CS/ECE 752 Spring 2025: HOMEWORK # 3
(Due by **12:30 PM** on **Friday, March 7**, via upload of PDF to Canvas)

Contact Rajesh Srivatsav (rsuresh6@wisc.edu)

1. Problem 1 (20 points)

For this problem you will be implementing a 3-bit branch predictor and evaluating its performance relative to some other branch predictors. A 3-bit predictor has eight states and it is a natural extension of the 2-bit predictor (implemented as the bimodal predictor in ChampSim).

Edit ***bimodal.cc*** to support the 3-bit prediction scheme with BHT consisting of 4096 entries, direct mapped. For each entry in the prediction buffer, the state of the branch predictor entries should be set to an initial value of '**000**' (strongly not taken). The predictor is going to predict dynamic values from '**000**' through '**010**' (3 values) as not taken, and '**011**' through '**111**' (5 values) as taken.

(a) Run the 3-bit predictor with ChampSim's default configuration (*champsim.json*) using the following three benchmarks:

456.hmmer-191B.champsimtrace.xz, 433.milc-127B.champsimtrace.xz and 410.bwaves-1963B.champsimtrace.xz434

Run 50M instructions for each benchmark (after a warmup of 1M instructions as in HW2).

Similarly, run the (i) built-in 2-bit and (ii) hashed perceptron branch prediction schemes. Compare and comment on your results of the three different branch predictors in terms of IPC and the branch MPKI.

(b) Include a print-out of the changes to ***bimodal.cc*** and ***champsim.json*** with your changes highlighted.

(c) Now modify the bimodal predictor implementing a 2-bit prediction scheme **with the BHT consisting of 1024 entries** and which selects the BHT entry in the 1024-entry prediction table by EXORing bits 19-10 of the branch instruction address with the low-order bits (bits 9-0) and report the IPC and branch MPKI results for the three benchmarks above.

2. Problem 2 (20 points)
Consider the following piece of code.

```
ldf Y(r2),f2
mulf f2,f3,f0 ; <f0 = f2*f3>
ldf X(r1),f1
addf f1,f0,f2 ; <f2 = f1 + f0>
stf f2,X(r2)
addi r1,4,r1
addi r2,4,r2
slt r1,r3,r4
```

Assume that the above code is being run on MIPS R10K like out-of-order processors.

(a) Construct and fill out the following tables after 9 cycles and in the last cycle of the program's execution.

1) ROB
2) Reservation Stations
3) Map Table
4) Free List

(b) For how many cycles does this program run? That is the number of cycles until the last instruction in the program retires.

Assumptions:
1) The processor can dispatch, issue, complete and retire one instruction at a time.
2) Floating point multiplication operations take 5 cycles, floating point add takes 3 cycles, load operations take 3 cycles, store operations take 2 cycles, and integer operations take 1 cycle to execute. The execute cycles do not include other cycles of processing.
3) Functional units are fully pipelined.
4) There are only 16 physical registers and they can be used to store both floating point and integer values.
5) There are 6 ROB slots available and 5 reservation stations.

The tables' status at the end of cycle 2 is as below:

ROB:

ht	#	Inst	T	Told	S	X	C	R
h	1	ldf Y(r2),f2	PR#9	PR#2	c2			
t	2	mulf f2,f3,f0	PR#10	PR#1				
	3							

Reservation Stations:

#	FU	Busy	op	T	T1	T2
1	ALU	N				
2	L/S1	Y	ldf	PR#9	-	PR#5+
3	L/S2	N				
4	FP1	Y	mulf	PR#10	PR#9	PR#4+
5	FP2	N				

Map Table:

f0	PR#10
f1	PR#2+
f2	PR#9
f3	PR#4+
r1	PR#5+
r2	PR#6+
r3	PR#7+
r4	PR#8+

Free List:

PR#11, PR#12, PR#13, PR#14, PR#15, PR#16