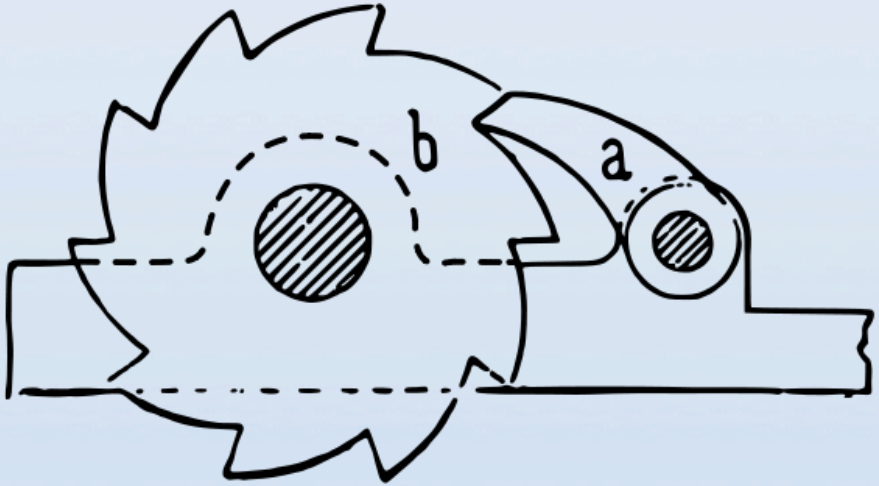


Ratchet

Intermittent Computation without Hardware Support or Programmer Intervention

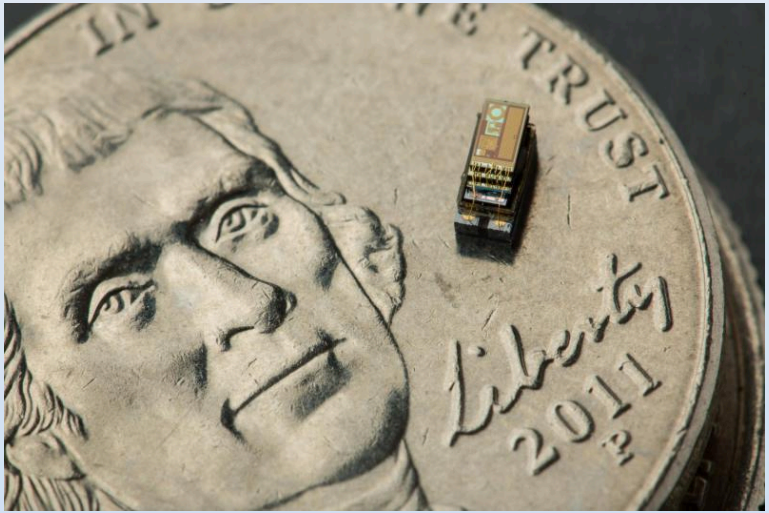
University of Michigan

Joel Van Der Woude
Sandia

Matthew Hicks
MIT/LL


Medtronic
Advisa DR MRI™ SureScan®


4.06MI
35:19.7
135

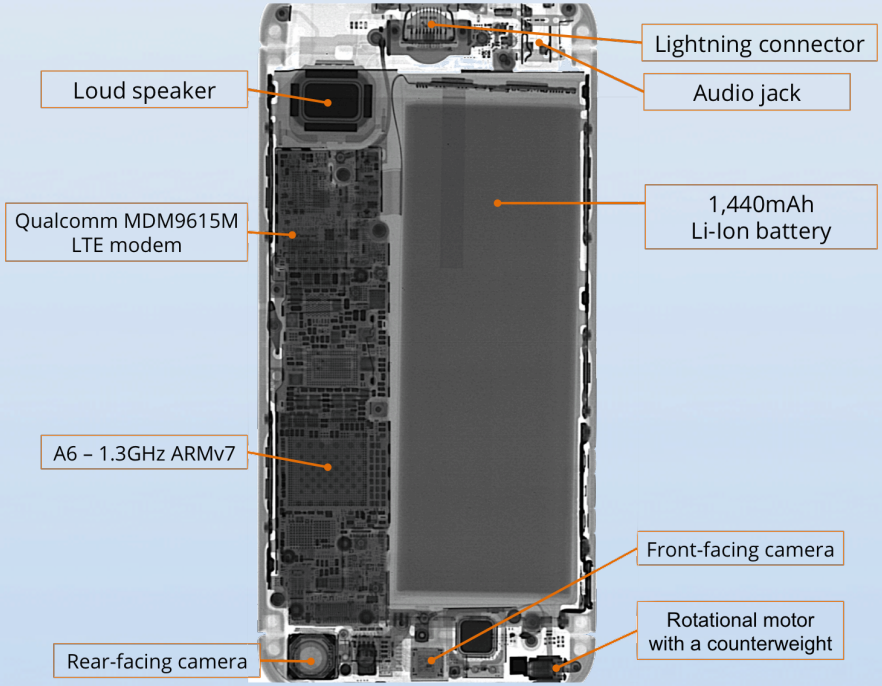

IN GOD WE TRUST
Liberty
2011


9:41 AM


1:31 THRESH
SUSPEND
Medtronic
MiniMed
ACT



Batteries are bulky and inconvenient



Energy harvesting



Hardware monitoring vs. programmer insight



or

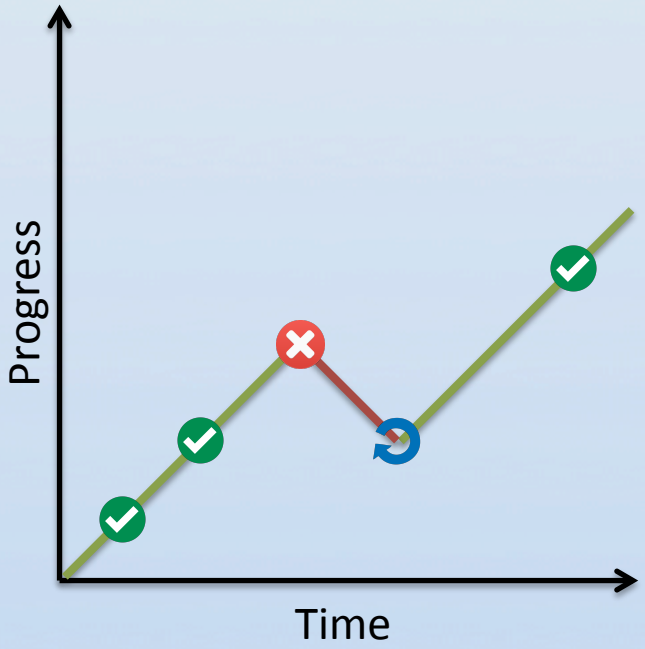


Goals of Ratchet

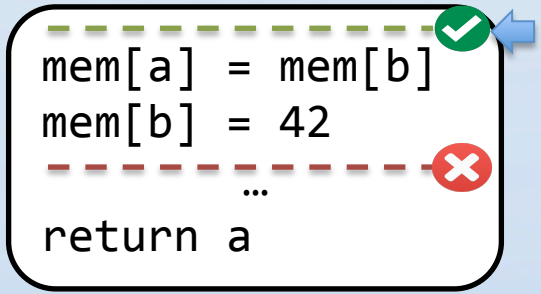
