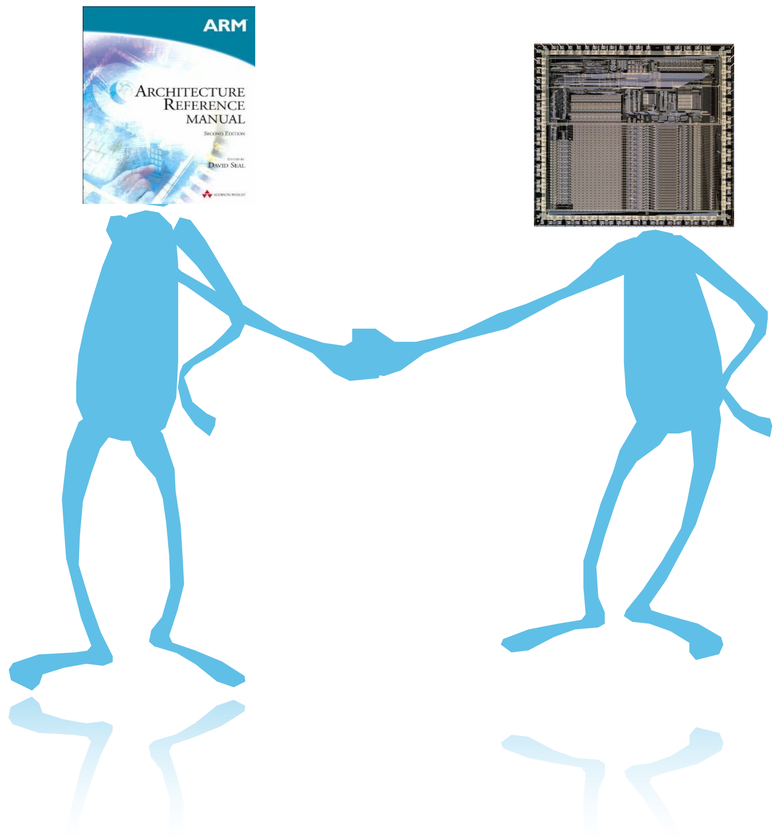
Trusting Large Specifications: The Virtuous Cycle



