ECE552 Lab 2

Jay Mohile (Predictors, Benchmarks, Report)

Letian Zhang (Predictors, Benchmarks, Report)

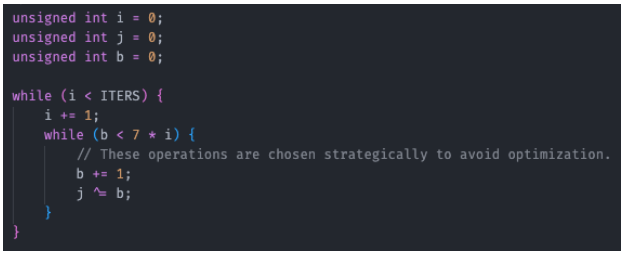
I. MICROBENCHMARK

Due to the stateful nature of branch prediction, this lab was significantly harder to measure than the first one. In particular, a benchmark's results depend on the configuration left by the overhead code, as well as the precise addresses of the branches. As such, rather than the cycle-accurate benchmarks from lab 1, a number of 'sanity check' benchmarks are instead used for the 2 level predictor, all with -O1 optimization.

Benchmark 0) **Base Case**: using an empty main loop, 1813 mispredictions are measured.

Benchmark 1) **"Memorizable" Nested Loop**

A detailed breakdown of this code/its assembly is provided in mb.c, lines 90-140. In summary, a nested loop is created such that the inner loop has a period of 7, the maximum memorizable with 6-bit history. The generated assembly is very different from this C, so please refer to *mb.c* for a detailed analysis of the expected statistics.



```
unsigned int i = 0;
unsigned int j = 0;
unsigned int b = 0;

while (i < ITERS) {
    i += 1;
    while (b < 7 * i) {
        // These operations are chosen strategically to avoid optimization.
        b += 1;
        j ^= b;
    }
}
```

Assuming each loop gets its own pattern/history register, ~7-10 mispredictions are expected [Appendix B, mb.c L90-140]. GCC appeared to be incredibly sensitive, either collapsing into a single branch instruction or convoluting into three. Ultimately 7 mispredic

Benchmark 2) **"Unmemorizable" Nested Loop**

The same loop as (2) was used, but using a period of 8. Given the inside loop usually 'takes', this would lead to a conflict where a history of "TTTTTT" should lead to "T" on the second last iteration, and "N" on the last one. This manifests as a misprediction every cycle, which was successfully measured.

II. RESULTS

Mispredictions

Bench. Name	**2 Bit Saturating**		**2 Level**		**TAGE**	
	# Mis.	MPKI	# Mis.	MPKI	# Mis.	MPKI
astar	3695830	24.639	1785464	11.903	629744	4.198
bwaves	1182969	7.886	1071909	7.146	614255	4.095
bzip2	1224967	8.166	1297677	8.651	1565922	10.439
gcc	3161868	21.079	2223671	14.824	558027	3.72
gromacs	1363248	9.088	1122586	7.484	986491	6.577
hmmer	2035080	13.567	2230774	14.872	1941903	12.946
mcf	3657986	24.387	2024172	13.494	1605604	10.704
soplex	1065988	7.107	1022869	6.819	811295	5.409

Please note: all relevant content is in the first two pages. An appendix has been added with some raw data, in case it is required.

III. OPEN-ENDED PREDICTOR

Background

The open-ended predictor developed for this lab is based on TAGE, a published architecture from several years ago [1]. While much of this paper's overall organization was utilized, the implementation is completely original (i.e, no code was reviewed). Furthermore, significant work went into tuning the architecture, as described below.

TAGE builds on a body of predictors that yield two interesting insights.

Firstly, different loops are suited for different history lengths. Short histories can be trained faster by short loops, but are saturated by longer loops — and of course the converse is true. TAGE belongs to a family of Geometric History Length predictors, which maintain several pattern-history tables of varying history length (1, 2, 4, 8…). This way, each branch draws its predictions from the history most suited for it — which can also change over time.

Second, TAGE disconnects the number of rows in a pattern-history table, from the length of history it tracks. For example, a table tracking 128 bit history could be stored in a much smaller table by applying a fixed-length hash to the nominal index. While aliasing is guaranteed, mechanisms exist (see below) to ensure this is better than nothing.