- Burden the compiler not the programmer
- Without hardware to measure voltage
- Ensure forward progress under frequent power failures (~100ms)

Checkpointing

- Save volatile state to non-volatile memory
- After a power fail
  - restart
  - restore
  - continue



Re-execution causes incorrectness



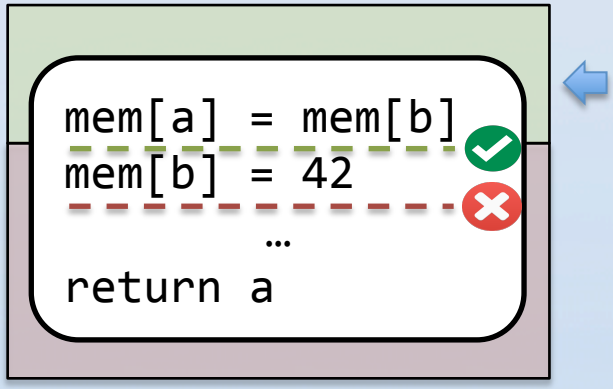
	Initial	Correct	At Fail	Error
mem[a]	x	y	y	42
mem[b]	y	42	42	42

Write after Read (WAR)

What is idempotence?

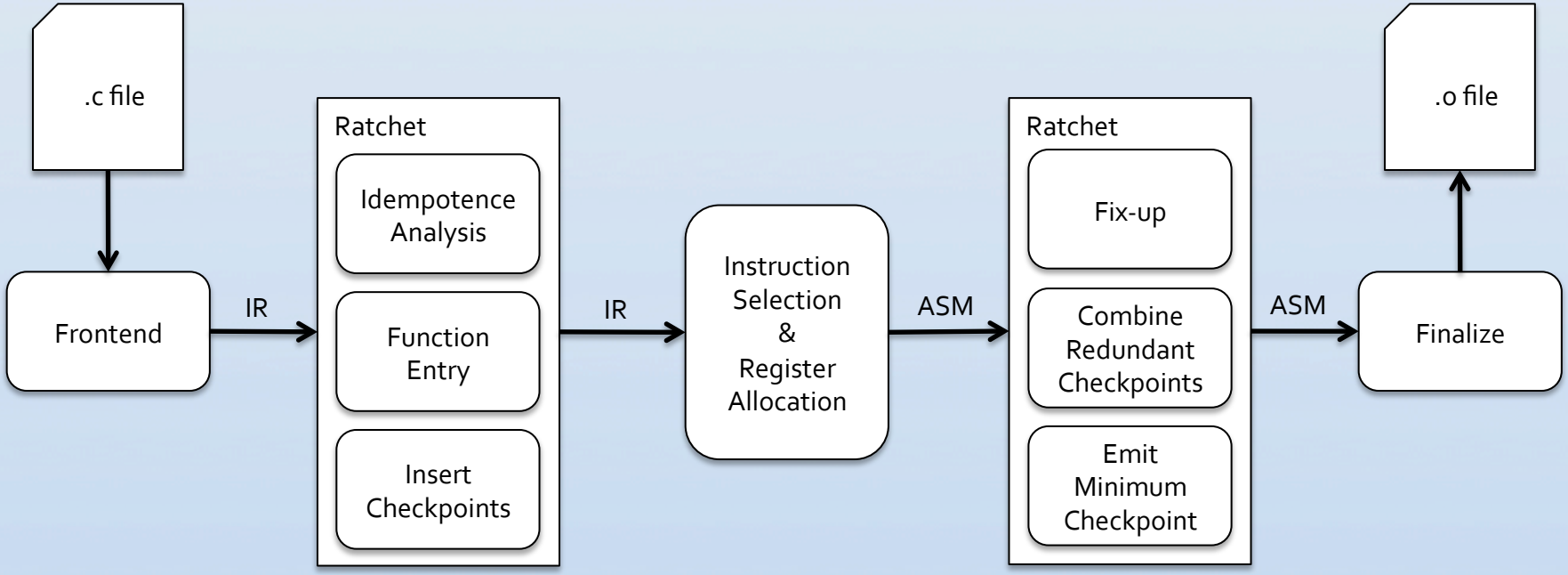
- Re-execution generates semantically correct result
- A section of code that does not overwrite/ update inputs

