Spring 2020 Semester Final Answer Sheet

Name: Syeda Nowshin Ibnat
ID: 17183103020
Intake: 39 (1)

Course Title: Computer Architecture

Course code: CSE 215

Answer to the question: 1(a)

Soln:

Given, $(-0.2525_{ten}) + (-0.455_{ten})$

converting this numbers to binary:

$-0.2525 = -0.0100 = -0.1000 \times 2^{-1} = -1.0000 \times 2^{-2}$

$-0.455 = -0.0111 = -0.1110 \times 2^{-1} = -1.1100 \times 2^{-2}$

Step 1: (change the smaller number)

We don't need to do this step. Because it's already done.

$$\left.\begin{array}{l} -1.0000 \times 2^{-2} \\ -1.1100 \times 2^{-2} \end{array}\right\}$$

Name: Syeda Nowshin Ibnat
ID: 17183103020

Step 2: (Addition of the Significants)

$$\begin{array}{r} -1.0000 \\ (+)\ -1.1100 \\ \hline -10.1100 \end{array} \longrightarrow \therefore -10.1100 \times 2^{-2}$$ in the ~~add~~ result of ~~the~~ addition.

Step 3: (Find the nonmalized form & check overflow ~~and~~ or underflow)

The number in,

$$-10.110 \times 2^{-2}$$

$$= -0.1011 \times 2^{0}$$

= nonmalized

There is no overflow or underflow.

Step 4: (Round ~~of~~ the sum)

∴ No need to round the number.

The sum is then,

$$= -0.1011 \times 2^{0}$$

$$= -0.1011_{two}$$

$$= -0.7075_{ten}$$

This sum is what we would expect from adding $-(0.2525) + (-0.455)$

Ans:

Name: Syeda Nowshin Ibnat
ID: 17183103020

Answer to the question: 1(b)

Soln: Given,

Scenario:

add $t1, $t2, $t3

beq $t1, $t4, 40

lw $t2, 100($t1)

If we want to implement the pipelining method on the given instructions then ~~control~~ a hazard will be occure. Here the control hazard will be occure. control hazard in an occurrence in which the proper instruction cannot execute in the proper clock cycle because the instruction that was fetched is not the one that is needed; that is, the flow of instruction ~~addressis not~~ addresses is not what the pipeline expected. control hazard occurs because it attempts to make decision before condition is evaluated. When we need to find the destination of a branch, and can't fetch any new instructions untill we know the destination then it is called a control hazard.

Name: Syeda Nowshin Ibnat
ID: 17183103020

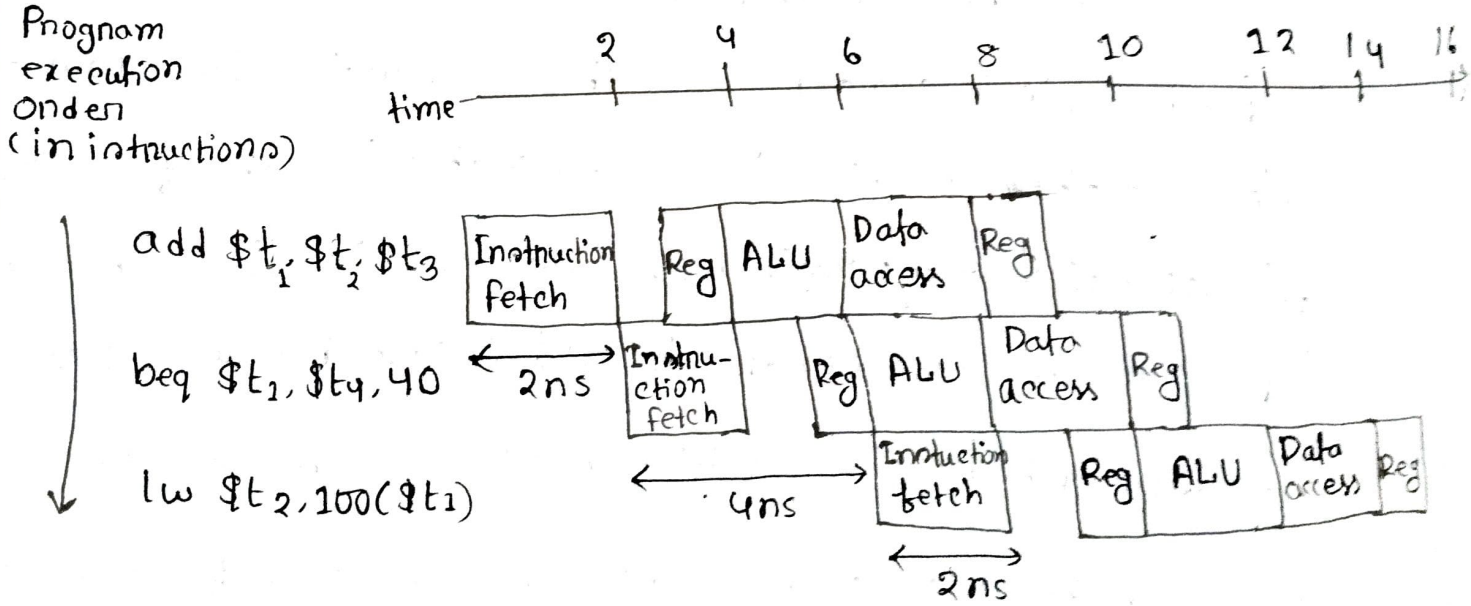
A branch is either:

