LLVM Machine Code Analyzer (llvm-mca)

Souradip Ghosh

Agenda

1. **Background**
2. Mechanics of llvm-mca
3. Using llvm-mca
4. Potential applications for current projects

Background: History and Motivation

- Development History:
  - Built at Sony to debug their internal instruction scheduling models
  - Merged with LLVM's tools in early 2018 (LLVM 7/8)

- Motivations for llvm-mca [1]:
  - Measure performance for certain blocks of code
  - Diagnose bottlenecks, examine resource pressure, calculate latency, etc.
  - Compete with Intel (IACA)

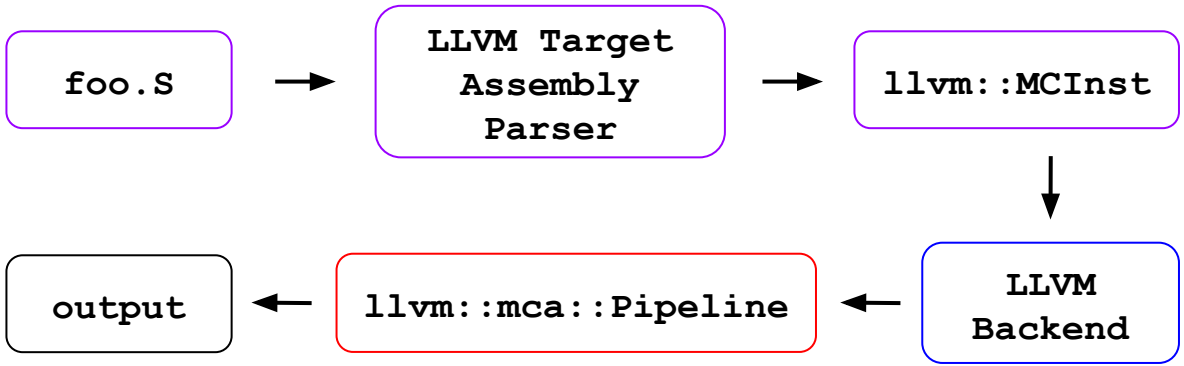
[1] Andrea di Biagio, Matt Davis, LLVM Development. "llvm-mca - LLVM Machine Code Analyzer" LLVM 10, 2019.

