L41: Lab 3
Micro-architectural implications of IPC

Lecturelet 3

Dr Graeme Jenkinson

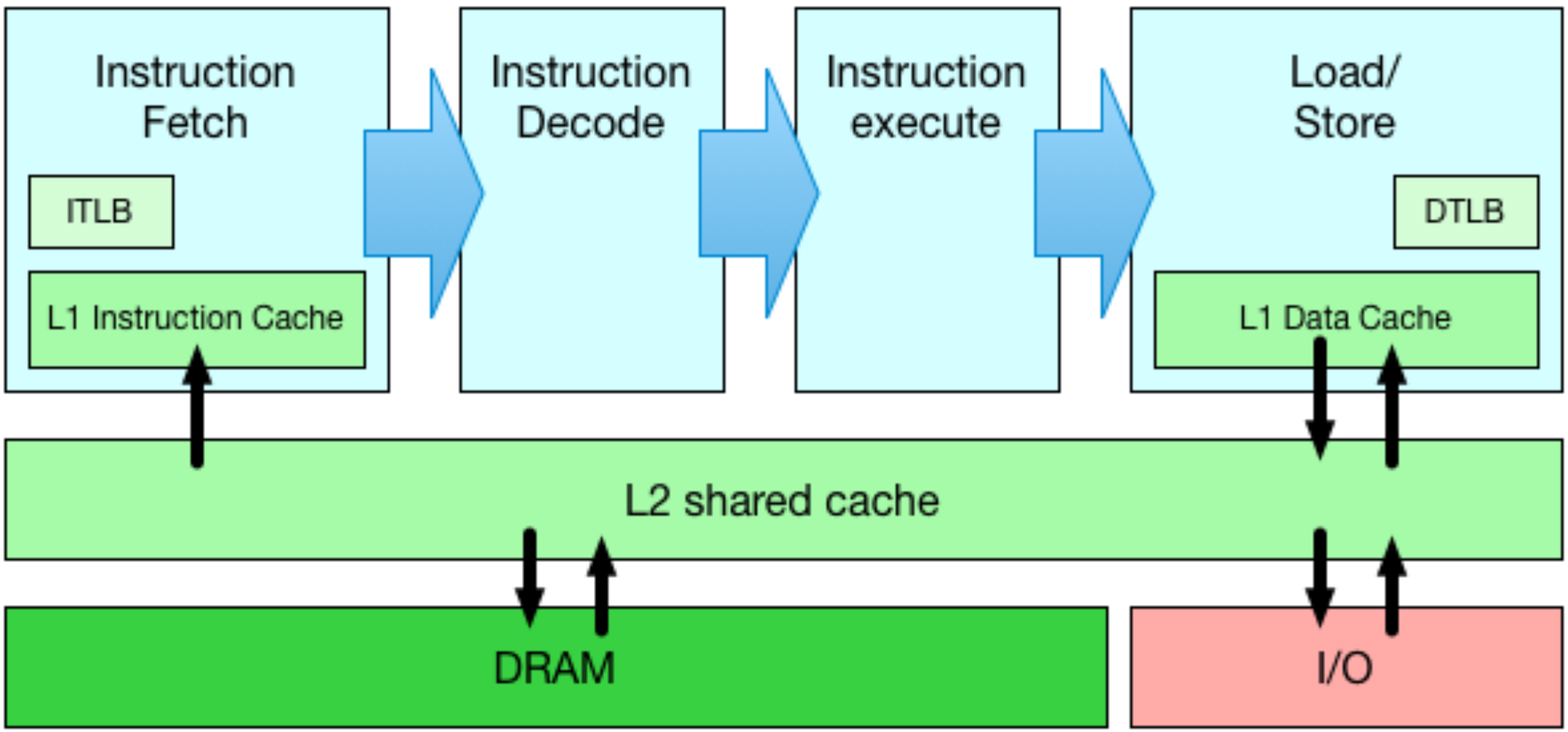
18 February 2019

L41: Lab 3 - Micro-architectural implications of IPC

- Hardware performance counters
- Extending Lab2 from OS effects to architecture/micro-architecture
- Gather further data for assessed **Lab Report 2**

Sketch of ARM Cortex A8 memory hierarchy

- **Architectural** refers to an ISA-level view of execution
- **Micro-architectural** refers to behaviours below the ISA



This is a very, very rough sketch indeed!

