

CS107e

Computer Systems from the Ground Up

Chris Gregg,
Maria Fernandez, Blake Jones,
Liana Keesing, Anna Mistele

Autumn 2022

<https://cs107e.github.io/>



Chris



Maria



Blake



Liana



Anna

Who is Chris Gregg?

- Career:
 - Johns Hopkins University Bachelor's of Science in Electrical and Computer Engineering
 - Seven years active duty, U.S. Navy (14+ years reserves)
 - Harvard University, Master's of Education
 - Seven years teaching high school physics (Brookline, MA and Santa Cruz, CA)
 - University of Virginia, Ph.D. in Computer Engineering
 - Three years teaching computer science at Tufts University
 - Senior Lecturer at Stanford (arrived, Fall 2016)
- Personal website: <https://web.stanford.edu/~cgregg>



Learning Goal I

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

OK

```
int counter;  
int calc() {...}
```

```
int a = 20;  
unsigned int b = 6;  
if (a < b) {...}
```

Not OK

```
int calc() {  
    int counter;  
    ...  
}
```

```
int a = -20;  
unsigned int b = 6;  
if (a < b) {...}
```

OK

```
int counter;  
int calc() {...}
```

Not OK

```
int calc() {  
    int counter;  
    ...  
}
```

```
int a = 20;  
unsigned int b = 6;  
if (a < b) {...}
```

```
int a = -20;  
unsigned int b = 6;  
if (a < b) {...}
```

Why?

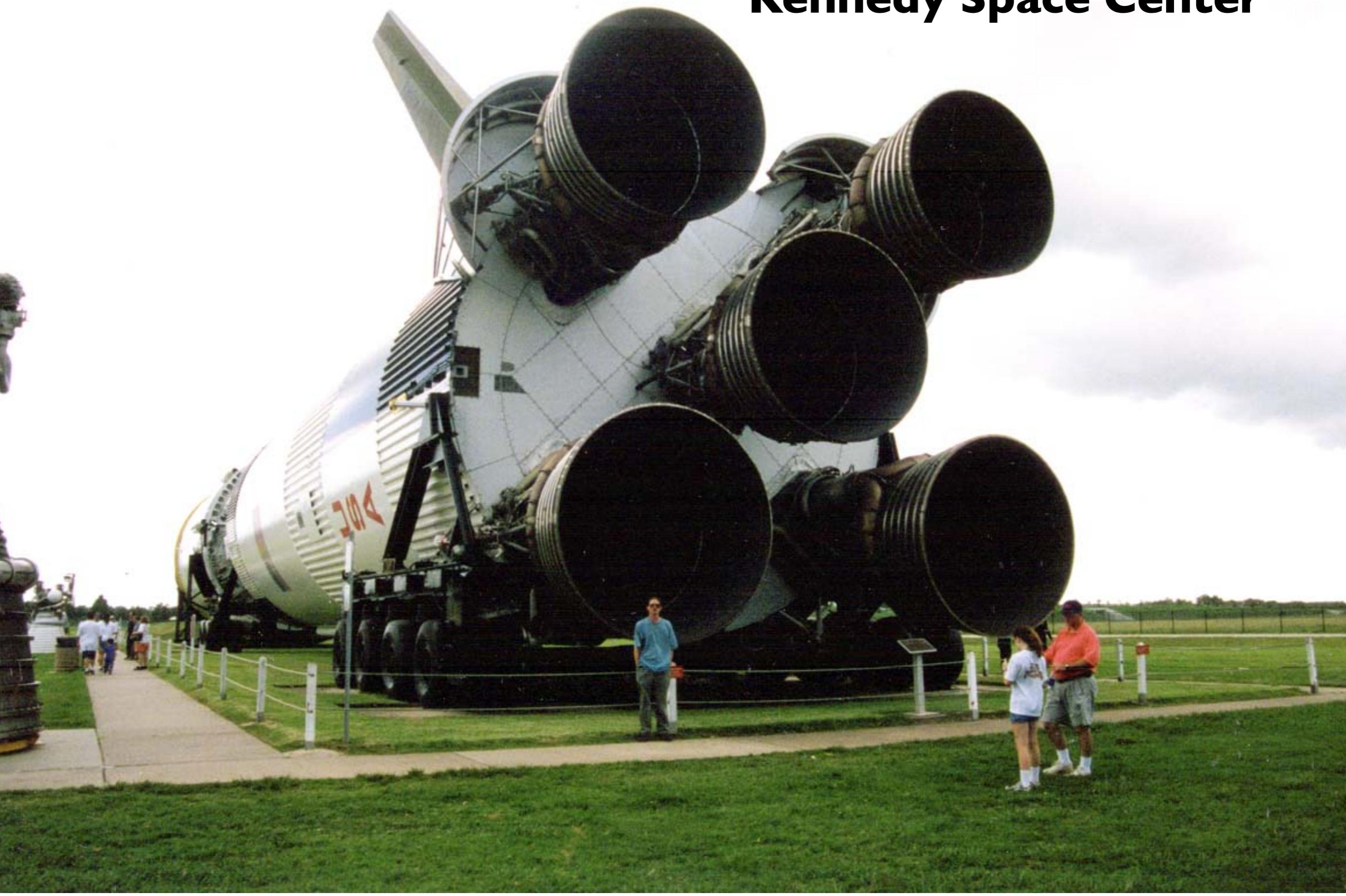
```
int main( ) {  
    ...  
}
```

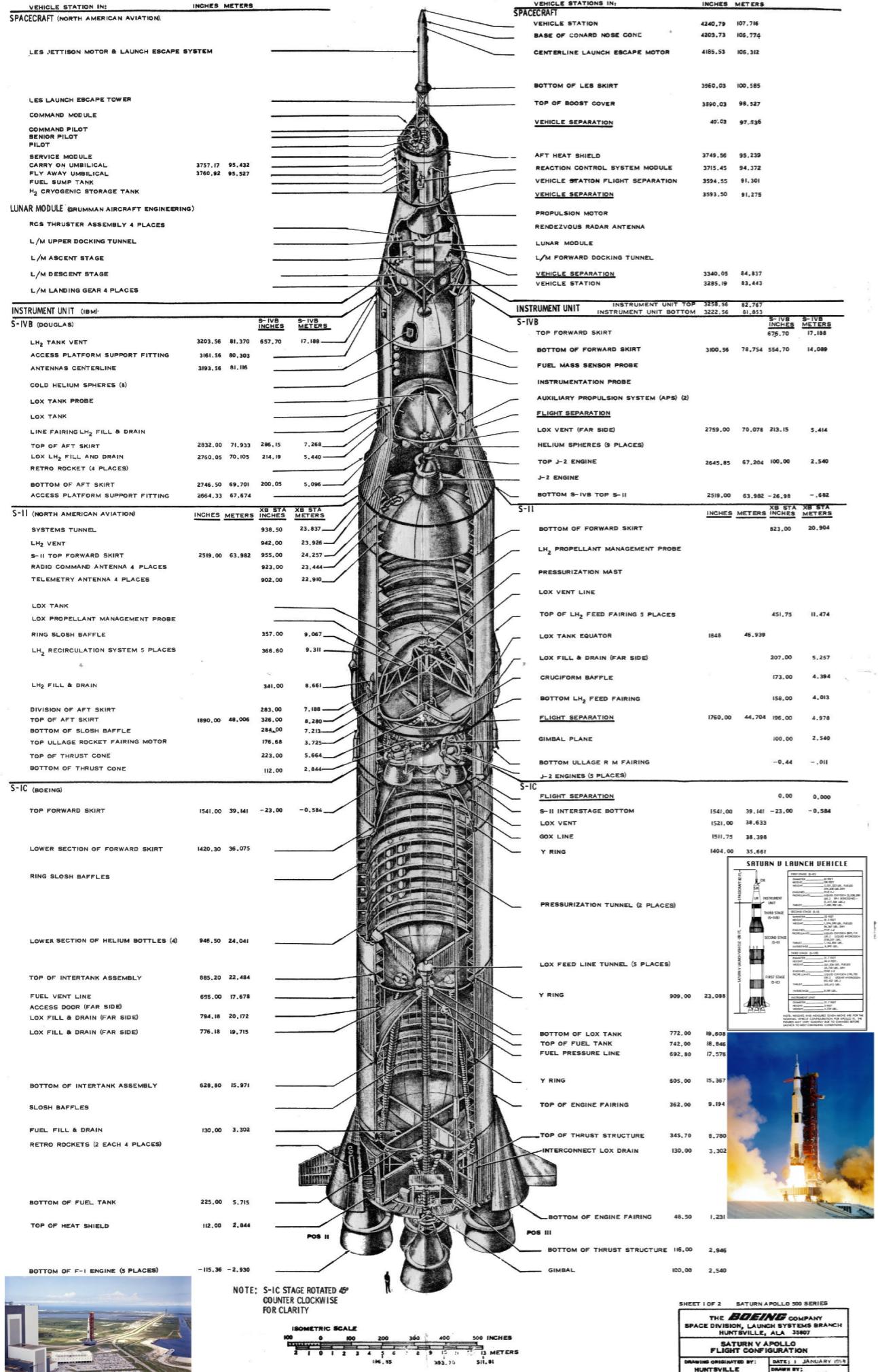
How does your program start
at the first instruction of
main()?
Or does it really?

First steps are often the hardest

That's why we're here!

Saturn V Kennedy Space Center





Command Module 64,000 lbs

Saturn V 6,200,000 lbs

Payload 1.5% of total weight

Falcon 9





Engineer for Excellence!

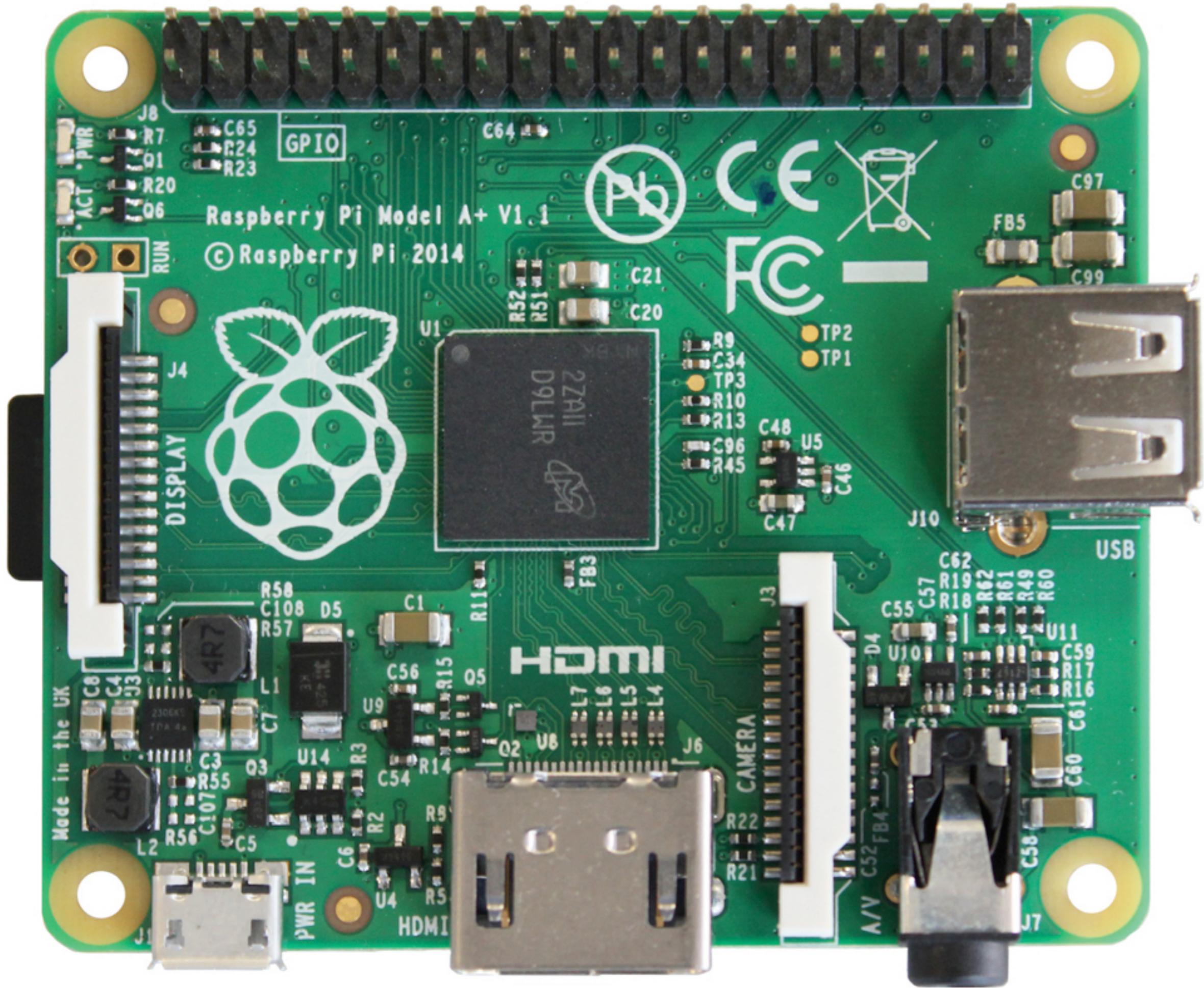
Perseverance!