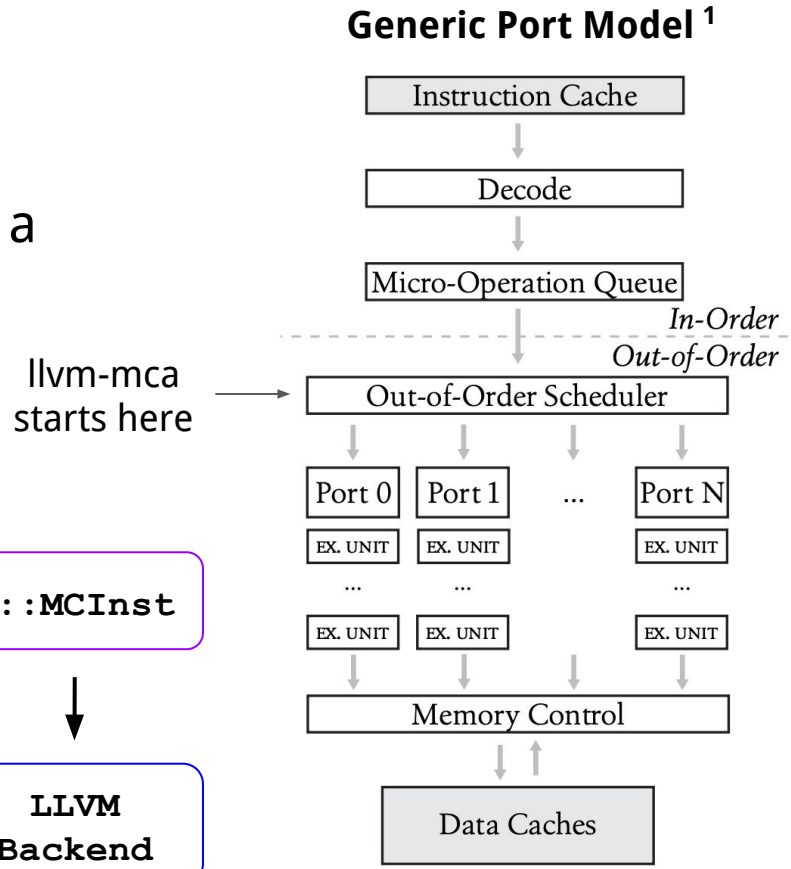
Agenda

1. Background
2. **Mechanics of llvm-mca**
3. Using llvm-mca
4. Potential applications for current projects

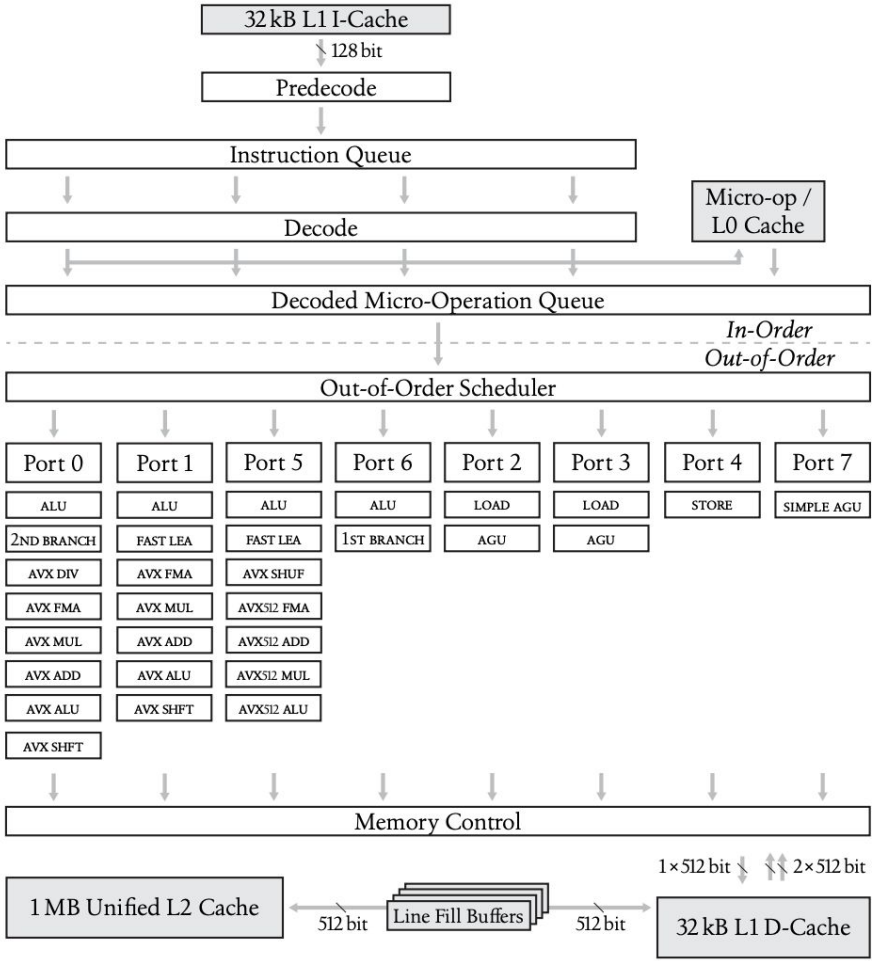
Mechanics: Static Analysis

- llvm-mca simulates the execution of a block of assembly
  - Uses available LLVM backend that supports out-of-order execution



