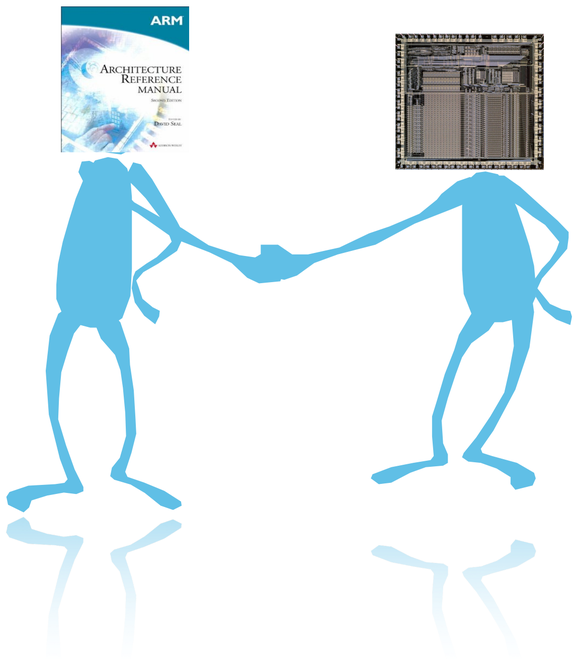
What makes processors fail and how to prevent it

Alastair Reid



alastair.reid@arm.com
@alastair_d_reid

Processors always work

More technical version: https://alastairreid.github.io/papers/CAV_16/

Related talk: Mate Soos “Hacking using SAT and SMT solvers”

Related workshop: Jonny Austin “Be a Computer! Demystifying assembly language and CPUs”

alastair.reid@arm.com @alastair_d_reid

Processors *almost* always work

More technical version: https://alastairreid.github.io/papers/CAV_16/

Related talk: Mate Soos “Hacking using SAT and SMT solvers”

Related workshop: Jonny Austin “Be a Computer! Demystifying assembly language and CPUs”

alastair.reid@arm.com @alastair_d_reid

Correct

Fast

More technical version: https://alastairreid.github.io/papers/CAV_16/

Related talk: Mate Soon “Hacking using SAT and SMT solvers”

Related workshop: Jonny Austin “Be a Computer! Demystifying assembly language and CPUs”

alastair.reid@arm.com @alastair_d_reid

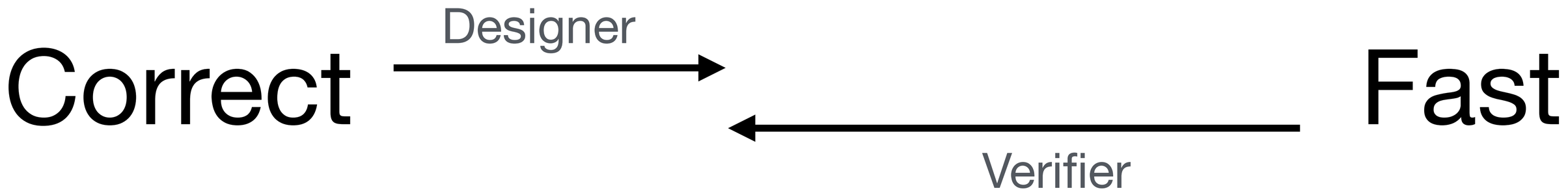
Correct → Designer → Fast

More technical version: https://alastairreid.github.io/papers/CAV_16/

Related talk: Mate Soon “Hacking using SAT and SMT solvers”

Related workshop: Jonny Austin “Be a Computer! Demystifying assembly language and CPUs”

alastair.reid@arm.com @alastair_d_reid

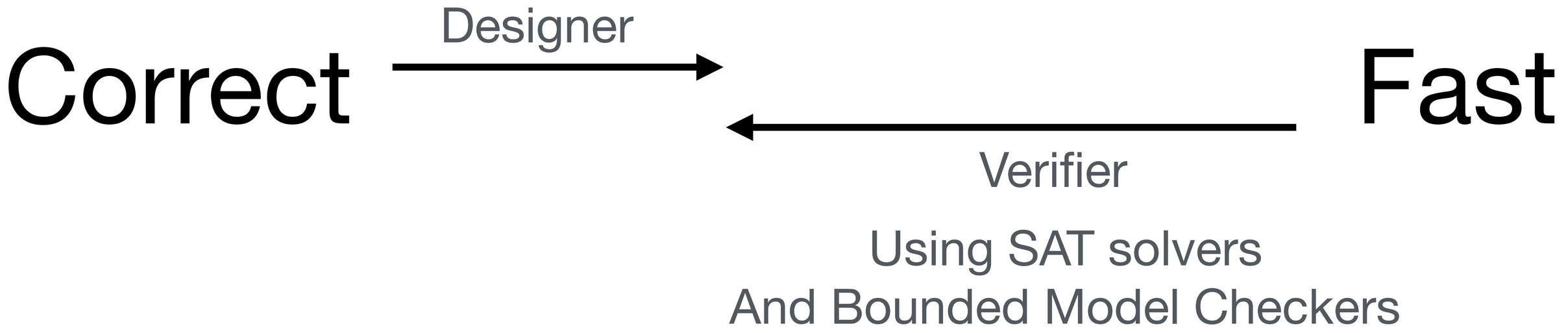


More technical version: https://alastairreid.github.io/papers/CAV_16/

Related talk: Mate Soon “Hacking using SAT and SMT solvers”

Related workshop: Jonny Austin “Be a Computer! Demystifying assembly language and CPUs”

alastair.reid@arm.com @alastair_d_reid

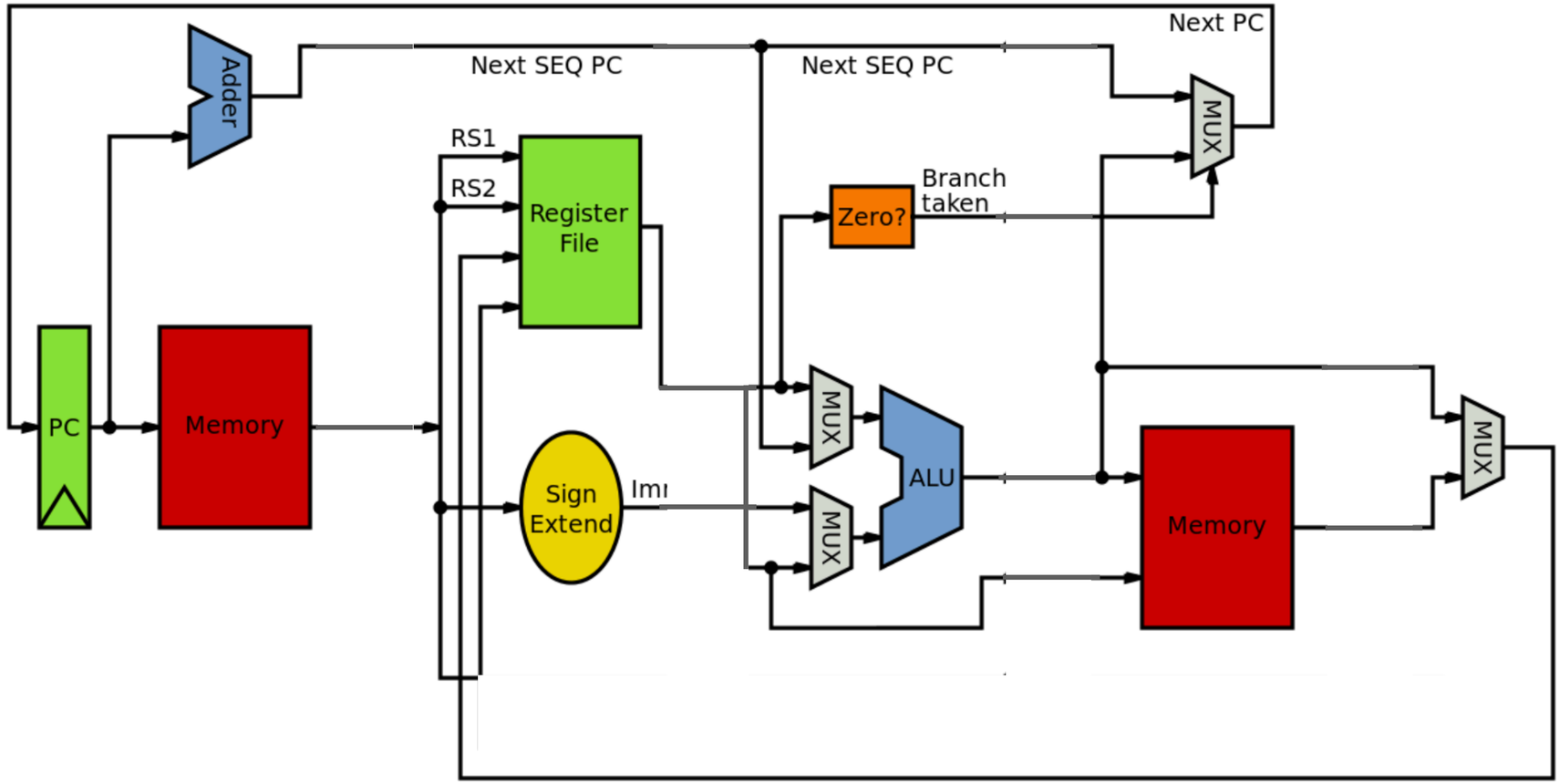


More technical version: https://alastairreid.github.io/papers/CAV_16/

Related talk: Mate Soon “Hacking using SAT and SMT solvers”

Related workshop: Jonny Austin “Be a Computer! Demystifying assembly language and CPUs”

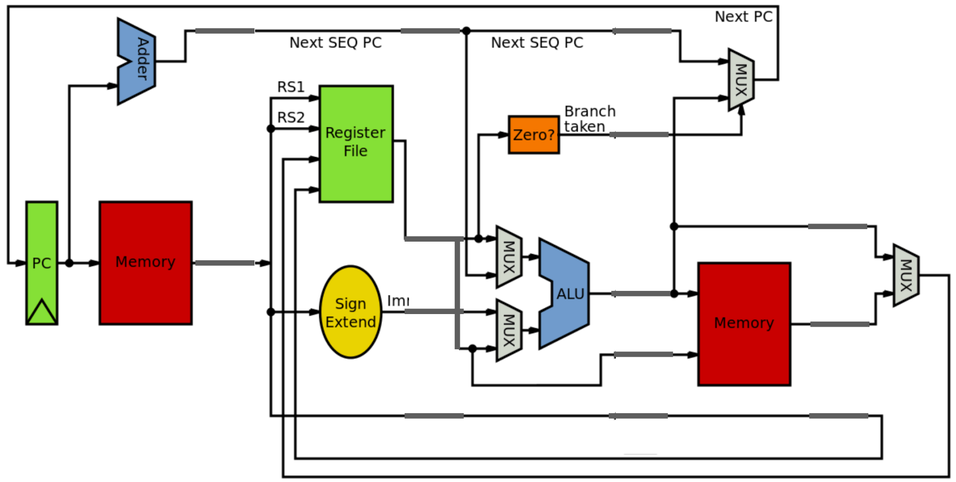
alastair.reid@arm.com @alastair_d_reid

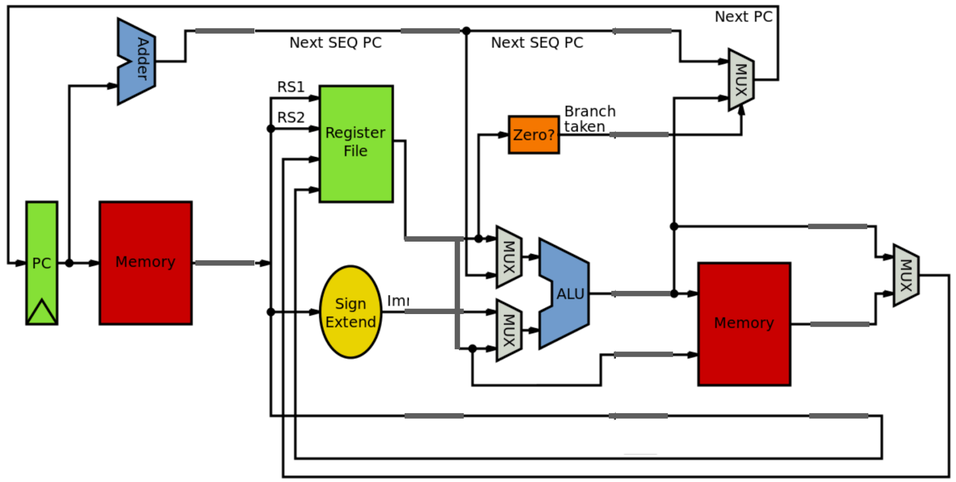


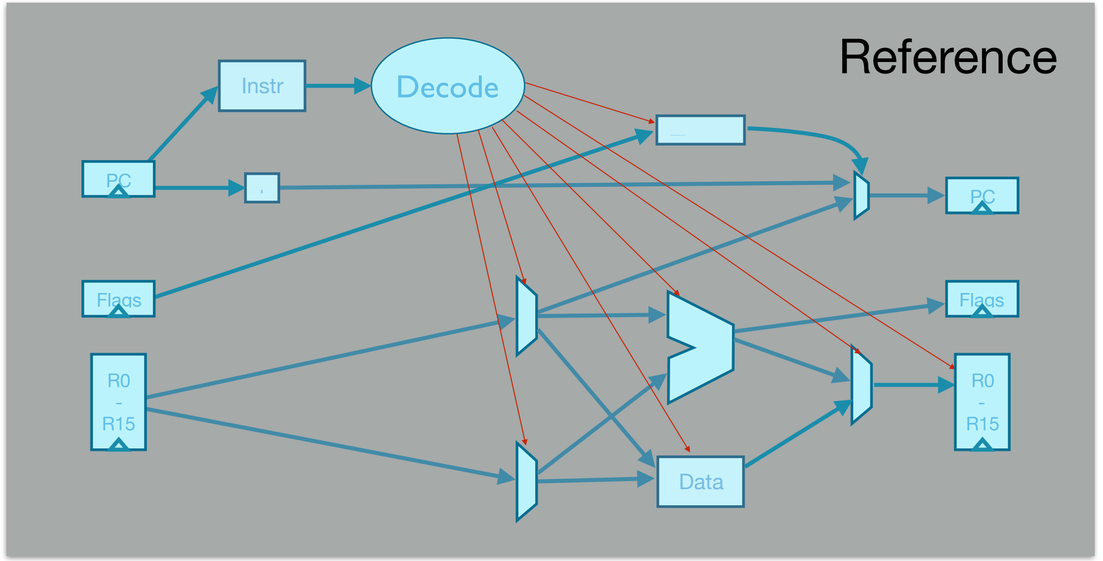
Butchered version of https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

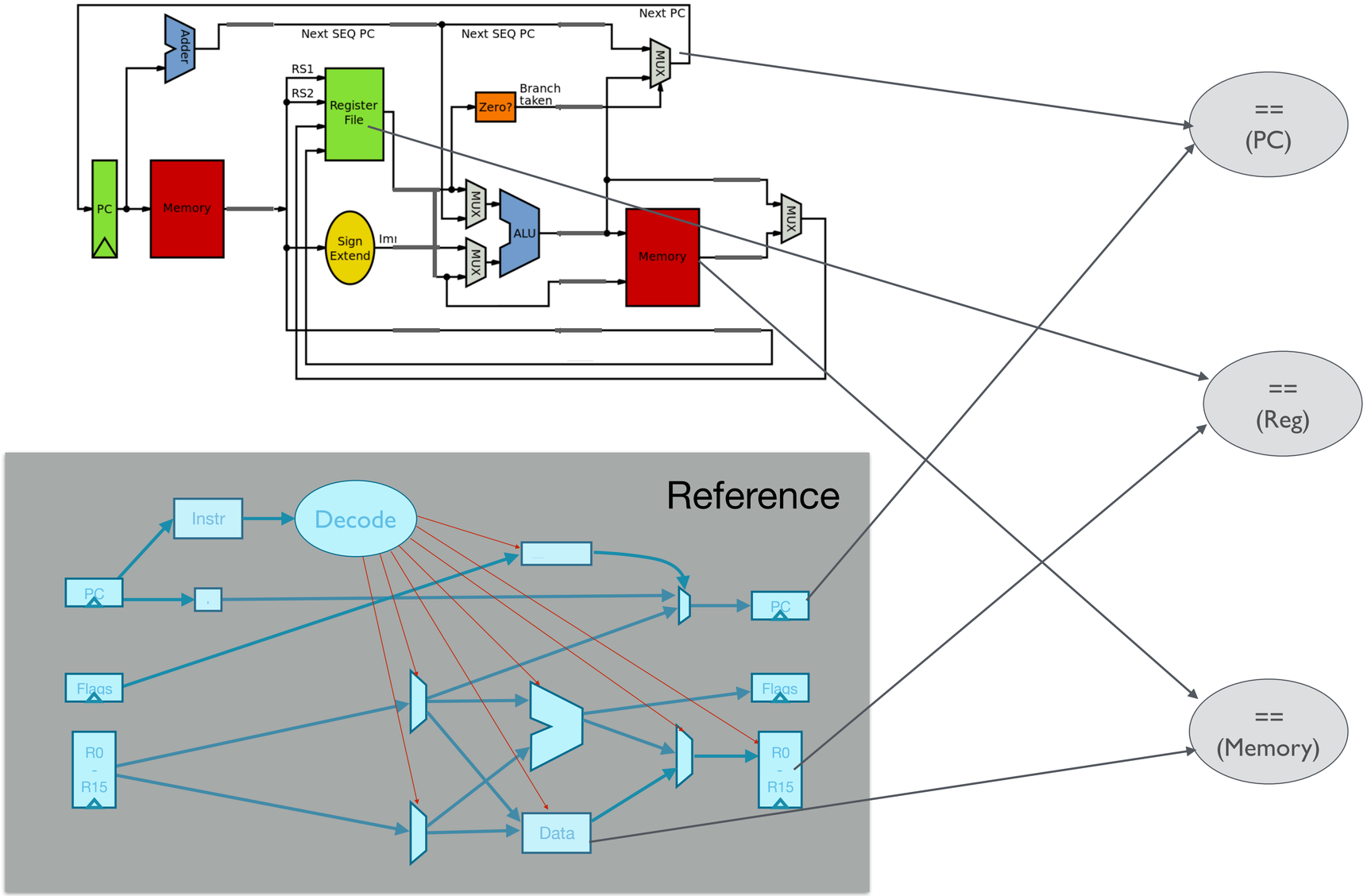
What tests to run?

Every instruction	x	Every corner case	x	Fudge Factor
ADD		Big		1?
CMP		Small		
LDR	x	Equal	x	50?
STR		Min		
BNE		Max		1000?

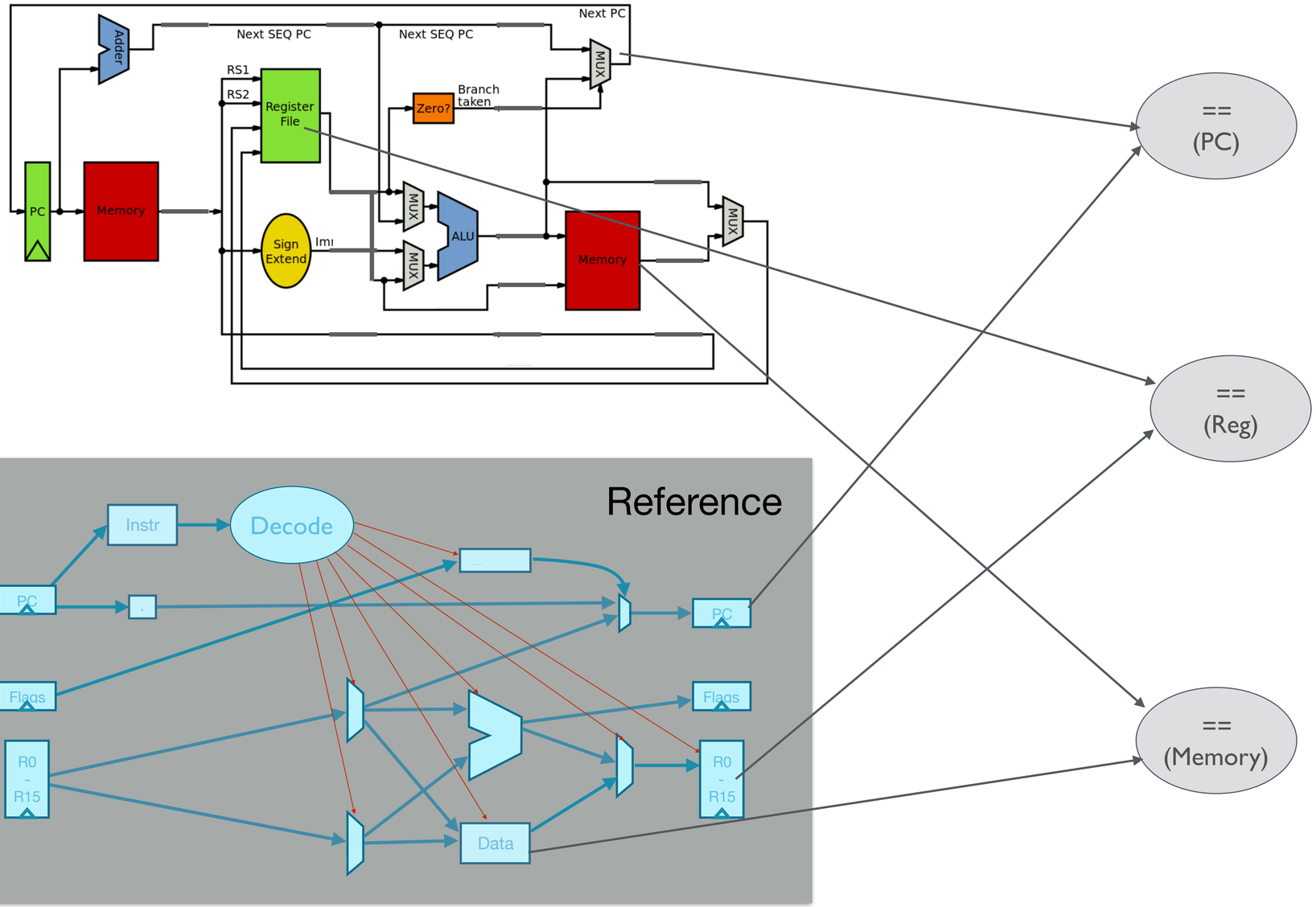

Next PC
Adder
Next SEQ PC
Next SEQ PC
MUX
RS1
RS2
Register File
Branch taken
Zero?
PC
Memory
MUX
ALU
Sign Extend
Imm
MUX
Memory
MUX

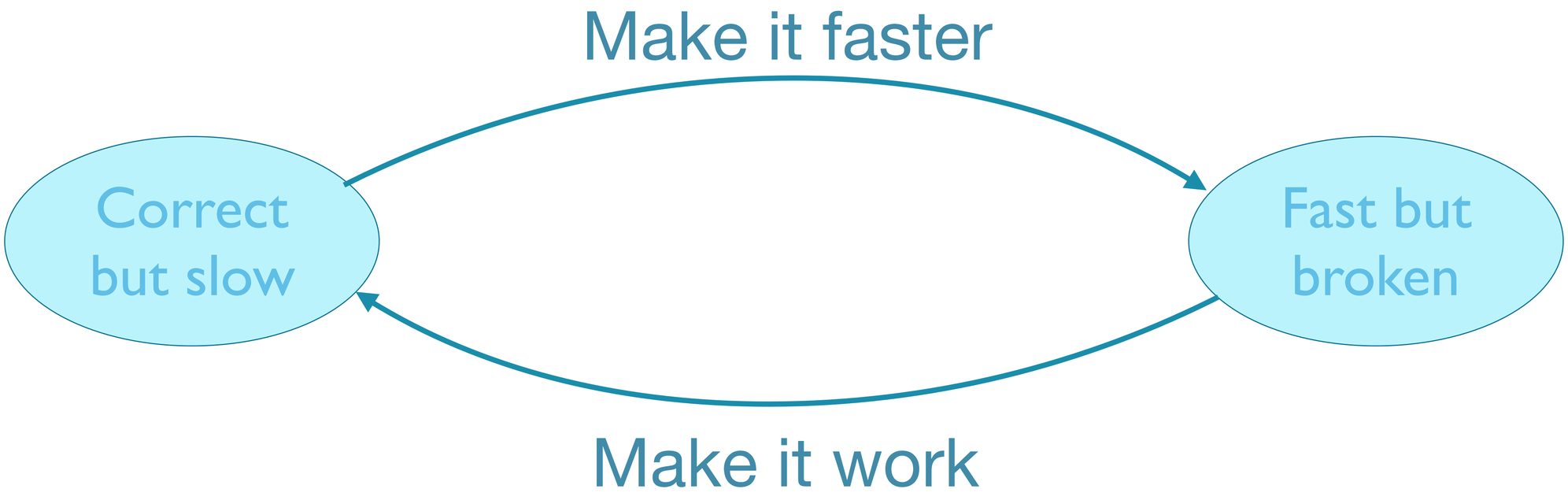

Next PC
Adder
Next SEQ PC
Next SEQ PC
MUX
RS1
RS2
Register File
Zero?
Branch taken
PC
Memory
MUX
ALU
Sign Extend
Imm
MUX
Memory
MUX

