

- 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 your ID number) * consider there are 5 stages and each takes one clock cycle.
- * 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:

Iw \$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 ~~no~~ instructions No: X+7

$$\begin{aligned}
 &= 3+7 \\
 &= 10
 \end{aligned}$$

Let assume clock cycle time = 1 sec

No of stages = 5

Each stage takes 1 ~~clock~~ cycle ~~time~~

Now the page table ~~for~~ for this ~~is~~ is given below:

~~Instruction flow~~

| No of stage | IF | I ₁ | I ₂ | I ₃ | I ₄ | I ₅ | I ₆ | I ₇ | I ₈ | I ₉ | I ₁₀ | | | | | |
|------------------|----|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|-----------------|----------------|-----------------|-----------------|-----------------|--|
| 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 | | | | | |
| 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 $k = \text{number of stage}$

$n = \text{no of instructions}$

$$\begin{aligned}\text{pipeline execution time} &= (k \times 1) + (n-1) \times \text{clock cycle time} \\ &= (5 \times 1) + (10-1) \times 1 \\ &= (5 + 9) \times 1 \\ &= 14 \text{ sec}\end{aligned}$$

for Non pipeline

Execution Time

$$= (\text{No of stage} \times \text{clock cycle time})$$

$$\begin{aligned}&= (5 \times 1) \times 10 \\ &= 50 \text{ sec}\end{aligned}$$

$\times \text{No of instructions}$

$\boxed{\text{CPI}} = \frac{\text{Total clock cycle}}{\text{Total no. of instruction}}$

$$= \frac{14}{10}$$

$$= 1.4$$

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

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

$$= 3.57$$

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

$$= \frac{70}{50}$$

$$= 1.4$$

Comparison 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 for a non-pipelined machine executes only a single instruction at a time.

From the above calculation

here we can analyze that the execution time for non-pipelined machine is greater than pipelined machine. Therefore, pipelining provides more efficiency than non-pipelined architecture.

b) Given ~~clock~~

① No of instructions ≥ 3

Let assume three instructions

instr1 as I_1

instr2 as I_2

instr3 as I_3

No. of stages ≥ 5 (as required above)

Let assume clock cycle time ≥ 1 sec

Port Non pipelined instruction pipeline

| No of stage | IR | D1 | D2 | D3 | D4 | D5 | D6 | D7 | D8 | D9 | D10 | D11 | D12 | D13 | D14 | D15 |
|------------------|-------|-------|-------|-------|-------|-------|----|----|----|----|-----|-----|-----|-----|-----|-----|
| IR | I_1 | I_2 | I_3 | | | | | | | | | | | | | |
| ID | | I_1 | I_2 | I_3 | | | | | | | | | | | | |
| EXB | | | I_1 | I_2 | I_3 | | | | | | | | | | | |
| WB | | | | I_1 | I_2 | I_3 | | | | | | | | | | |
| clock cycle time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | |

Non pipeline execution time:

No of stage \times clock cycle time

\times No of instructions

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

$$= 5 \times 3$$

$$= 15 \text{ sec}$$

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

For pipeline

| No of stages | IF | I_1 | I_2 | I_3 | | | | |
|------------------|----|-------|-------|-------|-------|-------|-------|---|
| RD | | t_1 | t_2 | t_3 | | | | |
| BKB | | | I_1 | I_2 | I_3 | | | |
| MBN | | | | I_1 | I_2 | I_3 | | |
| WB | | | | | I_1 | I_2 | I_3 | |
| clock cycle time | | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

$$\begin{aligned}
 \text{# pipeline execution time} &= (k \times 1)(n-1) \times 1 \\
 &= (5 \times 1)(3-1) \times 1 \\
 &\approx (5 \times 2) \times 1 \\
 &\approx 10 \text{ sec}
 \end{aligned}$$

Now according to the previous

~~Efficiency~~

calculation we can get:

$$\begin{aligned}
 \text{speedup} &= \frac{\text{Non-pipelined machine execution time}}{\text{Pipelined execution time}} \\
 &= \frac{15 \text{ sec}}{10 \text{ sec}} \\
 &\approx 1.5
 \end{aligned}$$

So here pipelined machine will provide better performance than non-pipelined, as

here ~~it~~ 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.

With pipeline stages of size 1

Although we can find better performance

during pipelining but here for the above

MIPS instructions, we may face some

hazards at the end of the cycle.

As we know three types of hazards can be

happened in pipeline \rightarrow

Structural hazards

Control hazards

Data hazards

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

occur.

In pipeline, an instruction ~~in the~~ 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 ~~are~~ execute parallelly, then

We can find during load instruction the

value of register ~~is~~ is stored in a

temporary register, and after that that

The value will be loaded ~~to~~ to memory

or the value may be stored in that register. So, as ~~this~~ during pipelining

the stages will be overlapped so, if

that value is ~~set~~ in both places

at the same time then ~~we~~ during

addition when we ~~will~~ will write it

back then we can find the updated

value ~~in~~ in the register but if before

loading it to memory the addition instruction execute, then we may not get the updated value of that ~~register~~ had been stored in register '32'. So, here this type of hazard may occur during pipelining.

