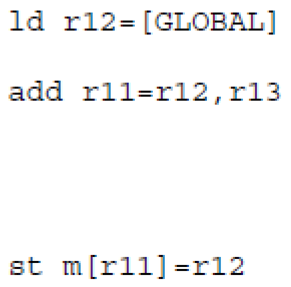
SoftWare Implemented Fault Tolerance (SWIFT)

Ankur Sharma

Dec10, 2012

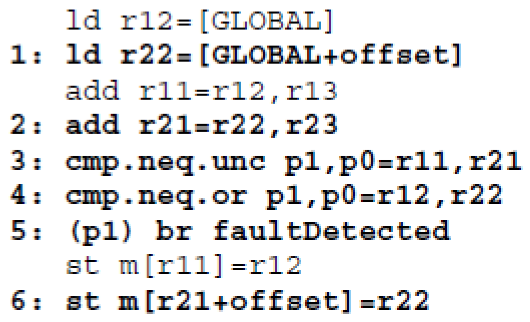
Error Detection by Duplicating Instructions (EDDI) [1]

- Insert a shadow (duplicate) instruction for every master (original) instruction
  - Master and shadow instructions use different registers
- Compare the results before stores



```
ld r12=[GLOBAL]

add r11=r12,r13



st m[r11]=r12
```

(a) Original Code



```
   ld r12=[GLOBAL]
1: ld r22=[GLOBAL+offset]
   add r11=r12,r13
2: add r21=r22,r23
3: cmp.neq.unc p1,p0=r11,r21
4: cmp.neq.or p1,p0=r12,r22
5: (p1) br faultDetected
   st m[r11]=r12
6: st m[r21+offset]=r22
```

(b) EDDI Code

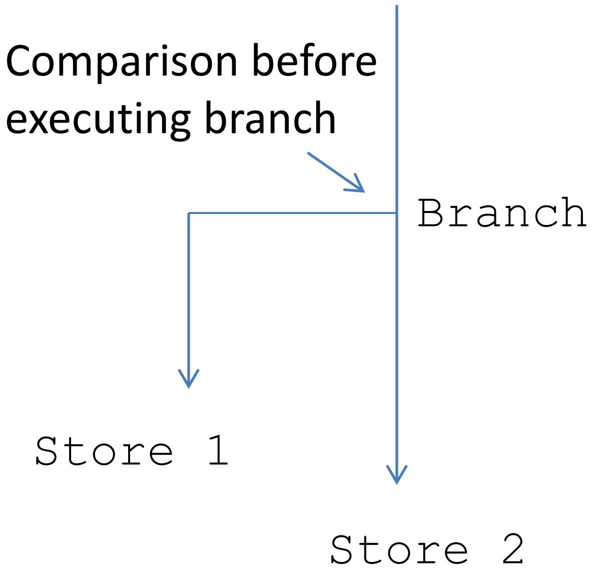
Comparing at Branches

For correctness, need to verify

- What is getting stored
- Where its getting stored
- Is this store supposed to happen

‘What’ and ‘Where’ checks require comparing operands

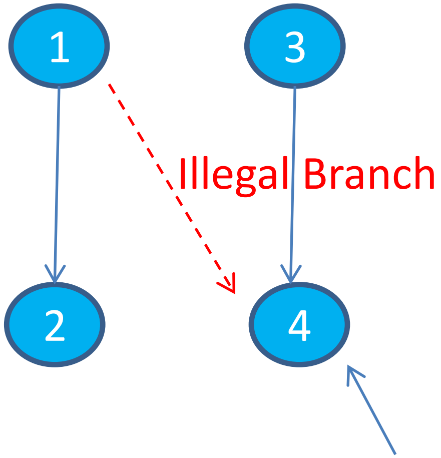
Last check requires comparison at every branch as well



Error During Branch Execution

- Comparison before branch indicates no error so far
- Branch executes …
- ERROR while executing the branch
- Result: Incorrect target

