COE 301 – Computer Organization

Assignment 7: Pipelined Processor

1. Identify all the RAW data dependencies in the following code. Which dependencies are data hazards that will be resolved by forwarding? Which dependencies are data hazards that will cause a stall? Using a graphical representation of the pipeline, show the forwarding paths and stalled cycles if any.

```
add $3, $4, $2
sub $5, $3, $1
lw  $6, 200($3)
add $7, $3, $6
```

2. We have a program of 10^6 instructions in the format of "**`lw,add,lw,add,…`**". The **`add`** instruction depends only on the **`lw`** instruction right before it. The **`lw`** instruction also depends only on the **`add`** instruction right before it. If this program is executed on the 5-stage MIPS pipeline:

a) Without forwarding, what would be the actual CPI?
b) With forwarding, what would be the actual CPI?

3. A 10-stage instruction pipeline runs at a clock rate of 1 GHz. The instruction mix is such that 15% of instructions cause one bubble to be inserted into the pipeline, and 10% of instructions cause two bubbles to be inserted. The equivalent single-cycle implementation would lead to a clock rate of 150 MHz.

a) What is the increase in the pipeline CPI over the ideal CPI as a result of bubbles?
b) What is the speedup of pipelined implementation over single-cycle?

4. **Store-after-load data dependence.** Consider copying an array of *n* words from one address in memory to another. This can be accomplished by placing a sequence of **lw** and **sw** instructions in a loop, with each loop iteration copying one word. In the current pipelined implementation shown in the lecture slides, this leads to one bubble (stall cycle) between **lw** and **sw**. Is it possible to avoid this stalling via additional data forwarding hardware? Discuss how this can be done or explain how the bubble is unavoidable.

5. We have a program core consisting of five conditional branches. The program core will be executed millions of times. Below are the outcomes of each branch for one execution of the program core (T for taken and N for not taken).

Branch 1: T-T-T
Branch 2: N-N-N-N
Branch 3: T-N-T-N-T-N
Branch 4: T-T-T-N-T
Branch 5: T-T-N-T-T-N-T

Assume that the behavior of each branch remains the same for each program core execution. For dynamic branch prediction schemes, assume that each branch has its own prediction buffer and each buffer is initialized to the same state before each execution. List the predictions and the accuracies for each of the following branch prediction schemes:

a) Always taken
b) Always not taken
c) 1-bit predictor, initialized to predict taken
d) 2-bit predictor, initialized to weakly predict taken

Prepared by Dr. Muhamed Mudawar