Computer Systems from the Ground Up

Winter 2024

https://cs107e.github.io

Who?

Anna

Ben

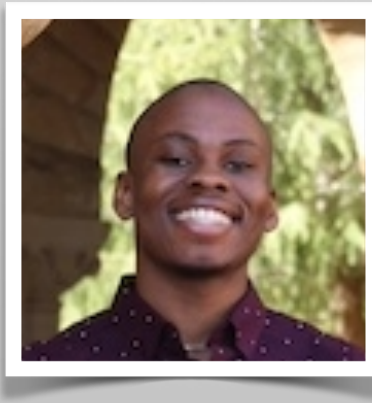
Didi

Ishita

+ you!

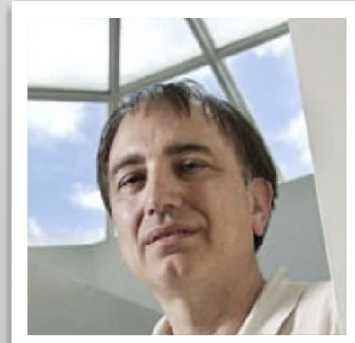
Intrepid young padawans

Julie



Kenny

Liana



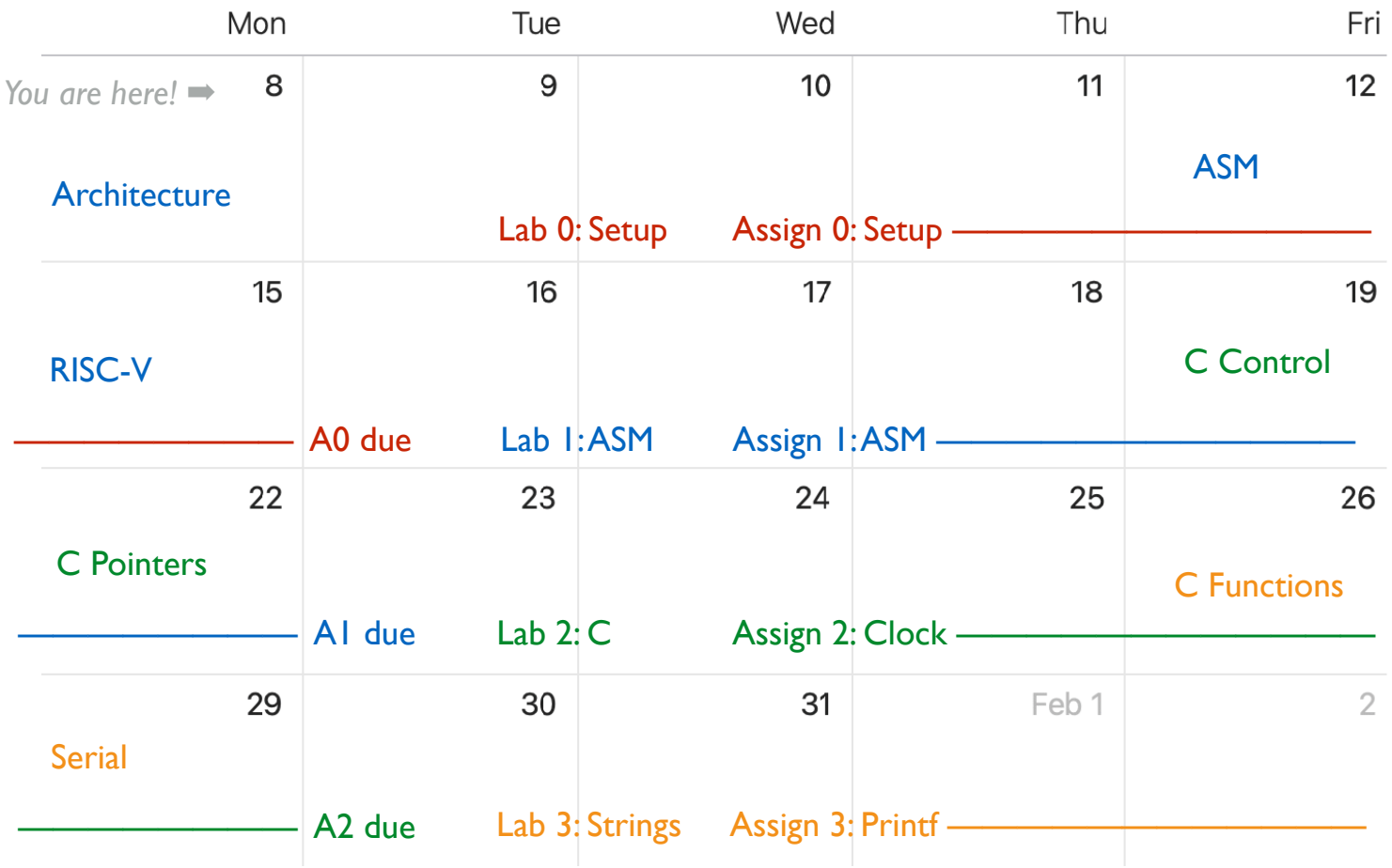
Pat

Learning goals

1) Understand how computers represent data, execute programs, and control peripherals

2) Master your tools

Weekly Cadence



Each week has a focus **topic**

Pair of coordinated **lectures** on Fri and Mon

Lab on Tue/Wed evening

Assignment handed out Wed after lab, due following Tuesday 5pm

Staying on pace leads to best outcomes!

