a) Compare between pipeline machine and non-pipeline machine. suppose there are (X+7) instructions in a program. Draw the page table and compute the following: (where X is the last digit of you ID number) * consider there are 5 stages and each takes one clock cycle. **10**

- Total time for pipeline and non-pipeline.
- CPI
- Speedup
- Efficiency or utilization

b) This question considers the basic MIPS, 5-stage pipeline (IF, ID, EXE, MEM, WB). **10**

Assume that you have the following sequence of instructions:

lw $s2, 0($s1) (instr1)
add $s3, $s4, $s2 (instr2)
Sub $s6, $s2, $s3. (instr3)

Show the implementation through 5 stages and explain the implementation for both pipelined and non-pipelined design. (explain if there is any pipeline hazards)

a) Given Instructions No: X+7
= 3+7
= 10

Let assume clock cycle time = 1 sec

No of stages = 5

Each stage takes 1 clock cycle

Now the page table for this is given below:

No of stage															
IF	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}					
ID		I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}				
EXE			I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}			
MEM				I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}		
WB					I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	
Clock cycle time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	

For pipeline

Let k = number of stage

n = no of instructions

pipeline execution time = $(k \times 1) + (n-1) \times$ clock cycle time

$= (5 \times 1) + (10-1) \times 1$

$= (5 + 9) \times 1$

$= 14$ sec

For Non pipeline

Execution time = (No of stage $\times$ clock cycle time) $\times$ No of Instructions

$= (5 \times 1) \times 10$

$= 50$ sec

- CPI = $\frac{\text{Total clock cycle}}{\text{total instruction}}$

$= \frac{14}{10}$

$= 1.4$

$\cong 1$

- Speedup = $\frac{\text{Total time in Non pipeline}}{\text{Total time in pipeline}}$

$= \frac{50 \text{ sec}}{14 \text{ sec}}$

$= 3.57$

- Efficiency = $\frac{\text{Total Block in Page table}}{\text{Total used block}}$

$= \frac{70}{50}$

$= 1.4$

Comparision between Pipeline machine and Non pipeline machines

Pipelining represent the scenario in such a way where we'll let instructions execution in terms of a parallel way - like when the first instruction's decode happens, we'll find 2nd instruction's fetch is done whereas a non-pipelined machine executes only a single instruction at a time.

From the above calculation here we can analyse that the execution time for non-pipelined machine is greater than pipe-lined machine. Therefore, pipelining provides more effecieny and less execution time than non-pipelined architecture.

