Discussion 03

Floating Point; RISC-V Intro

Aditya Balasubramanian

`aditbala [at] berkeley [dot] edu`

Slides available at `teaching.aditbala.com`

Announcements 📣

Agenda

- Floating Point
- RISC-V Intro

Credits to Rosalie Fang

Floating Point

Credits to Rosalie Fang

Floating Point Conversion

ADD PICTURES

Credits to Rosalie Fang

Decimal -> Floating Point

$$Value = (-1)^{Sign} * 2^{Exp+Bias} * 1.mantissa_2$$

10.75

- Convert to Binary w.r.t the floating point
  - `1010.11`
- Shift floating point to match formula format
  - `1.01011 * 2³`
- Read Output
  - `Mantissa = 01100...0`
  - `Exp = 3 - Bias`

Credits to Rosalie Fang

Step Size

- Given a certain exponent, step size is the change in decimal value when we add 1 to the mantissa of binary FP

$$2^{exp+bias} * 1.mantissa => 1M...M.MMMM$$

- Step Size is difference between `1MM...M.MMM0` and `1MM...M.MMM1`
- 2^{-4} in this example

Credits to Rosalie Fang

RISC-V

Credits to Rosalie Fang

Assembly Basics

Assembly is...

- The direct output of compiled code
- Is not the final form of code
- Is still human-readable
- A set of instructions that can be directly understood by the system, after maybe some minor adjustments
- Built up of a single operation at a time
- Even more dumb than regular computer programs
- Read line by line when executed, except when told not to

Credits to Rosalie Fang

Storage: Registers

- On-chip memory
- RV32 has 32 of them numbered x0-x31 (why not x32?)
- They are all functionally the same but conventionally different
- They're all 32 bits wide
- Anything can be stored in them (no types)
- NOT A VARIABLE

INSERT IMAGE

Credits to Rosalie Fang

Register Specifics

There are 4 general categories (and some special!):

1. Argument registers: `a0` - `a7`
2. Return value registers: `a0`, `a1`
3. Saved registers: `s0` - `s11`
4. Temporary registers: `t0` - `t6`

   Special registers

- Return address: `ra` NOT RETURN VALUE!!!

  Zero: `x0`

  Stack pointer: `sp`

  Other pointers: `tp`, `gp`

Credits to Rosalie Fang

RISC-V Greensheet!

Credits to Rosalie Fang

Loads

l<u>x</u> rd imm(rs1)

- Loads `n` bits worth of data from memory address: `rs1 + imm` where `rs1` <u>should</u> already be a valid address.
- `n` will depend on the instruction: `sb` = 8 bits, `sh` = 16 bits, `sw` = 32 bits, `sd` (not used usually) = 64 bits
- If we have too few bits...
  - Sign-extend
- If we have too many bits...
  - Take the 32-most LSB bits!

Credits to Rosalie Fang

Stores

s<u>x</u> rd imm(rs1)

- Saves `n` bits worth of data to the memory address: `rs2 + imm`
- Truncates to `n` LSB bits to be stored

Credits to Rosalie Fang

Jump Instructions, `jal`, `jalr`, `j`, `jr`

	Saves return address	Jump and no return
PC-relative address	`jal ra, label`	`j label`
Address in a register	`jalr ra, rs1, imm`	`jr rs1`

`j` and `jr` are shorthand for common combinations of certain instructions and register usages.

```
j label # jal x0, label
jr rs1 # jalr x0, 0(rs1)
```

Credits to Rosalie Fang