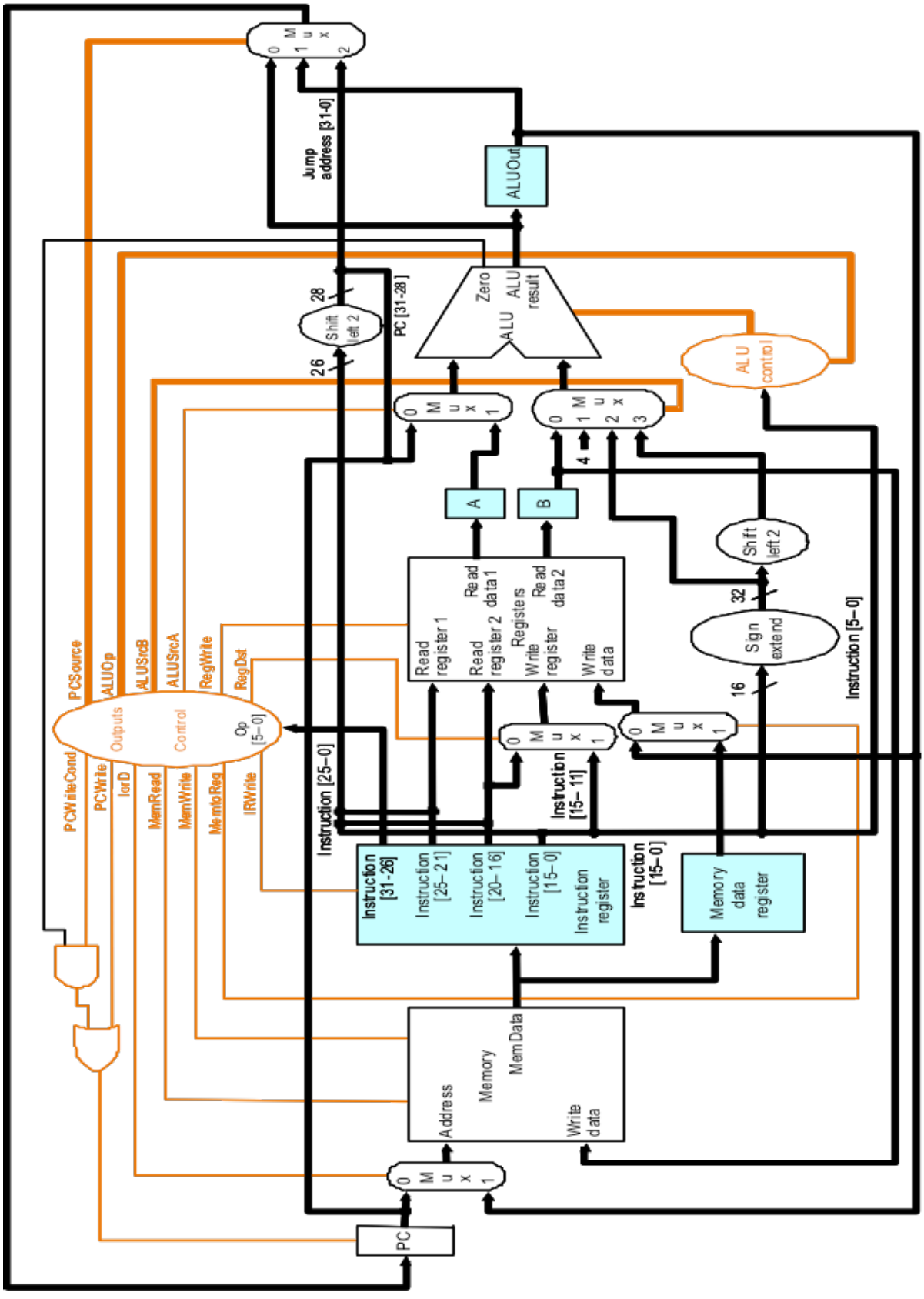
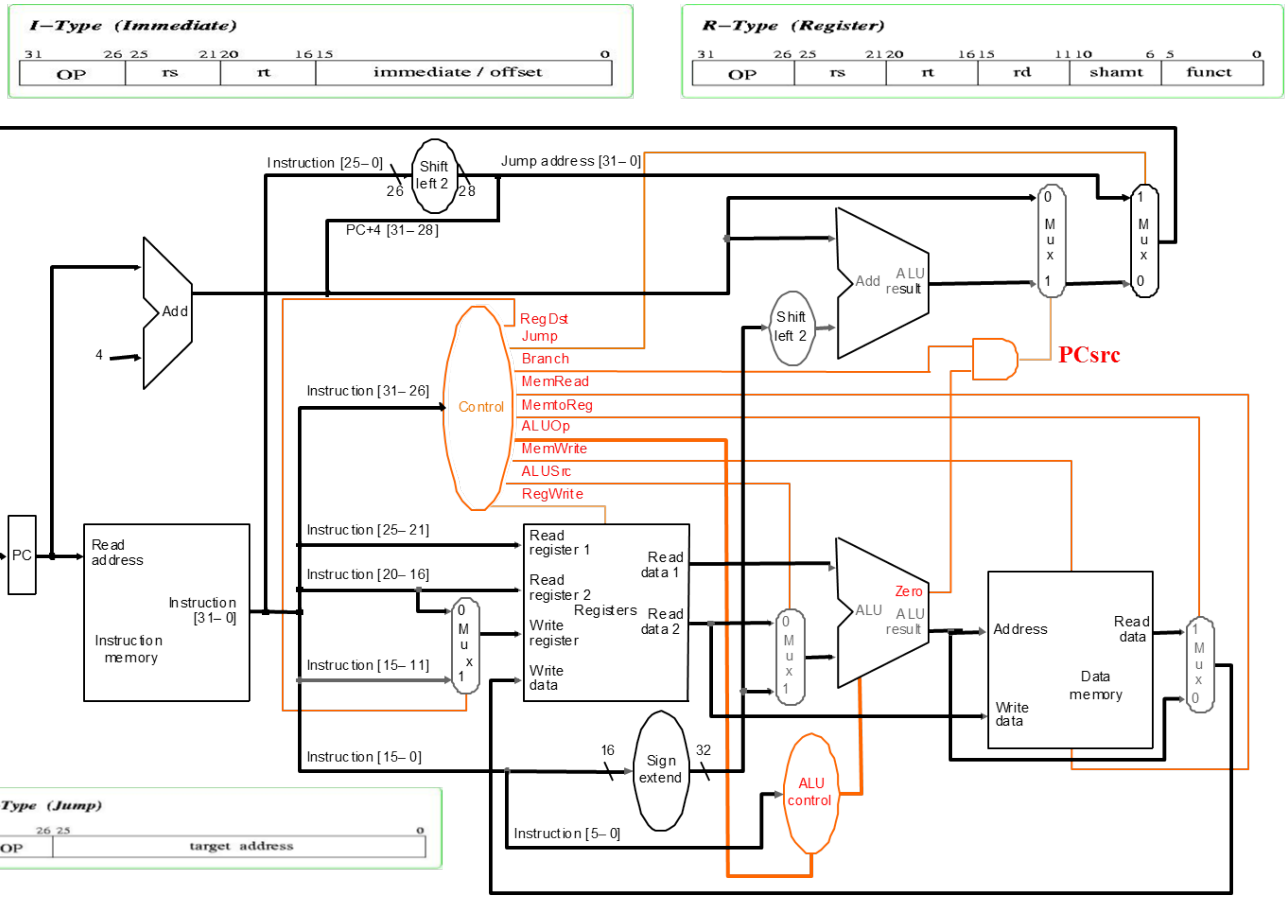


Fig 2 – Multi-cycle implementation of Datapath




I–Type (Immediate)
31 26 25 21 20 16 15 0
OP rs rt immediate / offset
R–Type (Register)
31 26 25 21 20 16 15 11 10 6 5 0
OP rs rt rd shamt funct
Instruction [25–0]
Shift left 2
26
28
Jump address [31–0]
PC+4 [31–28]
Add
4
RegDst
Jump
Branch
MemRead
MemtoReg
ALUOp
MemWrite
ALUSrc
RegWrite
Control
Instruction [31–26]
Shift left 2
Add
ALU result
0 M u x 1
1 M u x 0
PCsrc
PC
Read address
Instruction [31–0]
Instruction memory
Instruction [25–21]
Instruction [20–16]
Instruction [15–11]
Instruction [15–0]
Instruction [5–0]
Read register 1
Read register 2
Registers
Write register
Write data
Read data 1
Read data 2
0 M u x 1
16
Sign extend
32
ALU control
Zero
ALU
ALU result
Address
Read data
Data memory
Write data
1 M u x 0
J–Type (Jump)
31 26 25 0
OP
target address