Developing a Representativeness Measurement for Program Execution with Instruction-level Visualization

[1]Federico Cifuentes-Urtubey

Mentor: [2]Dr. Koppelman

CCT REU – July 29, 2015

[1]University of Maryland, Baltimore County, Baltimore, MD 21250

[2]Division of Electrical and Computer Engineering,
Louisiana State University, Baton Rouge, LA 70803

LSU
LOUISIANA STATE UNIVERSITY

UMBC
AN HONORS UNIVERSITY IN MARYLAND

Overview

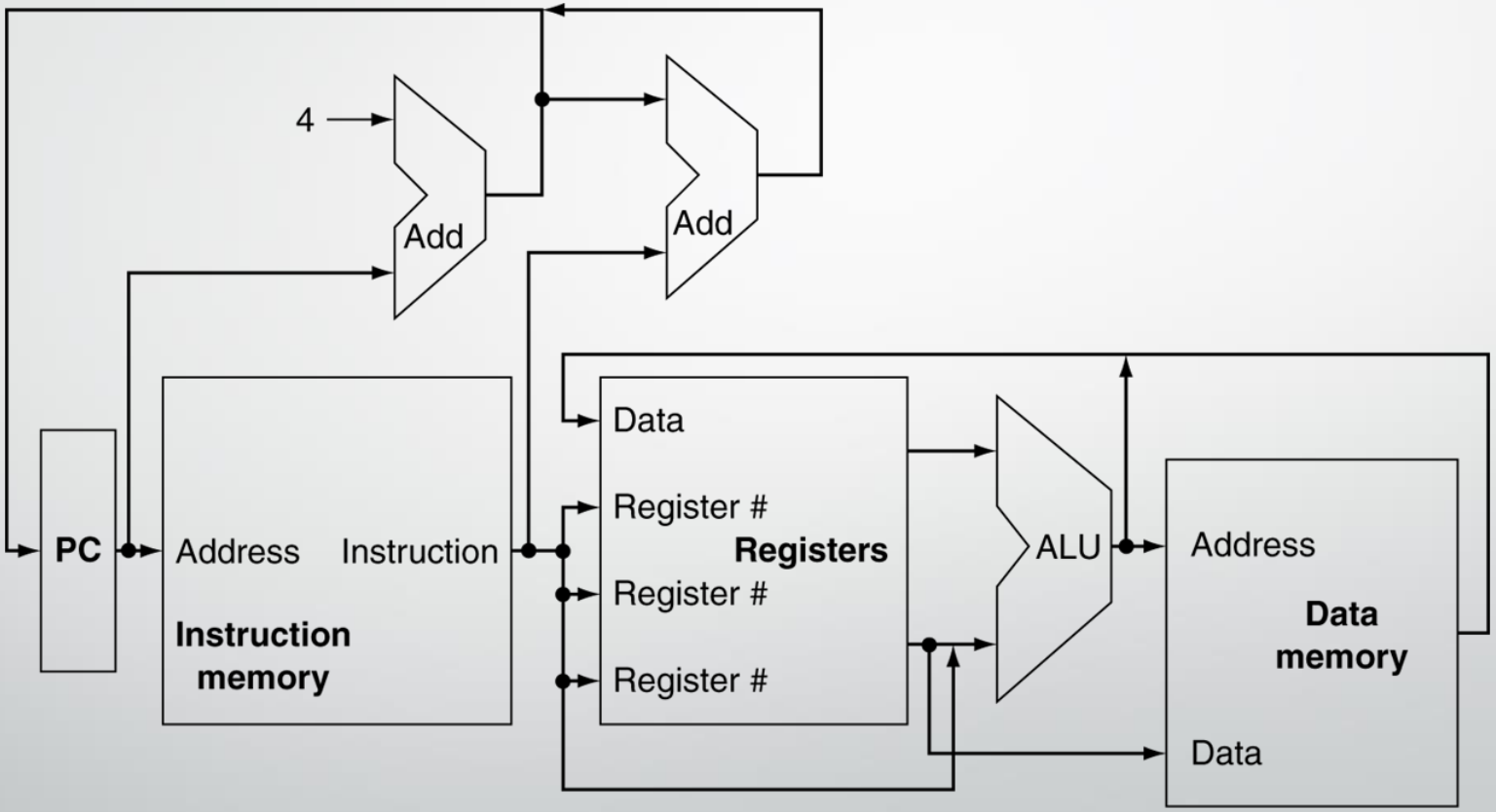
- Motivations and Background
- Purpose of Research
- Implementation of Research
- Measuring Representativeness
- Conclusions

