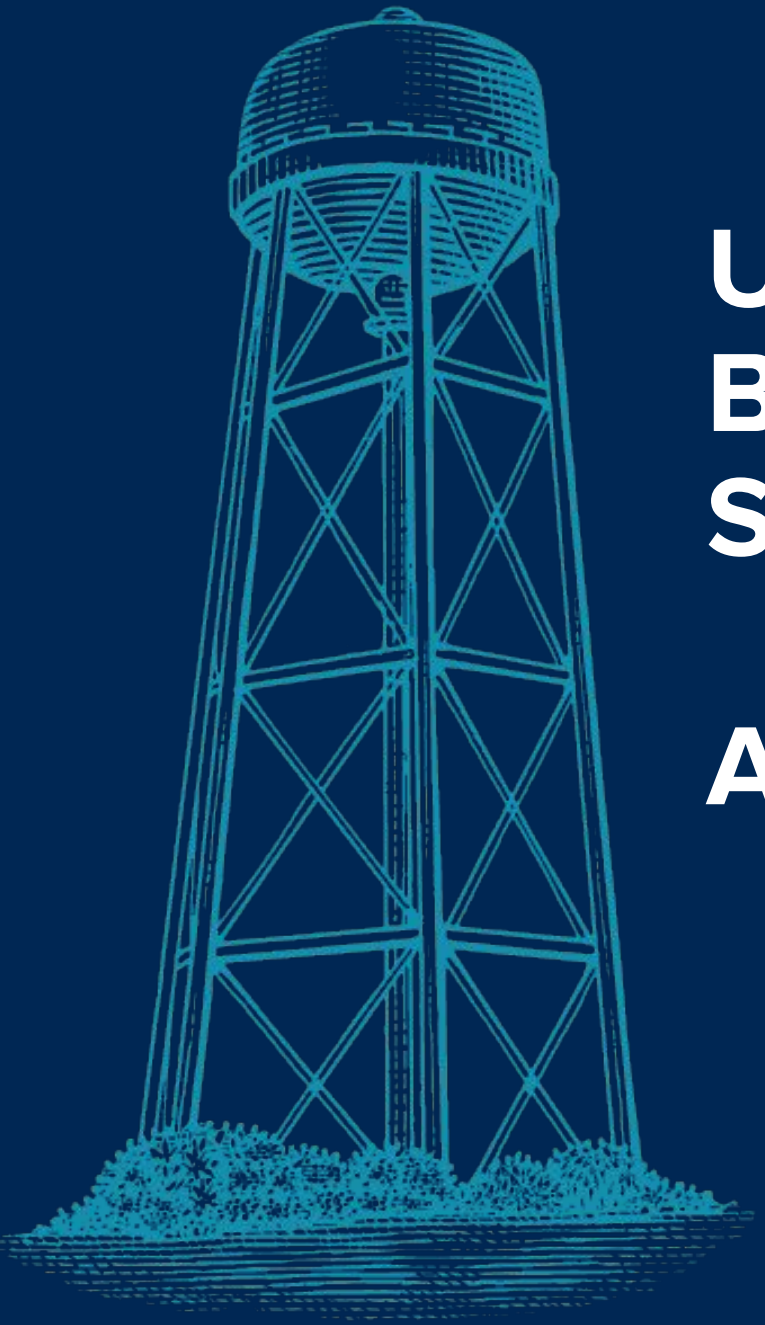


Using llvm-exegesis to Benchmark Memory-accessing Straightline Assembly

Aiden Grossman

UCDAVIS
COMPUTER SCIENCE

The Case for Benchmarking

- Tools like llvm-mca will never be 100% accurate.
- Need some obtaining ground truth data for systematic evaluation of performance analysis tools.
- Generation of training data for learned cost models.

