1. State whether each of the following are [T]rue or [F]alse next to the statement in the space provided OR fill in the blanks appropriately.

   (a) (1 point) The endianness of a machine is an important consideration when one character is to be stored and retrieved from memory. [F]

   (b) (1 point) The binary number 1111 is the minimum(most negative) number that you can represent in 2s complement representation of an integer using 4-bit signed representation. [F]

   (c) (1 point) In the IEEE Floating point representation the number of bits used to represent the mantissa decides the range of numbers that can be represented. [F]

   (d) (2 points) The decimal value represented by the 8 bit floating point number 11100111 (where the exponent is 3 bits) is -1.5

   (e) (1 point) A common expression for execution time is CPU time is

   ```
   y * cycle time * cycles per instruction
   ```

   where y is: (check the right option)

   1. the number of instructions in the program.
   2. the number of instructions executed.
   3. the average time for an instruction including cache misses.

   (f) (1 point) In the MIPS single cycle data path that we discussed in class, the `RegWrite` control is set to 1 for for *sw* (Store Word) instruction. [F]

   (g) (3 points) A machine has 3 classes of instructions with the following CPI for each class:

Instruction class	CPI
A	1
B	2
C	3

For a given high-level language program, two compilers produced the following executed instruction counts for each instruction class (in millions of instructions):

Code from:	A	B	C
Compiler 1	5	1	1
Compiler 2	10	1	1

The machine is assumed to run at a clock rate of 100 MHz. The MIPS rating of the machine according the two compilers are 70 and 80.

Comp1 : Avg CPI $= \frac{5\times1 + 1\times2 + 1\times3}{7} = 10/7$

$MIPS = \frac{freq}{CPI \times 10^6} = \frac{100\times10^6 \times 7}{10\times10^6} = 70$

Comp2 Avg CPI $= \frac{10\times1 \times 1\times2 + 1\times3}{12} = 15/12$

$MIPS = \frac{100\times10^6 \times 12}{15\times10^6} = 80$

2. (3 points) Suppose your manager asks you to speed up a program by a factor of 2 by enhancing a subroutine that the program spends half its time on. Would you take on the assignment? Justify quantitatively.

Apply Amdahl's law

$$2 = \frac{1}{(1-0.5) + \frac{0.5}{s}}$$

$$\Rightarrow s = 1/0 = \infty$$

So NO!

(2) (1)

Where s is how much the sub routine needs to be speeded up by.

3. (2 points) Suppose you were to implement a fused Multiply-Add instruction in MIPS that would allow us to do operations such as A = A ? B + C. Would you use one of the existing instruction formats (R, I or J) or would you need a new instruction format for this? Justify in one sentence!

(0.5) A = A * B + C. (1.5)

R type since we need only 3 registers to be specified.

4. (6 points) Your job is to translate the following C code into MIPS assembly.

```
struct Node {
      int data;
      struct Node *next;
};

int sumList (struct Node *nptr) {
       if (nptr == NULL) return 0;
        else return (nptr->data + sumList (nptr->next));
}
```

To help you out, a skeleton program has been given below. Your job is to fill out the boxes with MIPS Assembly code AND comments so that the program is a translation of the C code shown above to sum elements of a linked list. Do NOT write ANY code outside of the boxes. Only use instructions from the reference shown at the end of this paper.

```
sumList:
     addi $sp, $sp, -8          # allocate space on the stack
```

(1)
```
sw $a0, 0($sp)
sw $ra, 4($sp)        # Push a0 & ra to stack
```

```
     beqz  $a0, return0       # If (nptr == NULL) goto return0
```

(1.5)
```
lw $a0, 4($a0)        # get nptr -> next
```

(1)
```
jal sumList           # recursive call
```

```
     lw $a0, 0($sp)           # retrieve nptr
```

(1.5)
```
lw $a0, 0($a0)        # get nptr -> data
```

```
     add $v0 ,$a0, $v0        # add to result of recursive call
     j return
return0:
     li $v0, 0

return:
```

(1)
```
lw $ra, 4($sp)
jr $ra
```

5. More MIPS programming

(a) (3 points) Fill in the missing instruction with ONE R type instruction only in the box shown so that the C code is implemented by the MIPS assembly shown below. Only an exact solution will be accepted (missing or incorrect registers will be treated as incorrect).

C code: `if (t0 < 0) { t0 = 42; }`

Equivalent MIPS code assuming the registers have the same names as a C code variables:

```
slt $t1,$t0,0
        beq $t1,$0,end   // Yes, its $t1 and NOT a typo.
        ori $t0,$0,42
end:
```

No partial

(b) (2 points) How many cycles will this program take to get to finish? (Assume each instruction takes 1 cycle)

∞

```
start:
   jal middle
finish:
middle:
   jal last
   jr $ra
last:
   jr $ra
```

No partial.