Need for control flow checking

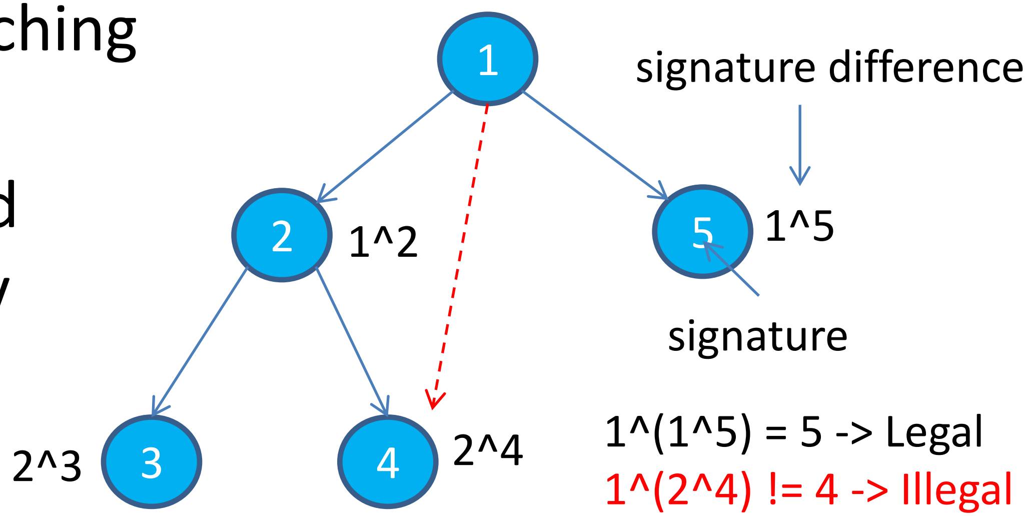


Basic Block: Sequence of instructions without any branching in between, may be at the end

Control Flow Checking by Software Signatures (CFCSS) [2]

Aim: Avoiding illegal branching

Key Idea: Source node and destination node uniquely determine the branch



1^(1^5) = 5 -> Legal

1^(2^4) != 4 -> Illegal

Solution:

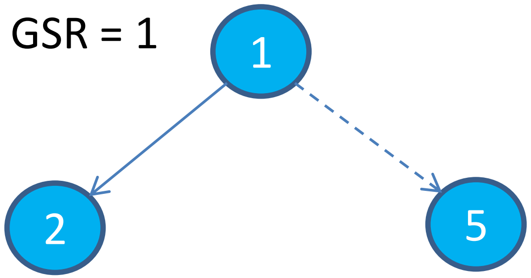
1. Assign a unique signature to every node (basic block) at compile time.
2. Store signature and signature difference with each node.
3. Maintain a general signature register (GSR) containing signature of the current node
4. Add the difference stored with destination node to the current GSR and compare if it matches the signature of the destination node

SWIFT [3] Contribution 1: Enhanced Control Flow Checking

- CFCSS detects legality of the branch
- But doesn't detect the correctness of the branch