b) Given

(a) No of instructions = 3

Let assume

instr1 as I_1

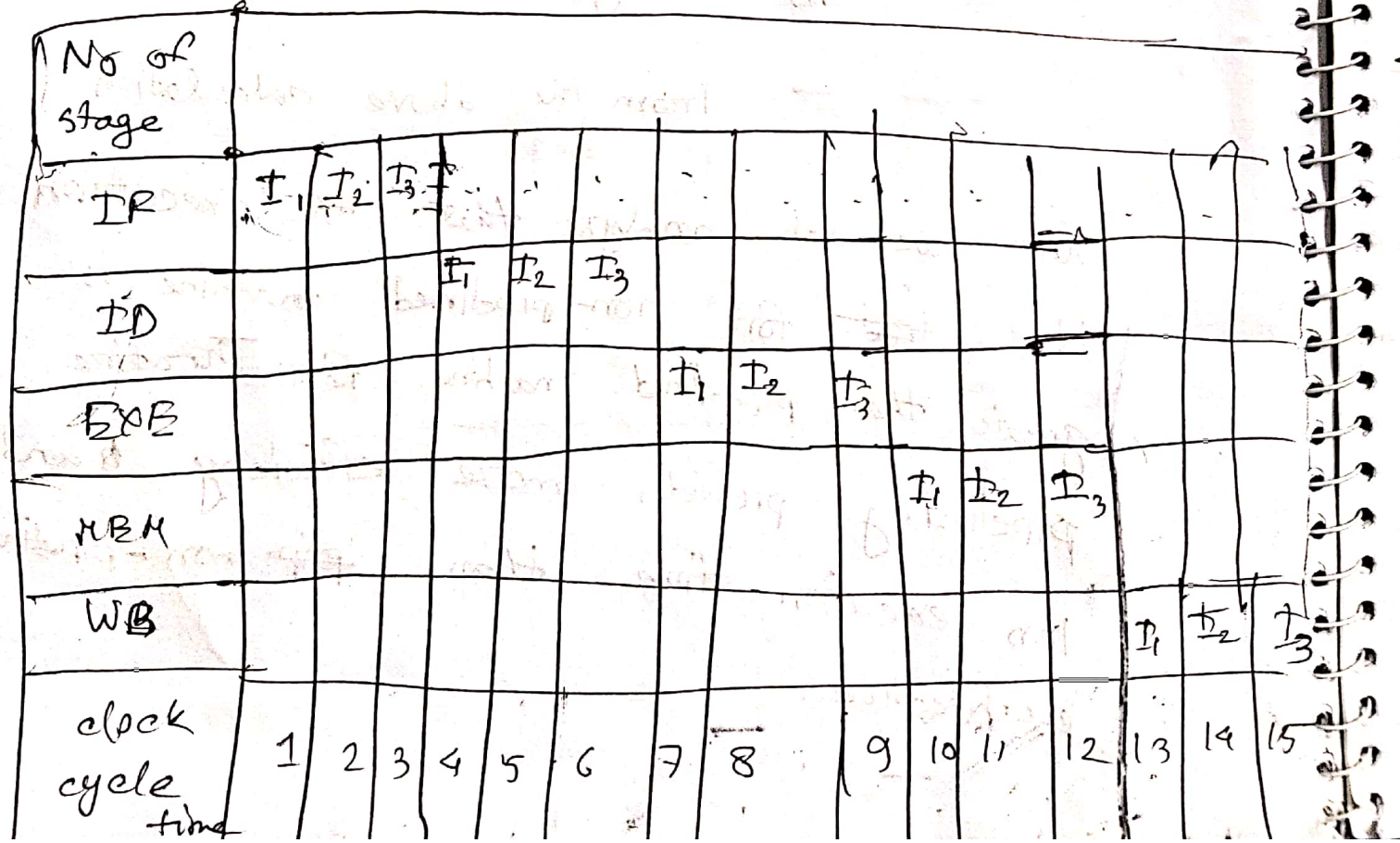
instr2 as I_2

instr3 as I_3

No of stages = 5

let assume clock cycle time = 1 sec

For Non pipeline:



Non pipeline execution time = (No of stage × clock cycle time) × No of instruction

= (5 × 1) × 3

= 5 × 3

= 15 sec

~~Efficiency = Total block in Page table / Total used block~~

~~= 5 / 15 = 1/3~~

For pipeline

No of stages							
IF	I_1	I_2	I_3				
ID		I_1	I_2	I_3			
EXE			I_1	I_2	I_3		
MEM				I_1	I_2	I_3	
WB					I_1	I_2	I_3
clock cycle time	1	2	3	4	5	6	7

pipeline execution time = $(k \times 1)(n-1) \times 1$

$= (5 \times 1)(3-1) \times 1$

$= (5 \times 2) \times 1$

$= 10$ sec

Now according to the previous ~~Efficiency~~ calculation we can get:

$$\text{speedup} = \frac{15 \text{ sec}}{10 \text{ sec}} = 1.5$$

(15 sec → Non-pipelined machine execution time; 10 sec → pipelined execution time)

So here pipelined machine will provide better performance than non-pipelined. as here requires less execution time. Moreover from the page table, we can also find that for non-pipelined machine, it has

required 14 cycles whereas for pipelined it has required only 7 cycles which is half of the non-pipelined one for executing the given 3 instructions completely.

Although we can find better performance during pipelining here for the above MIPS instructions, we may face some hazards.

As we know three types of hazards can be happened in pipeline →

- Structural hazards
- control "
- data "

Since here the load and add instructions are executed parallely so, here data hazards can

occur.

