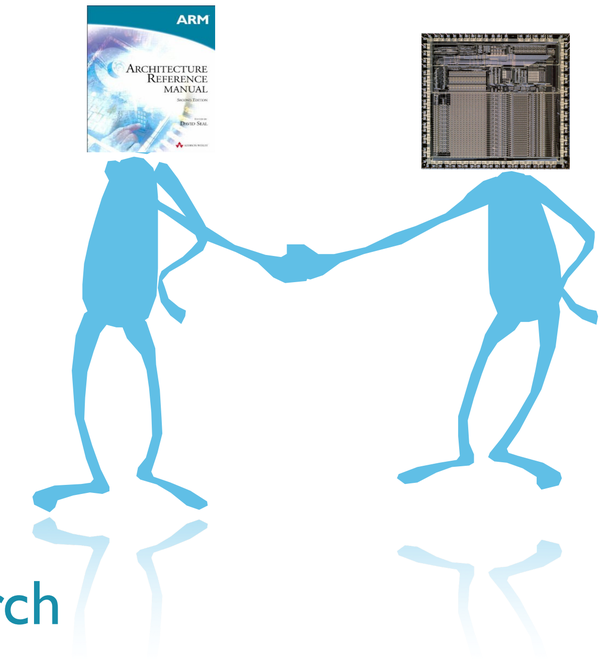
End-to-End Verification of ARM® Processors with ISA-Formal

Alastair Reid, Rick Chen, Anastasios Deligiannis, David Gilday, David Hoyes, Will Keen, Ashan Pathirane, Owen Shepherd, Peter Vrabel, Ali Zaidi



