arm

University of Cambridge

Software System Architecture View: Why Concurrency and Memory Models matter?

PART 3

Jahić Jasmin, Jade Alglave

2024-01-17

AGENDA

10:00 Part 1: Fundamental Issues with Concurrency in Embedded Software Systems from Architectural Point of View

11:00

Coffee break

11:30 Part 2: Synchronization in Concurrent Software is an Architectural Decision

12:15

Coffee break

12:25 Part 3: Arm's Memory Model

Q&A

13:00

PART 3

12:25

Memory model

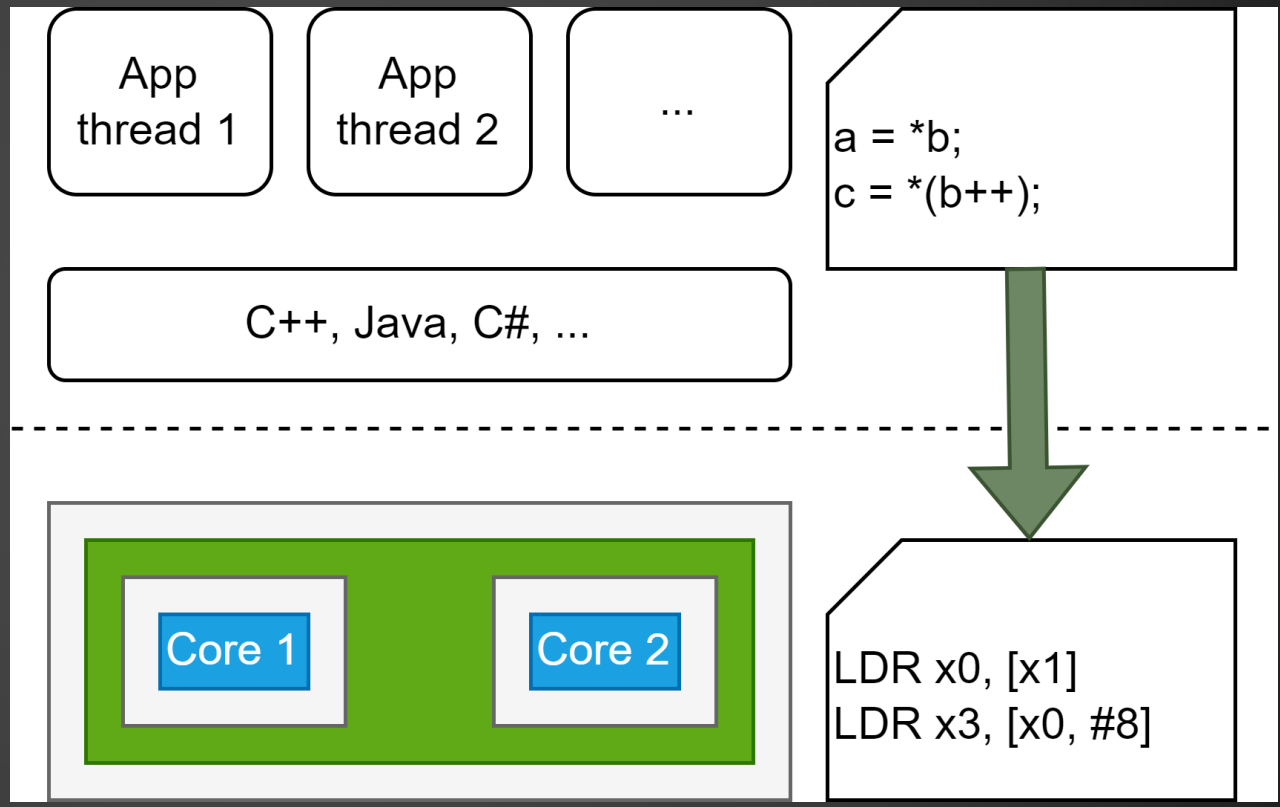
Arm’s memory model and consistency

Concurrency and herd7 tool

Q&A

13:00

MEMORY MODEL



WHAT CAN GO WRONG?

- Reordering of the instructions:
  - Address dependency: the value returned by a read access is used for the computation of the virtual address of a subsequent read or write access.
    - Example (are we allowed to re-order these):
      - LDR x0, [x1]
      - LDR x3, [x0, #8]
  - Control dependency: the data value returned by a read access determines the condition flags, determining the address of a subsequent read access.
- Concurrent access to memory locations
  - Multiple execution flows (e.g., pipeline, threads, processes)
- Which result values are permissible for a read access?