- Taken : PC <= PC + 4 + immediate
- Not taken : PC <= PC + 4

Solutions to solve this problem:

Solution 1: Stall

- The lw instruction executed if the branch fails, is stalled one extra 2-ns clock cycle before starting.
- The cost of this option is high for most.



Name: Syeda Nowshin Ibnat
ID: 17183103020

Solution 2: Predict the Branch Target

One simple approach is to always predict that branch will fail. When the prediction is right, the pipeline proceeds at full speed.

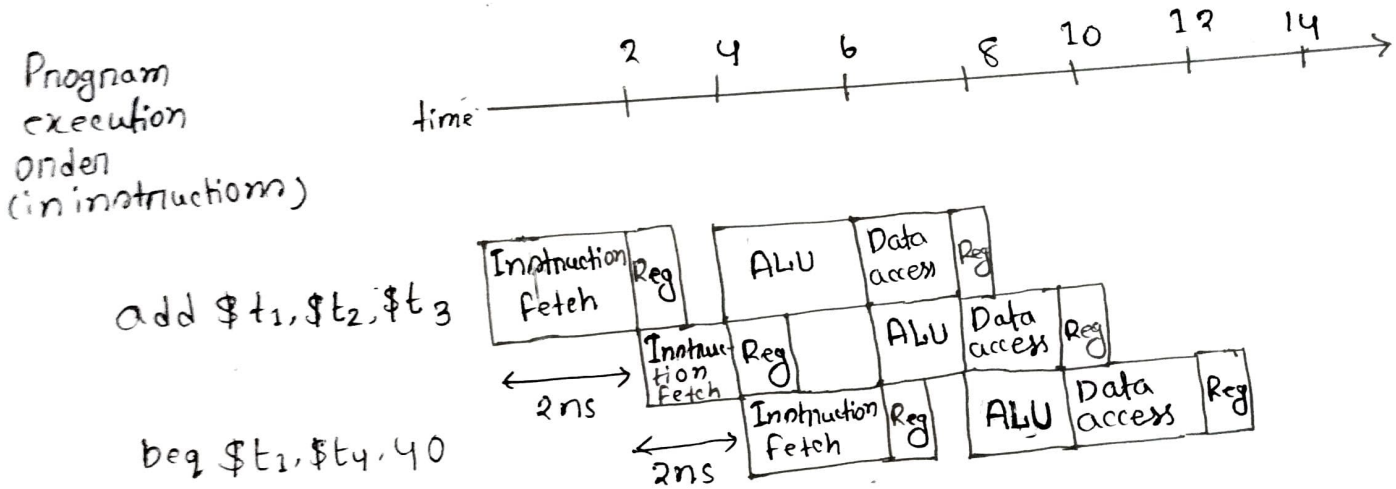
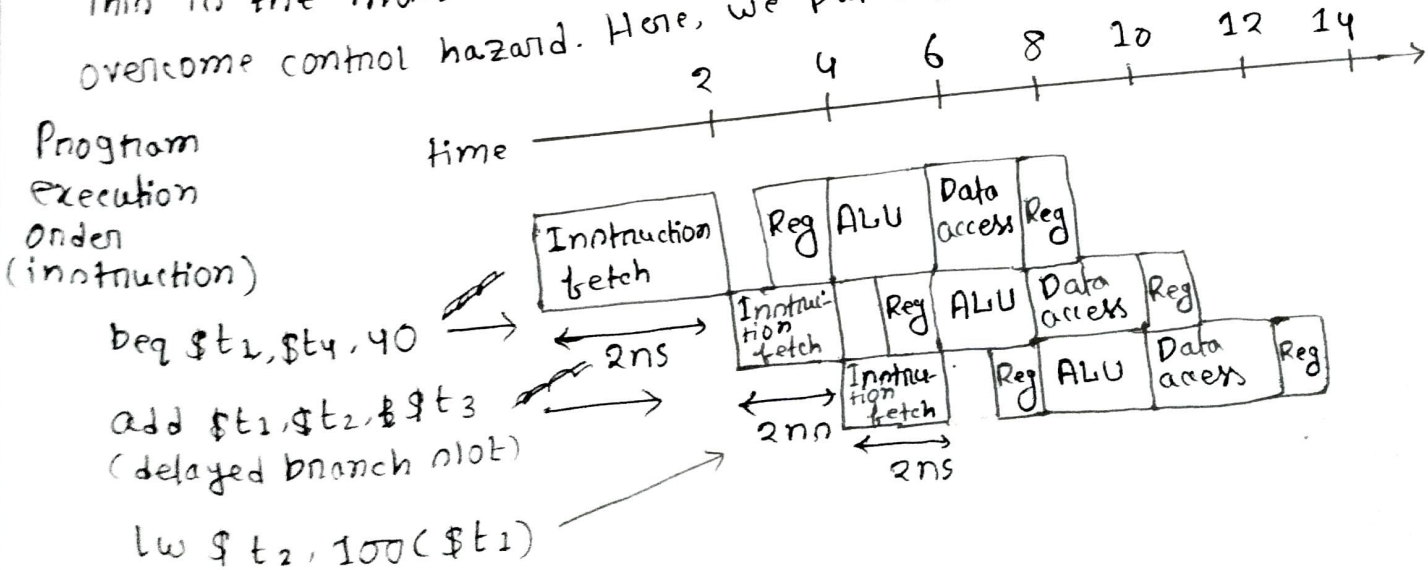