alastair.reid@arm.com
@alastair_d_reid

ARMResearch

The Architecture for the Digital World®

ARM

Scale

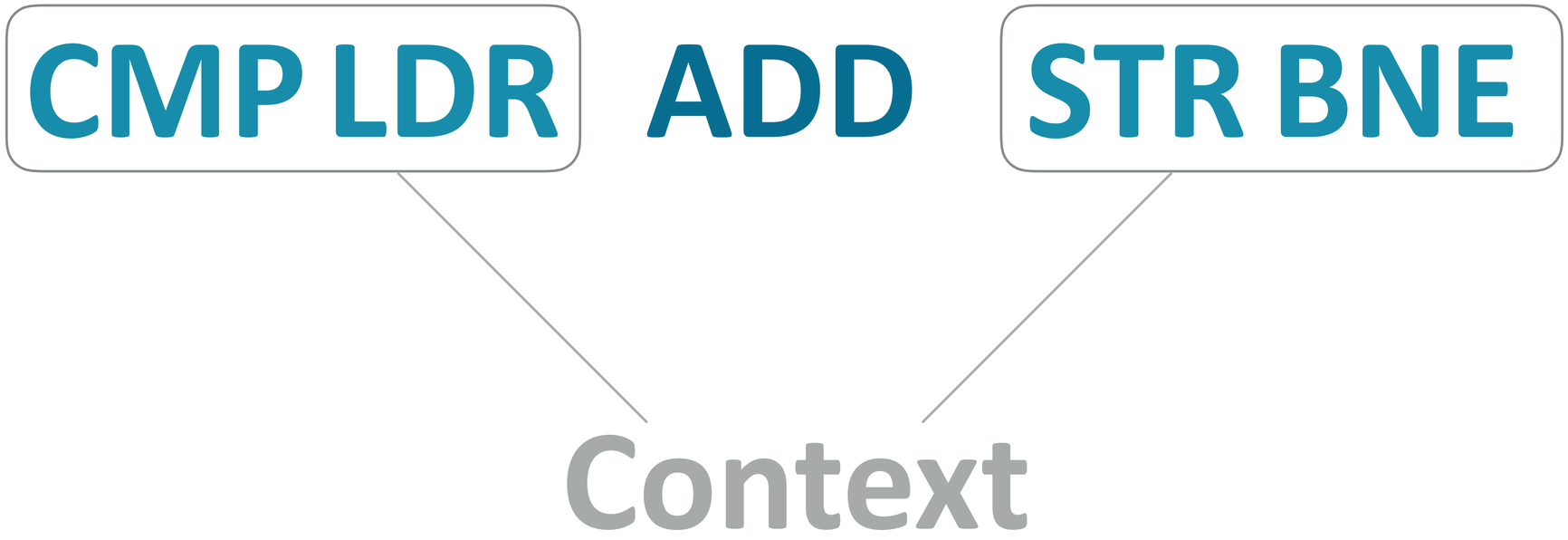
Large Specifications

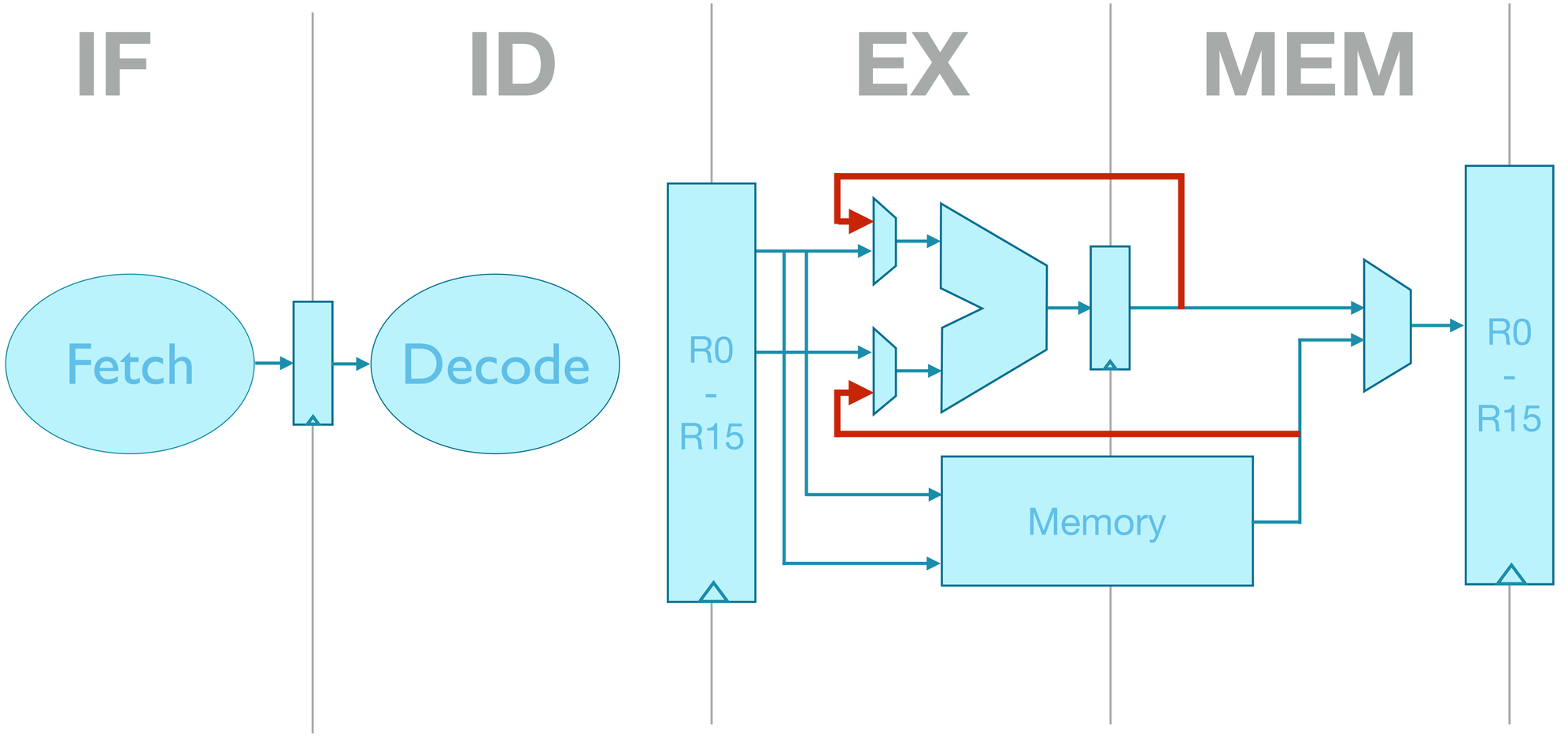
Large Implementations

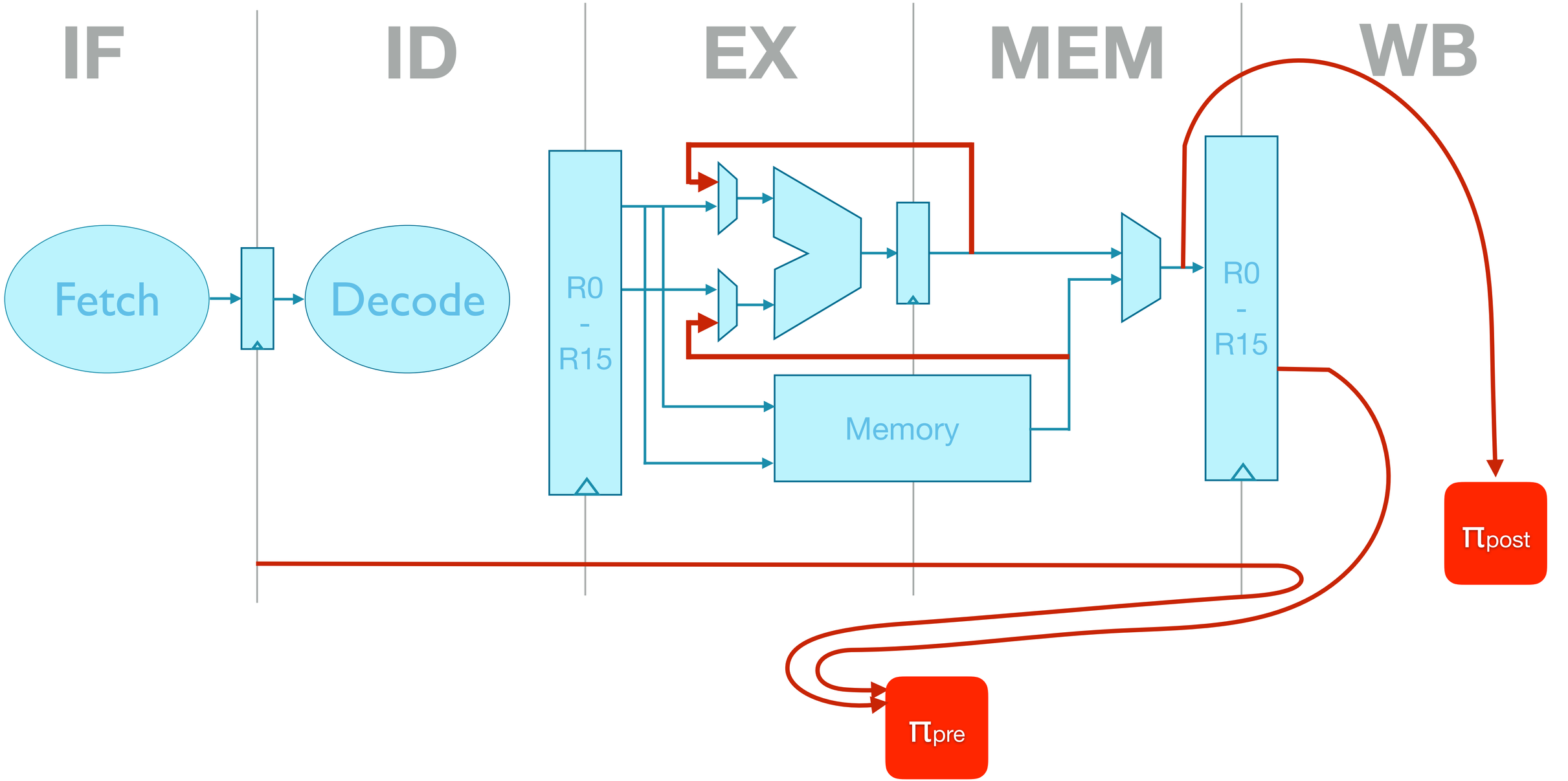
Checking an instruction

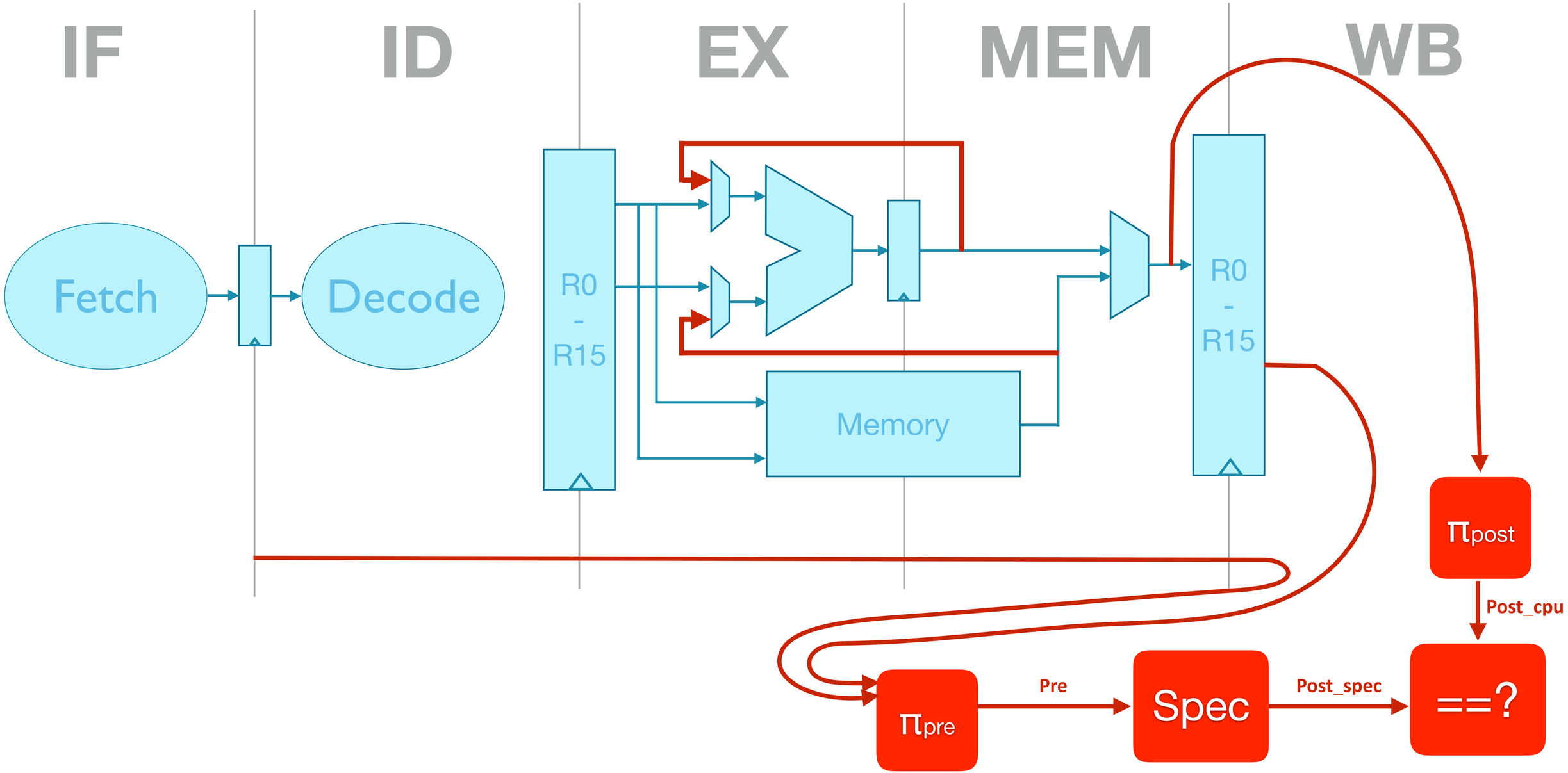
ADD

Checking an instruction




IF
ID
EX
MEM
WB
Fetch
Decode
R0
-
R15
Memory
R0
-
R15


IF
ID
EX
MEM
WB
Fetch
Decode
R0
-
R15
Memory
R0
-
R15
πpost
πpre


IF
ID
EX
MEM
WB
Fetch
Decode
R0
-
R15
Memory
R0
-
R15
πpost
Post_cpu
==?
Post_spec
Spec
Pre
πpre

Errors ISA-Formal can catch

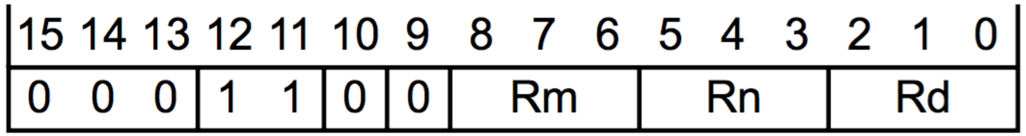
No Context

- Errors in decode
- Errors in data path

