EECE 7352: Computer Architecture

Assignment 3

Jiayun Xin

NUID: 001563582

College of Engineering

Northeastern University Boston, Massachusetts

Spring, 2022

Q1

The four main findings are listed below:
1. There are large performance gaps among the four platforms because of the difference of frequency and microarchitectures, in which the execution time gaps are >2x and the average cycle count gaps are <= 2.5x.
2. The effects of ISA were unclear between x86 and ARM implementations through considering the comparison results of instruction count and mix.
3. The microarchitecture decides the performance of different platforms, such as both more accurate branch predictor and larger caches can produce a higher performance.
4. x86 implementation will cost much more power than ARM implementations. There is a power gap >= 3x.

The key limitation and weaknesses can be divided into four parts. In terms of cores, many variables cannot be controlled such as uniform platform, design teams and even the purity of RISC and CISC implementations. In addition, the paper did not state the reasons why chosen the specific benchmark on workload. There are also some errors caused on tools like decoder power measure, performance counters and simulations. For scaling, there are some limitation factors. For example, the effects of memory rates are not linear, the minimum voltage affects frequency scaling and the numbers in the ITRS table are not accurate.

Q2

Dynamic Branch Prediction with Perceptrons

1. The perceptron predictor can produce a more accurate prediction result than other dynamic global branch predictors. Within the same budget and number of branches executed, the perceptron predictor always has the lowest misprediction rate nearly.

2. The perceptron predictor can consider a much longer history length than those traditional predictors and the longest history length considered is 62 with 4.64% misprediction rate.

3. The perceptron predictor can have a better prediction result on average when branches are linearly separable. The perceptron predictor sometimes produces a less accurate result when branches are linearly inseparable.

BranchNet: A Convolutional Neural Network to Predict Hard-To-Predict Branches

1. The Convolutional Neural Network - BranchNet can have lower branch mispredictions per kilo instructions (MPKI) than TAGE-SC-L when predicts some hard-to-predict branches because it ignored those unrelated noise in the history.

2. With unlimited storage of CBP 2016, using both Big-BranchNet and MTAGE-SC has a lower MPKI values than using MTAGE-SC only in which Big-BranchNet shows vary improvement effects with different benchmark.

3. BranchNet can have lower branch MPKI than Tarsa's CNNs because the mini-BranchNet architecture predicts branch with longer histories, deeper network and less storage.

4. There is a maximum input that reveals all input-independent correlations and minimums MPKI values which related to storage budget of branch prediction model.

Q3

Main problem:

The main problem of this paper is trying to find dynamic branch predictions other than history-based branch predictions that uses stored value to compute branch flags ahead of time.

```
#define DCTSIZE2        64                                          1
#define DIVIDE_BY(a,b)  a /= b                                      2
                                                                    3
void forward_DCT ( ... )                                            4
{                                                                   5
    /* work area for FDCT subroutine */                             6
    DCTELEM workspace[DCTSIZE2];                                    7
                                                                    8
    /* Perform the DCT */                                           9
    (*do_dct) (workspace);                                          10
                                                                    11
    register DCTELEM temp, qval;                                    12
    register int i;                                                 13
    register JCOEFPTR output_ptr = coef_blocks[bi];                 14
                                                                    15
    for (i = 0; i < DCTSIZE2; i++) {                                16
        qval = divisors[i];                                         17
        temp = workspace[i];                                        18
                                                                    19
        if (temp < 0) {                                             20
            temp = -temp;                                           21
            temp += qval>>1;                                        22
            DIVIDE_BY(temp, qval);                                  23
            temp = -temp;                                           24
        } else {                                                    25
            temp += qval>>1;                                        26
            DIVIDE_BY(temp, qval);                                  27
        }                                                           28
        output_ptr[i] = (JCOEF) temp;                               29
    }                                                               30
}                                                                   31
```

The screenshot captured from the paper shows a LD-BR execution sequence in which line 20 is a hard-to-predict data-dependent branch. The existing history-based branch predictor TAGE did not improve its prediction accuracy. If the stored value to compute branch condition flags in the c code example have been calculated before branching, it would be a good branch prediction.

Why do authors argue that history-based branch predictions are not sufficient?

The reason can attribute to as follow:

1. There is a “store to branch delay” in the history-based branch predictions because it does not have repeatable pattern of those data values stored in data structure.

2. Tested by many benchmarks, using history-based branch predictions alone has a relatively high MPKI. TAGE + SLB can reduce MPKI.

3. Using history-based branch predictors like TAGE predictor has a low prediction speed. Adding SLB into TAGE can effectively improve the prediction speedup.

4. In terms of area, power and access time analysis, there are no obvious differences between tradition branch predictor and SLB predictor.