Figure: When the branch is not taken

Solution 3: Delayed Decision (Used in MIPS)

This is the most effective solution. So, we can use this to overcome control hazard. Here, we put beq instruction before add & lw.



Name: Syeda Nowshin Ibnal

ID: 17183103020

Answer to the question: 2(a)

Soln: Muhammad is following the Final version of Division Algorithm.

Now, using this algorithm computing Muhammad's division result in below:

Given, (0001 1111) ÷ (0011)

Initial	Step	Division	Remainder
0	Initial values	0011	0001 1111
	Shift Rem left 1	0011	0011 1110
1	2: Rem = Rem - Div	0011	0000 1110
	3a: Rem ≥ 0 ⇒ SllR, $R_0 = 1$	0011	0001 1101
2	2: Rem = Rem - Div	0011	1110 1101
	3b: Rem < 0 ⇒ +Div, SllR, $R_0 = 0$	0011	0011 1010
3	2: Rem = Rem - Div	0011	0000 1010
	3a: Rem ≥ 0 ⇒ SllR, $R_0 = 1$	0011	0001 0101
4	2: Rem = Rem - Div	0011	1110 0101
	3b: Rem < 0 ⇒ +Div, SllR, $R_0 = 0$	0011	0010 1010
	Shift left half of Rem right 1	0011	0001 1010

Ans:

Name: Syeda Nowshin Ibnat
ID: 17183103020

Answer to the question: 2(b)

Soln: Given,

word addresses = 20, 28, 16, 20, 29, 3, 16, 14

Decimal address of reference	Binary address of reference	Hit or miss in cache	Assigned cache block (where found or place)
20	10100_{two}	miss a	$(10100_{two} \bmod 8) = 100_{two}$
28	11100_{two}	miss b	$(11100_{two} \bmod 8) = 100_{two}$
16	10000_{two}	miss c	$(10000_{two} \bmod 8) = 000_{two}$
20	10100_{two}	hit	$(10100_{two} \bmod 8) = 100_{two}$
29	11101_{two}	miss d	$(11101_{two} \bmod 8) = 101_{two}$
3	00011_{two}	miss e	$(00011_{two} \bmod 8) = 011_{two}$
16	10000_{two}	hit	$(10000_{two} \bmod 8) = 000_{two}$
14	01110_{two}	miss f	$(01110_{two} \bmod 8) = 110_{two}$

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	N		

a. The initial state of the cache after power on.

