 18/19 (ans avail)

1. **[12 marks]**
Study the following MIPS code, which has one input **$s0** and two outputs **$t0** and **$t1**.

```
    addi $t0, $zero, 32
    addi $t1, $zero, 32
L:  beq  $s0, $zero, N
    andi $t2, $s0  , 0x0001
    beq  $t2, $zero, E
    addi $t1, $t1  , -1
E:  addi $t0, $t0  , -1
    srl  $s0, $s0  , 1
    j    L
N:
```

```
t0 = 32
t1 = 32
while (s0 ≠ 0) {
    t2 = s0 & 1
    if (s0 & 1 == 1) {
        t1--;
    }
    t0--;
    s0 /= 2;
}
```

(a) What are the values of **$t0** and **$t1** at the end of the execution if the value of **$s0** at the start is 32? [2 marks]
$t1 = 31
$t0 = 26

(b) If the value of **$s0** is **43** at the start of execution, what is the total number of times both **beq** instructions branch? That is, when both "**beq $s0, $zero, N**" branches to **N** and "**beq $t2, $zero, E**" branches to **E**. [2 marks]
2

(c) Give a value of **$s0** at the start such that the values of **$t0** and **$t1** at the end of the execution are both 0. [2 marks]
0x FFFFFFFF

(d) What is the encoding of the only **R-format** instruction above in *hexadecimal*? [2 marks]
000000 10000 00000 10000 00001 000010 0x02008042 0x02008042.

(e) Write the relationship between **$t0** and **$s0** as well as between **$t1** and **$s0** in a *single sentence each*. [2 marks]
$t0 represents number of bits out of 32 bit number not utilized by $s0. $t1 represents total number of 0 bits in a 32 bit number $s0.

(f) Our current MIPS instruction set does not have **load half-word** since **lhw** is a pseudo-instruction. **lhw** loads 16 bits from memory to the *lower half* of the register and sets the *upper half* of the register to all zeros. The pseudo-instruction **lhw $t0, 80($zero)** will be translated into actual MIPS instructions before being run. Write down the equivalent actual instructions to perform **load half-word** in the fewest number of MIPS instructions possible. [2 marks]
lhw $t0, 80($0)
lui $t0, 0
lw $t0, 80($0)
lw $t0, 80($0)
srl $t0, $t0, 16.

2. **[4 marks]**
As the number -0.3_{10} cannot be represented precisely in binary, it also cannot be represented precisely in the IEEE 754 standard single precision floating point format. However, we can *approximate* the value by *truncating the bits* to the nearest representation.

Write the approximation of -0.3_{10} in IEEE 754 standard single precision floating point format. Give your answer in hexadecimal. [4 marks]
1.0100 1100 1100 1100 1100 1100 × 2^-2
1 01111101 00110011001100110011001
0xBD333333
0xBE999999

3. **[14 marks]**
You are given the implementation of MIPS processor on the next page with partially incorrect modification to include the Jump instruction (**j**). Note that for the added multiplexer (the one with control signal **IsJump?**), if **IsJump?** is 0, the top input is chosen; otherwise the bottom input is chosen.

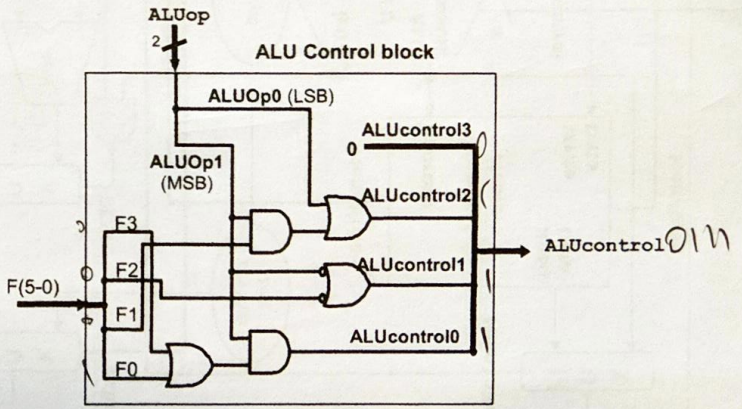
(a) Describe what is wrong with the implementation in one sentence. [2 marks]
Does not take into account of most significant bits of PC.