Idempotence enabling re-execution

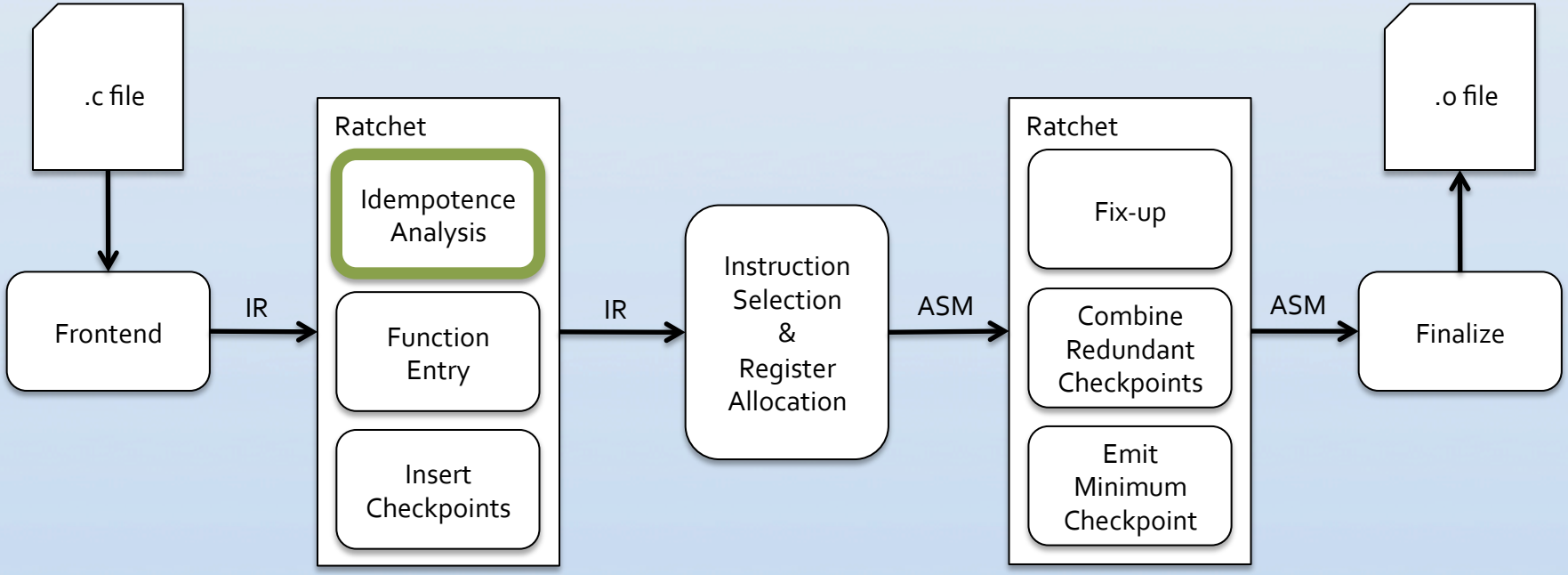


	Initial	Correct	At Fail	Error
a	x	y	y	y
b	y	42	42	42

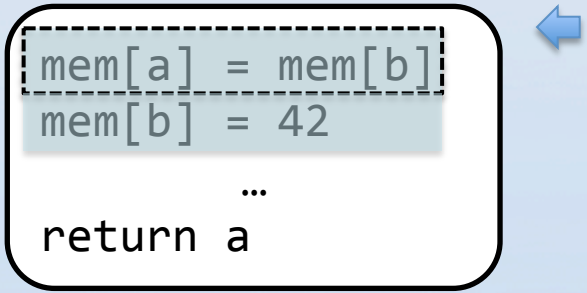
Ratchet system overview



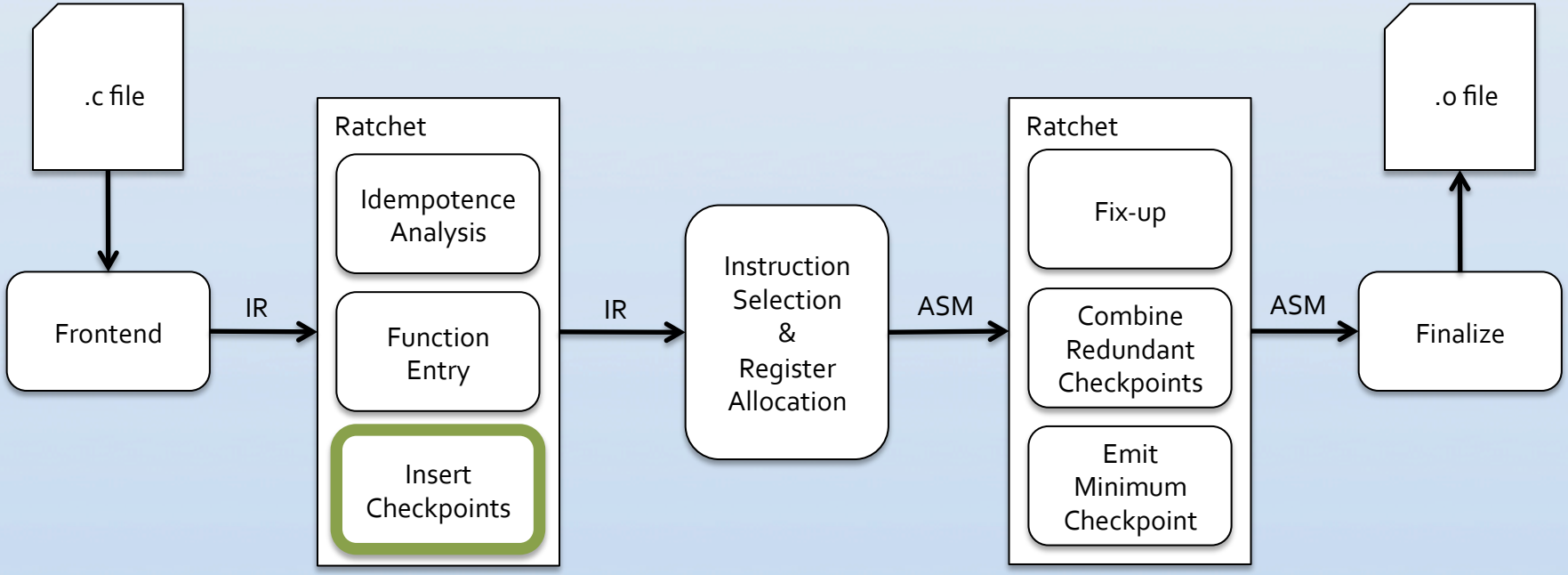
Idempotence analysis



Identifying WARs

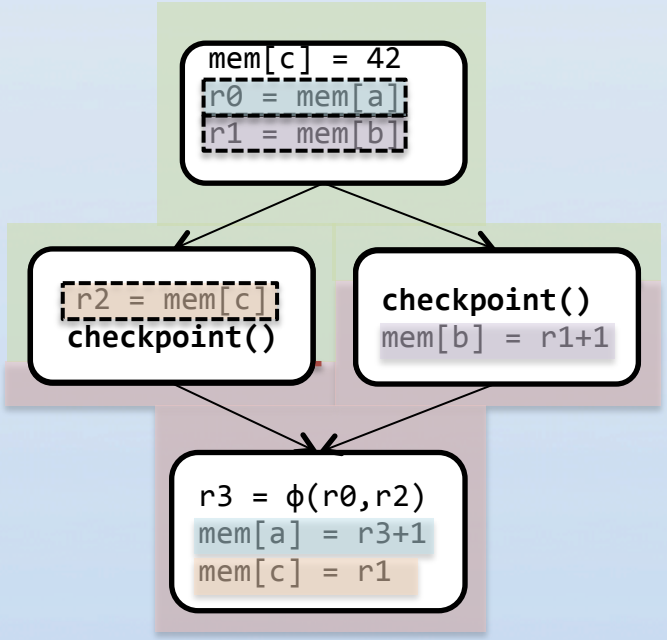


Insert checkpoints

