Homework 3

Problem 1 (2 points): Consider a dynamically scheduled superscalar (2-wide). The processor has the following characteristics:

The processor pipeline is capable of fetching two instructions per cycle. Once instruction(s) are fetched, they can be decoded in the next cycle, and issued in the cycle after that if resources are available (otherwise, wait).

There are three pipelined execution units:

- One ALU: that can execute both integer and floating point add, sub, not, and, or. Execution takes 1, and 4 cycles, for integer and floating point, respectively.
- One load/store: execution takes 2 cycles for load/store.
- One multiply/divide unit: that can execute both integer and floating point multiply and divide. Execution takes 1, and 4 cycles, for integer and floating point, respectively.

Each unit can begin execution of one instruction in each cycle, so the processor can begin to execute 3 instructions in a cycle if instructions of the correct type of are ready to execute. There are:

- 4 unified reservation stations (numbered 0, 1, 2, 3),
- 8 ROB(Re-Order Buffer) entries (numbered 0, 1, …, 7),
- one load-store queue with 3 entries (numbered 0, 1, 2).
- 2 CDBs (Common Data Bus): each instruction writes the result on the CDB in the cycle that follows the one in which it completes execution.
- An instruction becomes eligible for execution in the cycle that follows the one in which its last operand was broadcast on the CDB.
- An instruction becomes eligible for commit (if other constraints permit it) in the cycle after it has broadcast its results to the CDB (not the same cycle).
- The processor can commit up to two instructions per cycle.

Note: integer operations are indicated by "i" at the end of the opcode.

Assuming that no instruction has yet been fetched at the beginning, for each instruction determine the cycle in which it is fetched (**T**$_{\textbf{Fet}}$). decoded **(T**$_{\textbf{Dec}}$**)**, issued (**T**$_{\textbf{Iss}}$), begins executing (**T**$_{\textbf{Exe}}$), uses the CDB(**T**$_{\textbf{CDB}}$), and commits(**T**$_{\textbf{Com}}$). For instruction issue, also specify which reservation station(**RS**), load/store queue entry (**LSQ**), and ROB entry the instruction is using. The program executed by the processor is as follows:

Add/Sub/Not/And/Or	1 cycle INT	4 cycles Float
Load/Store	2 cycles	
Mult/Divide	1 cycles INT	4 cycles Float

TFet = IF = cycle #
TDec = De
TExe = Ex + cycle = CBD
Tiss = Issued → List reservation station and LSQ and Re order buffer
TCBD = Common Data bus time +1 to get commit
Tcom = commits
LSQ = load store queue - only applicable for load/store instructions, will label as X if not
ROB - is just a queue, so it should wait for something to fill up and fills in the spot
RS (RAT) - also a queue and waits for space to open up, F0 - F3, since it is unified, no need to label them
MOVE instruction is load/store, signlaling end
2 busses, two instructions go at a time

Instruction	TFet	TDec	TIss	RS	LSQ	ROB	TExe	TCDB	TCom	Cycle
`addi v0, a2, 0x3`	1	2	3	F0	x	0	4	5	6	1
`multi v0,v0,0x03`	1	2	3	F1	x	1	6	7	8	1
`sub sp, sp, v0`	2	3	4	F2	x	2	6	10	11	4
`addi a2, sp, 24`	2	3	4	F3	x	3	11	12	13	1
`sub sp, sp, v0`	3	4	5	F0	x	4	13	17	18	4
`addi a3, sp, 24`	3	4	7	F1	x	5	18	19	20	1
`sub sp, sp, v0`	5	6	8	F2	x	6	21	24	25	4
`addi t0, sp, 24`	5	6	10	F3	x	7	25	26	27	1
`sub sp, sp, v0`	11	13	18	F0	x	0	27	31	32	4
`lw v1, -32740(sp)`	11	16	20	F1	0	1	32	33	34	2
`addi a0, sp, 24`	15	20	25	F2	x	2	34	35	36	1
`sub sp, sp, v0`	15	22	27	F3	x	3	36	40	41	4
`lw v0, -32740(sp)`	21	22	27	F0	1	4	41	43	44	2
`sw a3, 18732(v1)`	26	27	31	F1	2	5	44	46	47	2
`lw v1, -32740(sp)`	27	28	35	F2	0	6	47	49	50	2
`sw a2, 18728(v0)`	36	37	38	F3	1	7	50	52	53	2
`lw v0, -32740(sp)`	38	39	41	F0	2	0	53	55	56	2
`sw a0, 18740(v1)`	38	40	42	F1	0	1	56	58	59	2
`lw a0, -32740(sp)`	41	42	44	F2	1	2	59	61	62	2
`sw t0, 18724(v0)`	41	42	46	F3	2	3	62	64	65	2
`lw v0, -32740(sp)`	45	47	50	F0	0	4	65	67	68	2
`lw t9, -28608(sp)`	51	52	53	F1	1	5	68	70	71	2

`addi a1, sp, 24`	51	52	56	F2	x	6	71	72	73	1
`addi a0, a0, 18804`	55	53	59	F3	x	7	73	74	75	1
`addi a2, zero, 124`	55	53	59	F0	x	0	75	76	77	1
`sw a1, 18736(v0)`	60	56	63	F1	2	1	77	79	80	2
`move a1, zero`	60	61	64	F2	0	2	80	82	83	2

Problem 2 (2 points): A processor encounters the sequence of conditional branches given below. For each branch, we list its address (the address of the branch instruction) and the direction (taken or not taken).

