EECS 470 Fall 25 HW 4

Name: _____ unique name: _____

1) Consider a local correlated predictor with 2-bit saturating counters for entries in a global pattern history table (PHT), and 7-bits of the PC as index to the branch history table (BHT). The designers of the machine are looking at the following branch sequence:

T T T N T T T T N N T T T N T T T T N N T T T N.....

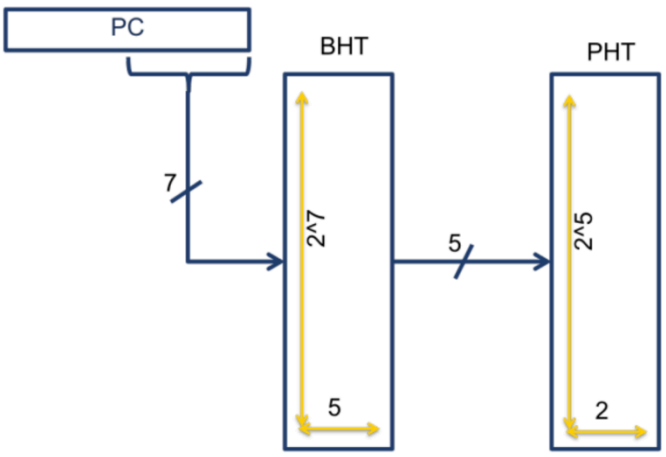
a) How many bits must each branch history register contain to achieve 100% accuracy in the steady state operation?

Ans: 5-Bits
There are 10 branches before it repeats, so look how each would be predicted.
4-bits would have an alias on TTTN, but 5 bits does not have an alias.

The patterns for 4-bits are:
TTTN -> T (alias)
TTNT -> T
TNTT -> T
NTTT -> T
TTTT -> N
TTTN -> N (alias)
TTNN -> T
TNNT -> T
NNTT -> T
NTTT -> N
...

With 5 bits, the patterns are:
T T T N T -> T
T T N T T -> T
T N T T T -> T
N T T T T -> N
T T T T N -> N
T T T N N -> T
T T N N T -> T
T N N T T -> T
N N T T T -> N
N T T T N -> T
...

b) Draw a diagram of the system.



2) Consider a new branch sequence:

T T N T T N T T N T T N T T N.....

In a similar system with BHT entries that are 4 bits long. If the machine starts with all BHT entries as Not Taken (i.e. NNNN) and all predictors set as Strongly Not Taken answer the following:

	Cycle															
PHT Row	**Warmup**								**Steady State**							
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
NNNN	N*	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
NNNT	N	N*	n	n	n	n	n	n	n	n	n	n	n	n	n	n
NNTN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
NNTT	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
NTNN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
NTNT	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
NTTN	N	N	N	N*	n	n	n*	t	t	t*	T	T	T	T	T	T
NTTT	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
TNNN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
TNNT	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
TNTN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
TNTT	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
TTNN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
TTNT	N	N	N	N	N*	n	n	n*	t	t	t*	T	T	T	T	T
TTTN	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
TTTT	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Outcome	T	T	N	T	T	N	T	T	N	T	T	N	T	T	N	T

Key:

Accessed in warmup	Accessed in steady state	PHT entry read

* : mispredicted; PHT entry updated after misprediction

a) How many branches does it take for the predictor to reach the steady state (when is the last mis-predicted branch)?

Ans: 8 Branches to reach the steady state

b) How many pattern history entries does the branch touch during steady-state operation?

Ans: 3 (NTTN, TTNT, TNTT) (green patterns)

c) How many, if any, additional PHT entries does the predictor touch before steady-state operation is achieved (warmup)?

Ans: 3 (NNNN, NNNT, NNTT) (orange patterns)

3) The 0x1eaf consulting firm has been hired to evaluate the branch predictor design in the newest Leg Cortex-G7 processor. Being recent graduates of EECS 470, they recognize the current design is a PAg style predictor. They quickly analyze the benchmarks for the customer and recognize that a GAp style predictor can achieve a 4% better accuracy. When they bring the design to the chief architect, she says that there is no additional silicon real-estate. Given the following parameters, determine if the new GAp design is larger or smaller than the original PAg design.

If needed, both designs use:
16-Bits of the PC
10-Bits of history
2-Bit Saturating Counters

a) The area for the Original PAg predictor (in bits) is?

BHT size + PHT size = (2^16 * 10) + (2^10 * 2) = 657,408 bits

b) The area for the new GAp predictor (in bits) is?

BHT size + PHT size = (10) + (2^10 * 2^16 * 2) = 134,217,738 bits

c) Which should the designers choose, PAg or GAp?

PAg since it is 200x smaller

d) The most astute of the 0x1eaf group remembers the GSHARE predictor from EECS 470. A quick analysis shows that by changing the parameters, the GSHARE can outperform both the GAp and PAg designs on the workloads. How large would the GSHARE predictor be given the following parameters?

18-bits of PC
18-bits of History
2-bit predictors

GHR size + PHT size = 18 + (2^18 * 2) = 524,306 bits

e) Finally, The 0x1eaf team is now asked to evaluate a new set of workloads for the design they chose, there are several benchmarks with a nested for-loop, but each benchmark has different sized inner loops (and a really large outer loop count). What is the largest size inner for-loop that each design can predict correctly in the steady-state (after it is

warmed-up). You can assume that no other branch aliases to the same location for the PAg design

i) PAg

Loop size of 11
Interesting scenario that the outer loop eventually gets to a TTTTTTTTTT->T, while the inner loop wants TTTTTTTTTT->N, so one might consider a loop of one smaller. However, because it is a 2-bit saturating counter, the inner loop will move the state to N before the outer loop gets enough history to see it. After that the outer loop will move the N->n, but the inner-loop will still get the correct result and move it back n->N. So the inner loop can reach a perfect prediction in the steady state at the full history length, while the outer loop always mispredicts. For one shorter history, then both loops can have perfect prediction. However, the question only asks about inner loop.

ii) Gap

Loop size of 10, the outer for-loop inserts a T into the history pattern because it is global.

iii) GSHARE

Loop size of 18, again the outer for-loop inserts a T into the history pattern.
This one is tricky, as the XOR could cause aliasing of the inner and outer loop into the PHT. If we assume there is no aliasing, 17 is the correct answer, but it might be smaller if aliasing is assumed.

COMMON ERROR

Note a for loop of size 11 has the following pattern TTTTTTTTTTN (10 T and 1 NT). It is common for students to call this a loop of size 10, but it is actually a loop of size 11. This results in some students getting an off-by-one error in the prior problem and answers of 10, 9, 17.

4) [TEAM PROBLEM] Your entire team should each turn in the same answer to this question. Write a "long" (thousands of instructions executed, it may be shortish in terms of instructions in the program) RISC-V test vector (test program) that would help you figure out:

a) If your X-way superscalar processor meets the goal of getting an IPC greater than X-1.
b) If your processor meets the expected IPC when executing a chain of dependent instructions.
c) If your load-store queue can correctly handle back-to-back stores and loads to the same address.

In each case provide not only the test vector, but explain how looking at performance counters (e.g., instructions retired, branch mispredicts, stall counters, etc) would help you know if your design was meeting the stated goal.

Looking for a relatively tight loop so I-cache can hit on all the instructions. Too small of a loop would create branch issues. If loop is more than 64 instructions, it won't fit in the I-cache

a) No dependencies between instructions within the loop.
b) Dependencies between subsequent instructions.
c) Consecutive loads and stores to the same address