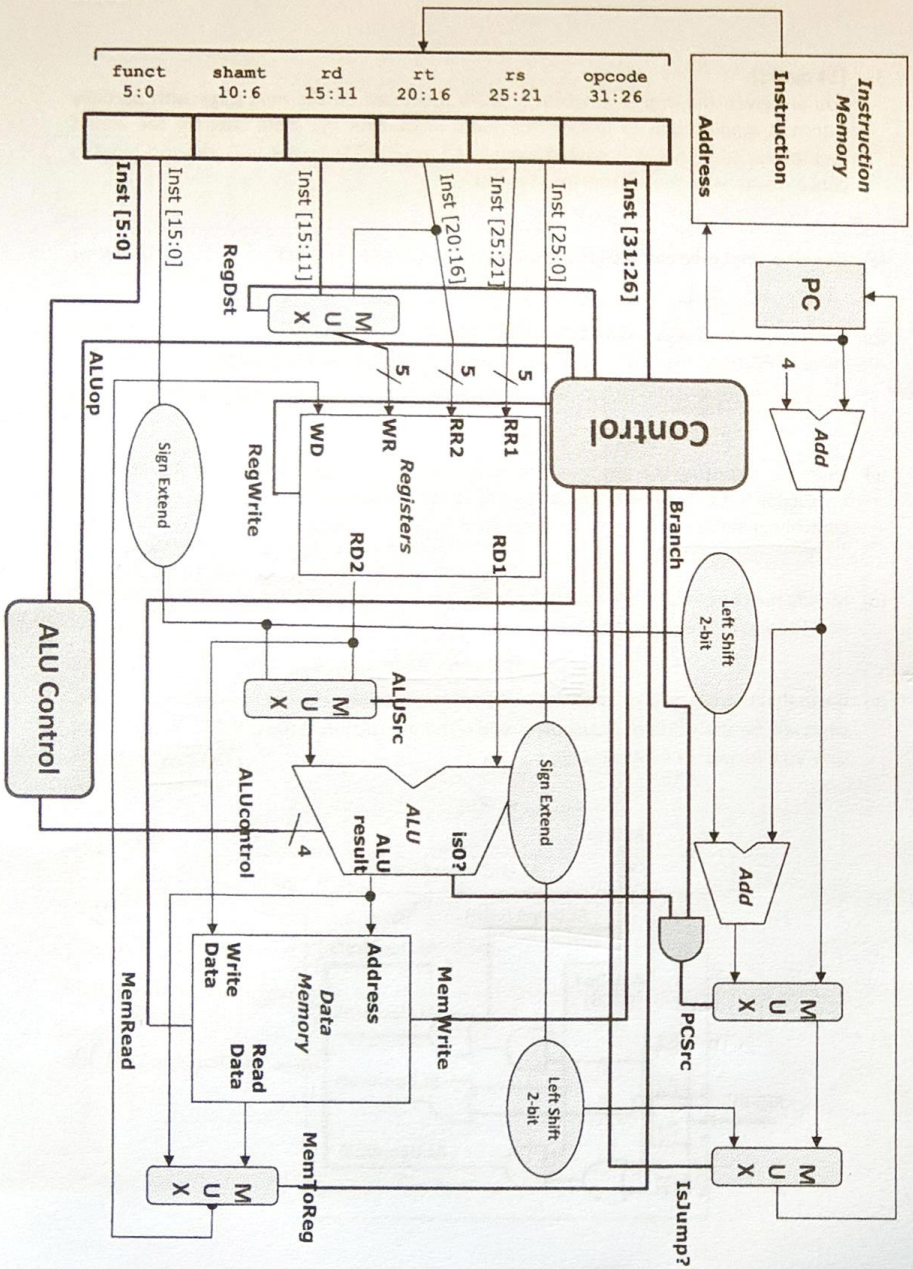
(b) Consider an instruction **0x0800C840** at address **0x2100FFFC**. What is the next value of PC when the instruction is executed using the incorrect processor above? [3 marks]
0000 1000 0000 0000 1100 1000 0100 0000
PC = 0x00032100

(c) Since we are using the intermediate signal **ALUop**, we specify that the **ALUop** for **j** instruction is **11**. The rest of the **ALUop** does not change. Fill in the missing values in the control signal table in the answer booklet. [3 marks]
000010
RegDst X, ALUSrc X, MtoR X, RegWrite 0, MemRead X, Branch X, IsJump 1, ALUOp 11

(d) Modify the combinational circuit given in the answer booklet to include **ALUop1**, **ALUop0** and **IsJump?** control signals. [4 marks]

