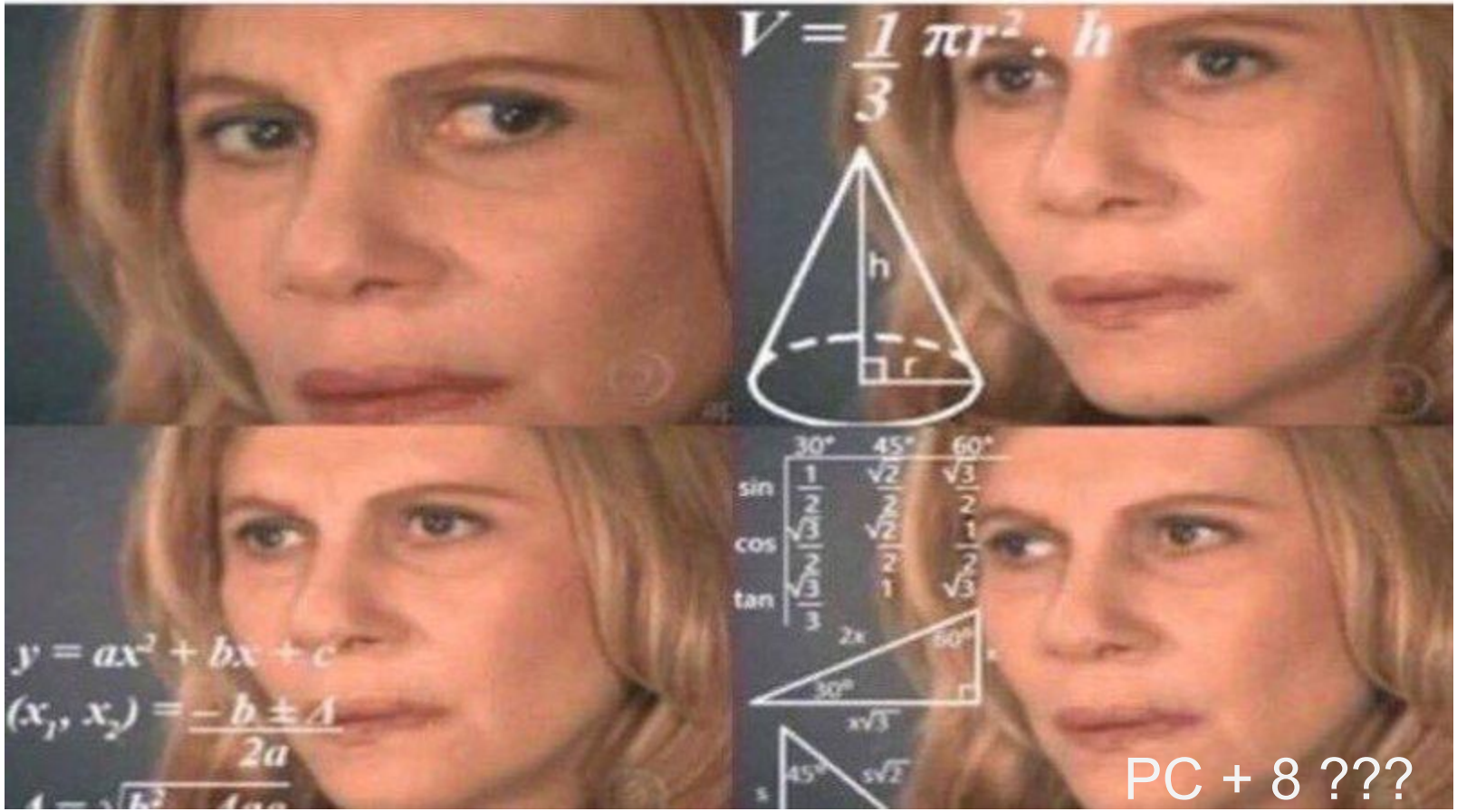
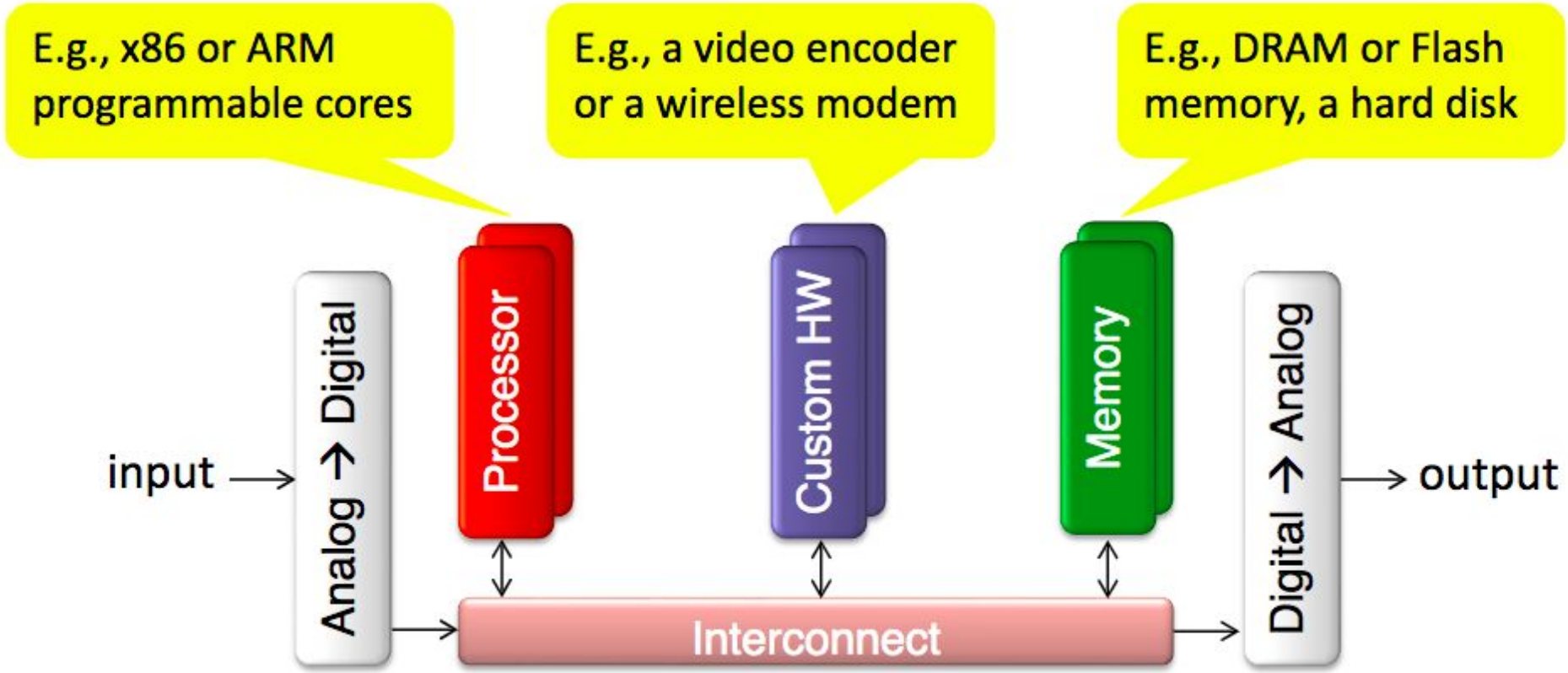
Computer Architecture