In pipeline, an instruction requires data to be computed by a previous instruction still in the pipeline. So here if there two instructions (instr1 and instr2) which are load → Add execute parallelly, then we can find during load instruction the value of register is stored in a temporary register, and after that that the value will be loaded to memory or the value may be stored in that register. So, as during pipelining the stages will be overlapped so, if that value is in both places at the same time then here during addition when we will write it back then we can find the updated value in the register but if before

loading it to memory the addition instruction execute, then we may not get the updated value that register had been stored in register '$2'. So, here this type of hazard may occur during pipelining.

1

a. Compare between pipeline machine and non-pipeline machine. Suppose there are I instructions in a program. Draw the page table and compute the following:

- **Total time for pipeline and non-pipeline.**
- **Speedup**
- **Efficiency or utilization.**

  Where I+5= last digit of your ID number.(if last digit is in between 7to 9, subtract -3from your total count.)

b. Consider a non-pipelined machine with 6 execution stages of lengths 20 ns, 20 ns, 30 ns, 25 ns, 20 ns, and 20 ns.

- Find the instruction latency on this machine.
- How much time does it take to execute 80 instructions?

Suppose we introduce pipelining on this machine. Assume that when introducing pipelining.

- What is the instruction latency on the pipelined machine?
- How much time does it take to execute 80 instructions?

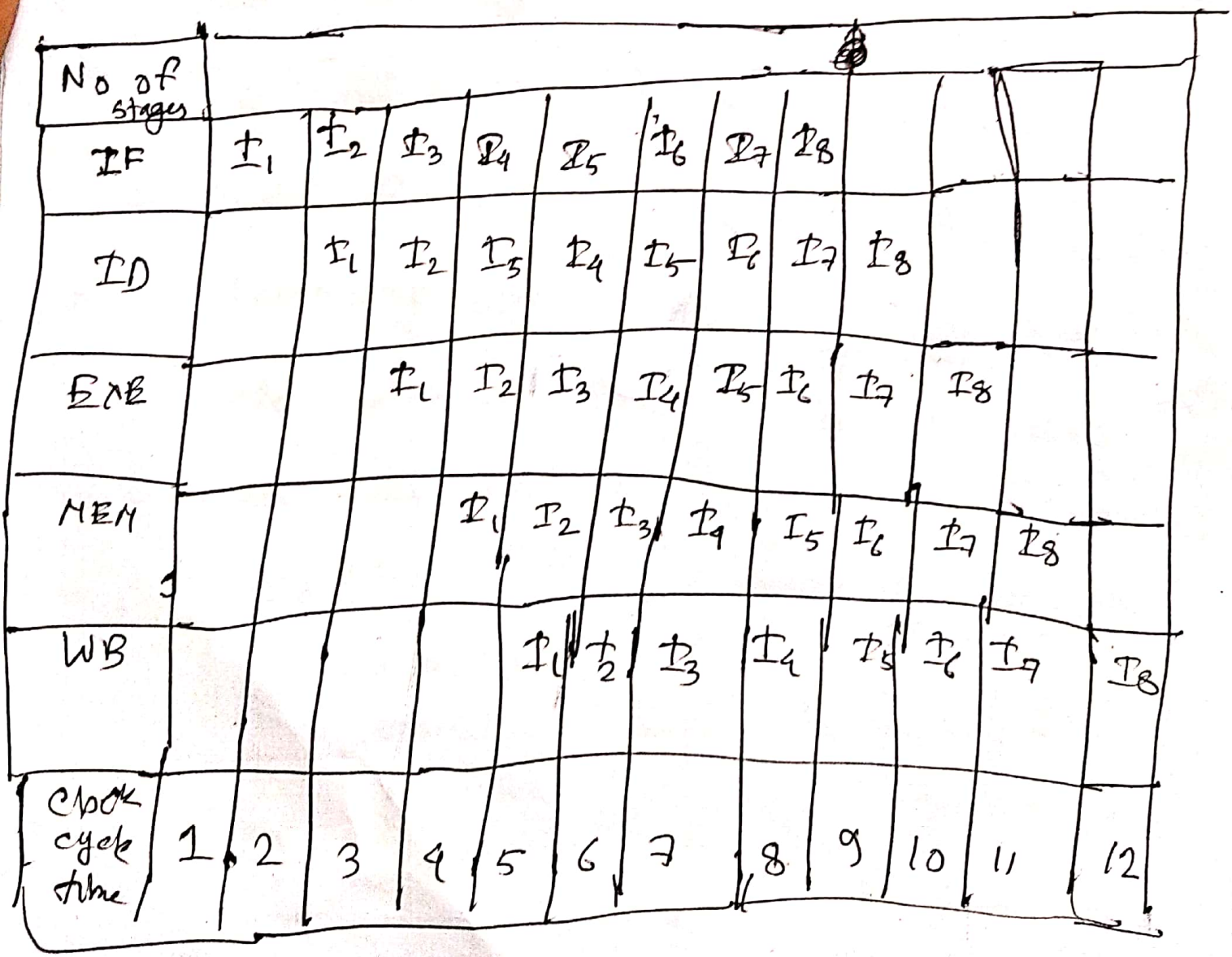
Also calculate the speedup.

