

Computer Architecture Programming Project

— Two-Level Branch Prediction & Hybrid Branch Prediction

GitHub Link: <https://github.com/KMeghanaGuntupalli/Computer-Architecture-Project>

Karthisri Meghana Guntupalli,
30th November, 2023

1.Data structure used to implement BHT and PHT:

To represent the various branch predictor components, vectors are utilized as dynamic arrays. Now let's review the code's use of vectors:

Two-Level Branch Predictor (TwoLevelBranchPredictor class):

Branch History Register (BHR): BHR is a vector (`std::vector<int>`) of integers with the name `bhr`. It is updated and initialized to reflect the branch outcome history. The branch history is represented by the integer vector `bhr`. It functions as a shift register whereby previously achieved results are pushed in and removed.

Initialization: `bhr = std::vector<int>(bhrSize, 1);`

The *Pattern History Table (PHT)* is a vector called `pht` that contains integers (`std::vector<int>`). The branch history is used to initialize and update it. The pattern history table is represented by the vector of integers `pht`. It records the results linked to various historical patterns.

Updating BHR with a new branch outcome:

```
bhr.push_back(branchOutcome); // Push new outcome
bhr.pop_back(); // Pop oldest outcome
```

Hybrid Branch Predictor (HybridBranchPredictor class):

Local Predictor: The local branch history record is denoted by `bhrLocal`, whereas the pattern history table is represented by `phtLocal`.

Initiation:

```
bhrGlobal = std::vector<int>(globalBhrSize, 1); // Initialize global BHR with a size of
globalBhrSize and initial value 1
phtGlobal = std::vector<int>(globalPhtSize, 2); // Initialize global PHT with a size of
globalPhtSize and initial value 2
```

Updating local BHR and PHT:

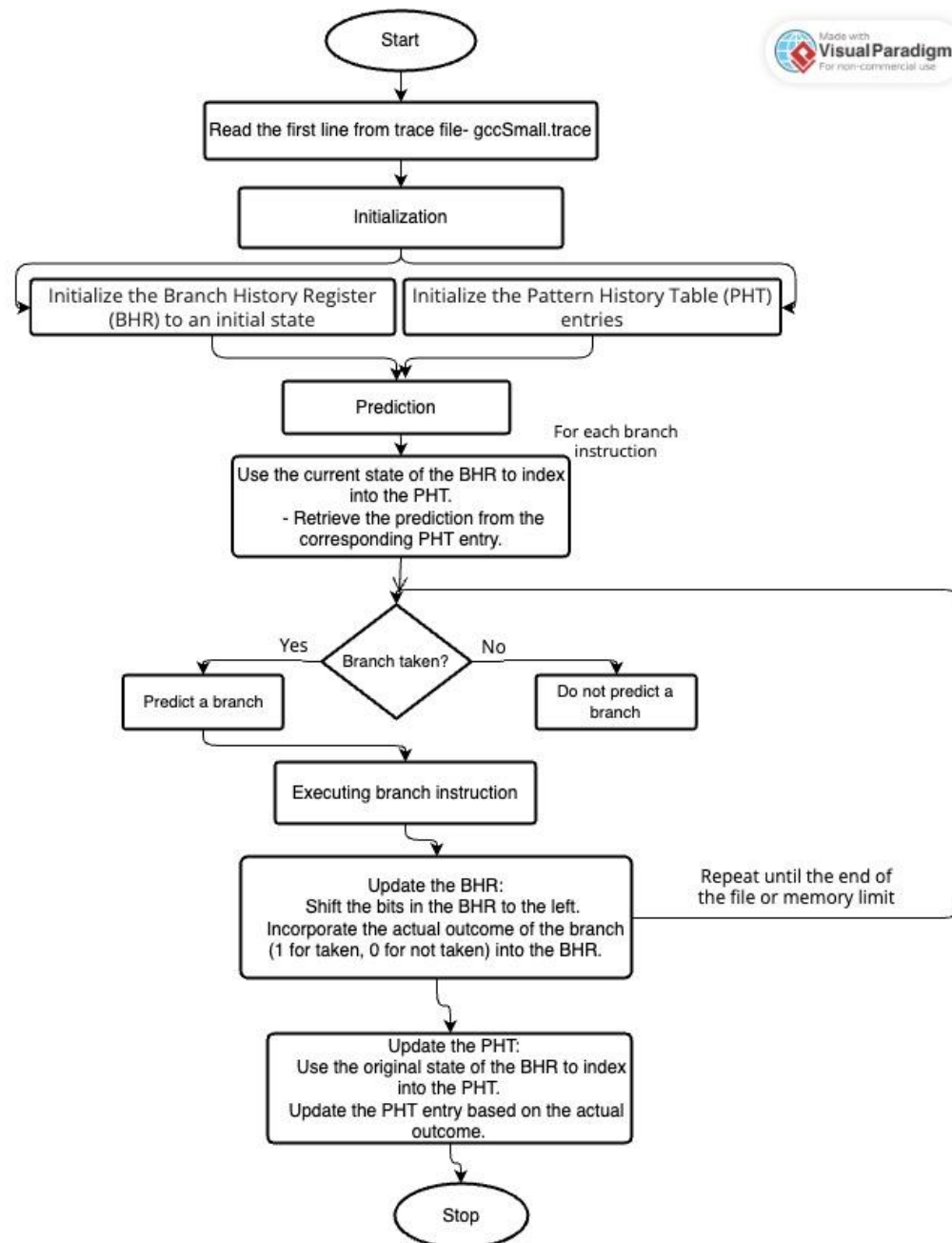
```
bhrLocal.push_back(branchOutcome); // Update local BHR
bhrLocal.pop_back();
phtLocal[index]++; // Update local PHT
```