Alastair Reid

alastair.reid@arm.com

@alastair_d_reid

History of ARM

Joint venture between Acorn Computers and Apple

1990

Designed into first mobile phones and then smartphones

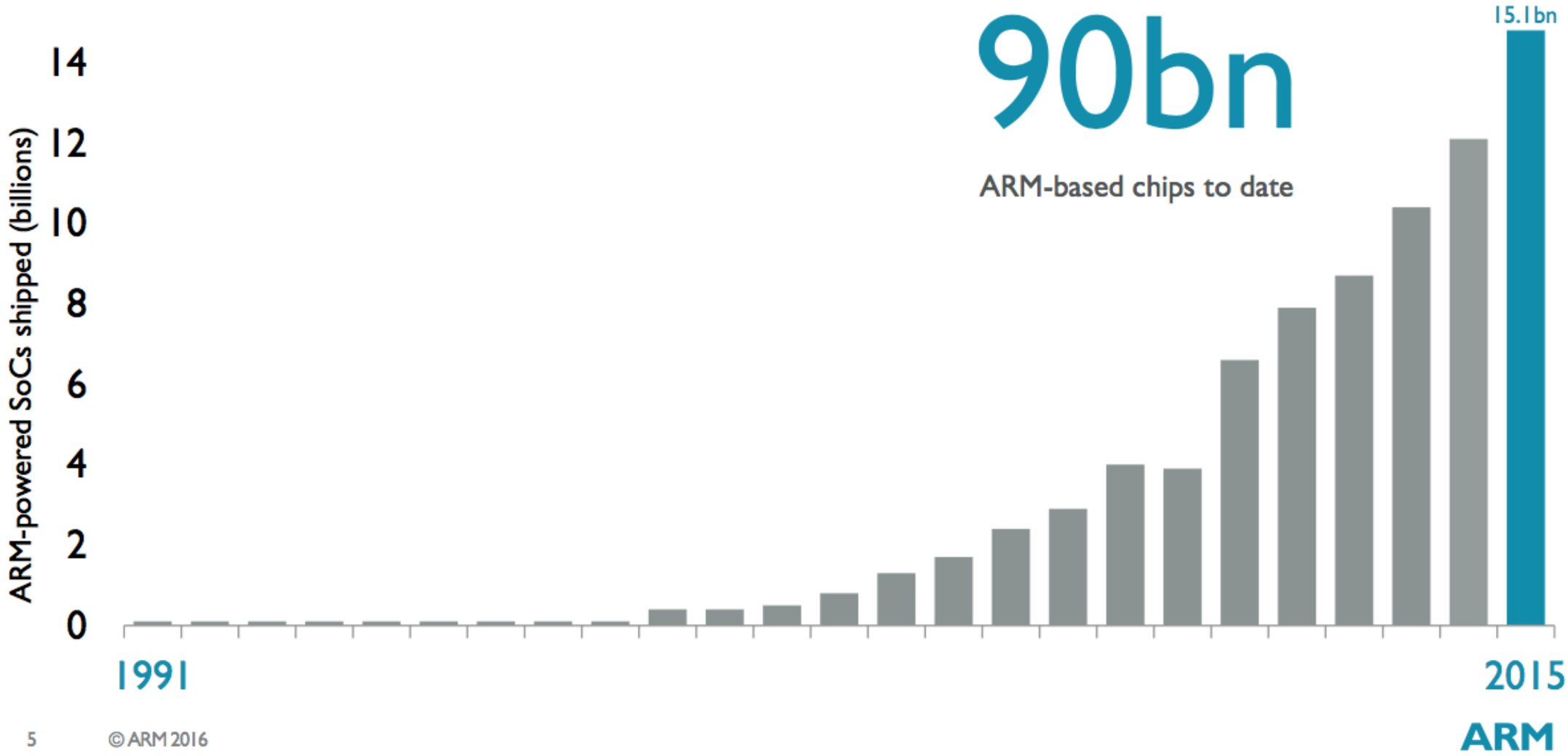
1993 onwards

Now all electronic devices can use smart ARM technology

Today

ARM

ARM-based chip shipments



Correctness

Portability / Predictability

Security

Commercial pressures

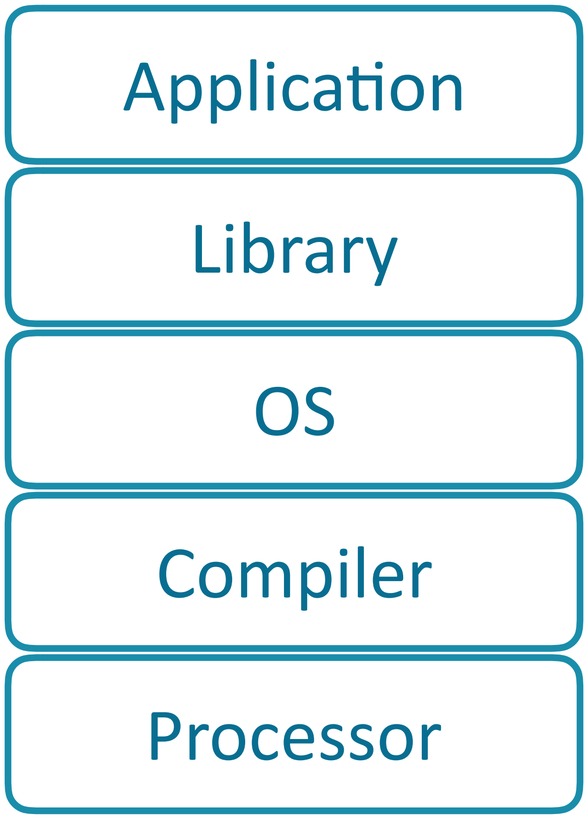
“It’s nice not to suck” — Adam Langley

Specifications

What to build

What to test

What to expect


Application
Library
OS
Compiler
Processor

Specifications: The New Bottleneck

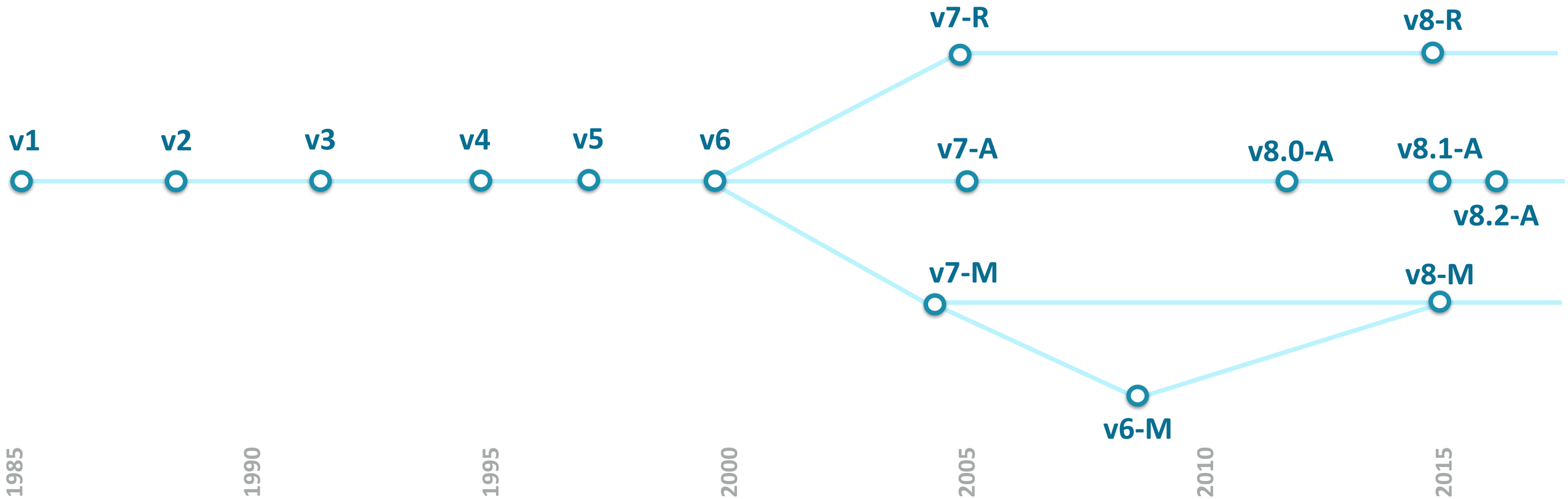
Qualities of Specifications

- Applicability
- Scope
- Trustworthiness

Testing and Using Specifications

The Virtuous Cycle

Applicability



Scope

Compiler targeted instructions?

User-level instructions?

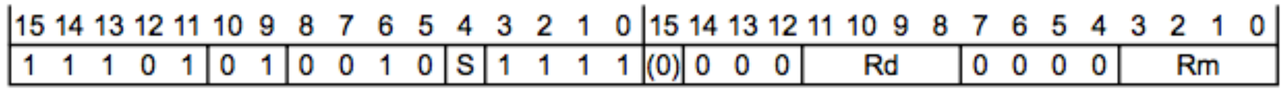
User+Supervisor?

User+Supervisor+Hypervisor+Secure Monitor?

ISA Specification - ASL

Encoding T3 ARMv7-M

MOV{S}<c>.W <Rd>,<Rm>



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

Opcode →

Decode →

```
d = UInt(Rd);  m = UInt(Rm);  setflags = (S == '1');
if setflags && (d IN {13,15} || m IN {13,15}) then UNPREDICTABLE;
if !setflags && (d == 15 || m == 15 || (d == 13 && m == 13)) then UNPREDICTABLE;
```

Execute →

```
if ConditionPassed() then
    EncodingSpecificOperations();
    result = R[m];
    if d == 15 then
        ALUWritePC(result);  // setflags is always FALSE here
    else
        R[d] = result;
        if setflags then
            APSR.N = result<31>;
            APSR.Z = IsZeroBit(result);
            // APSR.C unchanged
            // APSR.V unchanged
```

System Architecture Specification

```
AArch64.DataAbort(bits(64) vaddress, FaultRecord fault)

    route_to_el3 = HaveEL(EL3) && SCR_EL3.EA == '1' && IsExternalAbort(fault);
    route_to_el2 = (HaveEL(EL2) && !IsSecure() && PSTATE.EL IN {EL0,EL1} &&
                    (HCR_EL2.TGE == '1' || IsSecondStage(fault)));

    bits(64) preferred_exception_return = ThisInstrAddr();
    vect_offset = 0x0;

    exception = AArch64.AbortSyndrome(Exception_DataAbort, fault, vaddress);

    if PSTATE.EL == EL3 || route_to_el3 then
        AArch64.TakeException(EL3, exception, preferred_exception_return, vect_offset);
    elsif PSTATE.EL == EL2 || route_to_el2 then
        AArch64.TakeException(EL2, exception, preferred_exception_return, vect_offset);
    else
        AArch64.TakeException(EL1, exception, preferred_exception_return, vect_offset);
```

ARM Spec (lines of code)

	v8-A	v8-M
Instructions Int/FP/SIMD	26,000	6,000
Exceptions	4,000	3,000
Memory	3,000	1,000
Debug	3,000	1,000
Misc	5,500	2,000
(Test support)	1,500	2,000
Total	**43,000**	**15,000**

System Register Spec

	v8-A	v8-M
Registers	586	186
Fields	3951	622
Constant	985	177
Reserved	940	208
Impl. Defined	70	10
Passive	1888	165
Active	68	62
Operations	112	10

Trustworthiness

Trustworthiness

ARM's specification is correct *by definition*

Trustworthiness

~~ARM's specification is correct *by definition*~~

Trustworthiness

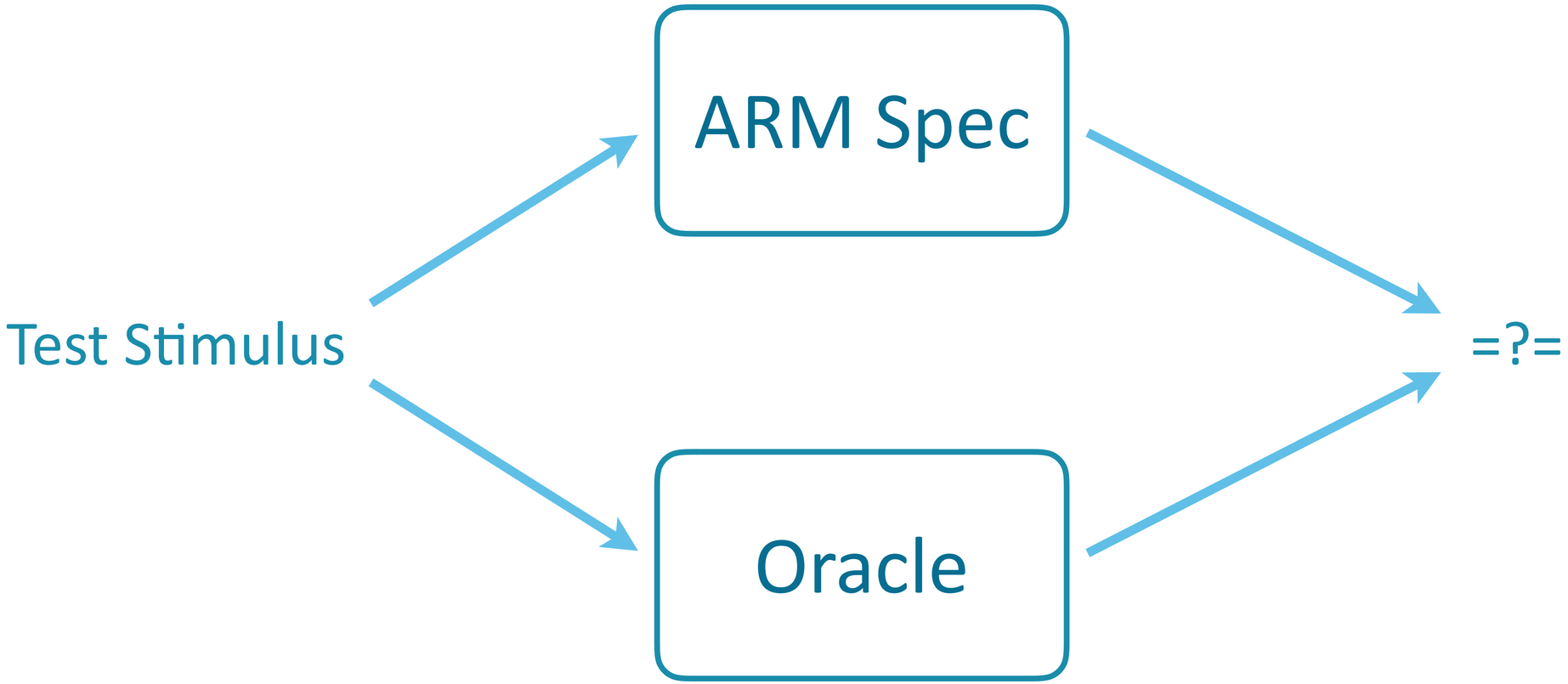
Does the specification match the behaviour of all ARM processors?