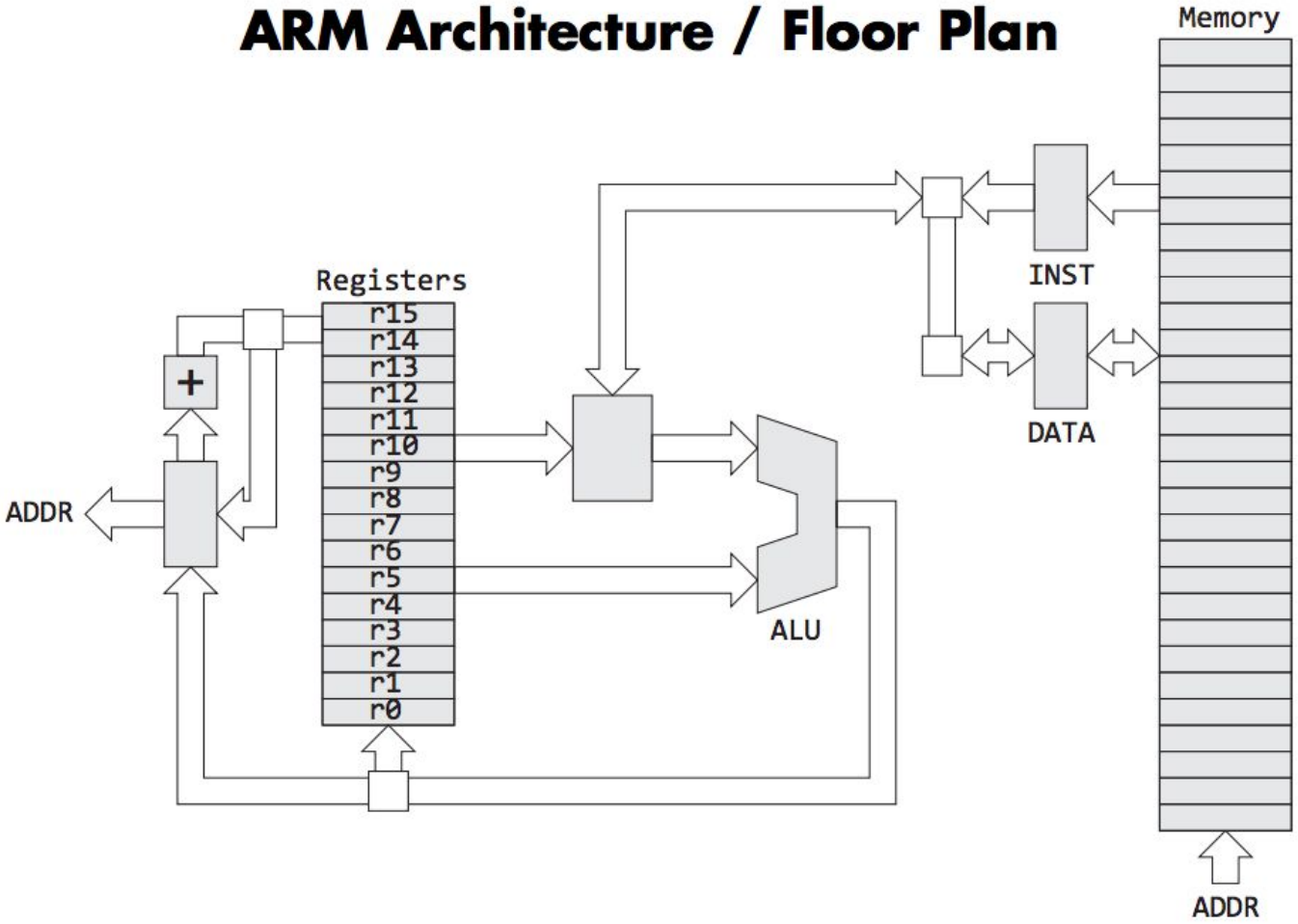
Sean Konz

Some slides and graphics adapted from George Michelogiannakis (EE180)


$V = \frac{1}{3}\pi r^2 . h$
h
r
$y = ax^2 + bx + c$
$(x_1, x_2) = \frac{-b \pm \Delta}{2a}$
30° 45° 60°
sin $\frac{1}{2}$ $\frac{\sqrt{2}}{2}$ $\frac{\sqrt{3}}{2}$
cos $\frac{\sqrt{3}}{2}$ $\frac{\sqrt{2}}{2}$ $\frac{1}{2}$
tan $\frac{\sqrt{3}}{3}$ 1 $\sqrt{3}$
2x
60°
30°
$x\sqrt{3}$
45°
$s\sqrt{2}$
s
PC + 8 ???


E.g., x86 or ARM programmable cores
E.g., a video encoder or a wireless modem
E.g., DRAM or Flash memory, a hard disk
input
Analog → Digital
Processor
Custom HW
Memory
Digital → Analog
output
Interconnect

ARM Architecture / Floor Plan



What is the ARM Architecture anyway?