a) Here

No of instructions I = 3+5
= 8

No of stages = 5

Let assume clock cycle time = 1 sec



- Total time for pipeline = $(k*1)+(n-1) \times$ clock cycle time

$= (5 \times 1) + (8-1) \times 1$

$= (5+7) \times 1$

$= 12$ sec

Total time for Non-pipeline $= (5 \times 1) \times 8$

$= 40$ sec

- Speedup $= \frac{40 \text{ sec}}{12 \text{ sec}}$

$= 3.33$

- Efficiency $= \frac{60}{40}$

$= 1.5$

b) Given

No of execution stages = 6

the clock cycle time for 6 instructions are : 20 ns, 20 ns, 30 ns, 25 ns, 20 ns and 20 ns

For non pipeline

Instruction latency = (20 + 20 + 30 + 25 + 20 + 20) ns

= 135 ns

Total execution time for 80 instructions will be : (135 × 80) ns

= 10800 ns

For pipeline

Instruction latency = (30 × 6) ns → max stage value from the given values

= 180 ns

Total execution time for 80 instructions will be : (5 × 1) + (80 − 1) × 30

$= 5 + (5 \times 30)$

$= 155$ ns

$$\text{Speedup} = \frac{10800 \text{ ns}}{155 \text{ ns}}$$

$= 69.67$

a. Suppose there are I instructions in a program. Draw the page table and compute the following:

- **Total time for pipeline and non-pipeline.**
- **CPI**
- **Speedup**
- **Efficiency or utilization.**

Where I= 6+last digit of your id.

a) No of instructions : 6+3

= 9

No of stages = 5

Let assume clock cycle time 1 sec

No of stages													
IF	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9				
ID		I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9			
EXE			I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9		
MEM				I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	
WB					I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9
clock cycle time	1	2	3	4	5	6	7	8	9	10	11	12	13

Total time for pipeline = $(k \times 1) + (n-1) \times$ clock cycle time

$= (5 \times 1) + (9-1) \times 1$

$= 5 + (8 \times 1)$

$= 13$ sec

Total time for non pipeline = $(5 \times 1) \times 9$

$= 45$ sec

CPI = $\frac{\text{Total clock cycle}}{\text{total instruction}}$

$= \frac{13}{9}$

$= 1.4$

$\cong 1$

Speed up = $\frac{45 \text{ sec}}{13 \text{ sec}}$

$= 3.4615$

Efficiency = $\frac{65}{45}$

$= 1.4$

3. **a)** Compare between pipeline machine and non-pipeline machine. suppose there are (x+7) instructions in a program. Draw the page table and compute the following: (where X is the last digit of you ID number) * consider there are 5 stages and each takes one clock cycle. **10**

- Total time for pipeline and non-pipeline.
- Speedup
- Efficiency or utilization

b) Consider a non-pipelined machine with 5 execution stages of lengths 12 ns, 15 ns, 8 ns, 18 ns, and 20 ns. **10**

- Find the instruction latency on this machine.
- How much time does it take to execute (**i**+1000) instructions?

Suppose we introduce pipelining on this machine. Assume that when introducing pipelining.

- What is the instruction latency on the pipelined machine?
- How much time does it take to execute (**i**+1000) instructions?

Also calculate the speedup.

