HW3 CS2323

Name:- Darpan Gaur
Roll No:- CO21BTECH11004

DATE: / /
PAGE:

Que 1 Delays → 130, 125, 110, 132, 135, 145 ps

$n = 1000$, $k = 6$

n → no. of independent instructions

k → no of stages in pipeline

Delay due to registers used between pipeline stages = 12 ps

a) Cycle Time = Max delay (stages) + Delay due to regis

= Max (130, 125, 110, 132, 135, 145) + 12

= 145 + 12

= 157 ps

No of clock cycles = $(n + k - 1)$

= 1000 + 6 - 1

= 1005

∴ Time taken = No. of cycles × Cycle time

= 1005 × 157

= **157785 ps**

b) Cycle time = max delay (stages) + delay time due to registers

= max(130, 125, 110, 132, 135, 145) + 12

= 145 + 12 = 157 ps

No of cycles = $(1000 + 0.2 \times 1000 + 6 - 1)$

20% inst have stall of 1 cycle

= 1205

∴ Time taken = No of cycles × Cycle time

= 1205 × 157

= **189185 ps**

DATE: / /
PAGE:

Que 2 a) 5 stages pipeline ∴ k = 5

latency (P_1) = 800 ps

∴ cycle time for P_1 = $\frac{800}{5}$ = 160 ps [As, total latency given equally divided btw pipeline stages]

∴ Clock period of P_1 = 160 ps

P_2 → Increase in latency of ALU by 250 ps

∴ Cycle time for P_2 → latency of ALU

= 250 + 160 = 410 ps

∴ clock period of P_2 = 410 ps [As all other stages have latency 160 ps]

Minimum clock period required for P_1

b) Let say code have n instructions.

For P_1 No. of cycles = n + k - 1 = n + 4

For P_2 No of cycles = n − 0.1 × n + k − 1 = 0.9n + 4

(0.1 × n: 10% reduced due to adding MULT)

Time taken for P_1 = (n + 4) 160 ps = 160n + 640

Time taken for P_2 = (0.9n + 4) 410 ps = 369n + 1640

∴

As 369n + 1640 > 160n + 640

∴ Processor P_1 execute a given code in smaller time.

DATE: / /
PAGE:

c) In part (b) we found out, time taken for P_1 & P_2 if code has n instructs.

$P_1 \rightarrow (160n + 640)$ ps
$P_2 \rightarrow (369n + 1640)$ ps

Here $n = 5000$,
∴ Time taken for P_1
$\Rightarrow 160 \times 5000 + 640 =$ **800640 ps**

∴ Time taken for P_2
$\Rightarrow 369 \times 5000 + 1640 =$ **1846640 ps**

Que 3 a) NO forwarding or Hazard detection is implemented

```
add x14, x12, x11   } → 2 NOP insert between them
add x15, x14, x12   }
```

as first add stores result in x14 which is written in write back stage and second add fetch x14 in ID state.

So if 2 NOP are inserted writeback & ID state for first add & second add respective occurs at same time [write first then read (first half of clock cycle) (second half of clock cycle)]

```
ld x13, 8(x13)   } → 1 NOP insert here.
ld x12, 0(x14)   }
and x13, x15, x13
```

As first load stores in x13 which is written in write back stage and read x13 in ID stage.

So if 2 NOP issue solve. But already a load btw them ∴ 1 NOP required.

DATE: / /
PAGE:

```
and x13, x15, x13
ld  x11, 4(x13)
sd  x13, 0(x15)
```

→ Insert 2 NOP Here.
As add stores result in x_{13} in writeback stage & load read x_{13} register in ID stage.

So 2 NOP required between them

Code after inserting NOPs.

```
add x14, x12, x11
NOP
NOP
add x15, x14, x12
ld  x13, 8(x13)
ld  x12, 0(x14)
NOP
and x13, x15, x13
NOP
NOP
ld  x11, 4(x13)
sd  x13, 0(x15)
```

Total 5 NOPs are inserted for correct execution

b) Data forwarding without hazard detection

```
add x14, x12, x11
add x15, x14, x12
```

: No NOP needed as forwarding x_{14} after ALU of first add to ID for second add

```
ld  x13, 8(x13)
ld  x12, 0(x14)
and x13, x15, x13
ld  x11, 4(x13)
sd  x13, 0(x15)
```

• NO NOP As ld x12, 0(x14) present So ld x13, 8(x18) in MEM stage forward x13 to and x13, x15, x13 in ID stage before ALU.

• No NOP needed After ALU and forward x13 to ID in load.

So no NOP required as forwarding takes care