		$BHive_U$		$BHive_L$	
µArch	Predictor	MAPE	Kendall	MAPE	Kendall
SKL	uiCA	0.45%	0.9798	0.38%	0.9895
	Ithemal	8.28%	0.8172	13.66%	0.7582
	IACA 3.0	13.49%	0.7802	14.26%	0.8290
	IACA 2.3	11.85%	0.8071	8.42%	0.8477
	OSACA	14.95%	0.7639	11.25%	0.8045
	llvm-mca-10	15.61%	0.7258	12.01%	0.8015
	llvm-mca-8	15.39%	0.7434	11.98%	0.8021
	Diff Tune	24.48%	0.6626	104.88%	0.6426
	CQA			7.44%	0.8847
	Measured [13]	4.40%	0.9113		
	Baseline	17.28%	0.7228	10.03%	0.7999

Chart from “uiCA: Accurate Throughput Prediction of Basic Blocks on Recent Intel Microarchitectures”.

UCDAVIS COMPUTER SCIENCE

Mircobenchmarking is difficult

- Details like kernel interrupts and TLB/cache misses can have a massive impact on benchmark results.
- Creating the execution environment in regards to elements like memory can be difficult.

Solution (for the second part): memory annotations in llvm-exegesis

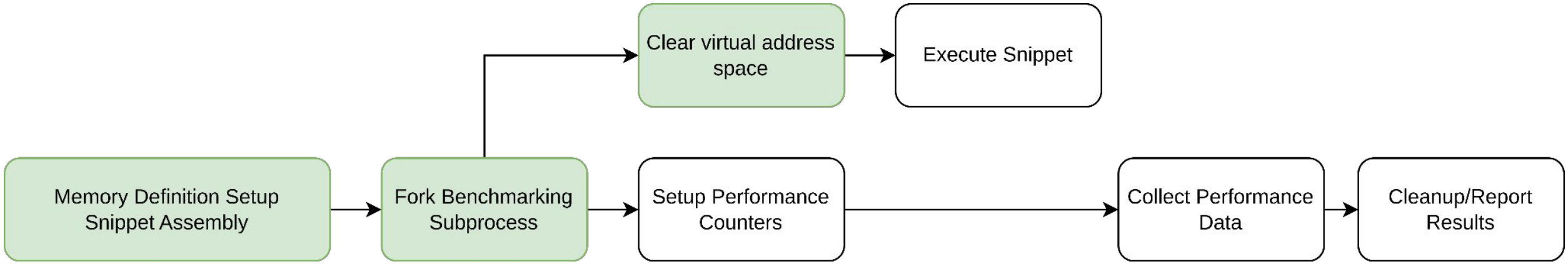
Memory Annotations

- *LLVM-EXEGESIS-MEM-DEF <value name> <size> <value>* - Allows the definition of a value that can be mapped into memory.
- *LLVM-EXEGESIS-MEM-MAP <value name> <address>* - Maps a memory value into the virtual address space of the benchmark.

Separate definitions/mapping to ensure control over pages.

Important if no L1 cache misses are desired.

How does this work?



Case Study: A Basic Block Accessing Memory

Let’s say we have the following basic assembly:

```
addq (%rax), %r11
addq (%rcx), %r11
```

Both instructions access memory, so we need to add memory annotations.

Case Study: Memory Annotations

Now let's add in some memory annotations

```
# LLVM-EXEGESIS-MEM-DEF example1 4096 42
# LLVM-EXEGESIS-MEM-MAP example1 131072
# LLVM-EXEGESIS-MEM-MAP example2 262144
# LLVM-EXEGESIS-DEFREG RAX 20000
# LLVM-EXEGESIS-DEFREG RCX 40000
addq (%rax), %r11
addq (%rcx), %r11
```

This will use a single page of memory.

Case Study: Benchmarking

Now let's run this through llvm-exegesis:

```
llvm-exegesis -mode=latency -snippets-file=path/to/snippet.S
-execution-mode=subprocess
```

Will return measurements:

```
measurements:
  - { key: latency, value: 1.0811, per_snippet_value: 2.1622 }
```

(These results are from a znver2 machine)

Limitations/Future Work

- Currently not a battle tested implementation. Systematic accuracy analysis/extensive testing being worked on.
- Minor quirks such as hex/decimal values in different places.
- Results can be fairly noisy.
  - No measurements/control over variables like kernel interrupts/cache misses currently.
- Automatic memory annotations given register values also being worked on.