Lectures

Attendance is **necessary**

Content is unique to our course, no textbook
The readings/slides are not a standalone resource
Lectures are not recorded

In-person attendance allows you to participate, ask questions, stay on schedule

Labs

Attendance is **mandatory**

Guided exercises, work with peers, **check in** with staff

Finish lab **ready** for assignment, esp. experience with tricky parts (hardware/software interface)

Philosophy: lots-of-help, hands-on, collaborative

Lab room: Gates B02

Assignments

7 weekly assignments

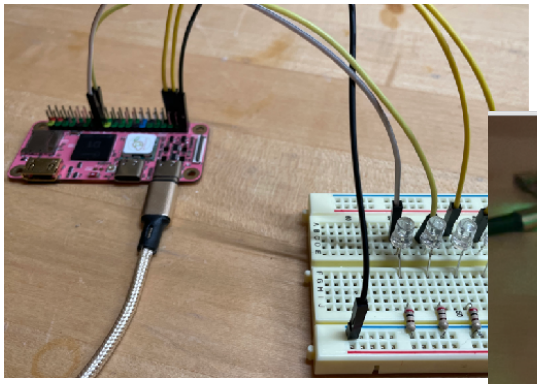
- **Build** on each other, complete full system

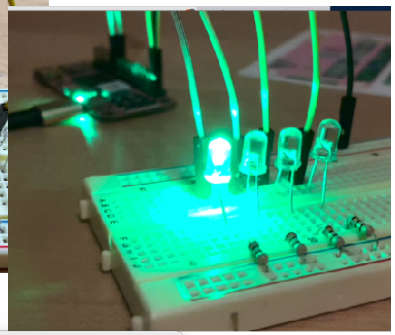
Assignment specifications

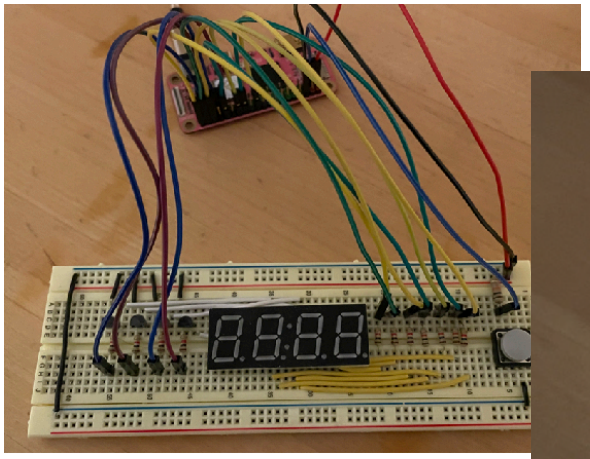
- **Core** (required, tight spec, guided steps)
- **Extension** (optional, opportunity for exploration/creativity)
- Revise and **resubmit** to address issues in core functionality

