CS223: Computer Architecture and Organization

Mid-semester Examination

Time: 2 hours

Q1. Consider a machine (clock rate 1 GHz) with three instruction classes (A, B, and C) and cycles per instruction (CPI) measurement as follows:

Instruction class	CPI
A	1
B	2
C	3

We measure the code for the same program from two different compilers and obtain the following data

Code from	Instruction counts (in billions)		
	A	B	C
Compiler-1	5	1	1
Compiler-2	10	1	1

Which code sequence will execute faster according to million instructions per second (MIPS)? According to the execution time? [5+5]

CPU clock cycles (Compiler 1) = $(5x1+1x2+1x3)10^9 = 10^{10}$

CPU clock cycles (Compiler 2) = $(10x1+1x2+1x3)10^9 = 15x10^9$

Execution time (Compiler 1) = $10^{10}/10^9$ = 10sec

Execution time (Compiler 2) = $15x10^9/10^9$ = 15sec (5 mark)

MIPS = IC/(Execution time x 10^6)

MIPS (Compiler 1) = $((5+1+1)x10^9)/(10x10^6) = 700$

MIPS (Compiler 2) = $((10+1+1)x10^9)/(15x10^6) = 12000/15 = 800$ (5 mark)

Q2. Translate the following loop into C. Assume that the C-level integer *i* is held in register $t1, $s2 holds the C-level integer called *result*, and $s0 holds the base address of the integer array *A*.

```
        addi $t1, $0, $0        // i = 0
LOOP:   lw $s1, 0($s0)          // s1 = A[0]
        add $s2, $s2, $s1       // result +=  s1
        addi $s0, $s0, 4        // s0 += 4
        addi $t1, $t1, 1        // i++
        slti $t2, $t1, 100      // t2 = (i < 100)  ? 1 : 0
        bne $t2, $0, LOOP       // if t2 != 0 goto LOOP
```

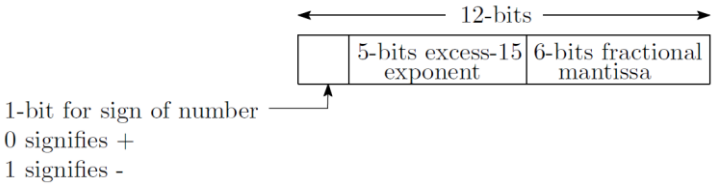
Rewrite the loop using five instructions. [3+7]

for (i = 0; i <100; i ++) result += A[i] (3 mark)

```
        addi $t1, $s0, 400
LOOP:   lw $s1, 0($t1)
        add $s2, $s2, $s1
        addi $t1, $t1, -4
bne $t1, $s0, LOOP
```

(7 mark)

Q3. Consider the following floating point representation in 12 bits that retains all the pertinent concepts of IEEE-754. The scale factor has an implied base of 2 and a 5-bit excess-15 exponent, with the two end values of 0 and 31 used to signify exact 0 and infinity, respectively. The 6-bit mantissa is normalized as in IEEE-754 format, with an implied 1 to the left of the binary point.



(a) Represent the numbers 1.7 and $\frac{1}{8}$ in this format. (1.5 each)

(b) What are the smallest and largest normalized positive numbers representable in this format? (1.5 each)

(c) Add two numbers of (a) (considering 12-bit floating point representations) and find the decimal value of the result. (4 mark) [3+3+4]

$1.7_{10} \sim 1.101100_2 \times 2^0$ E – 15 = 0 => E = 15 = $(01111)_2$ M = 101100 S = 0

$1/8 = 1.0 \times 2^{-3}$ E-15 = -3 => E = 12 = $(01100)_2$ M = 000000 S = 0

Smallest positive normalized no: S = 0, E = 1, M = 0 => 1.0x2-14

Largest positive normalized no: S = 0, E = 30, M = $(111111)_2$ => $(1.111111)_2 \times 2^{15} = 1.984375 \times 2^{15}$

To add, we need to equate the exponents by shifting smaller number right i.e., 0.001000×2^0

Now add mantissa

1.101100

0.001000

1.110100

Result = $1.110100 \times 2^0 = 1.8125_{10}$

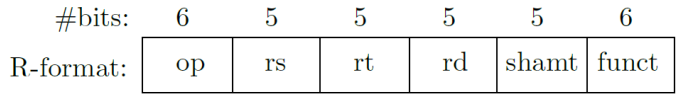
Q4. Consider the single cycle datapath (provided as a separate sheet). Fill the following table with appropriate control signal values (0, 1, X) for executing instructions.