ORDERING OF MEMORY ACCESSES

- Strictly Ordered
  - Denoted by <: must occur strictly in order.
- Ordered
  - Denoted by <=: can occur either in order or simultaneously.
- Example from Arm memory model:
  - If A1 and A2 are generated by two different instructions:
    - A1 < A2 if the instruction that generates A1 occurs before the instruction that generates A2 in program order
    - A2 < A1 if the instruction that generates A2 occurs before the instruction that generates A1 in program order.

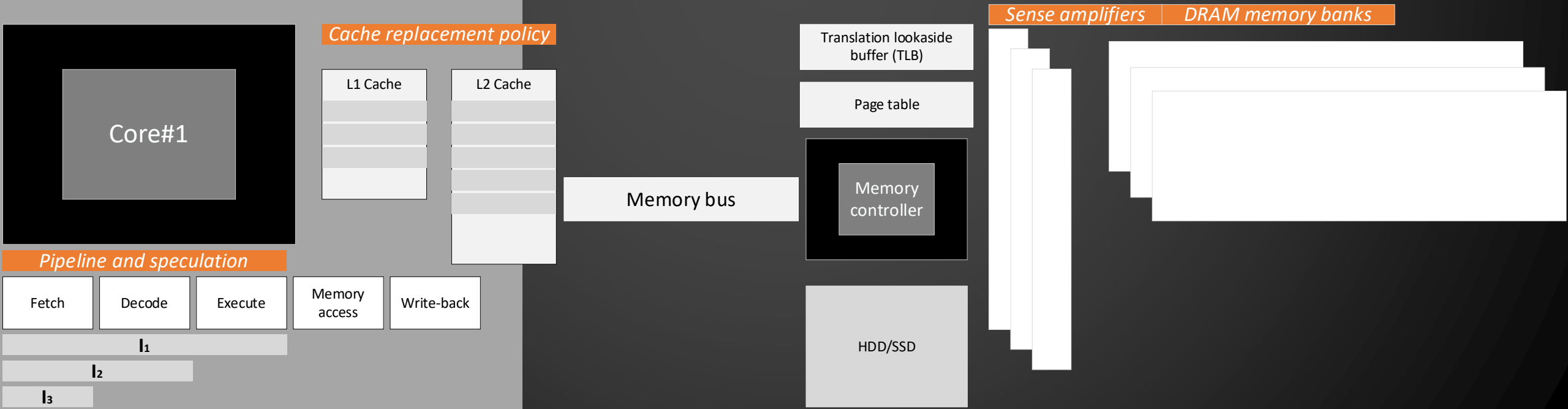
CONSISTENCY AND COHERENCE

- Coherence: value of shared resource data stored in multiple locations.

- Consistency: the order of execution of memory operations – how they appear to execute with respect to one another.
  - Strong consistency
  - Weak consistency

ISSUES HIDDEN FROM DEVELOPERS

- Intermediate states
- Virtual memory vs physical memory
- Caches
- Page tables mapping from virtual addresses to physical locations
- A translation lookaside buffer (TLB)



MEMORY MODEL

- Generating an exception on an unaligned memory access
- Restricting access by applications to specified areas of memory
- Translating virtual addresses provided by executing instructions into physical addresses
- Controlling the order of accesses to memory
  - Memory model gives ordering requirements over memory accesses, intuitively stating which local orderings must be respected by hardware.
- Synchronizing access to shared memory by multiple processors

MEMORY MODEL

- Instructions and their logic
  - Logic that includes synchronisation

LDR INSTRUCTION

```
if ConditionPassed() then
    EncodingSpecificOperations();
    offset_addr = if add then (R[n] + imm32) else (R[n] - imm32);
    address = if index then offset_addr else R[n];

    // Determine if the stack pointer limit should be checked
    if n == 13 && wback then
        violatesLimit = ViolatesSPLim(LookUpSP(), offset_addr);
    else
        violatesLimit = FALSE;
    // Memory operation only performed if limit not violated
    if !violatesLimit then
```

MEMORY MODEL

- Instructions and their logic
  - Logic that includes synchronisation
- Memory barrier is the general term applied to an instruction, or sequence of instructions, that forces synchronization events by a processor with respect to retiring load/store instructions:
  - ordering of load/store instructions
  - completion of load/store instructions
  - context synchronization.