Index	V	Tag	Data
000	N		
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	memory $(10100)_{two}$

b. After handling a miss off address $(10100)_{two}$

Name: Syeda Nowshin Ibnat

ID: 17183103020

Index	V	Tag	Data
000	Y	11	memory $(11100)_{two}$
001	N		
010	N		
011	N		
100	N		
101	N		
110	Y	10	memory (10100)

c. After handling miss of address $(11100)_{two}$

Index	V	Tag	Data
000	Y	11	memory $(11100)_{two}$
001	N		
010	Y	10	memory $(10000)_{two}$
011	N		
100	N		
101	N		
110	Y	10	memory $(10100)_{two}$

d. After handling miss of address $(10000)_{two}$

Index	V	Tag	Data
000	Y	11	memory $(11100)_{two}$
001	Y	11	memory (11101)
010	Y	10	memory $(10000)_{two}$
011	N		
100	N		
101	N		
110	Y	10	memory (10100)

e. After handling miss of address $(11101)_{two}$

Index	V	Tag	Data
000	Y	11	memory $(11100)_{two}$
001	Y	11	memory $(11101)_{two}$
010	Y	10	memory $(10000)_{two}$
011	Y	00	memory $(00011)_{two}$
100	N		
101	N		
110	Y	10	memory $(10100)_{two}$

f. After handling miss of address $(00011)_{two}$

Name: Syeda Nowshin Ibnat

ID: 17183103020

Index	V	Tag	Data
000	Y	11	memory (11100)two
001	Y	11	memory (11101)two
010	Y	10	memory (10010)two
011	Y	00	memory (00011)two
100	N		
101	Y	10	memory (10100)two
110	Y	01	memory (01110)two