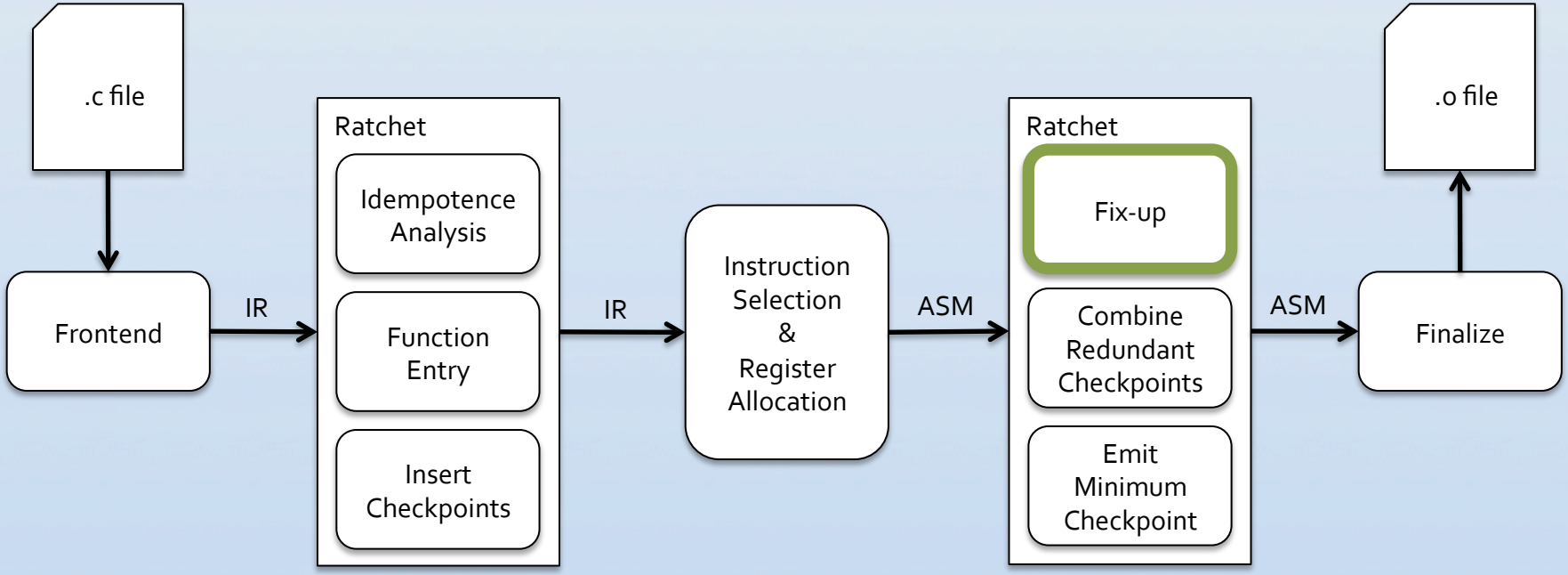


Seperating WARs with checkpoints

- Weighted with
  - Loop depth
  - # of WAR cut

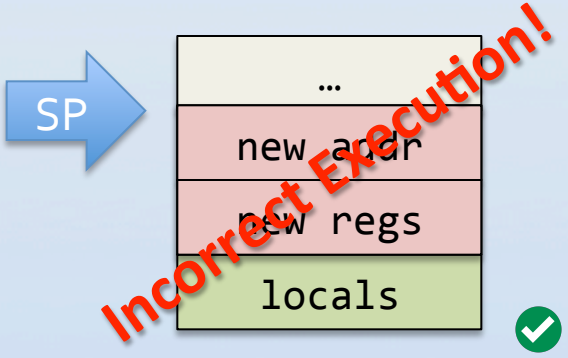


Fix-up

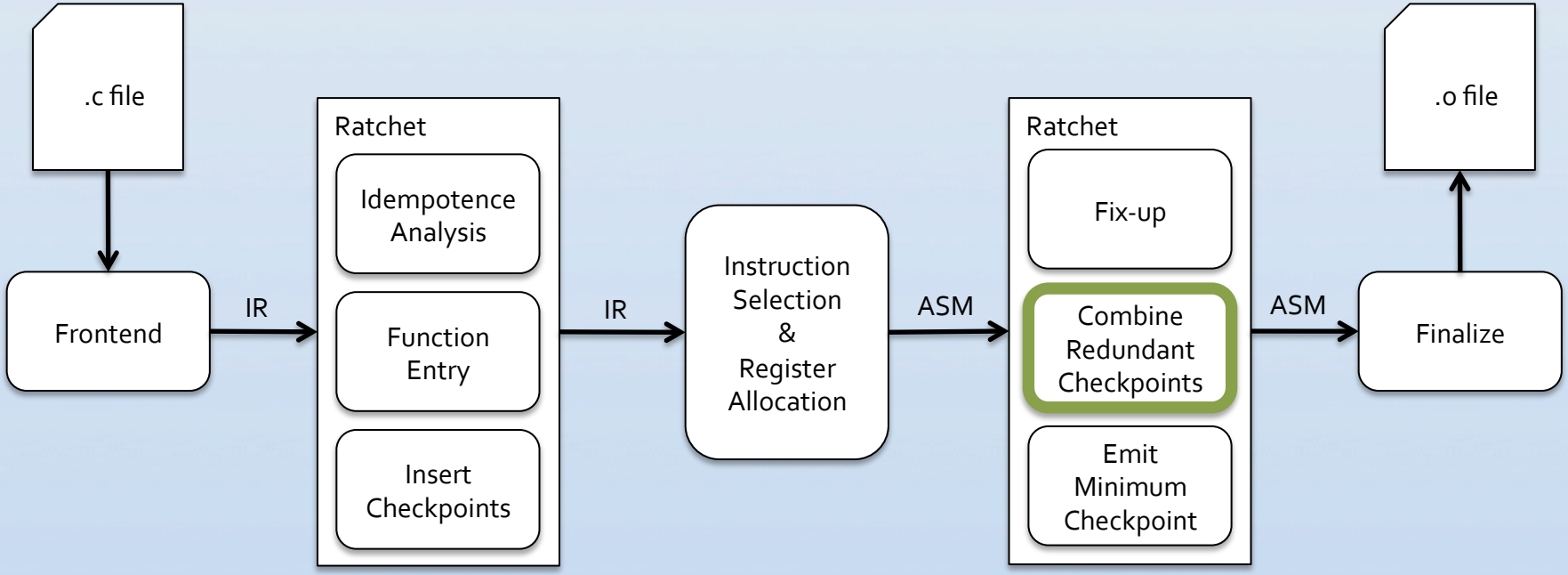

.c file
Frontend
IR
Ratchet
Idempotence Analysis
Function Entry
Insert Checkpoints
IR
Instruction Selection & Register Allocation
ASM
Ratchet
Fix-up
Combine Redundant Checkpoints