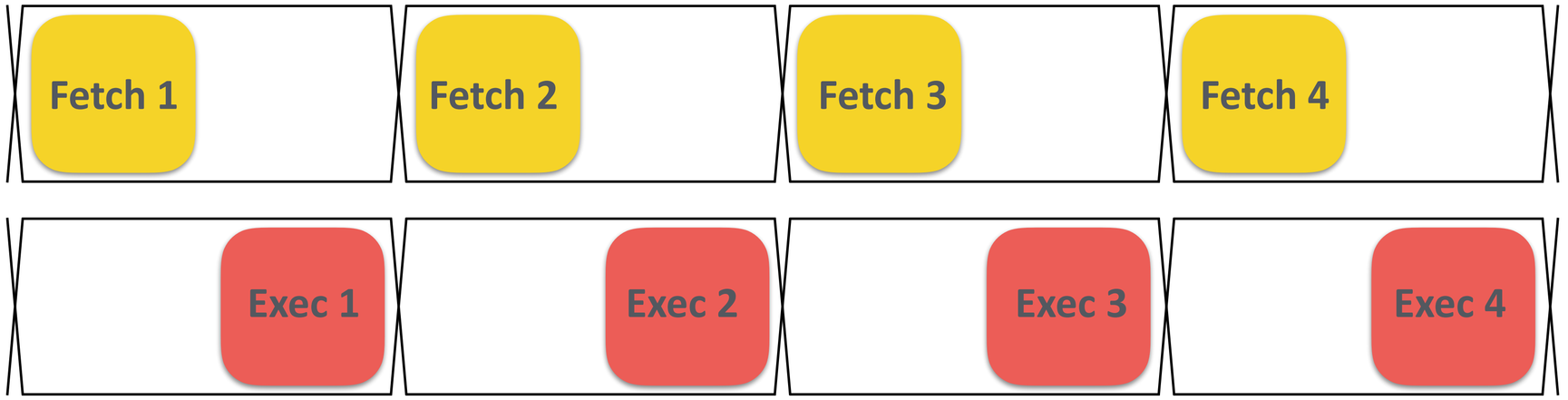

Reference
Instr
Decode
PC
Flags
R0 - R15
PC
Flags
R0 - R15
Data


Next PC
Adder
Next SEQ PC
Next SEQ PC
MUX
RS1
RS2
Register File
Zero?
Branch taken
PC
Memory
MUX
ALU
Sign Extend
Imm
MUX
Memory
MUX
==
(PC)
==
(Reg)
==
(Memory)
Reference
Instr
Decode
PC
PC
Flags
Flags
R0
-
R15
R0
-
R15
Data

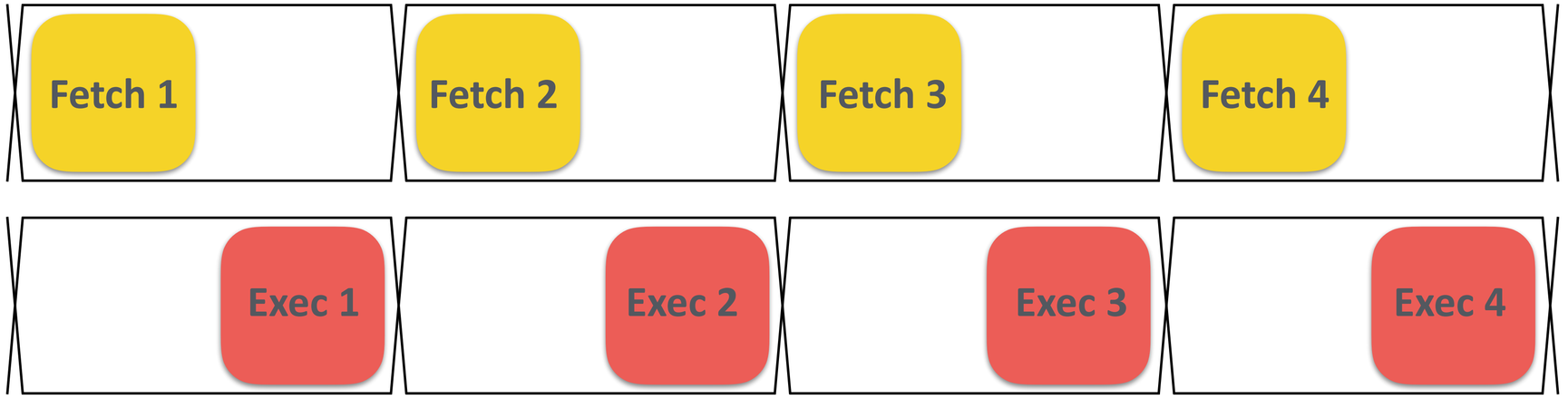
Logical Equivalence Check (LEC) / SAT

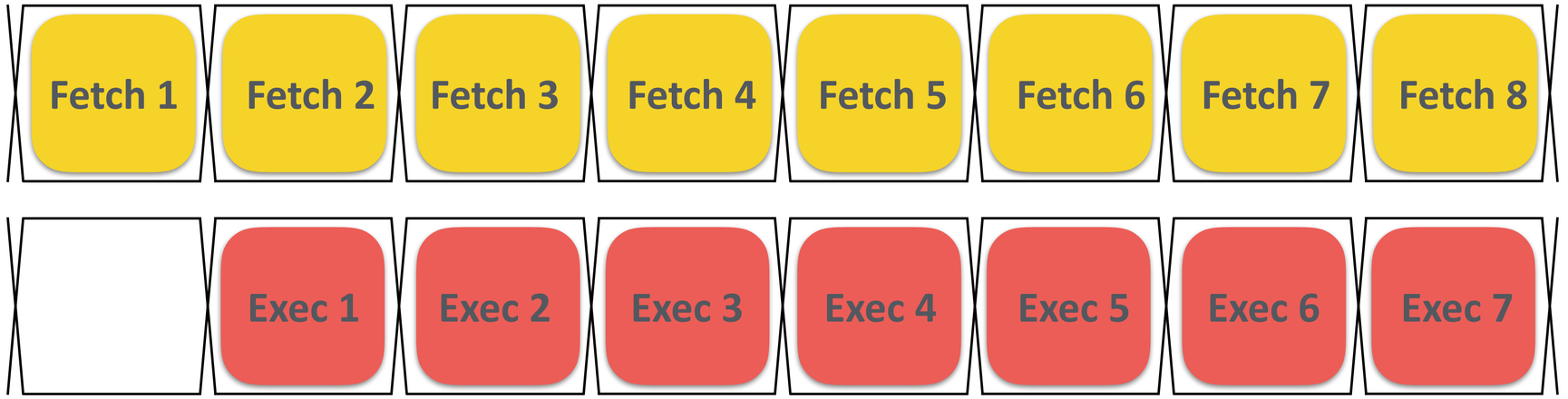



Make it faster
Correct
but slow
Fast but
broken
Make it work

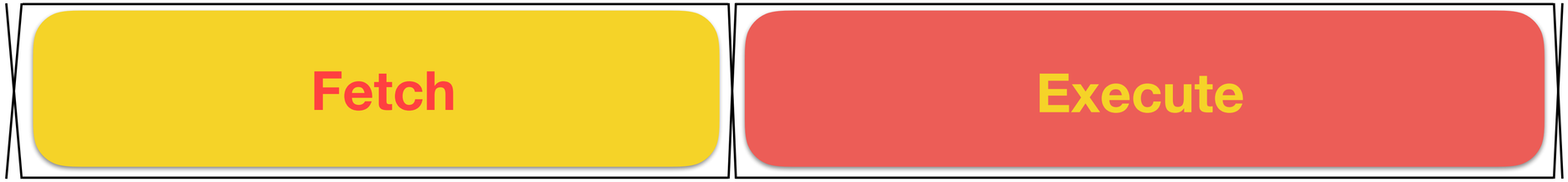


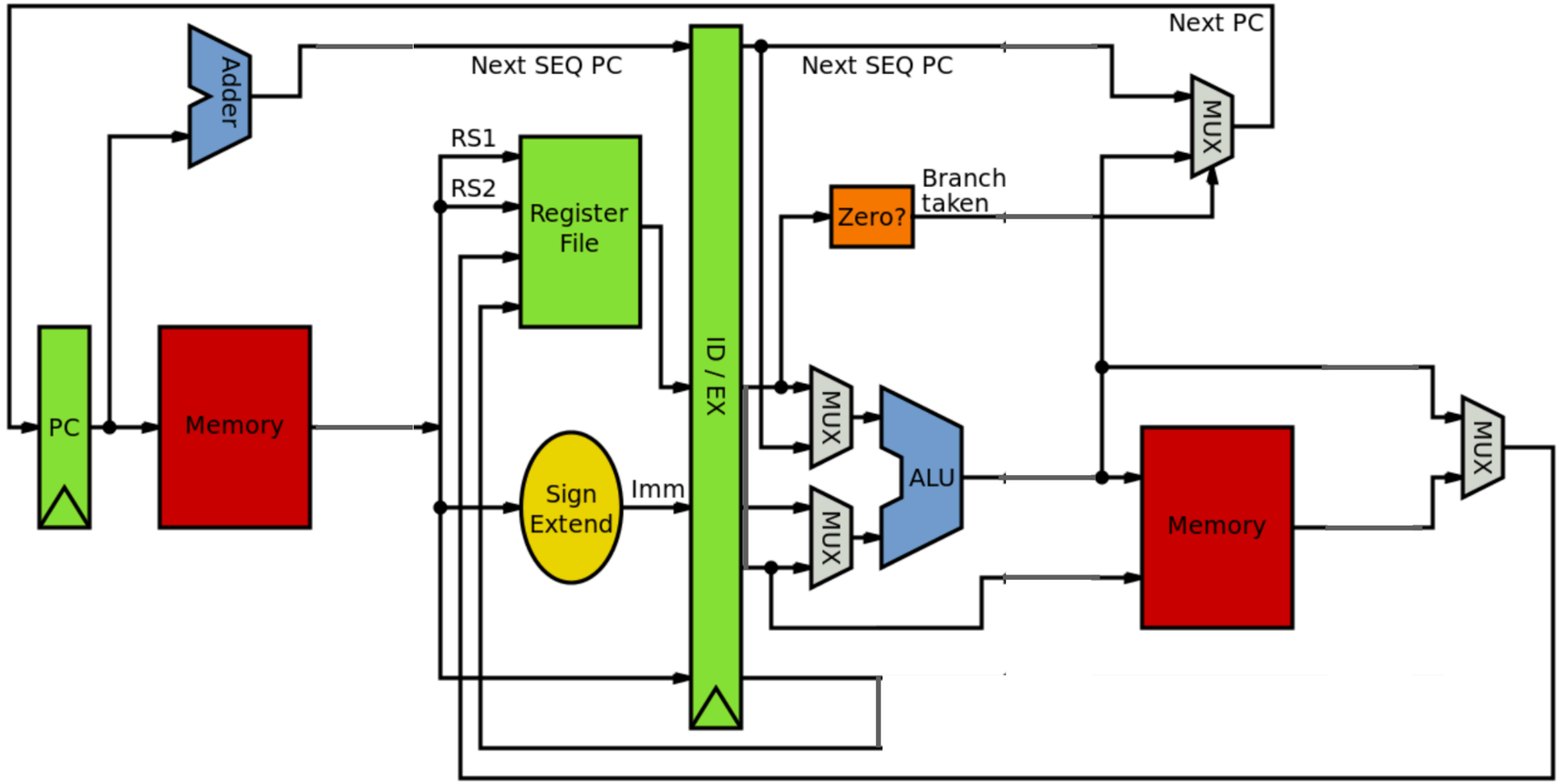
https://en.wikipedia.org/wiki/Instruction_pipelining



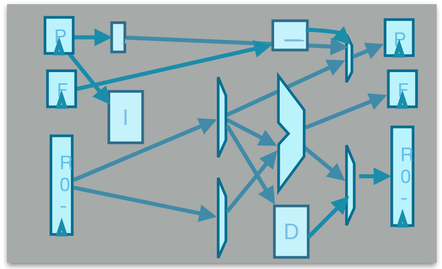


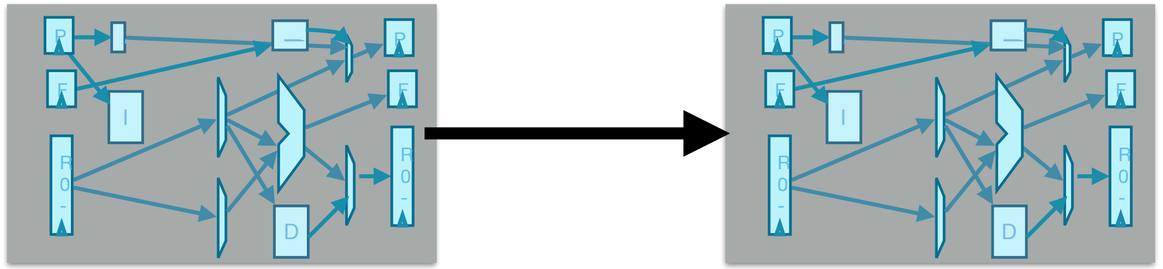
https://en.wikipedia.org/wiki/Instruction_pipelining

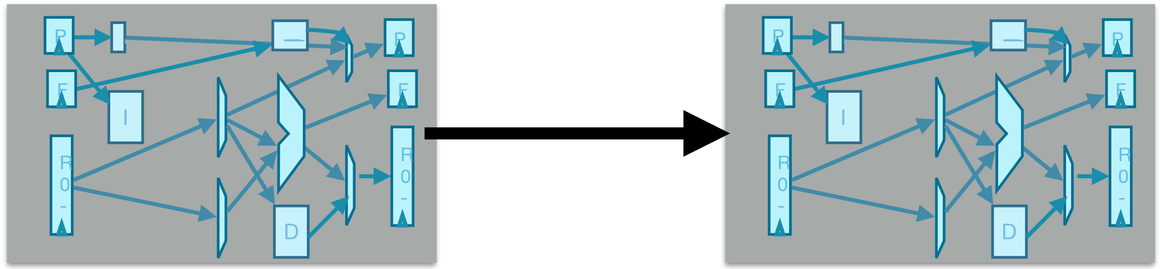


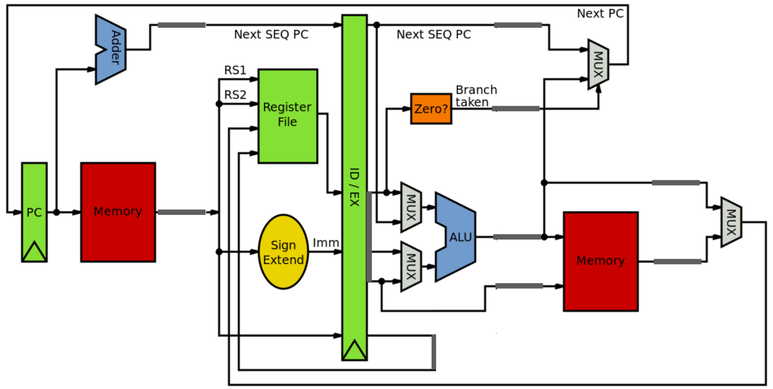


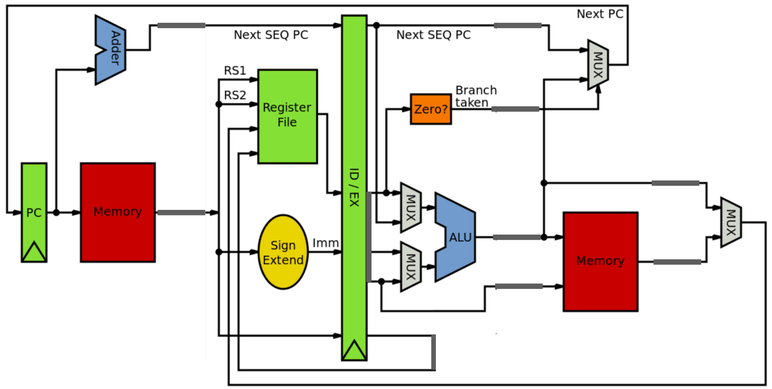
Butchered version of https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

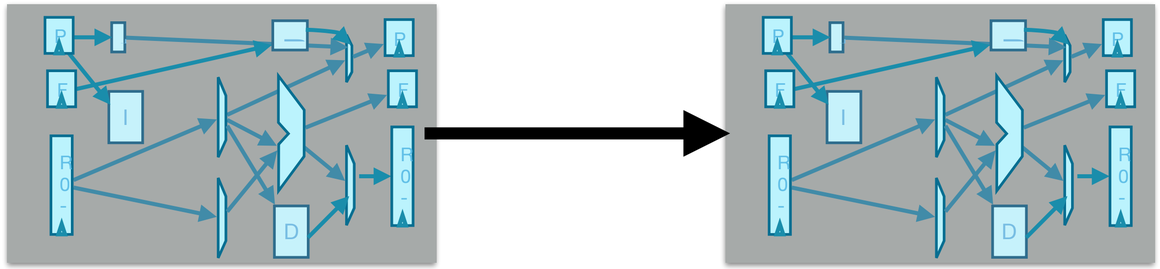

R
I
—
R
F
F
I
R
0
-
R
0
-
D

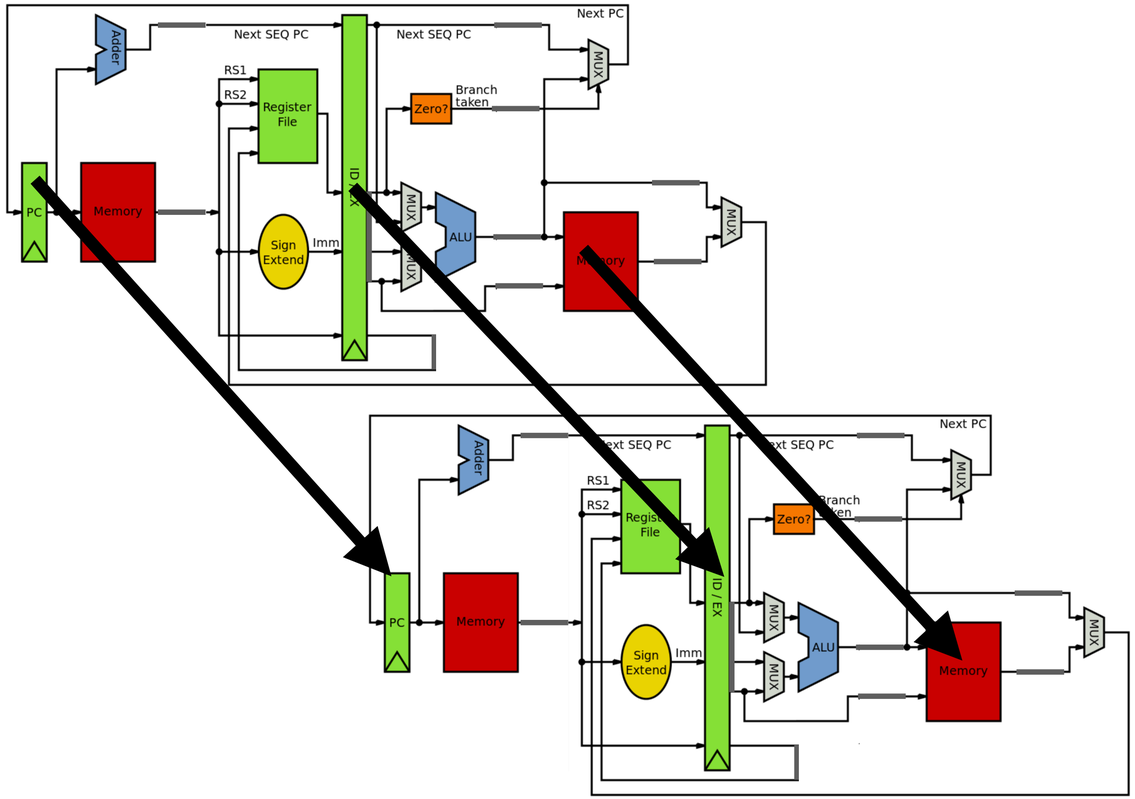

R
I
—
R
F
I
R
0
-
D
R
0
-
R
F
I
R
0
-
D
R
0
-

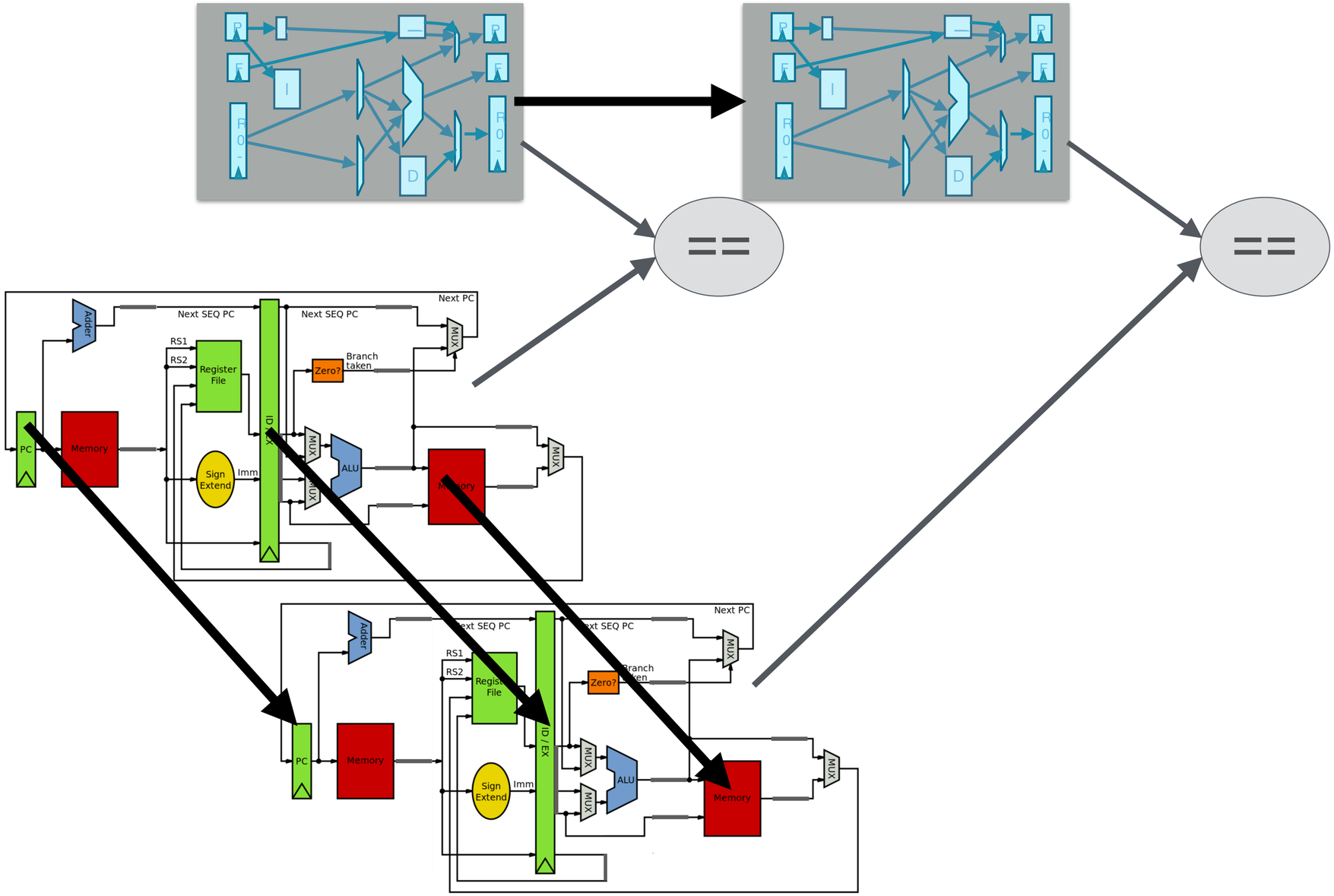

R
F
I
R
F
R
0
-
D
R
0
-


Next PC
Adder
Next SEQ PC
Next SEQ PC
MUX
RS1
RS2
Register File
Zero?
Branch taken
ID / EX
PC
Memory
MUX
ALU
Sign Extend
Imm
MUX
Memory
MUX


Next PC
Adder
Next SEQ PC
Next SEQ PC
MUX
RS1
RS2
Register File
Zero?
Branch taken
ID / EX
PC
Memory
MUX
ALU
Sign Extend
Imm
MUX
Memory
MUX

