Dmitry Borisenkov,
Leonid Kholodov ,
Dmitry Shtukenberg

Towards Better Code Generator Design and Unification for Stack Machine

TON LABS

Why do we care?

- New stack machines keep on appearing given the development of blockchain technology (EVM, TVM).
- The reason for choosing stack machines in blockchain is better code density. It allows reducing costs incurred from contract storage in the blockchain.
- Better verification is another reason to use stack machines.

Comparing existing stack machines

Property \ Architecture	TVM	EVM	WebAssembly	x87	.net
Stack operations	Materializing constants, copy, xchg, push, pop, and derivatives	Copy, push, pop, xchg	Push	Push, pop, index	Push, dup, pop
Non-stack entities	Non GP-registers	Memory, Storage	Locals, globals	Memory	Locals, globals, function arguments
Memory	Modelling	Conventional	Conventional	Conventional	Conventional
Control flow	Continuations	Branches	Branches + Loops	N/A	Branches
Instruction set	CISC w/o direct manipulations with registers	RISC	RISC	CISC	RISC

Sample TVM code

```
int func (int a, int b) {
    return (a + b) * (a - b);
}
```

```
    .globl  func
    .type   func,@function
func:
; %bb.0:
    PUSH2   s1, s0
    SUB
    XCHG s0, s2
    ADD
    MUL
.Lfunc_end0:
```

TON LABS

Stack machine as an abstraction

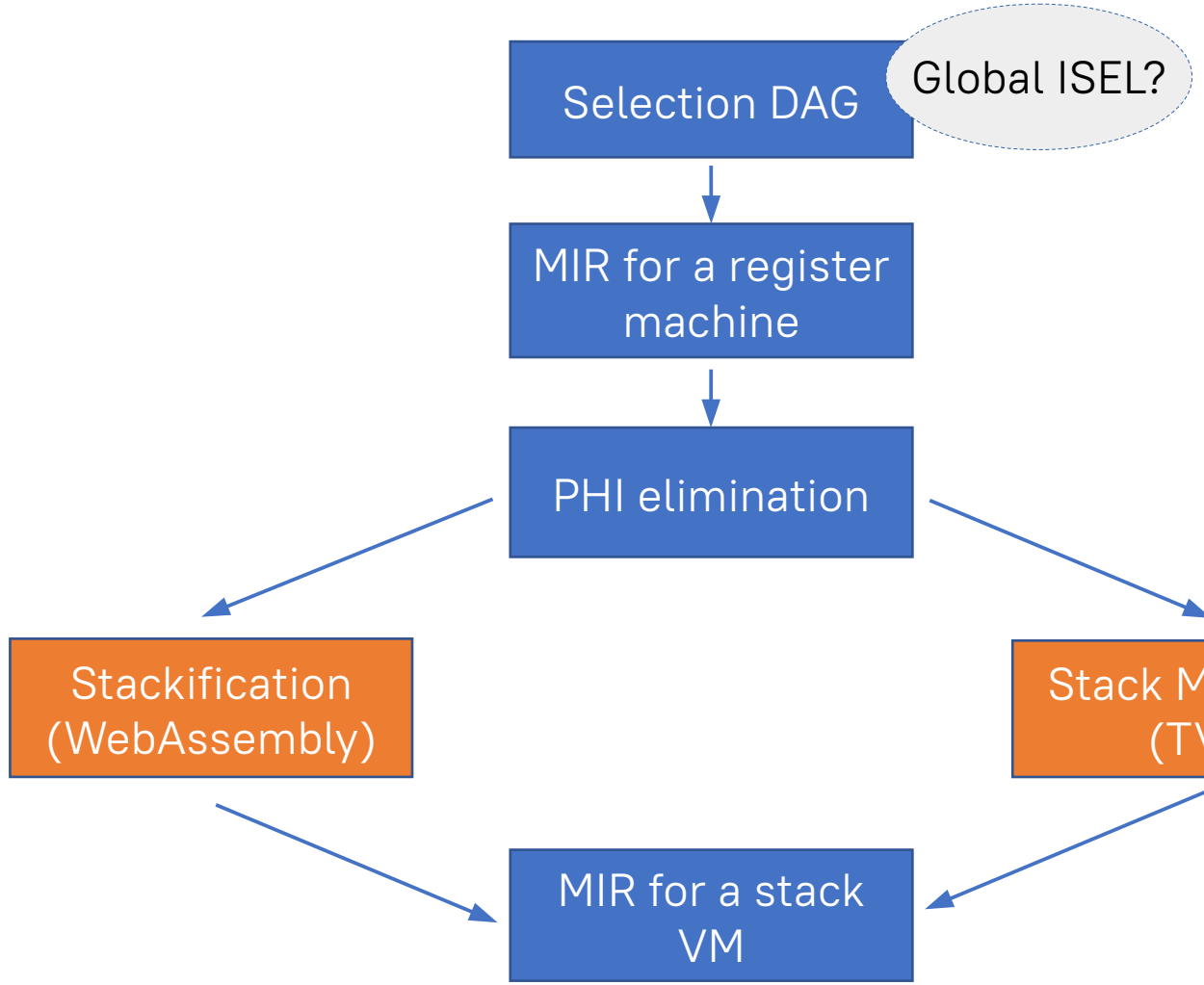
- Each N-th operation consumes N-topmost values from the stack and records the result on top of the stack.
- Stack rearrangements related to computation have costs. The TVM architecture uses copies and exchanges, other architectures - pushes.
- The code generator goal is to minimize stack manipulation costs.

Current approach to translation

```
a = x1 + x2
d = x3 + x4
b = x5 + x6
c = a + b
```

↓

```
a = x1 + x2
b = x5 + x6
c = a + b
d = x3 + x4
```




```
c = a + b;      [b, d, a]
```

↓

```
xchg s1, s2;    [b, d, a]
c = a + b;      [b, a, d]
```

TON LABS

What can be improved?

- In addition to ScheduleDAGFast & Schedule DAGVLIW, SelectionDAG configuration can be modified to provide better schedules for stack machines.
- Stack modeling generalization can be considered.
- Post-RA scheduling (control-flow stackification) and stack-aware reassociation are also options to consider.

Pre-RA scheduler optimizations

- LRN or RLN traversals of DAG for better stack machine scheduling.
- Passes that introduce copies of virtual registers with multiple uses.

Stack modelling potential

- While now we may seem the only users of stack machine modelling, it can be generalized to support various stack manipulation types.
- For both register and stack MIR, stack modelling can provide a helpful analysis pass to request stack configuration for a specific instruction.

Post-RA scheduling optimizations

- Output stack configurations of all predecessors must be equivalent.
- Initial stack configuration of the basic block can specify the required number of manipulations.
- The optimization goal is to minimize the number of stack manipulations in a current basic block and at the end of its predecessors.

Suggestions on stack aware reassociation

- a + b + c + d with **[a, c, b, d]** stack is better to calculate as a + c + b + d
- This type of reassociation cannot be performed before computation of initial stack configurations.

Thank you.

tonlabs.io

TON LABS