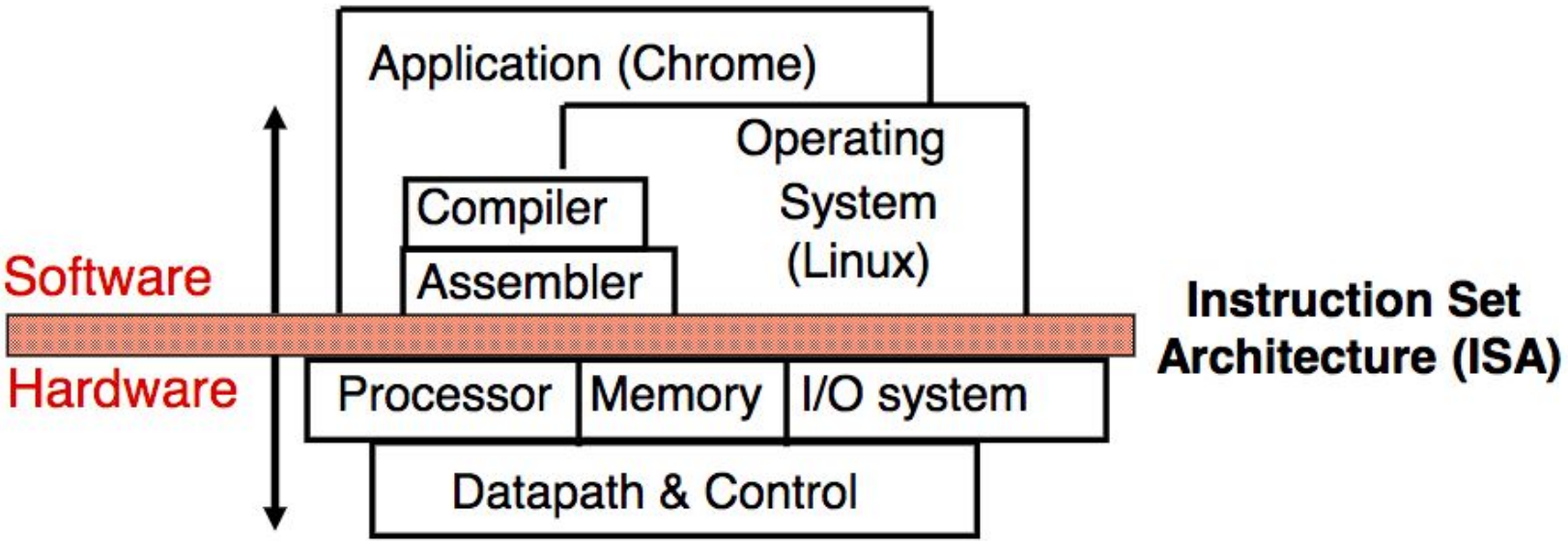
What is the ARM Architecture anyway?