(c) (4 points) A designer decides that having a branch-if-equal-to-register-plus-constant (beqc $t1 $t2 < *constant* >< *branchlabel* > means branch to label if contents of $t1 is equal to the sum of contents of $t2 and the constant) instruction is a great idea and implements it by using half of the immediate field for the constant as a MIPS I type instruction. Assuming each instruction takes one cycle, how many cycles would this new instruction save if used in the code below?

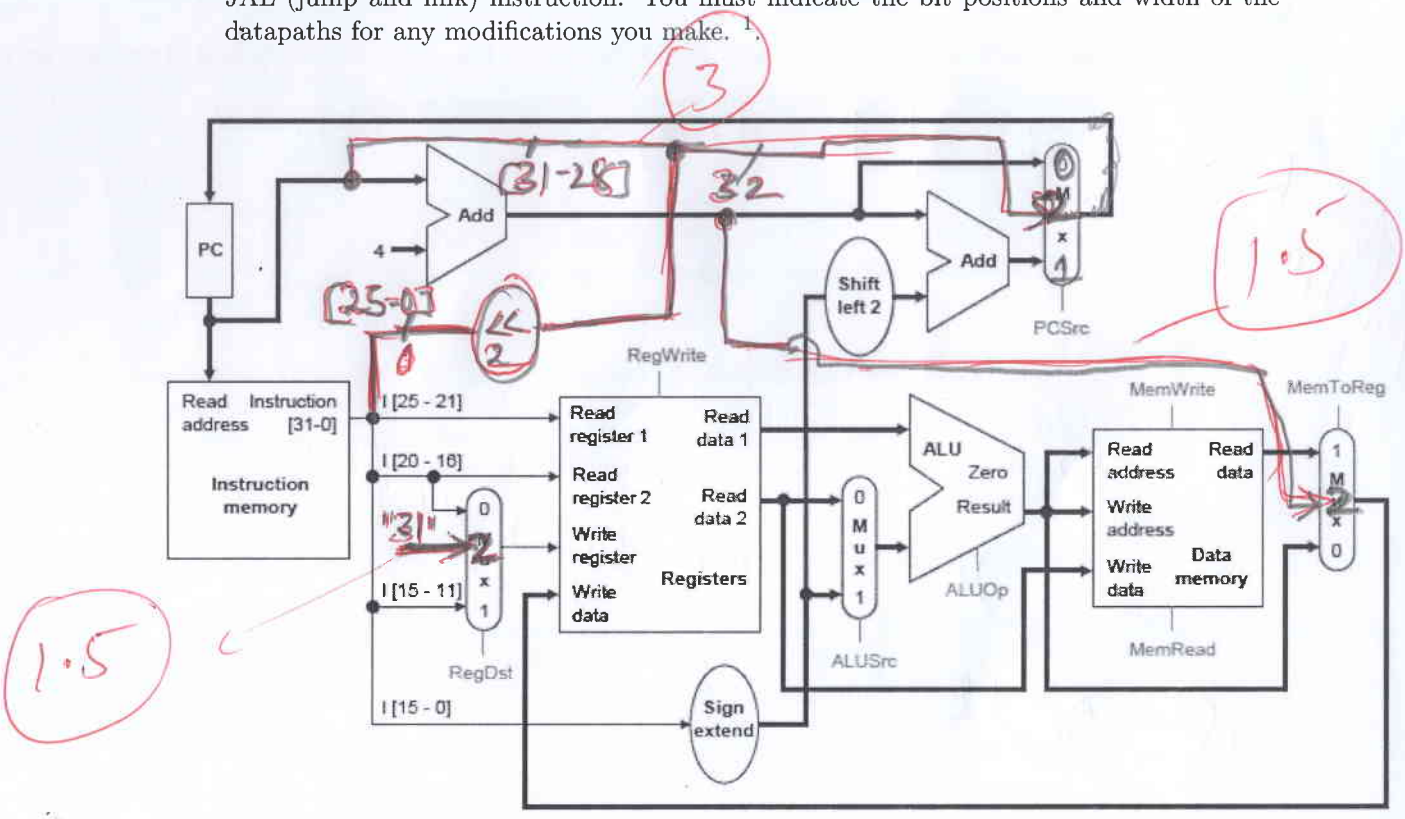
0

```
procedure:
   addi $t0, $a0, 255 // Check inputs to the procedure
   beq $t0, $a1, invalid // Branch based on the constant calculation
   addi $v0, $0r, 1 // Valid input, return 1
   jr $ra
invalid:
   addi $v0, $r0, 0 // Invalid input, return 0
   jr $ra
```

No partial

constant is a 8 bit qty. Cannot represent 255; which means I cannot use this new instruction.

