NWEN242 Lab 2

David Barnett (300313764)

Preparatory Questions

Q1

- Structural hazard: Using the same resource, such as memory, at the same time. An example would be reading & writing to registry file at the same time.
- Data hazard: Accessing data that has been modified by another instruction.
- Control hazard: The hazard introduced through branching instructions with a chance of having the wrong instruction in the pipeline. This can be caused by assuming the branch will not be taken but in running it is taken.

Q2

- Splitting reading & writing to the registry file to the start & end of the cycle to solve a structural hazard and help towards data hazard.
- Forwarding data computed but not written to the registry to instructions in earlier parts of the pipeline.
- The hazard control unit inserts bubbles (NOP) when the instruction depends on the result of the next instruction but cannot be forwarded the result, such as loading from memory.

Q3

Number of stages in the pipeline minus one, so the instruction causing the hazard will be completed as the next instruction can safely run after it.

Q4

The special registry file allows for writes at the start of the cycle and reads at the end of the cycle. This allows for a write back and instruction decode to happen at the same time with the register as the target of the write back and a source for the decoding instruction.

Q5

The forwarding unit would forward the value from the MEM/WB stage back to the EX stage. The forwarding unit itself does not handle any special case for `lw` unlike the Hazard Detection unit.

Q6

No bubbles are needed when two hazardous R-Type instructions are run in sequence as the forwarding from EX/MEM will overwrite the now out-of-date values gotten from registry file during the EX stage

Q7

No stalls are required when using a memory instruction after a R-type value as the forwarding unit will update the value if the memory instruction is dependent on the previous R-type instruction.

Q8

With a forwarding unit one nop is needed when a R-Type instruction follows and is dependent on the previous memory load instruction due to the value not being available until MEM/WB stage.

Q9

Given that the forwarding unit can forward to the value to the ID stage the number of NOPs would be 1 due to forwarding from the EX/MEM stage to the ID stage.

Q10

There needs to be no NOPs between the branch and R-type instruction as the result can be forwarded in the EX/MEM stage to the ID stage.

Q11

There needs to be two NOPs between the lw and a following branch instruction because the lw's result is only available for forwarding after MEM/WB.

Exercise 4.1

A

Basic

At cycle 10 of the program the register `$2` is updated to `6`

Bubbles

At cycle 10 of the program the register `$2` is updated to `6`

Forwarding

At cycle 10 of the program the register `$2` is updated to `6`

Unmodified

At cycle 10 of the program the register `$2` is updated to `6`

B

Basic

The value of register `$2` is needed at cycle 11 of the program

Bubbles

The value of register `$2` is needed at cycle 10 of the program

Forwarding

The value of register `$2` is needed at cycle 8 of the program but then updated to the new value at cycle 9 via forwarding

Unmodified

The value of register `$2` is needed at cycle 8 of the program

C

The problem is that there is a data hazard. This is due to the second `add` is dependent on the result of the previous `add` instruction. In the case of the unmodified version the wrong value of `$2` is used because the result has not been written back to the registry file yet.

Exercise 4.2

A

B

C

Exercise 4.3

A

B

C

Exercise 5.1

D

E

F

G

H

Exercise 5.2

Exercise 5.3