(where **i** is the last two digits of you ID number)

c) This question considers the basic MIPS, 5-stage pipeline (IF, ID, EXE, MEM, WB). **10**

Assume that you have the following sequence of instructions:

```
lw $s2, 0($s1) (instr1)
add $s3, $s4, $s2 (instr2)
Sub $s6, $s2, $s3. (instr3)
```

Show the implementation through 5 stages and explain the implementation for both pipelined and non-pipelined design. (explain if there is any pipeline hazards)

a) Given Instructions No: X+7

= 3+7

= 10

Let assume clock cycle time = 1 sec

No of stages = 5

Each stage takes 1 clock cycle

Now the page table for this is given below:

No of stage														
IF	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}				
ID		I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}			
EXE			I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}		
MEM				I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	
WB					I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}
Clock cycle time	1	2	3	4	5	6	7	8	9	10	11	12	13	14

For pipeline

Let k = number of stage

n = no of instructions

pipeline execution time = $(k \times 1) + (n-1) \times$ clock cycle time

$= (5 \times 1) + (10-1) \times 1$

$= (5 + 9) \times 1$

$= 14$ sec

For Non pipeline

Execution time = (No of stage $\times$ clock cycle time) $\times$ No of Instructions

$= (5 \times 1) \times 10$

$= 50$ sec

CPI = Total clock cycle / total instruction

= 14 / 10

= 1.4

≅ 1

Speedup = Total time in Non pipeline / Total time in pipeline

= 50 sec / 14 sec

= 3.57

Efficiency = Total Block in Page table / Total used block

= 70 / 50

= 1.4

Comparision between Pipeline machine and Non pipeline machine:

Pipelining represent the scenario in such a way where we'll let instructions execution in terms of a parallel way. like when the first instruction's decode happens, we'll find 2nd instruction's fetch is done whereas a non-pipelined machine executes only a single instruction at a time.

From the above calculation here we can analyse that the execution time for non-pipelined machine is greater than pipe-lined machine. Therefore, pipelining provides more effeciency and less execution time than non-pipelined architecture.

Given

b) No of execution stages = 5

For non pipeline:

Instruction latency = (12+15+8+18+20) ns

= 73 ns

Total execution time for 1003 instructions:

(73 × 1003) ns

= 73219 ns

For pipeline:

Instruction Latency = (20 × 5) ns

= 100 ns

Total execution time for (1000 to 3) 1003 instructions: (5 × 1) + (5-1) × 20

= 5 + (4 × 20)

= 5 + 80 = 85 ns

$$\text{Speedup} = \frac{7321.9\ ns}{85\ ns}$$

$$= 86.14$$

b) Given

No of instructions = 3

Let assume

instr1 as I_1

instr2 as I_2

instr3 as I_3

No of stages = 5

let assume clock cycle time = 1 sec

For Non pipeline:

Non pipeline execution time = (No of stage × clock cycle time) × No of instruction

= (5 × 1) × 3

= 5 × 3

= 15 sec

~~Efficiency = Total block in Page table / Total used block~~

~~= 5 / 15 = 1/3~~

For pipeline

No of stages							
IF	I_1	I_2	I_3				
DD		I_1	I_2	I_3			
EXE			I_1	I_2	I_3		
MEM				I_1	I_2	I_3	
WB					I_1	I_2	I_3
clock cycle time	1	2	3	4	5	6	7

pipeline execution time = $(k \times 1)(n-1) \times 1$

$= (5 \times 1)(3-1) \times 1$

$= (5 \times 2) \times 1$

$= 10$ sec

Now according to the previous calculation we can get:

$$\text{speedup} = \frac{15 \text{ sec}}{10 \text{ sec}} = 1.5$$

(15 sec → Non-pipelined machine execution time; 10 sec → pipelined execution time)

So here piplined machine will provide better performance than non-pipelined. as here requires less execution time. Moreover from the page table, we can also find that for non-pipelined machine, it has

required 14 cycles whereas for pipelined it has required only 7 cycles which is half of the non-pipelined one for executing the given 3 instructions completely.

Although we can find better performance during pipelining here for the above MIPS instructions, we may face some hazards.