Bare Metal on the Raspberry Pi

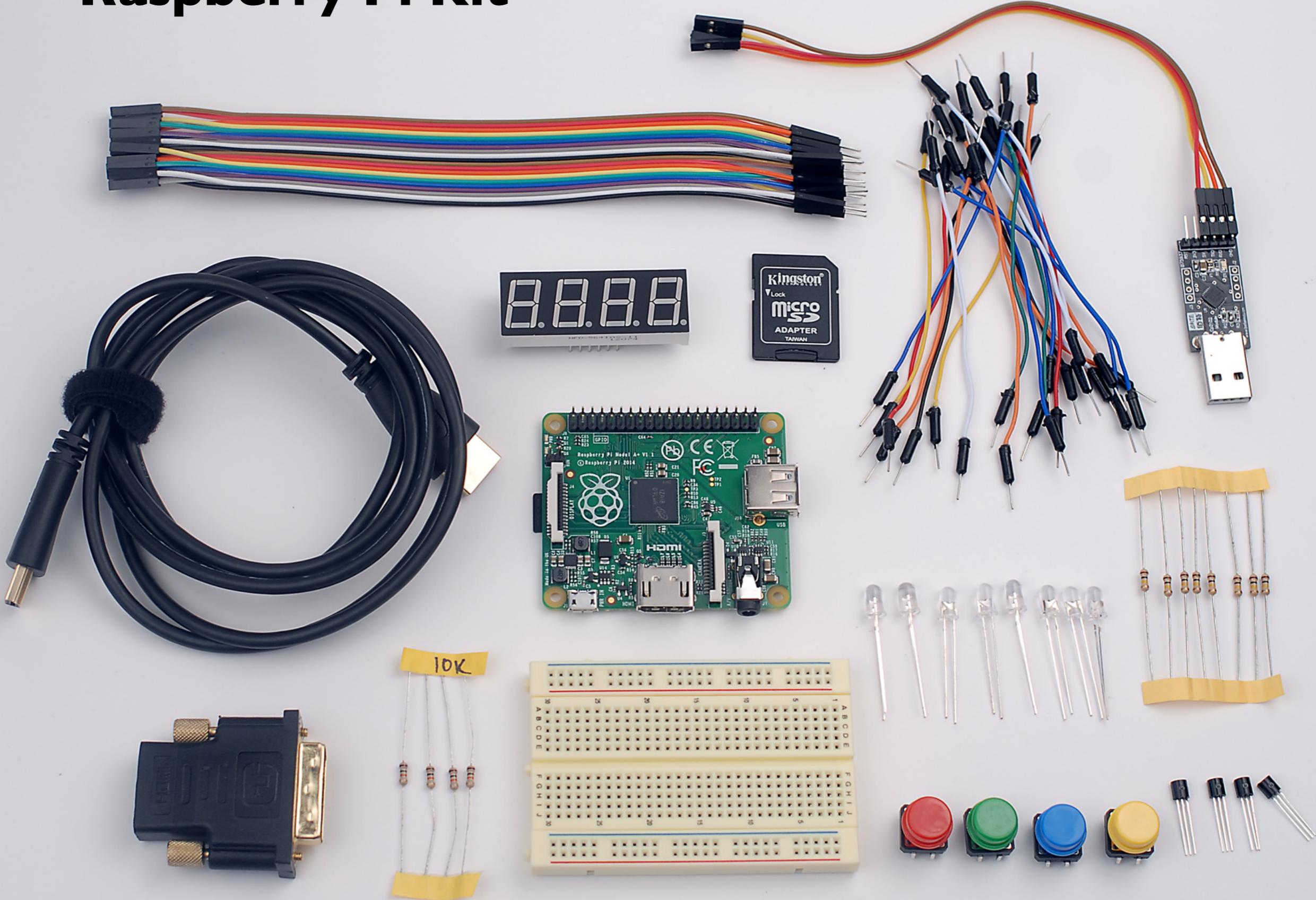
Definition: Bare metal programming involves no operating system (programmer constructs libraries)

Bare metal programs boot and startup on their own, and directly control peripherals

You'll understand every line of code in the system.



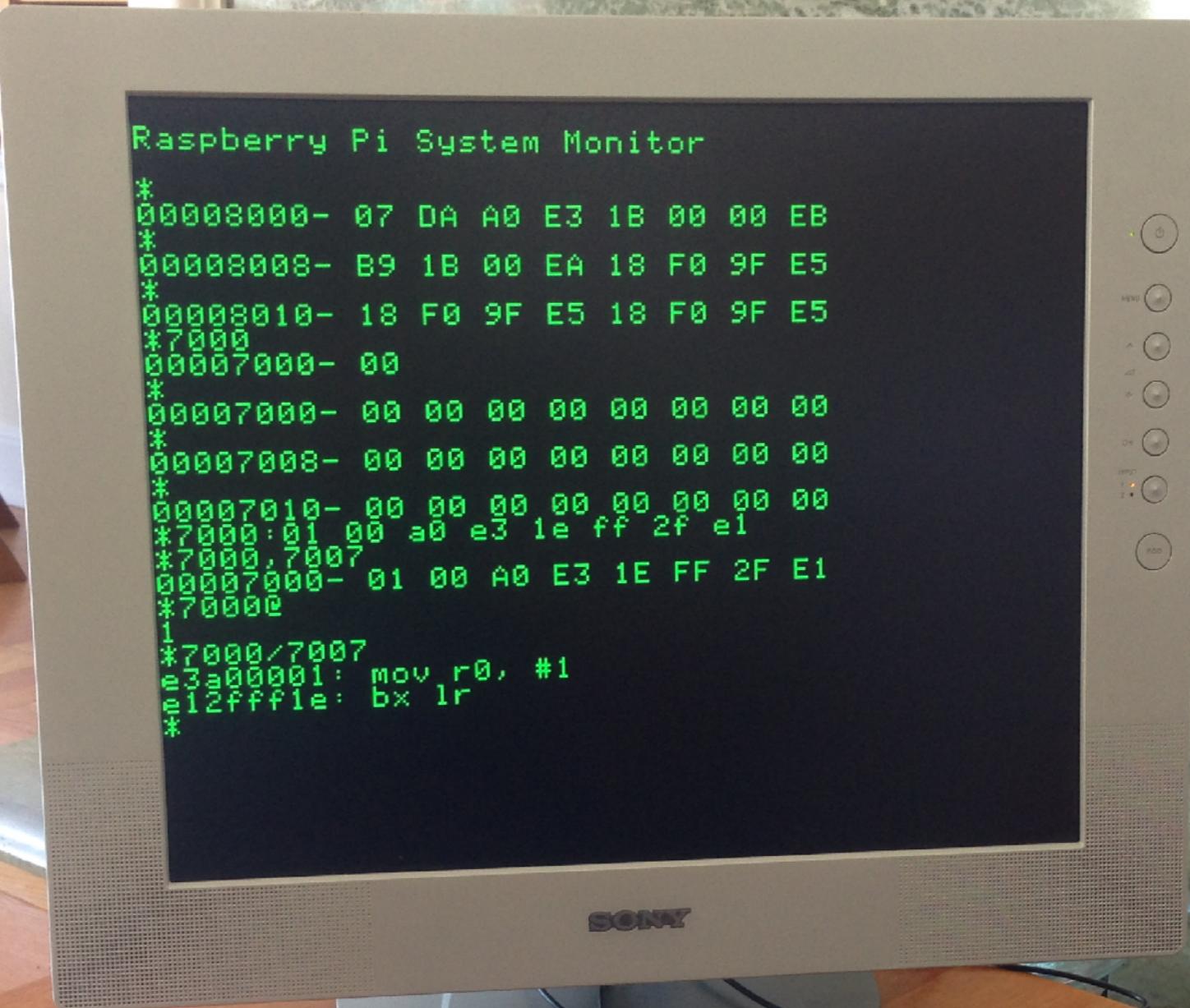
Raspberry Pi Kit



Raspberry Pi Shell

Raspberry Pi System Monitor

```
*00008000- 07 DA A0 E3 1B 00 00 EB
*00008008- B9 1B 00 EA 18 F0 9F E5
*00008010- 18 F0 9F E5 18 F0 9F E5
*7000
00007000- 00
*00007000- 00 00 00 00 00 00 00 00
*00007008- 00 00 00 00 00 00 00 00
*00007010- 00 00 00 00 00 00 00 00
*7000:01 00 a0 e3 1e ff 2f e1
*7000,7007
00007000- 01 00 A0 E3 1E FF 2F E1
*7000@
1
*7000/7007
e3a00001: mov r0, #1
e12fffffe: bx lr
*
```



**Almost every instruction
will be code you've written!**

Learning Goal 2

**Master your tools
Learn their value**

Software Tools

UNIX command line: bash, cd, ls, ...

Programming languages: C, ...

gcc

as

ld

binutils: nm, objcopy, objdump, ...