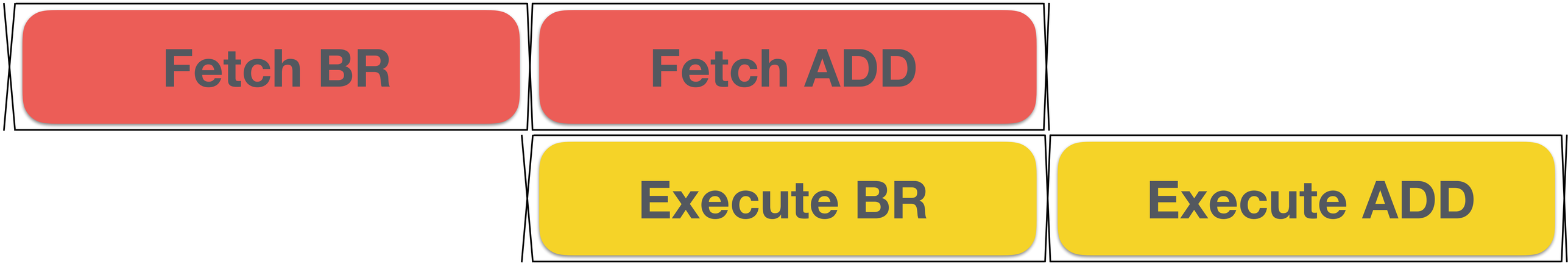

R
F
I
R
F
R
0
-
D
R
F
I
R
F
R
0
-
D


Next PC
Next SEQ PC
Next SEQ PC
Adder
RS1
RS2
Register File
Zero?
Branch taken
MUX
ID / EX
PC
Memory
Sign Extend
Imm
ALU
MUX
MUX
MUX
Next PC
Next SEQ PC
Next SEQ PC
Adder
RS1
RS2
Zero?
Branch taken
MUX
PC
Memory
ID / EX
Sign Extend
Imm
ALU
MUX
MUX
MUX
Memory
MUX

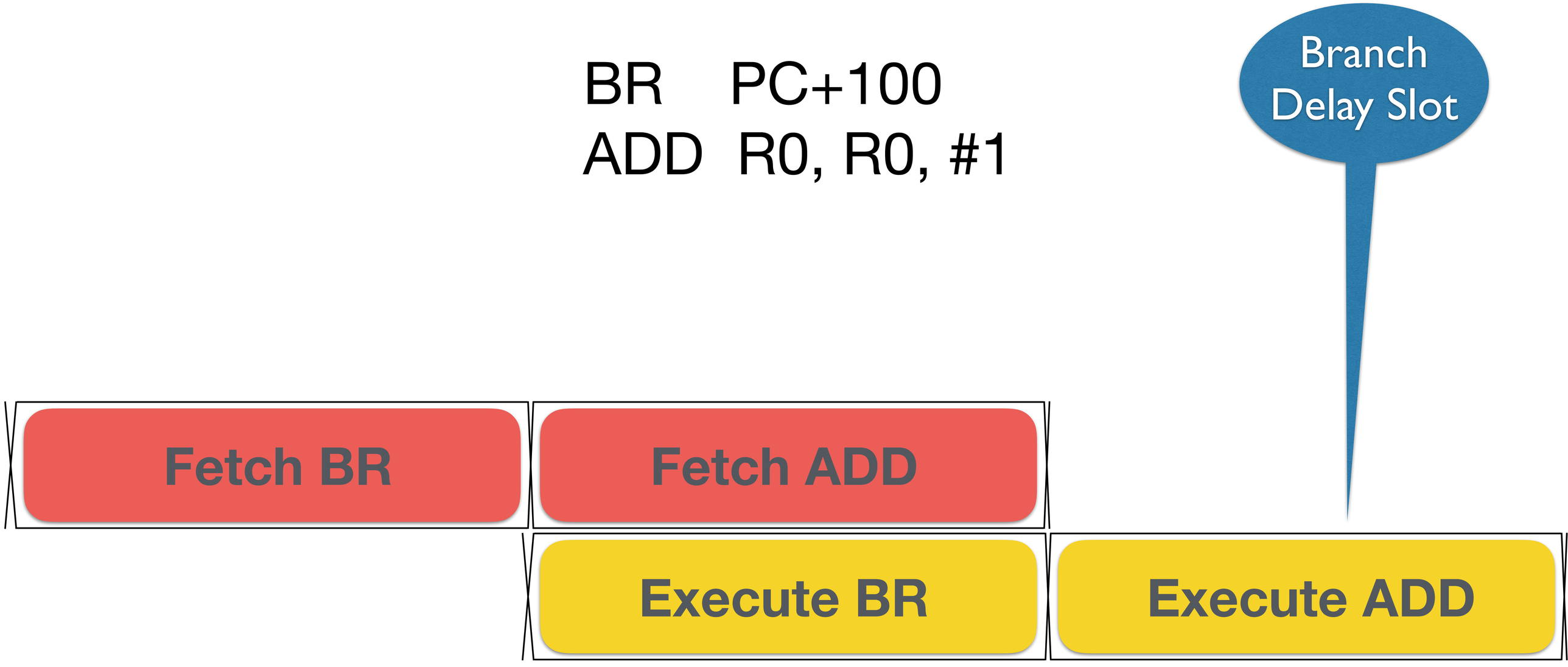

R
F
I
R
F
R0
D
==
==
Next PC
Next SEQ PC
Next SEQ PC
Adder
MUX
RS1
RS2
Register File
Zero?
Branch taken
PC
Memory
ID / EX
MUX
MUX
ALU
Sign Extend
Imm
Memory
MUX
Next PC
Next SEQ PC
Next SEQ PC
Adder
MUX
RS1
RS2
Register File
Zero?
Branch taken
PC
Memory
ID / EX
MUX
MUX
ALU
Sign Extend
Imm
Memory
MUX

```
BR    PC+100
ADD   R0, R0, #1
```

```
BR    PC+100
ADD   R0, R0, #1
```



```
BR    PC+100
ADD   R0, R0, #1
```

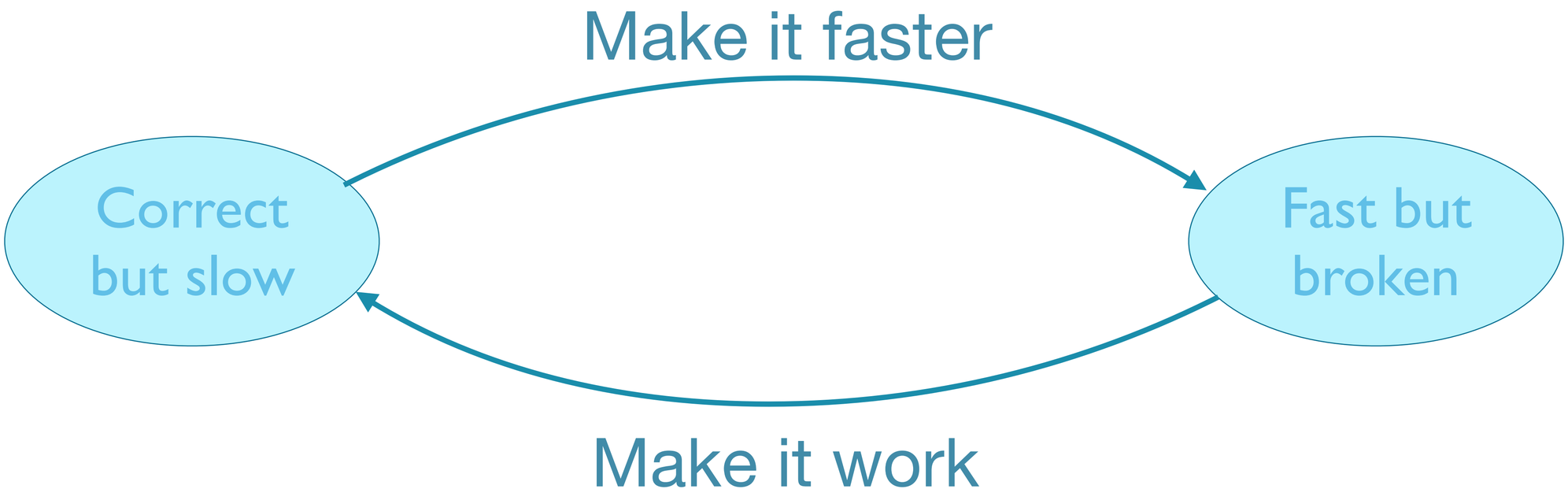


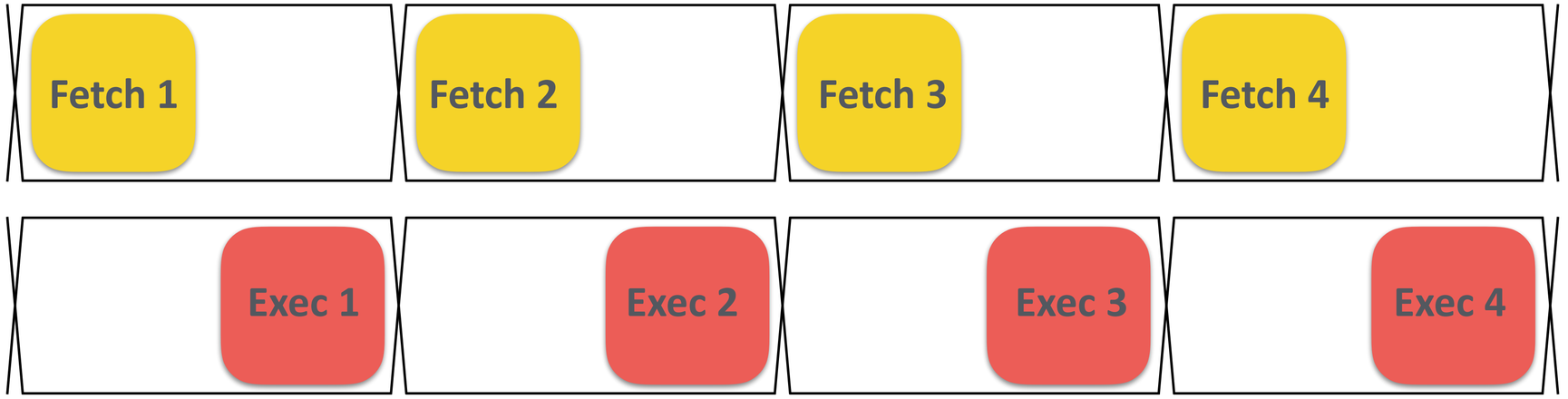
What tests to run?

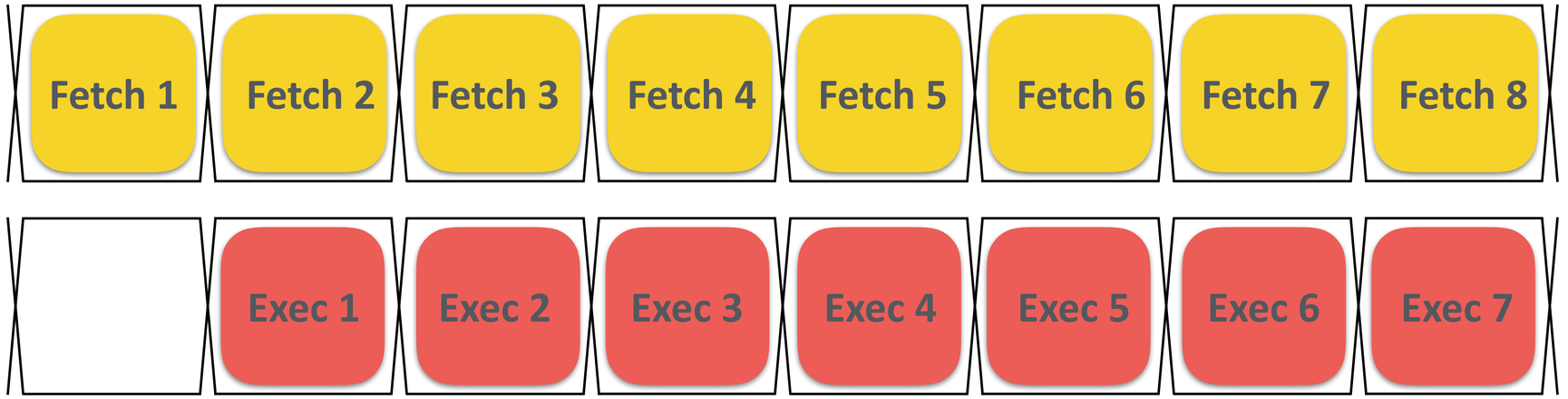
All the single-stage tests

Branches

All “interesting” pairs of instructions


Make it faster
Correct
but slow
Fast but
broken
Make it work


Fetch 1
Fetch 2
Fetch 3
Fetch 4
Exec 1
Exec 2
Exec 3
Exec 4

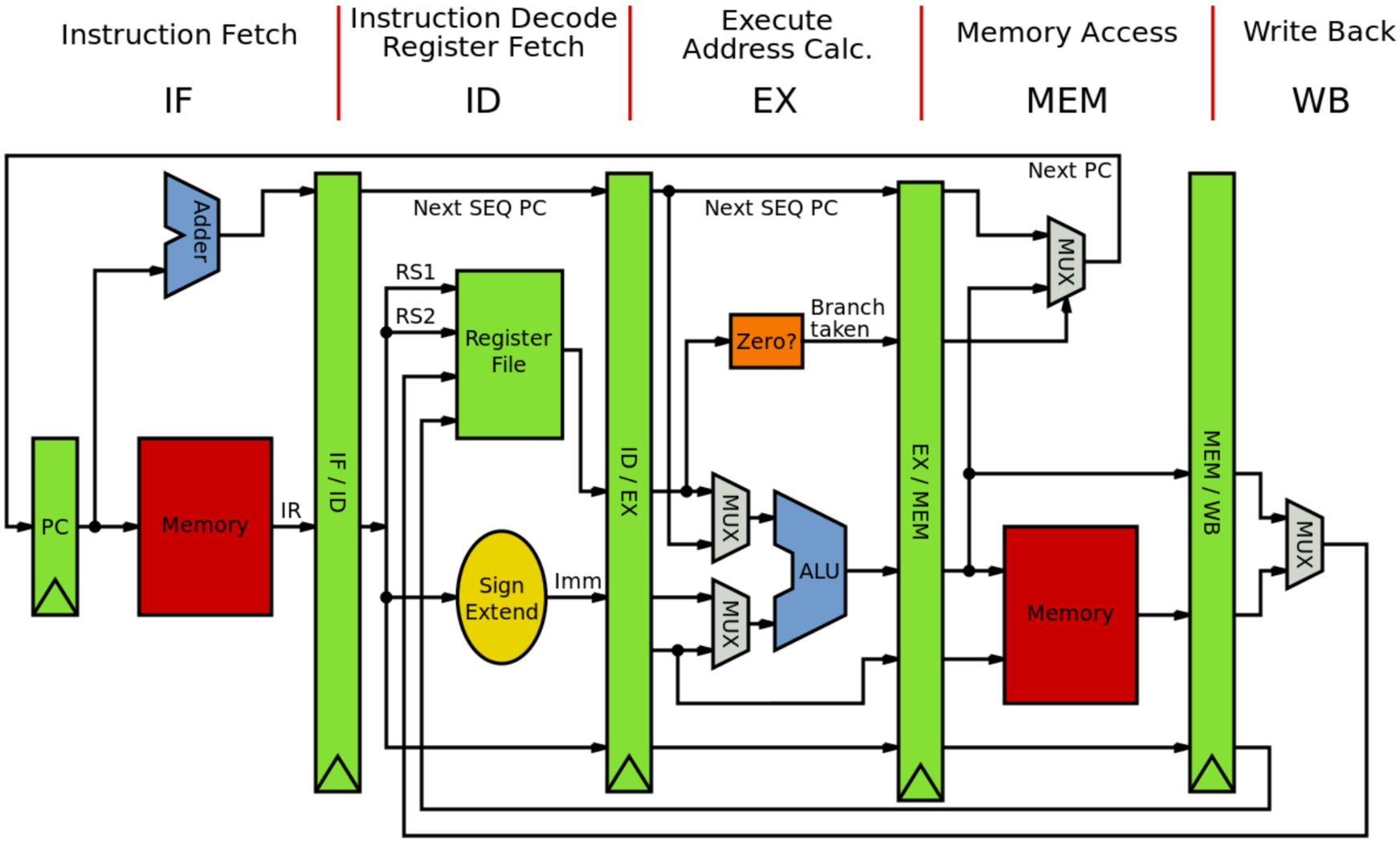

Fetch 1
Fetch 2
Fetch 3
Fetch 4
Fetch 5
Fetch 6
Fetch 7
Fetch 8
Exec 1
Exec 2
Exec 3
Exec 4
Exec 5
Exec 6
Exec 7


Fetch 1
Decode 1
Exec 1
Mem 1
Write 1
Fetch 2
Decode 2
Exec 2
Mem 1
Write 1

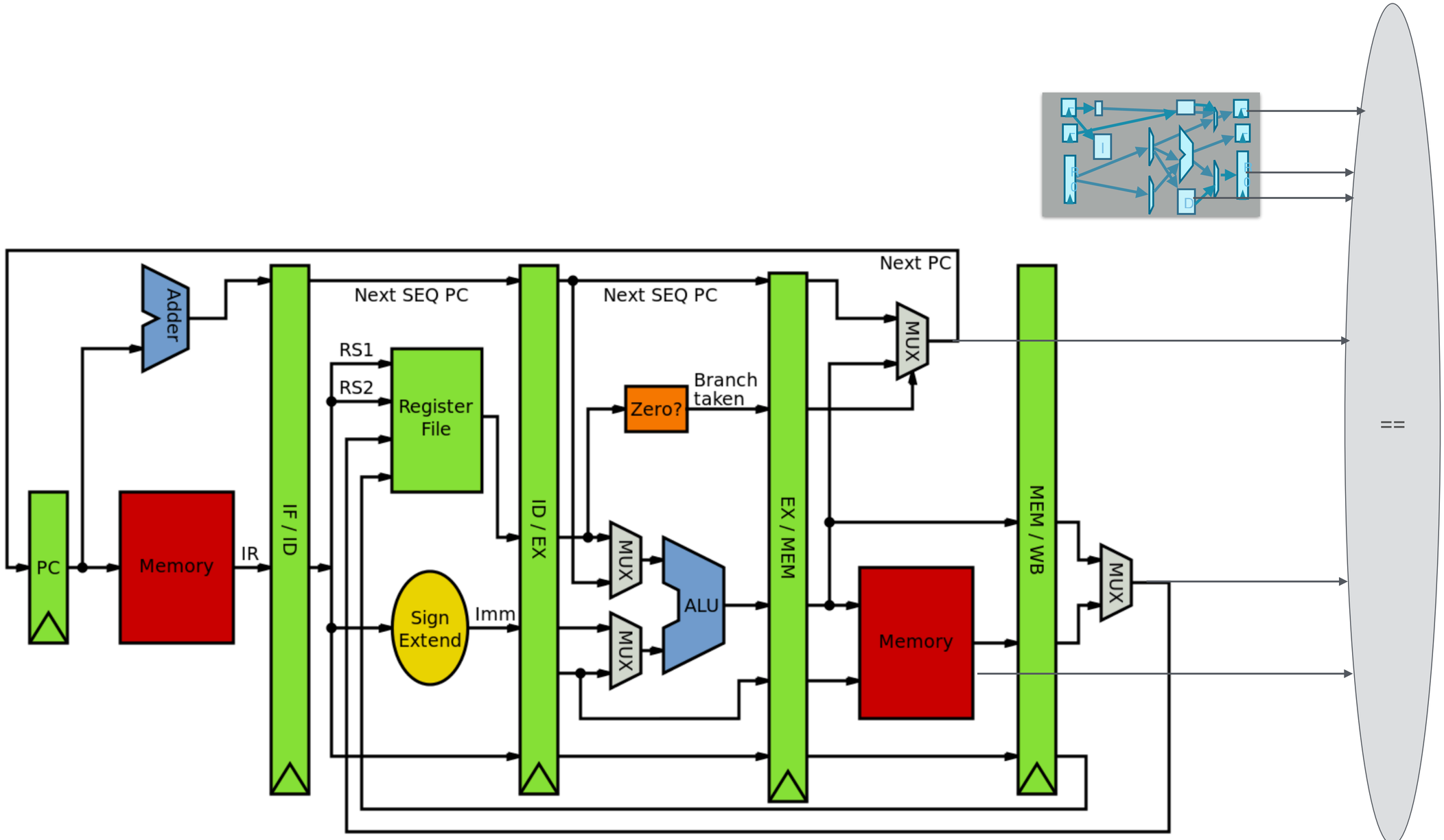

Fetch 1
Decode 1
Exec 1
Mem 1
Write 1
Fetch 2
Decode 2
Exec 2
Mem 1
Write 1


Fetch 1
Fetch 2
Fetch 3
Fetch 4
Fetch 5
Fetch 6
Fetch 7
Fetch 8
Fetch 9
Fetch 10
Decode 1
Decode 2
Decode 3
Decode 4
Decode 5
Decode 6
Decode 7
Decode 8
Decode 9
Exec 1
Exec 2
Exec 3
Exec 4
Exec 5
Exec 6
Exec 7
Exec 8
Mem 1
Mem 2
Mem 3
Mem 4
Mem 5
Mem 6
Mem 7
Write 1
Write 2
Write 3
Write 4
Write 5
Write 6


Fetch 1
Decode 1
Exec 1
Mem 1
Write 1
Fetch 2
Decode 2
Exec 2
Mem 1
Write 1
Fetch 1
Fetch 2
Fetch 3
Fetch 4
Fetch 5
Fetch 6
Fetch 7
Fetch 8
Fetch 9
Fetch 10
Decode 1
Decode 2
Decode 3
Decode 4
Decode 5
Decode 6
Decode 7
Decode 8
Decode 9
Exec 1
Exec 2
Exec 3
Exec 4
Exec 5
Exec 6
Exec 7
Exec 8
Mem 1
Mem 2
Mem 3
Mem 4
Mem 5
Mem 6
Mem 7
Write 1
Write 2
Write 3
Write 4
Write 5
Write 6

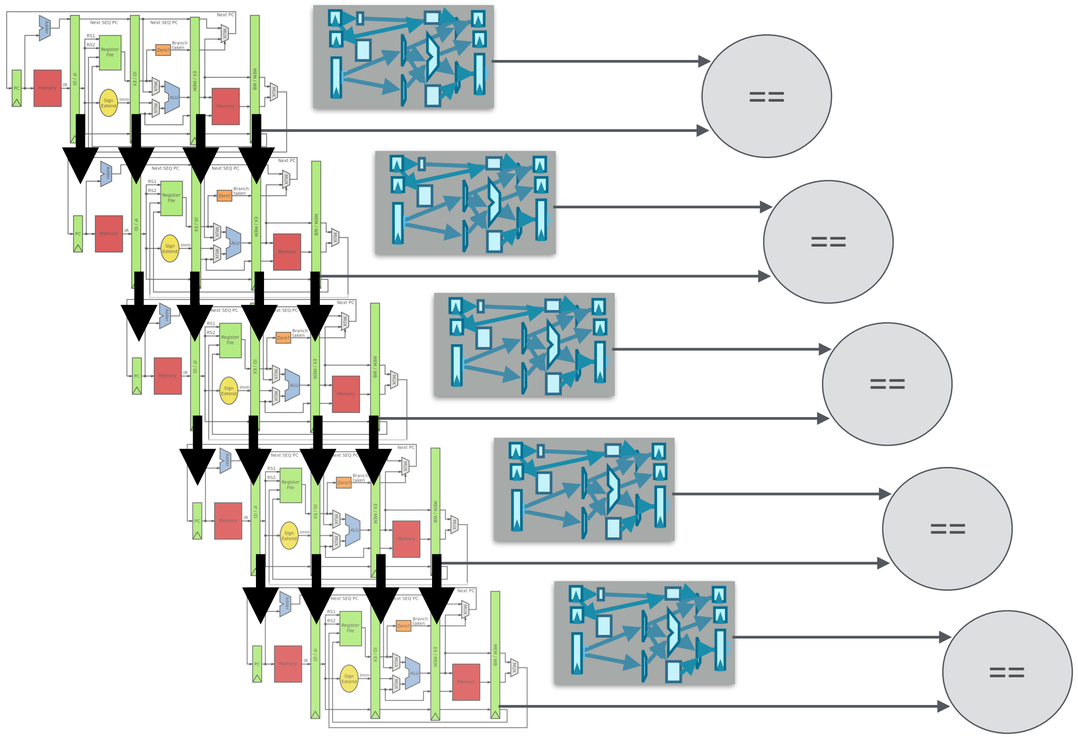


https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg



https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

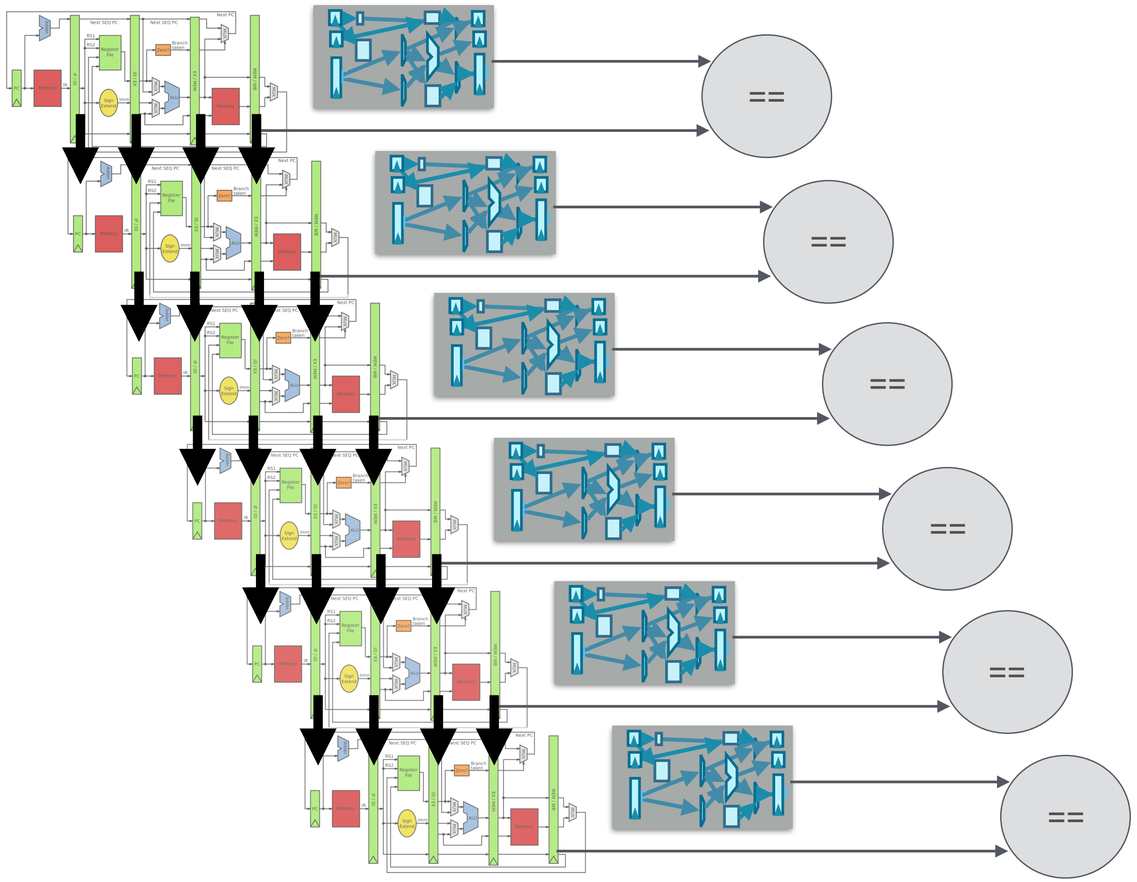
Bounded Model Checking



https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

Tools: Yosys http://www.clifford.at/yosys/
Talks: http://www.clifford.at/papers/2017/smtbmc-sby/slides.pdf
Blogs: https://zipcpu.com/blog/2017/10/19/formal-intro.html