Emit Minimum Checkpoint
ASM
Finalize
.o file

The backend may create WARs

- Stack Teardown
- Register Spilling

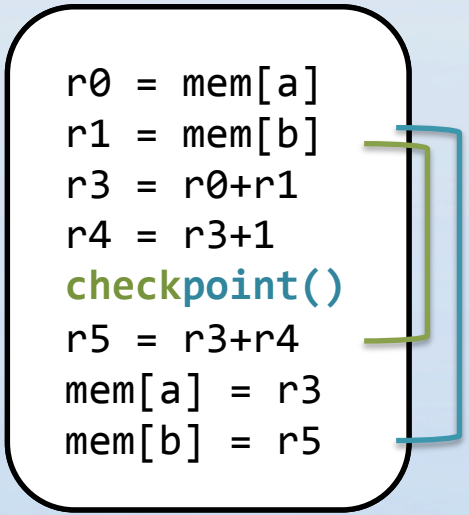


Redundant checkpoints

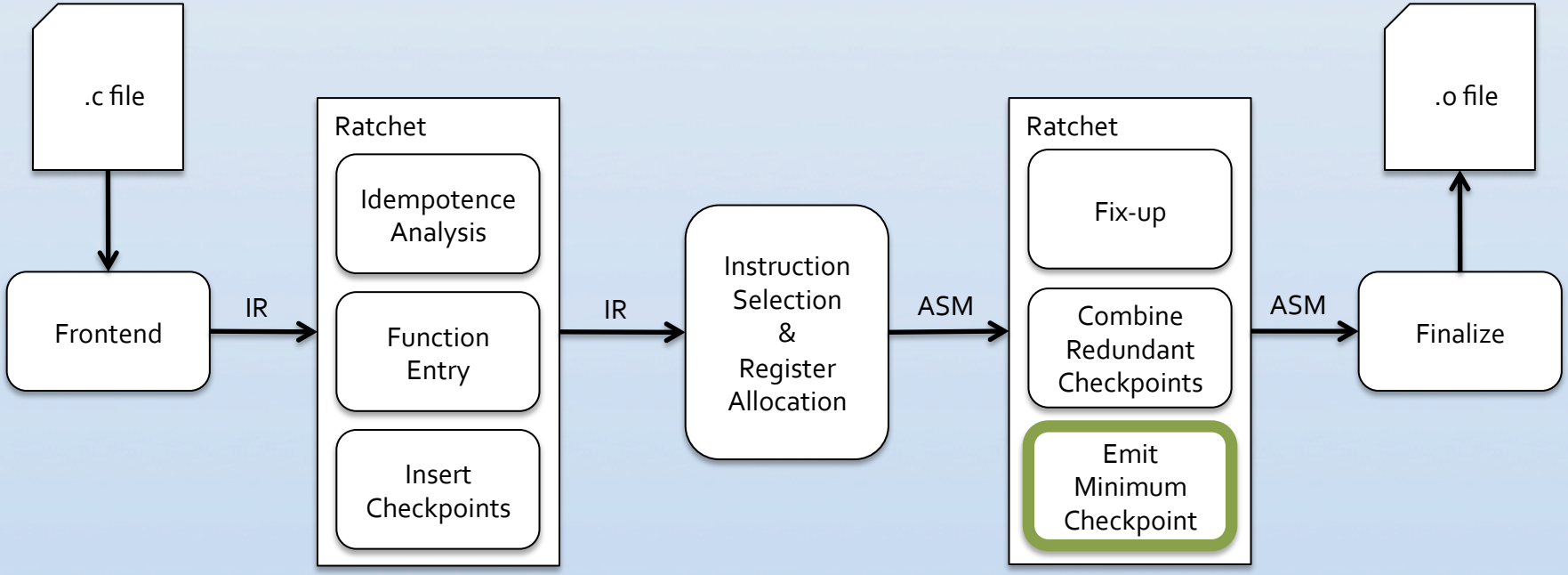


Relocating checkpoints to combine

- Redundant checkpoints caused by…
  - Optimizations
  - Scheduling Decisions
  - Register Spills