make

git and github.com

documentation: markdown

Different Tools for Different Jobs



<http://dans-woodshop.blogspot.com/>

Organized Development Environment



<http://amhistory.si.edu/juliachild/>

Don't Avoid Activation Energy

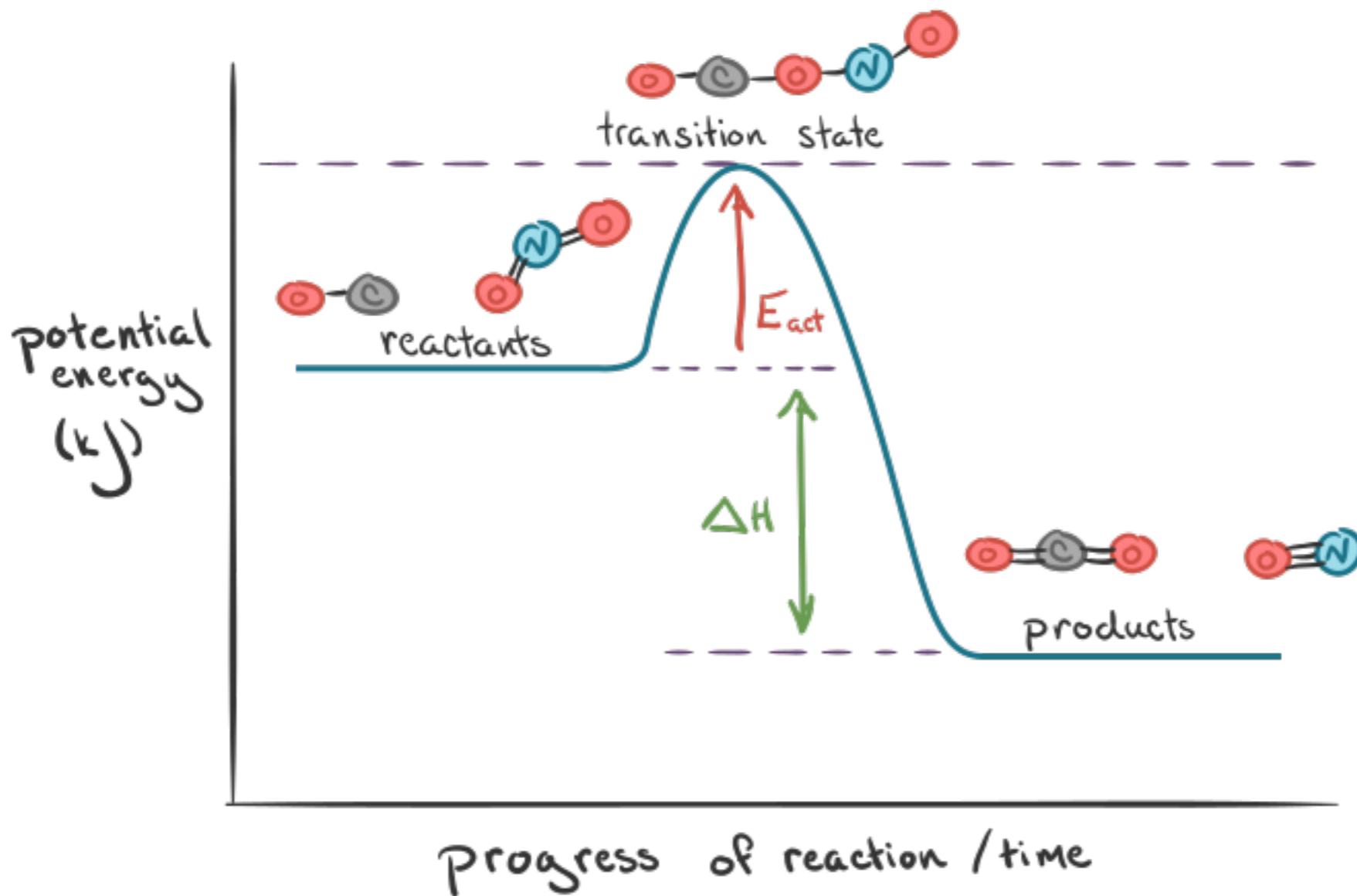


Figure from Khan Academy

<https://www.khanacademy.org/test-prep/mcat/chemical-processes/thermochemistry/a/endothermic-vs-exothermic-reactions>

Don't Avoid Activation Energy

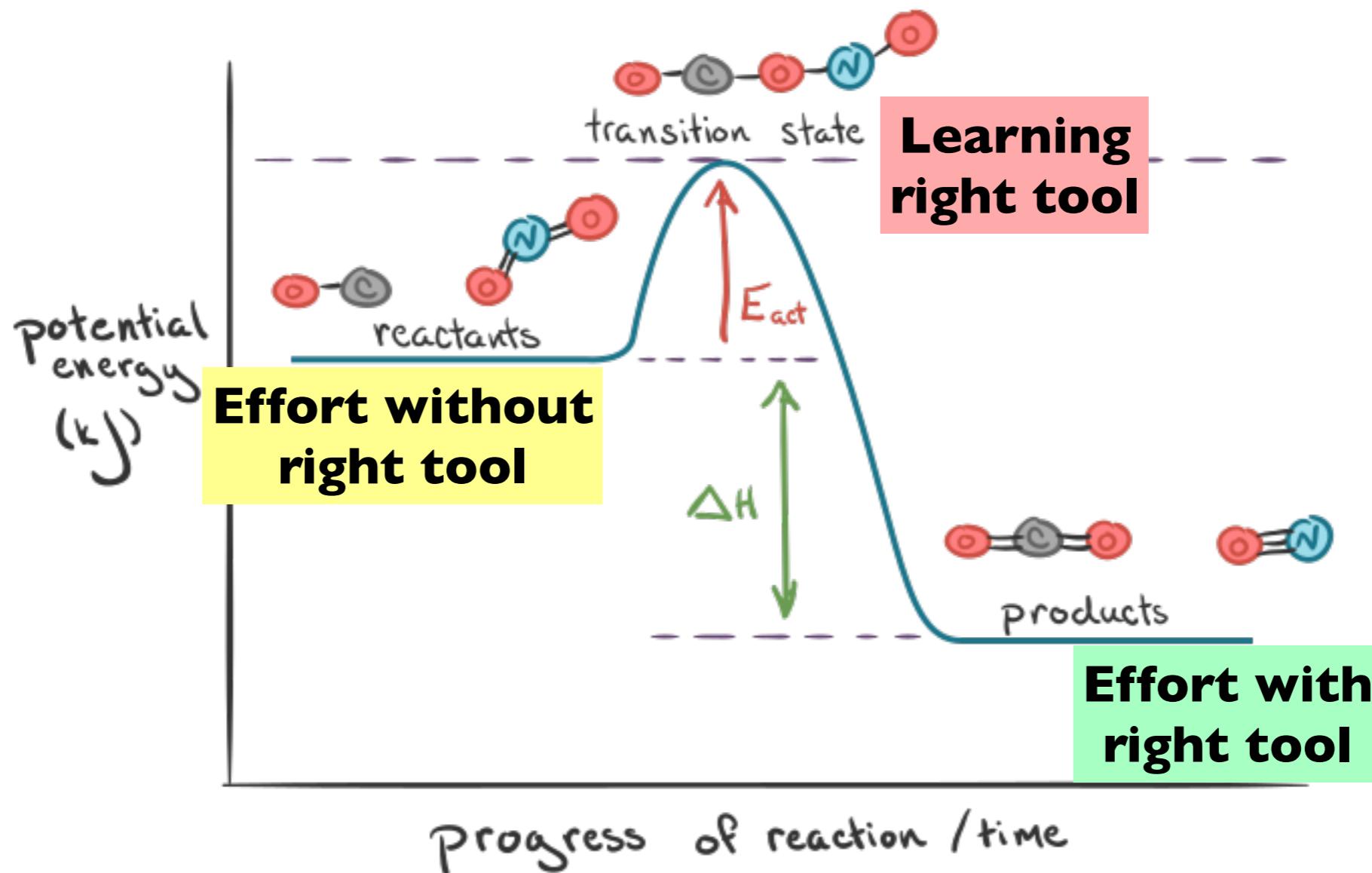


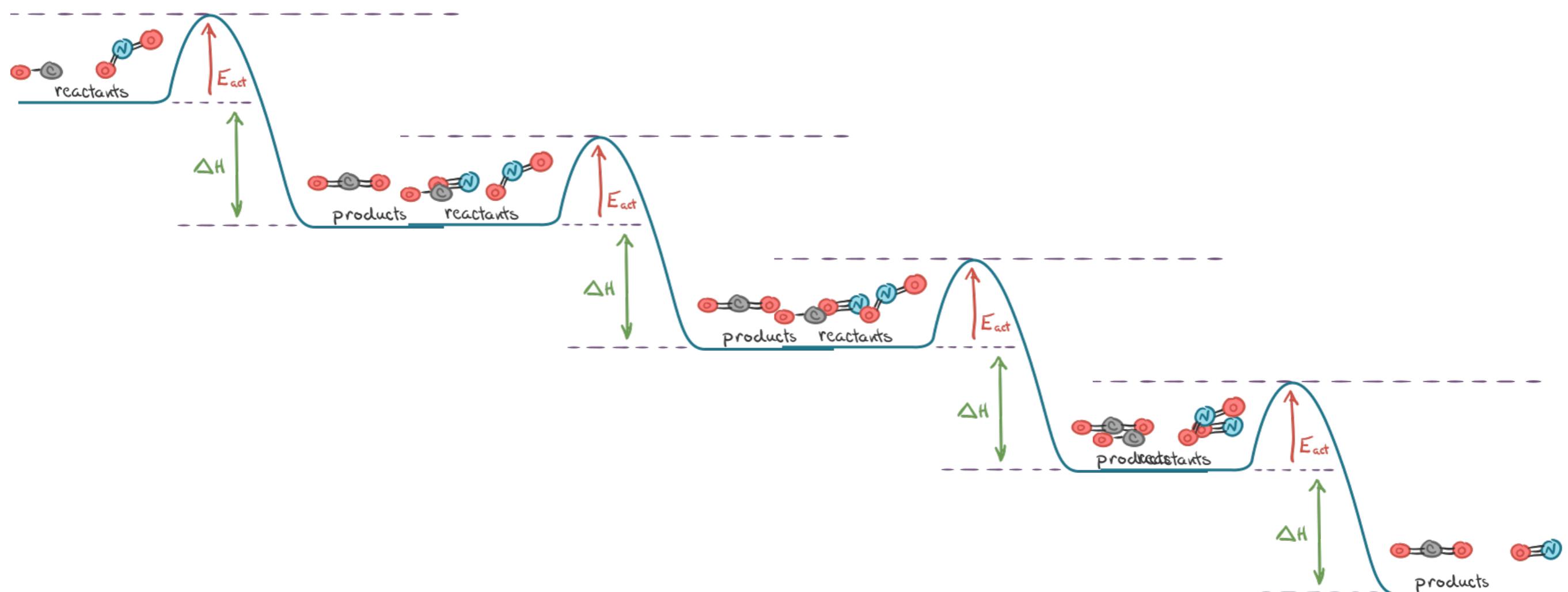
Figure from Khan Academy

<https://www.khanacademy.org/test-prep/mcat/chemical-processes/thermochemistry/a/endothermic-vs-exothermic-reactions>



A close-up photograph showing a person's hands working on a piece of wood. The person is using a chisel to shave off thin layers of wood, creating a pile of shavings on the workbench. The workbench is made of light-colored wood and shows signs of use. In the background, there are some tools and a bottle of water. The overall scene suggests a workshop or a woodworking environment.

Practice, Practice, Practice



It never ends... 1000x improvements possible!

Figure from Khan Academy

<https://www.khanacademy.org/test-prep/mcat/chemical-processes/thermochemistry/a/endothermic-vs-exothermic-reactions>

Debugging and Troubleshooting



Administration

Weekly Cadence

Each week has a focus topic

Pair of coordinated lectures on Fri and Mon

Mandatory lab on Wednesdays (either
10:30am-12:30pm, or 1:30pm-3:30pm)

Assignment handed out Wed evening (after lab),
YEAH hours on Thu, assignment due following Tue at
6 pm.

Laboratories

Attendance is **mandatory**

Do exercises and complete check-list

Leave lab ready for assignment: walks you through tricky bit (hardware/software interface) to get you started

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

We will organize your lab into small (2-3 person) breakout groups so you can do the lab with mute off and chat/collaborate. Initially groups are randomly assigned, later in the quarter we will let you choose partners if you want to.

