HomeWork_6

Name： 卫焱滨（Wei Yanbin）

SID：11710823

chapter 4

Exercise 4.8

IF	ID	EX	MEM	WB
250ps	350ps	150ps	300ps	200ps

Also, assume that instructions executed by the processor are broken down as follows:

alu	beq	lw	sw
45%	20%	20%	15%

§ 4.8.1

For pipelined processor :

clock cycle time = Max (IF + ID + EX + MEM + WB)

= The max time of these 5 stages.

= 350 ps

For non-pipelined processor :

clock cycle time = The sum of the Time of these 5 stages.

= 250ps + 350ps + 150ps + 300ps +200ps = 1250 ps

§ 4.8.2

For pipelined processor :

The LW instruction time = 5* clock cycle time

= 5 * 350 = 1750 ps

For non-pipelined processor :

The LW instruction time = 1 * clock cycle time

= 1* 1250 ps = 1250 ps

§ 4.8.3

To make the cycle time less, we need to split the Instuction which has longest latency ——ID.

By split the ID into two stages, the most long latency becomes 300ps for MEM stage.

So the new cycle time is 300 ps.

§ 4.8.4

Only lw and sw using the data memory.

SO the utilization of data memory is : 20% + 15% = 35%

§ 4.8.5

alu Instructions and lw instructions both use write-register port of the “Registers” unit .

So the utilization of write-register port of the “Registers” is : 45% + 20 % = 65 %

§ 4.8.6

We already computed clock cycle times for pipelined and single cycle organizations, and for multiple cycle organizations, It share the same clock cycle time as pipelined . So :

1) Compare clock cycle times :

For **single-cycle** organization clock cycle time is 1250 ps

For **pipelined** and **multiple cycle** organization clock cycle time is 350 ps

2)

① Compare the execution times between Single-cycle and pipeline:

By 1250 / 350 = 3.57, Single-cycle execution time is 3.57 times pipelined execution time.

② Compare the execution times between multiple-cycle and pipeline:

In pipelined, In every cycle there is a instruction completes. While for multiple cycle organization :

An ALU instruction needs 4 cycles (no MEM) And it has frequency 45 %

A BEQ is 3 cycles (no MEM and WB) And it has frequency 20 %

A LW needs all 5 cycles And it has frequency 20 %

A SW need 4 cycles (no WB) And it has frequency 15 %

Thus we get multiple cycle has 45 % * 4 + 20 % * 3 +20 % * 5 + 15 % * 4 = 4.00 times of execution t

-ime of pipeline.

Exercise 4.9

§ 4.9.1

Data dependence：

Read after Wirte（RAW） on R1 from I1 to I2 and I3

Read after Wirte (RAW） on R2 from I2 to I3

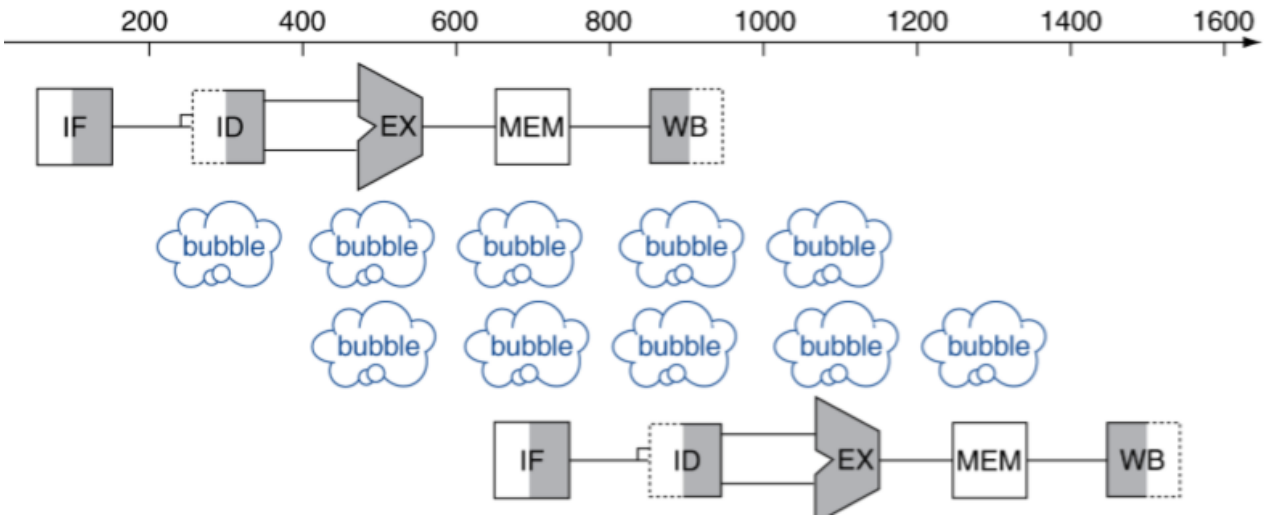
Write after Read (WAR) on R2 from I1 to I2

Write after Read (WAR) on R1 from I2 to I3

Write after Write (WAW) on R1 from I1 to I3

§ 4.9.2

Because "Write after Read(WAR)" and "Write after Write(WAW)" do not cause any stall(nop). However, "Read after Write(RAW)" can cause two stalls(here nop) between (the instrction which write the register) and (the instrction which read the register) as:



So by the above analysis, The code to eliminate Hazards should be :

```
or r1,r2,r3
nop                 //Delay I2 to avoid Read after Write hazard on R1 from I1
nop
or r2,r1,r4
nop                 //Delay I3 to avoid Read after Write hazard on R2 from I2
nop
or r1,r1,r2
```

§ 4.9.3

For Full forwarding :

ALU instuctions can avoid the hazard by using forwarding. Load-use hazard still has a stall though use fprwarding.

Because three instrctions I1,I2,I3 are all ALU instructions, there is no load-use hazard and all hazard can be eliminate by forwarding without any nop. So we just the maintain the original code is ok:

```
or r1,r2,r3
or r2,r1,r4
or r1,r1,r2
```

§ 4.9.4

By above questions, we have known that for no forwarding, we need 3(or) + 4(nop) =7 instrctions, so 7 + 4 cycles. Each cycle 250 ps.

And for full forwarding , 3 instrctions becomes 3 + 4 cycles. Each cycle 300 ps.

1) Execution time :

for no forwarding : (7 + 4) * 250 = 2750 ps

for with full forwarding : (3 + 4) * 300 = 2100 ps

2) speedup = 2750ps / 2100ps = 1.31

§ 4.9.5

For ALU-ALU forwarding only :

It can avoid the data hazards between two ALU instructions. Because three instrctions I1,I2,I3 are all ALU instructions, so all hazards can be solved by only use ALU-ALU forwarding. So we just the maintain the original code is ok:

```
or r1,r2,r3
or r2,r1,r4
or r1,r1,r2
```

§ 4.9.6

By above questions, we have known that for no forwarding, we need 3(or) + 4(nop) =7 instrctions, so 7 + 4 cycles. Each cycle 250 ps.

And for ALU-ALU only forwarding , 3 instrctions becomes 3 + 4 cycles. Each cycle 290 ps.

1) Execution time :

for no forwarding : (7 + 4) * 250 = 2750 ps

for with full forwarding : (3 + 4) * 290 = 2030 ps

2) speedup = 2750ps / 2030ps = 1.35