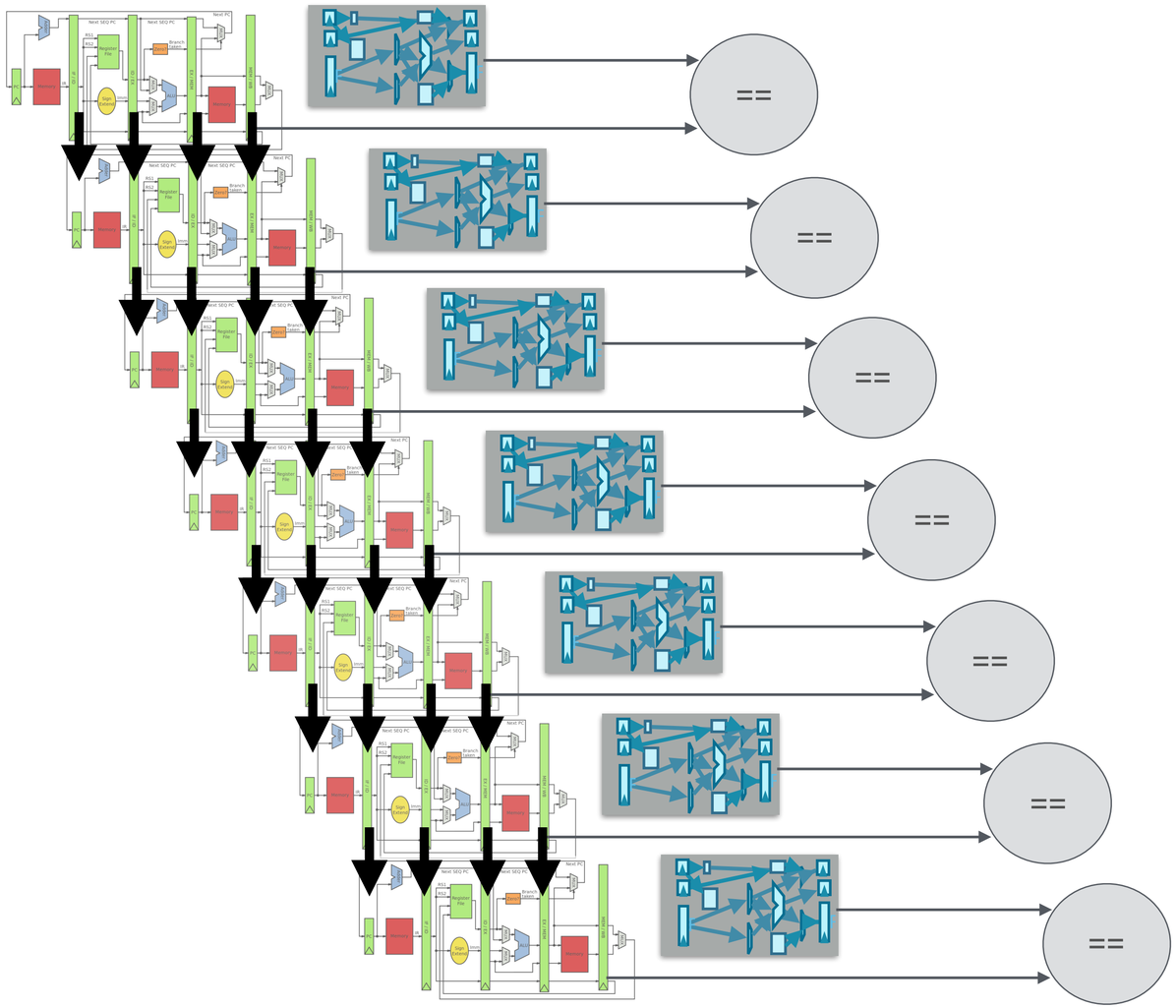
Bounded Model Checking



https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

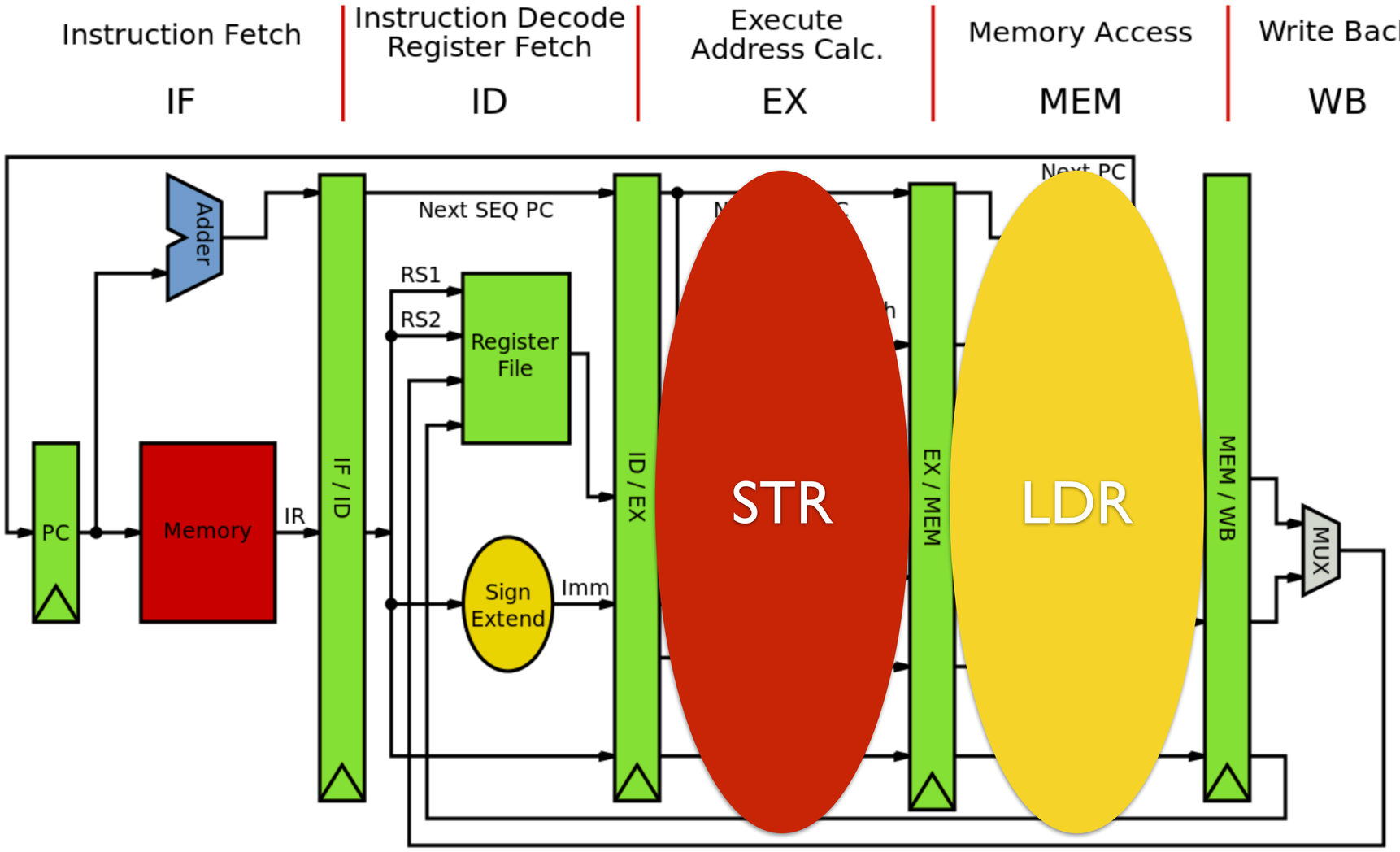
Tools: Yosys http://www.clifford.at/yosys/
Talks: http://www.clifford.at/papers/2017/smtbmc-sby/slides.pdf
Blogs: https://zipcpu.com/blog/2017/10/19/formal-intro.html

Bounded Model Checking



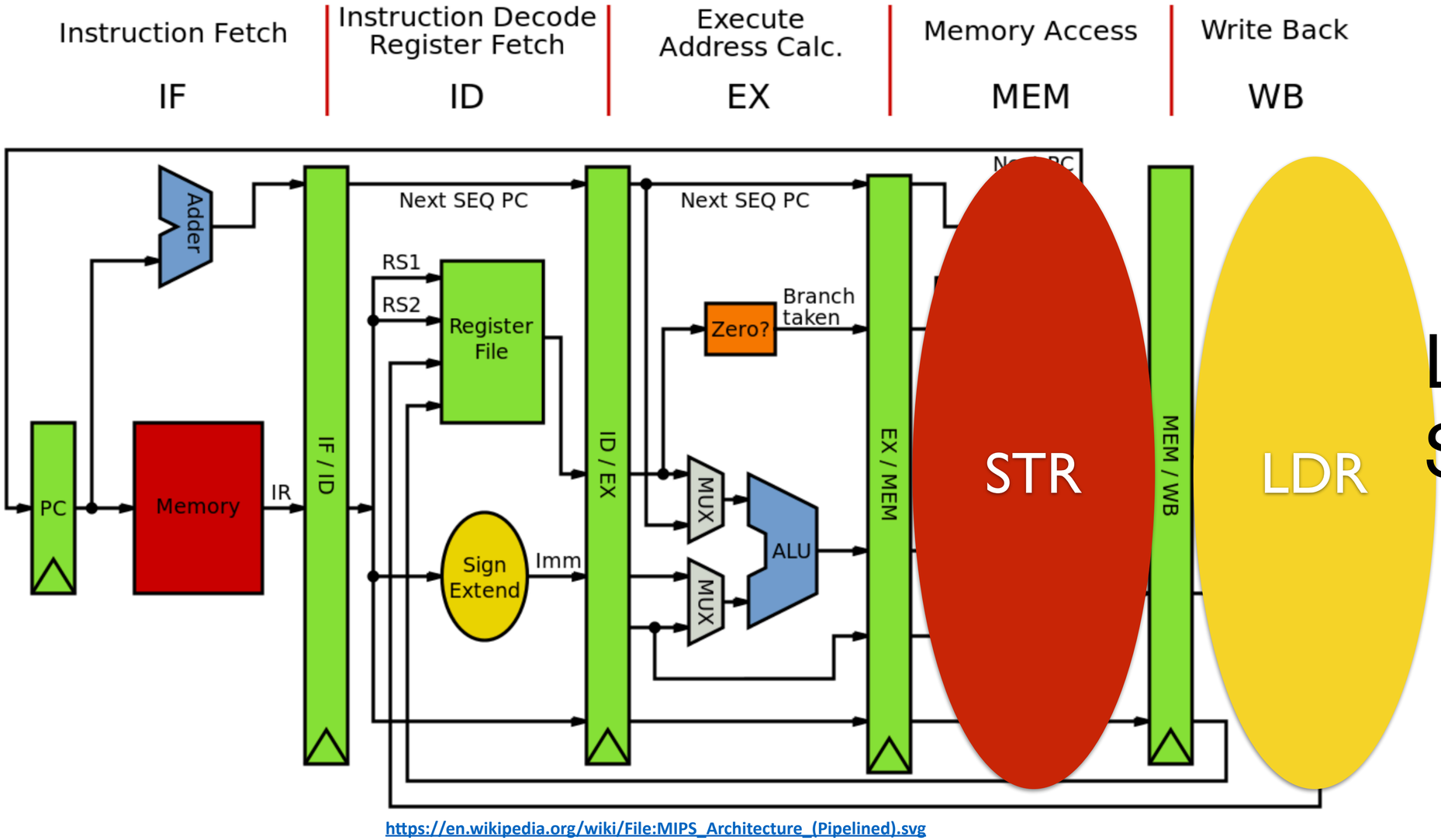
https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

Tools: Yosys http://www.clifford.at/yosys/
Talks: http://www.clifford.at/papers/2017/smtbmc-sby/slides.pdf
Blogs: https://zipcpu.com/blog/2017/10/19/formal-intro.html



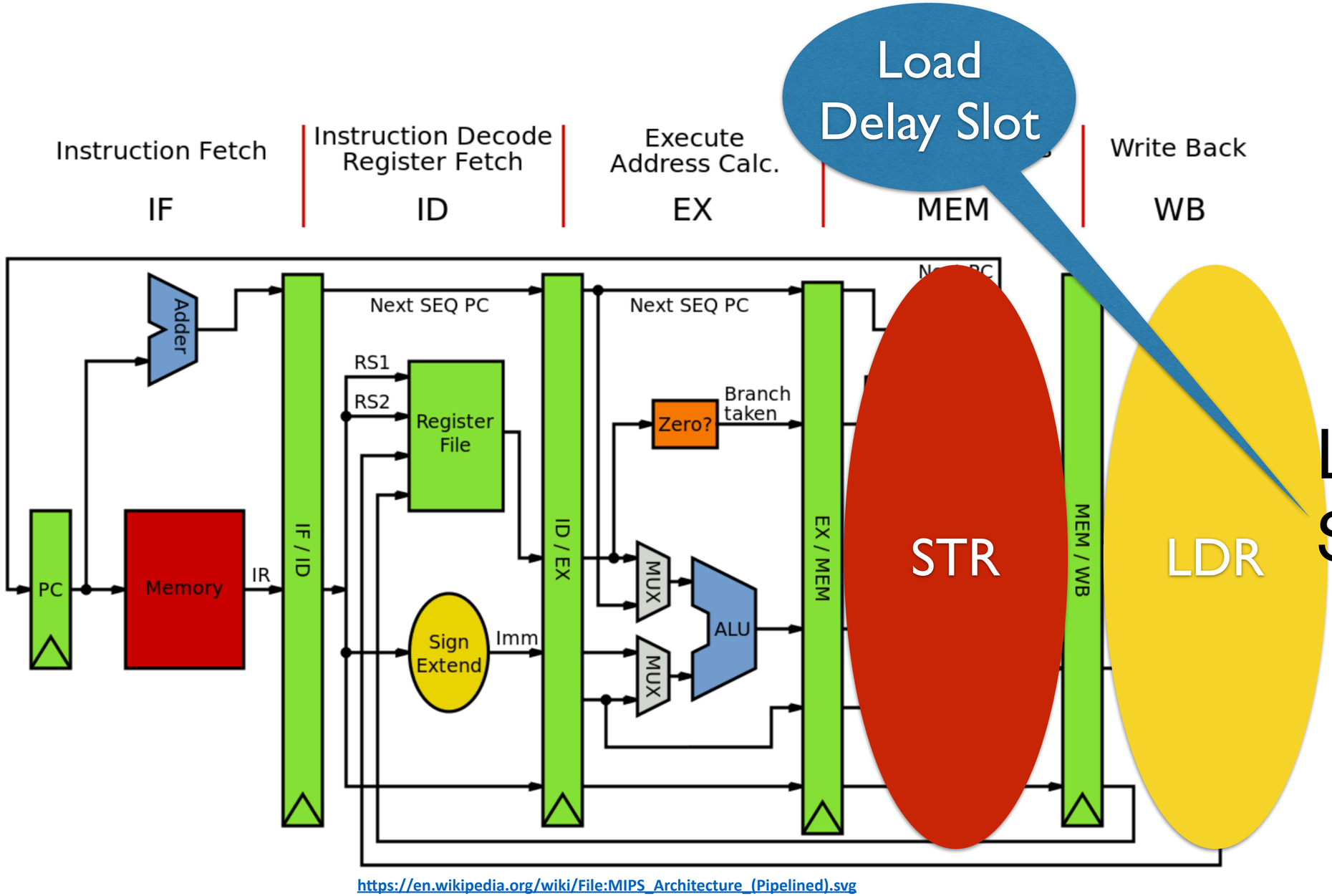
https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

```
LDR   R0, [R1]
STR   R0, [R2]
```



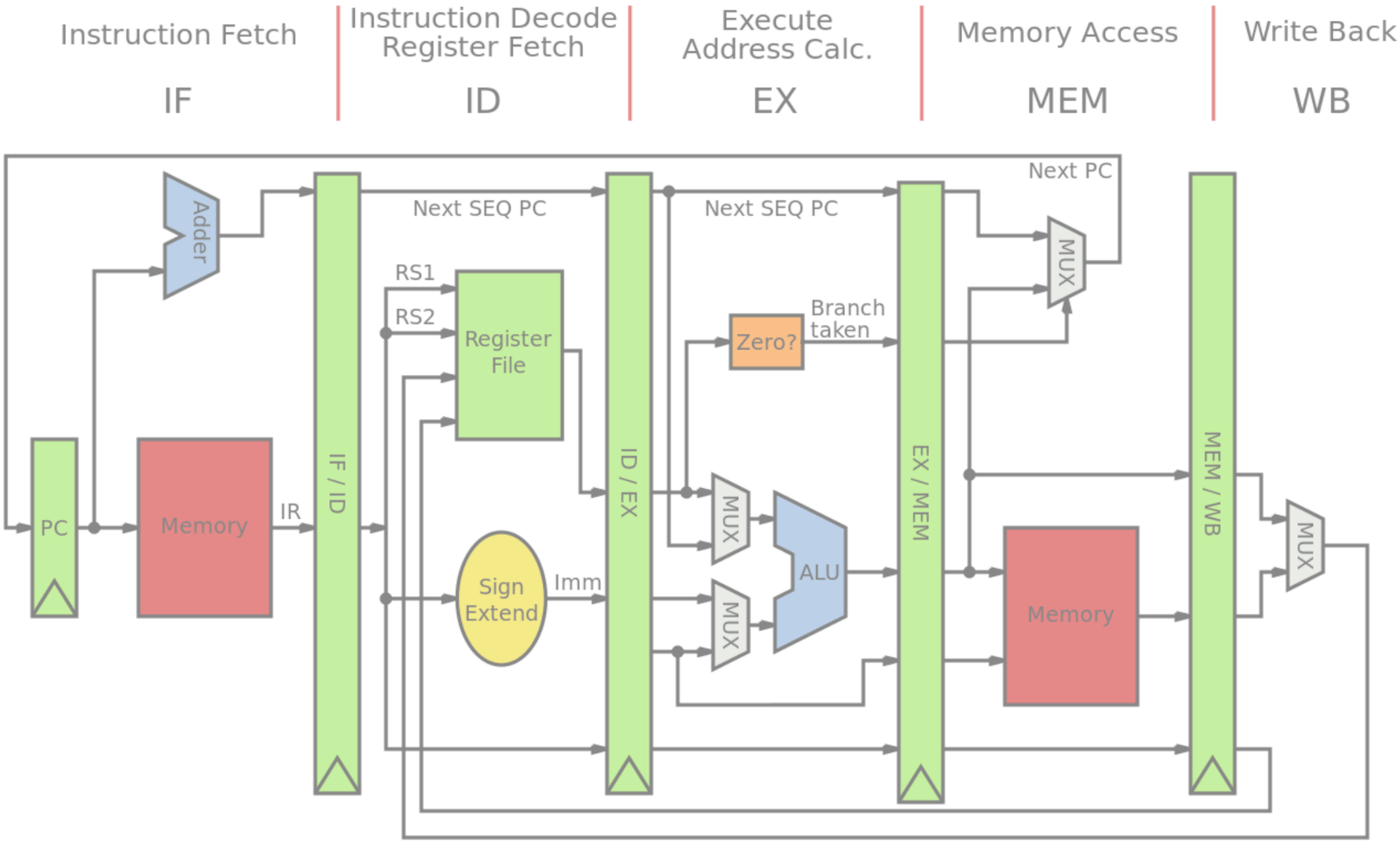
https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

```
LDR   R0, [R1]
STR   R0, [R2]
```

