Tanish Arora
CS 203

Project 2 Questions

1. In a 2-bit branch predictor with 8 bits of PC to index. The number of entries in the BHT will have 2^8 which is really 256 entries. The size of the table has 256 entries with 2 prediction bits (to represent the four states 00, 01, 10, and 11). This means it will have 512 bits or 512/8 = 64 bytes.

2. In a (4,2) predictor, this means it is a 2-bit branch predictor (n = 2) with 2^4 different 2 prediction bits in there (m = 4). Assuming still that there are 8 bits for the PC to index, the BHT will have 2^8 which is really 256 entries. The size of the table has 256 entries with 2^4 number of 2 prediction bits. This means it will have $2^8 \times 2^4 \times 2 = 2^{13}$ bits or 8,192 bits or 8,192/8 = 1,024 bytes.

3. The misprediction counts are shown in the table below:

Trace	(0, 1)	(0, 2)	(6, 1)	(6, 2)
gcc-10K.txt	3,771	2,534	2,029	2,615
gcc-8M.txt	3,169,685	2,140,658	731,106	622,058

   The misprediction rates (10,000 in the 10k file and 8,515,352 in the 8M file) are shown in the table below:

Trace	(0, 1)	(0, 2)	(6, 1)	(6, 2)
gcc-10K.txt	37.71%	25.34%	20.29%	26.15%
gcc-8M.txt	37.22%	25.14%	8.59%	7.31%

4. No, every entry is not utilized. In the (6,1)-predictor, only 7,966 entries were used.

5. In gcc-10K.txt, the global history improves the misprediction rate by a factor of 3771 / 2029 = 1.859x for 1-bit predictors and it worsens the misprediction rate by a factor of 2615 / 2534 = 1.032x for 2-bit predictors. In gcc-8M.txt, the global history improves the misprediction rate by a factor of 3169685 / 731106 = 4.335x for 1-bit predictors and it improves the misprediction rate by a factor of 2140658 / 622058 = 3.441x for 2-bit predictors.

   This set of data will benefit from a global branch history, however in certain special cases it will not. For instance, if the (0,1)-branching is always occurring or not occurring (for example, actual branches are TTTTTTT…, always Taken) then it will mis-predict on the first and then predict correctly every time after. But in a (6,1)-branching it will also mis-predict on the first and then predict correctly every time after. Suppose there are 100 of these predictions, then both branching predictors will have a 1% misprediction rate (1 miss out of 100 predictions) which is not an improvement. Therefore, it is better to say instead that all situations will be at least as good as standard predictors for global branch history predictors.

6. The local branch predictor is similar to the global branch predictor because it utilizes a memory of what previous m branches did but it remembers what prediction that particular address has. It is, in a sense, localized to that address. It helps prediction because it associates the prediction with the address in that global branching does not. It can be combined with (m,n)-predictors to give m

number of blocks in the branch table, n-bits of states, and a localized branch predictor to predict address branches.

How To Run:

To run this program, do the following commands:

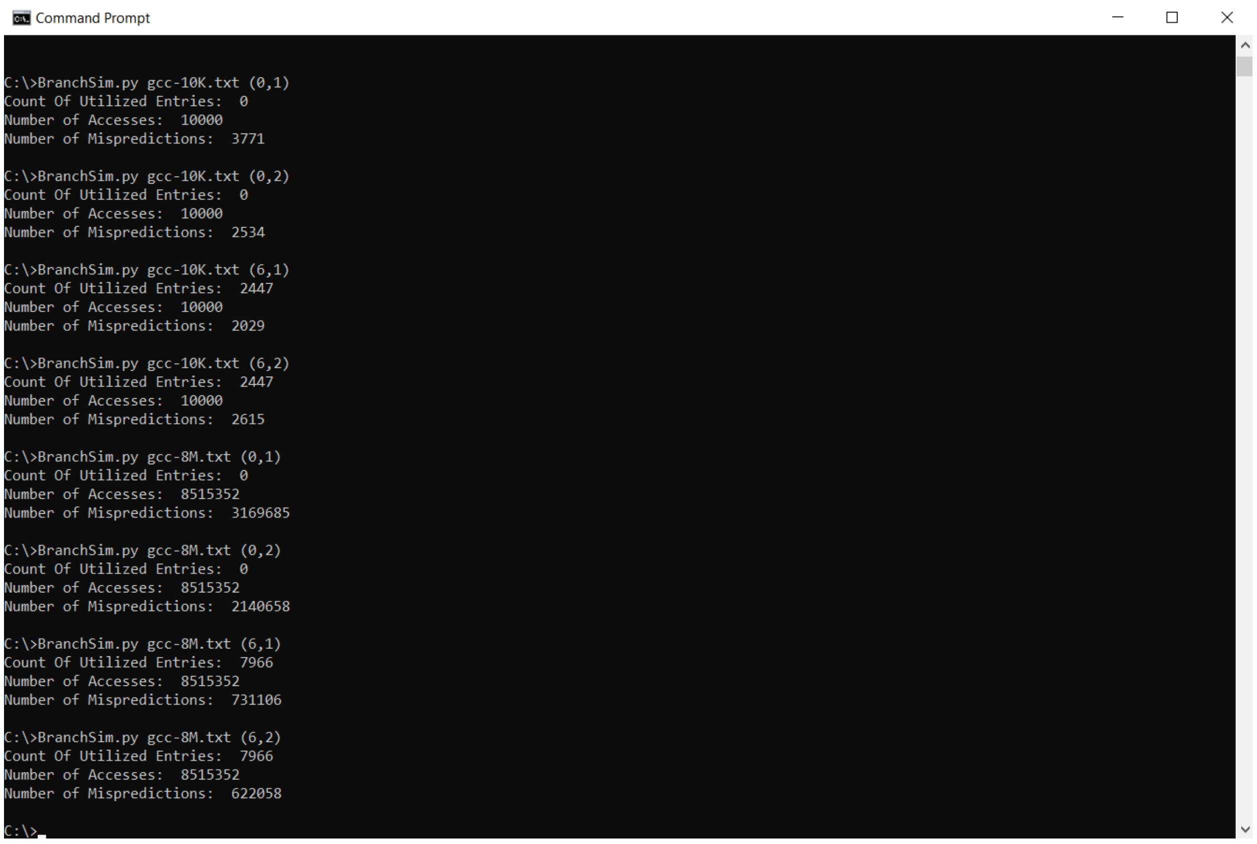
1. Python 3 must be installed on your computer (any version, preferably 3.6 or higher).
2. Place the Branchsim.py file and the .txt files together in one folder
3. Use Command Prompt (Windows) or Terminal (Linux/Ubuntu) to navigate to this folder with the files.
4. Call upon python3 or simply run the PY file with the following form (no space in between m and n:

   Branchsim.py <file name> (<m>,<n>)

5. <filename> is the text file to run, <m> is the number of bits used as column entries in the global history table, and <n> is the number of n-bits that the predictor uses. One possible call to this program is:

   Branchsim.py gcc-10K.txt (0,1)

6. Error messages will appear if the wrong arguments are given or the file cannot be read in.
7. Below is screenshot run of this program.



```
C:\>BranchSim.py gcc-10K.txt (0,1)
Count Of Utilized Entries:  0
Number of Accesses:  10000
Number of Mispredictions:  3771

C:\>BranchSim.py gcc-10K.txt (0,2)
Count Of Utilized Entries:  0
Number of Accesses:  10000
Number of Mispredictions:  2534

C:\>BranchSim.py gcc-10K.txt (6,1)
Count Of Utilized Entries:  2447
Number of Accesses:  10000
Number of Mispredictions:  2029

C:\>BranchSim.py gcc-10K.txt (6,2)
Count Of Utilized Entries:  2447
Number of Accesses:  10000
Number of Mispredictions:  2615

C:\>BranchSim.py gcc-8M.txt (0,1)
Count Of Utilized Entries:  0
Number of Accesses:  8515352
Number of Mispredictions:  3169685

C:\>BranchSim.py gcc-8M.txt (0,2)
Count Of Utilized Entries:  0
Number of Accesses:  8515352
Number of Mispredictions:  2140658

C:\>BranchSim.py gcc-8M.txt (6,1)
Count Of Utilized Entries:  7966
Number of Accesses:  8515352
Number of Mispredictions:  731106

C:\>BranchSim.py gcc-8M.txt (6,2)
Count Of Utilized Entries:  7966
Number of Accesses:  8515352
Number of Mispredictions:  622058

C:\>
```