Performance Analysis and Comparison of Alpha Branch Predictor and Perceptron Branch Predictor

Advanced Computer Architecture-1

TEAM - 16

Team Members:
Akhileswari Sirigineedi
Keerthanaa Bhoopathy
Enoch Vasimalla
Dhruthiee Nelli

Motivation

Why Branch Prediction?

- Essential for modern processors to reduce pipeline stalls.
- Reducing mispredictions improves execution speed.
- Increase the Performance .

ALPHA Processor 21264

Key Features

- Superscalar: Executes up to 4 instructions per cycle.
- High clock speed (500-600 MHz).

ALPHA Branch Predictor

Branch Predictor Architecture

Components

1. **Local Predictor**: Tracks individual branch history.
2. **Global Predictor**: Uses global execution paths.
3. **Choice Predictor**: Selects the better predictor dynamically.

Branch Predictor Highlights

- Combines local and global predictors.
- Dynamic chooser adapts to branch behavior.

Architecture

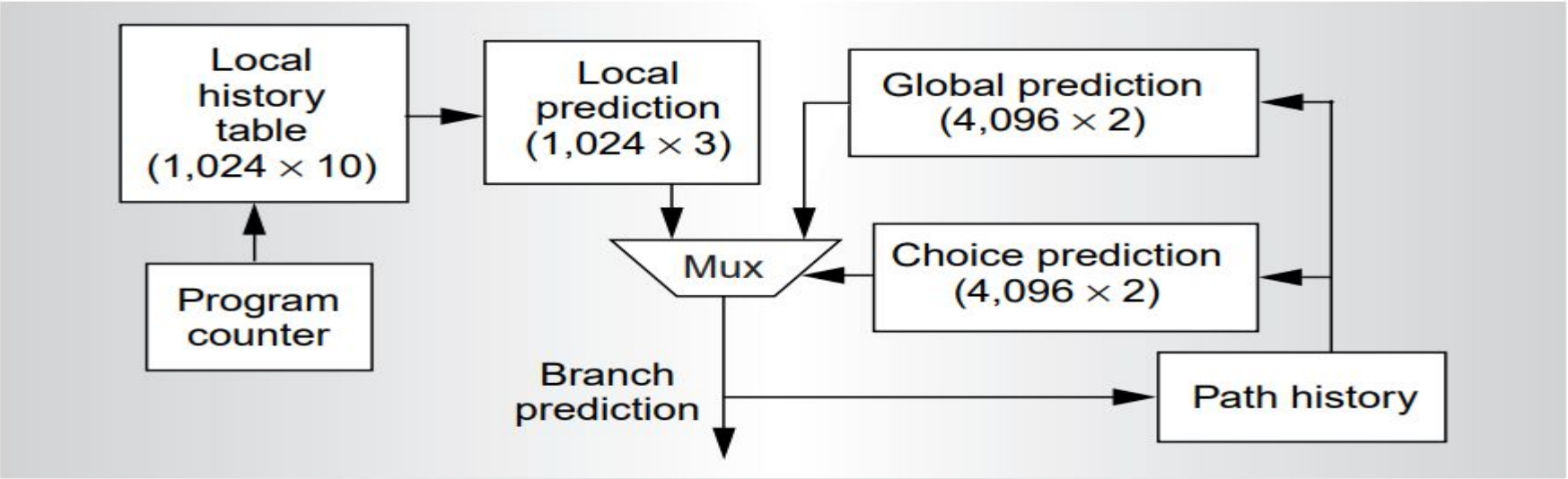


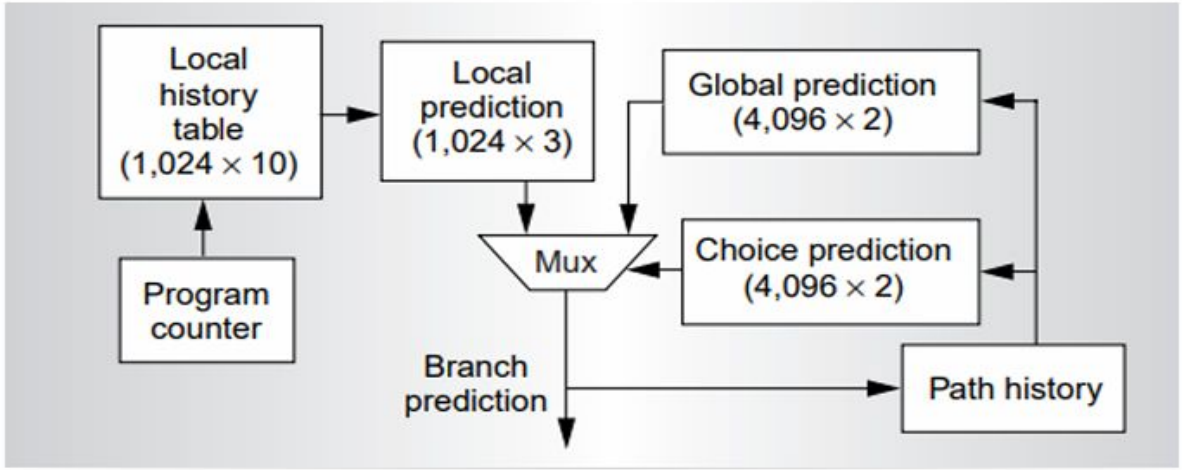
Figure 4. Block diagram of the 21264 tournament branch predictor. The local history prediction path is on the left; the global history prediction path and the chooser (choice prediction) are on the right.

ALPHA PREDICTOR:

BUDGET SIZE:

1. Local history table: 1K x 10 bits = 10 Kib ; Size calculation for the local history table.
2. Local predictors: 1K x 3 bits = 3 Kib ; Size calculation for local predictors.
3. Global predictors: 4K x 2 bits = 8 Kib ; Size calculation for global predictors.
4. Choice predictors: 4K x 2 bits = 8 Kib ; Size calculation for choice predictors.

Total = 29 Kib x 1 byte/8 bits = 3.625 KB ; Summing up the individual components and converting to KB.