MEMORY BARRIERS IN ARM

- „A memory barrier is an instruction that requires the core to apply an ordering constraint between memory operations that occur before and after the memory barrier instruction in the program.“
- Data Memory Barrier
- Data Synchronization Barrier
- Instruction Synchronization Barrier
  - „Flushes the pipeline in the processor, so that all instructions that come after the ISB instruction in program order are fetched from cache or memory only after the ISB instruction has completed.“

ARM‘S MEMORY MODEL

- Memory model in a language called cat
- A formally defined semantics
- A precise mathematical meaning
- https://developer.arm.com/architectures/cpu-architecture/a-profile/memory-model-tool

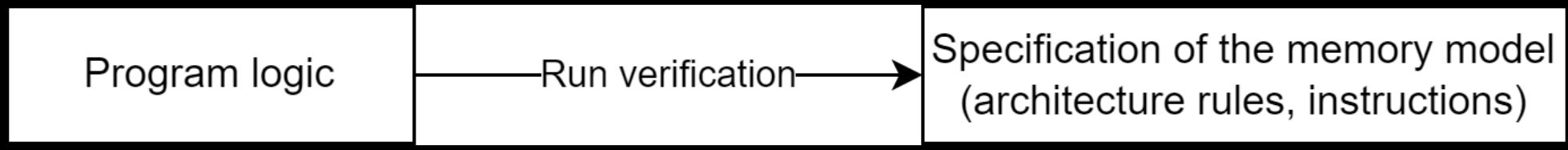
TESTING AGAINST A MEMORY MODEL

- Rules defined on the architectural level
  - Does the execution comply with the memory model?
- Freedom from concurrency bugs – formal verification?

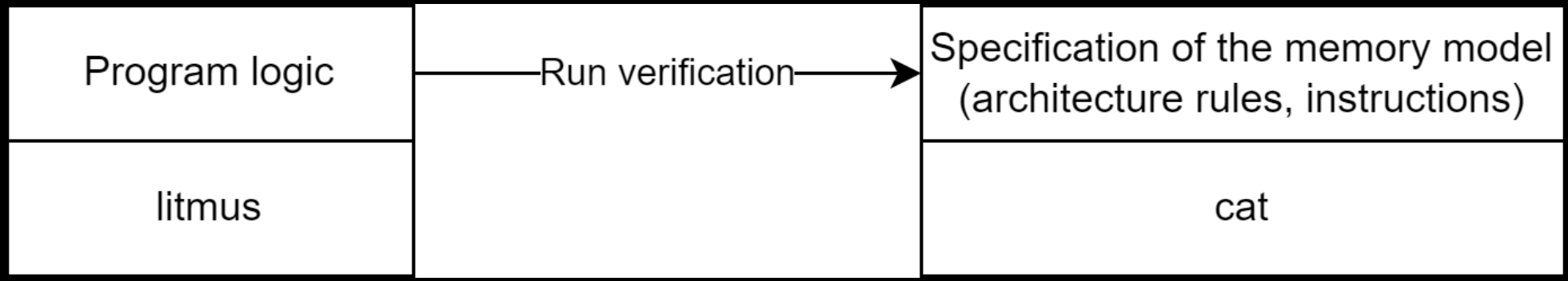
- „The memory consistency model of a shared-memory multiprocessor provides a formal specification of how the memory system will appear to the programmer, eliminating the gap between the behavior expected by the programmer and the actual behavior supported by a system.“

Shared Memory Consistency Models: A Tutorial Sarita V. Adve Kourosh Gharachorloo

TESTING MEMORY MODEL



TESTING MEMORY MODEL



https://developer.arm.com/herd7

TESTING MEMORY MODEL

tests/MP.litmus

```
AArch64 MP
"PodWW Rfe PodRR Fre"
Cycle=Rfe PodRR Fre PodWW
Generator=diycross7 (version 7.54+01(dev))
Prefetch=0:x=F,0:y=W,1:y=F,1:x=T
Com=Rf Fr
Orig=PodWW Rfe PodRR Fre
{
0:X1=x; 0:X3=y;
1:X1=y; 1:X3=x;
}
 P0          | P1          ;
 MOV W0,#1   | LDR W0,[X1] ;
 STR W0,[X1] | LDR W2,[X3] ;
 MOV W2,#1   |             ;
 STR W2,[X3] |             ;
```