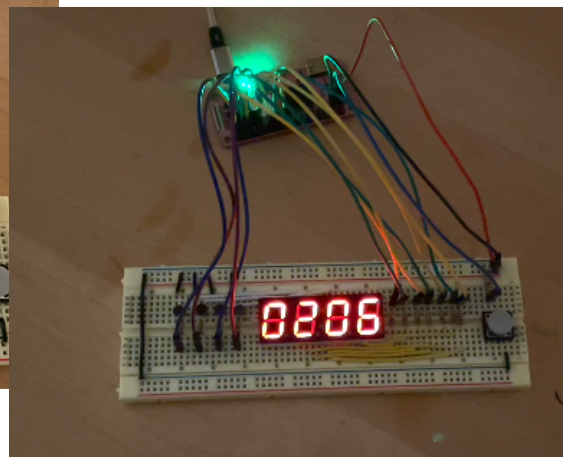
Project

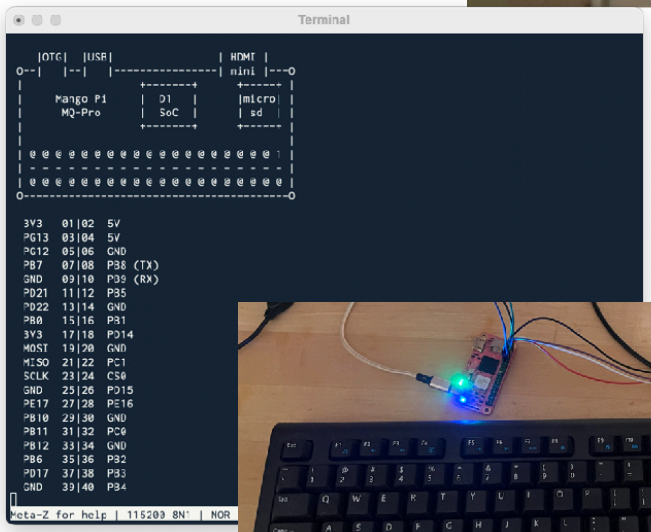
- Design and build **your own system**

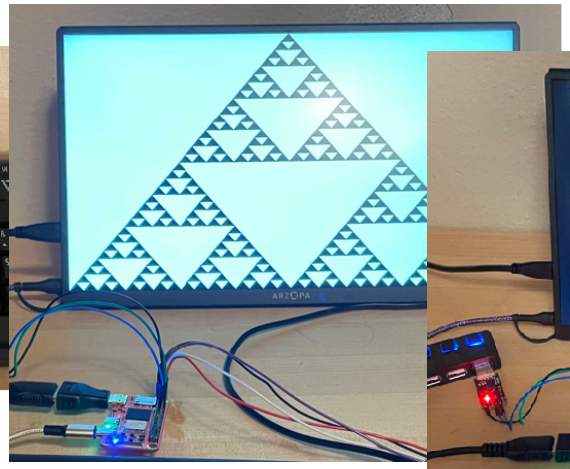


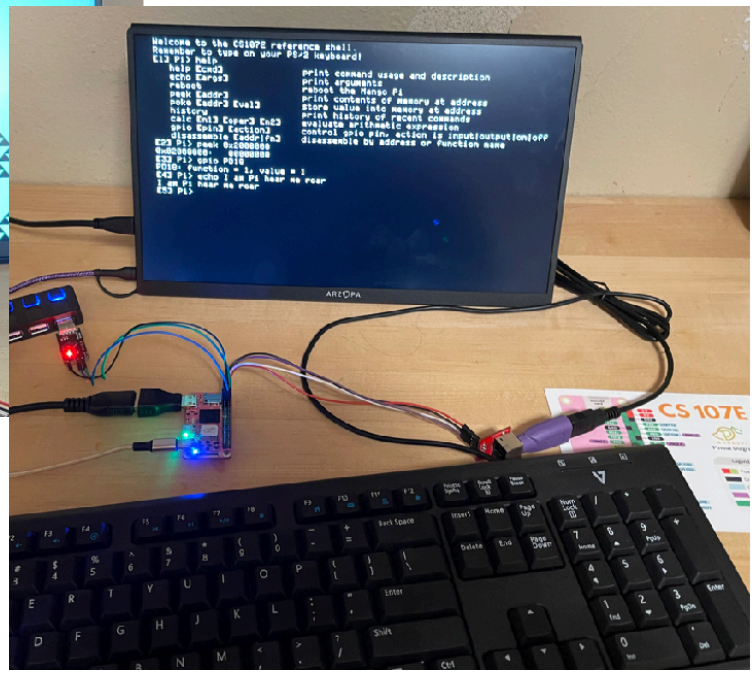






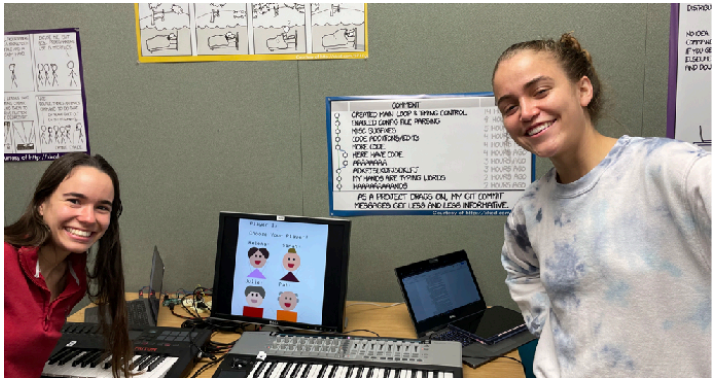


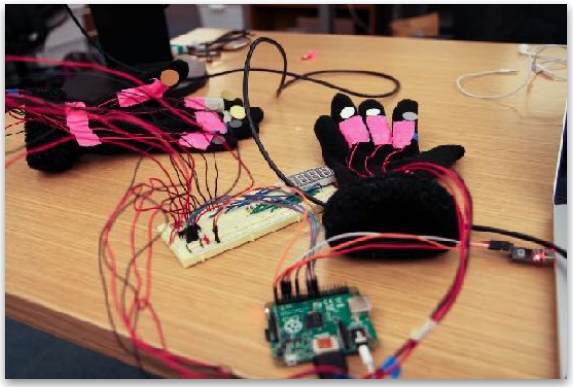


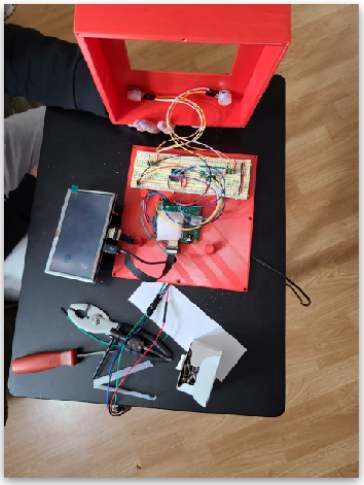


Nearly every instruction is code you wrote yourself!

Projects!