Context

- Errors in forwarding logic
- Errors in register renaming
- Errors in exception handling
- Errors in speculative execution

Specifying ADD

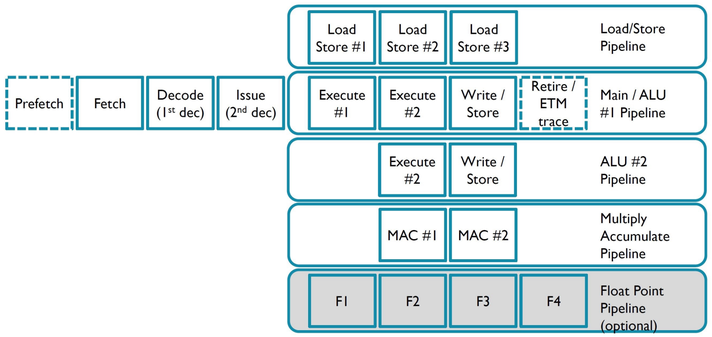


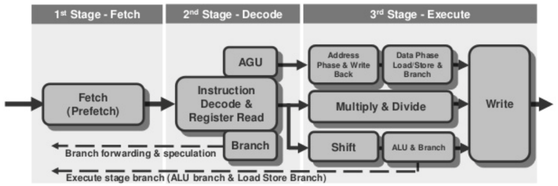
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	1	1	0	0	Rm			Rn			Rd		

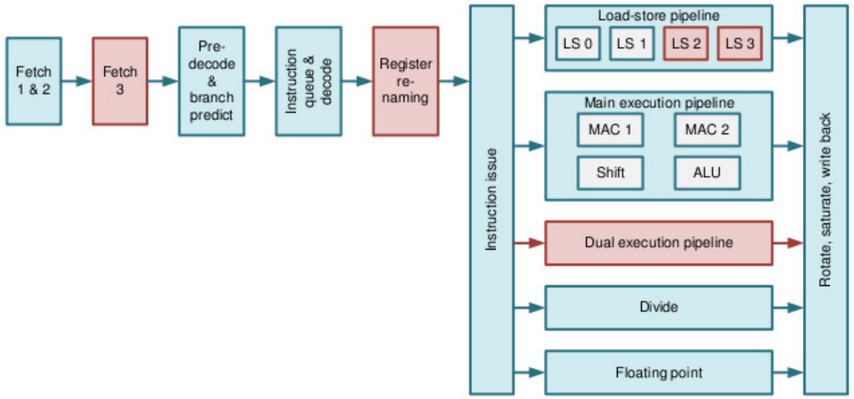
```
assign ADD_retiring = (pre.opcode & 16'b1111_1110_0000_0000)
                                  == 16'b0001_1000_0000_0000;
assign ADD_result   = pre.R[pre.opcode[8:6]] + pre.R[pre.opcode[5:3]];
assign ADD_Rd       = pre.opcode[2:0];

assert property (@(posedge clk) disable iff (~reset_n)
      ADD_retiring |-> (ADD_result == post.R[ADD_Rd]));
```

