Lecture 22

RISC-V

- Called instruction set architecture
- All programs running on the same computer use the same instruction set
- Instructions encoded as machine language, i.e. its binary representation
- instruction set does not define hardware
- Operands: registers, memory, constants
- Registers: 32 registers, 32 bit each

Arithmetic instructions

- $a = b + c$: `add s0, s1, s2`
- $a = b - c$: `sub s0, s1, s2`
- $a = b + c - d$:

```
add t0, s1, s2
s0, t0, s3
```

- Multiple simple instructions are generally faster than one complex instruction with more operands
- Registers are fast to access but there is only a limited number
  - X0: always 0
  - s0-s11, t0-t11: general purpose for variables
  - ra, a0-a7: functions calls
  - sp, gp, tp: will be covered later
- Constants: value is immediately available from instruction rather than register or memory

Example a += 4 becomes `addi s0, s0, 4`
b = a - 12 becomes `addi s1, s0, -12`
i = 0 becomes `addi, s4, zero, 0`
x = 2032 becomes `addi, s5, zero, 2032` y = -78 becomes `addi, s6, zero, -78`

- Number systems: prepend 0x, 0b, etc
- Immediates are 12 bit signed integers
- Larger numbers (lui)
  - loads upper immediate followed by `addi`
  - loads 20 bit value into most significant 20 bits of reg, rest are 0

Example `int a = 0xABCDE123`:

```
lui s2, 0xABCDE
addi s2, s2, 0x123
```

- pseudo instructions can also be used