As we know three types of hazards can be happened in pipeline →

- Structural hazards
- control "
- data "

Since here the load and add instructions are executed parallely so, here data hazards can

occur.

In pipeline, an instruction requires data to be computed by a previous instruction still in the pipeline. So here if these two instructions (instr1 and instr2) which are load → Add execute parallelly, then we can find during load instruction the value of register is stored in a temporary register, and after that that the value will be loaded to memory or the value may be stored in that register. So, as during pipelining the stages will be overlapped so, if that value is in both places at the same time then here during addition when we will write it back then. We can find the updated value in the register but if before

loading it to memory the addition instruction execute, then we may not get the updated value that register had been stored in register '$2'. So, here this type of hazard may occur during pipelining.

b) Consider a non-pipelined machine with 5 execution stages of lengths 35 ns,40 ns, 45ns, 30 ns, and 20 ns. **10**

- Find the instruction latency on this machine.
- How much time does it take to execute (**i**+1010) instructions?

Suppose we introduce pipelining on this machine. Assume that when introducing pipelining.

- What is the instruction latency on the pipelined machine?
- How much time does it take to execute (**i**+1010) instructions?

Also calculate the speedup.

(where **i** is the last two digits of you ID number)

c) This question considers the basic MIPS, 5-stage pipeline (IF, ID, EXE, MEM, WB). **10**

Assume that you have the following sequence of instructions:

lw $s2, 0($s1) (instr1)
add $s3, $s4, $s2 (instr2)
Sub $s6, $s2, $s3. (instr3)

Show the implementation through 5 stages and explain the implementation for both pipelined and non-pipelined design. (explain if there is any pipeline hazards)

6) Given No of stages = 5

For non pipeline

Instruction latency = (35+10+45+30+20) ns

= 170 ns

∴ Total execution time for (1010+03) = 1013 instructions: (170 X 1013) ns

= 172210 ns

For pipeline:

Instruction latency = (45 X 5) ns

= 225 ns (P.T.O)

Total execution time 1013 instructions:

$$(5 \times 1) + (5-1) \times 45$$

$$= 5 + (4 \times 45)$$

$$= 5 + 180$$

$$= 185 \text{ ns}$$

$$\text{Speedup} = \frac{172210 \text{ ns}}{185 \text{ ns}}$$

$$\Rightarrow 930.8649$$

3. a) Compare between pipeline machine and non-pipeline machine. suppose there are (X+7) instructions in a program. Draw the page table and compute the following: (where X is the last digit of you ID number) * consider there are 5 stages and each takes one clock cycle. **10**

- Total time for pipeline and non-pipeline.
- CPI
- Speedup
- Efficiency or utilization

b) Consider a non-pipelined machine with 5 execution stages of lengths 30 ns,25 ns, 24 ns, 12 ns, and 20 ns. **10**

- Find the instruction latency on this machine.
- How much time does it take to execute (i+1000) instructions?

Suppose we introduce pipelining on this machine. Assume that when introducing pipelining

- What is the instruction latency on the pipelined machine?
- How much time does it take to execute (i+1000) instructions?

Also calculate the speedup.

(where i is the last two digits of you ID number)

c) This question considers the basic MIPS, 5-stage pipeline (IF, ID, EXE, MEM, WB). **10**

Assume that you have the following sequence of instructions:

```
lw $s2, 0($s1) (instr1)
add $s3, $s4, $s2 (instr2)
Sub $s6, $s2, $s3. (instr3)
```

Show the implementation through 5 stages and explain the implementation for both pipelined and non-pipelined design. (explain if there is any pipeline hazards)

(a & c repeated)

b) Given N·O of stages = 5

For non pipeline:

Instruction latency = (30+25+24+12+26) ns

= 111 ns

∴ Total execution time for (1000+03) = 1003 instructions = (111 × 1003) ns

= 111333 ns

For pipeline:

Instruction latency = (30 × 5) ns

= 150 ns

∴ Total execution time for 1003 instructions is

{(5 × 1) + (5-1) × 30} ns

= {5 + (4 × 30)} ns

= 125 ns

$$\text{Speedup} = \frac{111333\ ns}{125\ ns}$$

$$= 890.664$$