Go on RV64G

Benjamin Barenblat
bbaren@google.com

Michael Pratt
mpratt@google.com

https://github.com/riscv/riscv-go

Google

What's Go?

```
package main

import (
        "fmt"
        "log"
        "strconv"
        "os"
)

func fact(n int) int {
        r := 1
        for ; n > 0; n-- {
                r *= n
        }
        return r
}
```

```
func main() {
        if len(os.Args) < 2 {
                log.Fatal("usage: fact NUM")
        }

        n, err := strconv.Atoi(os.Args[1])
        if err != nil {
                log.Fatalf("Invalid number: %v", err)
        }

        fmt.Printf("%s! = %d\n",os.Args[1], fact(n))
}
```

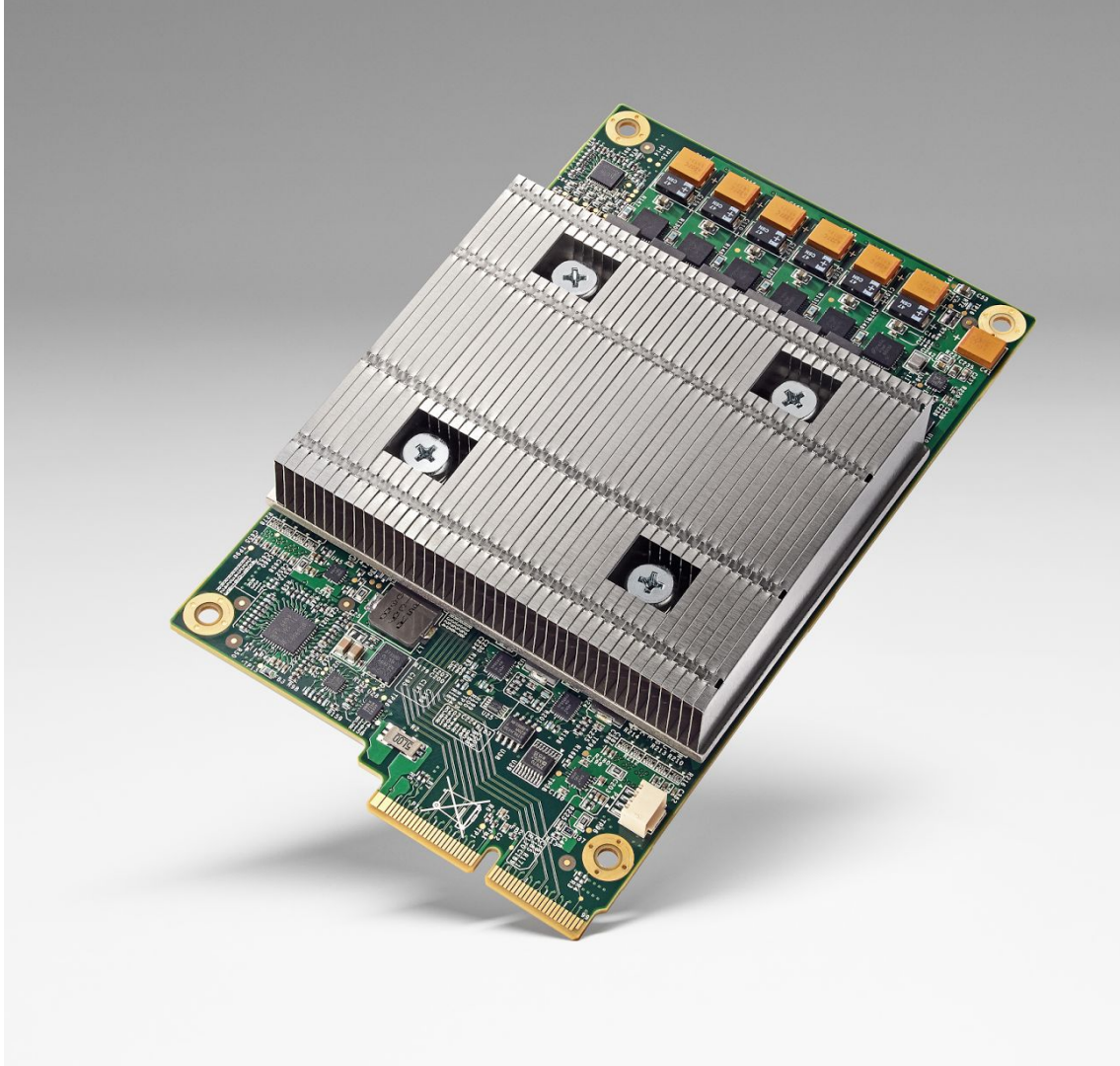
Why RISC-V?

Better architecture – lower power, faster processing, and easier accelerator development. Stable, open community – RISC-V's not going away.

https://goo.gl/7jWmvY (TensorFlow)

https://goo.gl/fimMVO (Zaius)

Google



The Go toolchain is complex.

It has its own compiler, assembler (and assembly language), and linker.

```
TEXT fact(SB),0,$-16

        MOV  n+0(FP), T0
        MOV  $1, T1
loop:
        BLT  ZERO, T0, done
        MUL  T0, T1
        ADDI $-1, T0
        JMP  loop
done:
        MOV  T0, ret+8(FP)
        RET
```

```
.global fact
fact:
        lw   t0,8(sp)
        li   t1,1
loop:
        bltz t0,done
        mul  t1,t0,t1
        addi t0,t0,-1
        j    loop
done:
        sw   t0,16(sp)
        ret
```

We’re getting close.

The assembler supports RISC-V, and the compiler and linker are almost done.

We’re aiming for runtime support in Q1 2017.

```
$ cat gotest.go
package main

func fact(n int) int {
        r := 1
        for ; n > 0; n-- {
                r *= n
        }
        return r
}

func main() {
        riscvexit(fact(5))
}
$ GOOS=linux GOARCH=riscv go build gotest.go
$ spike pk gotest; echo $?
120
```

How’s it been?

Good stuff

- The ISA has been very easy to work with.
- The concurrency model is excellent.
- The riscv-opcodes repository is a lifesaver. Merge our pull request!
- Conditional branches are well-designed.

Bad stuff

- Nobody else has good conditional branches, so we had to emulate a flags register.
- Loading 64-bit constants is a pain.

Community involvement

Josh Bleecher Snyder
josharian@gmail.com

Amol Bhave
ambhave@mit.edu

We need your help!

- We're volunteers and would love to have your help.
- Better ABI documentation would speed development.
- The GCC toolchain is frequently broken.
- We have no silicon to test on. Can you give us silicon?

Upstreaming

- We're working closely with upstream to keep in sync with the latest development.
- Upstream is deleting crufty code, which helps us quite a bit.
- We plan to merge our code into upstream as soon as it's finished.

Find us on GitHub: https://github.com/riscv/riscv-go

Josh Bleecher Snyder
josharian@gmail.com

Michael Pratt
mpratt@google.com

Amol Bhave
ambhave@mit.edu

Benjamin Barenblat
bbaren@google.com