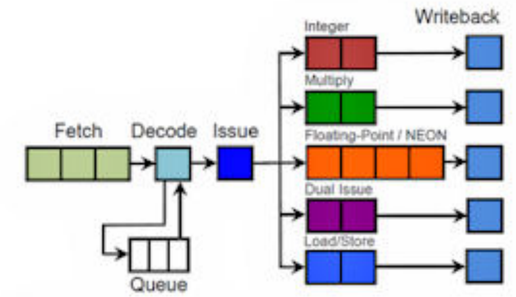
ISA Formal

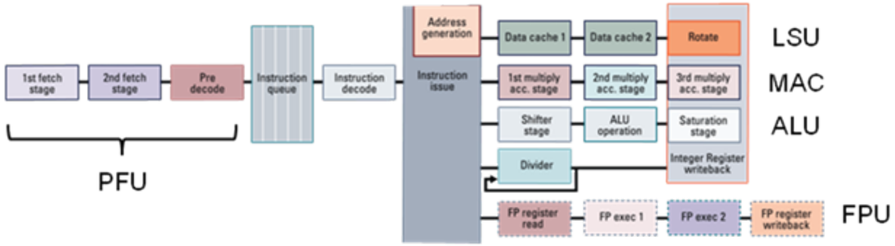
- Finds complex bugs in processor pipelines
- Applied to wide range of μArchitectures
- Uses translation of ARM's internal ISA specification

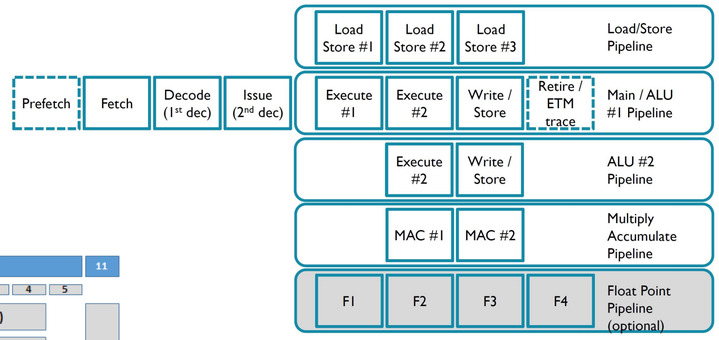

Prefetch
Fetch
Decode (1st dec)
Issue (2nd dec)
Load Store #1
Load Store #2
Load Store #3
Load/Store Pipeline
Execute #1
Execute #2
Write / Store
Retire / ETM trace
Main / ALU #1 Pipeline
Execute #2
Write / Store
ALU #2 Pipeline
MAC #1
MAC #2
Multiply Accumulate Pipeline
F1
F2
F3
F4
Float Point Pipeline (optional)

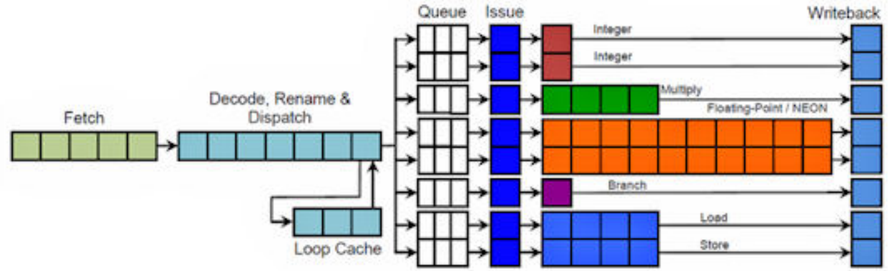

1st Stage - Fetch
2nd Stage - Decode
3rd Stage - Execute
Fetch (Prefetch)
Instruction Decode & Register Read
AGU
Branch
Address Phase & Write Back
Data Phase Load/Store & Branch
Multiply & Divide
Shift
ALU & Branch
Write
Branch forwarding & speculation
Execute stage branch (ALU branch & Load Store Branch)

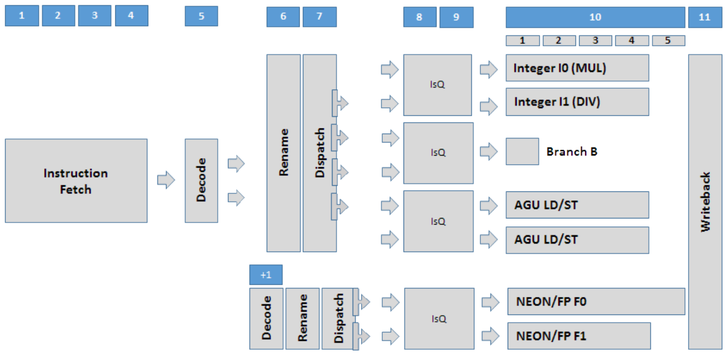

Fetch 1 & 2
Fetch 3
Pre-decode & branch predict
Instruction queue & decode
Register re-naming
Instruction issue
Load-store pipeline
LS 0
LS 1
LS 2
LS 3
Main execution pipeline
MAC 1
MAC 2
Shift
ALU
Dual execution pipeline
Divide
Floating point
Rotate, saturate, write back

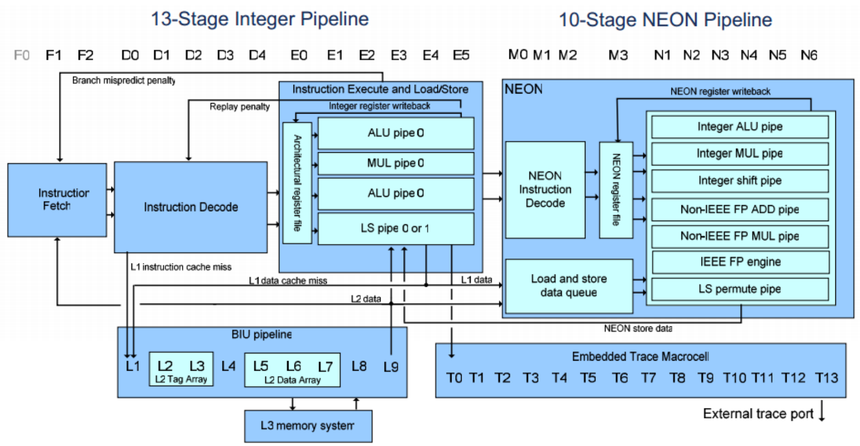

Writeback
Integer
Multiply
Floating-Point / NEON
Dual Issue
Load/Store
Fetch
Decode
Issue
Queue