Minimum checkpoint

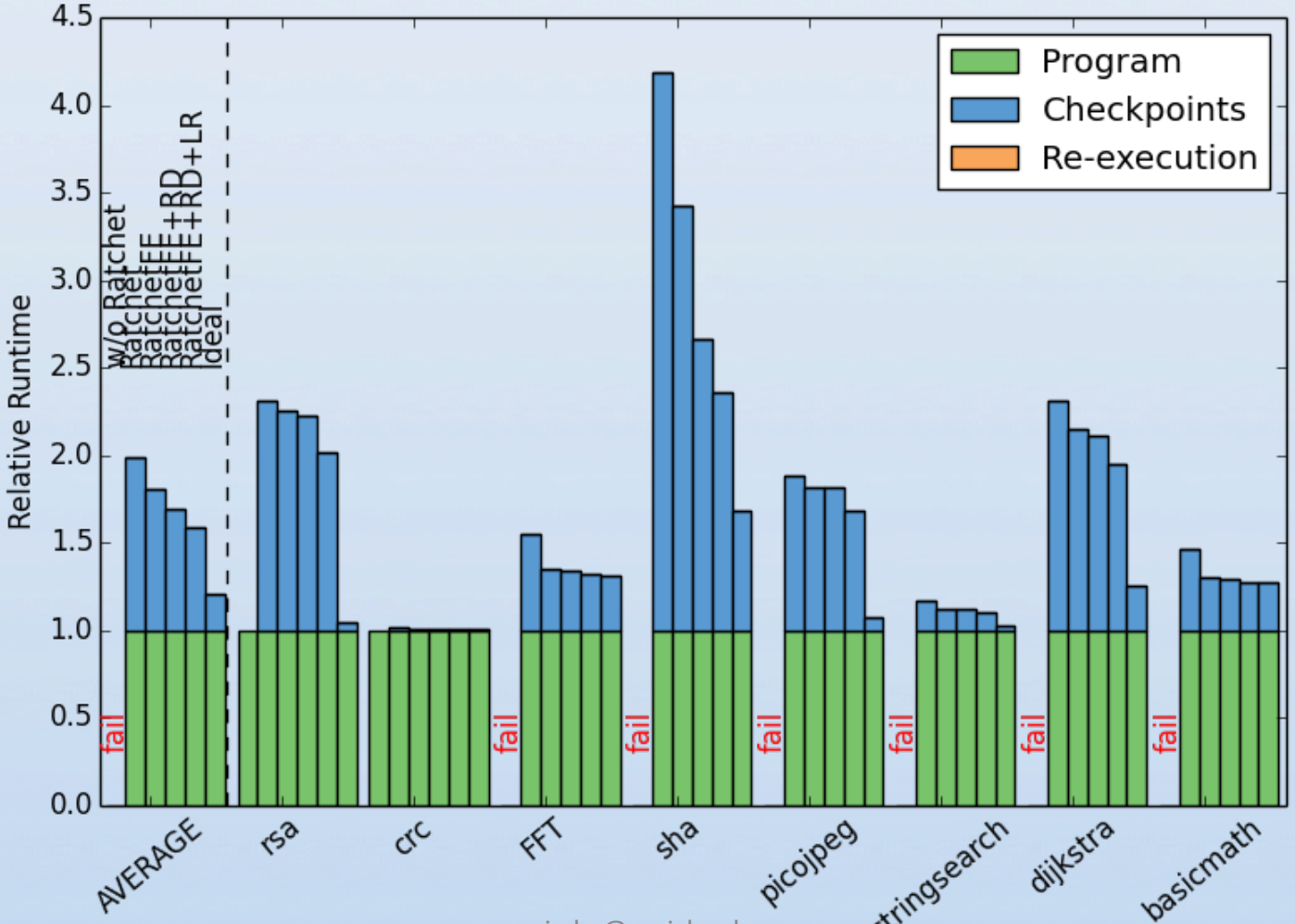


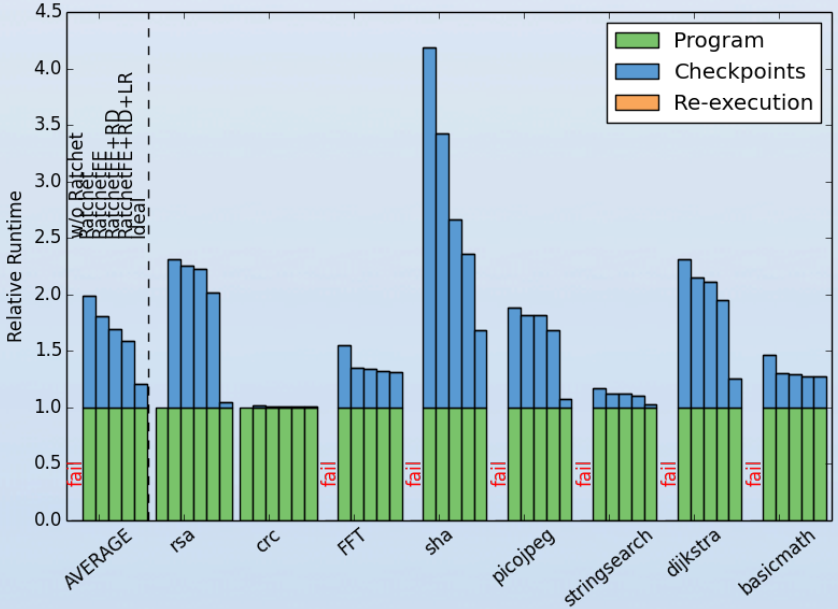
Optimizing Ratchet

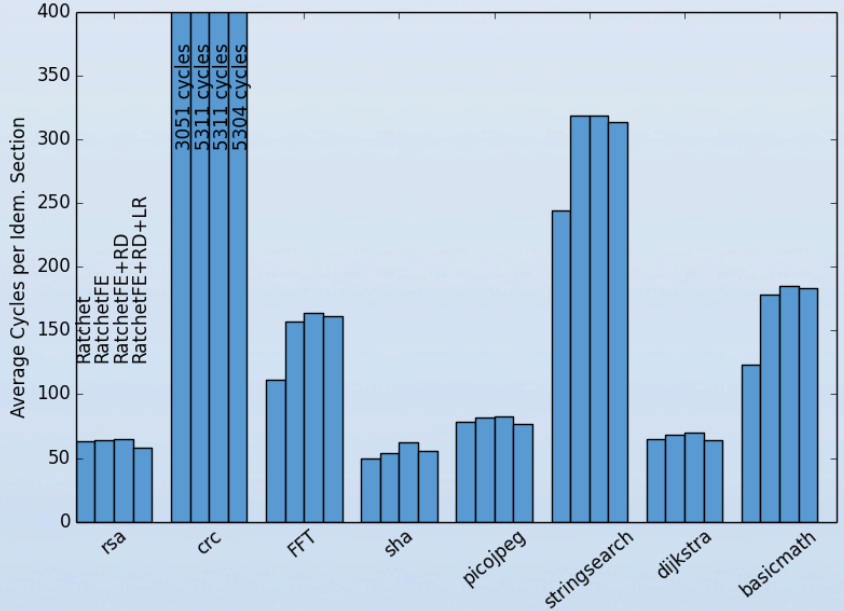
• Ratchet	Unoptimized
• RatchetFE	Function Entry
• RatchetFE+RD	+Remove Duplicates
• RatchetFE+RD+LR	+Live Registers
• Ideal	Single Checkpoint