RTS = 1^2
GSR = 1

Branch should go to 2
but ends up going to 5



GSR = GSR^RTS
= 2
NO ERROR

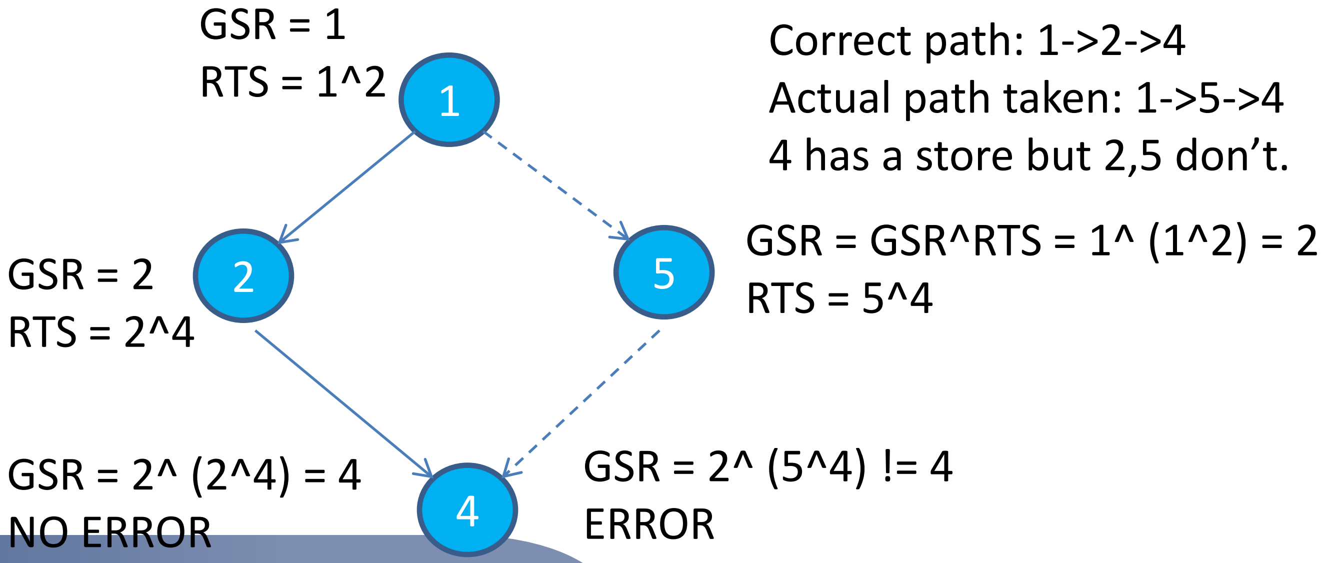
GSR = GSR^RTS
= 1^ (1^2) != 5
ERROR!!

- Source block stores signature difference in RTS
- Target updates GSR by adding RTS to it

SWIFT Contribution 2: Store Control Flow Optimization

Observation: Only stores are problematic

Optimization: Perform control flow checking only for those nodes that has a store. RTS and GSR computation happens in every block.



GSR = 1
RTS = 1^2

Correct path: 1->2->4
Actual path taken: 1->5->4
4 has a store but 2,5 don’t.

GSR = 2
RTS = 2^4

GSR = GSR^RTS = 1^ (1^2) = 2
RTS = 5^4

GSR = 2^ (2^4) = 4
NO ERROR

GSR = 2^ (5^4) != 4
ERROR

SWIFT Contribution 3: Branch Optimization

Observation: Control flow checks are super set of comparisons performed before executing branches. So latter can be eliminated.

```
…
bne $s,$t,L1
…

L1:
…
```