1st fetch stage
2nd fetch stage
Pre decode
PFU
Instruction queue
Instruction decode
Address generation
Instruction issue
Data cache 1
Data cache 2
Rotate
1st multiply acc. stage
2nd multiply acc. stage
3rd multiply acc. stage
Shifter stage
ALU operation
Saturation stage
Divider
Integer Register writeback
FP register read
FP exec 1
FP exec 2
FP register writeback
LSU
MAC
ALU
FPU


Load Store #1
Load Store #2
Load Store #3
Load/Store Pipeline
Prefetch
Fetch
Decode (1st dec)
Issue (2nd dec)
Execute #1
Execute #2
Write / Store
Retire / ETM trace
Main / ALU #1 Pipeline
ALU #2 Pipeline
MAC #1
MAC #2
Multiply Accumulate Pipeline
F1
F2
F3
F4
Float Point Pipeline (optional)

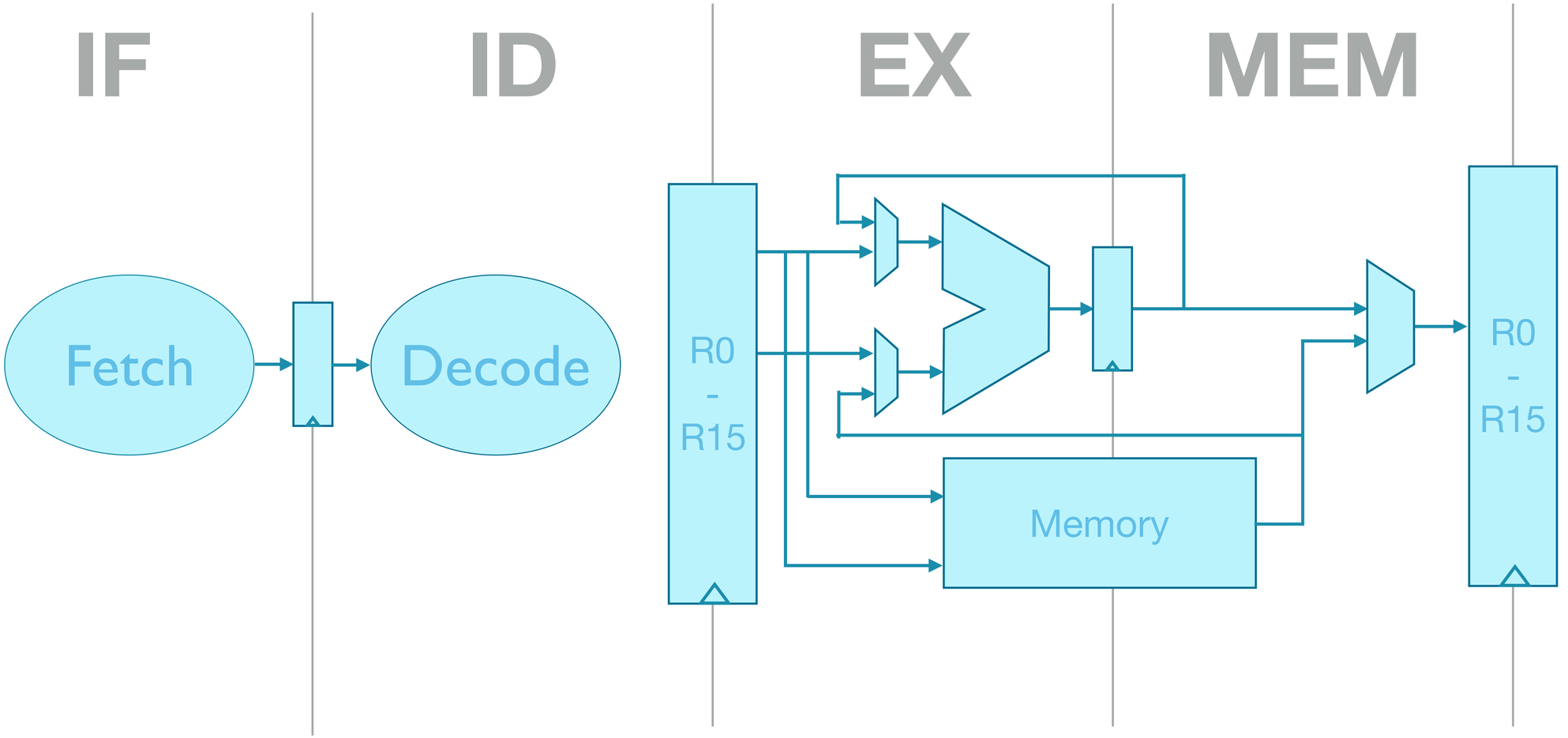

Queue
Issue
Writeback
Fetch
Decode, Rename & Dispatch
Loop Cache
Integer
Integer
Multiply
Floating-Point / NEON
Branch
Load
Store

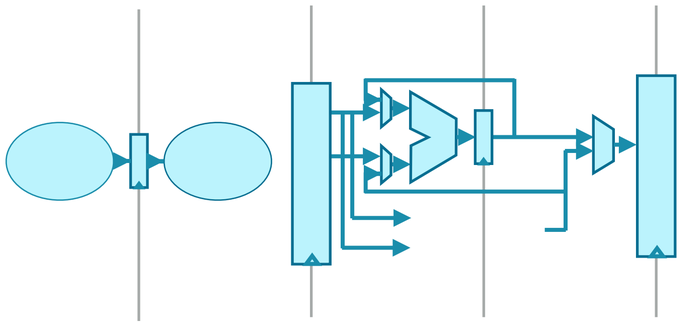

1
2
3
4
5
6
7
8
9
10
11
Instruction Fetch
Decode
Rename
Dispatch
IsQ
Integer I0 (MUL)
Integer I1 (DIV)
Branch B
AGU LD/ST
AGU LD/ST
NEON/FP F0
NEON/FP F1
Writeback
+1