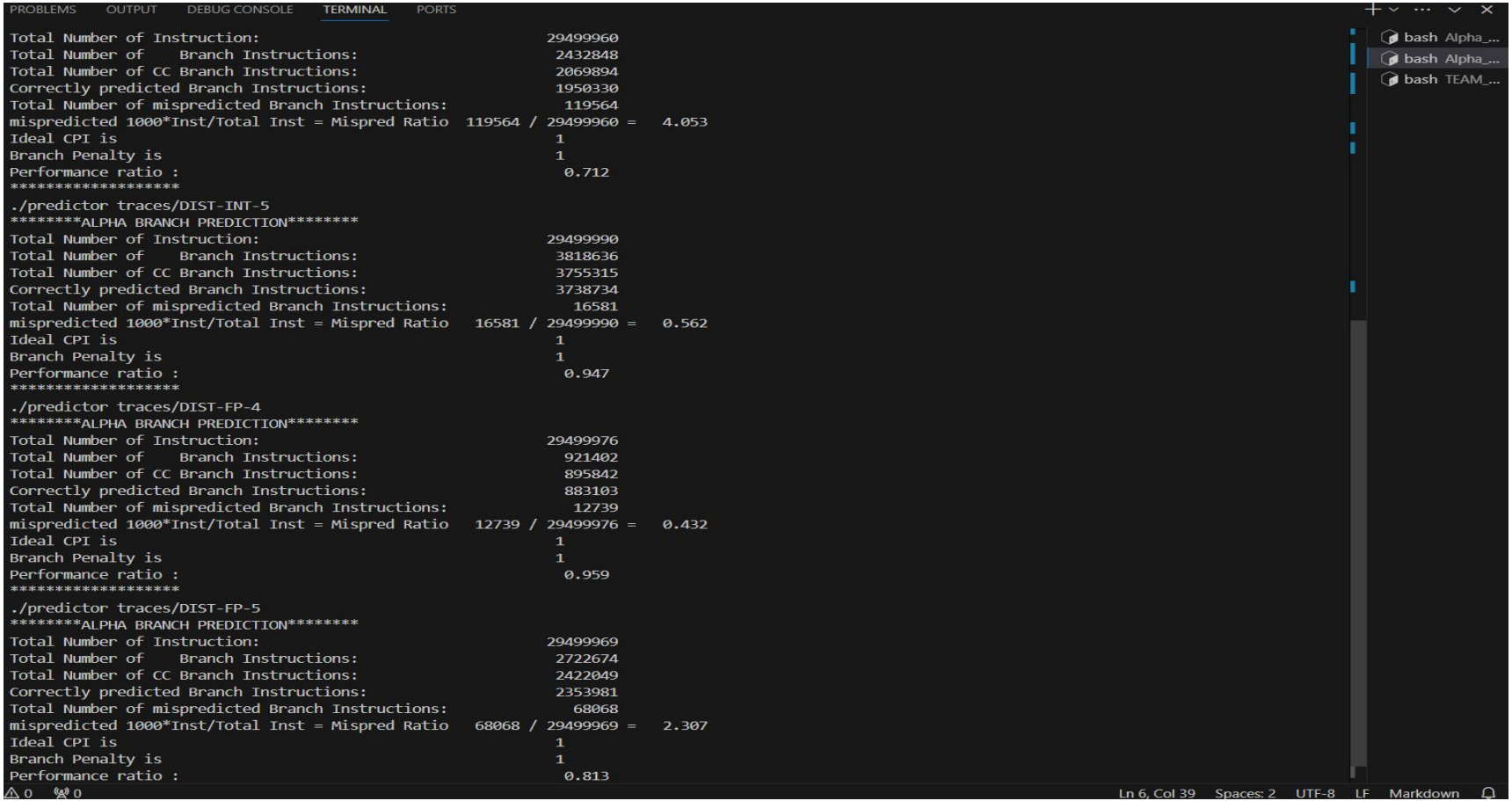
Budget ALPHA Increased

Increased Table Size:

- Local History Table: 4K x 12 =48Kib
- Local prediction: 4K x 5 = 20Kib
- Global prediction: 4K x 5 = 20Kib
- Choice predictor: 4K x 5 = 20Kib

Total = 108Kib
In Bytes = 13KB

Results -Alpha Predictor



Results -Alpha predictor Increased

```
akhi@rita:~/Desktop/TEAM_16_FALL_2024/TEAM_16/Alpha_Increased$ make run
./predictor traces/DIST-INT-4
********ALPHA BRANCH PREDICTION********
Total Number of Instruction:                          29499960
Total Number of    Branch Instructions:                2432848
Total Number of CC Branch Instructions:                2069894
Correctly predicted Branch Instructions:               1975652
Total Number of mispredicted Branch Instructions:        94242
mispredicted 1000*Inst/Total Inst = Mispred Ratio   94242 / 29499960 =   3.195
Ideal CPI is                                           1
Branch Penalty is                                      1
Performance ratio :                                     0.758
********************
./predictor traces/DIST-INT-5
********ALPHA BRANCH PREDICTION********
Total Number of Instruction:                          29499990
Total Number of    Branch Instructions:                3818636
Total Number of CC Branch Instructions:                3755315
Correctly predicted Branch Instructions:               3736027
Total Number of mispredicted Branch Instructions:        19288
mispredicted 1000*Inst/Total Inst = Mispred Ratio   19288 / 29499990 =   0.654
Ideal CPI is                                           1
Branch Penalty is                                      1
Performance ratio :                                     0.939
********************
./predictor traces/DIST-FP-4
