Papers and Performance

Computer Architecture: Design and Simulation
CMU 15-346, Spring 2022

Outline

- Reading / writing papers
- Further metrics of performance
- Things that can go wrong

Some content taken from 18-344 lecture 10

Reading a research paper

- Professor says, "Read this paper"
  - Now what?

Bit-level Perceptron Prediction for Indirect Branches

Elba Garza
elba@tamu.edu
Texas A&M University and AMD

Samira Mirbagher-Ajorpaz
Tahsin Ahmad Khan
parisamir@tamu.edu
tahsinkhan@tamu.edu
Texas A&M University

Daniel A. Jiménez
djimenez@tamu.edu
Texas A&M University and
Barcelona Supercomputing Center

Papers are dense, complex

- **Start with the highlights**

ABSTRACT

Modern software uses indirect branches for various purposes including, but not limited to, virtual method dispatch and implementation of switch statements. Because an indirect branch's target address cannot be determined prior to execution, high-performance processors depend on highly-accurate indirect branch prediction techniques to mitigate control hazards.

This paper proposes a new indirect branch prediction scheme that predicts target addresses at the bit level. Using a series of perceptron-based predictors, our predictor predicts individual branch target address bits based on correlations within branch history. Our evaluations show this new branch target predictor is competitive with state-of-the-art branch target predictors at an equivalent hardware budget. For instance, over a set of workloads including SPEC and mobile applications, our predictor achieves a misprediction rate of 0.183 mispredictions per 1000 instructions, compared with 0.193 for the state-of-the-art ITTAGE predictor and 0.29 for a VPC-based indirect predictor.

What did the paper "tell us"?

6 CONCLUSION & FUTURE WORK

In this paper, we introduced Bit-Level Perceptron-Based Indirect Branch Predictor, or BLBP. BLBP predicts indirect branch targets' select lower-order bits using perceptron-based learning. A selection of known target addresses for the branch are gathered from the IBTB for comparison; the target address that matches closest with the predicted bits is output as BLBP's prediction.

We have shown BLBP to outperform the state-of-the-art. Using a suite of 88 benchmarks with significant indirect branches, we show BLBP improves upon ITTAGE's prediction performance by 5%, reducing MPKI from 0.193 to 0.183.

What is this paper about?

- What is the problem?
- What do you know about solving it?
- How might you approach the problem?

Reading Process

- Read: Abstract, Introduction, Conclusion
  - Then think about what the paper is about and how to approach the problem

- Next read: evaluation
  - Are the metrics appropriate to the problem?
  - Do the results support the claims?

- Finally (if time / counterintuitive)
  - Implementation details

Reviewing Papers (aside)

- When reviewing a paper, I read it straight through, often multiple times
  - Is the problem relevant to the venue?
  - What are the claims?
  - How are they supported?

Writing Papers

Useful Metrics

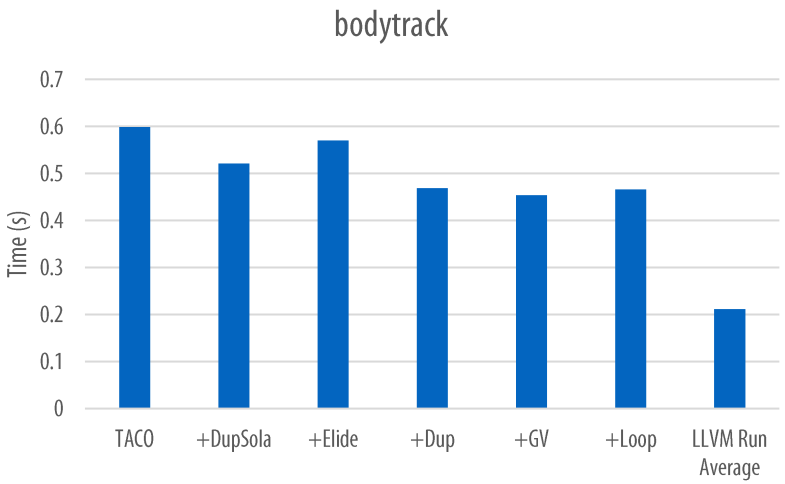
- **Performance:**
  - **Run time or "latency" (seconds to get a result)**
  - Throughput (results per second)
  - Instructions per Cycle (IPC) or Cycles per Instruction (CPI)
- **Energy:**
  - **Total energy**
  - Power (avg. or peak)
  - Area (absolute or normalized)
- **Efficiency:**
  - **Performance per power (results per second per Watt)**
  - Operations per Joule (same metric, just multiply by s/s!)
  - Throughput per unit area (results per second per mm^2)