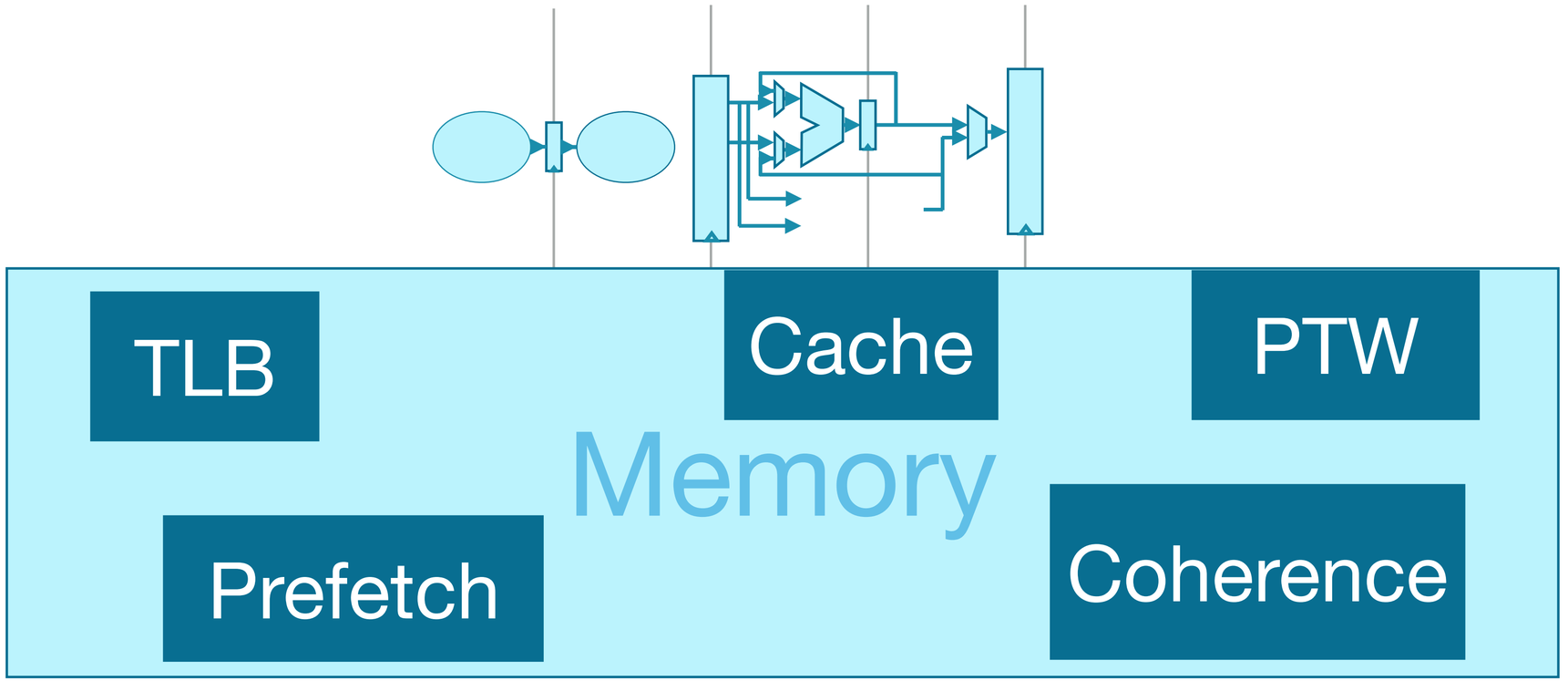

13-Stage Integer Pipeline
10-Stage NEON Pipeline
F0 F1 F2 D0 D1 D2 D3 D4 E0 E1 E2 E3 E4 E5
M0 M1 M2 M3 N1 N2 N3 N4 N5 N6
Branch mispredict penalty
Replay penalty
Instruction Fetch
Instruction Decode
Instruction Execute and Load/Store
Integer register writeback
Architectural register file
ALU pipe 0
MUL pipe 0
ALU pipe 0
LS pipe 0 or 1
L1 instruction cache miss
L1 data cache miss
L1 data
L2 data
NEON
NEON register writeback
NEON Instruction Decode
NEON register file
Integer ALU pipe
Integer MUL pipe
Integer shift pipe
Non-IEEE FP ADD pipe
Non-IEEE FP MUL pipe
IEEE FP engine
LS permute pipe
Load and store data queue
NEON store data
BIU pipeline
L1
L2 L3
L2 Tag Array
L4
L5 L6 L7
L2 Data Array
L8 L9
L3 memory system
Embedded Trace Macrocell
T0 T1 T2 T3 T4 T5 T6 T7 T8 T9 T10 T11 T12 T13
External trace port