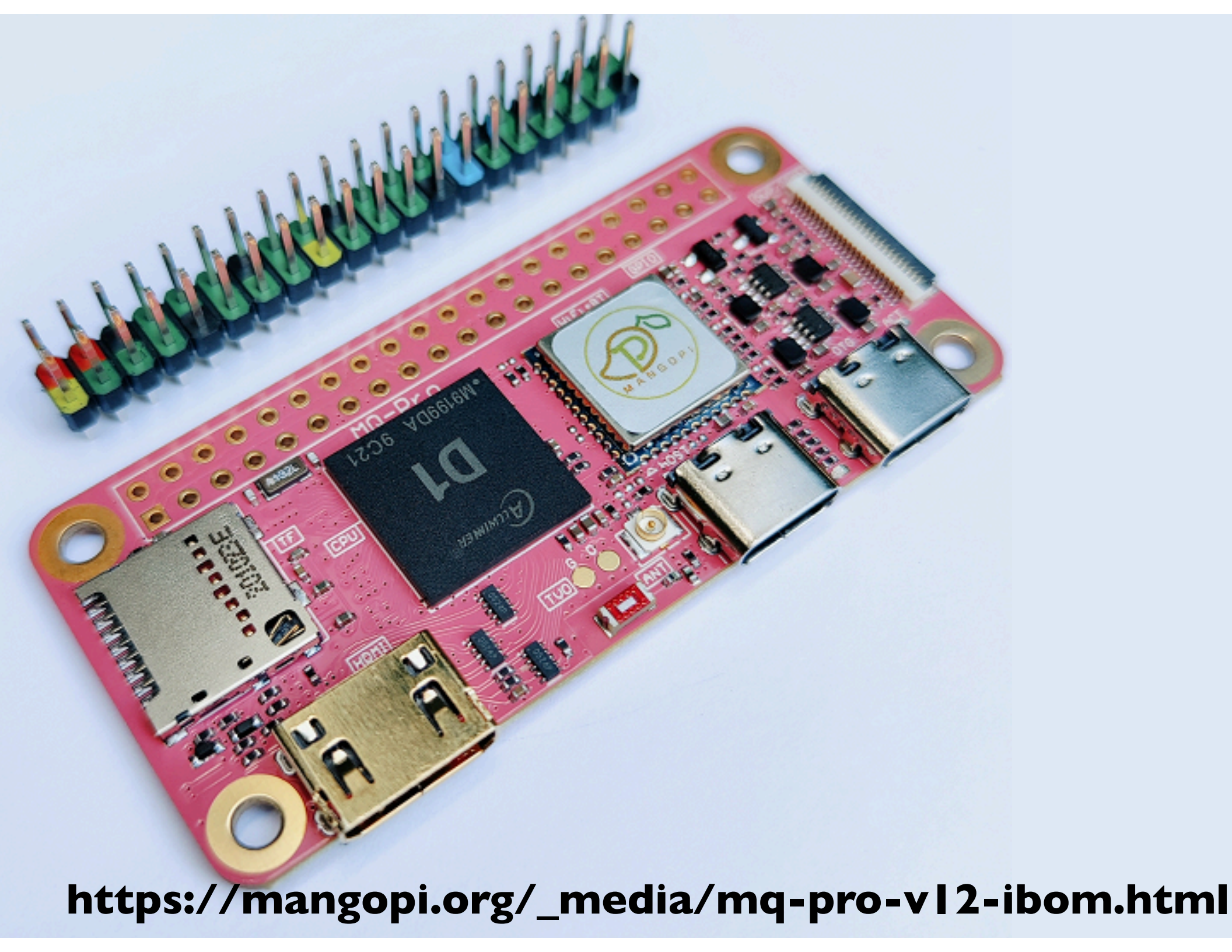


Learning community

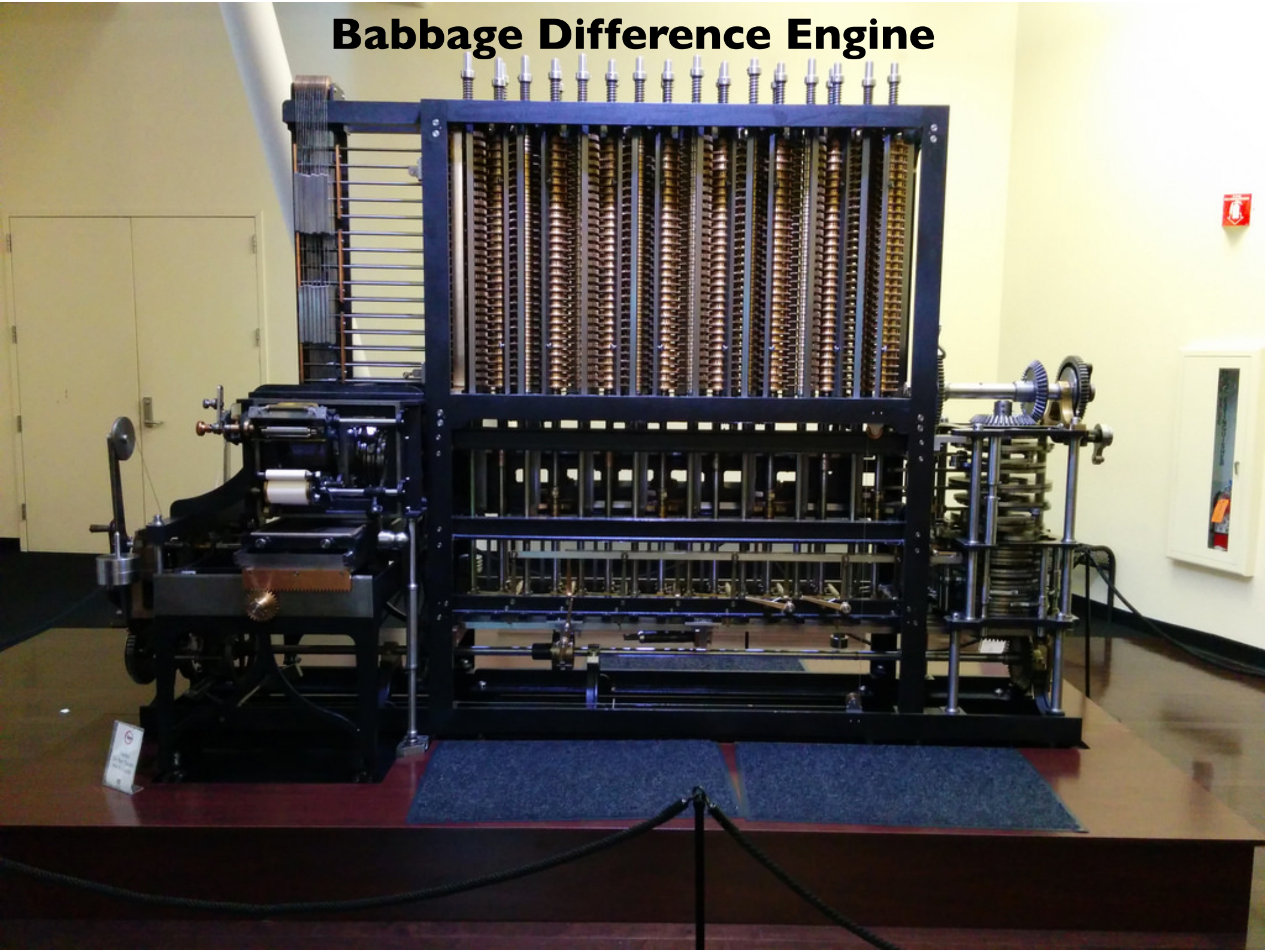
Stay **connected**

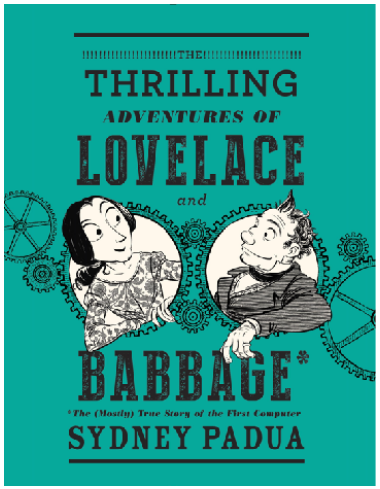
- **Participate** in lecture
- **Collaborate** in lab
- **Discuss** on Ed forum
- **Come** to office hours
- **Meet up** in lab room

Be **curious**. Learn by **doing**. Ask for and offer **help**.


https://mangopi.org/_media/mq-pro-v12-ibom.html

Babbage Difference Engine





Analytical Engine

Universal Computer

http://www.theguardian.com/technology/2015/apr/12/thrilling-adventures-ada-lovelace-charles-babbage-sydney-padua

von Neumann architecture

CPU:
- ALU
- Registers
- Control Unit
  - instruction register, program counter

Memory:
- Stores data and instructions

Mechanisms for input/output

Critical innovation:
- both instructions and data kept in memory
- "stored program" computer