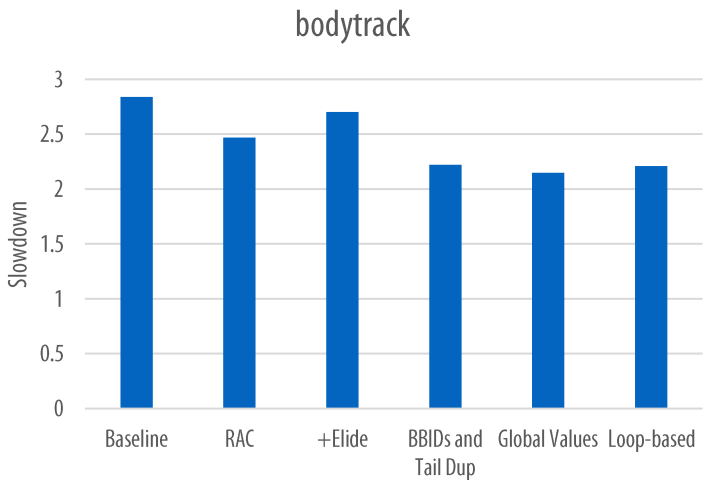
More Metrics

- **Cache-specific:**
  - Cache Misses per Kilo-Instruction
  - Cache Misses per Kilo-Memory-Instruction
  - Cache Misses per Cache Access
  - **Average Memory Access Time**
- **Control-specific:**
  - Branch prediction accuracy
  - **Branch Mispredictions per Kilo-Instruction**
- **Memory-specific:**
  - **Data throughput (GB/s)**
  - **Memory Latency (seconds per random byte)**

Normalizing and Simplifying

- Showing results (measurements are my own)





Significant Figures / Digits

- Show results based on some notion of significant figures
  - Ideally, you have statistical basis

	Baseline	RAC
bodytrack	2.836967	2.468247

	Baseline	RAC
bodytrack	2.84	2.47

Things that can go wrong

- Is –O3 really better than –O2?

Producing Wrong Data Without Doing Anything Obviously Wrong!

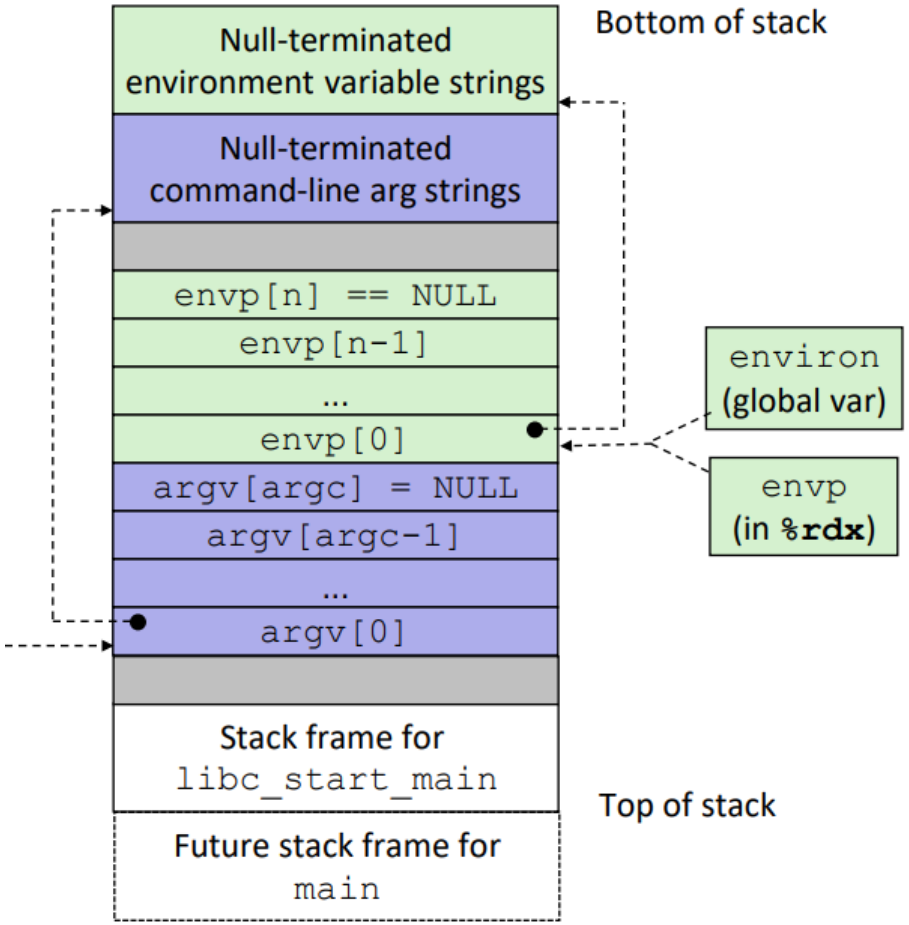
Todd Mytkowicz Amer Diwan
Department of Computer Science
University of Colorado
Boulder, CO, USA
{mytkowit,diwan}@colorado.edu

Matthias Hauswirth
Faculty of Informatics
University of Lugano
Lugano, CH
Matthias.Hauswirth@unisi.ch

Peter F. Sweeney
IBM Research
Hawthorne, NY, USA
pfs@us.ibm.com

Where are environment variables?