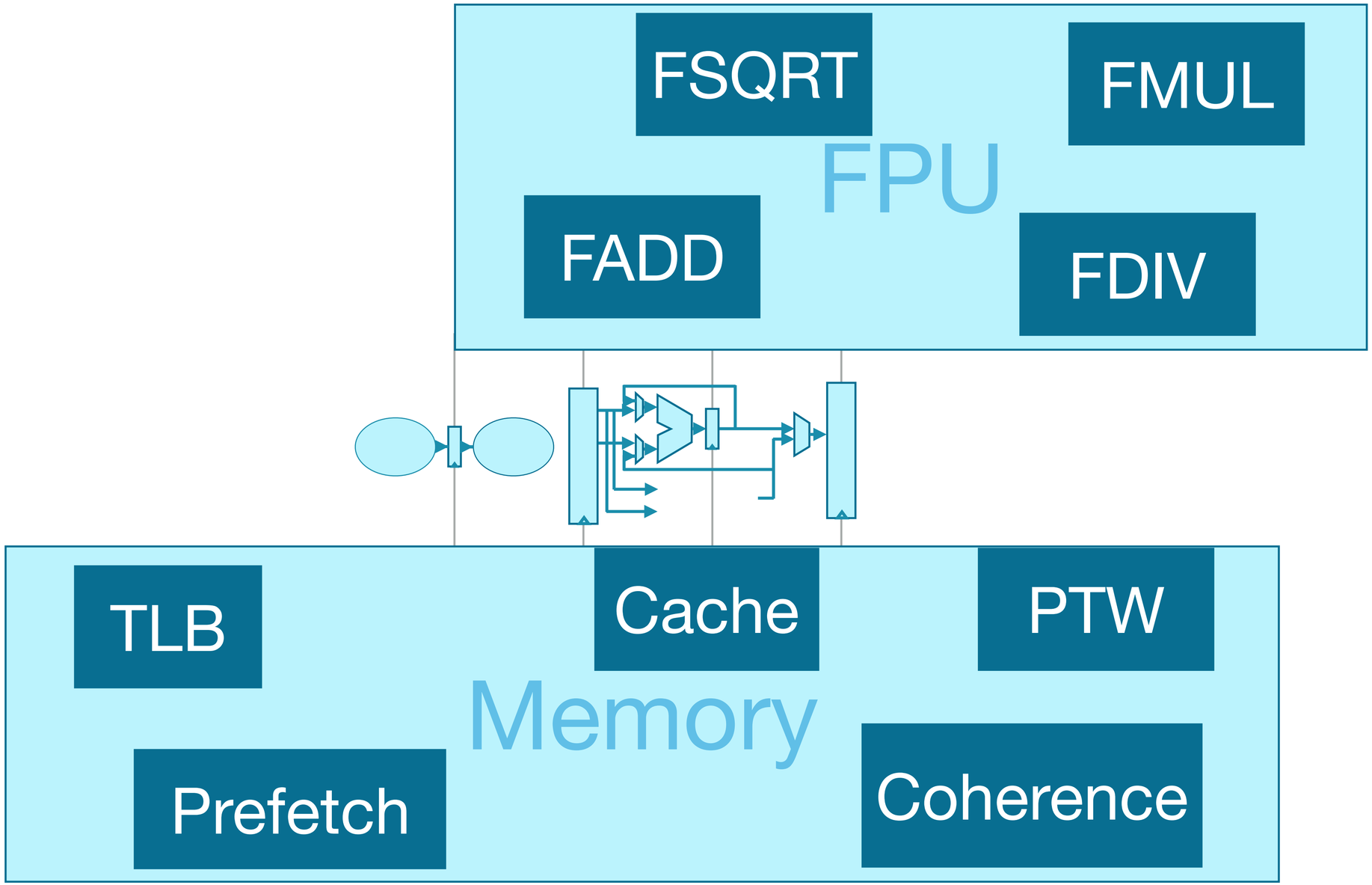
Challenges

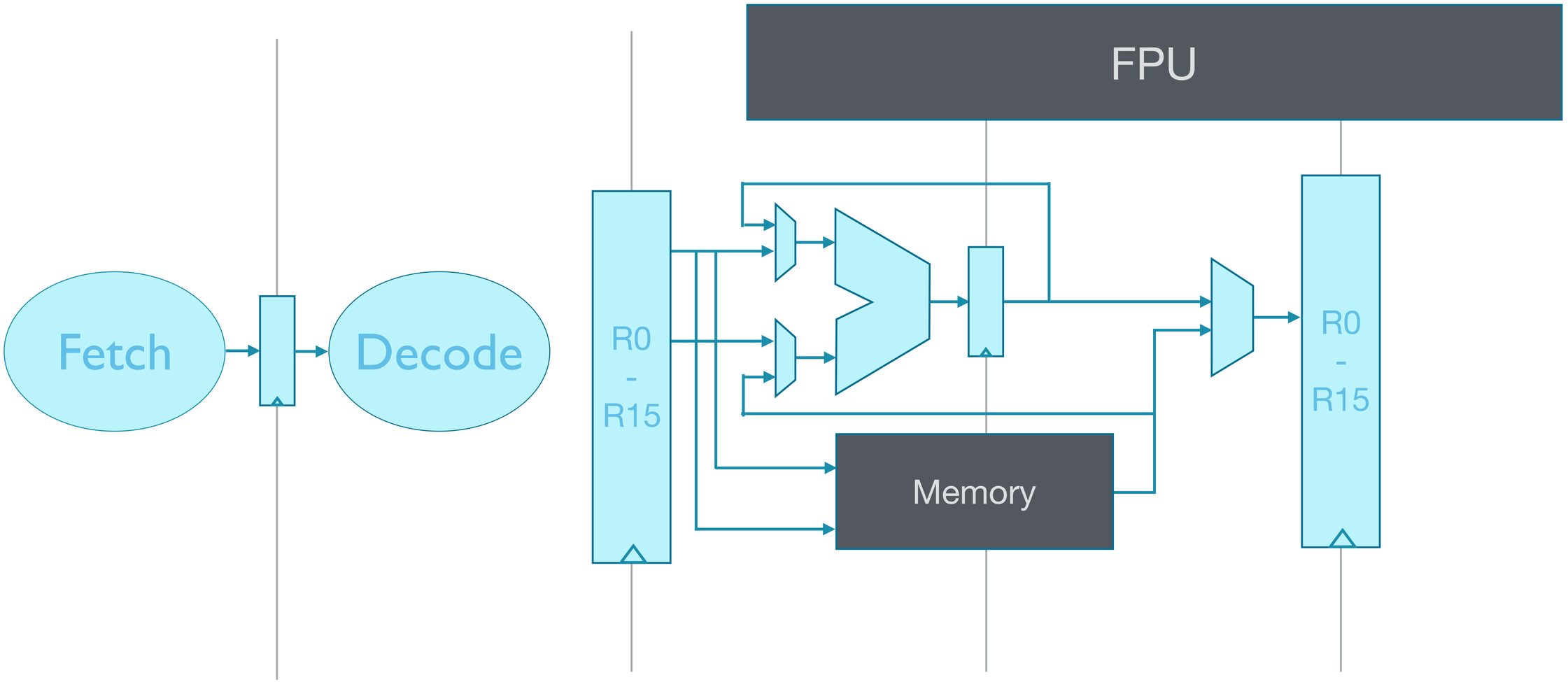
- Complex Functional Units
  - FP
  - Memory
- Dual Issue
- Instruction Fusion
- Register Renaming
- Out-of-order Retire


IF
ID
EX
MEM
WB
Fetch
Decode
R0
-
R15
Memory
R0
-
R15




TLB
Cache
PTW
Memory
Prefetch
Coherence


FSQRT
FMUL
FPU
FADD
FDIV
TLB
Cache
PTW
Memory
Prefetch
Coherence


FPU
Fetch
Decode
R0
-
R15
Memory
R0
-
R15

FP Subset Behaviour

FPAdd	-∞	-1	0	1	∞
-∞	-∞	-∞	-∞	-∞	
-1	-∞		-1	0	∞
0	-∞	-1	0	1	∞
1	-∞	0	1		∞
∞		∞	∞	∞	∞

ISA Formal

- Finds complex bugs in processor pipelines
- Applied to wide range of µArchitectures
- Uses translation of ARM's internal ISA specification

ISA-Formal Properties

	ADC	ADD	B	…	YIELD
R[]		✔			
NZCV					
SP					
PC					
S[],D[],V[]					
FPSR					
MemRead					
MemWrite					
SysRegRW					
ELR					
ESR					
…					