Qualities of Specifications

Testing and Using Specifications

- Testing Specifications (FMCAD 2016)
- Verifying Processors (CAV 2016)
- Generating Testcases
- Security Checking
- Booting an OS
- Fuzzing an OS

The Virtuous Cycle


ARM Spec
Test Stimulus
=?=
Oracle

Architecture Conformance Suite

Processor architectural compliance sign-off

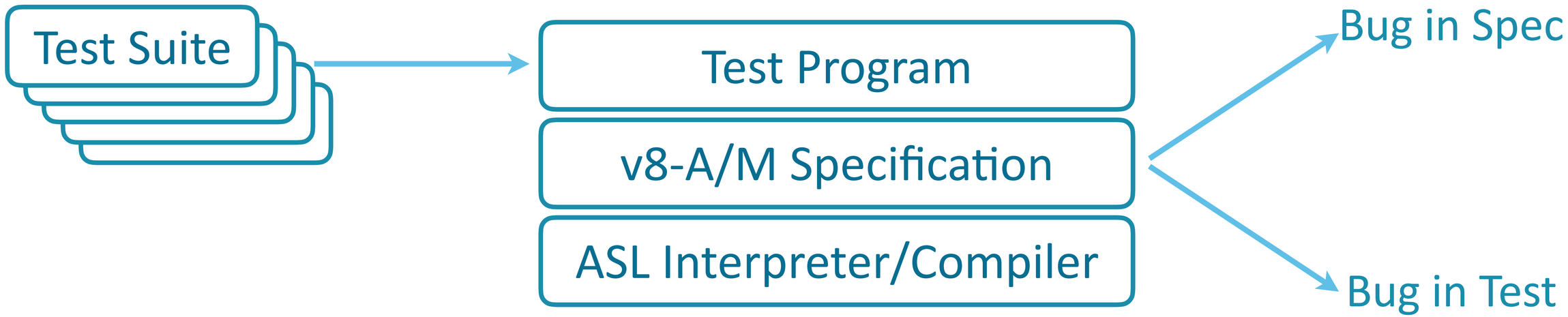
Large

- v8-A 11,000 test programs, > 2 billion instructions
- v8-M 3,500 test programs, > 250 million instructions