Without Predication

→

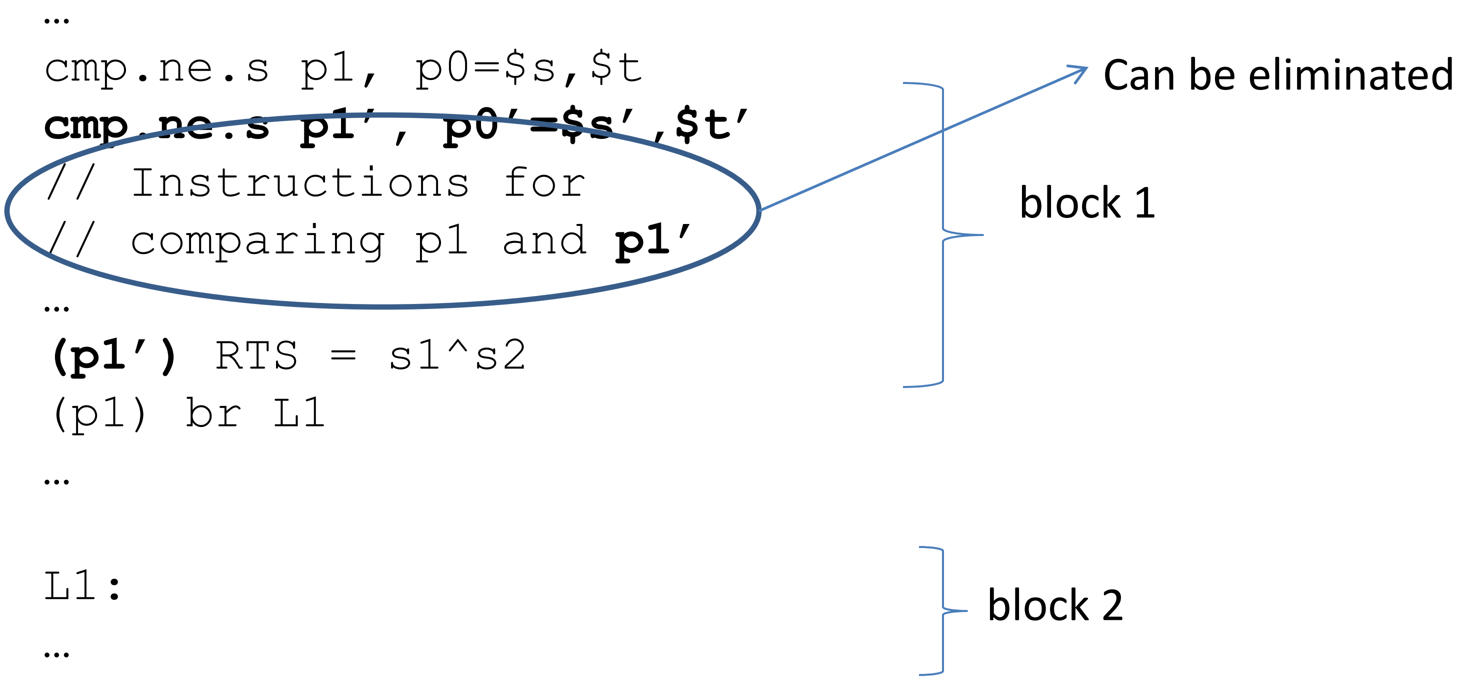
```
…
cmp.ne.s p1, p0=$s,$t
(p1) br L1
…

L1:
…
```

block 1

block 2

With Predication

Branch Optimization (cont’d)



```
…
cmp.ne.s p1, p0=$s,$t
cmp.ne.s p1’, p0’=$s’,$t’
// Instructions for
// comparing p1 and p1’
…
(p1’) RTS = s1^s2
(p1) br L1
…

L1:
…
```

Can be eliminated

block 1

block 2

Code with duplicated instructions

Before jumping to target, RTS is evaluated, if there was an error before branching, then RTS evaluated would be incorrect, and detected later on.

Results from Benchmarks

	No Fault Tolerance	EDDI+CFCSS	SWIFT
Execution Time	1.00	1.61	1.41
Static binary size	1.00	2.83	2.40
Fault Detection	0	100%	100%

- Optimizations helped reduce the static binary size and improve the performance over EDDI+CFCSS
- No loss in reliability

Undetected Errors

- Opcode changed to store instruction
- Multibit error – both master and shadow get similarly corrupted

References:
[1] Reis, George A., et al. "SWIFT: Software implemented fault tolerance." *Proceedings of the international symposium on Code generation and optimization*. IEEE Computer Society, 2005.
[2] Oh, Nahmsuk, Philip P. Shirvani, and Edward J. McCluskey. "Control-flow checking by software signatures." *Reliability, IEEE Transactions on* 51.1 (2002): 111-122.
[3] Oh, Nahmsuk, Philip P. Shirvani, and Edward J. McCluskey. "Error detection by duplicated instructions in super-scalar processors." *Reliability, IEEE Transactions on* 51.1 (2002): 63-75.