NWEN 242 LAB2

Vincent Yu——ID:300390526

October 11, 2017

4.1

source code

```
start:
addi $5, $5, 5
addi $1, $1, 1
addi $3, $3, 3
nop
nop
add $2, $5, $1
add $4, $2, $3
nop
nop
nop
END
```

a

I count the number of cycles from the beginning of the code. For each mode we all have to use 10 cycles.

b

For basic mode, after 8 cycles the value of $2 is needed in the second instruction.
For insert bubbles mode,after 10 cycles the value of $2 is needed in the second instruction.
Note: The next instruction has a stalled at decoded for two bubbles.
For forwarding mode, after 8 cycles the value of $2 is needed in the second instruction.
Here I used the same source code for these three modes. But for forwarding mode if I reduce the nops between the last addi and the first add, it can reduce cycles to 8. But in order to have correct value, we can't reduce the nops for basic mode. For insert bubbles mode it will be the same cycles as before.

c

The hazard is date hazards.
The value of destination register in the first line is needed as a source register in the second instruction. So if we use the basic mode it hasn't been updated but it has already been used in the next instruction.

4.2

a

For basic mode, it takes 5 cycles. For insert Bubbles mode, it takes 5 cycles. For forwarding mode, it only takes 4 cycles.

b

For insert bubbles mode it will be an infinite loop as expected. But for forwarding and basic mode it will reach the end. And the value store in $2 will be 2.

c

This is a control hazard. For Basic and Forwarding mode the next addi instruction is being decoded while the beq is calculating. So it will keep on finishing the whole add instruction when the branch is taken. It will change the value stored in $2 from zero to 1. So the next time meet the beq instruction the prediction will be not taken. Then it will keep on finishing the following instructions. But for insert bubbles mode, it will be an infinite loop as expected, because it automatically inserts bubbles between beq and add instructions.So the addi instruction won't reach decode stage.

d

We can simply use branch assumption which will flash the following instruction. Another way is inserting a nops between beq instruction and add instruction.

4.3

a

For insert bubbles mode, after 12 cycles $3 will receive the correct value.
For basic mode, after 13 cycles $3 will receive the correct value.
Note: I insert three nops between addi and sw.
For forwarding mode, after 10 cycle $3 will receive the correct value.

b

For basic mode,after 14 cycles $3 will be needed for next instructions
For forwarding mode, after 10 cycles $3 will be needed for next instruction.
For insert bubble mode,after 10 cycles $3 will be needed for next instruction.

c

The problem here is we have two hazard. The first one is between addi and sw instructions. The destination register of addi instruction is needed as source register for sw instruction. The value stored in $2 is requested before the new value write back. The same problem happens between lw and addi instructions. The value of $3 haven't been updated when the next instruction need it as a source register. This is a Data Hazard.

d

For forwarding mode we can just insert one nop between lw and addi instructions.
For basic mode we can insert three nops between addi and sw. And insert three nops between lw and addi to make the $4 can receive the correct result.
For insert bubble mode we can just use the original code without any change.

5.1

Mode Option	Instructions	Cycles	CPI
Basic	205	211	1.029268264
Bubbles with Branch Assumption	76	154	2.026316
Forwarding	107	113	1.056074738
Forwarding with Branch Assumption	97	109	1.123711347

5.2

a

From the table above we can find

b

5.3

Mode Option	Instructions	Cycles	CPI
Forwarding Optimized	97	103	1.061856
Forwarding with Branch Assumption Optimized	87	99	1.137931