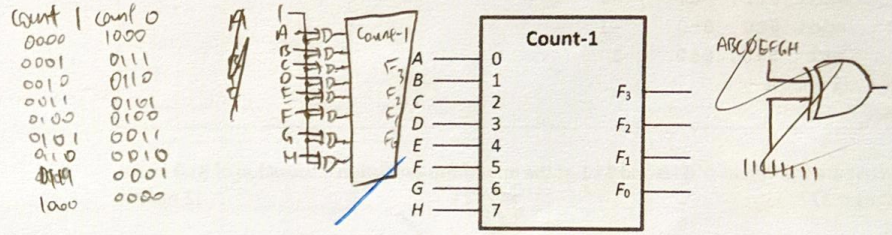
(e) Given that there is no change to the ALU Control unit shown below for your convenience, what will be the value of **ALUcontrol** when the instruction **0x08000031** is executed? Give your answer in 4 bits binary. [2 marks]
0111
0011 0001





5. **[22 marks]**
For the parts below, you are to assume that complemented literals are not available. Note also that circuit that is correct but uses more logic gates than necessary will be given partial credit.

(a) The **8-bit count-1** device, whose block diagram is shown below, takes in an 8-bit input *ABCDEFGH* and outputs $F_3F_2F_1F_0$ which is the number of 1s in the input. For example, if *ABCDEFGH* = 11101101, then $F_3F_2F_1F_0$ = 0110 (six).



How would you implement an **8-bit count-0** device to count the number of 0s in the input using the above 8-bit count-1 device and XOR gates? No other gates or devices besides the count-1 device and XOR gates are allowed. [3 marks]

(b) Assuming that the 8-bit input *ABCDEFGH* is an unsigned binary number. Let *P*(*A*,*B*,*C*,*D*,*E*,*F*,*G*,*H*) be a Boolean function that returns 1 if *ABCDEFGH* contains an odd number of 1s and *ABCDEFGH* is an even number, or returns 0 otherwise. For example, the function *P* returns 1 for the following inputs: 01110000, 10111010, 00010000, but returns 0 for the following inputs: 00111001, 10100001, 11110000.

Implement *P* using the **Count-1 device** as shown in part (a) above, with the fewest number of additional logic gates. [3 marks]

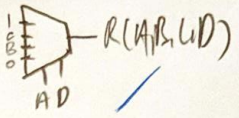
(c) Assuming that the 8-bit input *ABCDEFGH* is an unsigned binary number. Let *Q*(*A*,*B*,*C*,*D*,*E*,*F*,*G*,*H*) be a Boolean function that returns 1 if *ABCDEFGH* is a multiple of 17 (eg: 0, 17, 34, 51, etc.), or returns 0 otherwise.

Given a **parallel adder**, a **magnitude comparator**, a **decoder**, an **encoder**, and a **demultiplexer**, implement *Q* using only ONE of these devices, without any additional logic gates. Your device should be the smallest possible (for example, if an 8-bit parallel adder is sufficient, you should not use a 16-bit parallel adder). [4 marks]

5. (continue...)

(d) Implement the following four-variable function $R(A,B,C,D)$ using a single **4:1 multiplexer** without any additional logic gates. [6 marks]

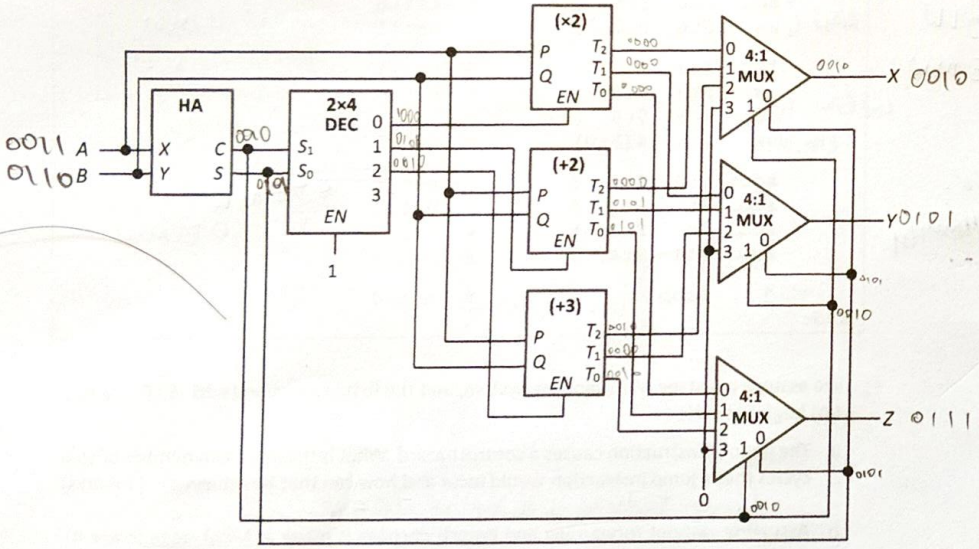
$$R(A,B,C,D) = \Sigma m(0, 2, 3, 4, 6, 7, 12, 14)$$