********ALPHA BRANCH PREDICTION********
Total Number of Instruction:                          29499976
Total Number of    Branch Instructions:                 921402
Total Number of CC Branch Instructions:                 895842
Correctly predicted Branch Instructions:                885546
Total Number of mispredicted Branch Instructions:        10296
mispredicted 1000*Inst/Total Inst = Mispred Ratio   10296 / 29499976 =   0.349
Ideal CPI is                                           1
Branch Penalty is                                      1
Performance ratio :                                     0.966
********************
./predictor traces/DIST-FP-5
********ALPHA BRANCH PREDICTION********
Total Number of Instruction:                          29499969
Total Number of    Branch Instructions:                2722674
Total Number of CC Branch Instructions:                2422049
Correctly predicted Branch Instructions:               2404914
Total Number of mispredicted Branch Instructions:        17135
mispredicted 1000*Inst/Total Inst = Mispred Ratio   17135 / 29499969 =   0.581
```

bash Alpha_...
bash TEAM_...
bash TEAM_...

Results Comparison

Traces	**ALPHA** **Miss-predictions per every thousand instructions**	**ALPHA_Increased** **Miss-predictions per every thousand instructions**
Floating point-1	0.432	0.349
Floating point-2	2.307	0.581
Integer-1	4.053	3.195
Integer-2	0.562	0.654

Perceptron Branch Prediction

- First neural network learning model in the 1960’s
- Simple and limited (single layer model)
- Basic concepts are similar for multi-layer and deep models so this is a good learning tool
- Still used in some current applications (large business problems, where intelligibility is needed, etc.)
- Most commonly used dynamic branch prediction.

How Perceptrons Work?

