CS 2200 - Introduction to Systems

Fall 2016

Homework 6

Rules:

- Please print a copy of the assignment and handwrite your answers. No electronic submissions are allowed. **Please print as one double-sided page.**
- This is an individual assignment. No collaboration is permitted.
- Due Date: **19th October 2016** – 6:05 PM (Start of recitation). Bring your BuzzCards.

Name (please print): CS2200 TAs **GTLogin:** cs2200 **Section** CS2200

[I] For the program in number [II], write down all the instructions that are dependent on another instruction, and the kind of dependence they exhibit. Also write the potential hazard they can cause in any pipeline. (Hint: There are 3 kinds of dependencies) **[30 Pts]**

Format: Say you had the below program, the answer should be formatted as:

```
add  R1, R2, R3    ; I1
addi R4, R1, 1     ; I2

I2 depends on I1 : Pure dependence : RAW hazard
```

I3 depends on I1: pure dependence: RAW hazard

I4 depends on I2: pure dependence: RAW hazard

I5 depends on I4: pure dependence and anti-dependence: RAW/WAW hazards

I6 depends on I5: pure dependence: RAW hazard

I7 depends on I2: pure dependence and anti-dependence: RAW/WAW hazard

I8 depends on I3: anti-dependence: WAR hazard

[II] You are given a five stage pipeline with the following features: **[70 Pts]**

- There is Data forwarding from EX and MEM to ID/RR
- There is a static predictor in the IF stage, that always predicts the branch as not taken
- Branches are resolved in the ID/RR stage
- Writes happen before reads (i.e. Register writes precede register reads)

The following sequence of instructions is run on this pipeline, fill out the cycle by cycle waterfall diagram. Assume all registers are initialized to zero, and the static predictor knows which addresses are branches.

```
addi R1, R0, 10                  ; I1
la   R3, arr                     ; I2
loop:   beq  R2, R1, end         ; I3
        ldr  R4, 0(R3)           ; I4
        addi R4, R4, 1           ; I5
        STR  R4, 0(R3)           ; I6
        addi R3, R3, 1           ; I7
        addi R2, R2, 1           ; I8
        beq  R0, R0, loop        ; I9
end:    halt                     ; I10

arr: 0x4000
```

Cycle	IF	ID/RR	EX	MEM	WB
1	I1				
2	I2	I1			
3	I3	I2	I1		
4	I4	I3	I2	I1	
5	I5	I4	I3	I2	I1
6	I6	I5	I4	I3	I2
7	I6	I5	NOP	I4	I3
8	I7	I6	I5	NOP	I4
9	I8	I7	I6	I5	NOP
10	I9	I8	I7	I6	I5
11	I10	I9	I8	I7	I6
12	I3	NOP	I9	I8	I7
13	I4	I3	NOP	I9	I8
14	I5	I4	I3	NOP	I9
15	I6	I5	I4	I3	NOP
16	I6	I5	NOP	I4	I3
17	I7	I6	I5	NOP	I4
18	I8	I7	I6	I5	NOP
19	I9	I8	I7	I6	I5
20	I10	I9	I8	I7	I6

.

.

..

.