(e) Study the following circuit which uses a **half adder (HA)**, a **2×4 decoder** with 1-enable and active high outputs, three **4:1 multiplexers** and three devices each with a 1-enable control (*EN*):

- A **(×2)-device**: it takes in two inputs *P* and *Q* and produces 3-bit output with value $(P+Q)\times 2$.
- A **(+2)-device**: it takes in two inputs *P* and *Q* and produces 3-bit output with value $P+Q+2$.
- A **(+3)-device**: it takes in two inputs *P* and *Q* and produces 3-bit output with value $P+Q+3$.

For the three devices above, if a device is not enabled, its outputs are all zeroes.



Redesign the above circuit so that it can be implemented using the fewest logic gates. Write your expressions for *X*, *Y* and *Z*. You do not need to draw your circuit. [6 marks]

A	B	X	Y	Z
0	0	0	0	0
0	1	0	1	1
1	1	1	0	1
1	0	0	1	1

X = A AND B
Y = A XOR B
Z = A OR B.

6. **[18 marks]**

Given three integer arrays *A*, *B*, *C*, where arrays *B* and *C* each contains *n* elements and array *A* contains 2*n* elements, a MIPS code is written to update the elements in *A* with the elements in *B* and *C* as follows:

$A[k] = A[k] + B[k/2]$ if *k* is even
$A[k] = A[k] + C[(k-1)/2]$ if *k* is odd

For example, suppose $A = \{1, 2, 3, 4, ...\}$, $B = \{101, 102, ...\}$ and $C = \{201, 202, ...\}$, then the final values in *A* are {102, 203, 105, 206, ...}.

The MIPS code fragment is shown below.

```
        # $s0 = base address of array A
        # $s1 = base address of array B
        # $s2 = base address of array C
        # $s3 = n, the number of elements in array B
        add  $t0, $s0, $0       # Inst1, Address: 0x00FFFF18
        add  $t1, $s1, $0       # Inst2
        add  $t2, $s2, $0       # Inst3
        add  $t3, $s3, $s3      # Inst4: $t3 = 2n
        add  $t4, $0,  $0       # Inst5: $t4 = k (loop variable)

Loop:   slt  $t5, $t4, $t3      # Inst6: k < 2n?
        beq  $t5, $0,  End      # Inst7

        lw   $t6, 0($t0)        # Inst8
        lw   $t7, 0($t1)        # Inst9
        add  $t6, $t6, $t7      # Inst10
        sw   $t6, 0($t0)        # Inst11

        lw   $t8, 4($t0)        # Inst12
        lw   $t9, 0($t2)        # Inst13
        add  $t8, $t8, $t9      # Inst14
        sw   $t8, 4($t0)        # Inst15

        addi $t0, $t0, 8        # Inst16
        addi $t1, $t1, 4        # Inst17
        addi $t2, $t2, 4        # Inst18
        addi $t4, $t4, 2        # Inst19

        j    Loop               # Inst20
End:
```

For parts (a), (b), (c): Given a **two-way set associative data cache** with 64 words in total, and each block containing 4 words with each word being 4 bytes long. LRU (least recently used) algorithm is used for replacement. Each integer occupies one word.

Assuming that the integer arrays *B* and *C* each contains 2^{10} **elements**. Arrays *A*, *B* and *C* are stored starting at memory addresses 0x**00000080**, 0x**00100000** and 0x**00108040** respectively.

