CDA5155/CDA4150 - Spring 2026
Assignment 4
Branch Target Buffer and Branch Prediction Simulation

Your assignment is to write a program that simulates a branch target buffer and a branch predictor. First, you will read in a configuration file, which will be called *branch.config* in the current directory. Next, you will read in a trace of pairs of addresses, where the first is a branch address and the second is the address of the next instruction to execute. For each of these branches you will determine if the branch is in the buffer or not and a prediction for the branch. You will print out the reference number, current address of the branch, address of the instruction executed after the branch, the index of the entry in the buffer, the tag of the address, an indication if the branch was in the buffer, a prediction if the branch was taken or not, the predicted next address, and an indication if the prediction was correct or incorrect. Finally, you will print some summary statistics.

The configuration file contains three lines. The first line will specify the number of branch prediction entries. Remember that the lower portion of the address of each branch is used as an index into the buffer. The size of this index is determined by the number of branch prediction entries. Note that all branches are aligned on a four byte boundary, so the two least significant bits are ignored. You may assume that the number of entries will be a power of two and that it will not exceed 1024. The second line will specify if a tournament predictor is to be used ('Y' or 'N'). The third line will specify an (m, n) predictor, where 2^m predictors will be used for each branch prediction buffer entry (m is the number of the most recent branches encountered to be used for correlation) and n is the number of bits to be used for each predictor. The allowable values for m will be 0 to 4. The only valid values for n will be 1 or 2. A tournament predictor requires that m be between 1 and 4. When a tournament predictor is specified, then both a local (0, n) and global (m, n) predictor are used to determine if the branch is taken or not. The initial state for the 1-bit predictor will be predict-not-taken. The initial state for the 2-bit predictor will be weakly predict-not-taken (bottom left state of the figure in the 2-bit Branch Prediction Buffer slide). The initial state for the tournament predictor will be the weakly use predictor 1 (bottom left state of the Tournament Branch Prediction Buffers slide). Note that for this assignment predictor 1 represents the local predictor and predictor 2 represents the global predictor. The initial result for the last m branches will be not-taken. An example configuration file is available in *~whalley/cda5155exec/branch.config*.

The trace of current addresses and next addresses will be read from standard input. Each line in the trace of references will have the following format:

<hexaddress>:<hexaddress>

where the first address is for the branch and the second address is for the instruction executed after the branch. An example trace file is available in *~whalley/cda5155exec/trace.dat*.

As each branch address is read, you will print information in the following format, which is given in the form of a C printf format specification:

```
"%4d %9x %9x %5d %7d %6s %10s %9x %7s"
```

The summary statistics will be printed after the entire trace has been read. The statistics will indicate the number of hits and misses in the buffer, the miss ratio for the buffer, the number of correct and incorrect predictions, and the misprediction ratio.

The file *~whalley/cda5155exec/trace.stats* contains the output of running my branch prediction buffer using the *~whalley/cda5155exec/branch.config* as the configuration file and *~whalley/cda5155exec/trace.dat* as the trace data. You can check your execution with mine using the executable *~whalley/cda5155exec/btbbpb.exe*. Your output should match my format exactly so we can check for correctness using the *diff* command. If you invoke my executable with the verbose option:

~whalley/cda5155exec/btbbpb.exe -v

then you will get diagnostic output printed to *stderr* that you can use to resolve errors with your simulator.

You should upload the source code for your program in Canvas before class on March 12. The header comments should identify you, the assignment, and the command you used to compile your program. You should also

use readable and consistent indentation and comments throughout the program. Your program should be able to compile and execute on *linprog*. Below are example *configuration* and *trace* files and the corresponding output.

configuration

```
Number of entries: 8
Tournament: N
Branch predictor: (1,1)
```

trace data

```
10:1010
24:  28
38:1038
 4:1004
 4:1004
 4:1004
 4:   8
 4:1004
28:1028
24:1024
```

output

```
Number of entries: 8
Number of index bits: 3
Tournament: No
Branch predictor: (1,1)

 ref curr addr next addr entry   tag   buffer prediction pred addr correct
---- --------- --------- ----- ------- ------ ---------- --------- -------
   1        10      1010     4       0   miss  not taken        14      no
   2        24        28     1       1   miss  not taken        28     yes
   3        38      1038     6       1   miss  not taken        3c      no
   4         4      1004     1       0   miss  not taken         8      no
   5         4      1004     1       0    hit      taken      1004     yes
   6         4      1004     1       0    hit      taken      1004     yes
   7         4         8     1       0    hit      taken      1004      no
   8         4      1004     1       0    hit  not taken         8      no
   9        28      1028     2       1   miss  not taken        2c      no
  10        24      1024     1       1   miss  not taken        28      no

buffer hits           : 4
buffer misses         : 6
buffer miss ratio     : 0.600000

correct predictions   : 3
incorrect predictions : 7
misprediction ratio   : 0.700000
```

Assignment 4 Suggestions

I recommend that you accomplish assignment 4 in stages. This means that you can implement a portion of the assignment and test it before proceeding to the next portion. Our test cases will likely also follow these stages. For each step compare your output with the output generated by my executable using the Unix *diff* command to ensure your output exactly matches mine. CDA4150 students need not implement the Tournament predictor, but will receive extra credit if they do so.

(1) Read in the configuration and trace data and print the configuration, the headers for the output, and for each line of trace data print the reference number, the current address, and the next address.

(2) Make the configuration indicate that a Tournament predictor is not used. Create a data structure to contain a valid bit and tag for each entry. Test with a set of current and next addresses. Indicate for each reference the *entry* index, the *tag* value, and whether there is *miss* or *hit* in the branch target buffer.

(3) Make the configuration indicate that the Branch predictor is (0,1). Extend your data structure to hold a 1-bit predictor for each entry. Be sure to initialize each predictor entry to the appropriate state. Test with a set of current and next addresses. Indicate for each reference whether the predication was *not taken* or *taken*, the predicted address, and whether the prediction was correct (*yes*) or incorrect (*no*).

(4) Increment counters regarding buffer accesses and predictions. Print the summary statistics after the information for each reference.

(5) Make the configuration indicate that the Branch predictor is (0,2). Test with a set of current and next addresses for the 2-bit predictor.

(6) Make the configuration indicate that the Branch predictor is (*m*,*n*), where *m* has a value ranging from 1 to 4 and *n* has a value ranging from 1 to 2. Test for different configurations of the correlating predictor.

(7) Make the configuration indicate that the Tournament predictor is used. Test for different configurations of the Tournament predictor, where the value of *m* is not 0.