- They use stack space
  - Different alignments might matter



From 213 lecture 17 slides

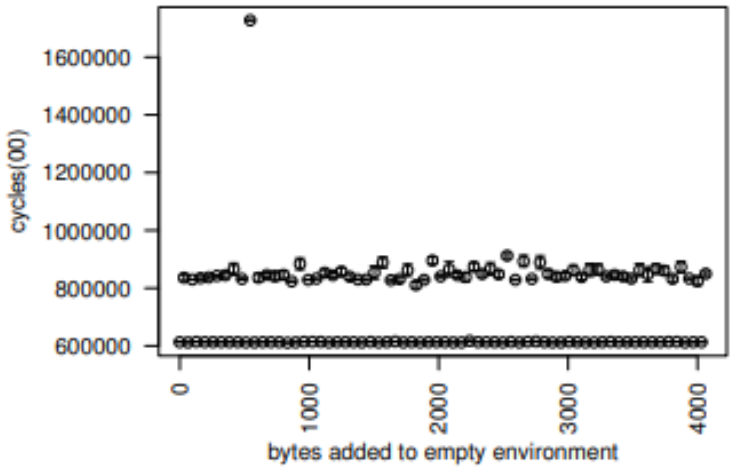
Environment Variable Size

- Page, cache line alignment

```
static int i = 0, j = 0, k = 0;

int main() {
  int g = 0, inc = 1;
  for (; g<65536; g++) {
    i += inc;
    j += inc;
    k += inc;
  }
  return 0;
}
```

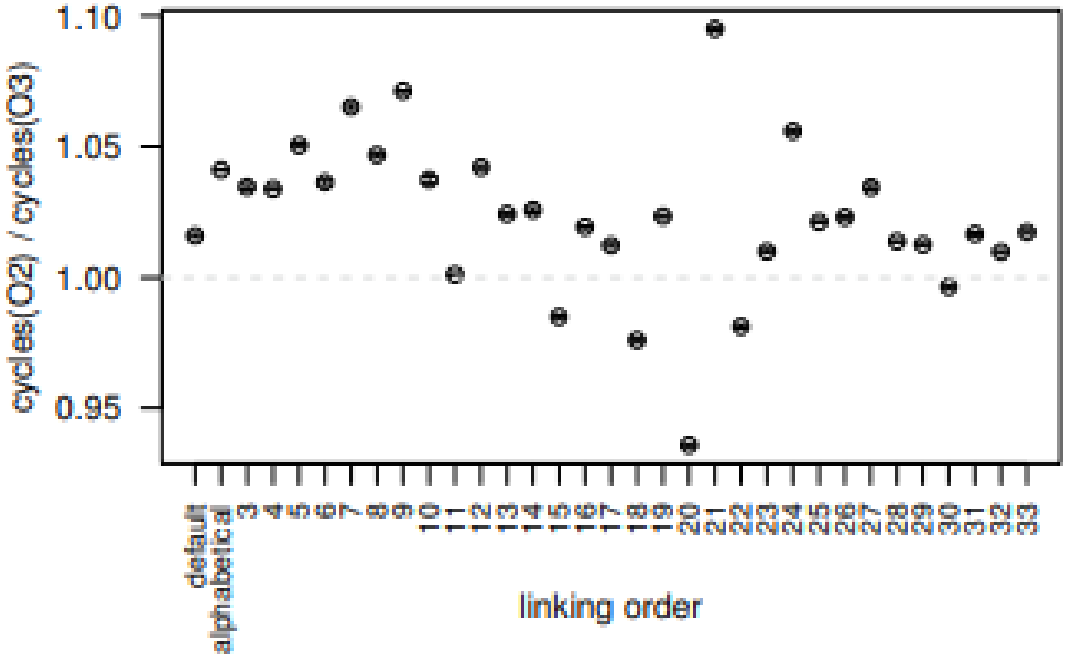
(a) C code for micro-kernel



(b) Effect of environment variable size on performance.

Linking order

- What order to link the required libraries



(a) Perlbench