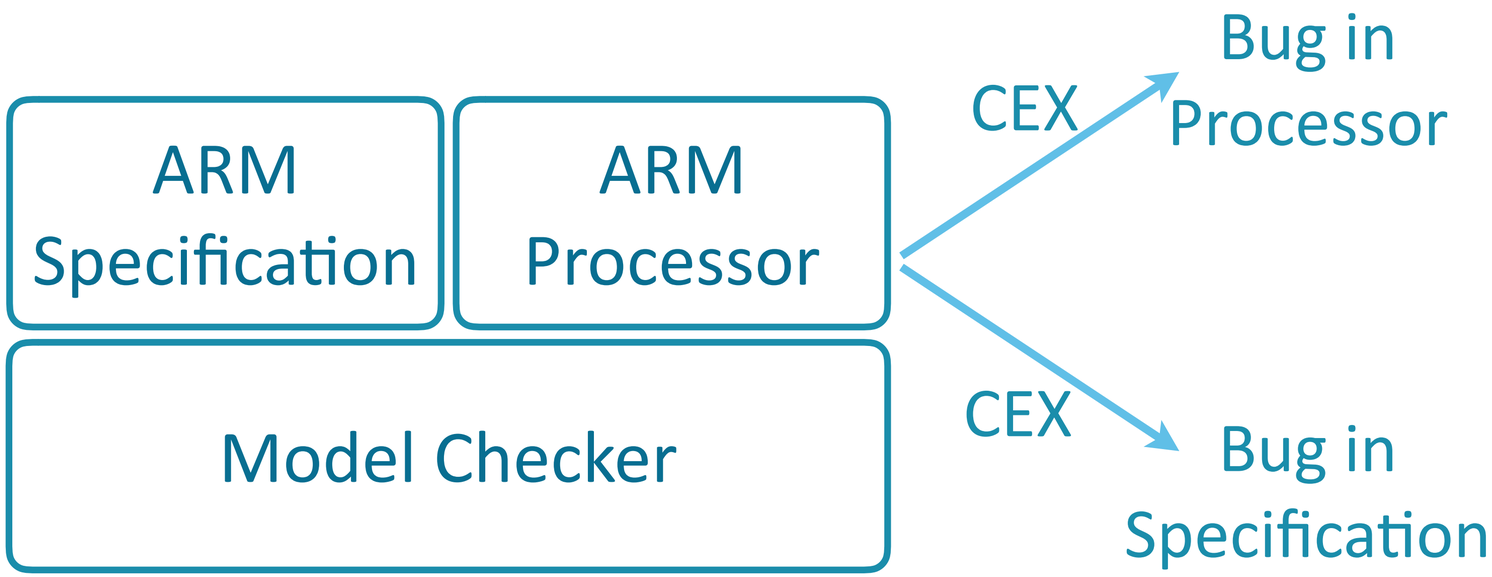
Thorough

- Tests dark corners of specification

Testing Specifications



Verifying Processors

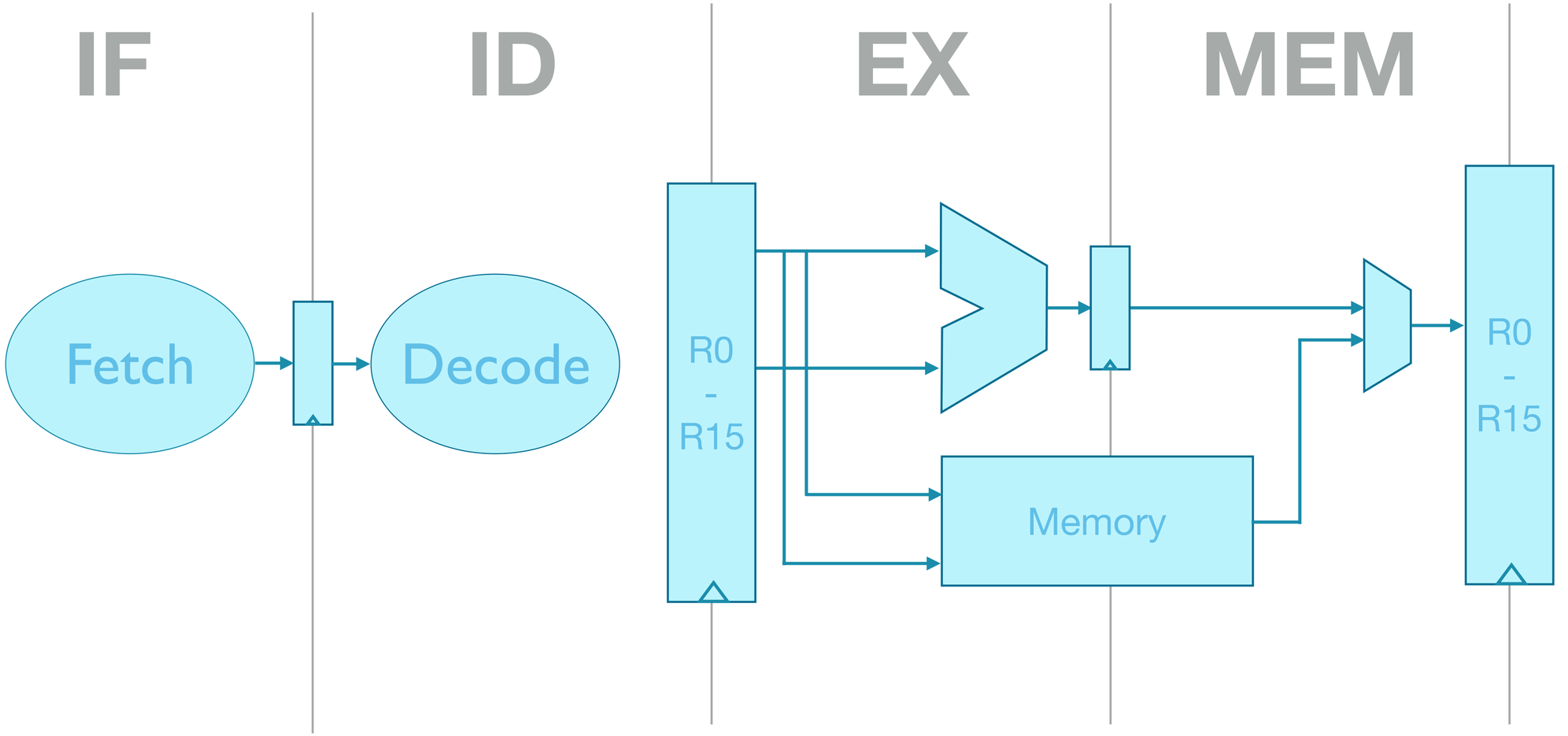


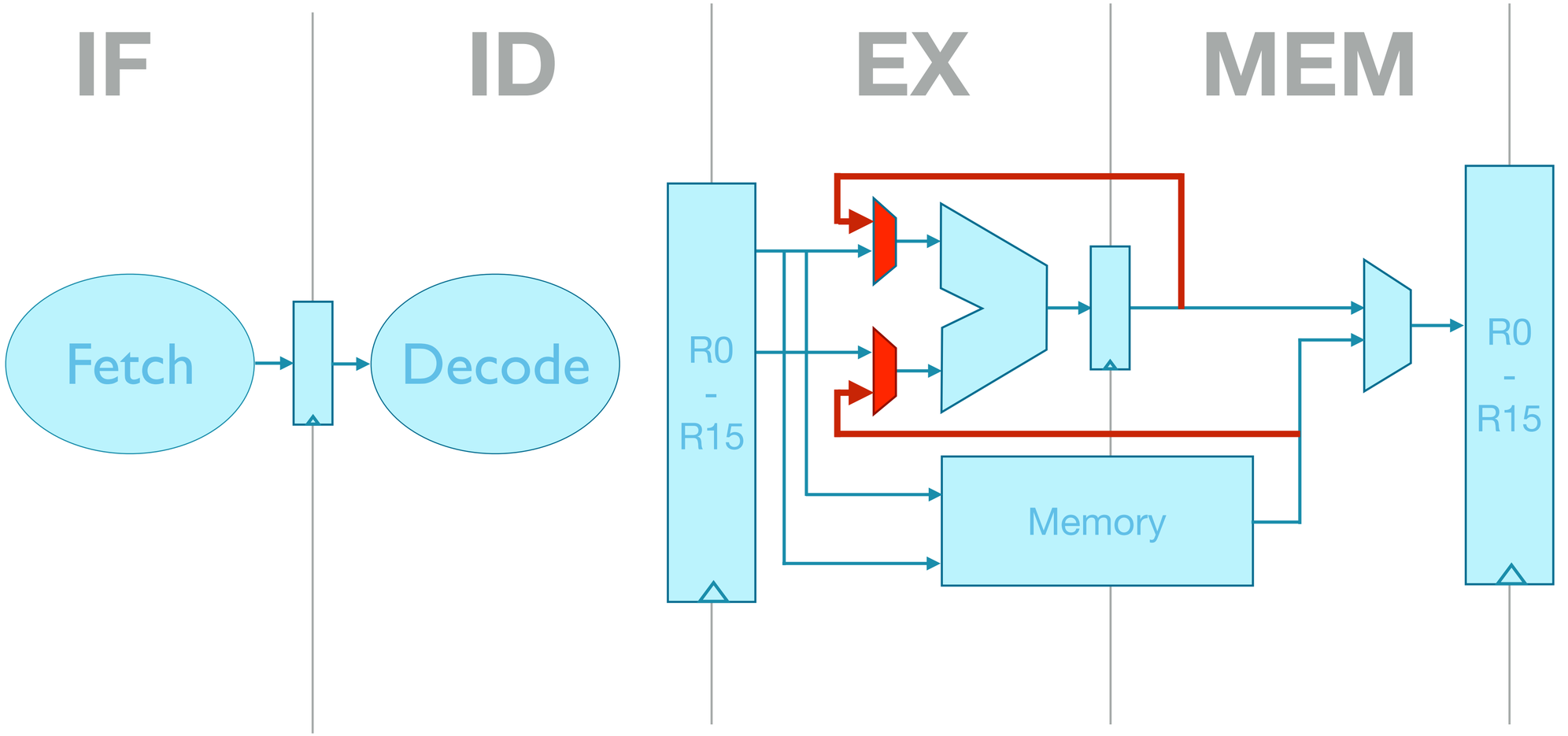
Verification Challenges

Detect the hard-to-find bugs

Large Specifications / Processors

Fit the development flow

