CDA3101
Assignment 5
Pipelining Exercises

Objectives. Learn how instructions are pipelined and branches are predicted.

Solve the following problems. You must show your work in order to receive full credit.
(1) Fill in the traditional pipeline diagram that shows in which cycle each of the instructions in problem 1 will be in each stage of the 5-stage MIPS pipeline discussed in class. (25 points)

cycle	1	2	3	4	5	6	7	8	9	10
`or  $t4,$t2,$t4`	IF	ID	EX	MEM	WB					
`lw  $t3,0($t4)`		IF	ID	EX	MEM	WB				
`add $t5,$t3,$t2`			IF	ID	stall	EX	MEM	WB		
`sw  $t3,0($t5)`				IF	stall	ID	EX	MEM	WB	

(2) Consider a code sequence consisting of 100 consecutive `lw` instructions. Except for the first instruction, each instruction uses the register that was loaded in the previous instruction as shown below. What would be the total number of cycles required for this sequence of instructions to execute given the pipelined datapath of Figure 4.65 on page 325 of your text? (25 points)

```
lw $t2,0($t1)
lw $t2,0($t2)
...
```

cycle	1	2	3	4	5	6	7	8	9	10
`lw $t2,0($t1)`	IF	ID	EX	MEM	WB					
`lw $t2,0($t2)`		IF	ID	stall	EX	MEM	WB			
`lw $t2,0($t2)`			IF	stall	ID	stall	EX	MEM	WB	

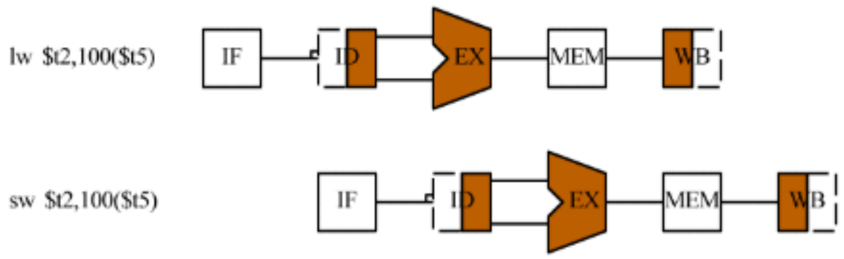
First instruction takes 5 cycles. Every other instruction takes 2 additional cycles to complete after the previous instruction.

total_cycles = 5 + 99*2 = 5 + 198 = 203

(3) Consider an instruction sequence used for a memory-to-memory copy.

```
lw   $t2,100($t5)
sw   $t2,200($t6)
```

The value in `$t2` to be stored into memory by the `sw` is actually not needed until *MEM* stage of the MIPS pipeline. Below is a drawing showing the two instructions being pipelined. Modify this figure in a manner similar to Figure 4.29 on page 279 of your text to show how the value could be forwarded to avoid a stall.



Rewrite the following stall formula so this code sequence won't stall. (25 points)

```
if (ID/EX.MemRead AND                                  // lw in EX stage?
    (ID/EX.RegisterRt == IF/ID.RegisterRs OR           // same register?
     ID/EX.RegisterRt == IF/ID.RegisterRt) AND         // same register?
    ID/EX.RegisterRt != 0 AND                          // load dst not r0?
    NOT (IF/ID.MemWrite AND                            // sw in ID stage?
         ID/EX.RegisterRt == IF/ID.RegisterRs))        // reg to be stored?
   stall the pipeline
```

This should be IF/ID. opcode == 0x2B

31 26
0 1 1

In fact we need to compare 3bits to determine whether the instruction is store

(4) Fill in the following branch prediction information for the table below for the various types of branch predictors for the predictions of a single branch in an application. Assume the initial state of the 1-bit predictor is not taken and the initial state of the 2-bit predictor is the bottom right state in Figure 4.63. (25 points)

branch result	1-bit predictor		2-bit predictor	
	prediction	correct?	prediction	correct?
taken	not taken	no	not taken	no
taken	taken	yes	taken	yes
not taken	taken	no	taken	no
taken	not taken	no	taken	yes
taken	taken	yes	taken	yes