Assignments

7 assignments

- Build on each other

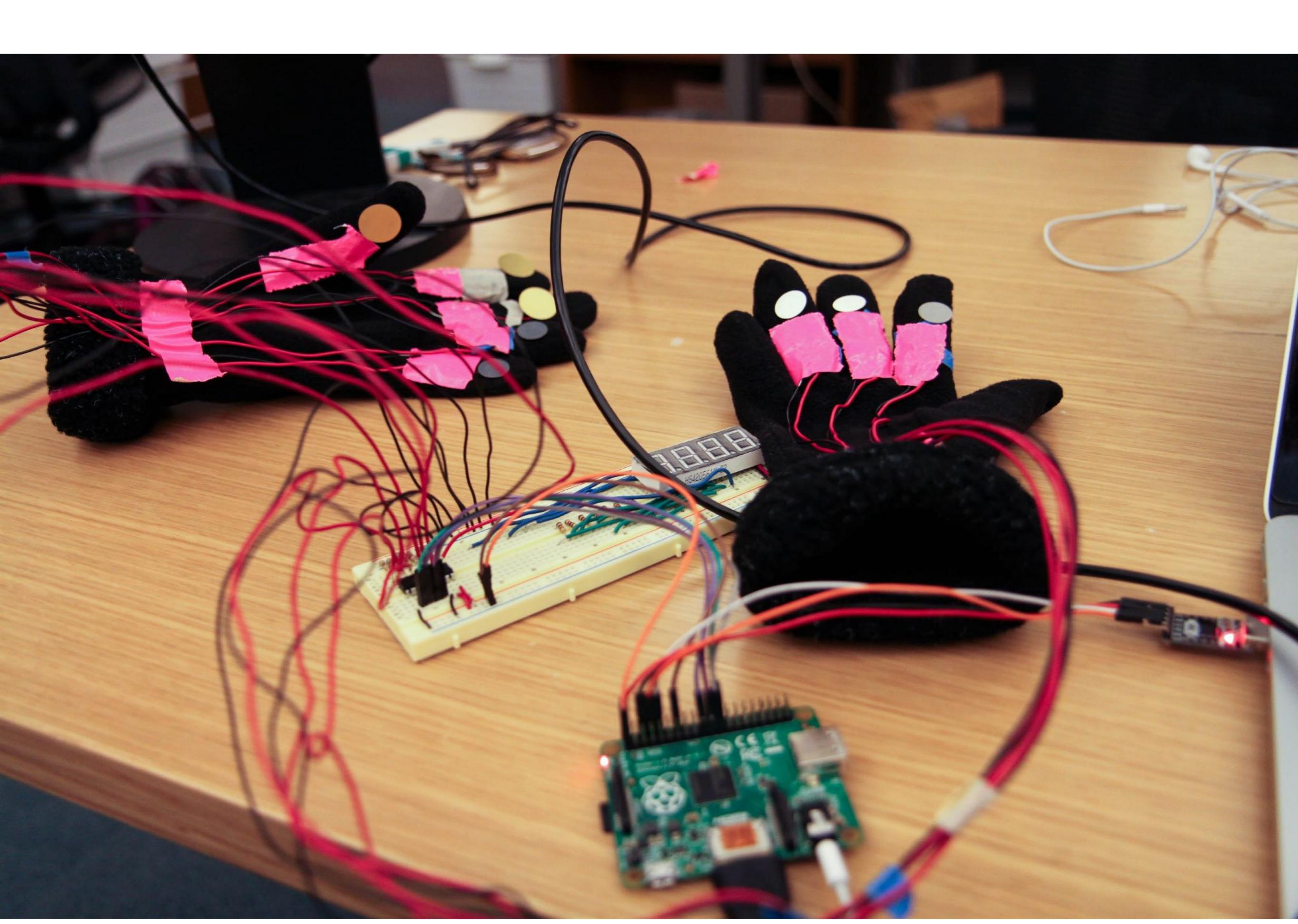
Two parts for each assignment

- Basic (required, tight spec, guided steps)
- Extension (optional, opportunity for your exploration/creativity)

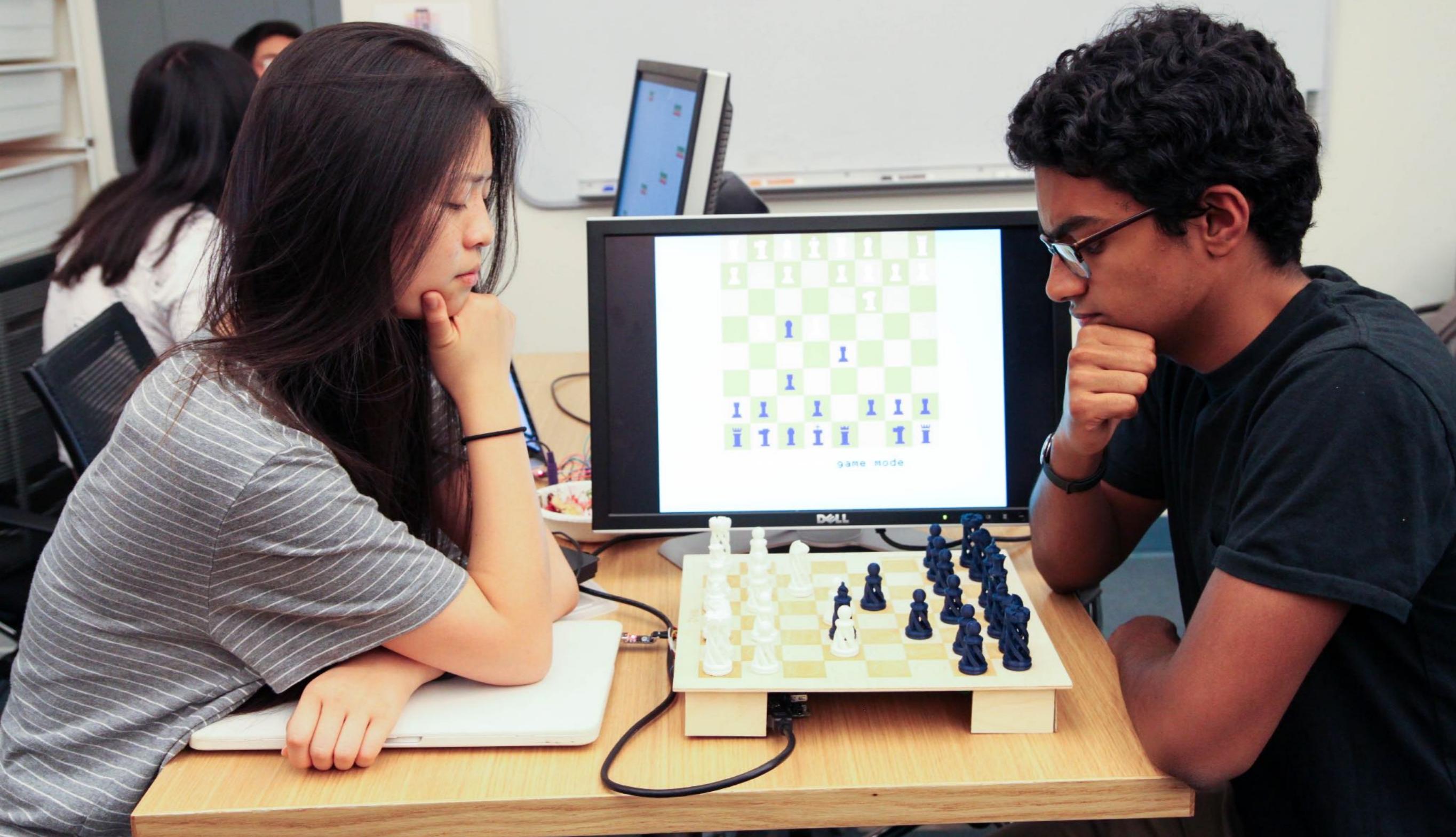
Final project demonstrations on Thursday, December 9

Scaled back due to lack of exam period, physical lab resources

- Encourage you to play with some hardware (e.g. sensors)

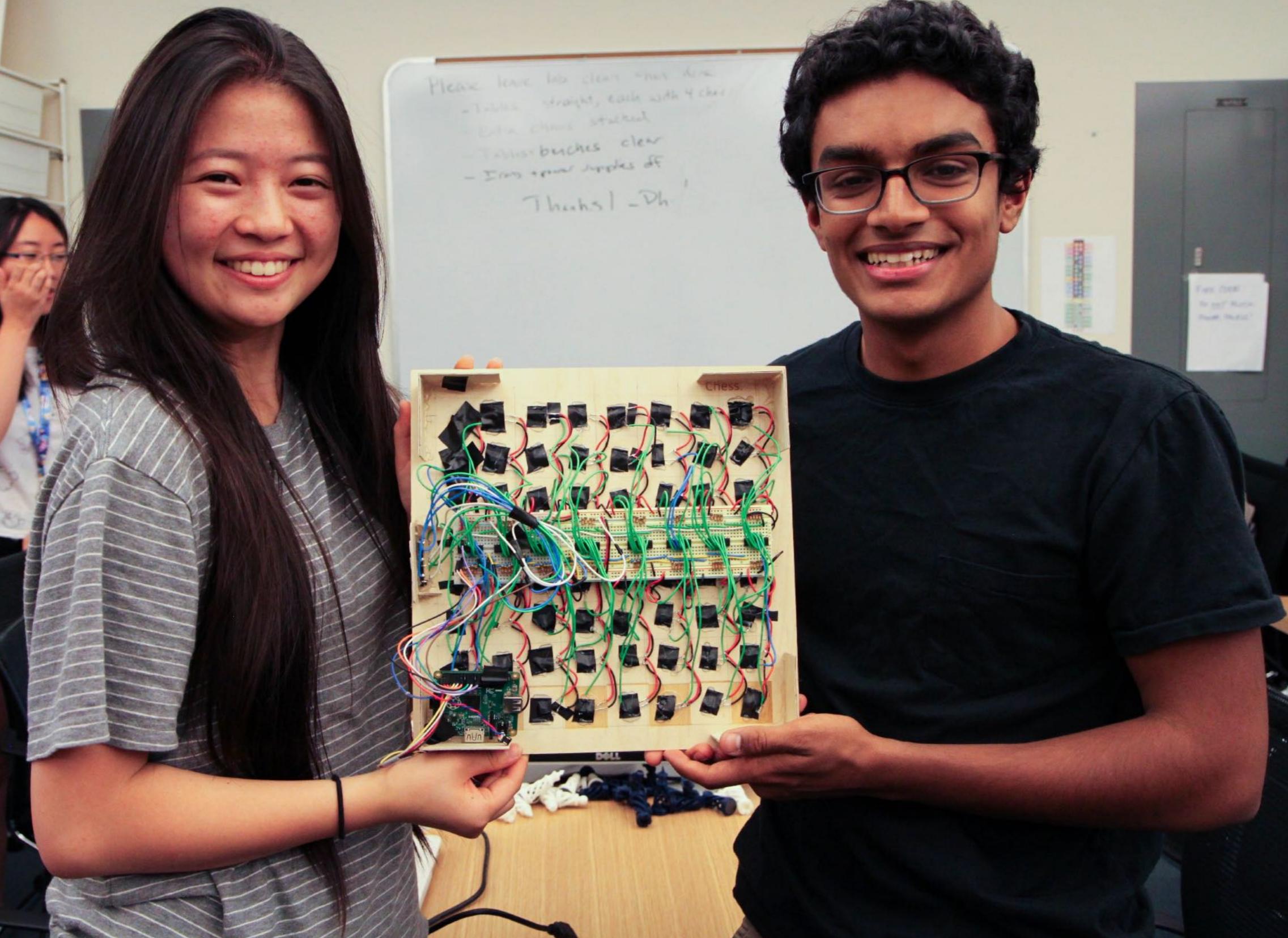
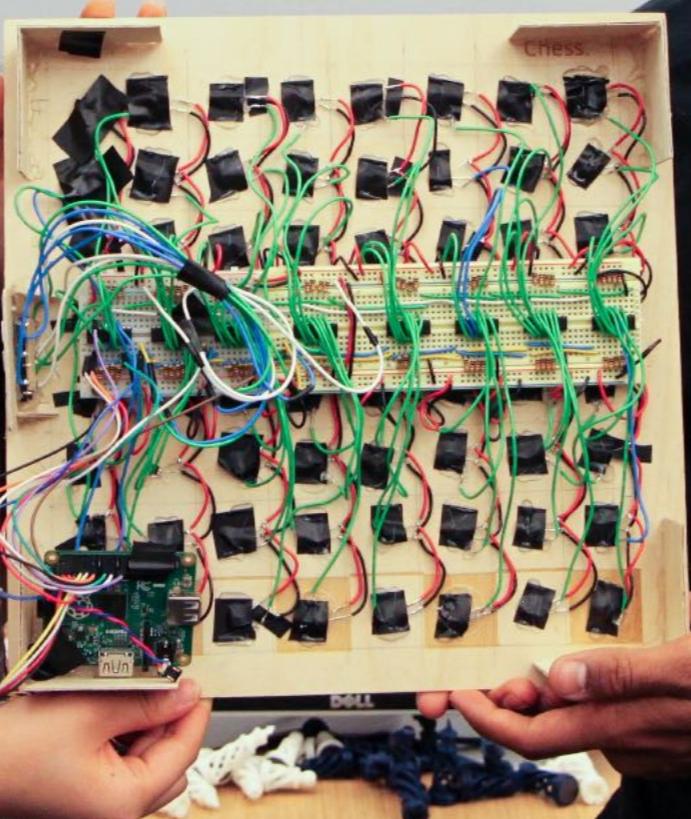