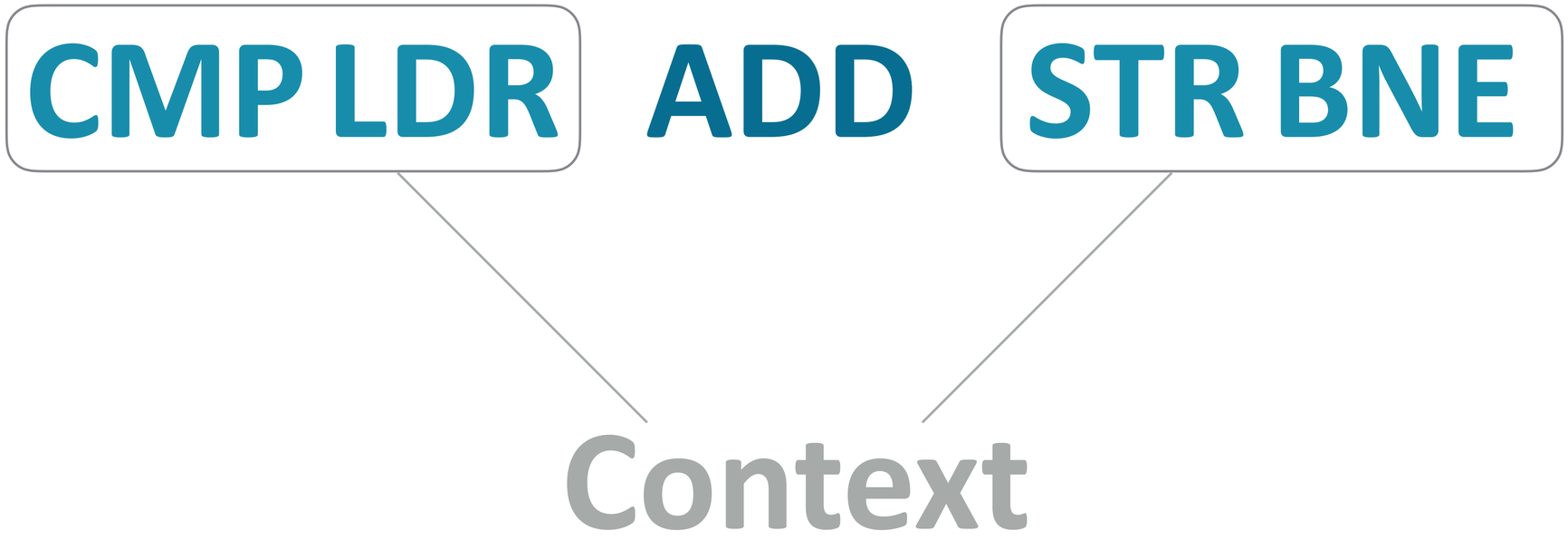

IF
ID
EX
MEM
WB
Fetch
Decode
R0
-
R15
Memory
R0
-
R15

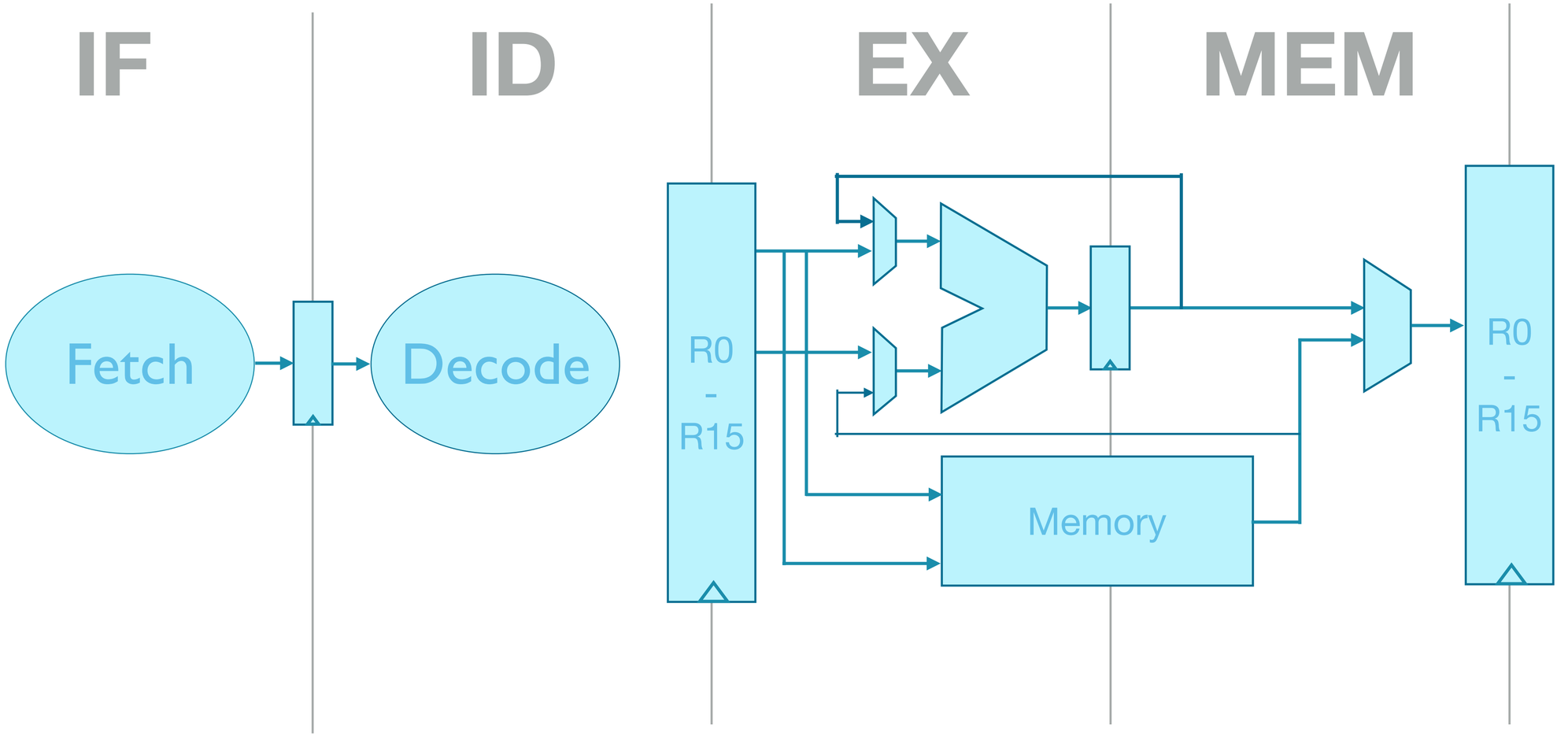

IF
ID
EX
MEM
WB
Fetch
Decode
R0
-
R15
Memory
R0
-
R15

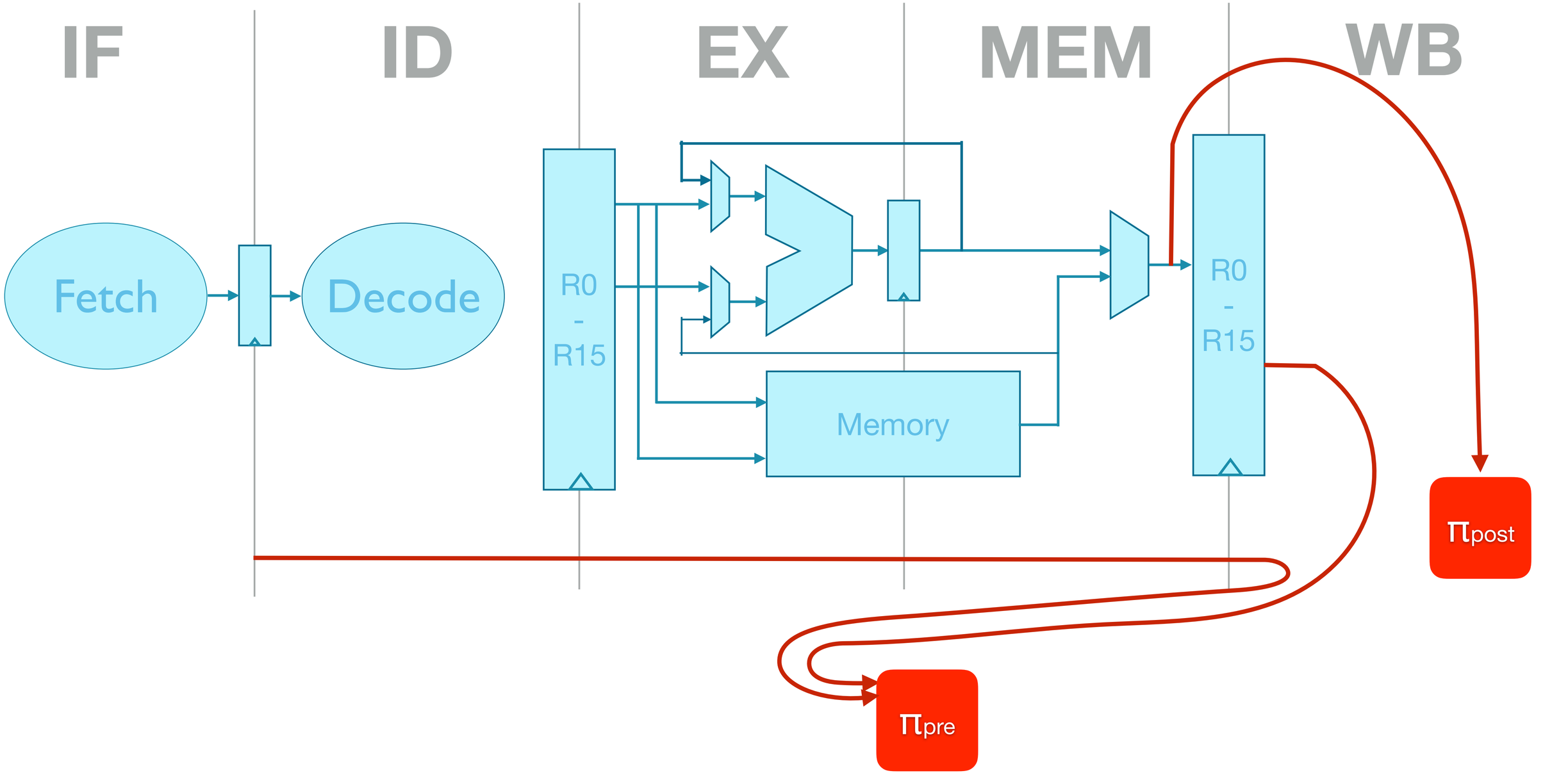
Checking an instruction

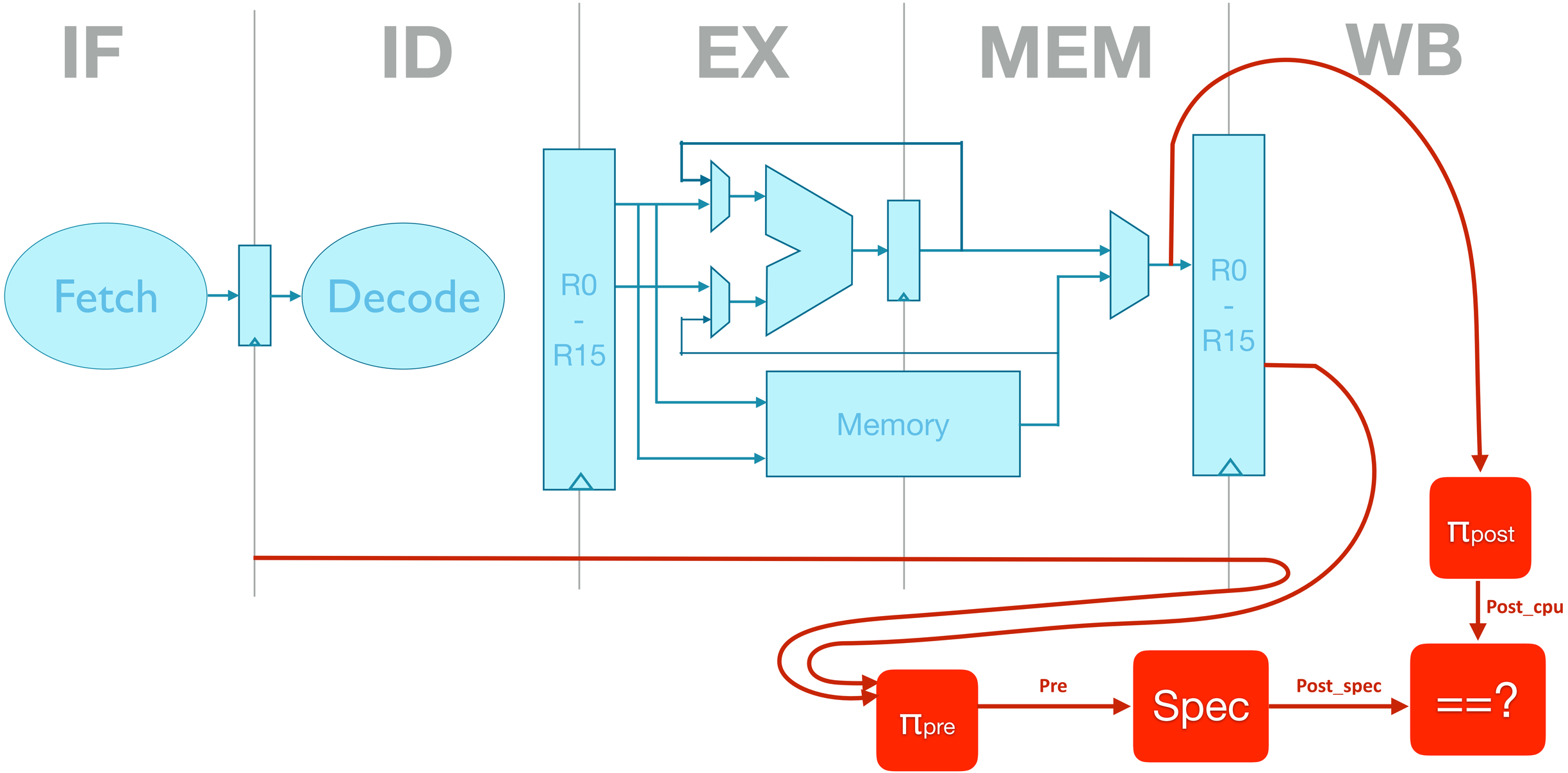
ADD