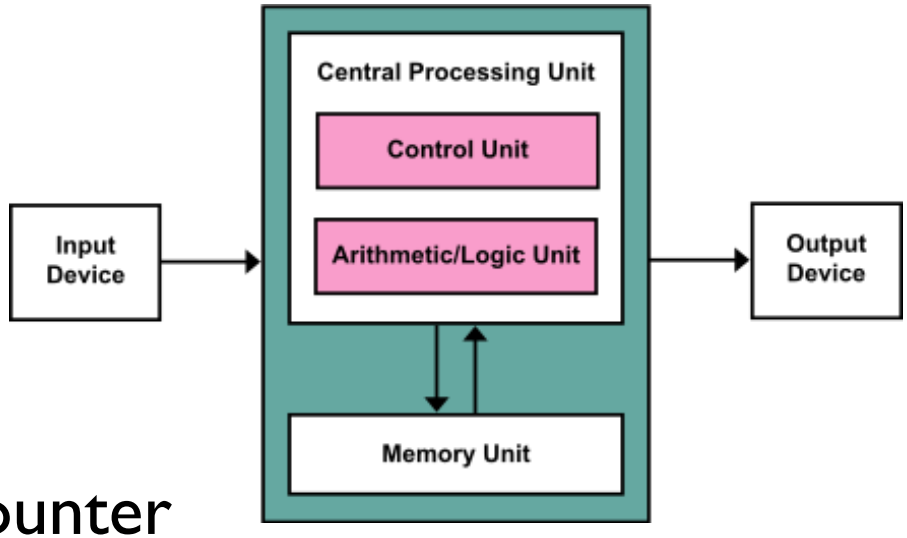


Image credit https://en.wikipedia.org/wiki/Von_Neumann_architecture#/media/File:Von_Neumann_Architecture.svg

Memory Map

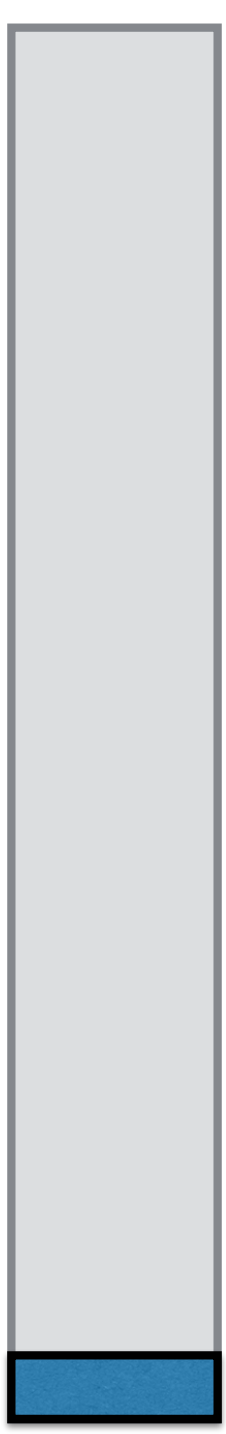
Memory is a large array

Storage locations are accessed using a 64-bit index, called the *address*

Address refers to a *byte* (8-bits)