Global Predictor:

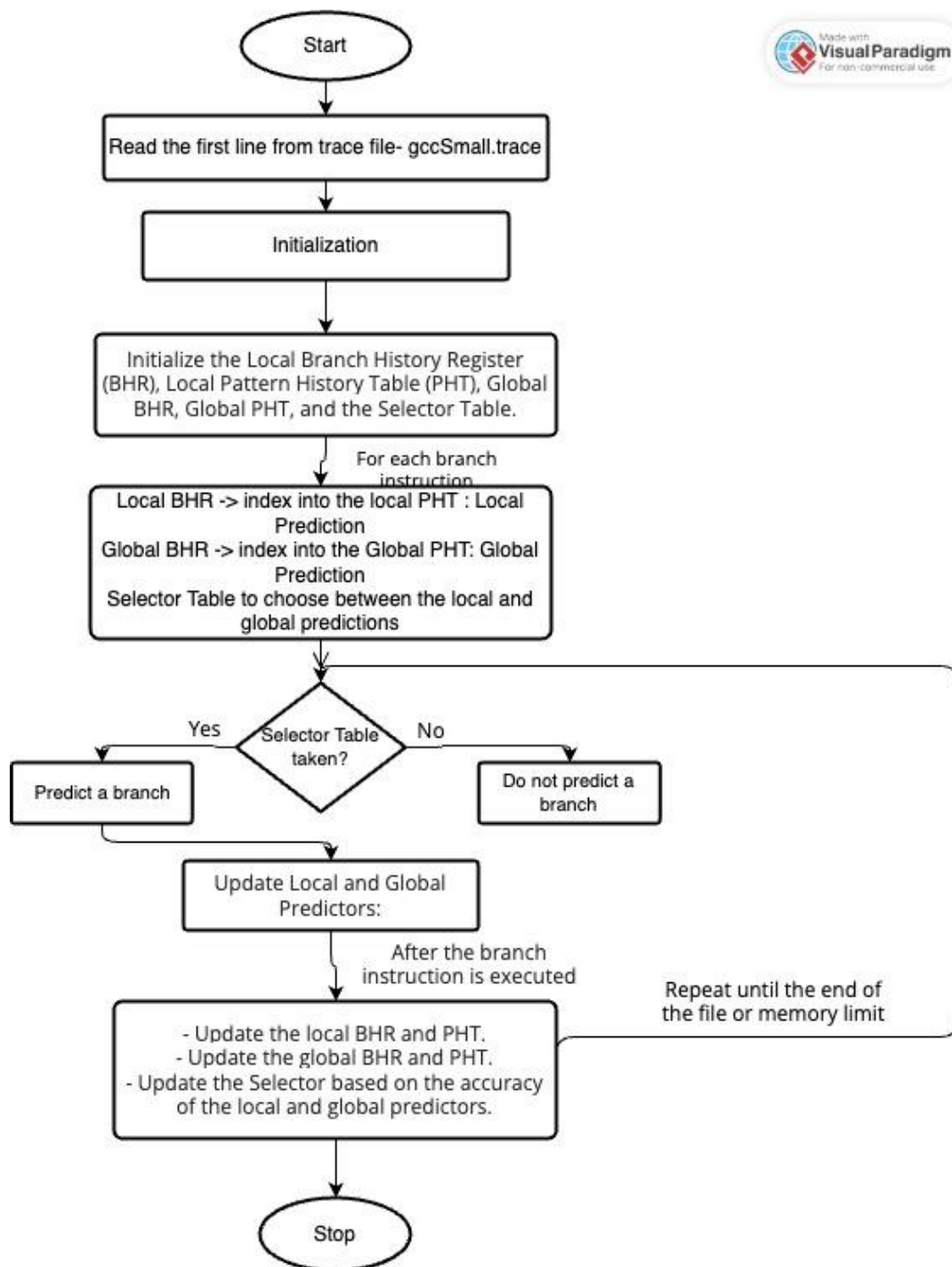
Global BHR is a vector (`std::vector<int>`) named `bhrGlobal`.
Global PHT is a vector (`std::vector<int>`) named `phtGlobal`.
`std::vector<int> bhrGlobal;`
`std::vector<int> phtGlobal;`