The data cache is involved when memory is accessed (that is, when **lw** and **sw** instructions are executed).

a. How many bits are there in the set index field? In the byte offset field? [2 marks]

b. Which set is *A*[0] mapped to? Which set is *B*[60] mapped to? Which set is *C*[1032] mapped to? You may write your answer in decimal or binary. [3 marks]

c. What is the cache hit rate for array *A*? For array *B*? For array *C*? Write your answer as a fraction. [6 marks]

For parts (e), (f), (g): Given a **direct-mapped instruction cache** with 16 words in total and each block contains 4 instructions (words). The first instruction (**add $t0, $s0, $0**) is at memory address 0x**00FFFF18**. Recall that the integer arrays *B* and *C* each contains 2^{10} **elements**.

d. How many misses are there in the 1st iteration (Inst1 to Inst20 inclusive)? [2 marks]

e. How many misses are there in the 2nd iteration (Inst6 to Inst20 inclusive)? [2 marks]

f. How many misses are there in the execution of the whole code? [3 marks]

7. **[14 marks]**
Refer to the same MIPS code in the previous question:

```
        # $s0 = base address of array A
        # $s1 = base address of array B
        # $s2 = base address of array C
        # $s3 = n, the number of elements in array B
        add  $t0, $s0, $0       # Inst1, Address: 0x00FFFF18
        add  $t1, $s1, $0       # Inst2
        add  $t2, $s2, $0       # Inst3
        add  $t3, $s3, $s3      # Inst4: $t3 = 2n
        add  $t4, $0,  $0       # Inst5: $t4 = k (loop variable)
Loop:   slt  $t5, $t4, $t3      # Inst6: k < 2n?
        beq  $t5, $0,  End      # Inst7
        lw   $t6, 0($t0)        # Inst8
        lw   $t7, 0($t1)        # Inst9
        add  $t6, $t6, $t7      # Inst10
        sw   $t6, 0($t0)        # Inst11

        lw   $t8, 4($t0)        # Inst12
        lw   $t9, 0($t2)        # Inst13
        add  $t8, $t8, $t9      # Inst14
        sw   $t8, 4($t0)        # Inst15

        addi $t0, $t0, 8        # Inst16
        addi $t1, $t1, 4        # Inst17
        addi $t2, $t2, 4        # Inst18
        addi $t4, $t4, 2        # Inst19

        j    Loop               # Inst20
End:
```

We assume a 5-stage MIPS pipeline system, and the first instruction (**add $t0, $s0, $0**) begins at cycle 1.

a. The jump (j) instruction causes a control hazard. What is the minimum number of stall cycles that a jump instruction would incur and how can that be achieved? [2 marks]

b. Assuming without forwarding and branch decision is made at MEM stage (stage 4). No branch prediction is made and no delayed branching is used. How many cycles does the code from instructions 1 through 19 (leaving out the jump instruction) take? You need to count until the last stage of instruction 19. [3 marks]

c. Assuming with forwarding and early branching, that is, the branch decision is made at ID stage (stage 2). No branch prediction is made and no delayed branching is used. How many cycles does the code from instructions 1 through 19 (leaving out the jump instruction) take? You need to count until the last stage of instruction 19. [3 marks]

save 1 cycle.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

d: 5 +1 +1 +1 +1 +1 +1 +1

d. Assuming with forwarding and early branching, that is, the branch decision is made at ID stage (stage 2). Branch prediction is used, where the branch is predicted not taken. How many cycles does the code from instructions 1 through 19 (leaving out the jump instruction) take? You need to count until the last stage of instruction 19. [3 marks] 26. 25.

e. Assuming with forwarding, how would you rearrange the instructions to reduce the number of stall cycles, and how many stall cycles is reduced as a result of this? You do not need to rewrite the full code. Just describe the changes or show the portion that is changed. Your changes should be as minimal as possible. [3 marks]

Swap I0 and I2, I1 and I3.
Reduce 2 cycles ✓

~~~ END OF PAPER ~~~

(The next few pages contain the MIPS Reference Data sheet, blank truth tables, K-maps and pipeline charts.)

(This page is for your rough work.)

Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
I1 add	IF	ID	EX	MEM	WB																									
I2 add		IF	ID	EX	MEM	WB																								
I3 add			1	2	3	4	5																							
I4 add				1	2	3	4	5																						
I5 add					1	2	3	4	5																					
I6 slt						1	2	3	4	5																				
I7 beq							1	2	3	4	5			−1																
I8 lw									1	2	3	4	5																	
I9 lw										1	2	3	4	5																
I10 add											1	2		3	4	5														
I11 sw												1		2	3	4	5													

Cycle																														
I12 lw													1	2	3	4	5													
I13 lw														1	2	3	4	5												
I14 add															1	2		3	4	5										
I15 sw																1		2	3	4	5									
I16 addi																		1	2	3	4	5								
I17 addi																			1	2	3	4	5							
I18 addi																				1	2	3	4	5						
I19 addi																					1	2	3	4	5					