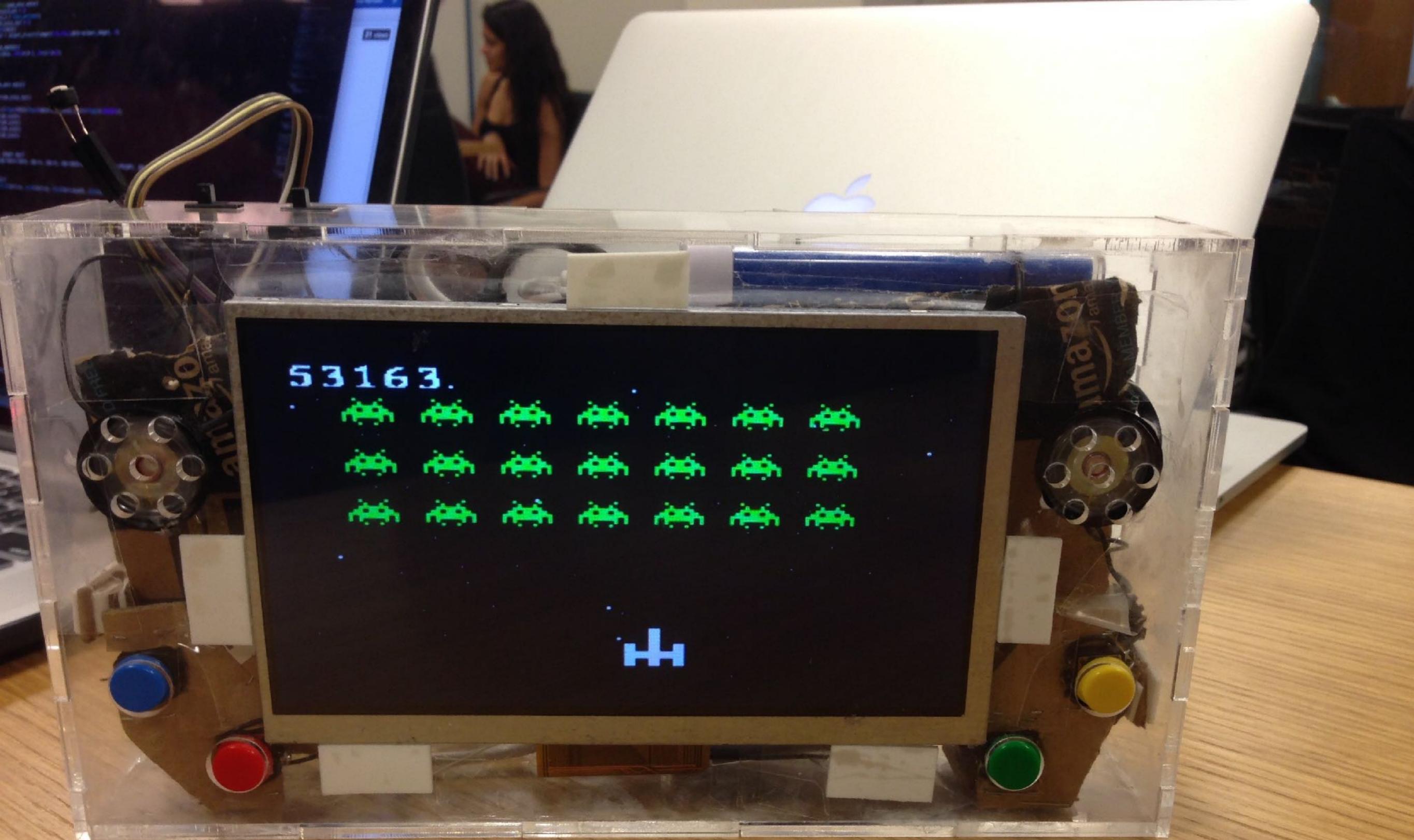




Please leave tea clean - no tea
- Tables straight, each with 4 chairs
- Extra chairs stacked
- Paper bushes clear
- Dirs equal widths of

Thanks! - Dh





First Week

Today

Fill out course application if you haven't already:

<https://web.stanford.edu/class/cs107e/>

We will decide on enrollment (capped at 40) by Tuesday, September 27th (tomorrow). You will receive an email about enrollment by then.

Assignment 0

- Join forum [https://edstem.org/us/courses/29339/
discussion/](https://edstem.org/us/courses/29339/discussion/)
- Read and understand our guides on basic topics:
electricity, numbers, and UNIX
- Create github account and send us your GitHub id
- Install/setup your development environment
- Fill out lab time poll (will be posted on EdStem)

Class Meeting Protocol I

Class attendance is expected. The slides are not intended to be a replacement for being in class. There are a lot of conceptually difficult topics, and having you there to ask us to stop and clarify is important.

In our experience teaching this class, students who stop regularly attending lectures struggle greatly and spend the last two weeks trying to earn more points back by fixing bugs and resubmitting.

Class Meeting Protocol 2

I will likely screencast the lectures, and they will *only* be available on a case-by-case basis for students who have a legitimate reason for missing class. Legitimate reasons include:

- COVID isolation
- Away sporting / academic event

Illegitimate reasons include:

- Don't want to attend lecture / too tired
- Leaving for returning from a weekend trip that isn't sports/academic

Dealing with COVID

Stanford requires masks are worn inside at all times unless you are speaking in a class. We will be masked in lab.

- Please wear them well and carefully! Even if you are alone in lab.

If you feel sick or have been exposed, get tested.

What is a computer?

You've taking CSI06A / 106B, and possibly some hardware-esque classes (e.g., ENGR 40M, EE 101A/B, EE108, etc.), but you probably haven't taken a course where you discussed what a computer really is.

Almost all computers you use today (including the Raspberry Pi, your phone, and your laptop/desktop, but not FPGAs) are built using the *von Neumann architecture*, which was laid out by John von Neumann et al. in a 1945 paper on the EDVAC computer.

The von Neumann architecture

The original architecture designed by von Newmann has the following features (see Wikipedia)

- A processing unit that contains an arithmetic logic unit and processor registers
- A control unit that contains an instruction register and program counter
- Memory that stores data and instructions
- External mass storage
- Input and output mechanisms

A critical part is the idea that instructions and data are both kept in the same memory

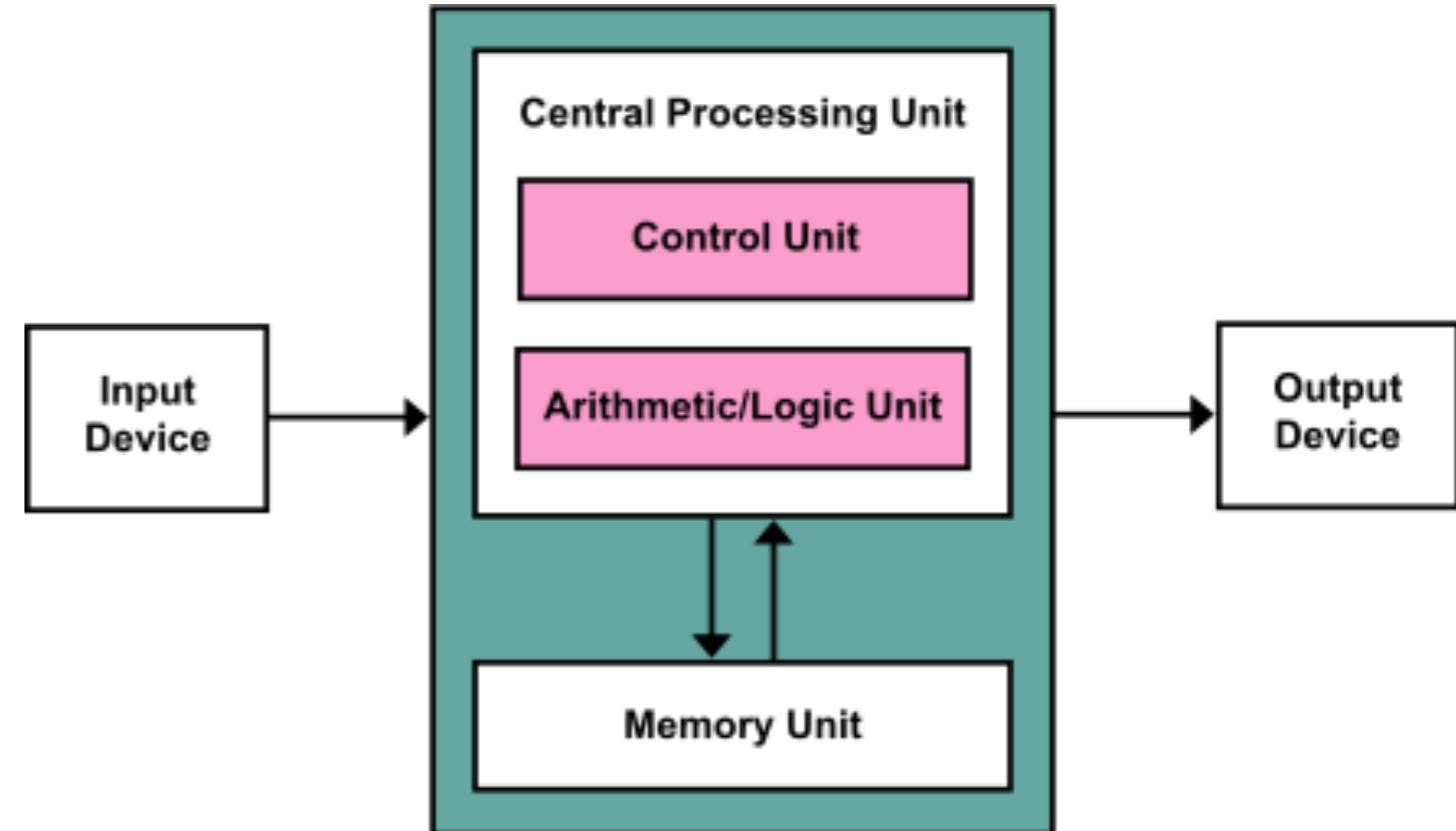
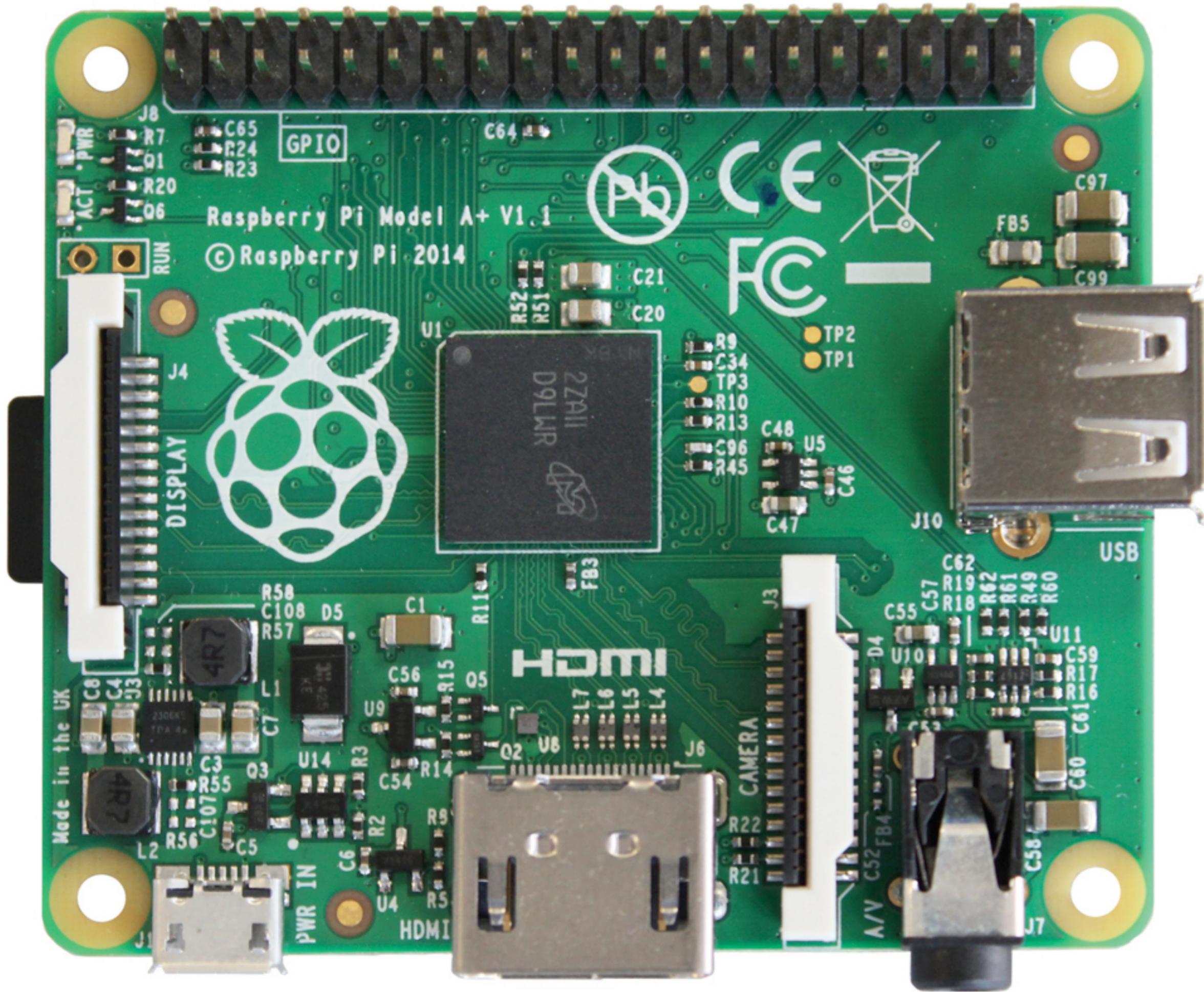


Image: https://en.wikipedia.org/wiki/Von_Neumann_architecture#/media/File:Von_Neumann_Architecture.svg

How is our Raspberry Pi a von Newmann architecture machine?