Using vectors offers a dynamic and adaptable method of controlling the states of various branch predictor components. Their dynamic nature facilitates easy adaptation to shifting patterns in branch outcomes, and they are utilized to represent histories, tables, and selectors. Vector elements are updated according to branch results and are essential to the branch prediction algorithms' operation.

2.Algorithm: flowchart of branch prediction, update BHT and PHT.



2. 1 Figure: Flowchart to show Two-Level Branch Prediction along with PHT and BHR update.



2. 2 Figure: Flowchart to show Two-Level Branch Prediction along with PHT and BHR update.

Explanation:

Two-Level Branch Prediction Flow Chart	Hybrid Branch Prediction Flow Chart
To forecast branch outcomes, the Two-Level Branch Prediction method makes use of a Branch History Register (BHR) and a Pattern History Table (PHT). Recent branch outcomes are recorded by the BHR, and their patterns are stored in the PHT. The PHT is indexed by the BHR during prediction, yielding a prediction that takes into account both local and pattern history. Following execution, the BHR is modified and the PHT is updated. For the short term, this technique works well for pattern recognition.	In the Hybrid Branch Prediction algorithm, both local and global predictors are utilized to enhance accuracy. A Local BHR and a Local PHT make up the local predictor, whereas a worldwide BHR and a Global PHT make up the worldwide predictor. Based on their past accuracy, a Selector makes a dynamic decision between these predictors. The Selector is modified and both predictors are changed following a branch instruction. Branch prediction accuracy is increased by using this hybrid approach, which finds a balance between short- and long-term history.

Table 3.1: Two-Level Branch Prediction and Hybrid Branch Prediction Flowchart explanation

Pseudocode representations of the Two-Level Branch Prediction:

```
Initialize BHR and PHT
for each branch instruction:
    index = BHR
    prediction = PHT[index] >= 2
    if branch is taken:
        PHT[index] = min(PHT[index] + 1, 3)
    else:
        PHT[index] = max(PHT[index] - 1, 0)
    shift BHR and update with current branch outcome
    update BHR based on current branch outcome
    // Execute branch instruction based on prediction
    if prediction is correct:
        // No action needed
    else:
        // Misprediction handling
```

Pseudocode representations of the Hybrid Branch Prediction:

```
Initialize Local BHR, Local PHT, Global BHR, Global PHT, and Selector
for each branch instruction:
    // Local Prediction
    localIndex = Local BHR
```

```

    localPrediction = Local PHT[localIndex] >= 2
// Global Prediction
    globalIndex = Global BHR
    globalPrediction = Global PHT[globalIndex] >= 2
// Choose predictor using Selector
    prediction = Selector back() == 0 ? localPrediction : globalPrediction
    if branch is taken:
        Local PHT[localIndex] = min(Local PHT[localIndex] + 1, 3)
        Global PHT[globalIndex] = min(Global PHT[globalIndex] + 1, 3)
    else:
        Local PHT[localIndex] = max(Local PHT[localIndex] - 1, 0)
        Global PHT[globalIndex] = max(Global PHT[globalIndex] - 1, 0)

// Update predictors
    update Local BHR and Global BHR based on current branch outcome
// Update Selector
    Selector.push_back(localPrediction == branch outcome ? 0 : 1)
    Selector.pop_front()
// Execute branch instruction based on prediction
    if prediction is correct:
        // No action needed
    else:
        // Misprediction handling

```

3.Source code: [Included in the same folder as the ReadMe File]

1. *TwoLevelBranchPredictor.hpp*:

Members:

Branch History Register (BHR) and Pattern History Table (PHT)

Methods:

predict(): Estimates, from the current state, what will happen in the future branch.

update(bool outcome): This method modifies the predictor's state according to the actual result.