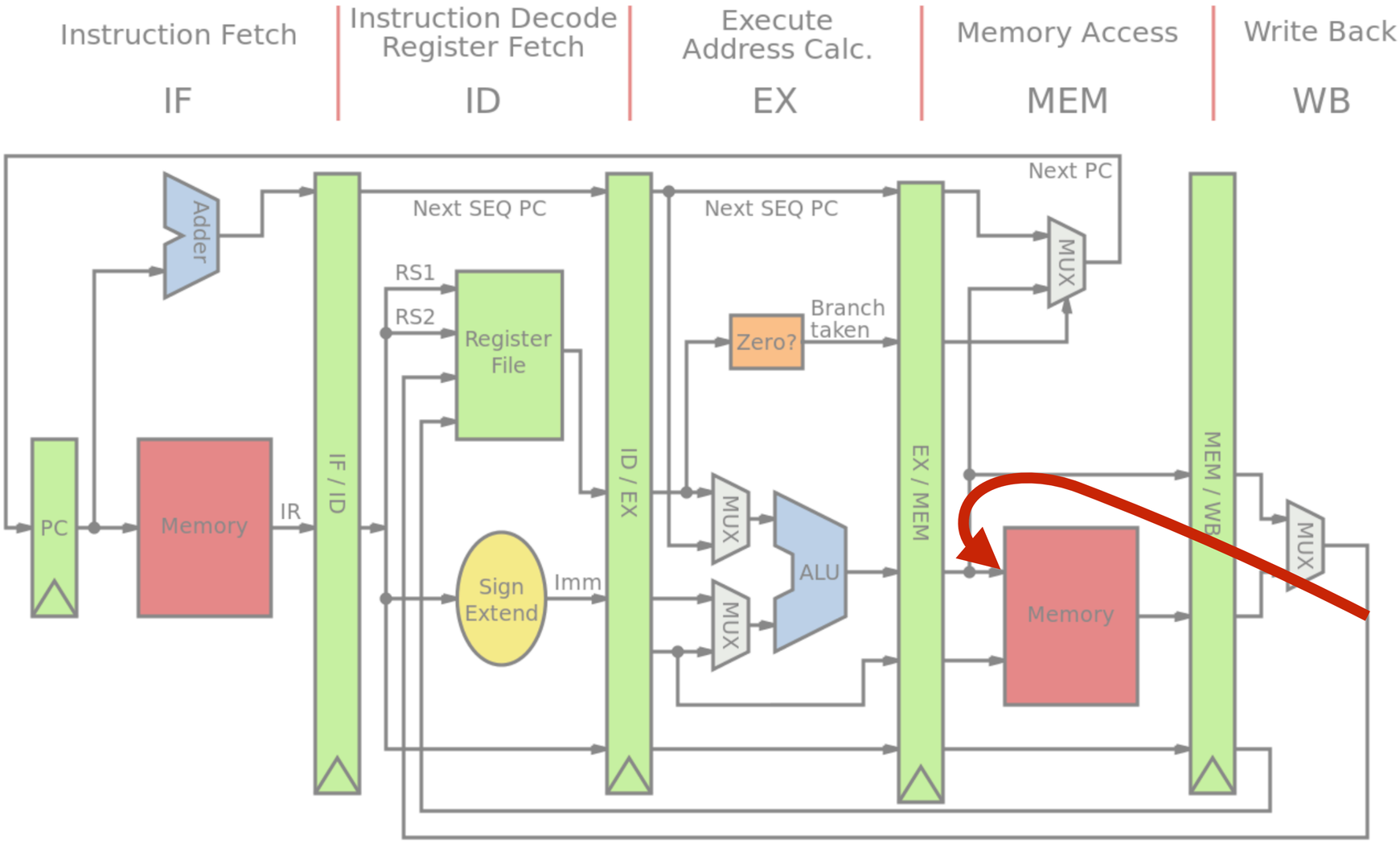


https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

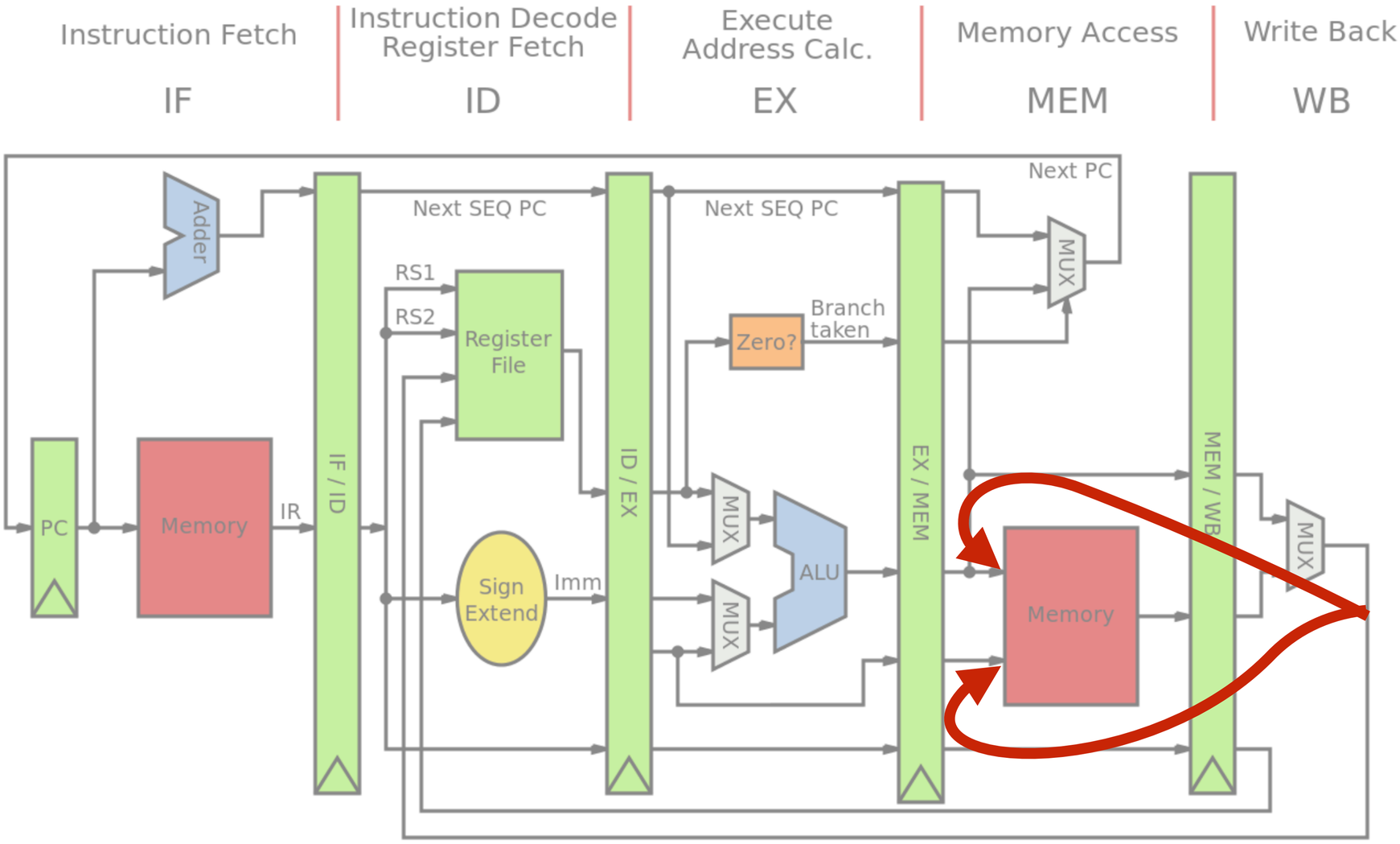
```
LDR   R0, [R1]
STR   R0, [R2]
```



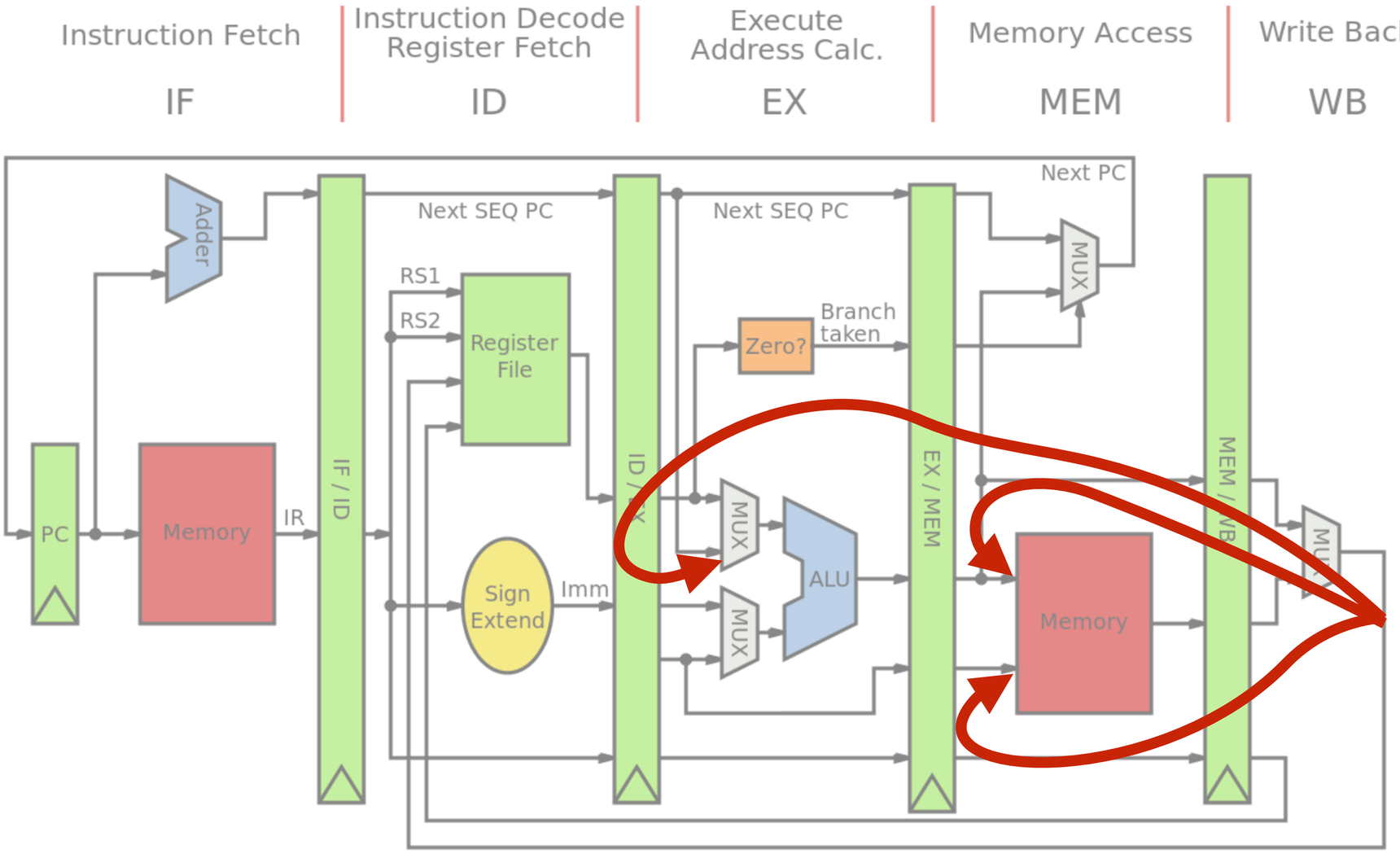
https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg



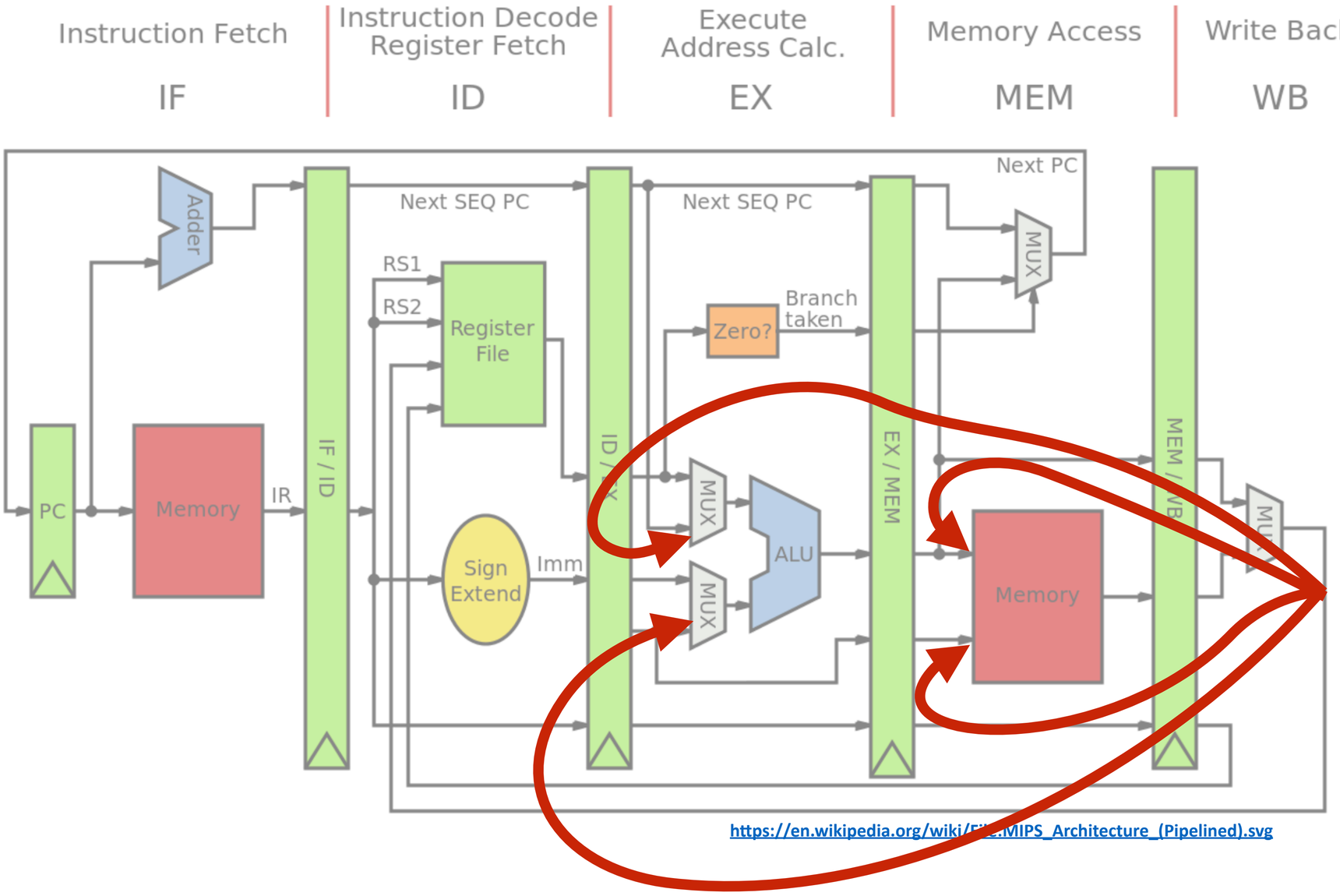
https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg



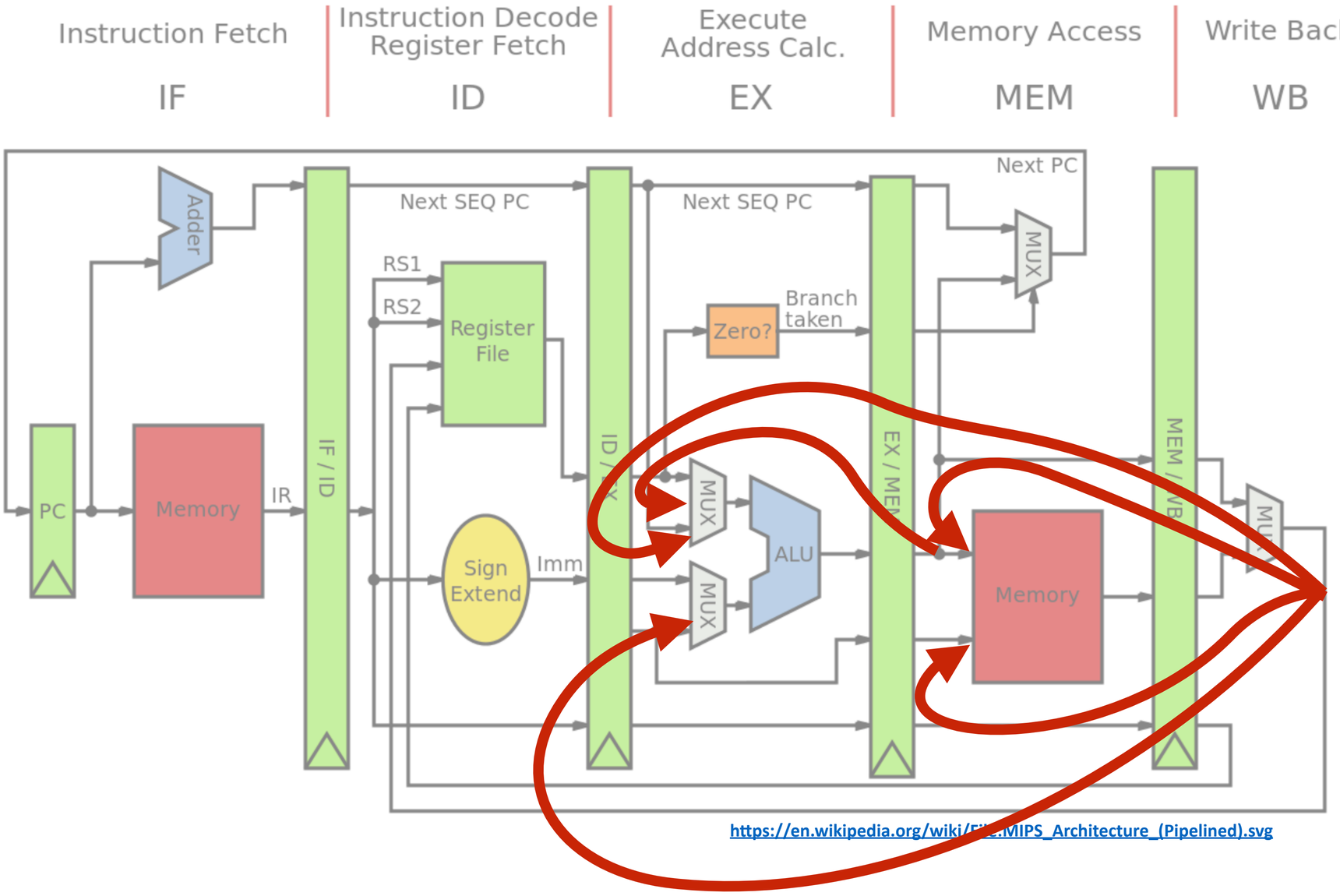
https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg



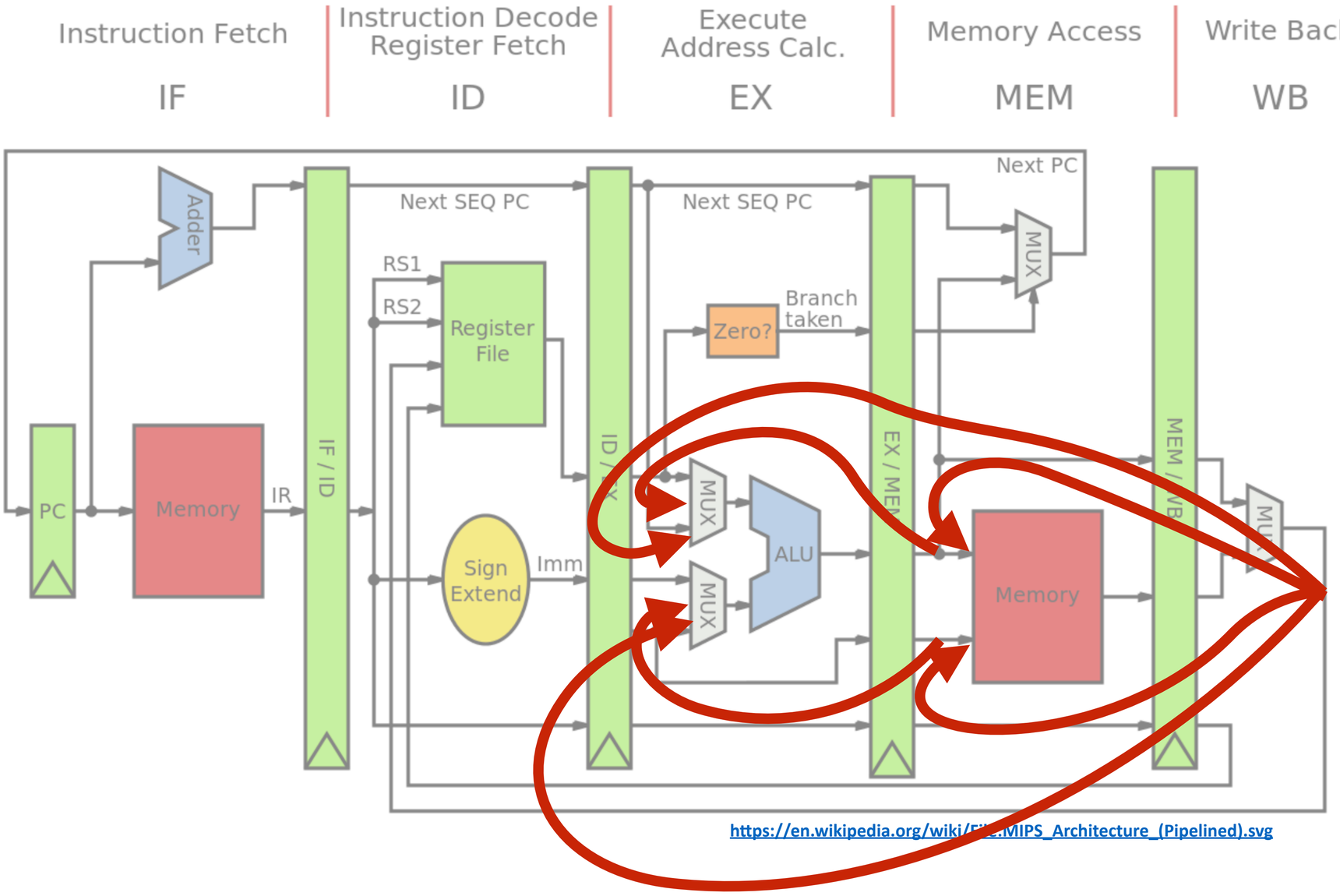
https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg



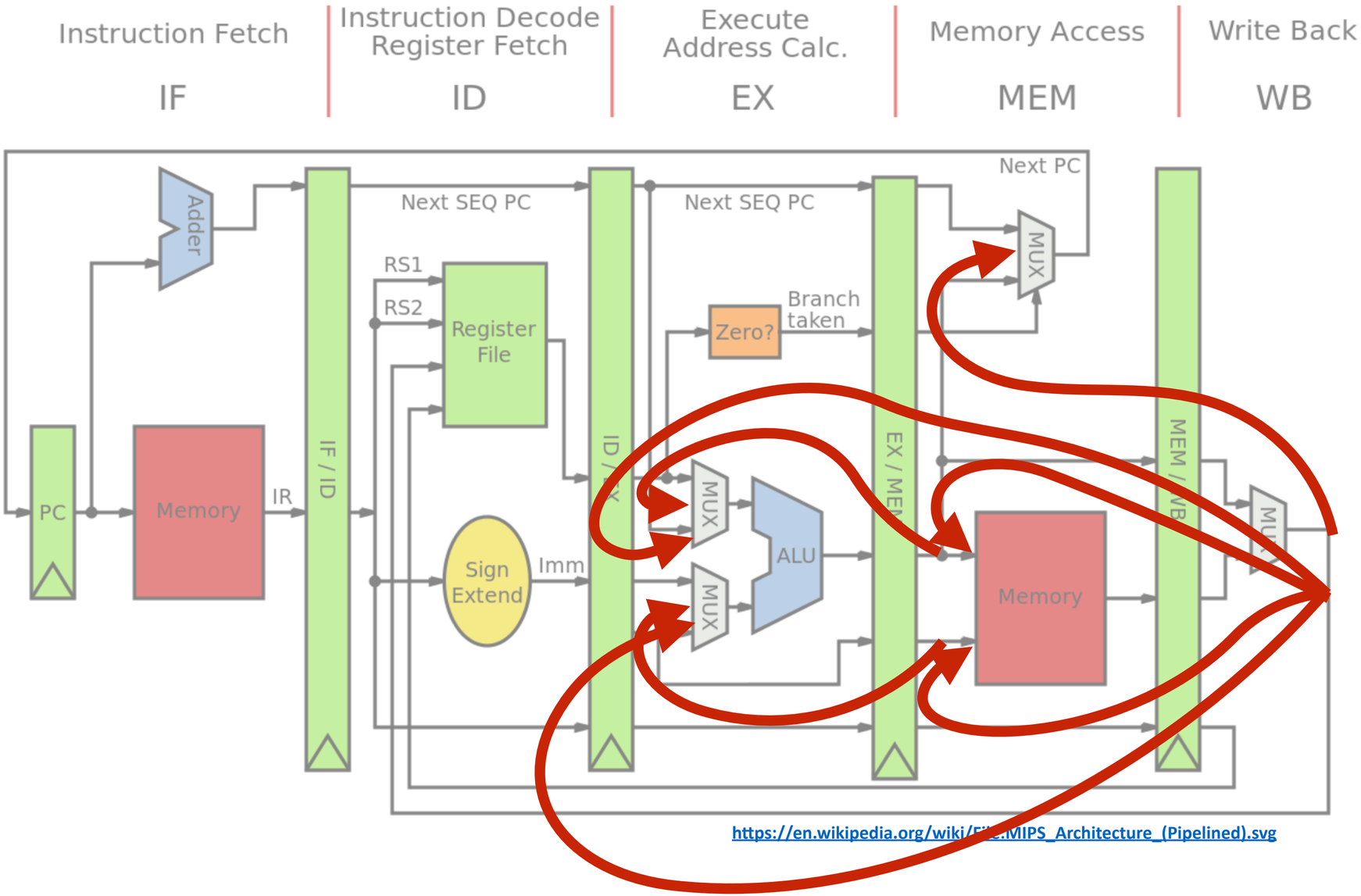
https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg



https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg



https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg



https://en.wikipedia.org/wiki/File:MIPS_Architecture_(Pipelined).svg

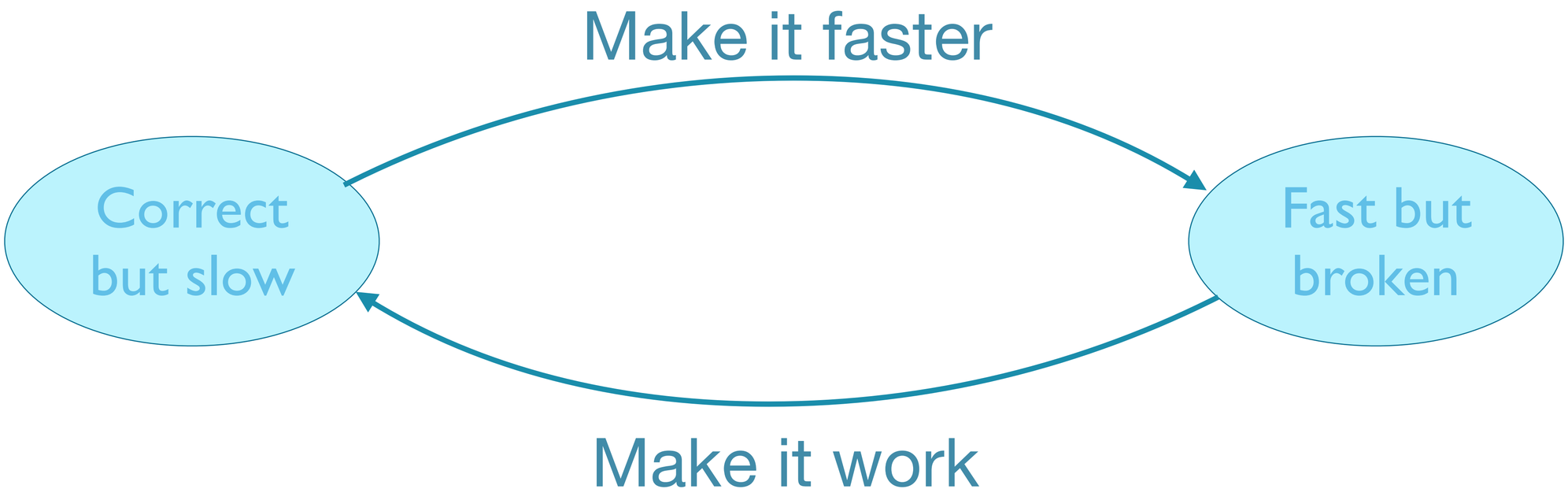
What tests to run?