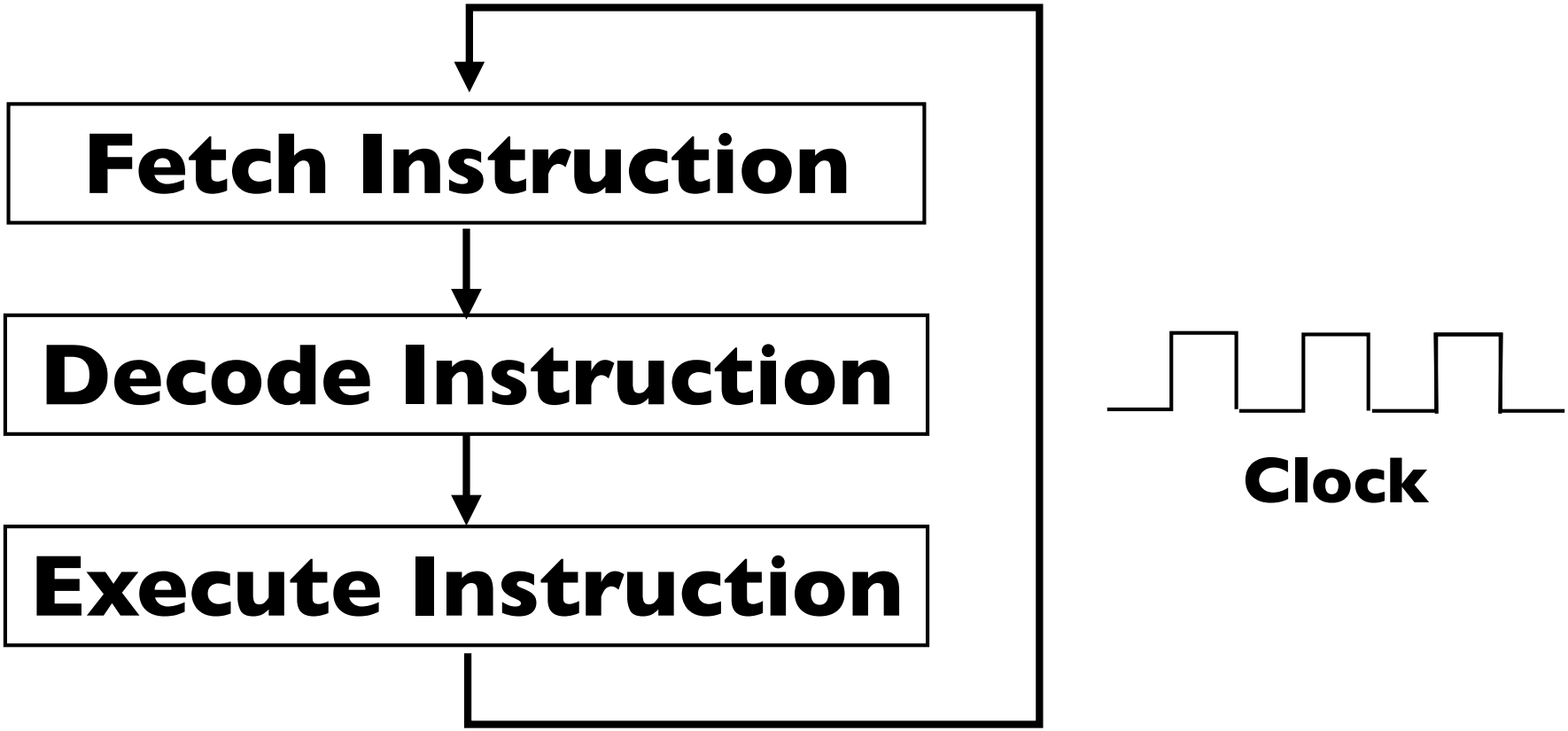
4 consecutive bytes form a *word* (32-bits)

Maximum addressable memory is 16EB (But... 42-bit address lines)