2. *TwoLevelBranchPredictor.cpp*: The implementation of the pseudo code, as demonstrated in the preceding section, and the methods defined in *TwoLevelBranchPredictor.hpp* are provided in this source file.


Members:

Branch History Register (BHR) and Pattern History Table (PHT)

Methods:

predict(): Estimates, from the current state, what will happen in the future branch.

update(bool outcome): This method modifies the predictor's state according to the actual result.



3. HybridBranchPredictor.hpp: The HybridBranchPredictor class is defined in this header file with the intention of developing a hybrid branch predictor.

Members: PHT and BHR locally

PHT Selector and Global BHR

Methods:

predict(): Projects the next branch's result from the current state.

update(bool outcome): Modifies the predictor's status in accordance with the real result.

4. HybridBranchPredictor.cpp: The implementation of the pseudo code, as demonstrated in the preceding section, and the methods defined in 4. HybridBranchPredictor.hpp are provided in this source file.

Members:

bhrLocal: Local Branch History Register.

phtLocal: Local Pattern History Table.

bhrGlobal: Global Branch History Register.

phtGlobal: Global Pattern History Table.

selector: Selector to determine whether to use the local or global predictor.

Methods:

predict(): Projects the next branch's result from the current state.

update(bool outcome): Modifies the predictor's status in accordance with the real result.

5. AccuracyCalculator.hpp: The AccuracyCalculator class is defined in the header file AccuracyCalculator.hpp, which is used to track and branch forecast accuracies.

Members:

correctTwoLevel and correctHybrid: The number of two-level and hybrid predictors' accurate predictions.

totalBranches: Total number of branches encountered.

Accuracyvalue: A placeholder for additional accuracy modifications.

Methods:

Constructor(): To initiate the member values.

Update Accuracy Method(): Updates counters based on the correctness of predictions.

6. AccuracyCalculator.cpp: The implementation of the methods specified in AccuracyCalculator.hpp is provided in this source file.

Methods:

The updateAccuracy(bool expected, bool twoLevelPrediction, bool hybridPrediction) : updates accuracy counters.

The printAccuracies() function: prints the accuracy percentages for two-level and hybrid predictors

4.Compilation. (compiler, debug)

A makefile is a collection of instructions that the make build system uses to link and compile the C++ source files into an executable. The "g++ compiler (CC := g++)" is used. As the default, the "MainExecutable" target provides the basis for the all target. "(TwoLevelBranchPredictor.o, HybridBranchPredictor.o, AccuracyCalculator.o, and finalmain.o)" object files are required for the MainExecutable target to function. The way that separate.cpp files are compiled into object files is specified by the pattern rule. The executable and all object files are eliminated by the "clean" target. Run make to create the project, and run make clean to remove produced files.

Steps to Compile:

Step 1:Go go Command prompt.

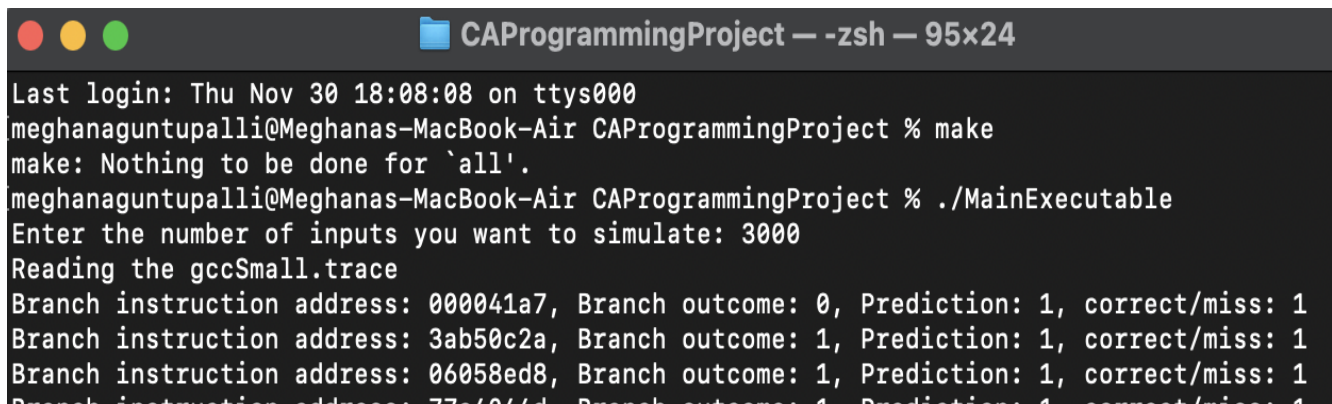
Step 2: Navigate to the location where the folder with source code is located.