6. (a) (6 points) Modify the MIPS single cycle data path in the figure to add support for the JAL (jump and link) instruction. You must indicate the bit positions and width of the datapaths for any modifications you make. [1].



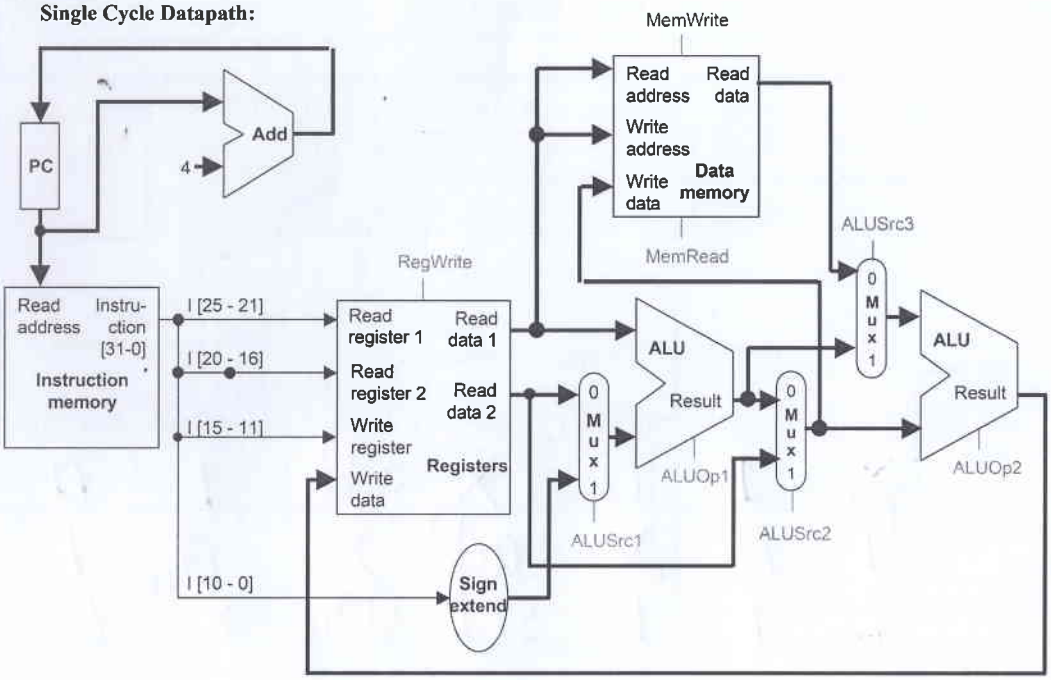
(b) (2 points) With the modified datapath, write down the values of the following control signals for executing the JAL instruction:

- RegDst
- RegWrite
- PCSrc
- MemToReg

[1]There is spare MIPS Datapath diagram on Page 9 in case you mess this one up. You MUST indicate which diagram we are to consider to marking. The default is the diagram on this page

7. Consider the single cycle datapath along with the instruction format shown below:

Single Cycle Datapath:



Field	op	rs	rt	rd	imm
Bits	31-26	25-21	20-16	15-11	10-0

It supports the following complex instructions:

```
lw_add rd, (rs), rt        # rd = Memory[R[rs]] + R[rt];
addi_st (rs), rs, imm   # Memory[R[rs]] = R[rs] + imm;
sll_add rd, rs, rt, imm # rd = (R[rs] << imm) + R[rt];
```

(a) (6 points) For each of the above instructions, specify how the control signals should be set for correct operation. Use X for don't care. ALUOp can be ADD, SUB, SLL, PASS_A, or PASS_B (e.g., PASS_A means pass through the top operand without change). Full points will only be awarded for the fastest implementation.

0.25×8=2

Inst	ALUsrc1	ALUsrc2	ALUsrc3	ALUop1	ALUop2	MemRead	MemWrite	RegWrite
lw_add	X	1	0	X	add	1	0	1
addi_st	1	0	X	add	X	0	1	0
sll_add	1	1	1	SLL	add	X	0	1

2 2 2

(b) (6 points) Given that the memory takes 3 ns, the ALU takes 4 ns and the Register file takes 2 ns with no other delays, compute the **minimum** time required to perform each instruction.

Instruction	Min Time	Justify
LW_add	14 ns	IMEM + Regfile + DMEM + ALU + RegWrite
addi_st	12 ns	IM + RF + ALU + DM
sll_add	15 ns	IM + RF + ALU + ALU + RFwrite

2 2 2

Spare Single Cycle MIPS Datapath Diagram for Question 6