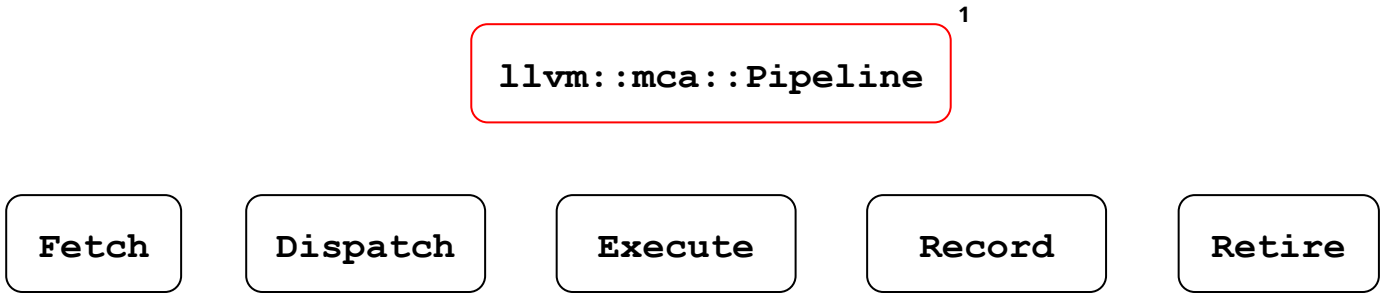


[1] Figure 1. Laukemann et al., *Automated Instruction Stream Throughput Prediction for Intel and AMD Microarchitectures* (2018). https://arxiv.org/pdf/1809.00912.pdf



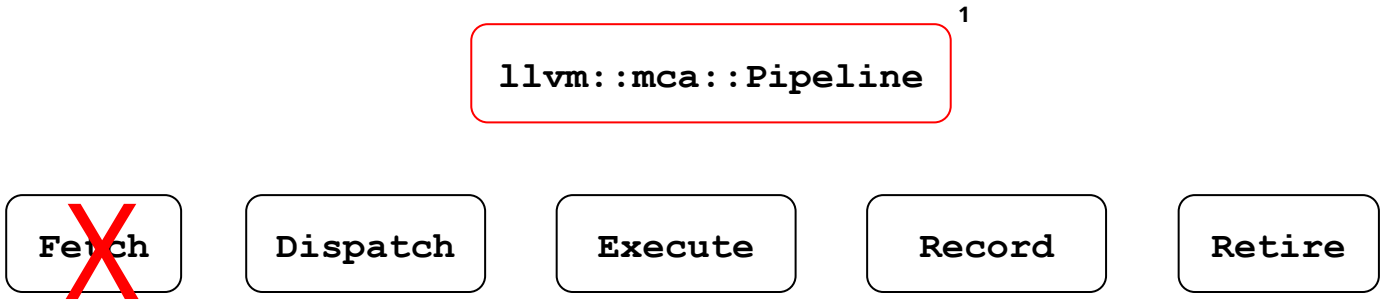
[1] Figure 2. Laukemann et al., *Automated Instruction Stream Throughput Prediction for Intel and AMD Microarchitectures* (2018). https://arxiv.org/pdf/1809.00912.pdf

Mechanics: Pipelining in llvm-mca



[1] Andrea di Biagio, Matt Davis. “Understanding the Performance of Code Using llvm-mca,” 2018 LLVM Developers’ Meeting.

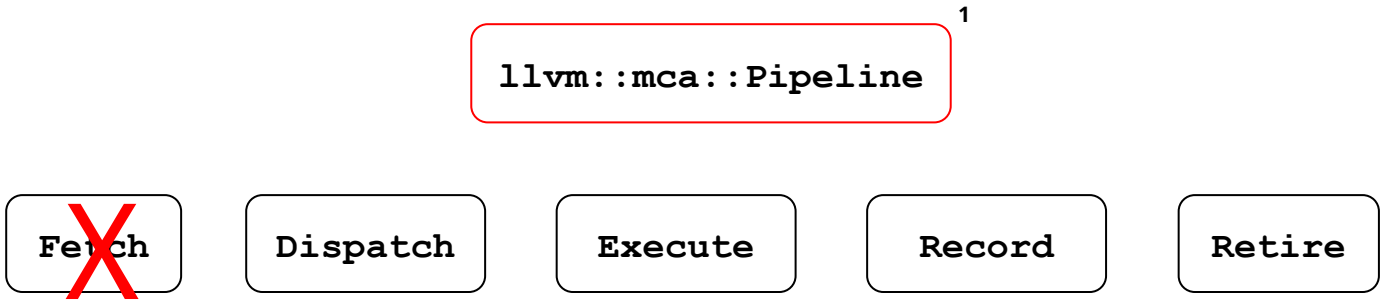
Mechanics: Pipelining in llvm-mca



- The fetching stage is assumed by llvm-mca (no front-end)
- Dispatches occur in groups to simulated schedulers based on the dispatch width
- Recording occurs towards the end of an instruction's execution
  - Known as the "write back stage"

[1] Andrea di Biagio, Matt Davis. "Understanding the Performance of Code Using llvm-mca," 2018 LLVM Developers' Meeting.