cats/aarch64.cat

```
46
47  catdep (* This option says that the cat file computes dependencies *)
48
49  include "aarch64hwreqs.cat"
50
51  (*** Coherence-after ***)
52  let ca = fr | co
53
54  (*** TLBI-after, DC-after, IC-after ***)
55  include "enumerations.cat"
56  with TLBI-after from (all-TLBI-Imp_TTD_R-enums local-hw-reqs)
57  with DC-after from (all-DC-Exp_W-enums local-hw-reqs)
58  with IC-after from (all-IC-Imp_Instr_R-enums local-hw-reqs)
59
60  (*** Hazard-ordered-before ***)
61
```

https://developer.arm.com/herd7

A generic simulator for weak memory models: https://github.com/herd/herdtools7

EXAMPLE WITH HERD7 – MESSAGE PASSING WITH A FLAG

```
AArch64 MP
{
0:X1=x; 0:X3=y;
1:X1=y; 1:X3=x;
}
P0              | P1          ;
MOV W0,#1       | LDR W0,[X1] ;
STR W0,[X1]     | LDR W2,[X3] ;
MOV W2,#1       |             ;
STR W2,[X3]     |             ;
exists (1:X0=1 /\ 1:X2=0)
```

The thread P0 writes 1 to memory location x, and 1 to memory location y.

The thread P1 reads from y and places the result into register W0 and reads from x and places the result into register W2. The registers W0 and W2 are private to P1.

Essentially, P0 writes a message in x, then sets up a flag in y, so that when P1 sees the flag (via its read from y), it can read the message in x.

At the bottom of the test, we ask "is there an execution of this test such that register W0 contains the value 1 and register W2 contains the value 0?".

https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/how-to-use-the-memory-model-tool

EXAMPLE WITH HERD7 – MESSAGE PASSING WITH A FLAG

```
AArch64 MP
{
0:X1=x; 0:X3=y;
1:X1=y; 1:X3=x;
}
P0              | P1          ;
MOV W0,#1       | LDR W0,[X1] ;
STR W0,[X1]     | LDR W2,[X3] ;
MOV W2,#1       |             ;
STR W2,[X3]     |             ;
exists (1:X0=1 /\ 1:X2=0)
```

The thread P0 writes 1 to memory location x, and 1 to memory location y.

The thread P1 reads from y and places the result into register W0 and reads from x and places the result into register W2. The registers W0 and W2 are private to P1.

Essentially, P0 writes a message in x, then sets up a flag in y, so that when P1 sees the flag (via its read from y), it can read the message in x.

At the bottom of the test, we ask "is there an execution of this test such that register W0 contains the value 1 and register W2 contains the value 0?".

Results:

1:X0=0; 1:X2=0;

1:X0=0; 1:X2=1;

1:X0=1; 1:X2=0;

1:X0=1; 1:X2=1;

https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/how-to-use-the-memory-model-tool

ARM EDUCATION & ACADEMIC ENGAGEMENTS: VISION, MISSION AND STRATEGY

Vision

- To play a **leading role** in plugging the growing **education, research and skills gap/mismatch** in **computing and STEM**

Mission

- Enable **educators, researchers and learners** at large to **harness state-of-the-art Arm and Arm-based technologies** to **learn, innovate** and **compete** effectively in the **modern economy**

Strategy

- Channel marketing based strategy, working in partnership, **intervening directly only when necessary**
- Serve the scholarship with low-cost, **curriculum-aligned** engaging educational and research materials, IP, tools, methodologies, platforms and insights
- Constantly **measure our performance** based on the requirements of our stakeholders, external benchmarks, and RoI to Arm

RESEARCH ENABLEMENT

Arm technology provides the best platform for academics to excel

Objective:

- Enable academics to increase their research impact with/on Arm

How:

- Provide access on a truly global basis to a wide range of commercially proven Arm IP/tools through a number appropriately tailored offers, all at zero upfront and ongoing cost
- Identify opportunities for hub-and-spoke arrangements that best suit local circumstances
- Support through enabling communities of practice

Benefits:

- Accelerate time to results – Arm IP forms the centerpiece of a robust and vibrant eco-system
- Demonstrate real-world relevance of academic conclusions
- Capitalize on new trends/innovations

AGENDA

10:00 Part 1: Fundamental Issues with Concurrency in Embedded Software Systems from Architectural Point of View

11:00 Coffee break

11:30 Part 2: Synchronization in Concurrent Software is an Architectural Decision

12:15 Coffee break

12:25 Part 3: Arm's Memory Model

Q&A

13:00

- Jasmin.jahic@gmail.com
- jj542@cam.ac.uk
- Jasmin.jahic@arm.com