- A processing unit that contains an arithmetic logic unit and processor registers
 - The Raspberry Pi has an ARM processor with processor registers
- A control unit that contains an instruction register and program counter
 - The ARM processor has an instruction register (kind of — it is not something we can get access to) and a program counter ("r15")
- Memory that stores data and instructions
 - The Raspberry Pi A+ has 512MB of RAM
- External mass storage
 - The Raspberry Pi A+ has an SD card reader
- Input and output mechanisms
 - The Raspberry Pi A+ has USB (we won't use it), HDMI (we will use it), and GPIO pins (you better believe we will use them)



ARMKCE MC1
V-OF3
1439 1-6

PP10
PP13



MICRO SD CARD

090604
JW406AAC

J9

C66

C2 R1 PP8 PP4

PP2

PP7 PP1

PP3

TRST_N
T01 T00 TMS TCK GND

J5

R12 C40 C17
C95 FB2
C94 C36
C51 C49 C69
C18 C14 C37
R25
C50 C9 C13 C12
C35 C30
C45 C29

L3
C23 C28
C26 C42 C39
C19 C63 C67
C41 C16 C31
C44 C17 C33 C22
C27 C38
C10 X1
C11 C15
C24 C19
C18 C17
PP15 PP10
PP14 PP16
PP19 PP17
PP5