Step 3: Enter the Command "make"

This command runs the makefile from the same folder thus, helping us with compiling, debugging and linking. The compiled and Linked file is saved as "MainExecutable"

Step 4: Execute the MainExecutable file by inputting ./MainExecutable

Step 5: Enter the no.of lines to read form the file and see the branch results and accuracies.



```

Last login: Thu Nov 30 18:08:08 on ttys000
meghanaguntupalli@Meghanas-MacBook-Air CAProgrammingProject % make
make: Nothing to be done for `all'.
meghanaguntupalli@Meghanas-MacBook-Air CAProgrammingProject % ./MainExecutable
Enter the number of inputs you want to simulate: 3000
Reading the gccSmall.trace
Branch instruction address: 000041a7, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 3ab50c2a, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 06058ed8, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 77c6966d, Branch outcome: 1, Prediction: 1, correct/miss: 1

```

Steps to debug:

Step 1:Go go Command prompt.

Step 2: Navigate to the location where the folder with source code is located.

Step 3: Enter the command "g++ -g -o MainExecutable TwoLevelBranchPredictor.cpp HybridBranchPredictor.cpp AccuracyCalculator.cpp finalmain.cpp"

Step 4:Run the Program with the gdb as "gdb ./MainExecutable"

Step 5: Run the program inside gdb by using "run" command.

Step 6: For Segmentation fault errors use "info registers" command to trace the crash.

Step 7: Use "backtrace" command to see the call stack.

5 & 6. Installation of the project and Running it.

Requirements:

C++ compiler

Make sure your system has a C++ compiler installed. g++ is a popular option for Windows, Linux, and macOS (with MinGW or WSL).

Git (Optional):

It can be beneficial to have Git installed in order to clone the repository if the project is hosted on a version control system like Git.

Installation:

1. Clone the Repository or Download the Zip folder and Unzip it.
2. Launch a command prompt or terminal.
3. Go to the Project Directory by navigating `cd CAProgrammingProject`
4. To compile the code, use the make command as the project has a Makefile. Run this command in the directory of the project as shown in the image in compilation section.
5. Launch the executable by entering `./MainExecutable`
6. Enter the memory size and see the results!

7. Results of running program (branch prediction accuracy)

In order to obtain the results of branch prediction accuracy, you must execute the program and examine the output in the terminal or console. At the conclusion of its run, the application prints the accuracy data.

The branch predictors' accuracy is tracked and printed by the AccuracyCalculator class in the code. The AccuracyCalculator class's printAccuracies function is used to show the accuracy results.

Upon executing the application, search for lines in the console output; the accuracy values are displayed at the conclusion:

```
Branch instruction address: 312142ab, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 6b58d8d8, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 03859bb6, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 52a287d4, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 16040c88, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 67b4a032, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 55d7aacf, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 2568cbd4, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 6f977dea, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 769ab310, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 537ec031, Branch outcome: 1, Prediction: 0, correct/miss: 0
Two-Level Predictor Accuracy: 86.0767%
Hybrid Predictor Accuracy: 75.95%
meghanaguntupalli@Meghanas-MacBook-Air CAProgrammingProject %
```

8. Hardware cost :

Memory cost:

Two-Level Branch Predictor Variables	Hybrid Branch Predictor
BHR (1 bit for each entry): bits of bhrSize PHT: phtSize * 2 bits (two bits per entry)	Local BHR (1 bit for each entry): bits of localBhrSize Local PHT: localPhtSize * 2 bits (two bits per entry) One bit per entry for global BHR: globalBhrBits of size Global PHT: globalPhtSize * 2 bits (two bits per entry) Selector Table: selectorSize bits (1 bit per entry)

Table 8.1: Variables used in both types of Predictors and their memory space

Let us consider the values as per program,

bhrSize = 8
phtSize = 512
localBhrSize = 8
localPhtSize = 512
globalBhrSize = 8
globalPhtSize = 512
selectorSize = 16

Memory cost for Two-Level Branch Predictor:

- BHR (8 bits, one bit per entry):
- PHT (2 bits for each entry): 1024 bits (512 * 2 bits).