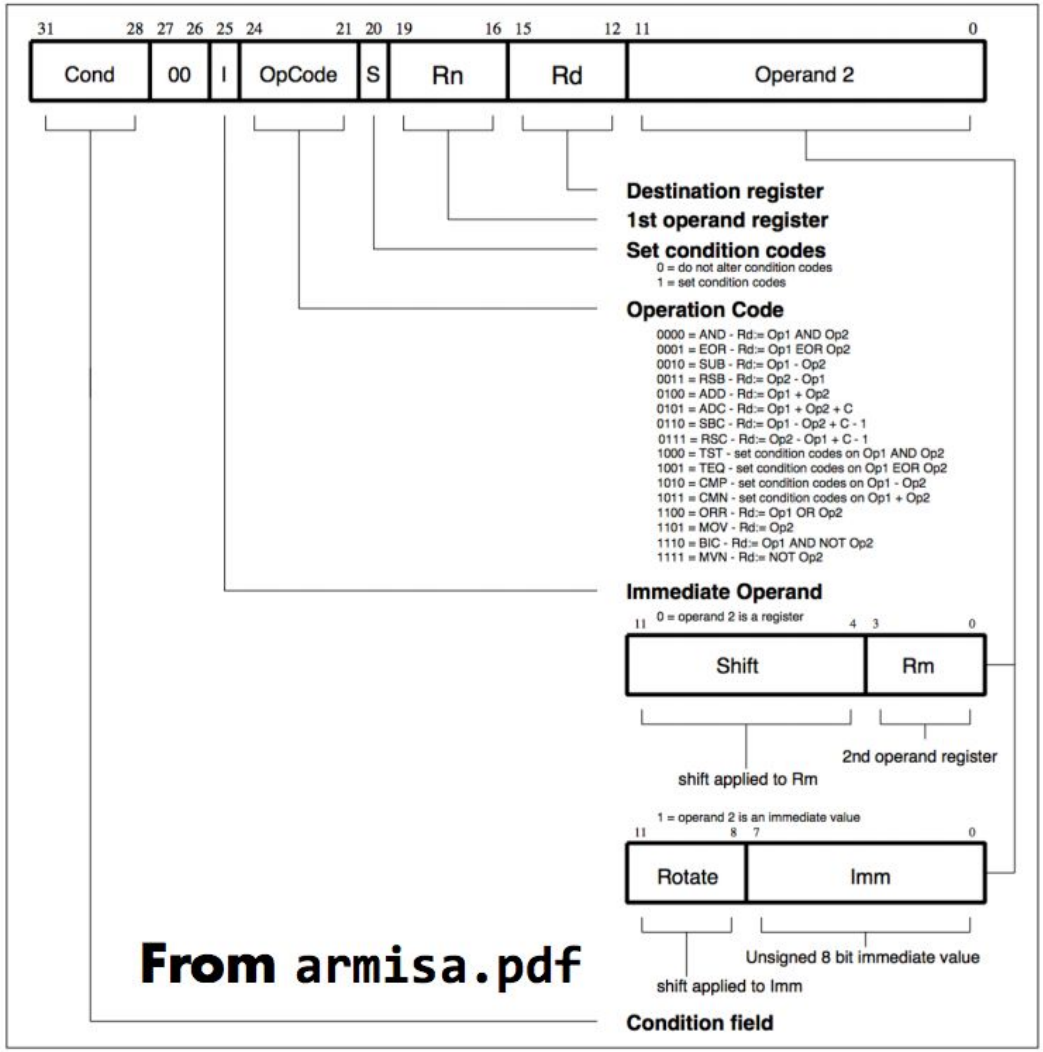
Instruction Set

Contract between the hardware and software

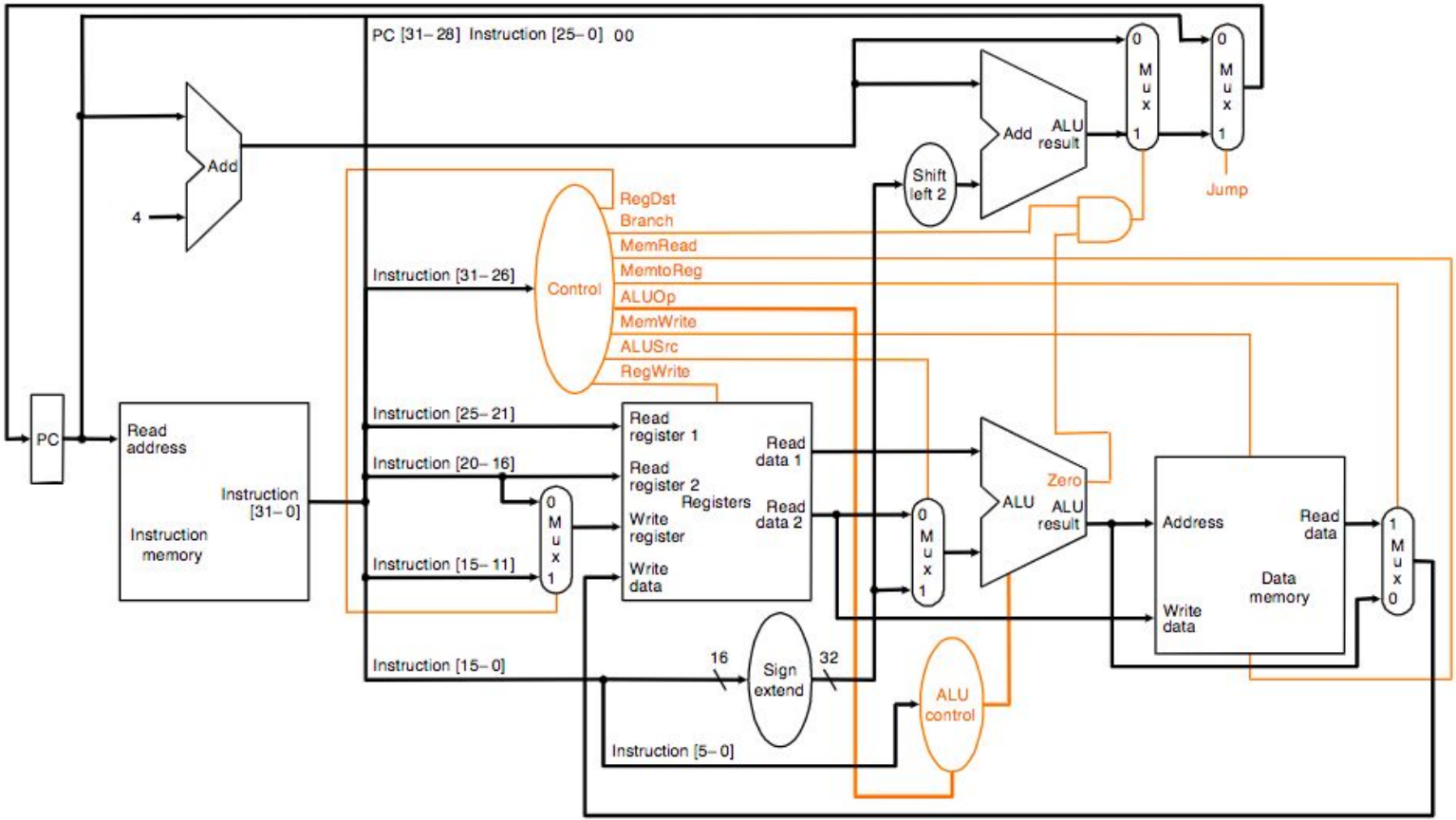
ISA Types

- Intel x86
- Intel IA64
- Intel x86-64
- ARM A32
- ARM A64
- Sun SPARC
- PowerPC
- IBM 390
- MIP32
- MIPS64

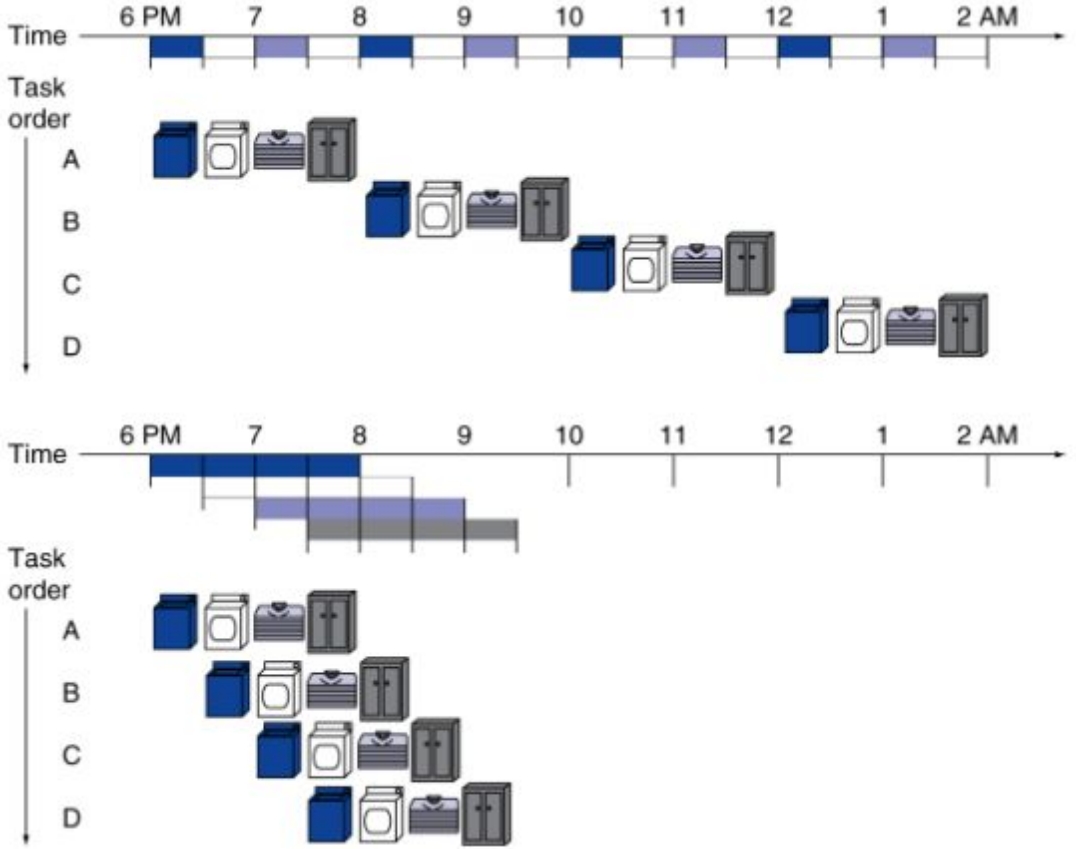

Application (Chrome)
Operating System (Linux)
Compiler
Assembler
Software
Hardware
Processor
Memory
I/O system
Datapath & Control
Instruction Set Architecture (ISA)