PP40 PP39

PP38

PP37

PP30

PP32 PP29 PP34

PP33 PP31

PP22 PP35

PP23 PP27

PP26 PP12

PP25 PP24

PP6 PP40

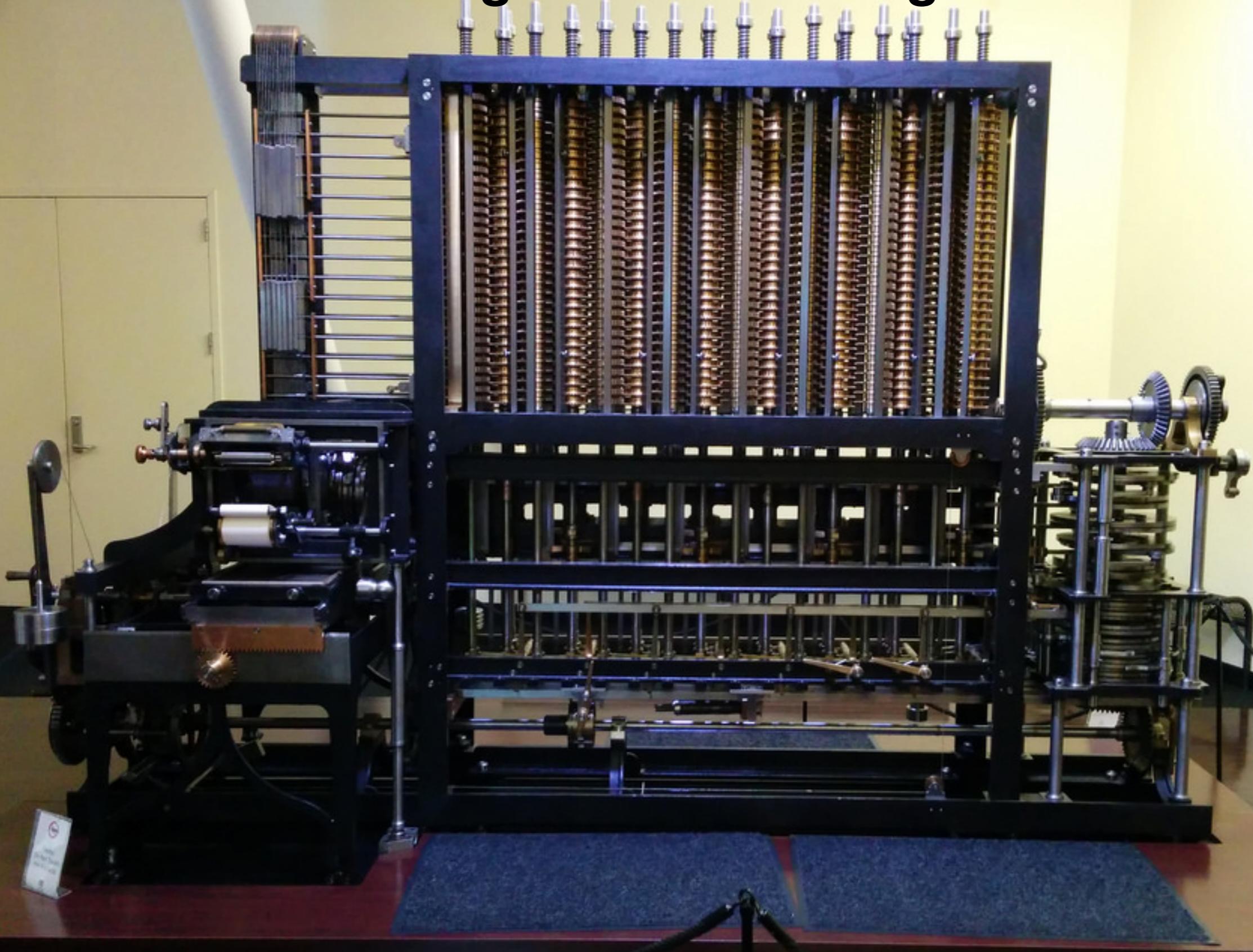
PP38 PP39

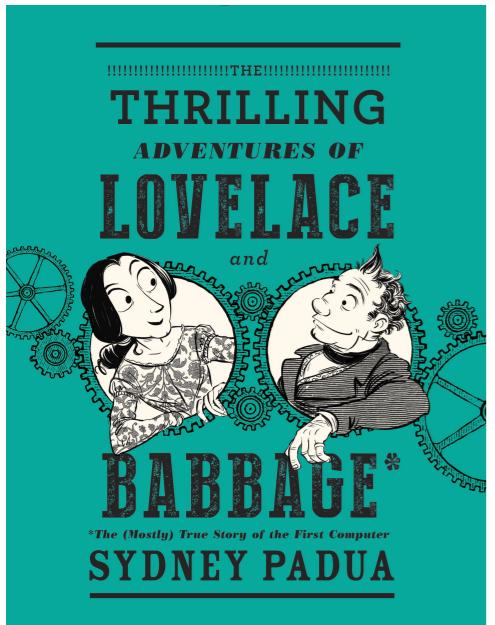
PP37 PP30

PP32 PP29 PP34

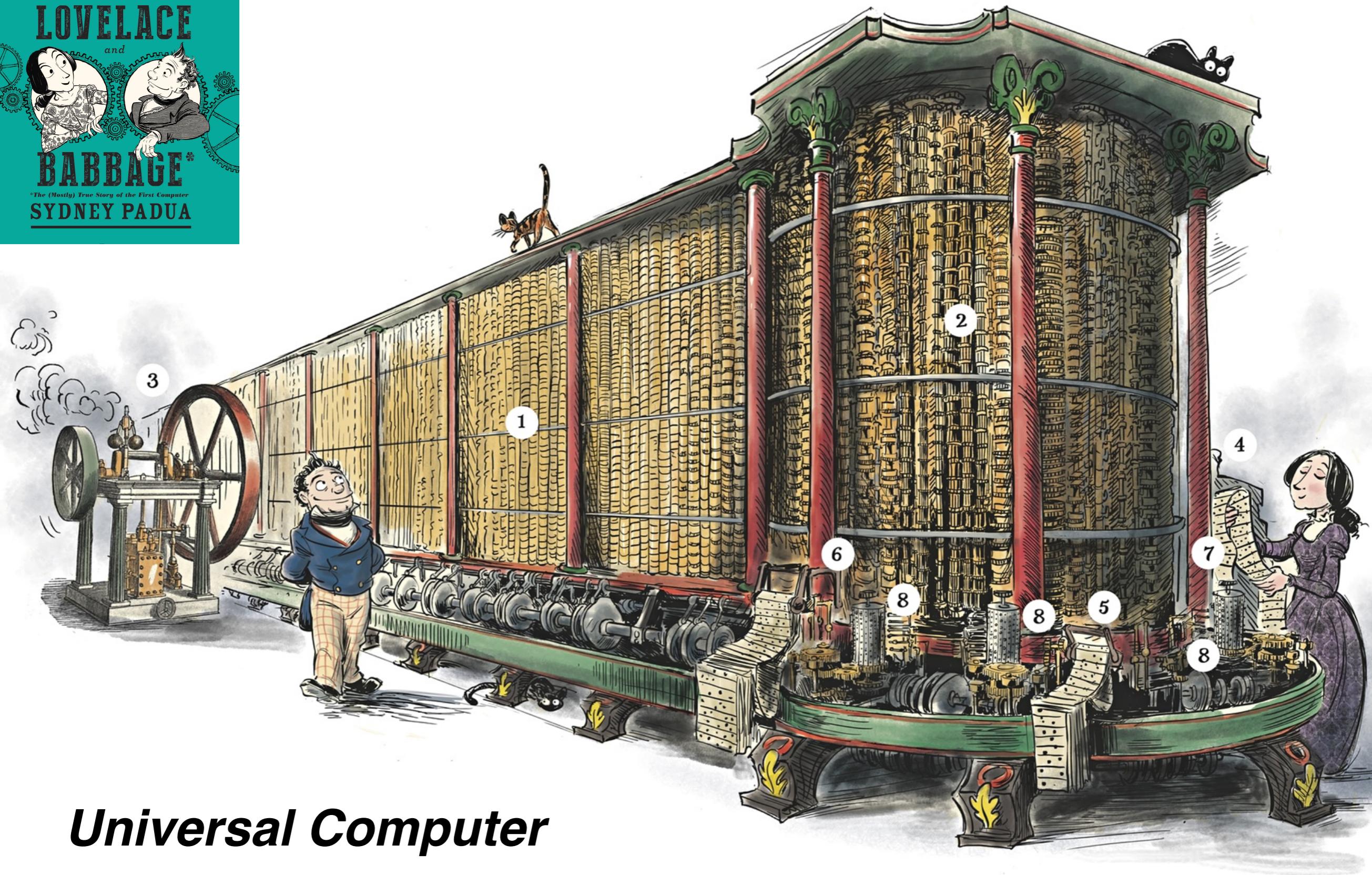
PP33 PP31

Babbage Difference Engine



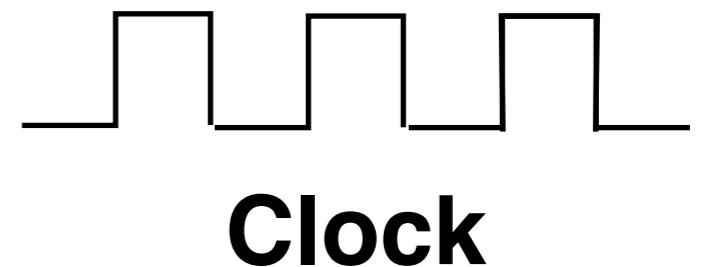
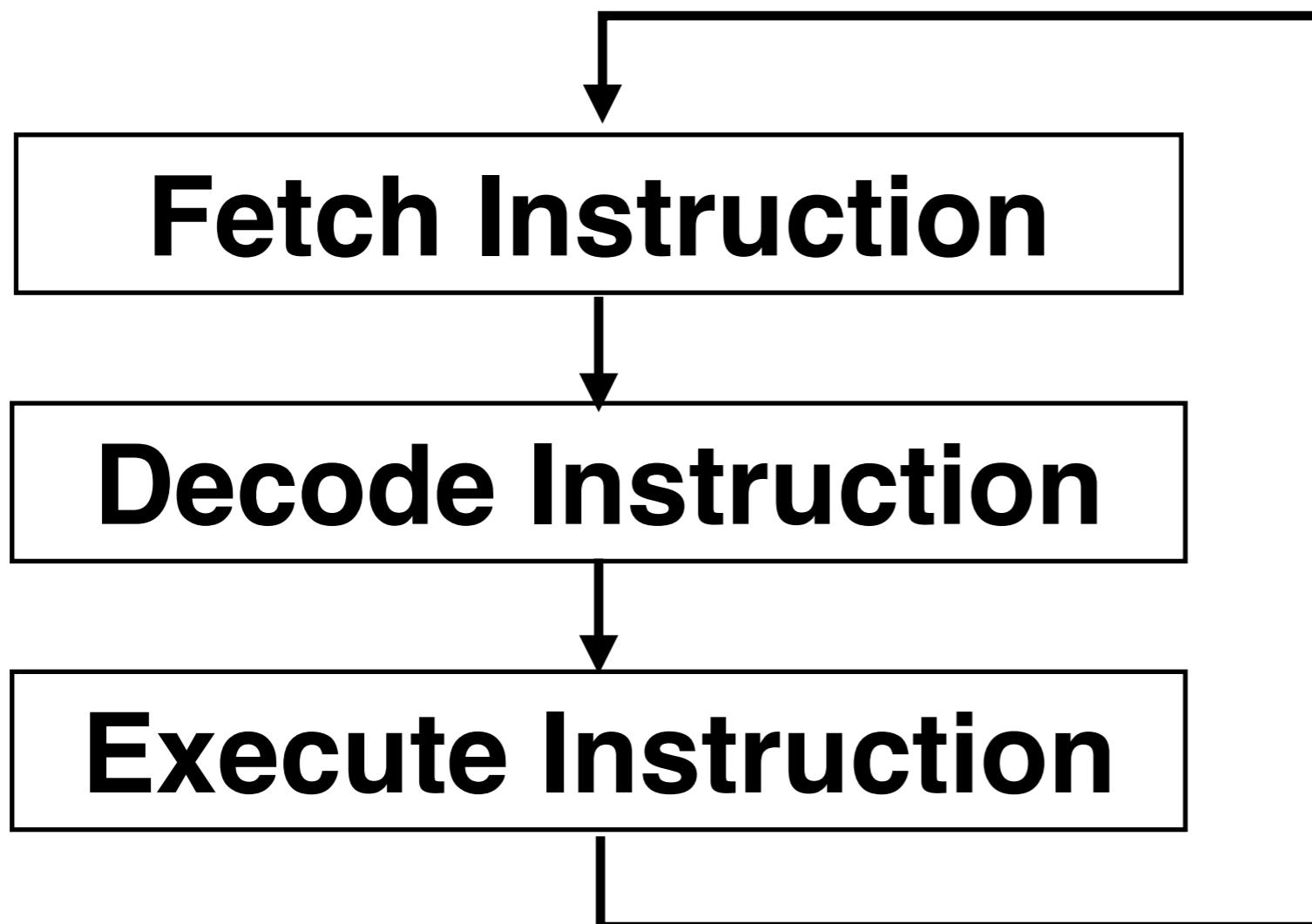


Analytical Engine



Universal Computer

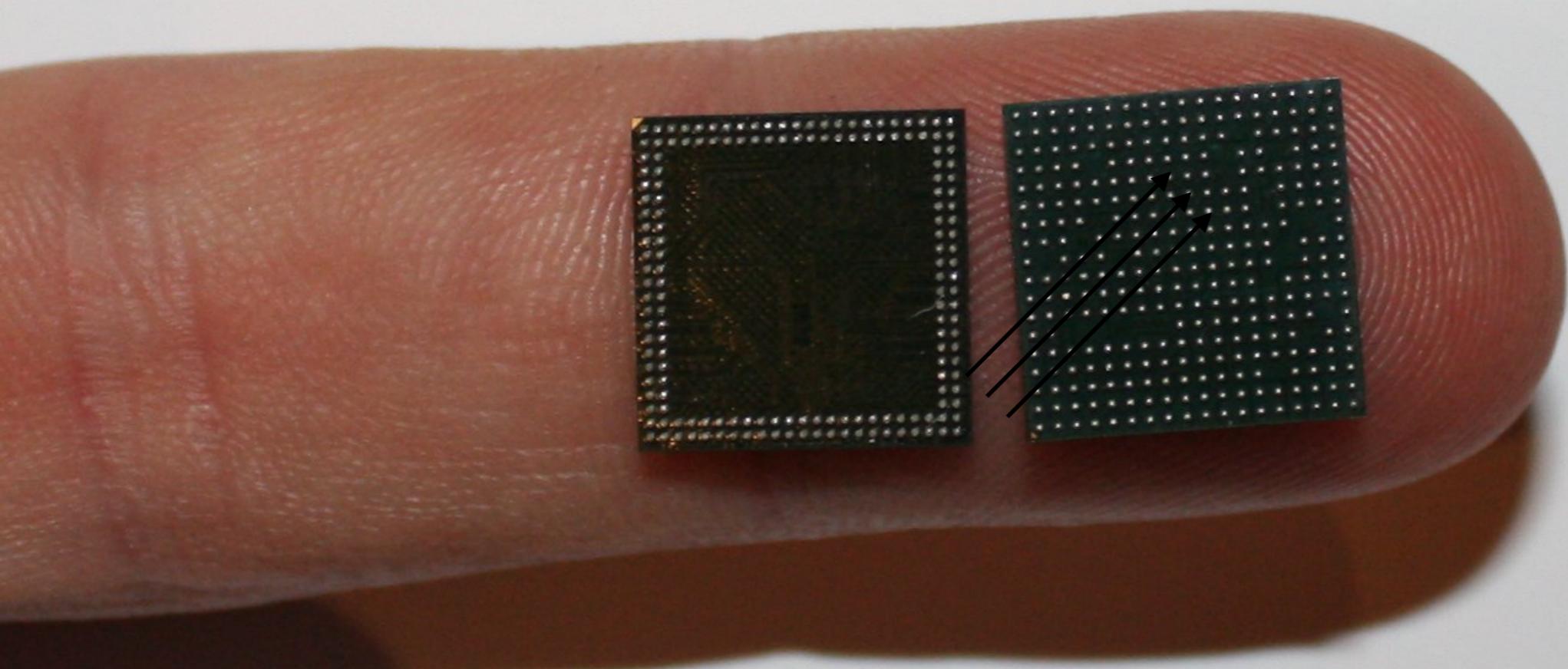
Running a "Program"