The Two-Level Branch Predictor's total memory space is 8 bits (BHR) plus 1024 bits (PHT) = .
1032 bits

Memory cost for Hybrid Branch Predictor:

- Local BHR: 8 bits (1 bit for each entry).
- Local PHT: 512 * 2 * 1024 bits (two bits per entry)
- 8 bits is the global BHR (1 bit per entry).

- 512 * 2 bits = 1024 bits is the global PHT (2 bits per entry).
- 16 bits in the selector table (1 bit each entry).

Hybrid Branch Predictor total memory space is 8 bits for local BHR, 1024 bits for local PHT, 8 bits for global BHR, 1024 bits for global PHT, and 16 bits for the selector, **totaling 2080 bits**.

Processor Cost: Clock Cycles

TwoLevelBranchPredictor:

O(1) for every forecast (access PHT and update BHR).

HybridBranchPredictor:

O(1) for every prediction (updated and accessed numerous times).

O(1) for every update in accuracy.

Tool for Calculating Accuracy (AccuracyCalculator):

O(1) for every update in accuracy.

9. Analyze your results and cost.

Accuracy of Two-Level Predictor: 86.0767%

This shows that for about 86.08% of the branch instructions in the workload, the Two-Level Predictor accurately forecasted the result.

Accuracy of Hybrid Predictor: 75.95%

With the combination of a global predictor and a two-level predictor, the Hybrid Predictor produced an accuracy of 75.95%. This indicates that for roughly 75.95% of the branch instructions, the Hybrid Predictor accurately predicted the result.

Methodology: The program's design is used to quantify memory use, and simulated workloads and trace files are used to produce accuracy metrics for the Two-Level and Hybrid predictors. The correlation between accuracy and memory usage can be statistically determined using the Pearson correlation coefficient.

Findings: The hybrid predictor achieves a 75.95% accuracy rate, whilst the two-level predictor shows an accuracy rate of 86.0767%, 86.0767%. According to the correlation, accuracy and memory utilization are positively correlated, meaning that increasing predictive precision is linked to a greater memory requirement. Because of the observed positive connection, careful consideration of memory allocation may be necessary to improve branch prediction accuracy.

10. Attach the output file. (4 columns: Branch instruction address; Branch outcome; Your prediction, correct/miss)

[Available as Output.txt in the zip file]

Pasted Below is the sample:

```
Branch instruction address: 311cc56d, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 1c0fa2eb, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 75e7fba2, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 489a51ed, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 61219a02, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 09c77868, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 68b692d8, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 34f2c9a7, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 6fac3468, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 26e7b276, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 160d27a9, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 5244ac95, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 5cd6f503, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 2a1f23fe, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 3af9f6b3, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 16277bc4, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 58790054, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 49af6bea, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 6e26fe96, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 228fe496, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 3ae50efb, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 5427a502, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 6c7cb4c5, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 29f5cb5b, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 53bc12b9, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 588460d1, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 518bca90, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 16254ff6, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 50efd2be, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 5fdb4d47, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 3eff3c1e, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 4bc9f07e, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 4f17b83a, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 281b6a0d, Branch outcome: 0, Prediction: 1, correct/miss: 1
Branch instruction address: 3816bde9, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 1ed01131, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 2b3ac282, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 0b165276, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 200dd4c1, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 04da6239, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 71854ec1, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 4ca74e8c, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 42971e4a, Branch outcome: 0, Prediction: 0, correct/miss: 0
Branch instruction address: 13a9496b, Branch outcome: 1, Prediction: 1, correct/miss: 1
Branch instruction address: 122b4d57, Branch outcome: 1, Prediction: 0, correct/miss: 0
Branch instruction address: 75968f31, Branch outcome: 1, Prediction: 0, correct/miss: 0
```