31 28 27 26 25 24 21 20 19 16 15 12 11 0
Cond
00
I
OpCode
S
Rn
Rd
Operand 2
Destination register
1st operand register
Set condition codes
0 = do not alter condition codes
1 = set condition codes
Operation Code
0000 = AND - Rd:= Op1 AND Op2
0001 = EOR - Rd:= Op1 EOR Op2
0010 = SUB - Rd:= Op1 - Op2
0011 = RSB - Rd:= Op2 - Op1
0100 = ADD - Rd:= Op1 + Op2
0101 = ADC - Rd:= Op1 + Op2 + C
0110 = SBC - Rd:= Op1 - Op2 + C - 1
0111 = RSC - Rd:= Op2 - Op1 + C - 1
1000 = TST - set condition codes on Op1 AND Op2
1001 = TEQ - set condition codes on Op1 EOR Op2
1010 = CMP - set condition codes on Op1 - Op2
1011 = CMN - set condition codes on Op1 + Op2
1100 = ORR - Rd:= Op1 OR Op2
1101 = MOV - Rd:= Op2
1110 = BIC - Rd:= Op1 AND NOT Op2
1111 = MVN - Rd:= NOT Op2
Immediate Operand
0 = operand 2 is a register
11 4 3 0
Shift
Rm
shift applied to Rm
2nd operand register
1 = operand 2 is an immediate value
11 8 7 0
Rotate
Imm
shift applied to Imm
Unsigned 8 bit immediate value
From armisa.pdf
Condition field

A more complete processor



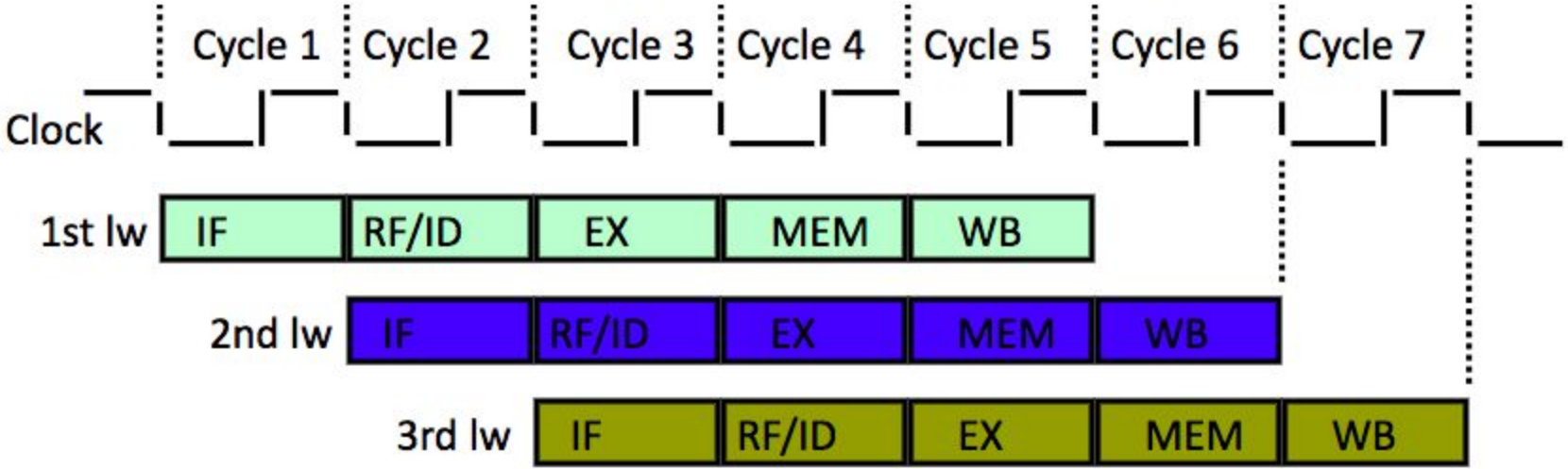
How would you rather do your laundry?



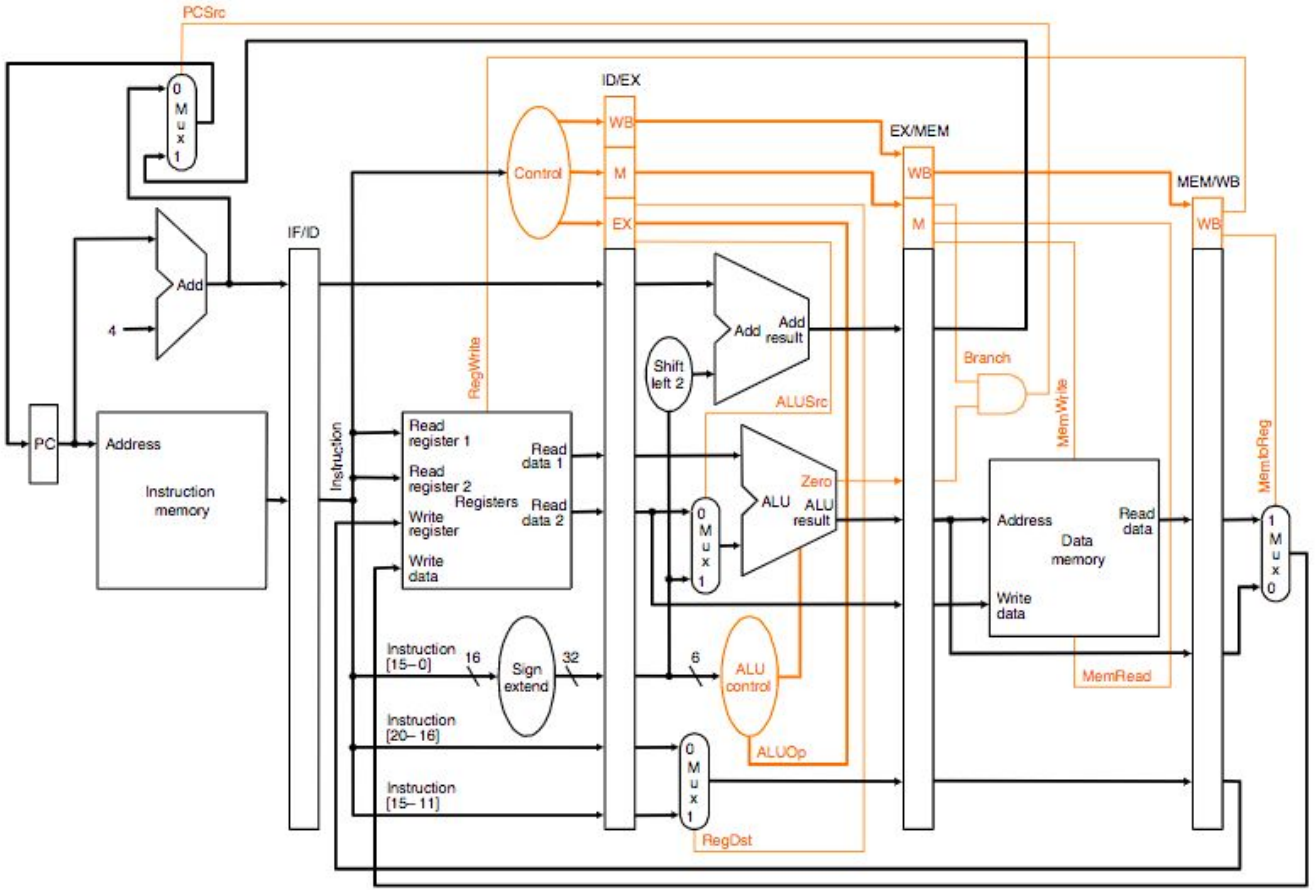
Pipelining to the rescue!