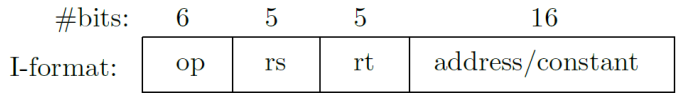
Instruction	*RegDst*	*ALUSrc*	*MemtoReg*	*RegWrite*	*MemRead*	*MemWrite*	*Branch*	*ALUop*
add	1	0	1	1	0	0	0	add
lw	0	1	0	1	1	0	0	add
sw	X	1	X	0	0	1	0	add
beq	X	0	X	0	0	0	1	subtract

Each row- 1.25 each, total –5 marks

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 on the multiplexors in the single cycle datapath. Which instructions (add, lw, sw, and beq), if any, would still work? Consider each of the following faults separately: RegDst = 0, ALUSrc = 0.

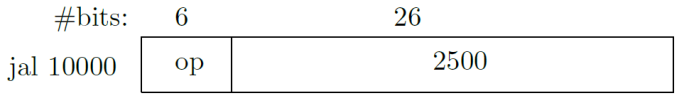
RegDst = 0: add will not work (2.5 mark) ALUSrc = 0: lw, sw will not work (2.5 mark)





[5+5]

Q5. We wish to add the instruction jal (jump and link) to the single cycle datapath (provided as a separate sheet). Add any necessary datapaths and control signals to the given datapath. Also provide each control signal values in your modified datapaths for executing jal. Instruction format of jal instruction is given below. (Note: $ra = 31) [5]

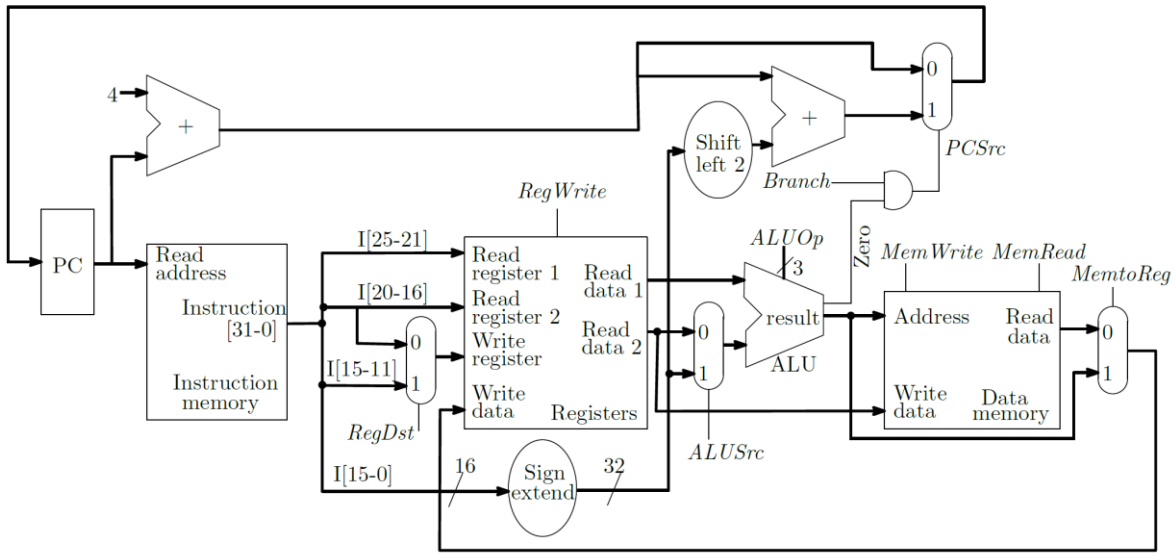


$ra = PC + 4$, goto Target Address

Target address $= (PC + 4)[31:28]:(2500 \times 4)$

3 corrections in the data path (3 mark)

Control signal for JAL (using the **table** above) (2 marks)