All the single-stage tests

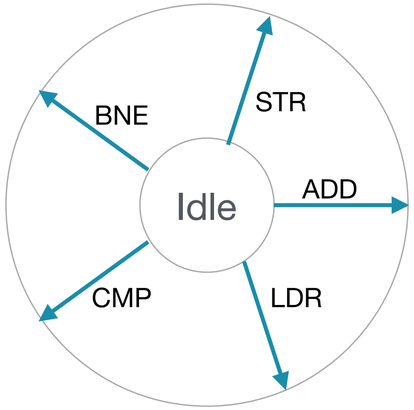
All the two-stage tests

Load followed by <anything>

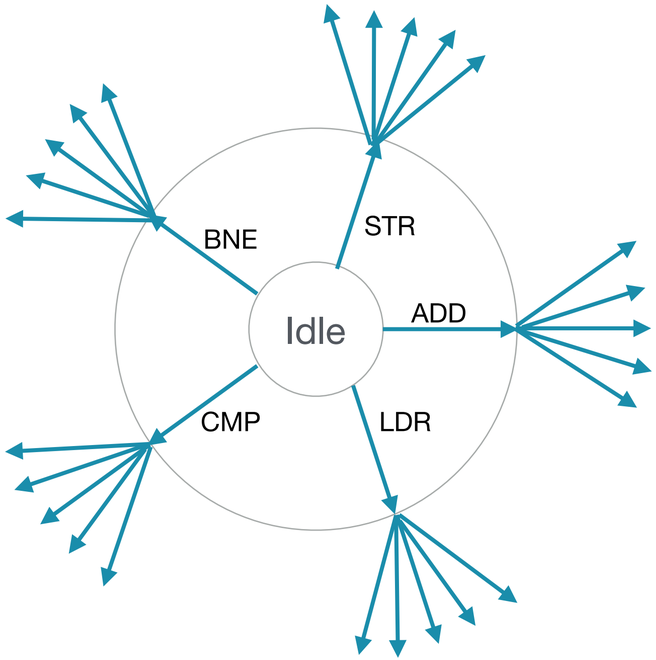
All “interesting” sequences of 5 instructions