1. Instruction Fetch (IF)
2. Instruction Decode / Register File Access (RF / ID)
3. Execute (EX)
4. Memory Access (MEM)
5. Write Back (WB)

Pipelining to the rescue!



Pipelining to the rescue!



Summary

- Instruction Set Architectures (ISA) define the hardware software contract
- Processors get more complex the more instructions you want to support
- Pipelining is a improves throughput and clock period

For more details take EE 108 and EE180!

System Design

or... building things with many moving parts

Mihir Garimella, CS107E Winter 2020

Why?

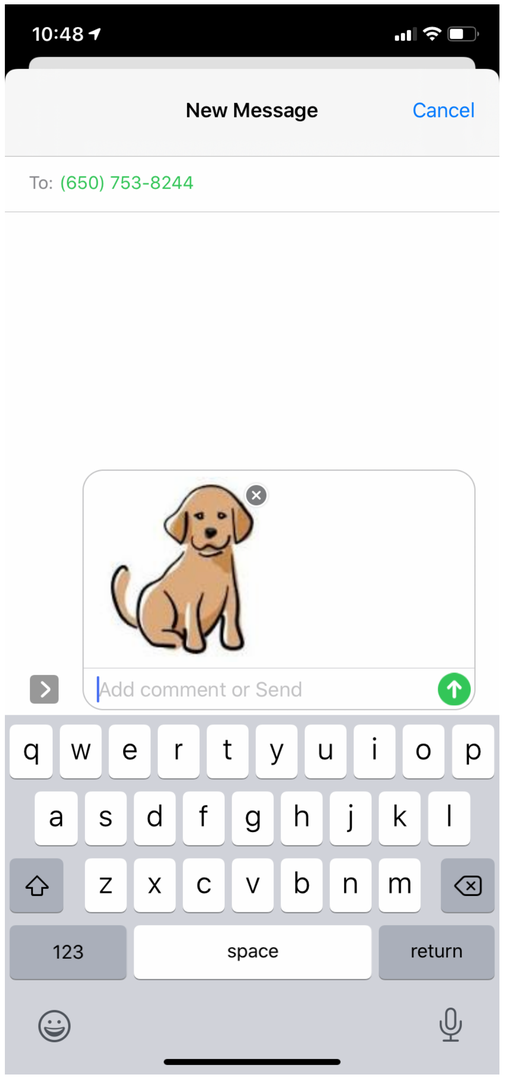
Real-world systems are complex

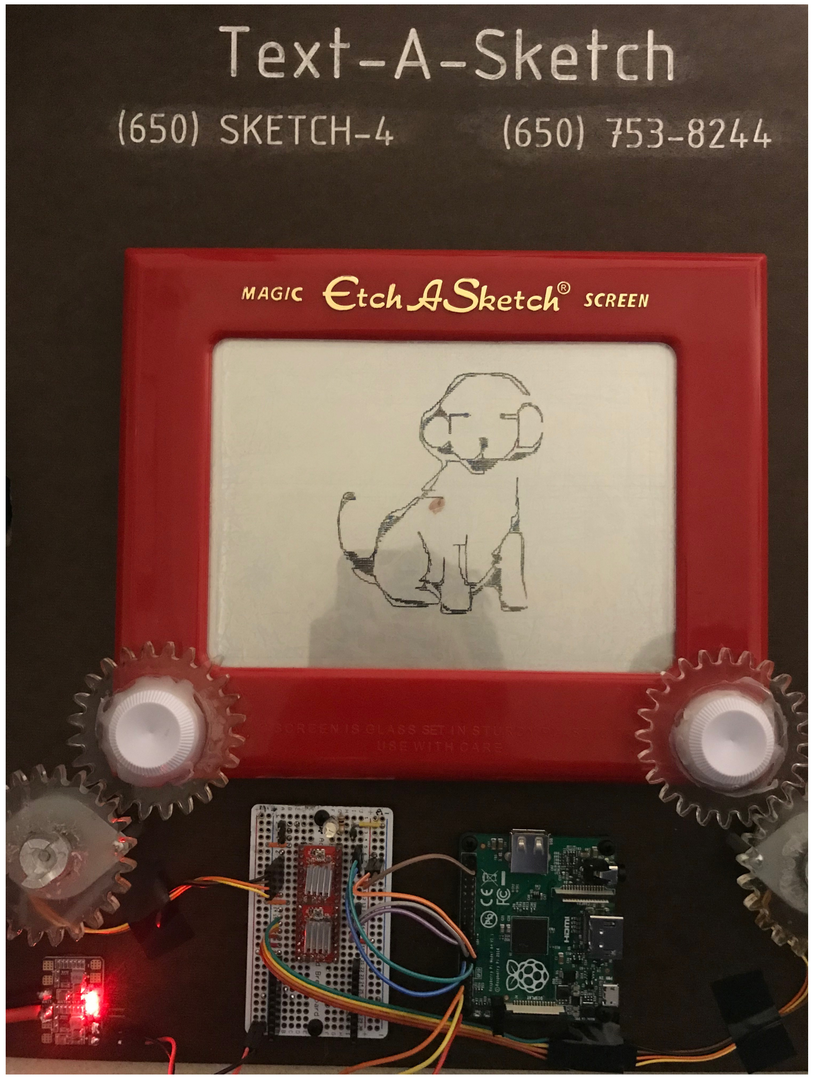
Figuring out what components you need and how to interface between them is hard