- 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 7 to 9, subtract -3 from 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) How does the following affect the charge?

$$\text{No. of instructions} = 3 + 5 \\ = 8$$

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 | | | | |
| ID | | I ₁ | I ₂ | I ₃ | I ₄ | I ₅ | I ₆ | I ₇ | I ₈ | | | |
| EXB | | | I ₁ | I ₂ | I ₃ | I ₄ | I ₅ | I ₆ | I ₇ | I ₈ | | |
| MEM | | | | I ₁ | I ₂ | I ₃ | I ₄ | I ₅ | I ₆ | I ₇ | I ₈ | |
| WB | | | | | I ₁ | I ₂ | I ₃ | I ₄ | I ₅ | I ₆ | I ₇ | I ₈ |
| Clock cycle time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

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

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

$$= (5+7) \times 1$$

$$= 12 \text{ sec}$$

Total time for Non-pipeline = $(\cancel{60} \times 1) \times 8$

$$= 40 \text{ sec}$$

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 ~~respectively~~ : 20 ns, 20 ns,

30 ns, 25 ns, 20 ns and 20 ns.

For non pipeline

$$\text{Instruction latency} = (20 + 20 + 30 + 25 + 20 + 20) \text{ ns}$$
$$= 135 \text{ ns}$$

Total execution time for 80 instructions
will be: $(135 \times 80) \text{ ns}$

$$= 10800 \text{ ns}$$

For pipeline

Instruction latency \approx (30×6) ns
 $\approx 180 \text{ ns}$

*max stage value from
the given
values*

Total execution time for 80 instructions - will be
 $\approx (5 \times (8 - 1)) \times 30$

$$= 5 + (5 \times 39)$$

$$= 155 \text{ ns}$$

Speedup

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

$$\cancel{20.144} \quad 69.67$$

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 $I = 6 + \text{last digit of your id.}$

$$\text{Q) No of instructions} = 6 + 3 \\ = 9$$

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 |

④ Total time for pipeline = $(k \times 1)H(n-1) \times$ clock cycle time

$$\approx (5 \times 1)H(9-1) \times 1$$

$$= \cancel{5 \times 2} + 5 + (8 \times 1)$$

$$\cancel{10 \text{ sec}}$$

$$\approx 13 \text{ sec}$$

Total time for non pipeline = ~~10~~ $(5 \times 1) \times 9$

$$\approx 45 \text{ sec}$$

⑤ CPI = $\frac{\text{Total clock cycle}}{\text{Total instruction}}$

$$\approx \frac{13}{9}$$

$$\approx 1.4$$

$$\approx 1$$

$$45 \text{ sec}$$

⑥ Speed up = $\frac{45 \text{ sec}}{13 \text{ sec}}$

⑦ 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.
- * 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.
- 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 ~~(a)~~ Instructions No: X+7

$$= 3 + 7$$

$$= 10$$

Let assume clock cycle time = 10 sec

No of stages = 5

Each stage takes 1 clock cycle ~~1~~

Now the page table for this ~~is~~ is given below:

4

for pipeline

Let k = number of stage

n , no of instructions

$$\begin{aligned}\text{pipeline execution time} &= (k \times 1) + (n-1) \times \text{clock cycle time} \\ &= (5 \times 1) + (10-1) \times 1 \\ &= (5 + 9) \times 1 \\ &= 14 \text{ sec}\end{aligned}$$

for Non pipeline

$$\text{Execution time} = \left(\text{No of stage} \times \text{clock cycle time} \right)$$

* No of
Instructions

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

$$= 50 \text{ sec}$$

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

$$= \frac{14}{10}$$

$$= 1.4$$

$$\approx 1$$

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

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

$$= 3.57$$

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

$$= \frac{70}{50}$$

$$= 1.4$$

Comparison between Pipeline machine and Non pipeline machines

Pipelining represent the scenario in such a way where we'll get 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 for 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 pipelined machine. Therefore, pipelining provides more efficiency and less execution time than non-pipelined architecture.

Given

b) No of execution stages = 5

For ~~non pipeline~~ non pipelining:

$$\text{Instruction latency} = (12 + 15 + 8 + 18 + 20) \text{ ns}$$
$$= 73 \text{ ns}$$

(12+15+8+18+20) = 73 ns

Execution time = 73 ns

Execution time = 73 ns

Total execution time for 10^3 instructions.

$$(73 \times 1000) \text{ ns}$$
$$= 73000 \text{ ns}$$

For pipeline:

$$\text{Instruction latency} = \frac{20}{5} = (20 \times 5) \text{ ns}$$
$$= 100 \text{ ns}$$