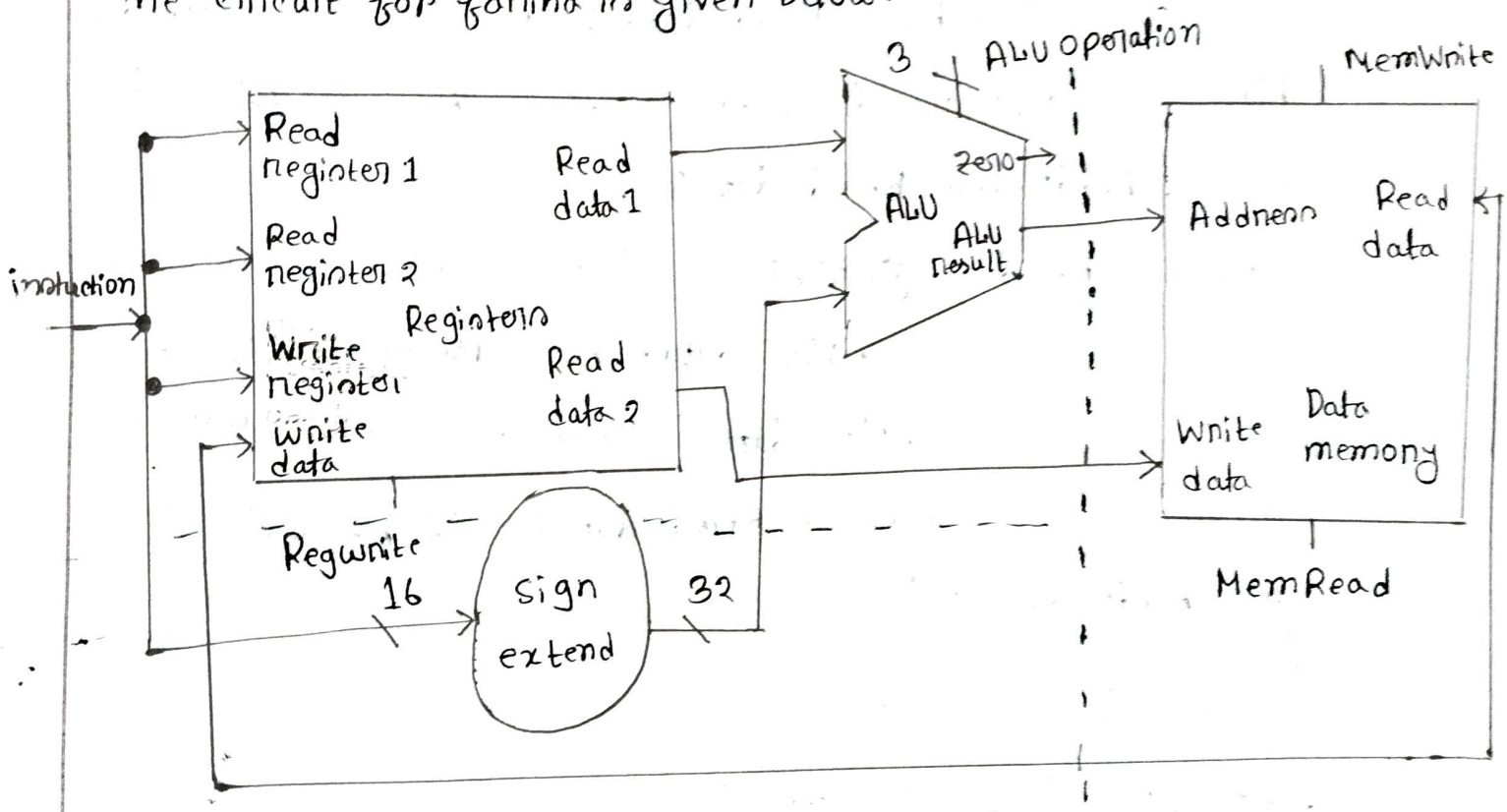
g. After handling miss of ~~adding~~ address (01110)two

Ans:

Name: Syeda Nowshin Ibnat
ID: 17183103020

Answer to the question: 3(a)

Soln: A simple circuit can be design for Fariha. Which can be able to work the only store and add instruction. The circuit for fariha is given below:



Examples from each store and add instructions:

☒ ~~ad~~ sw $S1, 100($S2)

data processing for this:

For this instruction at first ~~$s~~ $S2 will go to Read register 1. Here we have to calculate memory address that's why we will read $S2. Then data of $S1 will go

Name: Syeda Nowshin Ibnat
ID: 17183103020

to Read register 2. We don't need to use Write register. So, it will be empty. Through the instructions ~~4x100~~ 4x100=400 will go to ALU after converting from 16 bit to 32 bit. Here, in Read data 1 we have the value of $S2. And Read data 1 is directed with $S2. ALU will calculate the address; then ALU result will go to the Address. Read data 2 is consist with the data of Read register 2 ($S1 is here). After reading the data it will go to Write data. Here MemWrite will be enable here as write data is using to write data in memory. The value of Read data 2 will be write on the Address at last.

add $t1, $t2, $t3

data processing for this:-

For this instruction at first Read register 1 will read data of $t2 register. So, Read data 1 will received the data of $t2 from Read register 1. Then Read register 2 will read data of $t3 register. Read data 2 will be consist with the value of $t3 register. Here, we have to read

Name: Syeda Nowshin Ibnat
ID: 17183103020

two data. So, we have two Read register $t2 & $t3 to use. After reading $t2 and $t3 register, we will write it on $t1 register. So, we will write it on $t1 register. So, we will need one write register for ~~this~~. write data. We have to write the addition (result) of $t2 and $t3 register on memory. Write data will work for this. It will write the data on memory. Now, two inputes from Read data 1 ($t2) and Read data 2 ($t3) will go to ALU. Here, ALU will add the two registers $t2, $t3 using 3bit. ALU will do the add operation, because it received "0010" instruction. At last the result of ALU will go to write data.

For "add" we don't need to use Address. As well as sign extend ~~a~~ ~~will~~ ~~not~~ don't have to use.