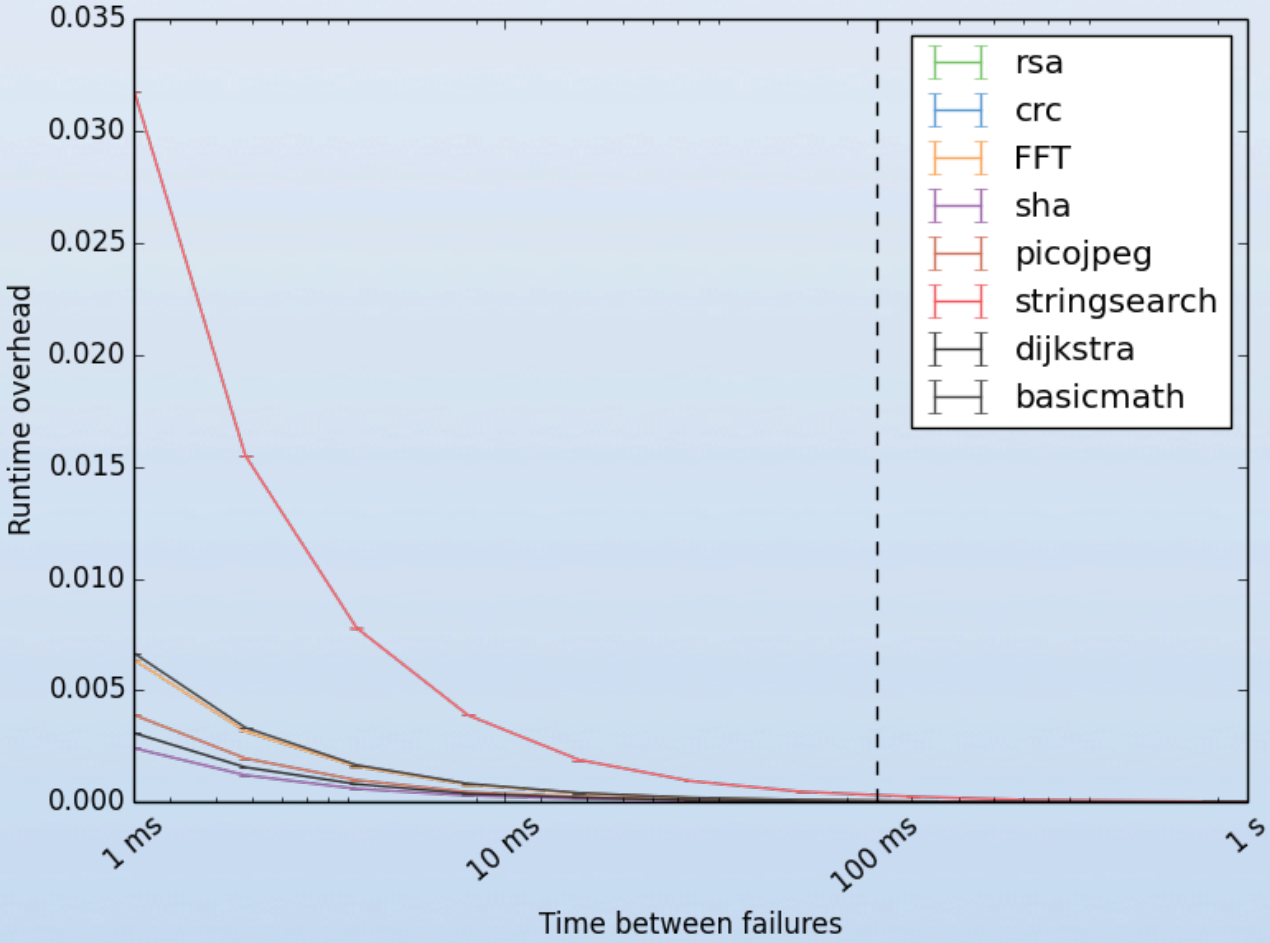
How we evaluated Ratchet

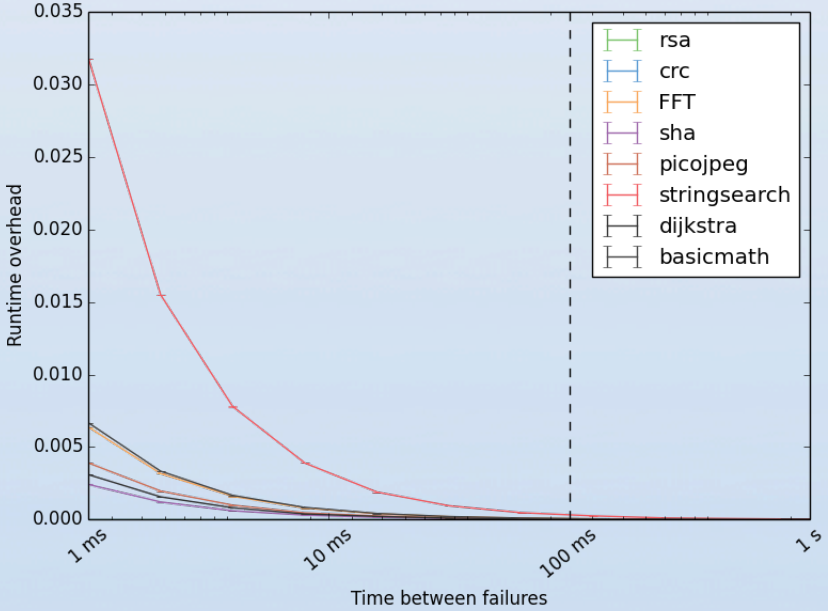
- Simulator
  - Cycle accurate
  - Random power failures
  - Dynamically track idempotence
  - https://github.com/impedimentToProgress/thumbulator
- Benchmarks
  - MIBench
  - newlib
  - https://github.com/impedimentToProgress/MiBench2

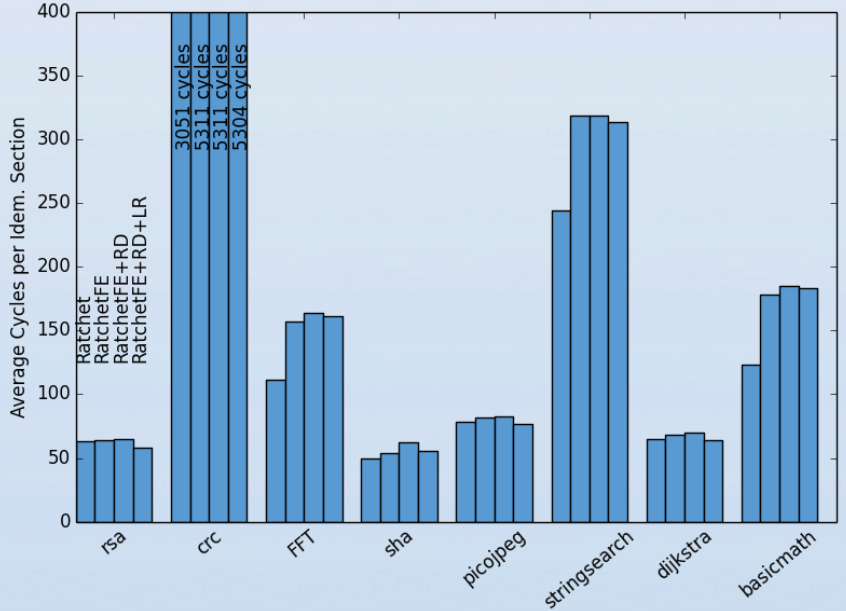

Program
Checkpoints
Re-execution
Relative Runtime
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
w/o Ratchet
Ratchet
Ratchet+FE
Ratchet+FE+RD
Ratchet+FE+RD+LR
Ideal
fail
AVERAGE
rsa
crc
FFT
sha
picojpeg
stringsearch
dijkstra
basicmath


Relative Runtime
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Program
Checkpoints
Re-execution
w/o Ratchet
Ratchet
RatchetFE
RatchetFE+RD
RatchetFE+RD+LR
Ideal
fail
AVERAGE
rsa
crc
FFT
sha
picojpeg
stringsearch
dijkstra