Mechanics: Pipelining in llvm-mca

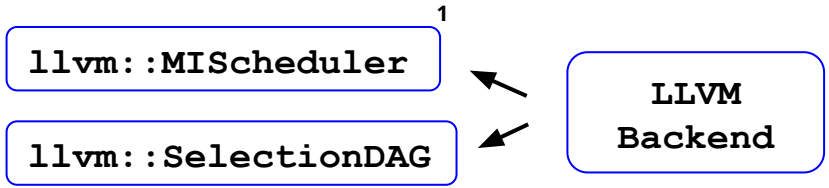


- Schedulers are simulated based on specific architecture / CPU
- Memory instructions are performed speculatively using a simulated LSUnit

- Other simulated units:
  - Register File Unit
  - Retire Control Unit

[1] Andrea di Biagio, Matt Davis. “Understanding the Performance of Code Using llvm-mca,” 2018 LLVM Developers’ Meeting.

Mechanics: LLVM Backend and Scheduling Models



- llvm-mca models *out-of-order scheduling* and *execution*
- LLVM API exists to select instructions and schedule micro-ops “out-of-order”
- This LLVM API works with llvm-mca to generate a simulated scheduler that resembles the given arch.

[1] Dave Estes “SchedMachineModel: Adding and Optimizing a Subtarget,” (2014), Qualcomm. https://llvm.org/devmtg/2014-10/Slides/Estes-MISchedulerTutorial.pdf

Agenda

1. Background
2. Mechanics of llvm-mca
3. **Using llvm-mca**
4. Potential applications for current projects

Using llvm-mca: Example

Example function **foo**:

```
double foo(double a)
{
  return (a + a) / (a * a);
}
```

Assembly generated (-O3, skylake)

```
vaddsd  %xmm0, %xmm0, %xmm1
vmulsd  %xmm0, %xmm0, %xmm0
vdivsd  %xmm0, %xmm1, %xmm0
retq
```

llvm-mca is available on Godbolt's Compiler Explorer: https://bit.ly/2Y685oQ

Using llvm-mca: Command Line

- Default flags includes the following: **-instruction-info, -resource-pressure, -summary-view**

```
llvm-mca -mcpu=skylake foo.S
```

- We'll explore these performance statistics/views:

```
llvm-mca -mcpu=skylake -timeline bar.S
```

```
llvm-mca -mcpu=skylake -bottleneck-analysis baz.S
```

Using llvm-mca: Drawbacks

- LSUnit is naive
  - No alias analysis is performed (would be an optimization)
  - Knows “nothing about cache hierarchy” [1]
- Simulation by llvm-mca is not always realistic
  - Simulation over an architecture/CPU not the same as real performance
  - Modeling only out-of-order scheduler leaves out the front-end
- llvm-mca is relatively new
  - Documentation, testing, benchmarks are lacking compared to IACA
  - Some statistics and views (register related) are not clear to developers
  - Sometimes you have to resort to LLVM email threads: https://bit.ly/2Z6O5E1

[1] Andrea di Biagio, LLVM Development. “llvm-mca: a static performance analysis tool”, https://lists.llvm.org/pipermail/llvm-dev/2018-March/121490.html, 2018.

Agenda

1. Background
2. Mechanics of llvm-mca
3. Using llvm-mca
4. **Potential applications for current projects**

Applications for Current Projects (Interweaving)

- How can we use llvm-mca?
  - Fibers --- Measure the performance of existing assembly (existing fibers code/assembly written directly in Nautilus?)
  - Code injection problem (my project) --- call injections may be simple enough to analyze, but llvm-mca could analyze *assembly injections*
- LLVM developers have reworked llvm-mca to be more **modular**
  - New pipeline designs could open possibilities to analyze the front-end (fetching, decoding, etc.)