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
πpre
Pre
Spec
Post_spec
==?

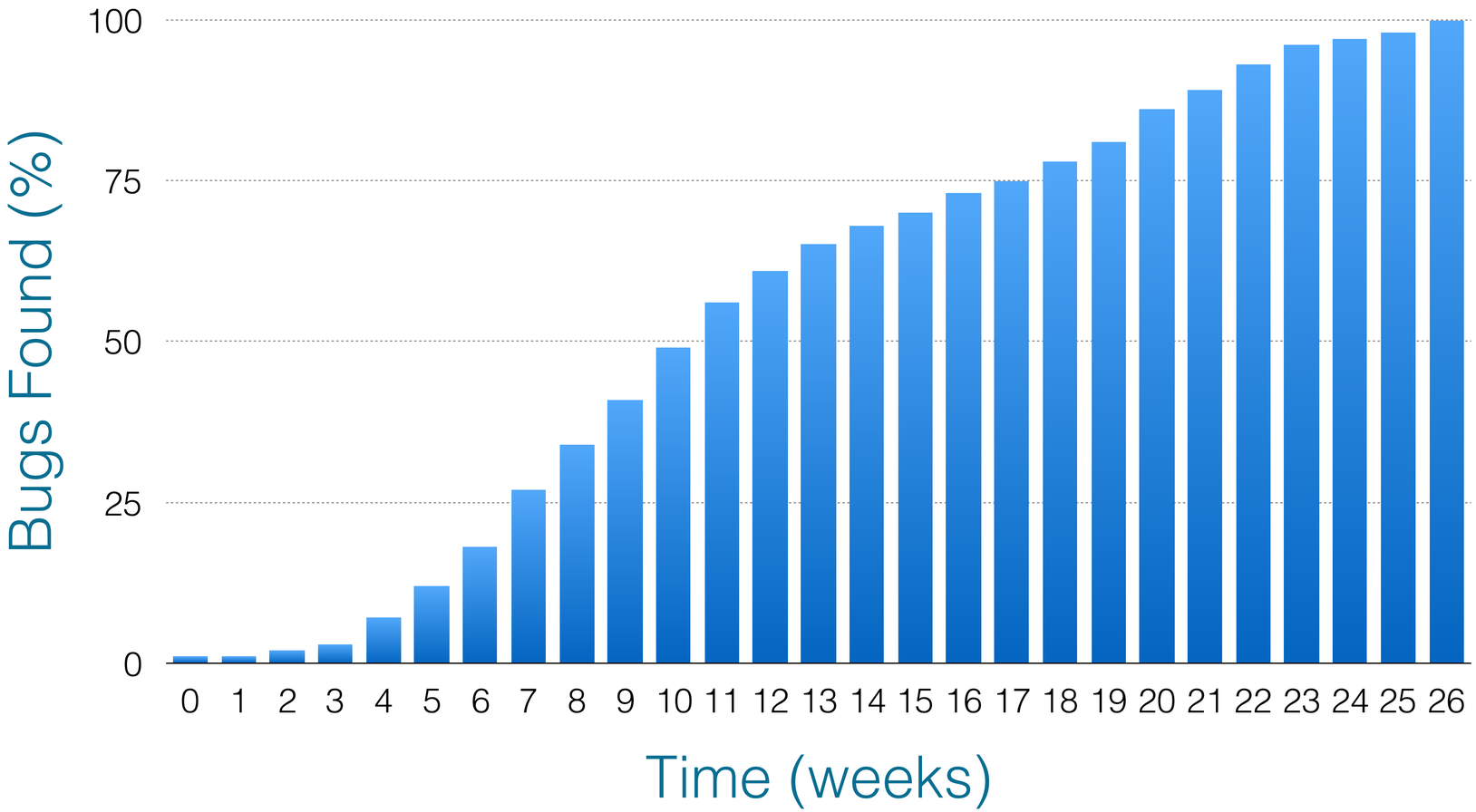
ISA-Formal Properties

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

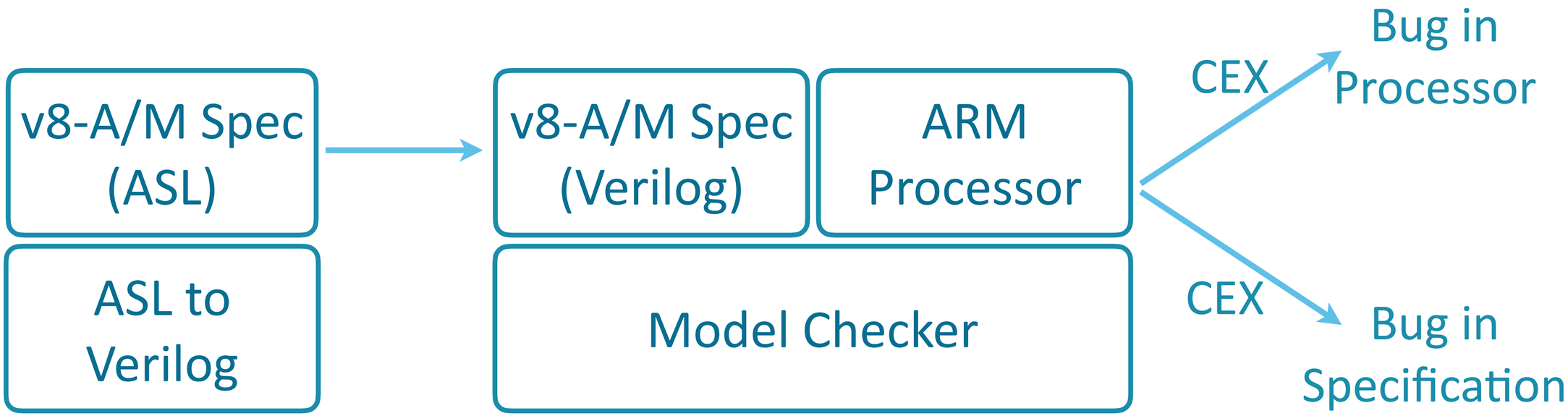
Automation

Specialize
Monomorphize
Constant Propagation
Width Analysis
Exception Handling
...