Ans:

Name: Syeda Nowshin Ibnat
ID: 17183103020

Answer to the question: 3(b)

Soln: The final version of the division algorithm is faster than the first version of the division ~~algorithm~~ algorithm. It is true and the explanation is given below:

Final version of Division Algorithm

An algorithm which is given two integers a and b ~~(b≠0)~~ computes their quotient and/or remainder. ~~the result of quotient division~~ Observation of first version division algorithm: i) 1/2 bits in divisor always 0. ii) 1/2 of 64 bit adder is wasted. iii) First step cannot produce a 1 in quotient bit.

Diagram

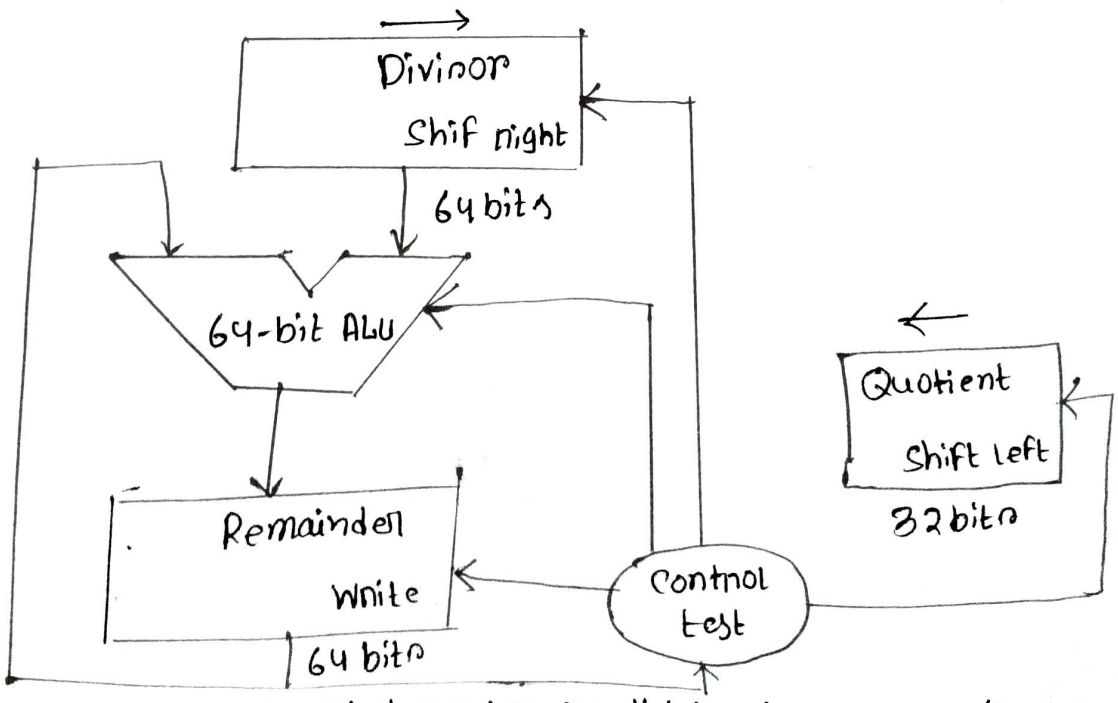


Fig: First version of division hardware diagram

Name: Syeda Nowshin Ibnat
ID: 17183103020

Final Version of Division Algorithm

Observation for this:

1. It just need ALU to add or subtract and 63 bit register to shift left or right. Low and high register in MIPS combine to act as 64 bit register for multiply and divide. possible for quotient to be too large but if divide 64 bit integer by 1, quotient is 64 bits.