Clock

Package on Package

Broadcom 2865 ARM Processor



Samsung 4Gb SDRAM

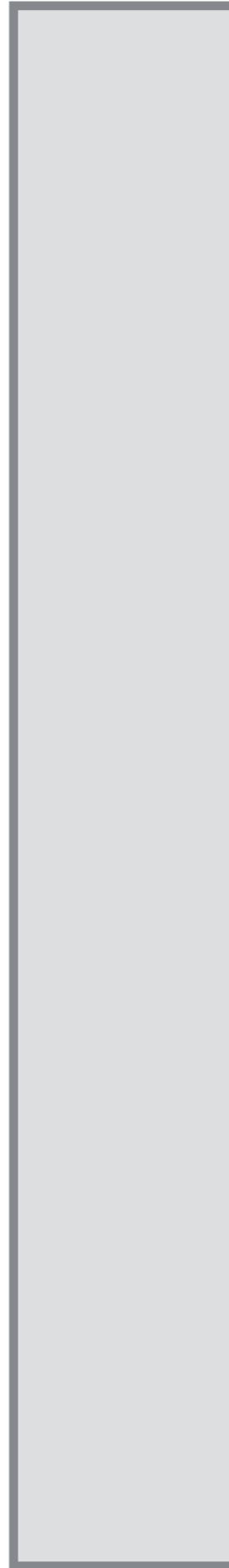
10000000_16

Memory used to store both instructions and data

Storage locations are accessed using 32-bit addresses

Maximum addressable memory is 4 GB

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



Memory Map

00000000_16

10000000₁₆



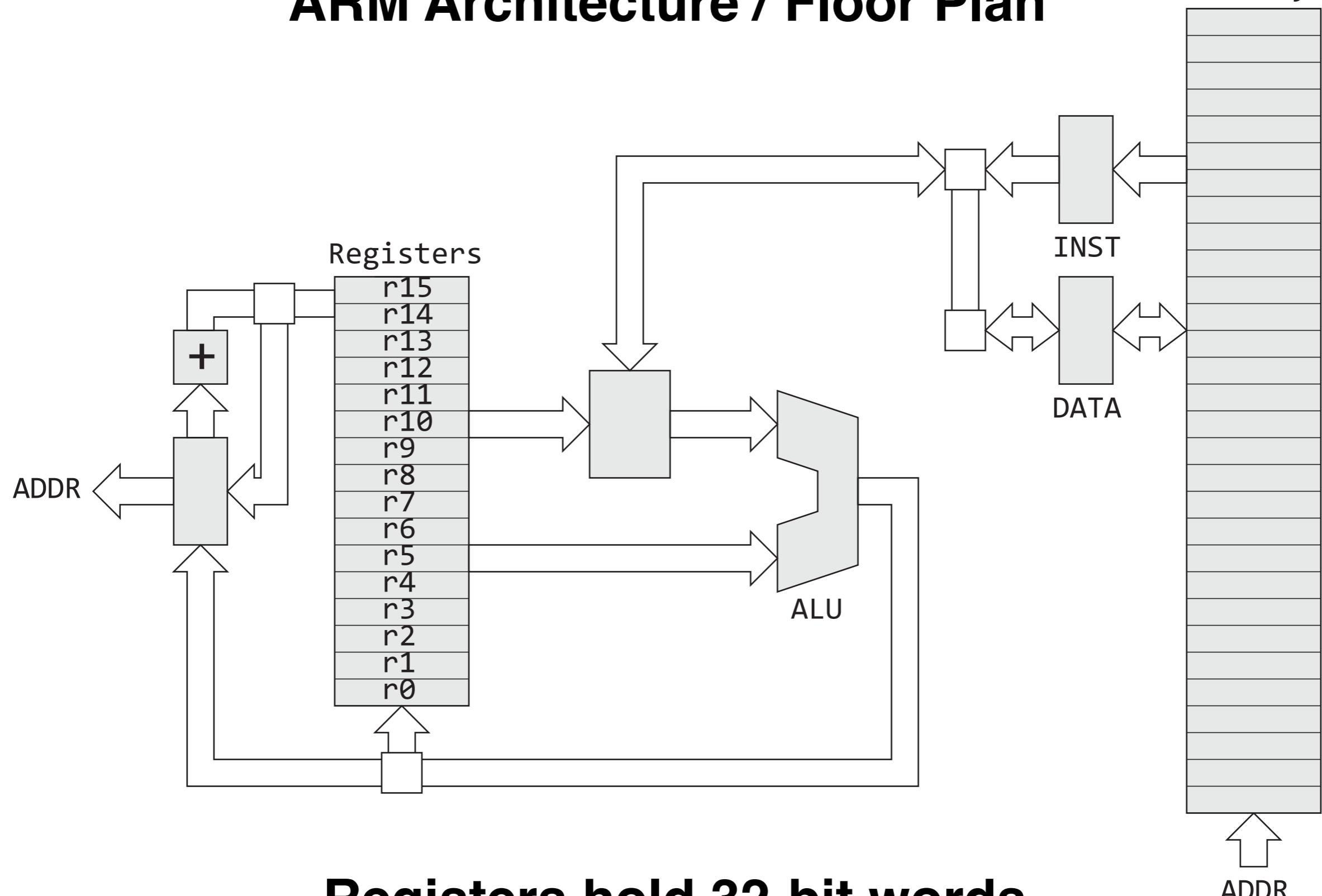
Memory Map

512 MB Actual Memory



02000000₁₆

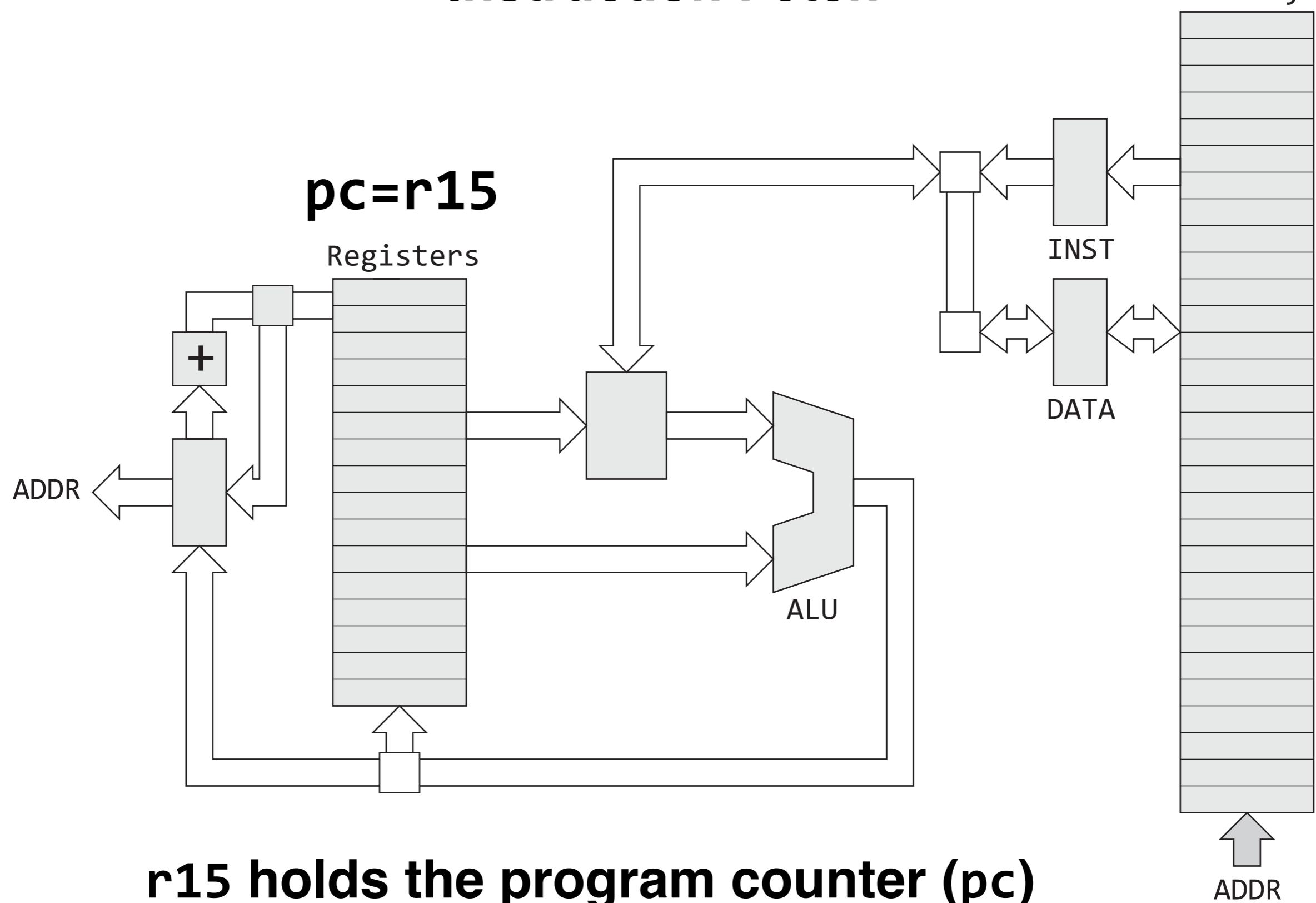
ARM Architecture / Floor Plan



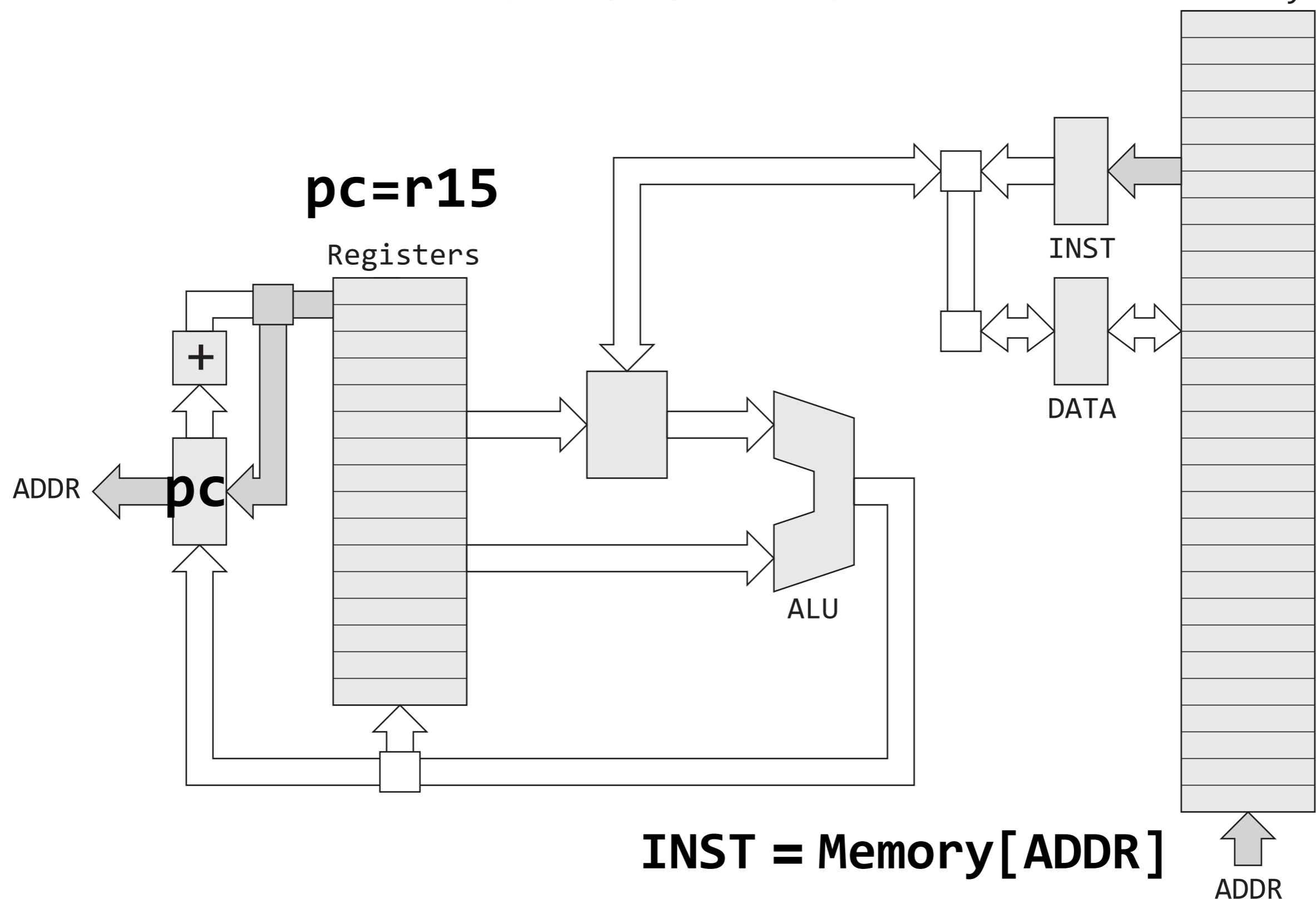
Registers hold 32-bit words

Arithmetic-Logic Unit (ALU) operates on 32-bit words

Instruction Fetch

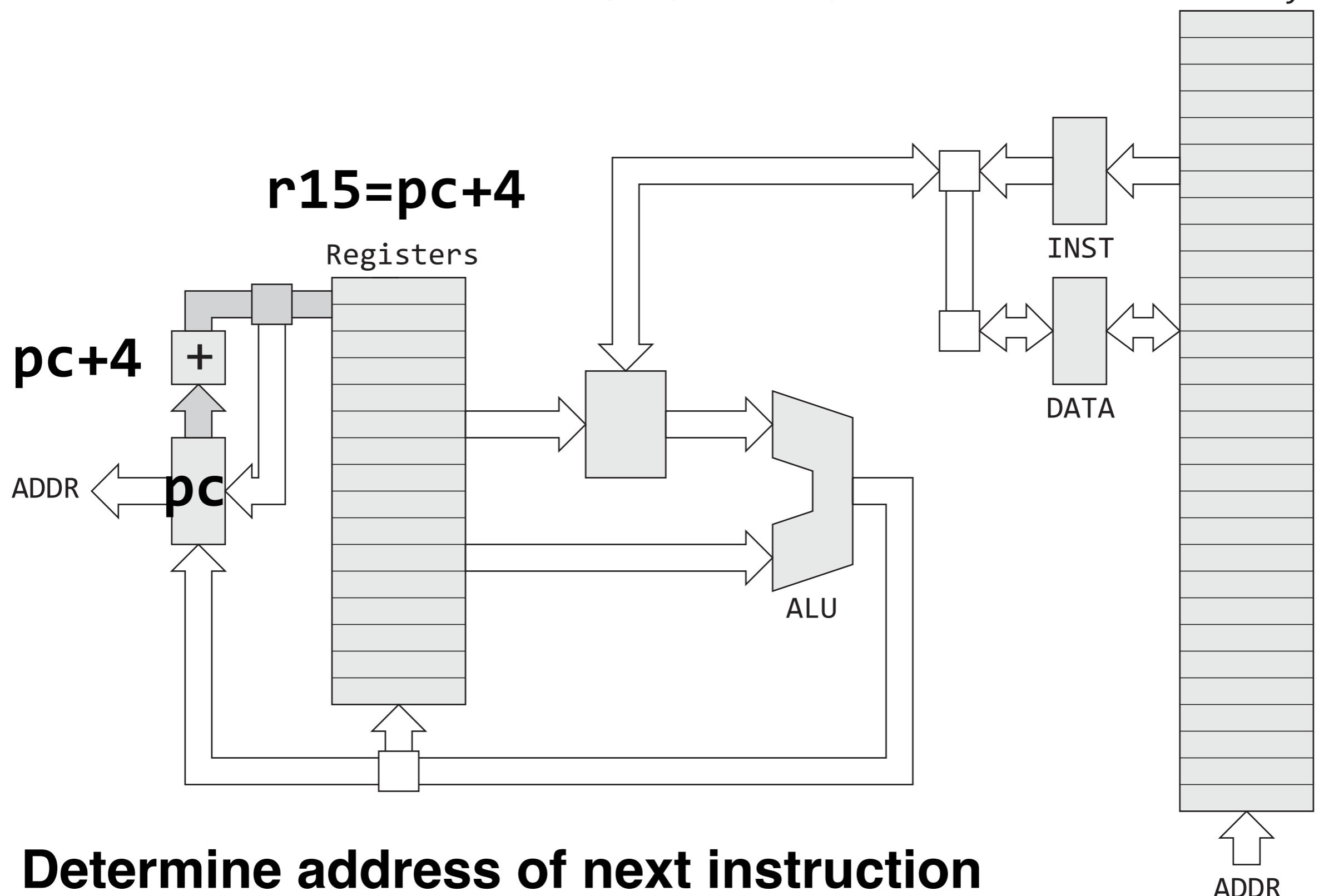


Instruction Fetch



Addresses and instructions are 32-bit words

Instruction Fetch



Determine address of next instruction

Why **pc+4**?