Working in a team means you can’t just “figure it out” — need to define components/interfaces beforehand

Hardware is cool, but becomes even more powerful with software behind it

Example: Text-A-Sketch

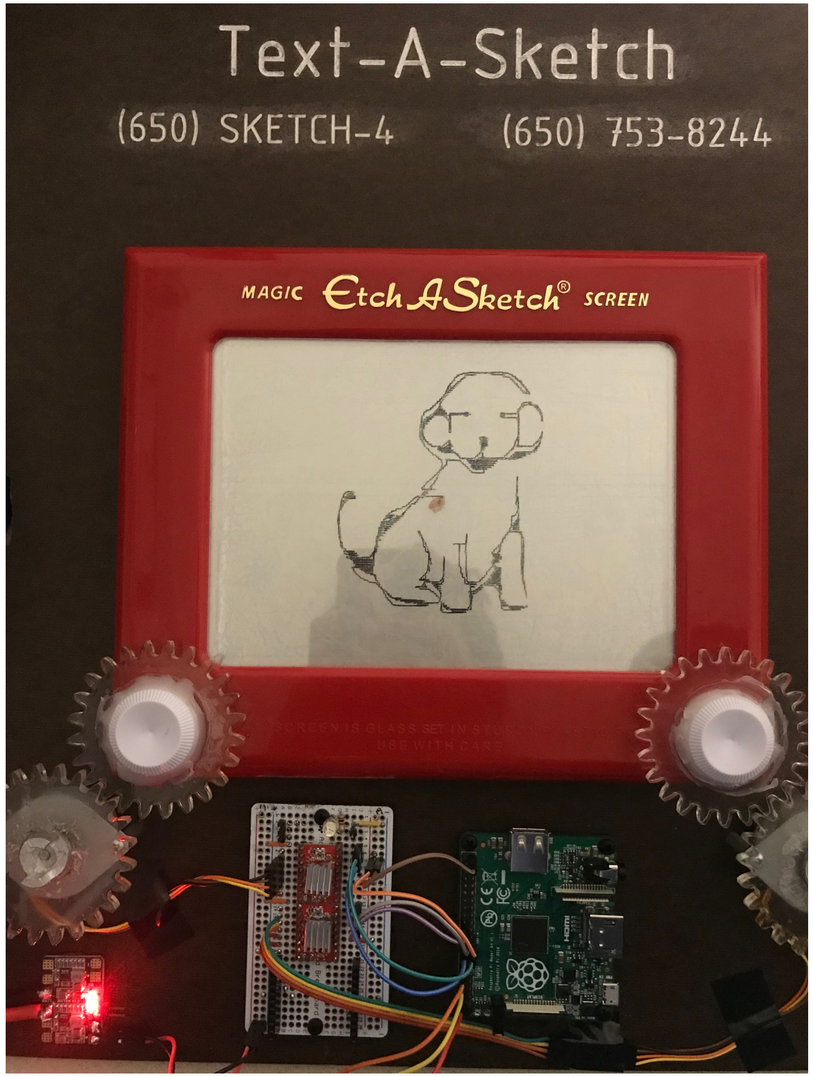




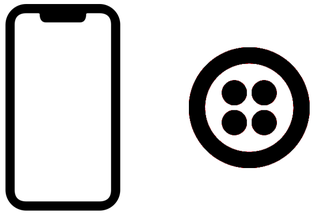
What's the goal?

We receive a picture over MMS...

...and draw it on an Etch-A-Sketch



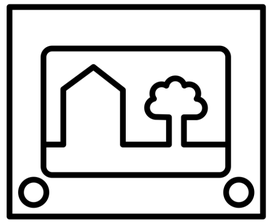
What happens on either end?



Receive a picture

Interface

- **Incoming webhook:** makes HTTP request on incoming message
- **`POST /Messages`:** send outgoing message

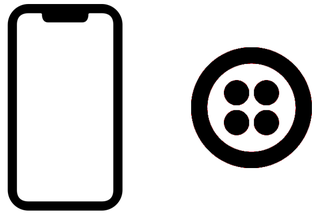


Draw on Etch-A-Sketch

Interface

- **Turn left knob:** move stylus vertically
- **Turn right knob:** move stylus horizontally
- **Shake:** clear

What goes in the middle?

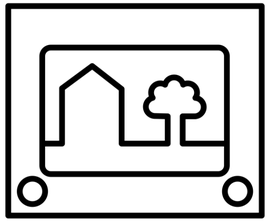


Receive a picture

Handle incoming webhook

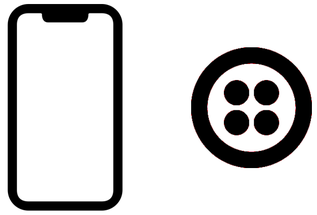
Detect edges

Translate to motor commands

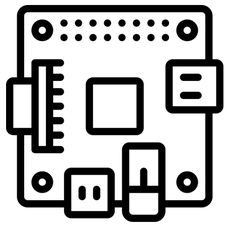


Draw on Etch-A-Sketch

Where do we do this?



Receive a picture

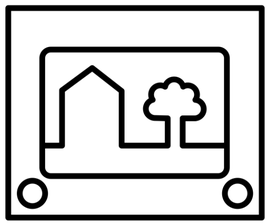


on the Pi!

Handle incoming webhook

Detect edges

Translate to motor commands



Draw on Etch-A-Sketch

Problem:

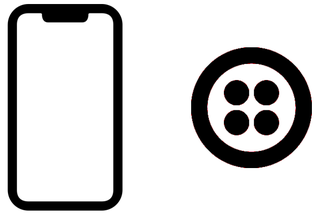
It’s too hard

Writing a Wi-Fi stack for the Pi is really difficult

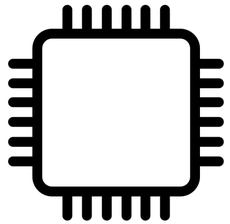
Writing a web server on the Pi is also really difficult

(each of those could be a separate project, or more!)