Diagram

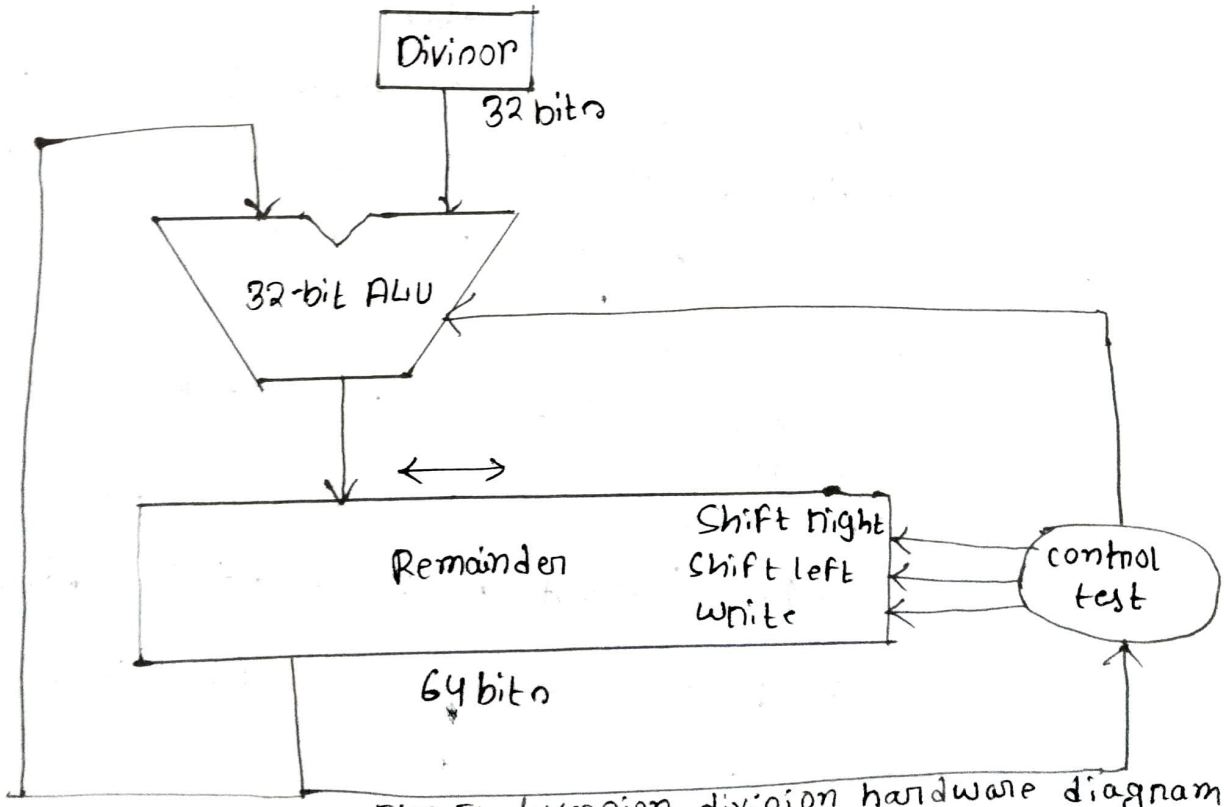


Fig: Final version division hardware diagram

From the above observation we can say that Final Version of division algorithm is faster then first version of division algorithm.

Ans:

Answer to the question: 4(a)