Motivations and Background

- Interest in improving computer efficiency (hardware focus)
- Studying microarchitecture is a basis for hardware
- PSE: an instruction-level visualization tool
  - Assembly language: a low-level language
  - Helps find the causes of inefficiency

Source: http://www.techspot.com/articles-info/266/images/Image_03T.jpg

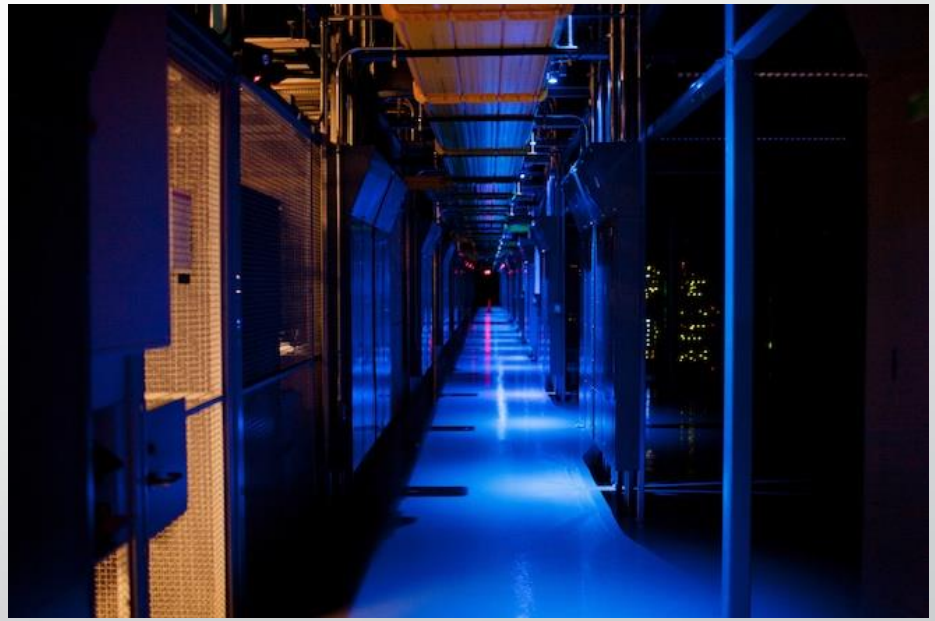
MIPS Microarchitecture



Source:http://1.bp.blogspot.com/-3x7rLV567sU/UM9SdcJRDeI/AAAAAAAAAMY/GG6UonlyCIk/s1600/multiplexor.png

Purpose

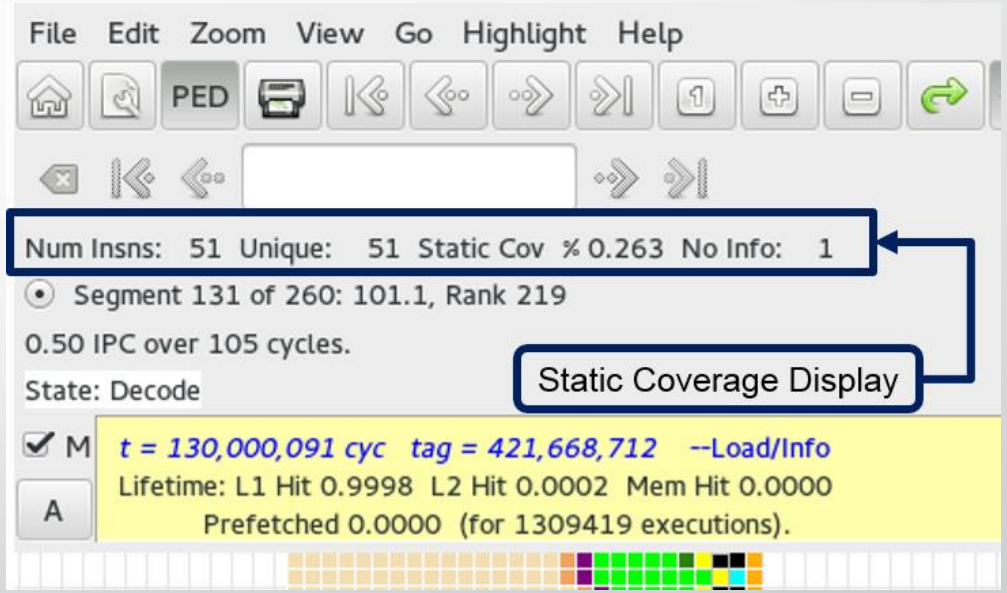
- PSE helps with identifying inefficiencies
  - Can lead to improved instruction code
  - Can aid in microarchitecture design and features
- Faster computers and other hardware
  - Supercomputers (i.e., distributed processing)
  - Servers (i.e., network traffic)
  - Game consoles