- Single layer perceptron consists of one artificial neuron connecting several inputs by weighted edges to one output unit.
- Perceptron keeps track of the Positive and negative correlation between Global history and branch predicted.
- Output of the perceptron is computed by

$$y = w_0 + \sum_{i=1}^{n} x_i w_i .$$

The inputs to our perceptrons are *bipolar*, i.e., each x_i is either -1, meaning *not taken* or 1, meaning *taken*. A negative output is interpreted as *predict not taken*. A non-negative output is interpreted as *predict taken*.

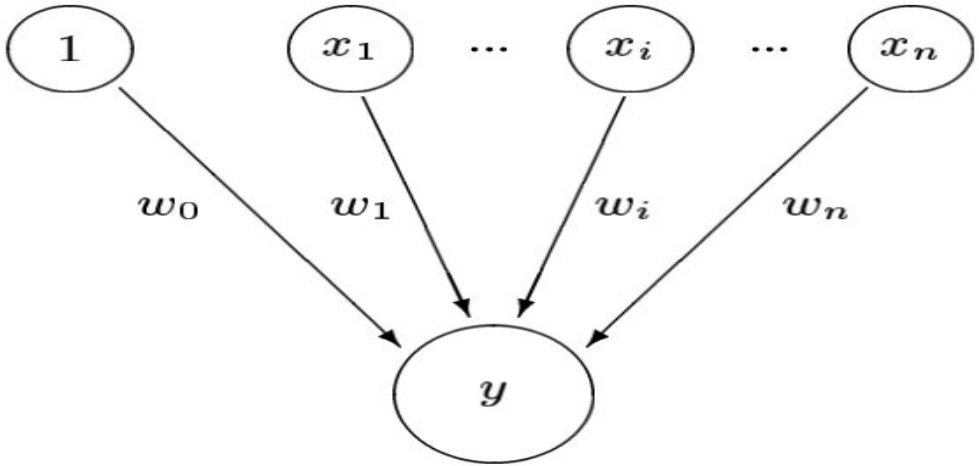


Figure 1: Perceptron Model. The input values $x_1, \ldots, x_n$, are propagated through the weighted connections by taking their respective products with the weights $w_1, \ldots, w_n$. These products are summed, along with the bias weight w_0, to produce the output value y.

3.3 Training Perceptrons

Once the perceptron output y has been computed, the following algorithm is used to train the perceptron. Let t be -1 if the branch was not taken, or 1 if it was taken, and let θ be the *threshold*, a parameter to the training algorithm used to decide when enough training has been done.

```
if sign(y_out) ≠ t or |y_out| ≤ θ then
        for i := 0 to n do
                w_i := w_i + t x_i
        end for
end if
```

How Branch Prediction is calculated ?