ISA-Formal Properties

	ADC	ADD	B	…	YIELD
R[]		✔			
NZCV					
SP		✔			
PC					
S[],D[],V[]					
FPSR					
MemRead					
MemWrite					
SysRegRW					
ELR					
ESR					
…					

ISA-Formal Properties

	ADC	ADD	B	…	YIELD
R[]		✔	✔		
NZCV					
SP		✔			
PC			✔		
S[],D[],V[]					
FPSR					
MemRead					
MemWrite					
SysRegRW					
ELR					
ESR					
…					

ISA-Formal Properties

	ADC	ADD	B	…	YIELD
R[]	✔	✔	✔		
NZCV	✔				
SP	✔	✔			
PC			✔		
S[],D[],V[]					
FPSR					
MemRead					
MemWrite					
SysRegRW					
ELR					
ESR					
…					

But this is slow and inconsistent

ISA-Formal Properties

	ADC	ADD	B	…	YIELD
R[]	✔	✔	✔	✔	✔
NZCV					
SP					
PC					
S[],D[],V[]					
FPSR					
MemRead					
MemWrite					
SysRegRW					
ELR					
ESR					
…					

ISA-Formal Properties

	ADC	ADD	B	…	YIELD
R[]	✔	✔	✔	✔	✔
NZCV	✔	✔	✔	✔	✔
SP	✔	✔	✔	✔	✔
PC	✔	✔	✔	✔	✔
S[],D[],V[]					
FPSR					
MemRead					
MemWrite					
SysRegRW					
ELR					
ESR					
…					

ISA-Formal Properties

	ADC	ADD	B	…	YIELD
R[]	✔	✔	✔	✔	✔
NZCV	✔	✔	✔	✔	✔
SP	✔	✔	✔	✔	✔
PC	✔	✔	✔	✔	✔
S[],D[],V[]	✔	✔	✔	✔	✔
FPSR	✔	✔	✔	✔	✔
MemRead					
MemWrite					
SysRegRW					
ELR					
ESR					
…					

ISA-Formal Properties

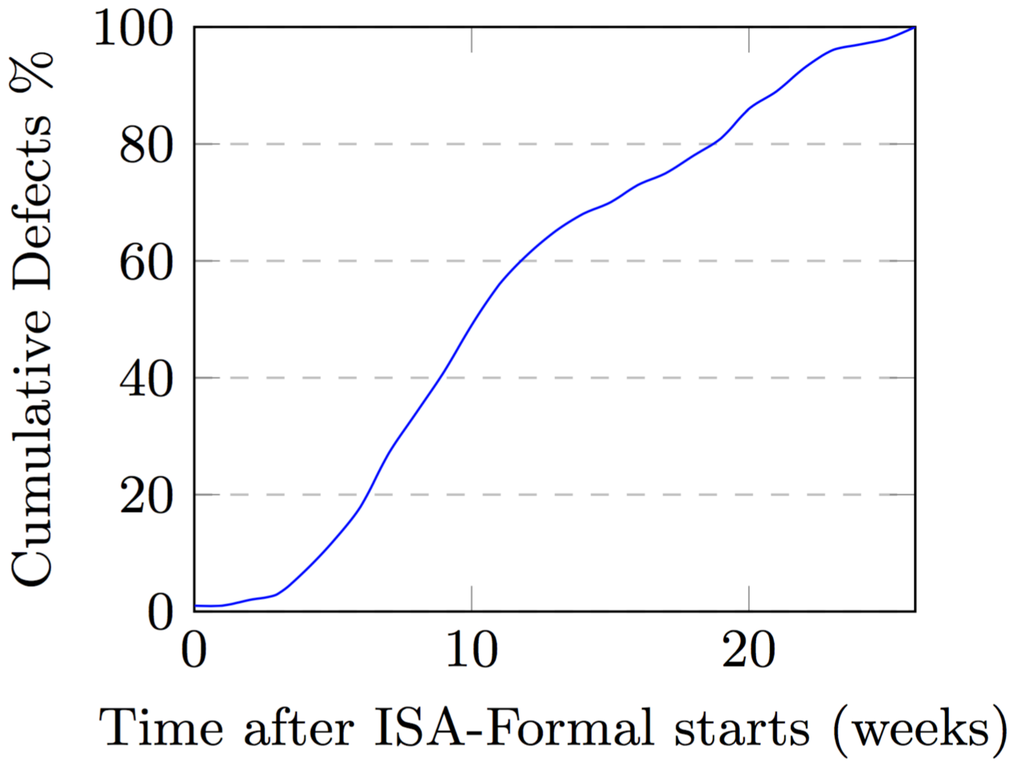
	ADC	ADD	B	…	YIELD
R[]	✔	✔	✔	✔	✔
NZCV	✔	✔	✔	✔	✔
SP	✔	✔	✔	✔	✔
PC	✔	✔	✔	✔	✔
S[],D[],V[]	✔	✔	✔	✔	✔
FPSR	✔	✔	✔	✔	✔
MemRead	✔	✔	✔	✔	✔
MemWrite	✔	✔	✔	✔	✔
SysRegRW					
ELR					
ESR					
…					

ISA-Formal Properties

	ADC	ADD	B	…	YIELD
R[]	✔	✔	✔	✔	✔
NZCV	✔	✔	✔	✔	✔
SP	✔	✔	✔	✔	✔
PC	✔	✔	✔	✔	✔
S[],D[],V[]	✔	✔	✔	✔	✔
FPSR	✔	✔	✔	✔	✔
MemRead	✔	✔	✔	✔	✔
MemWrite	✔	✔	✔	✔	✔
SysRegRW	✔	✔	✔	✔	✔
ELR	✔	✔	✔	✔	✔
ESR	✔	✔	✔	✔	✔
…					

Automation

Specialize
Monomorphize
Constant Propagation
Width Analysis
Exception Handling
...



FP/SIMD	25%
Memory	21%
Branch	21%
Integer	18%
Exception	8%
System	7%

Defect Detection by Area

Summary

- Finds complex bugs in processor pipelines
  - Complete RTL, not just model
  - Bug absence not being proved (yet)
- Applied to wide range of µArchitectures
  - 3 trials, 6 full deployments.
- Uses translation of ARM's internal ISA specification
  - Public release of ISA spec this fall in collaboration w/ Cambridge University
  - "Trustworthy Specifications of ARM® v8-A and v8-M System Level Architecture," to appear, FMCAD 2016

End

alastair.reid@arm.com
@alastair_d_reid