Make it faster
Correct
but slow
Fast but
broken
Make it work

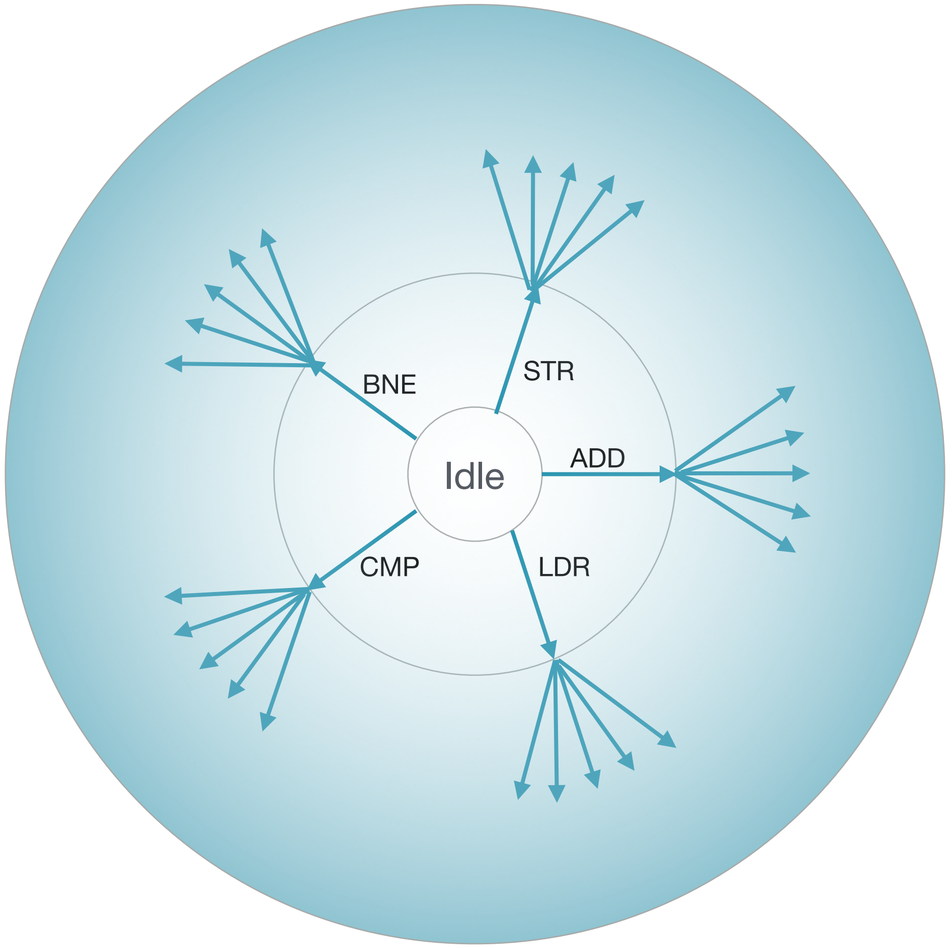
Formal verification is breadth first



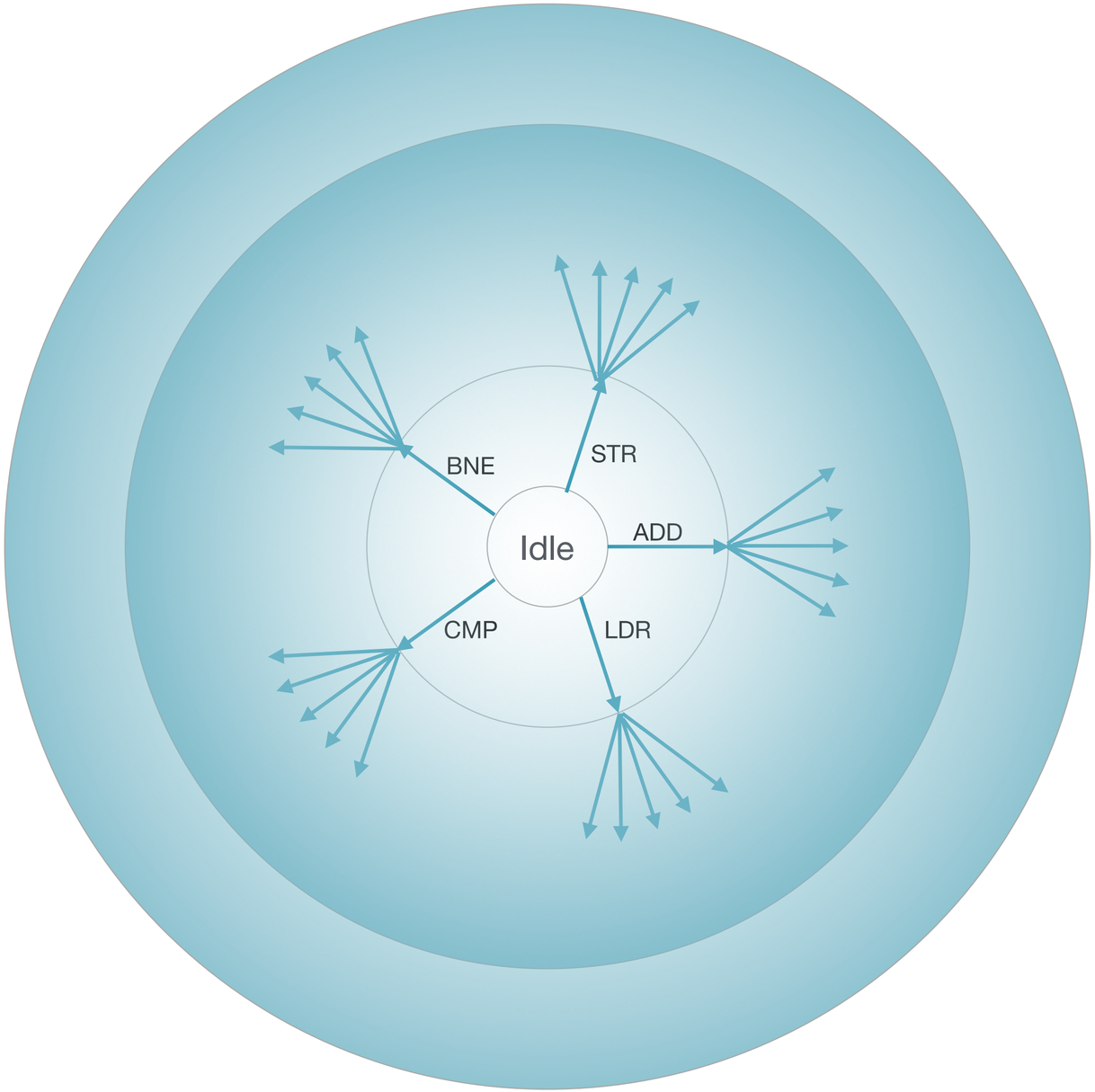
Formal verification is breadth first



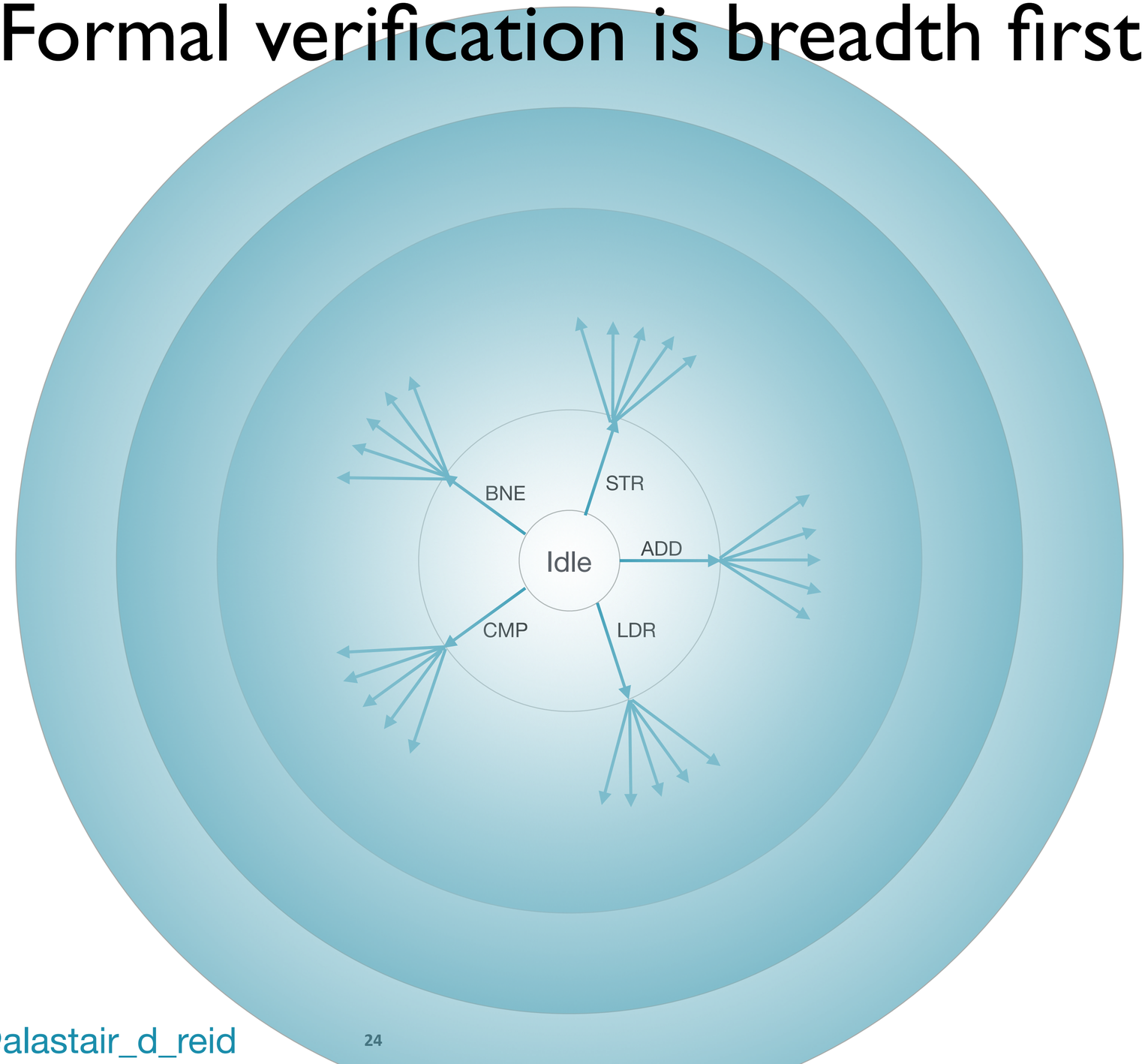
Formal verification is breadth first



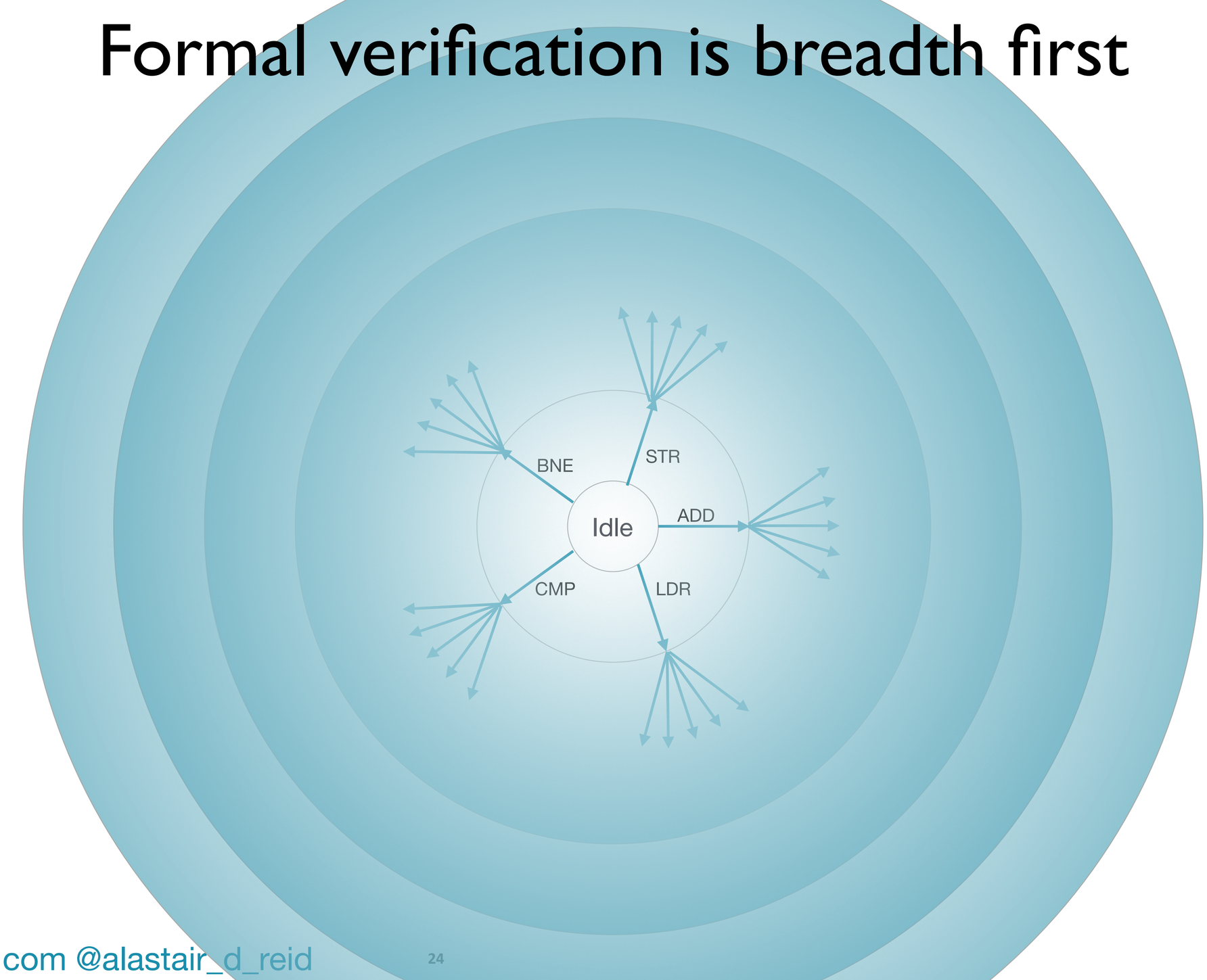
Formal verification is breadth first



Formal verification is breadth first

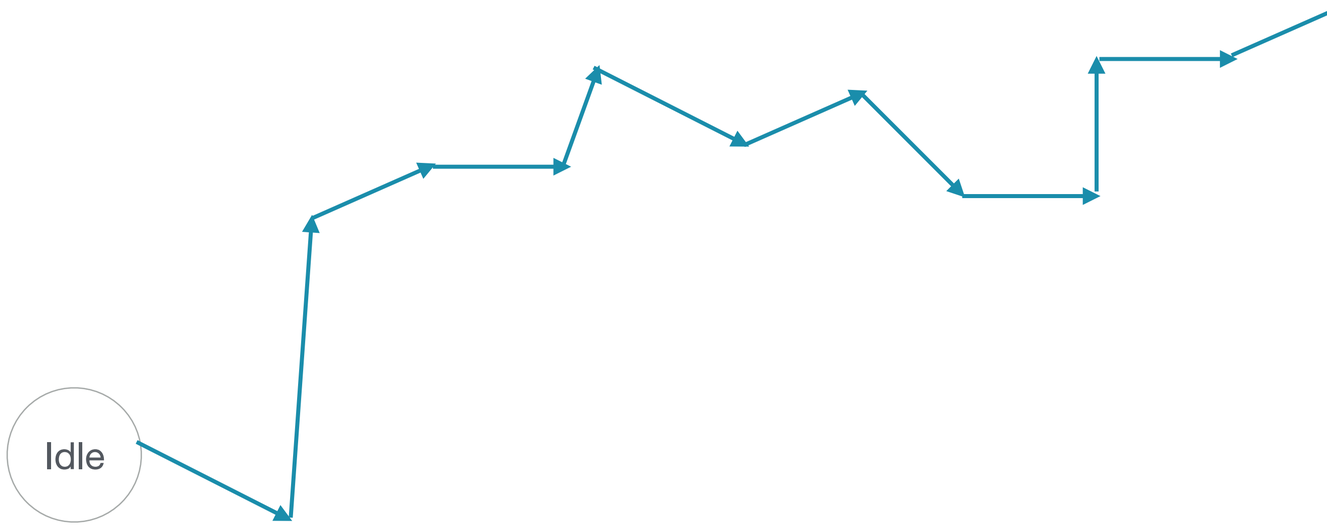


Formal verification is breadth first

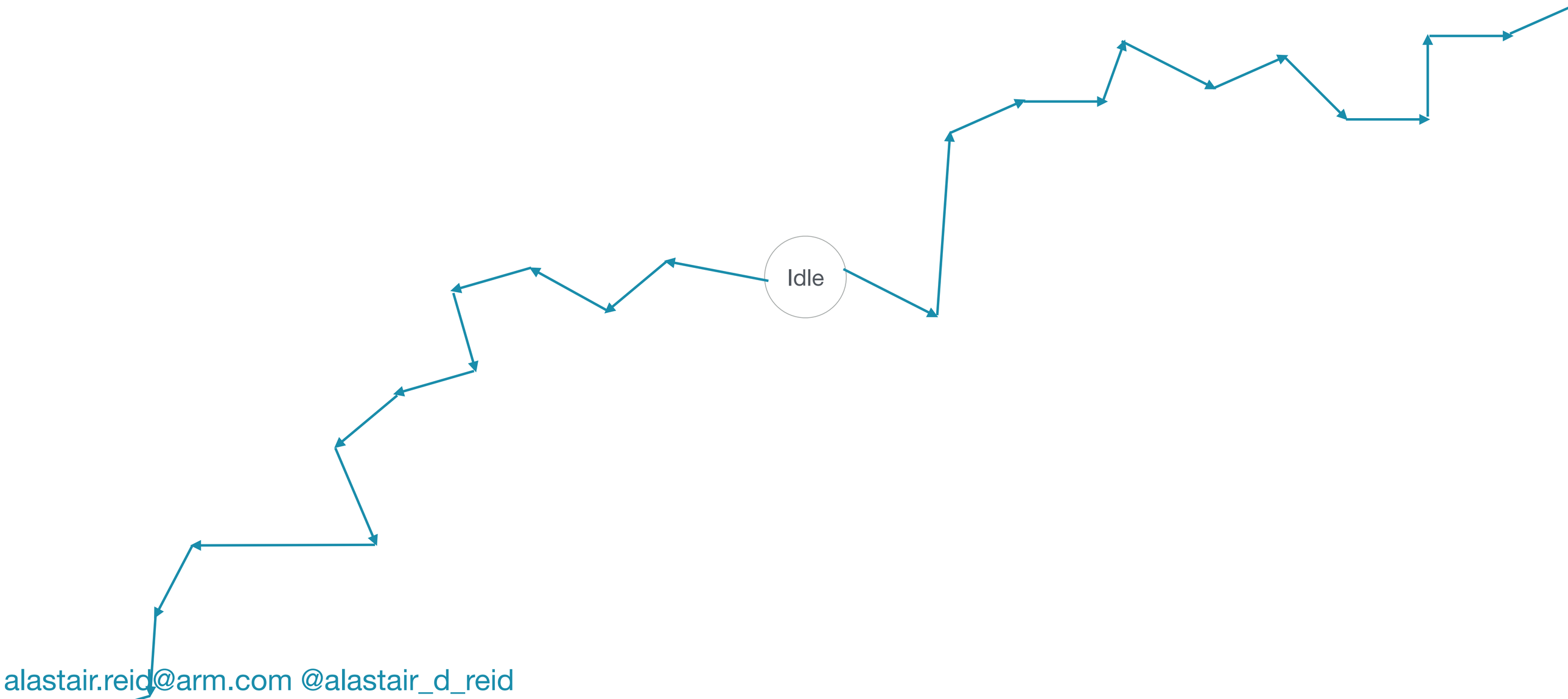


Testing is depth-first

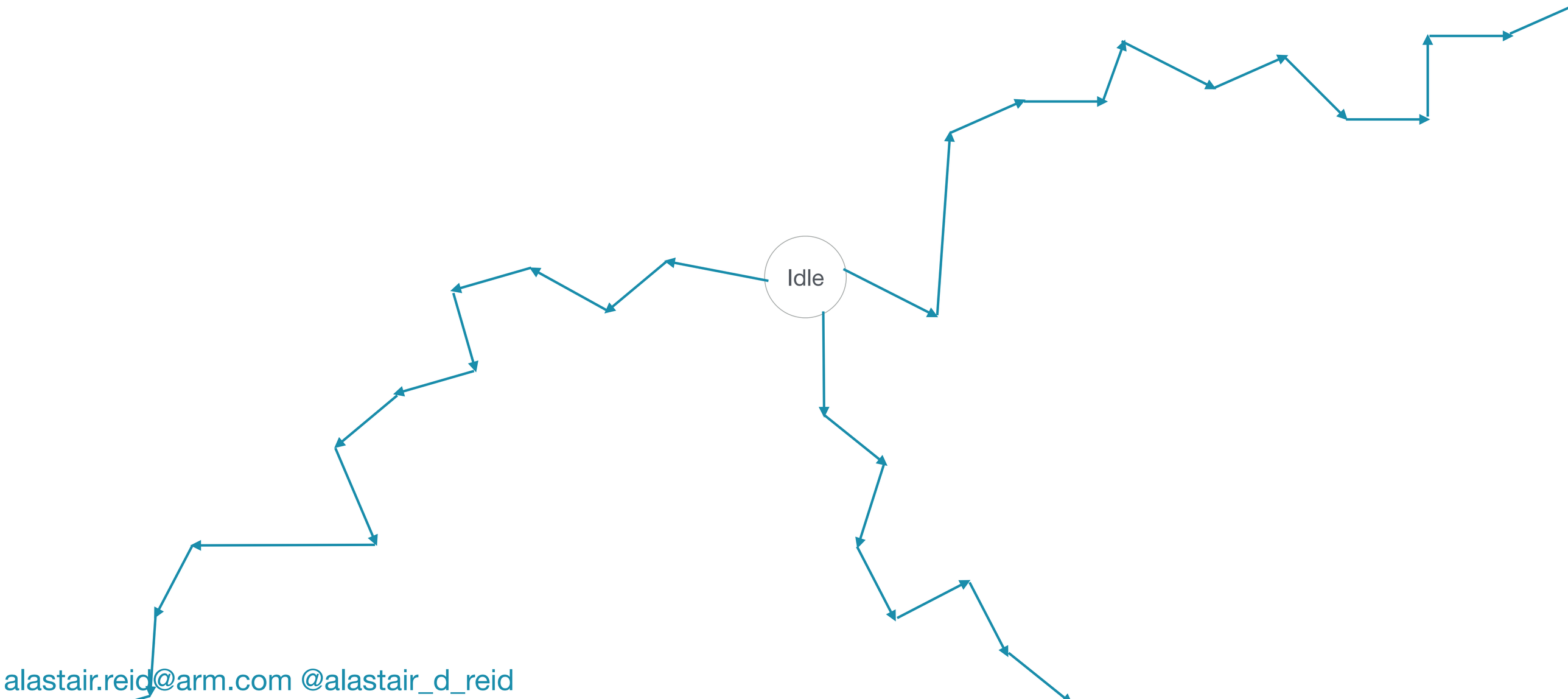
Testing is depth-first



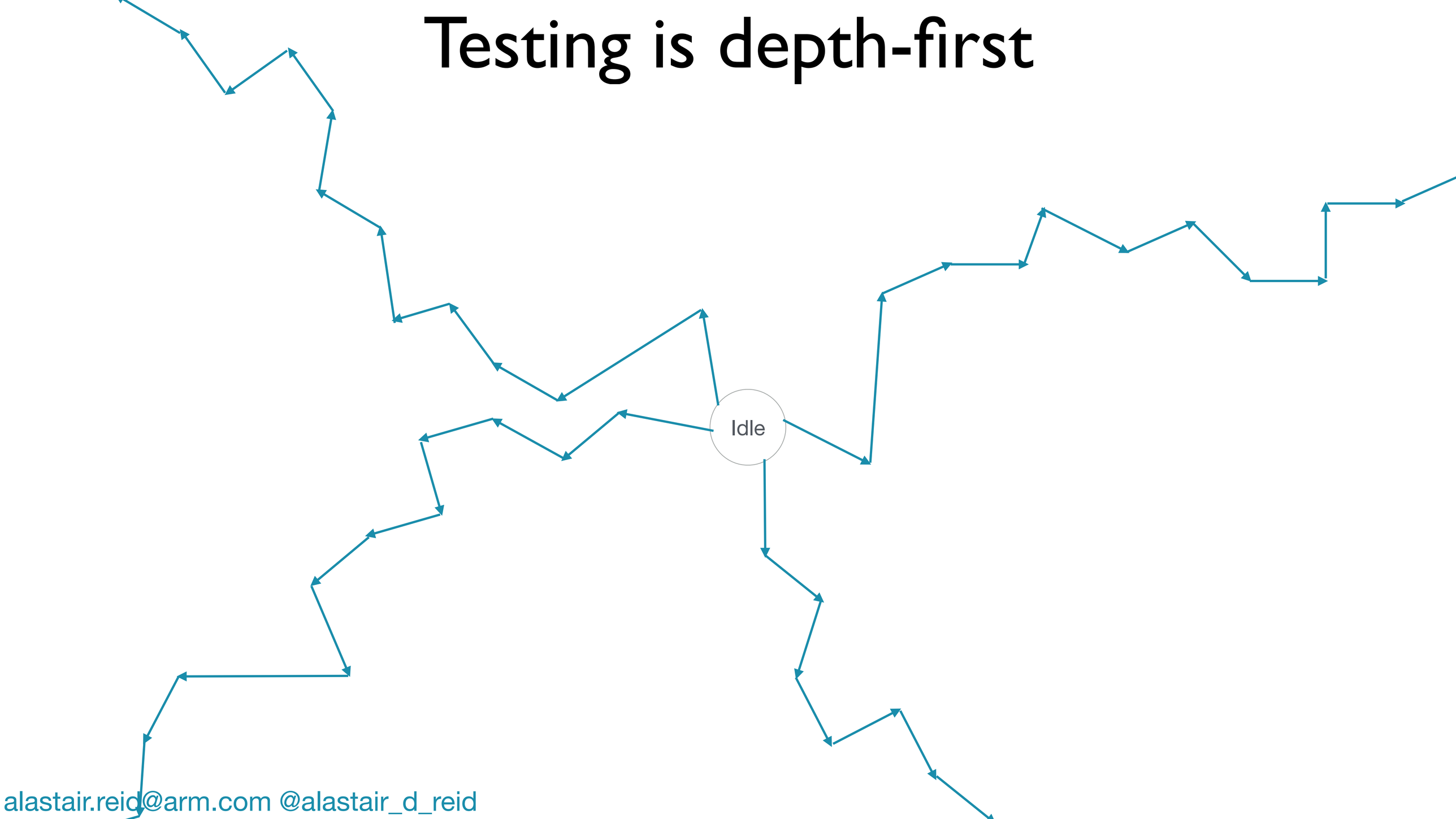
Testing is depth-first



Testing is depth-first

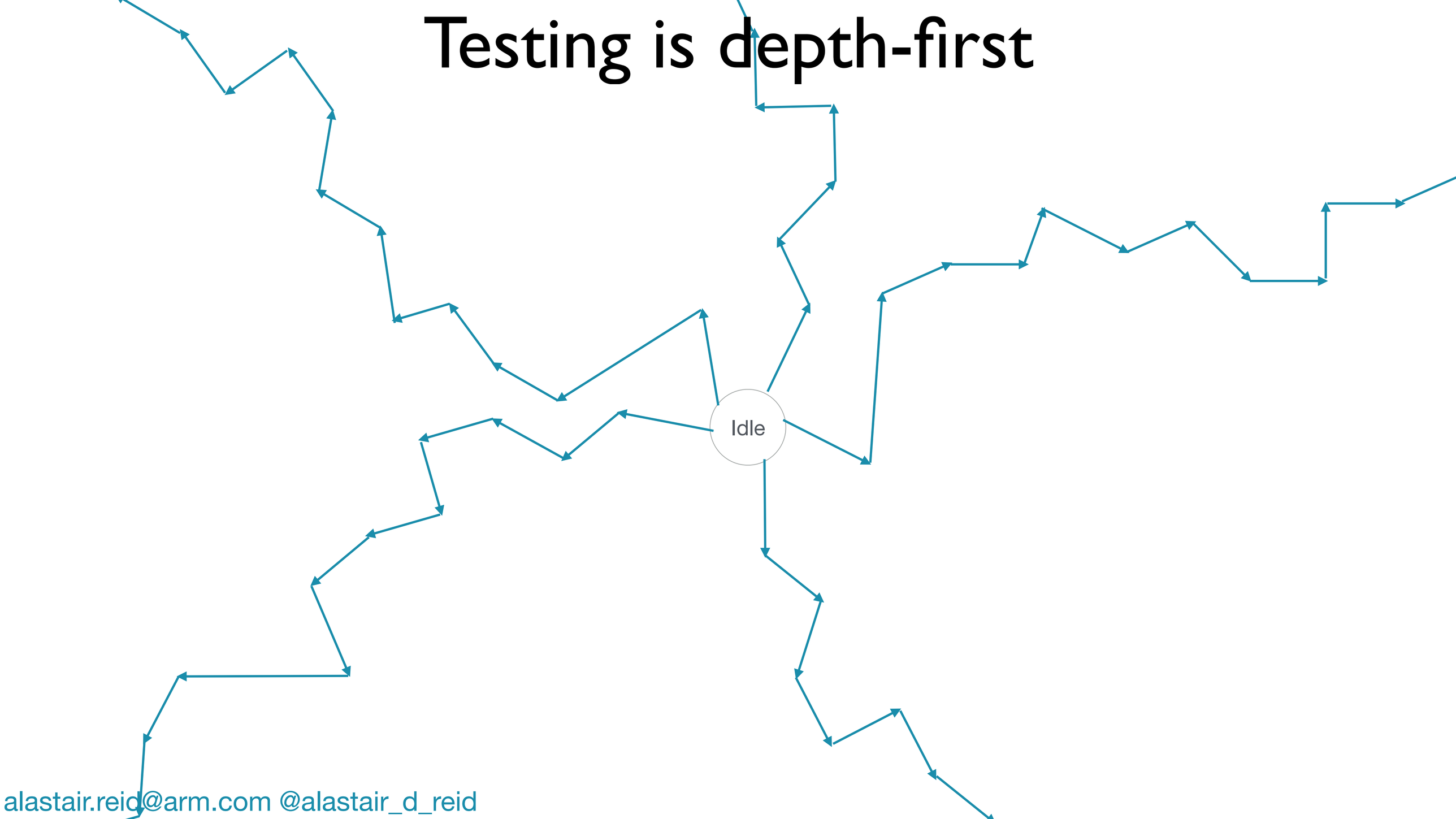


Testing is depth-first

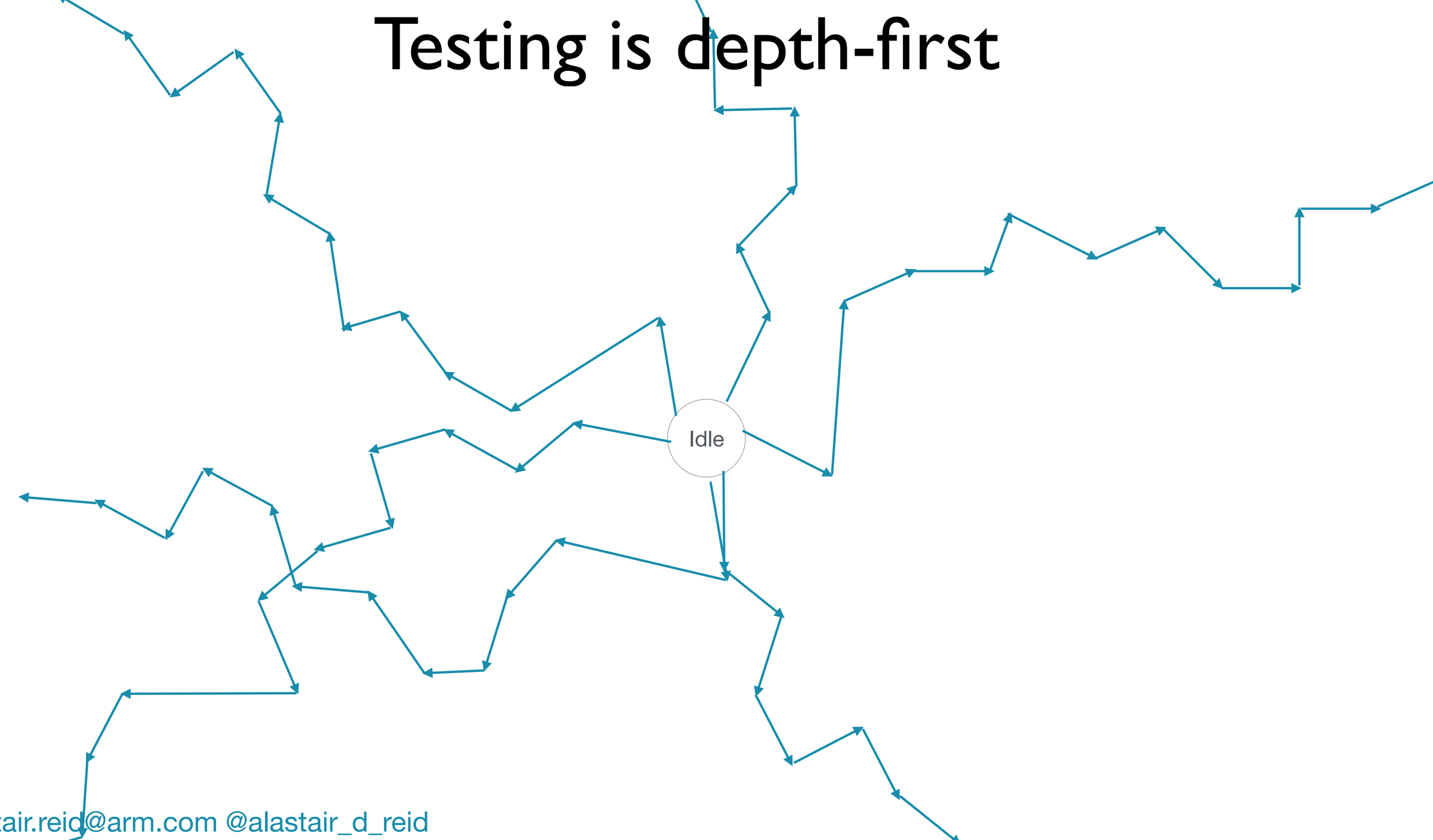


alastair.reid@arm.com @alastair_d_reid

Testing is depth-first

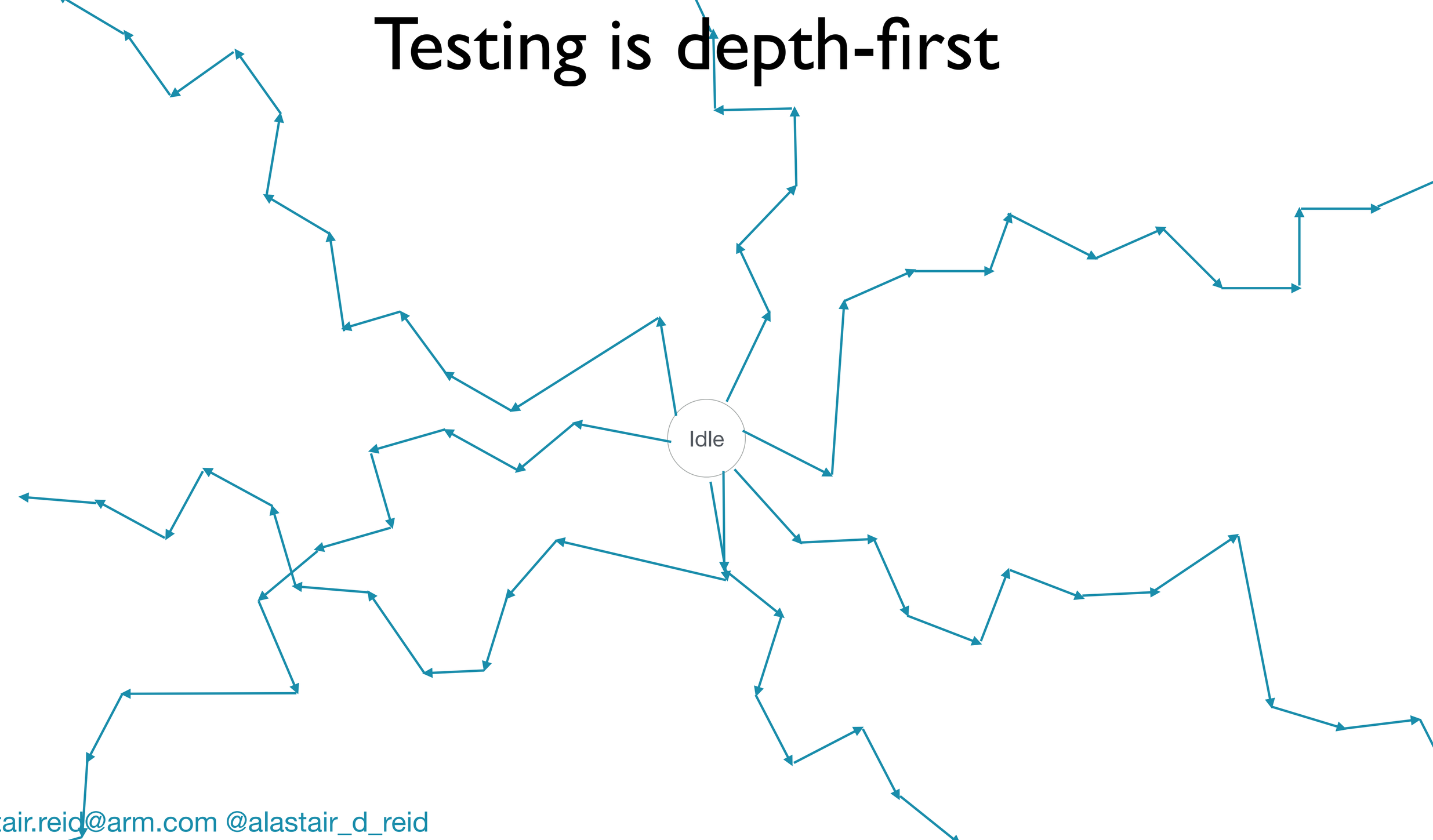


Testing is depth-first



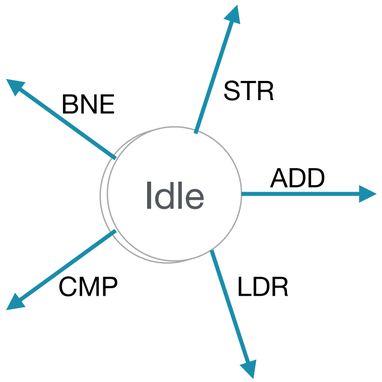
alastair.reid@arm.com @alastair_d_reid

Testing is depth-first

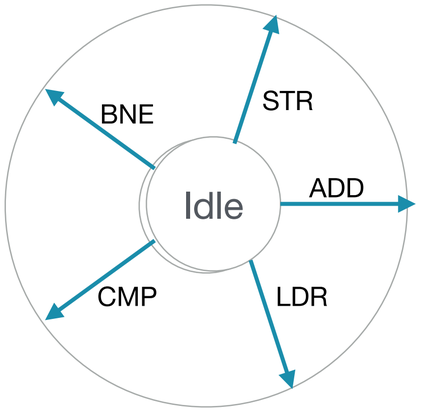


alastair.reid@arm.com @alastair_d_reid

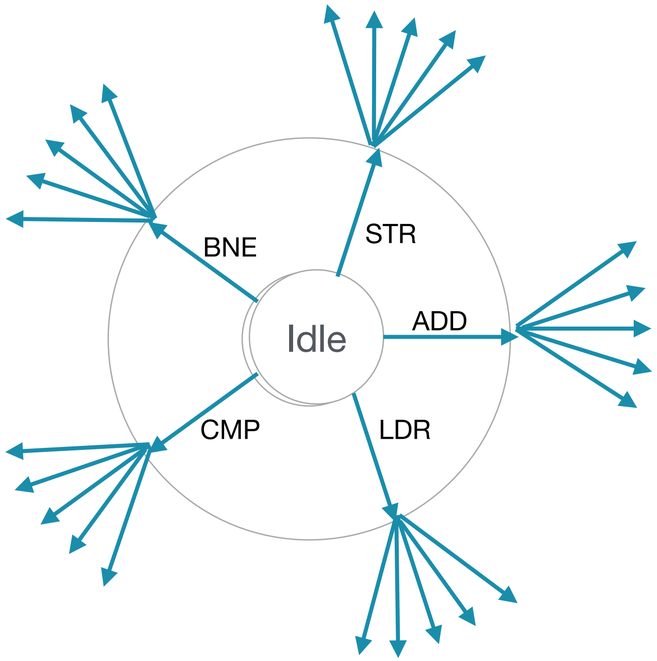
Mixed Mode Verification