Source: http://www.wired.com/wp-content/uploads/blogs/wiredenterprise/wp-content/uploads//2012/03/equinix.jpg

Implementation

- Creating conceptual programs in C++
  - Computing a representativeness score of character coverage within a string
- Adding a static coverage feature to PSE
  - Number of static instructions visible divided by the amount in the segment



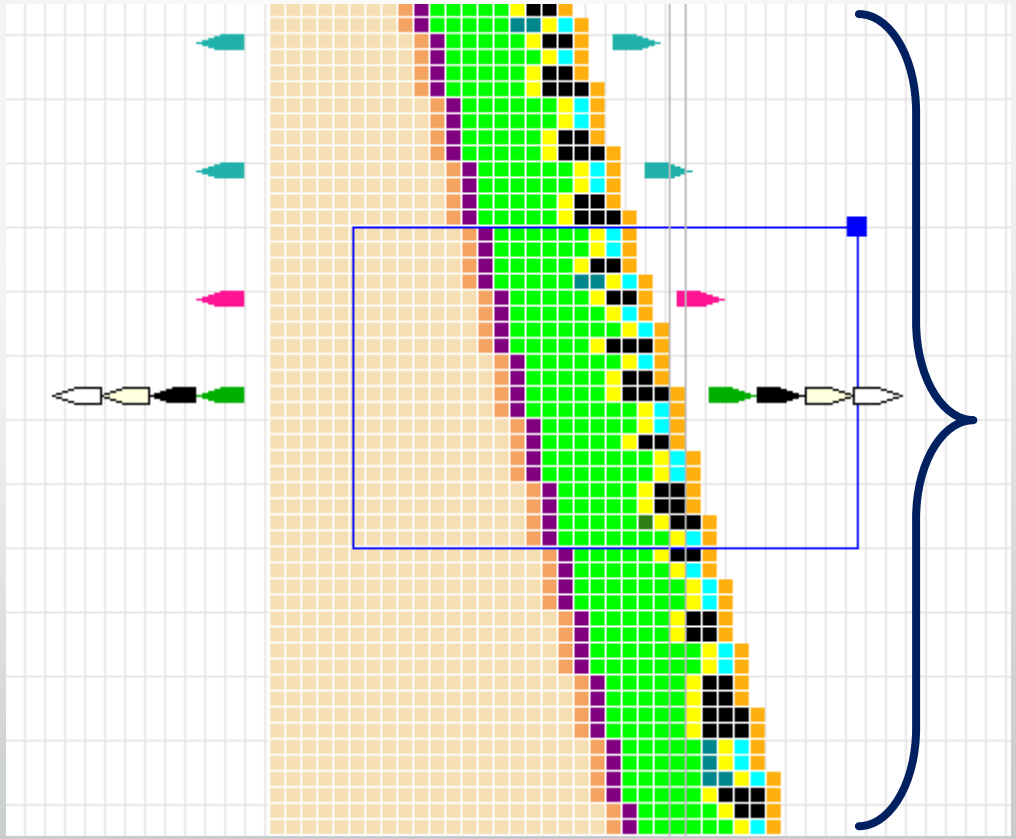
Measuring Representativeness

Using Integers

- *Static Coverage*:
  Unique instruction percentage
- *Dynamic Coverage*:
  Percentage using a frequency table

Using Suffix Trees

- *Path Representativeness*:
  Depicting jumps and branching
- *Event Representativeness*:
  Markers for an added trait to track streaming



Do these instructions cover some significant part of execution?

Conclusions

- Organizing instruction information saves time from manual searching
  - Representativeness gives coverage over various aspects in execution
- Developments can be further applied into:
  - Microarchitecture design research
  - Hardware performance evaluation

References

- Koppelman, D., & Michael, C. (2014). Discovering Barriers to Efficient Execution, Both Obvious and Subtle, Using Instruction-Level Visualization. Proceedings of the First Workshop on Visual Performance Analysis (VPA) 2014, 36-41. doi:10.1109/VPA.2014.11

- Kapoor, R. (2009). Avoiding the Cost of Branch Misprediction. Intel Developer Zone.

Acknowledgments

Thank you to Dr. Koppelman for a rich insight into computer architecture and the CCT for an excellent REU experience.

This material is based upon work supported by the National Science Foundation under award OCI-1263236 with additional support from the Center for Computation & Technology at Louisiana State University.