basicmath

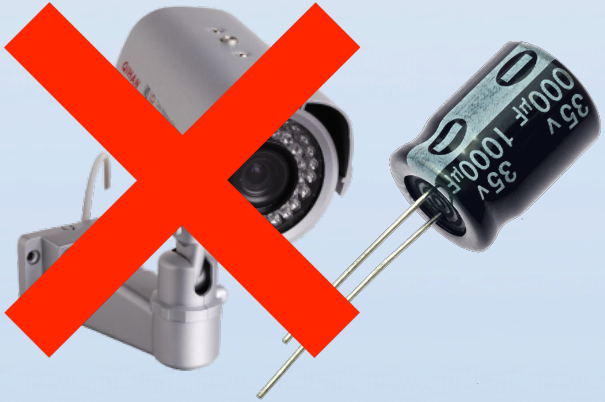

Average Cycles per Idem. Section
400
350
300
250
200
150
100
50
0
Ratchet
RatchetFE
RatchetFE+RD
RatchetFE+RD+LR
3051 cycles
5311 cycles
5311 cycles
5304 cycles
rsa
crc
FFT
sha
picojpeg
stringsearch
dijkstra
basicmath


0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Runtime overhead
1 ms
10 ms
100 ms
1 s
Time between failures
rsa
crc
FFT
sha
picojpeg
stringsearch
dijkstra
basicmath


0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Runtime overhead
1 ms
10 ms
100 ms
1 s
Time between failures
rsa
crc
FFT
sha
picojpeg
stringsearch
dijkstra
basicmath


400
350
300
250
200
150
100
50
0
Average Cycles per Idem. Section
Ratchet
RatchetFE
RatchetFE+RD
RatchetFE+RD+LR
3051 cycles
5311 cycles
5311 cycles
5304 cycles
rsa
crc
FFT
sha
picojpeg
stringsearch
dijkstra
basicmath

Conclusion







Questions?