Organization

- 64-bit Global History Buffer
- 7 Pattern History Tables
    - **7-bit Entries**: (common across tables): 3 tag bits, 2 "usefulness" bits, 2-bit saturating counter.
        - Tag bits prevent aliased PCs from simultaneously sharing an entry.
        - Usefulness bits track how often an entry has improved predictions, to decide when to release it to aliased PCs.
        - A single PC can own entries across tables, longest history is predicted.
    - **Sizes** *(history bits, rows, total bits)***:**
        - (0, 8, 56), (1,16, 112), (2, 32, 224), (4, 64, 224), (8, 800, 5600), (16, 1200, 8400), (32, 16100, 112700)
    - Tables are indexed by concatenating a 5-bit keyed PC and N-length history, and hashing.
- **Total Space:** 127604 bits = 15951 bytes (allowed: 128000 bits = 16kB)

IV. CACTI SIMULATION

The 2-level predictor contains two tables: a 512 entry history table with 6 bit histories, and a 512 (8 * 64) pattern table with 2-bit counters. Since these occupy almost a full block, and have no tags, they are best represented by a pair of identical pure RAM caches, each with 512 bytes total and 1 byte blocks (2level-bpred). In total, 0.0021056 mm2 of area, 0.39 mW of leakage, and 0.328ns of latency (Appendix A).

The open-ended predictor contains a total of eight tables: a single global history buffer, and 7 pattern tables. The GHB is ignored since (a) its 64-bit structure is not well modeled, and (b) it is anyway far smaller than the rest of the predictor (128kb). The remaining 7 tables (open-ended-bpred-1…7) all use a tagged cache with 3-bit tags and 1 byte blocks (actually 7 bits), varying their overall sizes from 8 bytes to 16100 bytes. These could have been combined into a single large cache for modeling, but were not because (a) they are accessed in parallel, (b) although entries are 1-way associative within a table, they are technically N-way associate *across* tables, and as such should be modeled as parallel caches.

In total, these tables require [Appendix A] 0.04mm2 of area and dissipate 6.8mW of leakage power. While the model is not perfect, it is a conservative estimate of the overall cache requirements.

REFERENCES

[1]

André Seznec, Pierre Michaud. A case for (partially) tagged geometric history length branch predic- tion. The Journal of Instruction-Level Parallelism, North Carolina State University, 2006, 8, pp.23.

2level-pred

Access time (ns): 0.163585

Total dynamic read energy per access (nJ): 0.000420231

Total leakage power of a bank (mW): 0.195006

Cache height x width (mm): 0.0391054 x 0.0269215

open-ended-bpred-4

Power Components:

Data array: Total dynamic read energy/access (nJ): 0.000177105

Total leakage read/write power of a bank (mW): 0.0277939

Tag array: Total dynamic read energy/access (nJ): 0.000102761

Total leakage read/write power of a bank (mW): 0.0205643

Area Components:

Data array: Area (mm2): 0.00020049

Tag array: Area (mm2): 0.000104454

open-ended-bpred-5

Power Components:

Data array: Total leakage read/write power of a bank (mW): 0.332344

Tag array: Total leakage read/write power of a bank (mW): 0.15548

Area Components:

Data array: Area (mm2): 0.00169655

Tag array: Area (mm2): 0.000813647

open-ended-bpred-6

Power Components:

Data array: Total leakage read/write power of a bank (mW): 0.494461

Tag array: Total leakage read/write power of a bank (mW): 0.243768

Area Components:

Data array: Area (mm2): 0.00231887

Tag array: Area (mm2): 0.00115372

open-ended-bpred-7

Power Components:

Data array: Total leakage read/write power of a bank (mW): 5.5741

Tag array: Total dynamic read energy/access (nJ): 0.0020913

Area Components:

Data array: Area (mm2): 0.0244582

Tag array: Area (mm2): 0.0122281

APPENDIX B: Assembly for Primary Microbenchmark

```
 .file "mb.c"
 .text
 .globl  main
 .type main, @function
main:
.LFB0:
 .cfi_startproc
 movl  $7, %ecx
 movl  $0, %eax
 movl  $0, %edx
 jmp .L4
.L2:
 addl  $7, %ecx
 cmpl  $70007, %ecx
 je  .L7
.L4:
 cmpl  %eax, %ecx
 jbe .L2
.L3:
 addl  $1, %eax
 xorl  %eax, %edx
 cmpl  %eax, %ecx
 jne .L3
 jmp .L2
.L7:
 leal  10000(%rax,%rdx), %eax
 ret
 .cfi_endproc
.LFE0:
 .size main, .-main
 .ident  "GCC: (Debian 10.2.1-6) 10.2.1 20210110"
 .section  .note.GNU-stack,"",@progbits
```