How do we solve this?

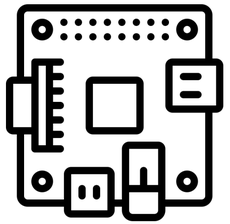


Receive a picture



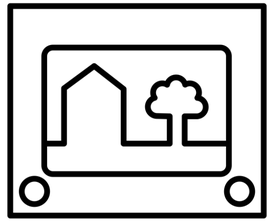
Handle incoming webhook

use a Wi-Fi-enabled microcontroller



Detect edges

Translate to motor commands



Draw on Etch-A-Sketch

Problem:

It’s too slow (and still too hard)

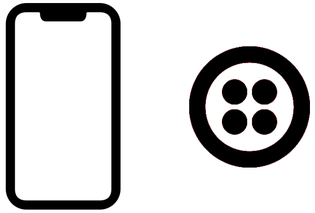
UART max transfer speed = 115,200 bits per second

iPhone image file size = 8.7 megabytes

Transfer time = 604 seconds per image! 🐌

Also, image processing on the Pi is slow + difficult, especially at full-res

How do we solve this?



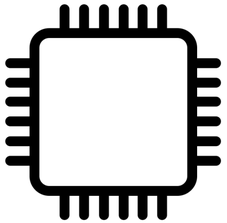
Receive a picture

add a server, do compression!

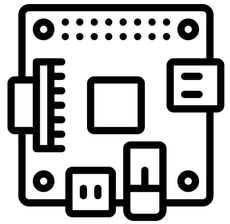
Handle incoming webhook

Detect edges

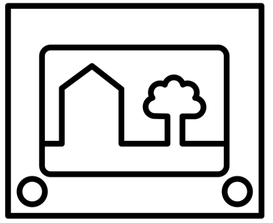
Compress image



Get images



Translate to motor commands



Draw on Etch-A-Sketch

Implementation:

Dealing with Wi-Fi failures

Problem: **Many Wi-Fi networks require a “captive portal” or complex authentication**

Solution: **write a Python script that does the same thing as the Wi-Fi microcontroller, but runs on a laptop**

↑ Benefit of separating *interface* and *implementation*

Implementation:

Making it robust

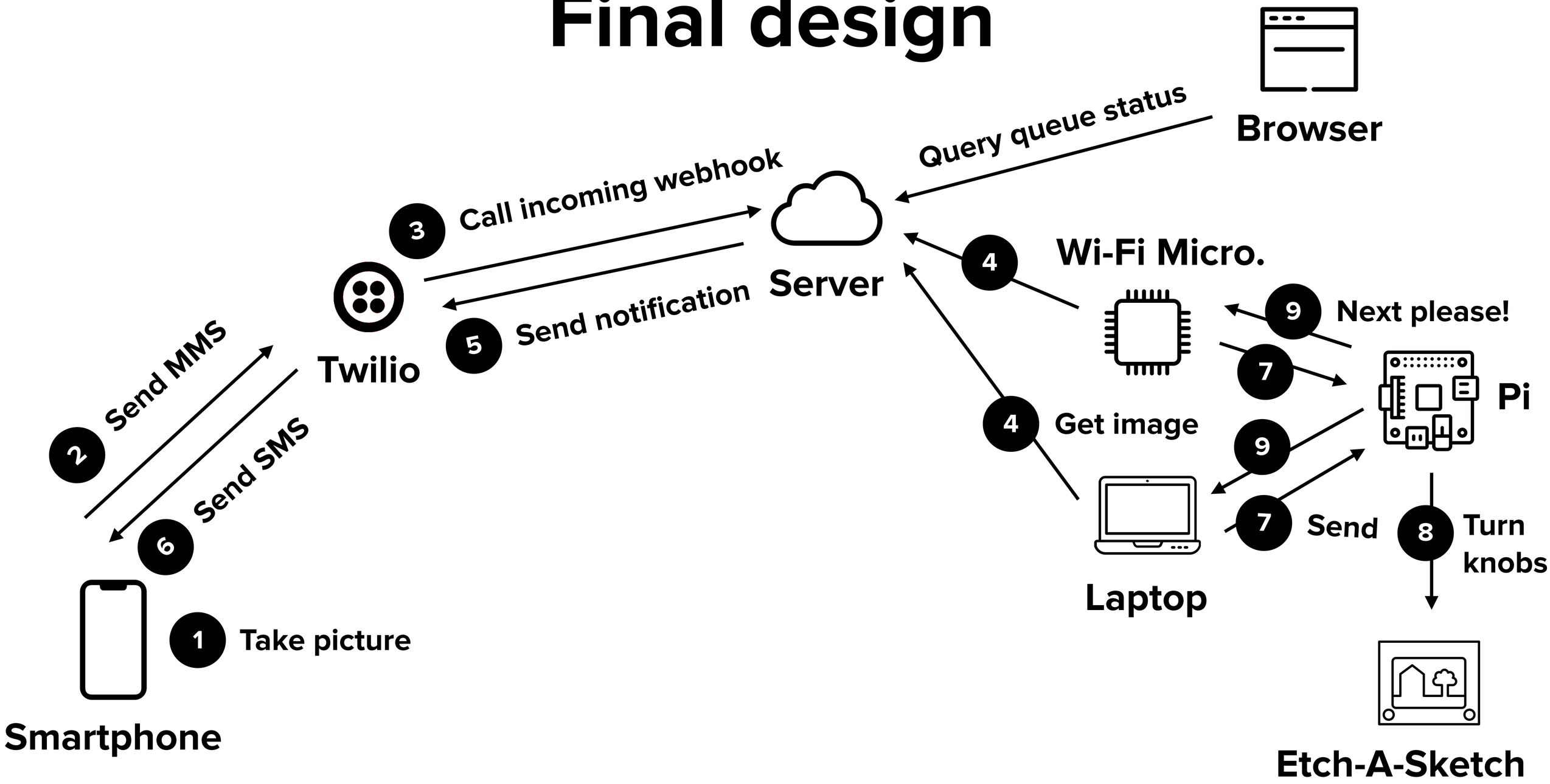
Rate-limiting

Threading

Request queueing

Monitoring

Final design



Talk through projects?

Questions?