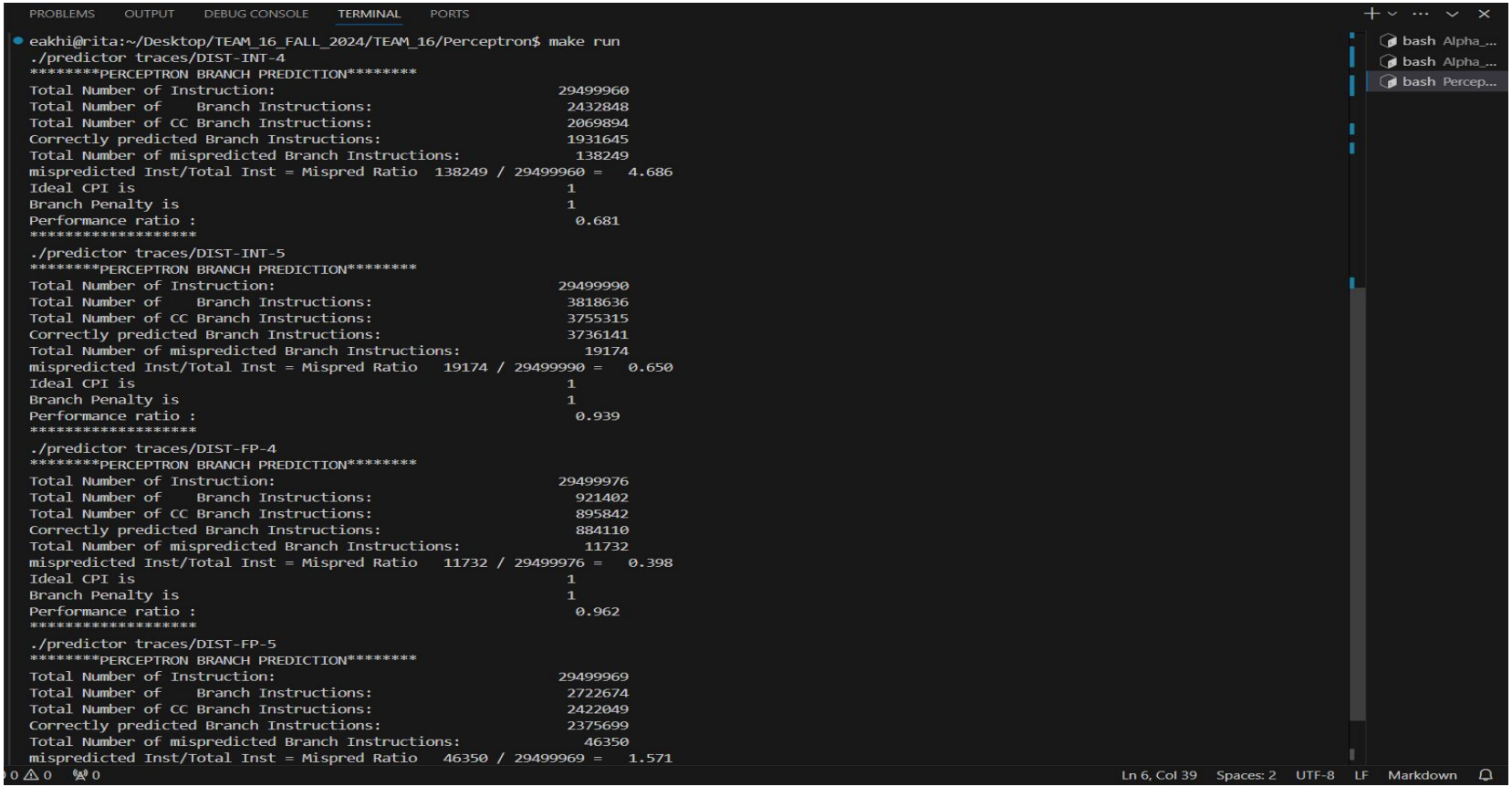
dw= x*t w=dw+w-old yin>0; taken yin<0 not taken ; t=1; taken t=-1 ;not taken

x1	x2	x3	x4	t	yin	yo	dw 1	dw 2	dw 3	dw 4	db	W 1	w2	w3	w4	b
1	1	1	1	1	0	-1	1	1	1	1	1	1	1	1	1	1
-1	1	-1	-1	1	-1	-1	-1	1	-1	-1	1	0	2	0	0	2
1	1	1	-1	-1	4	1	-1	-1	-1	1	-1	-1	1	-1	1	1
-1	1	-1	1	-1	1	1	-1	1	1	-1	-1	-2	2	0	0	0

Metrics Calculated

- Total number of Branching Instruction
- Total number of predictions
- Total Number of mispredictions
- Mispredicted ratio
- Performance ratio =(1/1+misprediction_ratio*miss_penalty)

Results Perceptron



Final Results Comparison (ALPHA Increased, ALPHA,Perceptron)

Traces	ALPHA Miss-predictions per every thousand instructions	ALPHA_Increased Miss-predictions per every thousand instructions	Perceptron
Floating point-1	0.432	0.349	0.398
Floating point-2	2.307	0.581	1.571
Integer-1	4.053	3.195	4.686
Integer-2	0.562	0.654	0.650

performance metrics of the Alpha and Perceptron predictors across different benchmarks

Predictor Type	Trace	Total Instructions	Correctly Predicted	Misprediction Ratio	Performance Ratio
Alpha (Default)	DIST-INT-4	29,499,960	19,765,252	3.195	0.758
Alpha (Default)	DIST-INT-5	29,499,990	37,563,427	0.654	0.939
Alpha (Increased)	DIST-INT-4	29,499,960	11,596,940	4.053	0.737
Perceptron	DIST-INT-4	29,499,960	13,824,940	4.686	0.681
Perceptron	DIST-INT-5	29,499,960	37,697,341	0.651	0.947

Observations

- Based on the observations Perceptron Branch Prediction gives better Performance in comparison to the Alpha Branch predictor.

- When the budget of the Branch Predictor is increased (Table size and Branch Prediction Bits) Alpha predictor gives better results when compared to perceptron Branch Prediction.

References

Alpha21264 - https://en.wikipedia.org/wiki/Alpha_21264

Perceptron - https://www.cs.utexas.edu/~lin/papers/hpca01.pdf

Framework - https://jilp.org/cbp/

THANK YOU