Complete the two cases of branch prediction methods below:

A: We are using an 4-entry predictor with 2-bit counters as entries, and all entries start off with a value of zero (strong not taken). For each branch show: which entry is used to predict it, the prediction, whether the prediction is correct, and the state of the entire 4-entry predictor (value of each entry) after the branch in question updates the predictor. The predictor entry is selected using the lowermost 2 bits of the branch address.

00 - strong not taken

01 weak not taken

10 - weak taken

11 - strong taken

Address	T/NT	Entry	Prediction	Correct?	Predictor State			
0x4652a8	NT	0	NT	Yes	0	0	0	0
0x4652b1	NT	01	NT	Yes	0	0	0	0
0x4652b3	T	11	NT	N	0	1	0	0
0x4652c1	NT	01	NT	Y	0	0	0	0
0x4652c2	NT	10	NT	Y	0	0	0	0
0x465300	T	00	NT	N	0	1	0	0
0x4652a8	NT	01	NT	Y	0	0	0	0
0x4652b1	NT	01	NT	Y	0	0	0	0
0x4652b3	NT	11	NT	Y	0	0	0	0
0x4652c1	T	01	NT	N	0	1	0	0
0x4652c2	NT	10	NT	Y	0	1	0	0
0x465300	T	00	NT	N	0	2	0	0
0x4652a8	NT	00	NT	Y	0	2	0	0
0x4652b1	T	01	T	N	0	2	1	0
0x4652b3	T	11	NT	N	0	3	1	0
0x4652c1	T	01	T	Y	0	3	1	1
0x4652c2	NT	10	T	N	0	3	1	0
0x465300	T	00	T	Y	0	3	2	0
0x4652a8	T	00	T	Y	0	3	1	0
0x4652b3	NT	11	T	N	0	3	2	0
0x4652a8	NT	00	T	N	0	3	1	0
0x4652b1	T	01	NT	N	0	0	1	0

0x4652c1	T	01	T	Y	1	3	1	0
0x4652c2	NT	10	T	N	1	3	1	0
0x465300	T	00	T	Y	0	3	2	1

B: We are using an 8-entry predictor, with a 1-bit history and two 1-bit counters in each entry (the first is for when the history bit is zero, the second is for when the history bit is one). All entries start off as all-zeroes: history is zero (not taken), and both 1-bit counters are zero (not taken). For each branch, show: the prediction, whether the prediction is correct, and the state of the entire 8-entry predictor (content of each entry: history, first counter, second counter) after the branch in question updates the predictor. Also, in the predictor state circle the 1-bit counter that was considered. The predictor entry is selected using the lowermost 3 bits of the branch address.

Address	T/NT	Pred?	Corr?	Predictor State							
0x4652a8	NT	NT	Yes	$0\textcircled{0}0$	000	000	000	000	000	000	000
0x4652b1	NT	NT	Y	100	100	100	100	100	100	100	100
0x4652b3	T	NT	N	100	100	100	101	100	100	100	100
0x4652c1	NT	T	N	000	000	000	001	000	000	000	000
0x4652c2	NT	NT	Y	100	100	100	100	101	100	100	100
0x465300	T	NT	N	101	100	100	101	100	100	100	100
0x4652a8	NT	T	N	000	000	000	001	000	000	000	000
0x4652b1	NT	NT	Y	000	000	000	000	001	000	000	000
0x4652b3	NT	NT	Y	100	100	100	100	101	100	100	100
0x4652c1	T	NT	N	100	100	101	100	101	100	100	100
0x4652c2	NT	T	N	000	001	000	001	000	000	000	000
0x465300	T	NT	N	110	101	100	101	100	100	100	100
0x4652a8	NT	T	N	010	001	000	001	000	000	000	000
0x4652b1	T	T	Y	110	101	100	101	100	100	100	100
0x4652b3	T	T	Y	010	011	000	001	000	00	000	000
0x4652c1	T	T	Y	110	111	100	101	100	100	100	100
0x4652c2	NT	T	N	110	111	100	101	100	100	100	100
0x465300	T	NT	N	111	111	100	111	100	101	111	100
0x4652a8	T	T	Y	011	011	000	000	111	100	100	100
0x4652b3	NT	T	N	111	111	100	101	100	100	100	100
0x4652a8	NT	NT	Y	110	111	100	101	100	100	100	100
0x4652b1	T	T	Y	010	011	000	001	000	000	000	000
0x4652c1	T	T	Y	110	111	100	101	001	100	100	100
0x4652c2	NT	T	N	110	111	100	101	100	100	100	100
0x465300	T	NT	N	111	111	100	100	101	100	100	100

C: Compare and contrast the two branch predictions above. Can you make any observations on the program behavior? What would you recommend for case A to get higher prediction accuracy?

Need to know if branch is taken or not(direction) and if the target address is taken(target), but ideally, we could just use a hardware or program to better predict for target and direction rather than introduce human error with 0's and 1's and miscounting. For case A to get a higher prediction accuracy, it could also follow the same structure as case B and use a 1 bit predictor and an 8 entry predictor state.