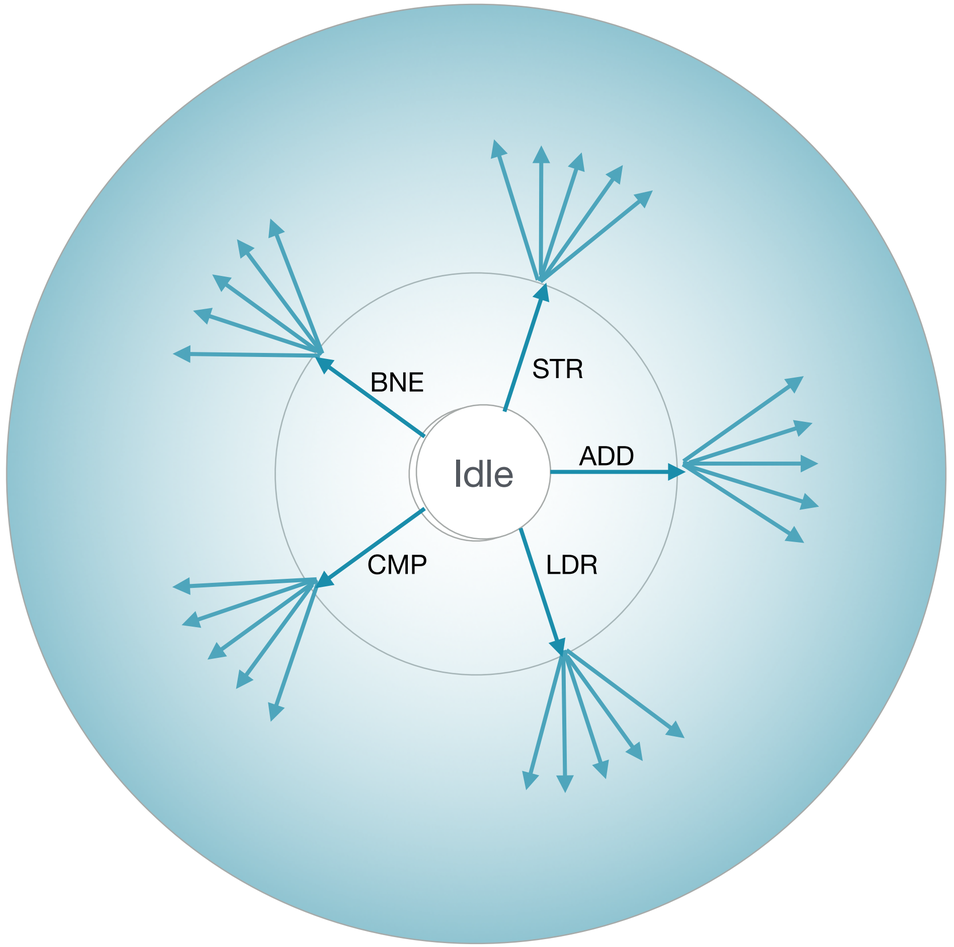
Mixed Mode Verification



Mixed Mode Verification

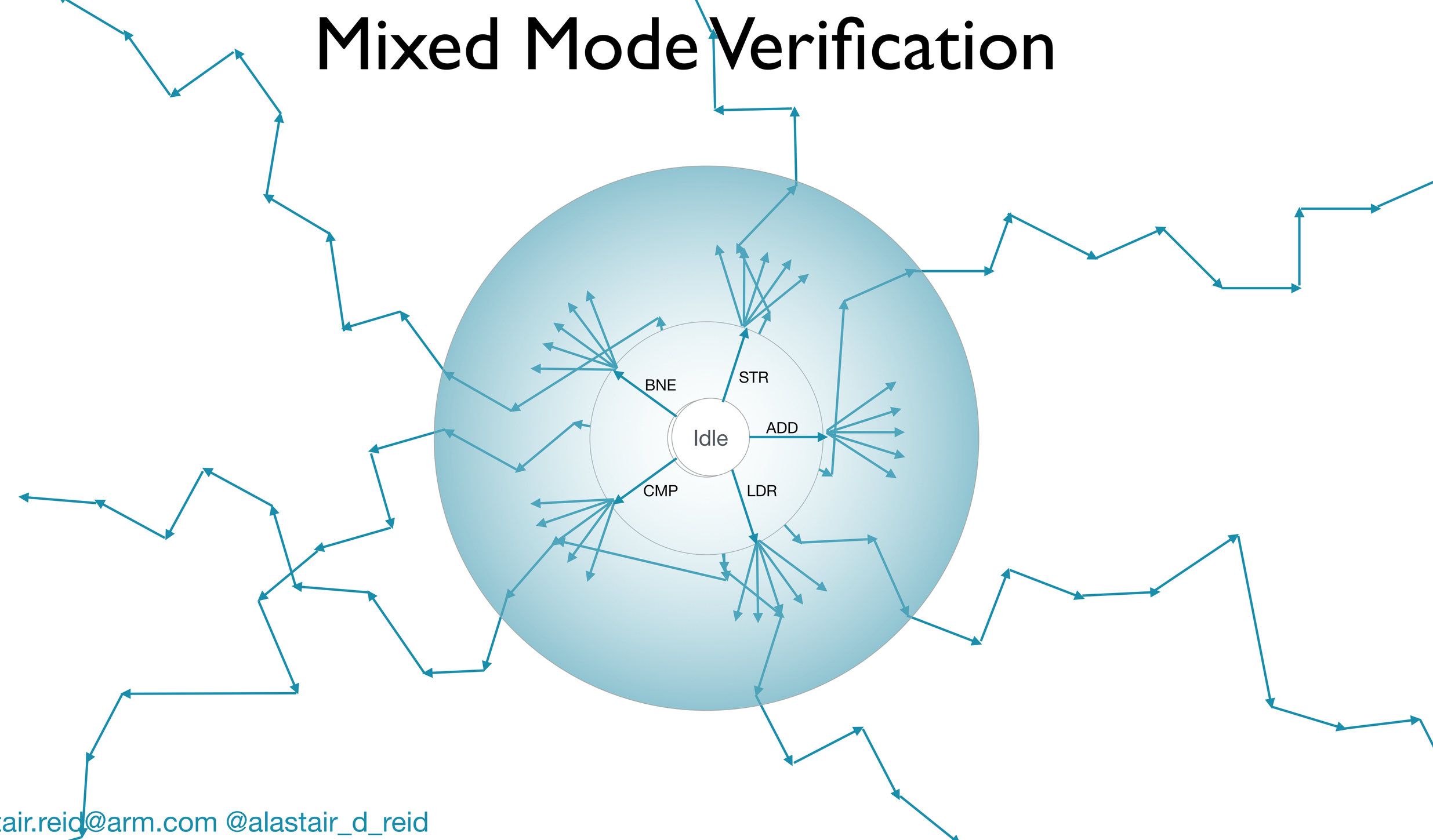


Mixed Mode Verification



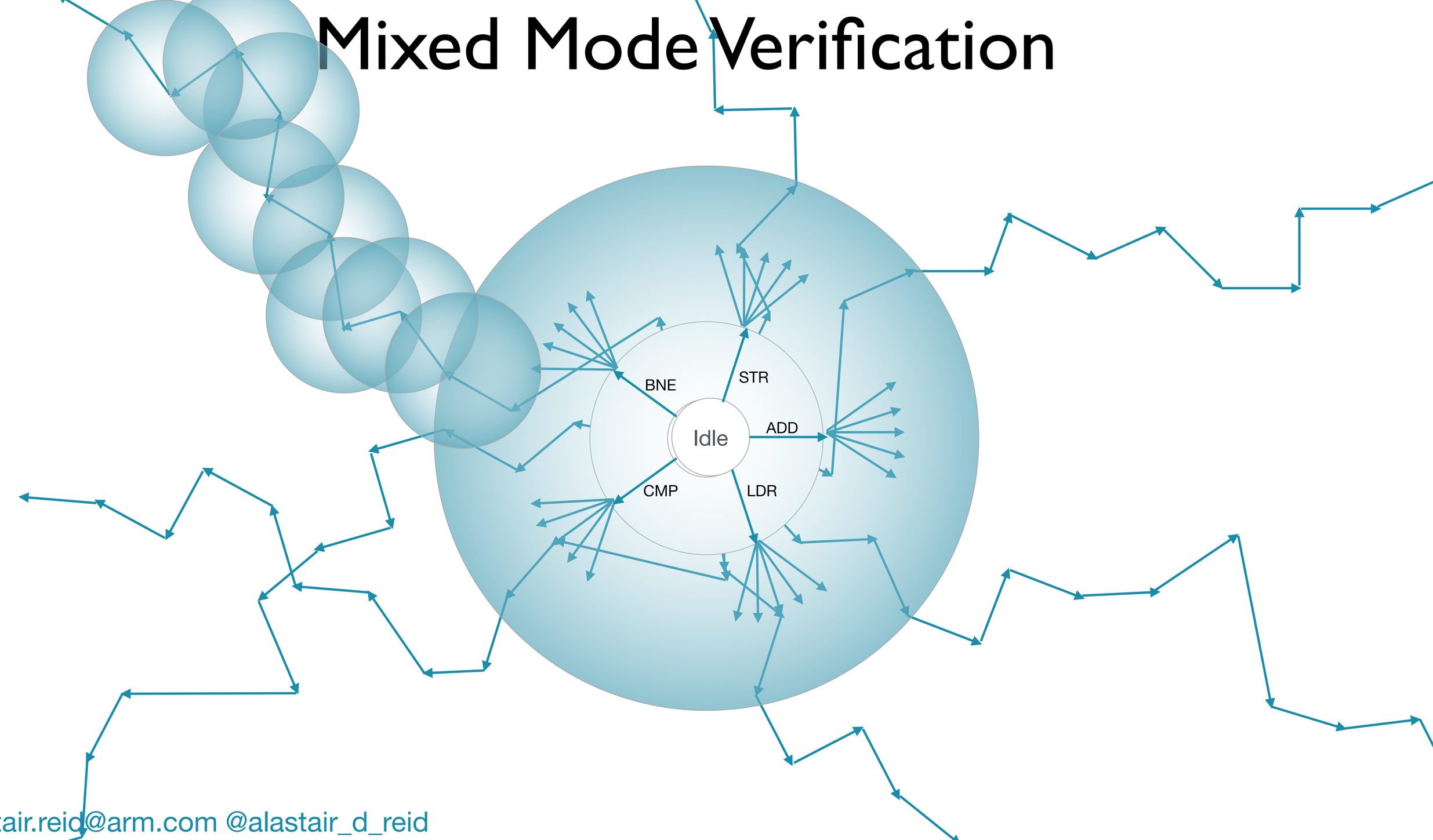
alastair.reid@arm.com @alastair_d_reid

Mixed Mode Verification



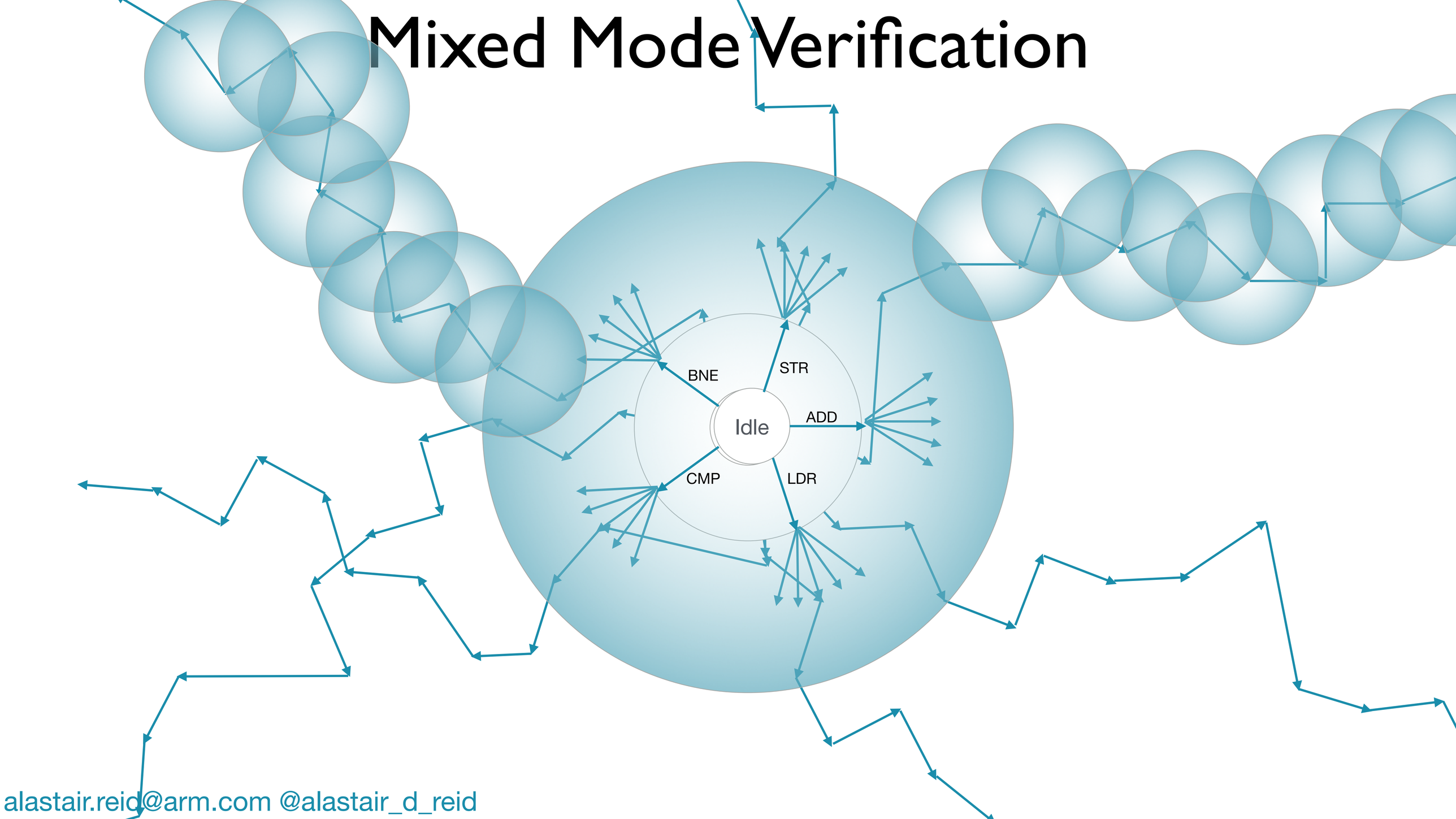
alastair.reid@arm.com @alastair_d_reid

Mixed Mode Verification



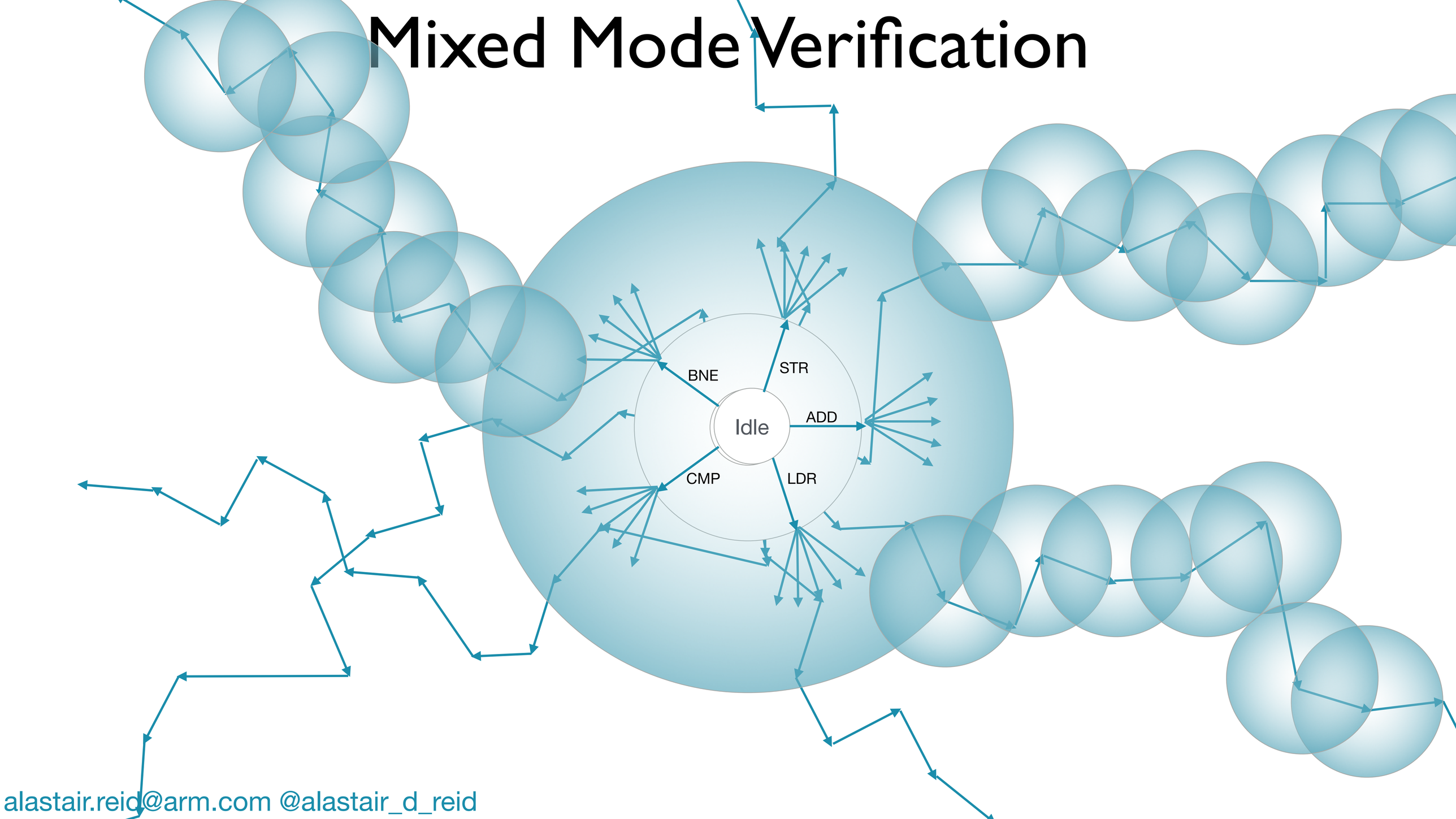
alastair.reid@arm.com @alastair_d_reid

Mixed Mode Verification



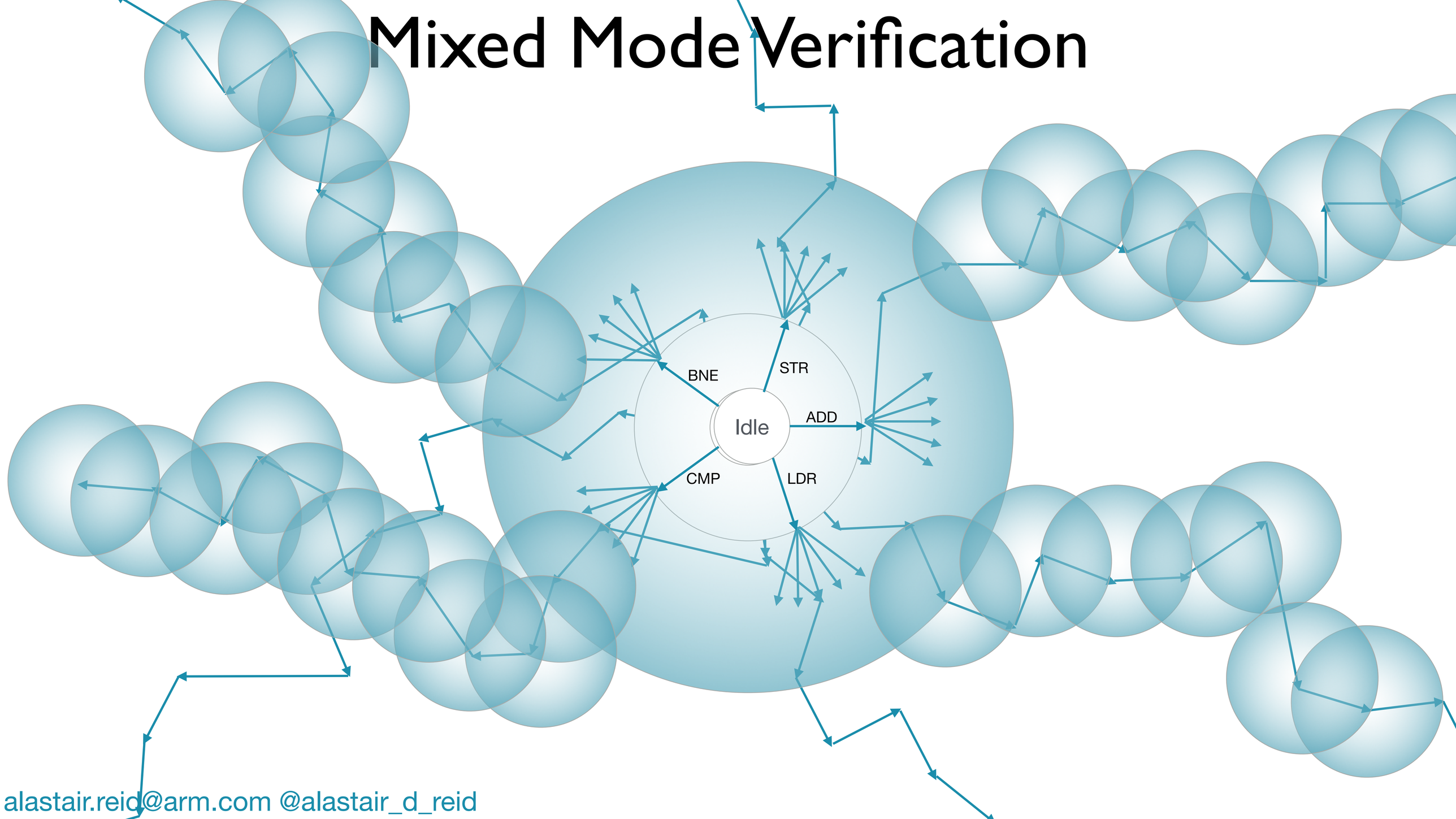
alastair.reid@arm.com @alastair_d_reid

Mixed Mode Verification



alastair.reid@arm.com @alastair_d_reid

Mixed Mode Verification

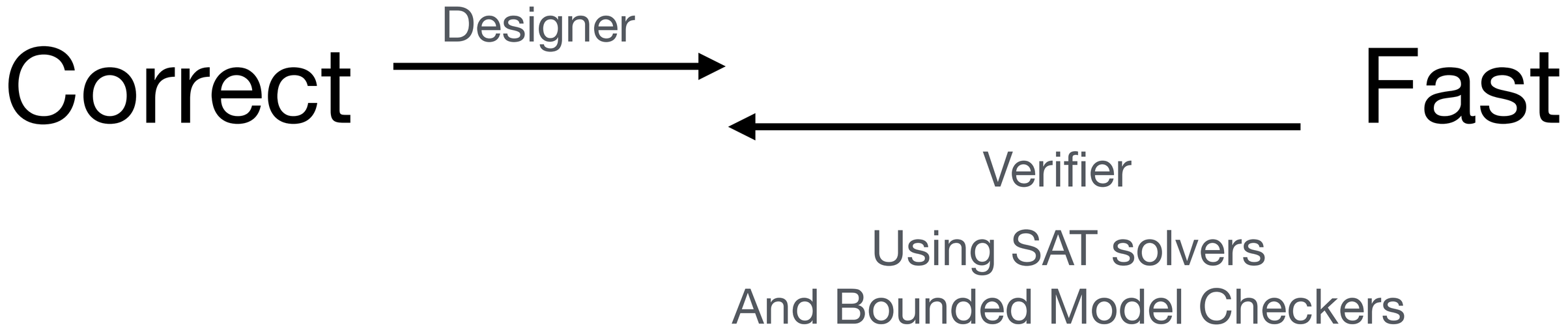


alastair.reid@arm.com @alastair_d_reid

Processors *almost* always work

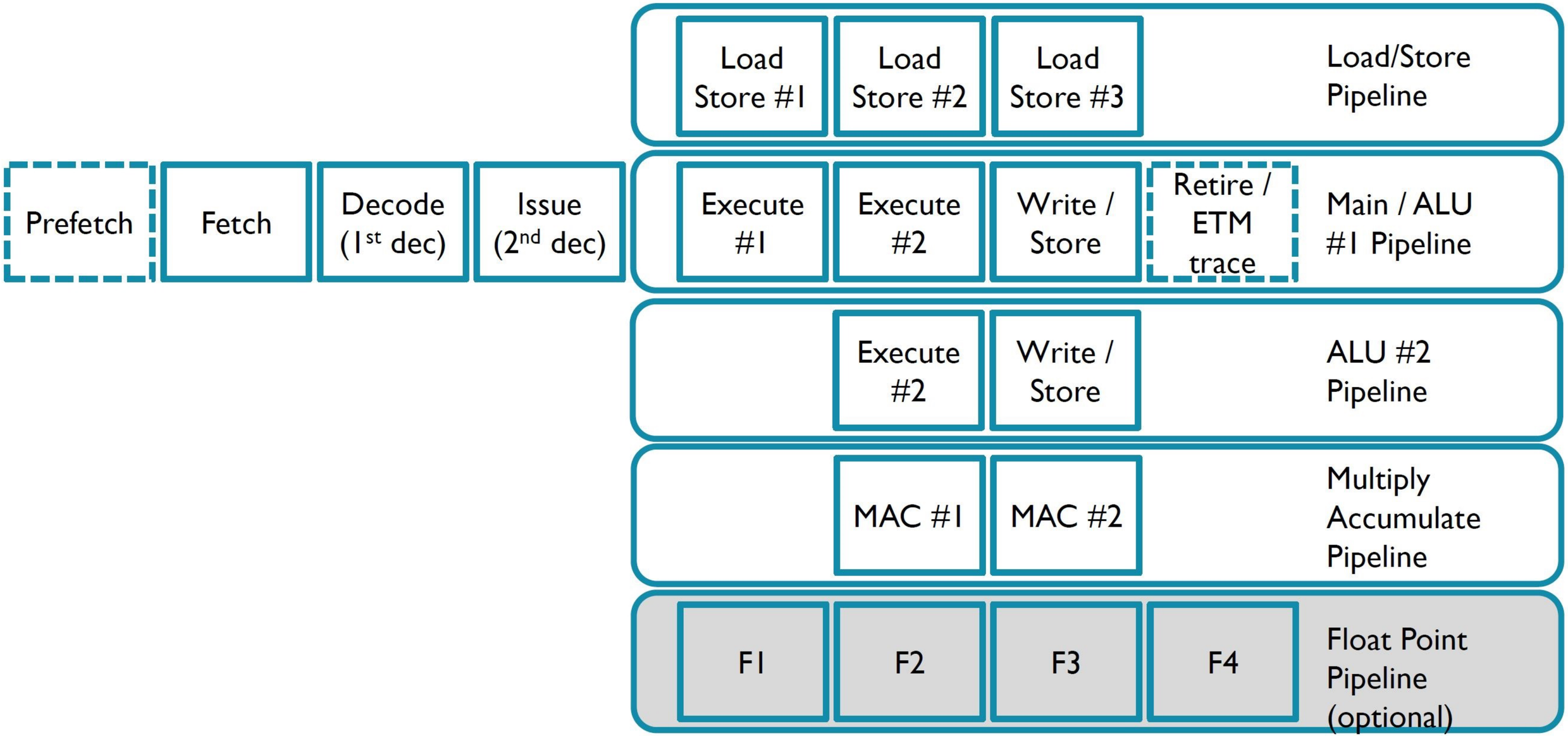
alastair.reid@arm.com

@alastair_d_reid


Correct
Designer
Fast
Verifier
Using SAT solvers
And Bounded Model Checkers

Using SAT solvers
And Bounded Model Checkers

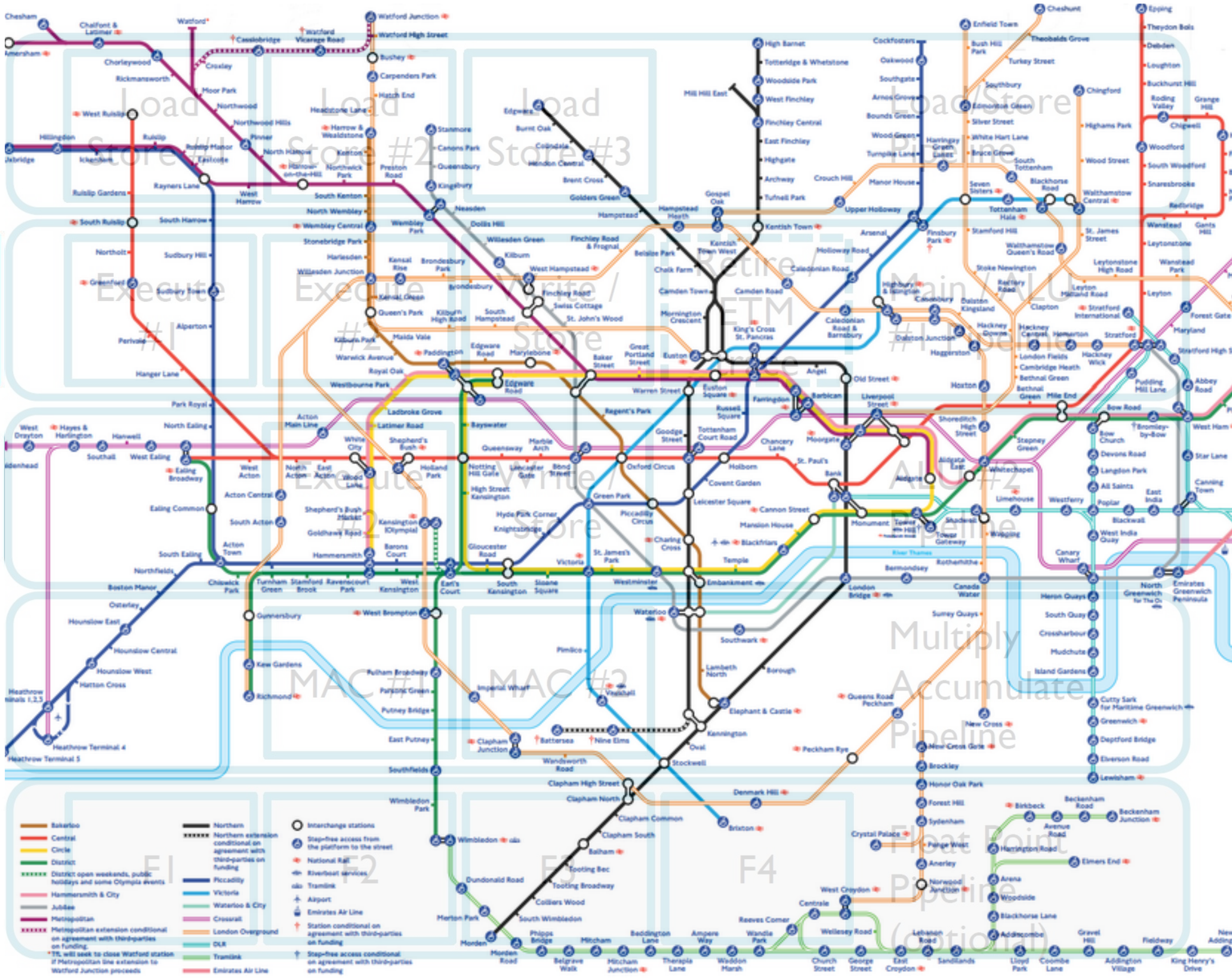
Cortex-M7 Pipeline



https://images.anandtech.com/doci/8542/pipeline2.JPG

Cortex-M7 Pipeline

Prefetch	Fetch	Decode (1st dec)	Issue (2nd dec)



https://images.anandtech.com/doci/8542/pipeline2.JPG