Total execution time for ~~10^3~~ 1000 instructions

= 1000 instructions : $(5 \times 1) + (5 - 1) \times 20$

$$= 5 + (4 \times 20)$$

$$= 5 + 80 = 85 \text{ ns}$$

~~85 ns~~

$$\text{Speedup} = \frac{73219 \text{ ns}}{85 \text{ ns}}$$

$$24 \times (12) + 861 \cdot 4$$

$$(24 \times 12) + 861$$

$$288 + 861$$

$$= 1149$$

b) Given ~~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 instruction

| No of stage | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|------------------|-------|-------|-------|-------|-------|-------|-------|---|---|----|----|----|----|----|----|
| DR | I_1 | I_2 | I_3 | | | | | | | | | | | | |
| ID | | I_1 | I_2 | I_3 | | | | | | | | | | | |
| EXB | | | I_1 | I_2 | I_3 | | | | | | | | | | |
| ALU | | | | I_1 | I_2 | I_3 | | | | | | | | | |
| WB | | | | | I_1 | I_2 | I_3 | | | | | | | | |
| clock cycle time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

Non pipeline execution time =

No of stage \times clock cycle time

\times No of instructions

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

$$= 5 \text{ sec}$$

$$= 15 \text{ sec}$$

~~Efficiency~~ \rightarrow Total block in Page table

~~Total used blocks~~

5

2

For pipeline

| No of stages | I_1 | I_2 | I_3 | | | | |
|------------------|-------|-------|-------|-------|-------|-------|-------|
| IF | T_1 | T_2 | T_3 | | | | |
| RD | | T_1 | T_2 | T_3 | | | |
| BKB | | | T_1 | T_2 | T_3 | | |
| MBN | | | | T_1 | T_2 | T_3 | |
| WB | | | | | T_1 | T_2 | T_3 |
| clock cycle time | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

$$\begin{aligned}
 \text{Pipeline execution time} &= (R \times 1) (n-1) \times 1 \\
 &= (5 \times 1)(3-1) \times 1 \\
 &= (5 \times 2) \times 1 \\
 &= 10 \text{ sec}
 \end{aligned}$$

Now ~~as~~ according to the previous

~~Efficiency~~

culation we can get

$$\begin{aligned}
 \text{speedup} &= \frac{\text{Non-pipelined machine execution time}}{\text{pipelined execution time}} \\
 &= \frac{15 \text{ sec}}{10 \text{ sec}} \\
 &= 1.5
 \end{aligned}$$

So here pipelined machine will provide better performance than non-pipelined. as

here ~~as~~ 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 but here for the above MIPS instructions, we may face some hazards.

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

- (a) structural hazards
- (b) control hazards
- (c) data hazards

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

in pipeline, an instruction ~~in the~~ 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~~ is stored in a temporary register, and after that that value will be loaded ~~to~~ to memory or the value may be stored in that register. So, as ~~this~~ during pipelining the stages will be overlapped so, if that value is ~~stored~~ in both places at the same time then ~~here~~ during addition when we ~~will~~ will write it back then we can find the updated value ~~in~~ 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 '32'. 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 your 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:

Iw \$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)

b) Given No of stages = 5

For pipeline

$$\text{Instruction latency} = 2(35+40+45+30+20) \text{ ns}$$
$$= 170 \text{ ns}$$

∴ Total ~~inst~~ execution time for (1010101) = 1013

$$\text{instructions: } (170 \times 1013) \text{ ns}$$
$$= 172210 \text{ ns}$$

For pipeline:

$$\text{Instruction latency} = (45 \times 5) \text{ ns}$$
$$= 225 \text{ ns} \quad (\text{P.T.O})$$

Total execution time to 13 instructions

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

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

$$= 5 + 180$$

$$= 185 \text{ ns}$$

$$\text{Speedup} = \frac{17220 \text{ ns}}{185 \text{ ns}}$$

$$= 930.8649$$

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 your ID number) * consider there are 5 stages and each takes one clock cycle.
- 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.
- Find the instruction latency on this machine.
 - How much time does it take to execute $(\bar{I} + 1000)$ 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 $(\bar{I} + 1000)$ instructions?
Also calculate the speedup.
(where \bar{I} is the last two digits of your ID number)
- c) This question considers the basic MIPS, 5-stage pipeline (IF, ID, EXE, MEM, WB).
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 No of stages = 5

For non pipeline:

$$\text{Instruction latency} = (30 + 25 + 24 + 12 + 26) \text{ ns}$$

$$= 111 \text{ ns}$$

∴ Total execution time for (1000+03) = 1003

$$\text{instructions} = (111 \times 1003) \text{ ns}$$
$$\Rightarrow 11333 \text{ ns}$$

For pipeline:

$$\text{Instruction latency} = (30 \times 5) \text{ ns}$$

$$= 150 \text{ ns}$$

for

∴ Total execution time \uparrow 1003 instructions

$$\begin{aligned} & \{ (5 \times 1) + (5-1) \times 30 \} \text{ ns} \\ & = \{ 5 + (4 \times 30) \} \text{ ns} \\ & = 125 \text{ ns} \end{aligned}$$

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

$$= 890.664$$

So for 1 segment 890.664 times faster

For 2 segments 890.664 times faster

For 3 segments 890.664 times faster

For 4 segments 890.664 times faster

For 5 segments 890.664 times faster

For 6 segments 890.664 times faster

For 7 segments 890.664 times faster

For 8 segments 890.664 times faster

For 9 segments 890.664 times faster

For 10 segments 890.664 times faster

For 11 segments 890.664 times faster

For 12 segments 890.664 times faster