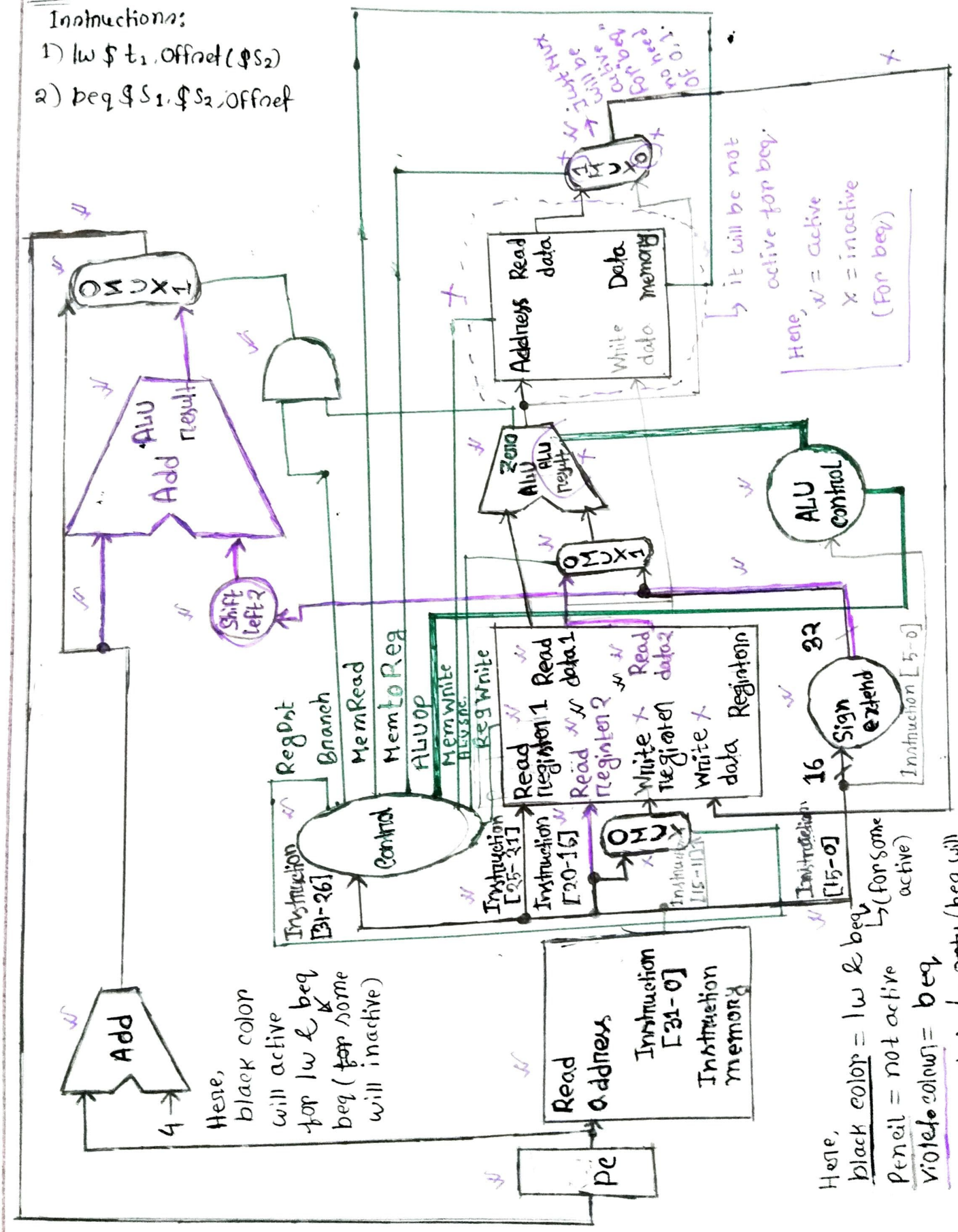
Name: Syeda Nawshin Ibnat
ID: 17183103020

Fig: Only one datapath with control line

Solⁿ: Given,

Instructions:

1) lw $t1, Offset($s2)
2) beq $s1, $s2, Offset



Name: Syeda Nowshin Ibnat
ID: 17183103020

Answer to the question : 4(b)

Soln: There are two instructions given in Q4(a).

$$\left\{\begin{array}{l} lw\ \$t_1,\ Offset(\$S_2) \\ beq\ \$S_1,\ \$S_2,\ Offset \end{array}\right\}$$

And this two instructions will work in only on datapath with control unit (we can see from the Ans of 4(a)).

For load (lw)

→ lw $t1, offset ($S2)

For this instruction:

1. First instruction is fetched.
2. $S2 value is read then.
3. Instruction[25-21], Instruction[25-21] will be active. Instruction[20-16] will go to MUX. So, it will be also active for it. Instruction[15-0] will be active. We have use sing extend (16 bit will be ~~conten~~ convert into 32 bit). (4X offset)
4. Then ALU operation will perform. Here, address will be calculated, (4×offset + $S2). i.e. sum of the ALU is used as the address for the data memory.
5. At last data from the memory ~~to~~ will write into the register ~~to~~ $t1. So, "Write data" will be active.

Name: Syeda Nowshin Ibnat

ID: 17183103020

For beq

→ beq $S1, $S2, offset.

1. At first instruction will be fetched from the instruction memory.
2. Register $S1 and $S2, are read from the register file.
3. ~~Instruction~~ "Read data 2" will be active here as well as "Read data 1" ~~one~~.
4. The ALU performs a subtract on the data values read from the register file. The value of PC+4 is added to the sign-extended, lower 16 bits of the instruction (offset) shifted left by two; The result is the branch target address.
5. The zero result from the ALU is used to decide which adder result to store into the PC.
6. "Address" will be inactive here.

Ans: