Name: ID:

Problem 1)

Pipeline	Total single-cycle datapath latency (Total useful work)	Latch latency	CPI
5 stages	2500 ps	50 ps	1.2
14 stages	2500 ps	50 ps	1.5

i) Give the maximum frequency for the 5 stage and 14 stage pipeline implementations.

ii) Give the speedup of the 14 stage pipeline implementation over the 5 stage pipeline implementation.

Problem 2)
Consider the following pipeline:

Fetch	Delay	Decode	Delay2	Execute	Delay3	Memory/ Branch	Delay4	Write Back

i) Assuming branch outcomes are determined in the Memory stage, and mispredicted branches are corrected in that stage, how many instructions must be flushed upon a mispredicted branch?

ii) Assume no other hazards, what is the CPI for this pipeline, assuming 20% of instructions are branches and assuming 90% of branches are correctly predicted?

Name: ID:

iii) Assume load results are returned to the pipeline in the Memory stage. How many stall cycles will occur when a load is immediately followed by an instruction that consumes the result of the load.

Example:

```
lw  R1, 0(R2)
add R3, R1, R4
```

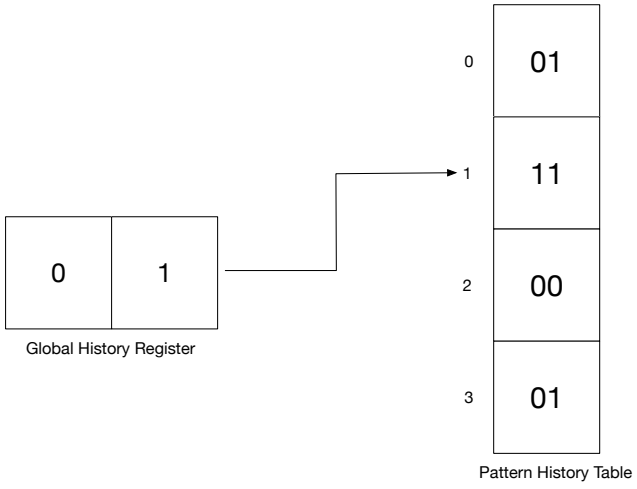
Problem 3)
Use the following pseudo-assembly code for the following problem:

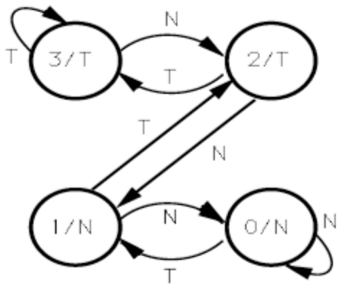
```
     LW R1, 0(R8) ; load X into R1
     Branch to FOO if (taken if R1 is even multiple of 2) ; b0
     ADDI R3, R3, 1
FOO: Branch to SNA if (taken if R1 is even multiple of 4) ; b1
     ADDI R4, R4, 1
SNA: ADDI R8, R8, 4 ; move to next X
```

Assume the above sequence is in a loop and completely ignore the loop back branch (not shown). Assume R0 initially holds the value 0 and the contents of memory are shown:

Memory value:	6	8	7	12
Address:	0	4	8	12

Consider the GAg predictor with a 2-bit global history register. Assume the actual outcome of each new branch is shifted in from the left. Assume each entry in the pattern history table is a two-bit, saturating up-down counter.





Automaton A2
(2-bit Saturating Up-down Counter)

Name: ID:

i) Complete the following:

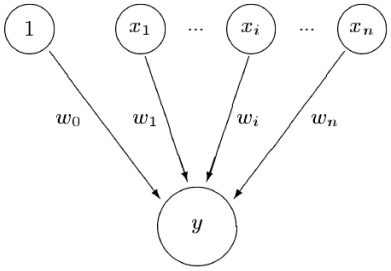
Branch		6	8	7	12
b0	History (before branch)				
	PHT value (before branch)				
	Prediction				
	Actual Outcome				
	PHT value (after branch)				
	History (after branch)				
b1	History (before branch)				
	PHT value (before branch)				
	Prediction				
	Actual Outcome				
	PHT value (after branch)				
	History (after branch)				

ii) Explain how *aliasing* in a branch predictor can cause mispredictions.

Name: ID:

Problem 4). Assume the following branch is predicted by a perceptron predictor such as the one discussed in class and in the paper on Canvas.

BGT R1, 3, TARG ; branch to TARG if R1 > 3



$$y = w_0 + \sum_{i=1}^{n} x_i w_i$$

```
for each bit in parallel
    if t = x_i then
        w_i := w_i + 1
    else
        w_i := w_i − 1
    end if
```

Perceptron Model (from Jimenez and Lin)

For the following values of R1, show in a table, the values of the weights, the output Y, the prediction (Taken vs. Not taken) and the actual output. Assume four bits of history are used. Start with a history of 0101. (Assume x4 is the input for the most recent branch and new, actual branch outcomes are shifted in on the right (i.e. history bits are shirted to the left).

X	w0 (bias)	w1	w2	w3	w4	y	Prediction	Outcome
4	1	2	3	4	-1			
7								
3								
1								
4								
7								
5								

Name: ID:

Problem 5)

i) For the instructions below, show in which cycle each instruction would issue on a traditional, in-order 5-stage pipeline. Show which cycles are stalls in which no instructions are issued. Assume the first instruction is issued in cycle 1.

```
lw  r3, 4(r2)
add r4, r3, r7
lw  r5, 0(r2)
sub r7, r5, r10
lw  r4, 0(r2)
lw  r11, 24(r4)
and r7, r5, r10
beq r11, r12, Loop
```

ii) Reorder the instructions above to minimize the number of stall cycles on our example 5-stage RISC pipeline.

Name: ID:

Problem 6) Consider the state of the following out-of-order processor below. The Reservation Stations, Register Alias Table and Reorder Buffer are shown. Notice that architectural register names have already been reordered to physical register names. Assume the instructions are numbered in program order (i.e. 101 comes before 102, etc.). Consider also the latencies given in the table for each instruction. The latencies in the table as such that if an instruction with a latency of 1 issues in cycle i, a consumer can issue in cycle i+1.

	Waiting Src1	**Waiting Src2**	**Dest**
101 add	-	-	PR9
102 mul	PR9	-	PR3
103 branch	PR9	PR3	-
105 add	-	-	PR4
106 store	-	PR4	-

Reservation Stations

R1	PR3
R2	PR9
R3	AR3
R4	PR5

RAT

			106 Store	105 Branch	104 Load	103 Branch	102 mul	101 add

ROB

opcode	**latency**
add	2
mul	3
branch	- (no register destination)
Store	- (no register destination)

Name: ID:

For the instructions in the Reservation Stations in the digram on the previous page, show in which cycle each instruction would issue assuming the latencies above.