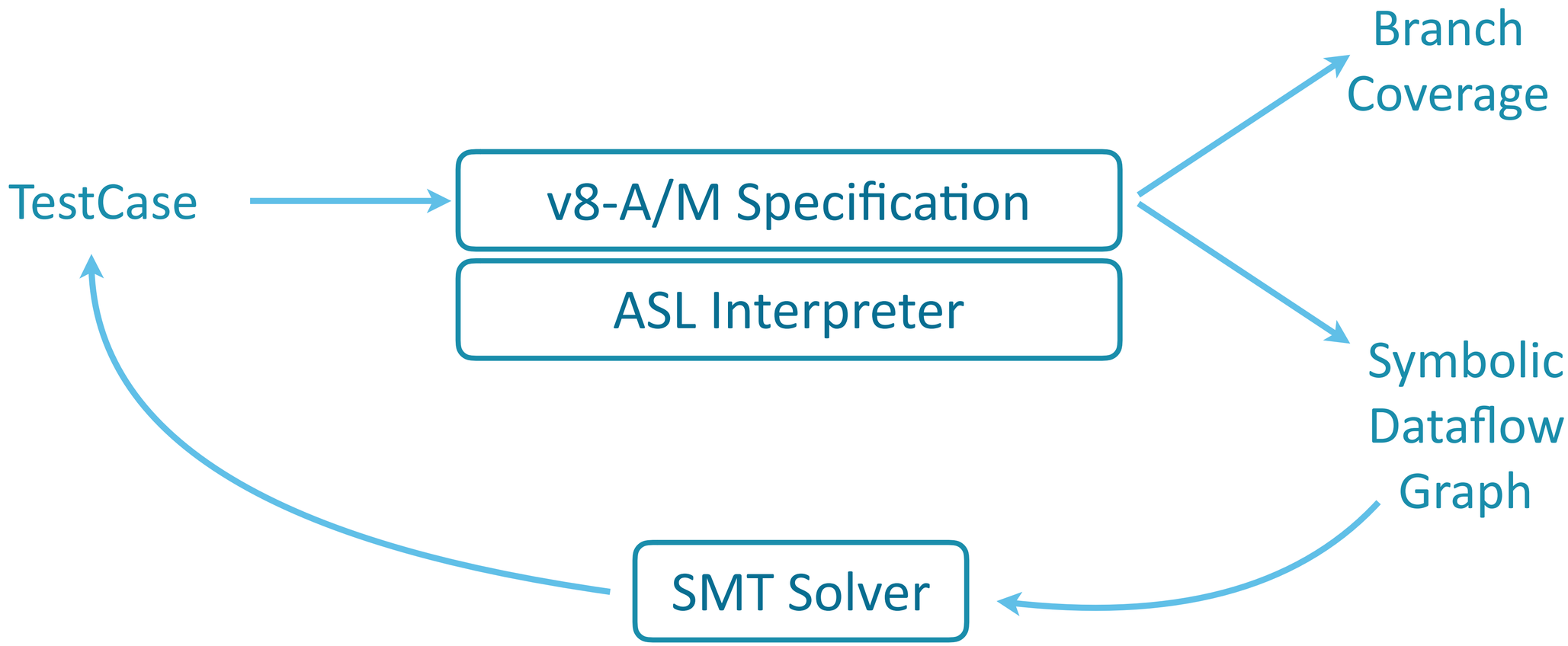
Verification Progress



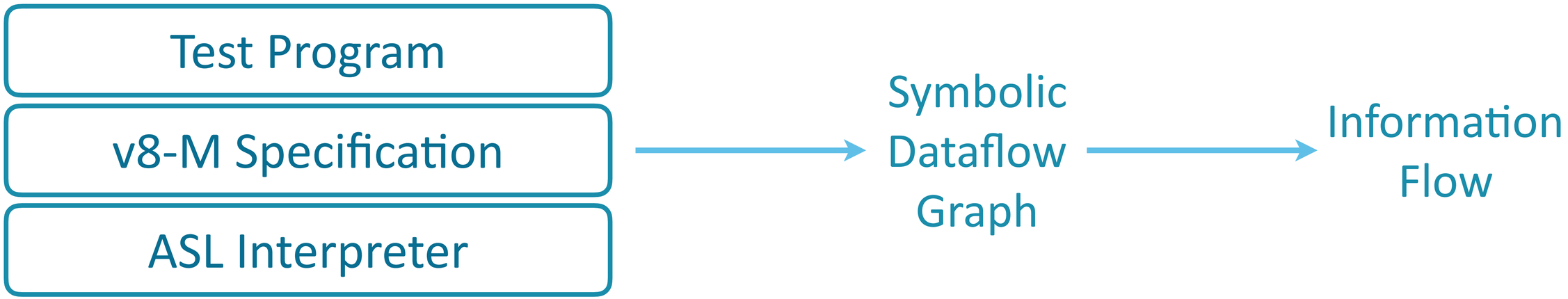
Verifying Processors



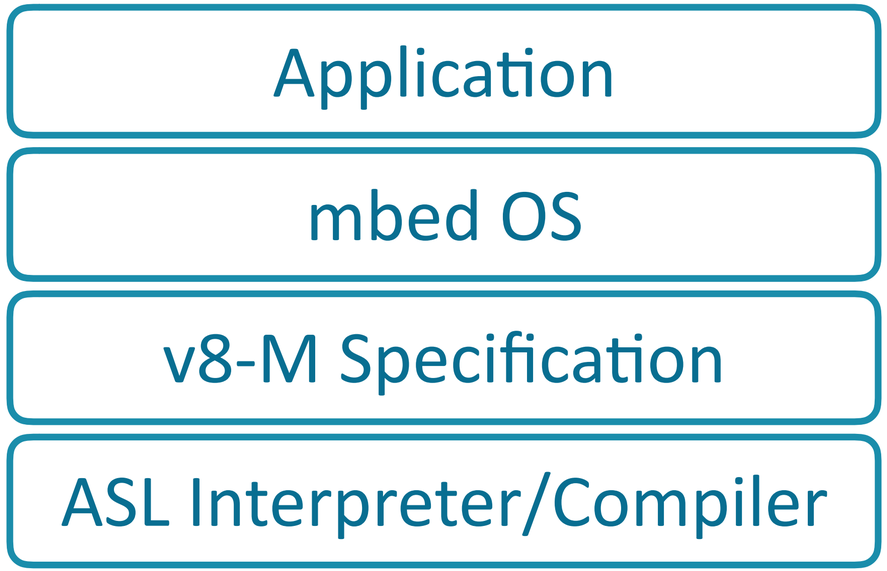
Testcase Generation



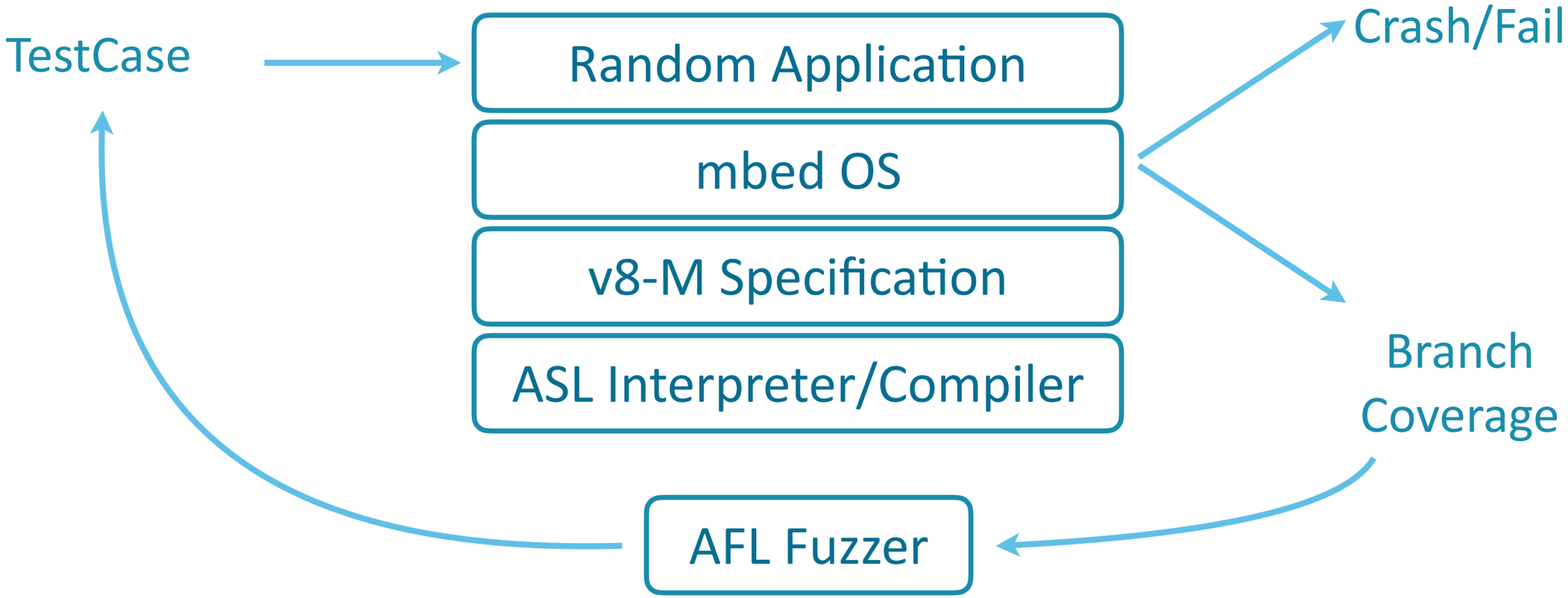
Security Checking



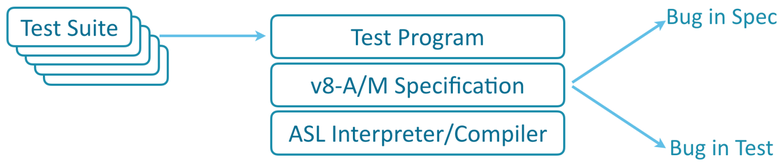
Booting an OS



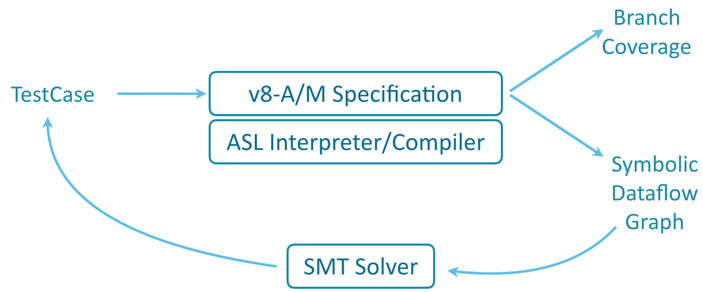
Fuzzing the mbed OS



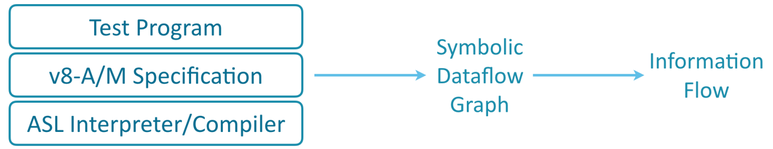
Testing Specifications



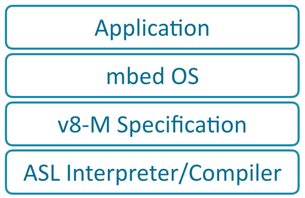
Testcase Generation



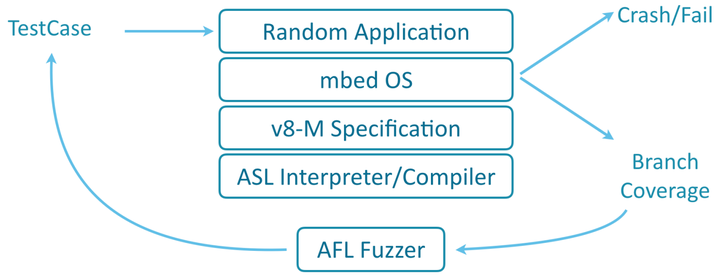
Security Checking



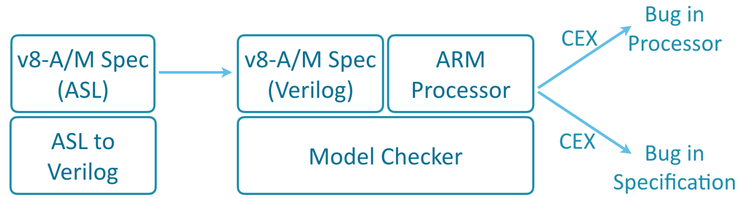
Booting an OS



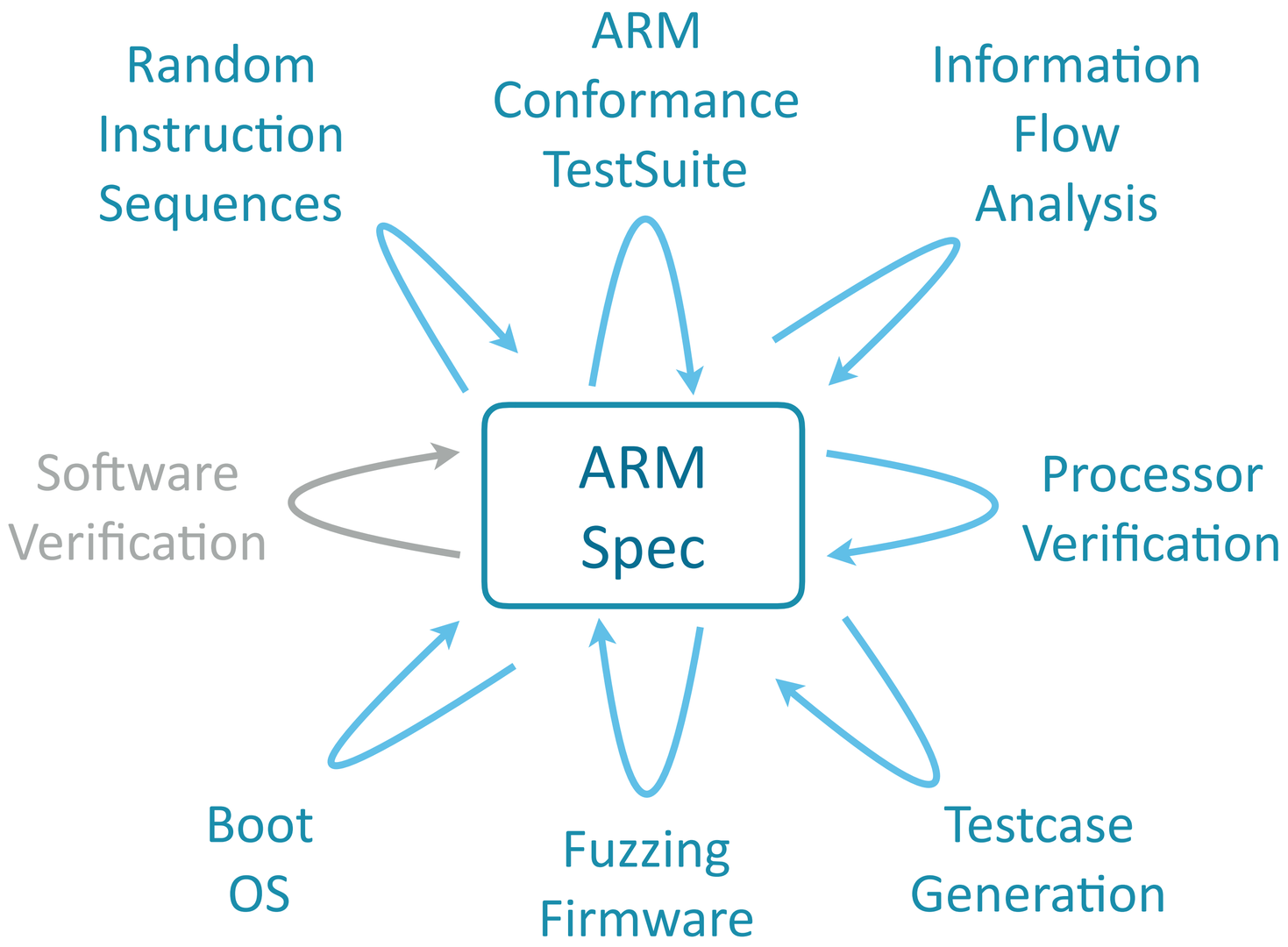
Fuzzing the mbed OS



Verifying Processors



Creating a Virtuous Cycle



Preparing public release of ARM v8-A specification

- Enable formal verification of software and tools
- Public release planned for ~~2016 Q4~~ 2017 Q1
- Liberal license
- Cambridge University REMS group currently translating to SAIL

Talk to me about how I can help you use it

The New Bottleneck: Specifications

Required for formal verification

Too large to be "obviously correct"

Reusable specs enable "virtuous cycle"

- Increases Scope / Applicability requirements
- Converge on correct specification

Looking for interns in Security and Correctness - contact me

End

alastair.reid@arm.com
@alastair_d_reid