Hardware performance counters (1/2)

- Seems simple enough:
  - Source code compiles to instructions
  - Instructions are executed by the processor
- But some instructions take longer than others:
  - Register-register operations generally single-cycle (or less)
  - Multiply and divide may depend on the specific numeric values
  - Floating point may take quite a while
  - Loads/stores cost different amounts depending on TLB/cache use

Hardware performance counters (2/2)

- Optimisation is therefore not just about reducing instruction count
  - Optimisation must take into account micro-architectural effects
  - TLB/cache effects tricky as they vary with memory footprint
  - How can we tell when the cache overflows?
- Hardware performance counters let us directly ask the processor about architectural and micro-architectural events
  - #instructions, #memory accesses, #cache misses, DRAM traffic...

The benchmark – now with PMC

```
root@l41-beaglebone data/ipc # ./ipc-static
ipc-static [-Bqsv] [-b buffersize] [-i pipe|local] [-t totalsize] mode

Modes (pick one - default 1thread):
    1thread            IPC within a single thread
    2thread            IPC between two threads in one process
    2proc              IPC between two threads in two different processes

Optional flags:
    -B                 Run in bare mode: no preparatory activities
    -i pipe|local      Select pipe or socket for IPC (default: pipe)
    -P l1d|l1i|l2|mem|tlb|axi  Enable hardware performance counters
    -q                 Just run the benchmark, don't print stuff out
    -s                 Set send/receive socket-buffer sizes to buffersize
    -v                 Provide a verbose benchmark description
    -b buffersize      Specify a buffer size (default: 131072)
    -t totalsize       Specify total I/O size (default: 16777216)
```

- `-P` argument requests profiling of load/store instructions, L1 D-cache, L1 I-cache, L2 cache, I-TLB, D-TLB, and AXI traffic

Example: Profile memory instructions

```
root@l41-beaglebone:/data/ipc # ./ipc-static -vP mem -b
1048576 -i local 1thread

 Benchmark configuration:
   buffersize: 1048576
   totalsize: 16777216
   blockcount: 16
   mode: 1thread
   ipctype: socket
   time: 0.084140708

   pmctype: mem
   INSTR_EXECUTED: 25463397
   CLOCK_CYCLES: 46233168
   CLOCK_CYCLES/INSTR_EXECUTED: 1.815672
   MEM_READ: 8699699
   MEM_READ/INSTR_EXECUTED: 0.341655
   MEM_READ/CLOCK_CYCLES: 0.188170
   MEM_WRITE: 7815423
   MEM_WRITE/INSTR_EXECUTED: 0.306928
   MEM_WRITE/CLOCK_CYCLES: 0.169044

 194721.45 KBytes/sec
```

Example: Profile memory instructions (1/2)

- Benchmark run pushed 16M data through a socket using 1M buffers for reads and writes
- Reasonable expectation of load and store memory footprints to be 16M $\times 2 + \varepsilon$ reflecting copies to and from kernel buffers
- Memory reads: 8,699,699
- Word size in ARMv7: 32bits
- $8{,}699{,}699 \times 4 \approx 32M$
  - Sum of buffer accesses in user and kernel memory:

Example: Profile memory instructions

- Could now query L1, L2 caches
  - How many of those accesses are in each cache, and how does it affect performance?
- How does L1,L2 cache miss rate relate to cycles/instruction?
- How would DTrace profiling show changed behaviour as cycles/instruction goes u?

Experimental questions for the lab report

- Experimental questions (2/2):
  - How does changing the IPC buffer size affect architectural and micro-architectural memory behaviour – and why?
  - Can we reach causal conclusions about the scalability of pipes vs. sockets from processor performance counters?
- Remember to consider the hypotheses the experimental questions are exploring.
- Ensure that you directly consider the impact of the **probe effect** on your causal investigation.

This lab session

- Use this session to continue to build experience:
  - Ensure that you can use PMC to collect information about the memory subsystem: instructions, cache behaviour, AXI behaviour
  - Continue data collection for the Lab Report2
  - Identify **inflection points** where performance trends change as a result of architectural or micro-architectural thresholds
- Remember to use data from both Lab 2 and Lab 3 to write the lab report.
- Do ask us if you have any questions or need help.