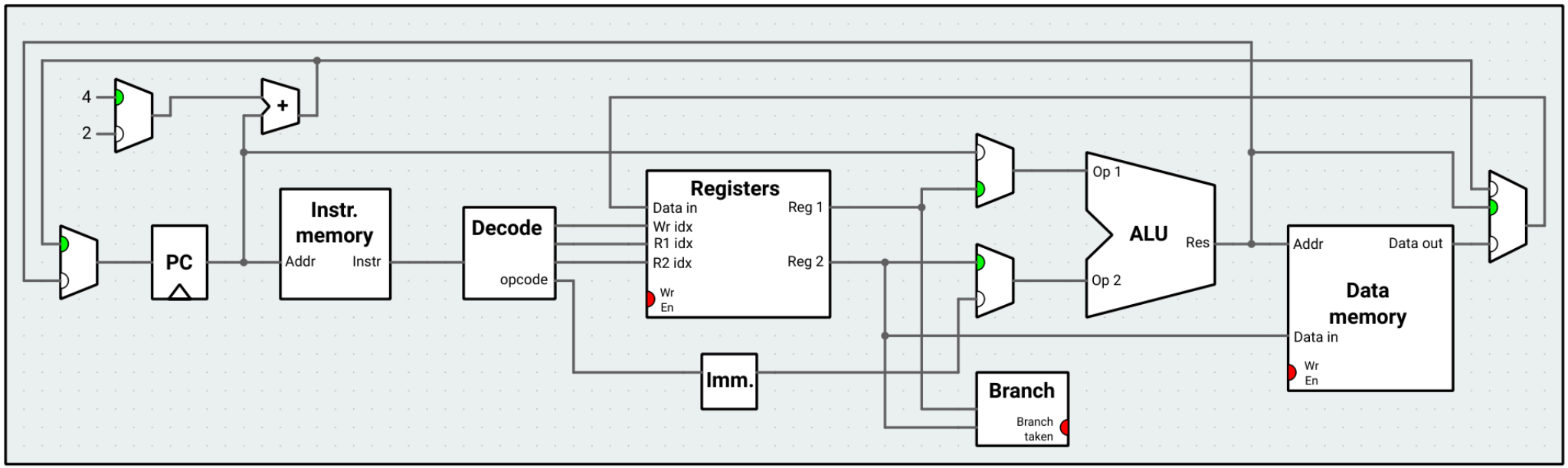


1GB physical memory

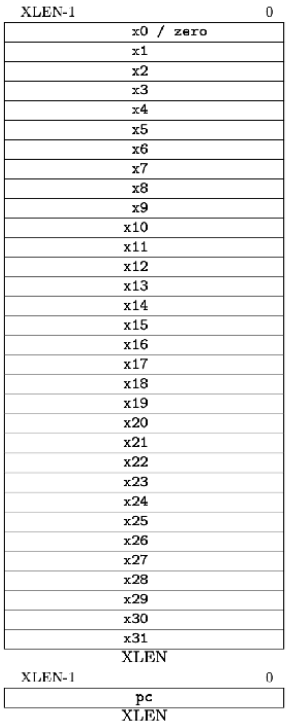
Running a Program



RISC-V Architecture / Floor Plan



https://ripes.me/

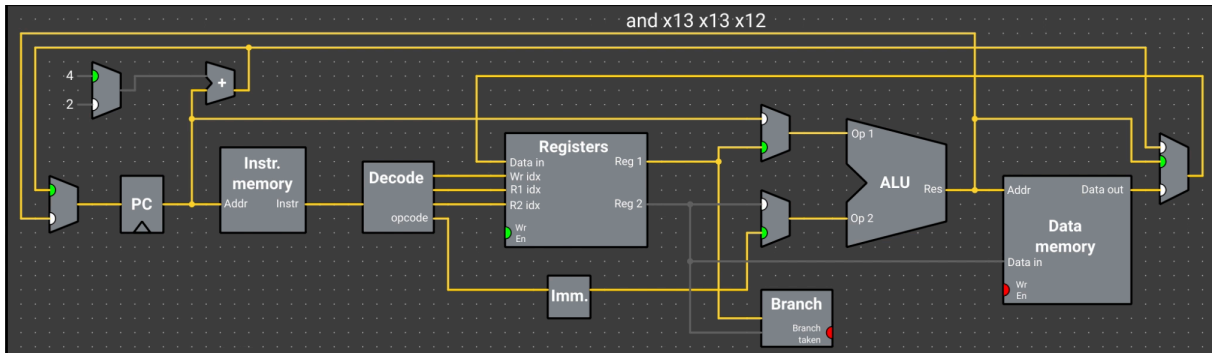


32 registers (x0-x31)

PC = program counter

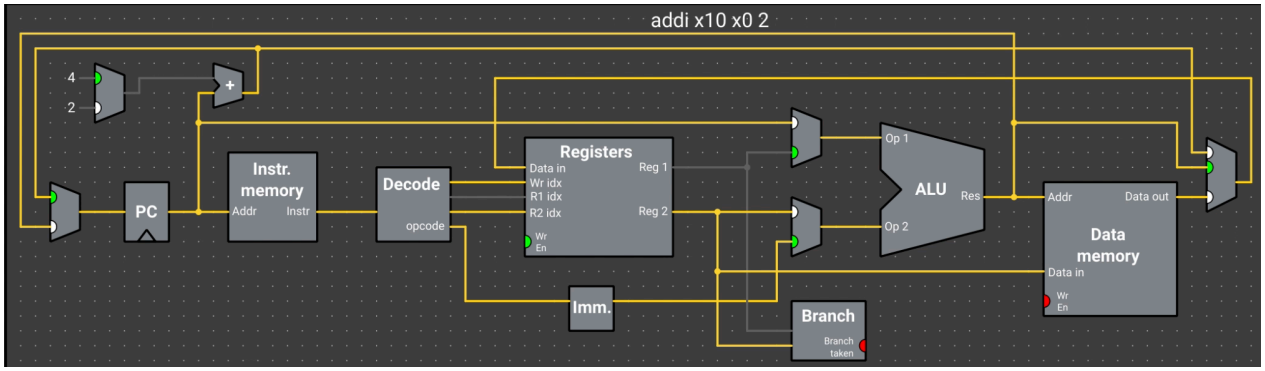
Data paths dictate constraints on operations

```
and a3,a3,a2
```



What are possible inputs to ALU?
Where does result go?

```
addi a2,zero,2
```



Where does an immediate value come from?
Where can it flow to?

Some ALU instructions

```
add rd,rs1,rs2
sub rd,rs1,rs2
and rd,rs1,rs2
or  rd,rs1,rs2
sll rd,rs1,rs2
srl rd,rs1,rs2
```

R-type (three registers: dest, source1, source2)

```
addi rd,rs,imm12
andi rd,rs,imm12
ori  rd,rs,imm12
slli rd,rs,imm12
```

I-type (source2 is constant not register)

First week: set up

Before lab:
- Review course guides:
    Unix tools (shell, editor, git)
    Electricity (simple circuits, Ohm's law)
    Number representation (binary, bit operations)
- Install development tools

During lab:
- Establish comfort with background topics
- Practice with environment/tools, build productive habits
- Get help resolving any installation snags
- Meet one another!