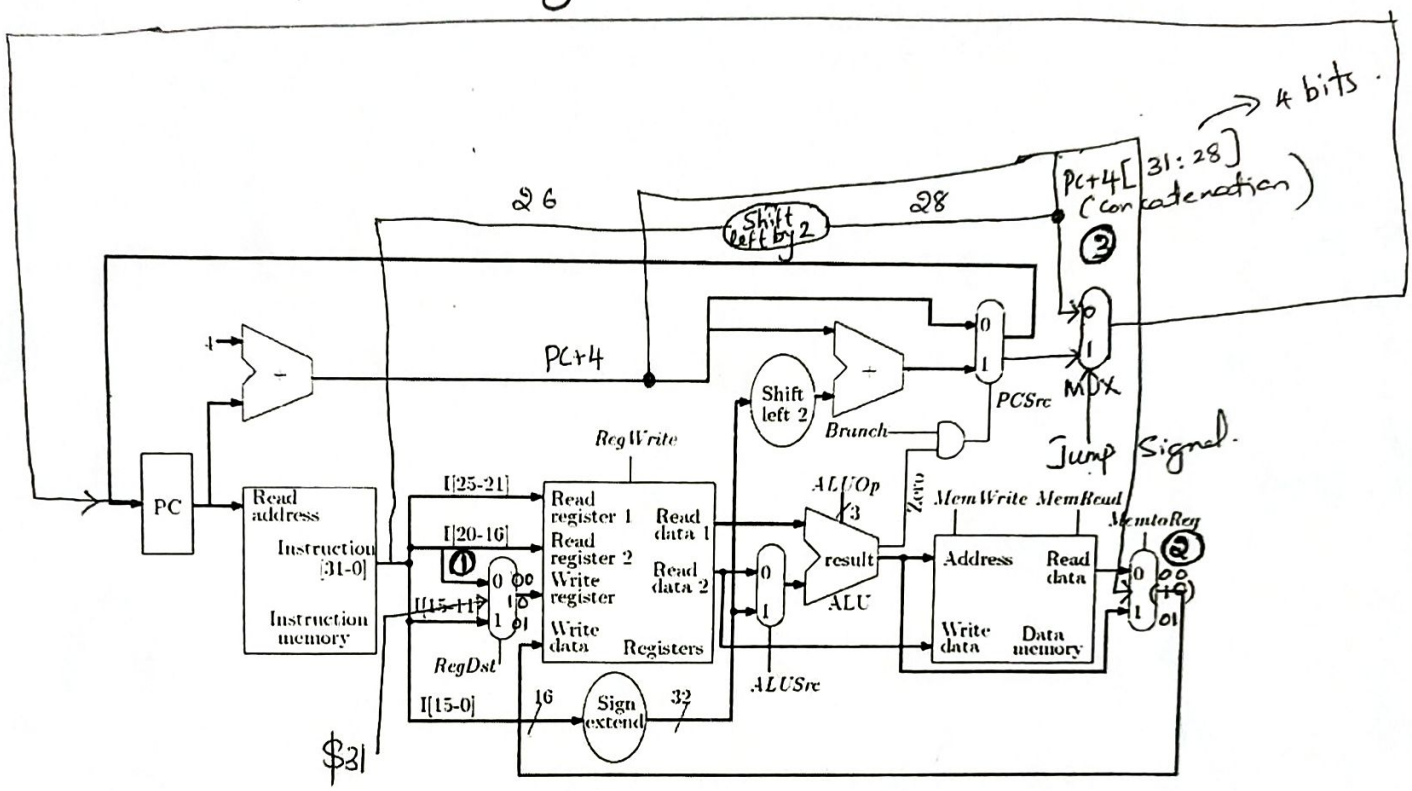

4
+
PC
Read address
Instruction [31-0]
Instruction memory
I[25-21]
I[20-16]
I[15-11]
I[15-0]
0
1
RegDst
RegWrite
Read register 1
Read register 2
Write register
Write data
Read data 1
Read data 2
Registers
16
Sign extend
32
Shift left 2
+
Branch
0
1
PCSrc
ALUOp
3
result
ALU
Zero
0
1
ALUSrc
MemWrite
MemRead
MemtoReg
Address
Read data
Write data
Data memory
0
1

CS223: COA Roll No:.....

Qst No : 5

R[31] ← PC+4

PC ← target Address.



(3 MARs, 1 each)

① ⇒ A new input to MUX passing the register, $31
[new input signal = 10]

② ⇒ MUX receives a new input, PC+4,
[new input signal = 10]

③ ⇒ A new MUX is added, with "jump" control signal
→ one input to MUX is the target address
→ output of MUX is passed as input to PC.

(2 marks)

Inst.	RegDst	ALUSrc	MemtoReg	RegWrite	MemRead	MemWrite	Branch	ALUOp	Jump
JAL	10	X	10	1	0	0	X	X	1