07: Memory

Stages of a microprocessor

Fetch - fetch next instruction from memory

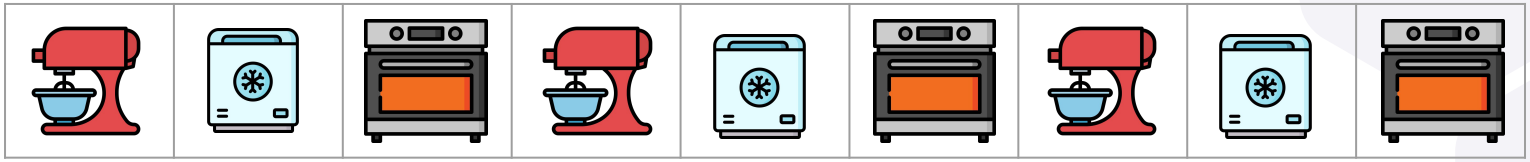
Decode - decode instruction

Execute - perform computation

- ALU (arithmetic logic unit): add, subtract, negate, bit operations
- Shift: used in multiplication/division

Memory access - read and write registers

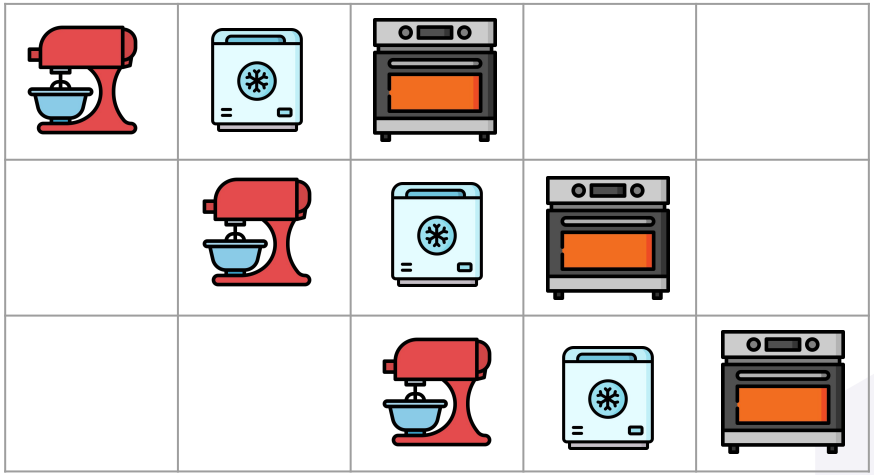
Sequential execution 



images: Flaticon.com

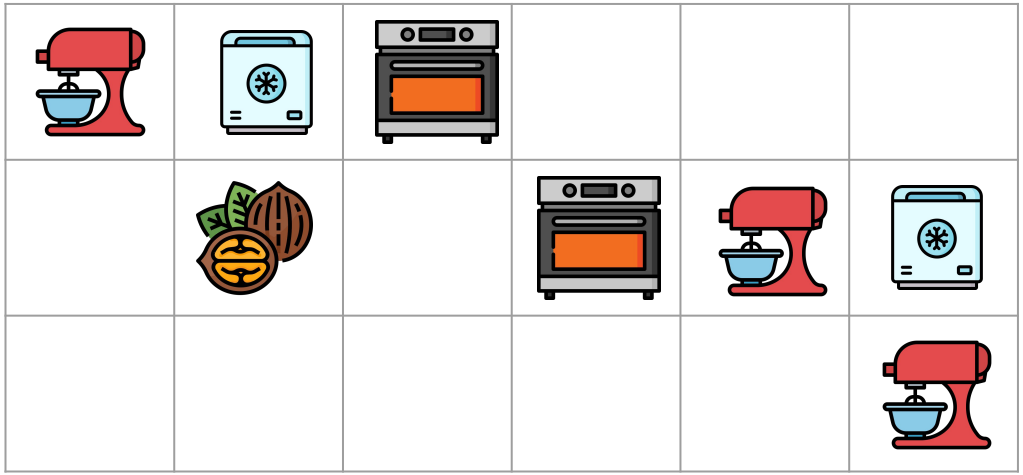
Pipelining 

vs



images: Flaticon.com

Pipeline hazards



images: Flaticon.com

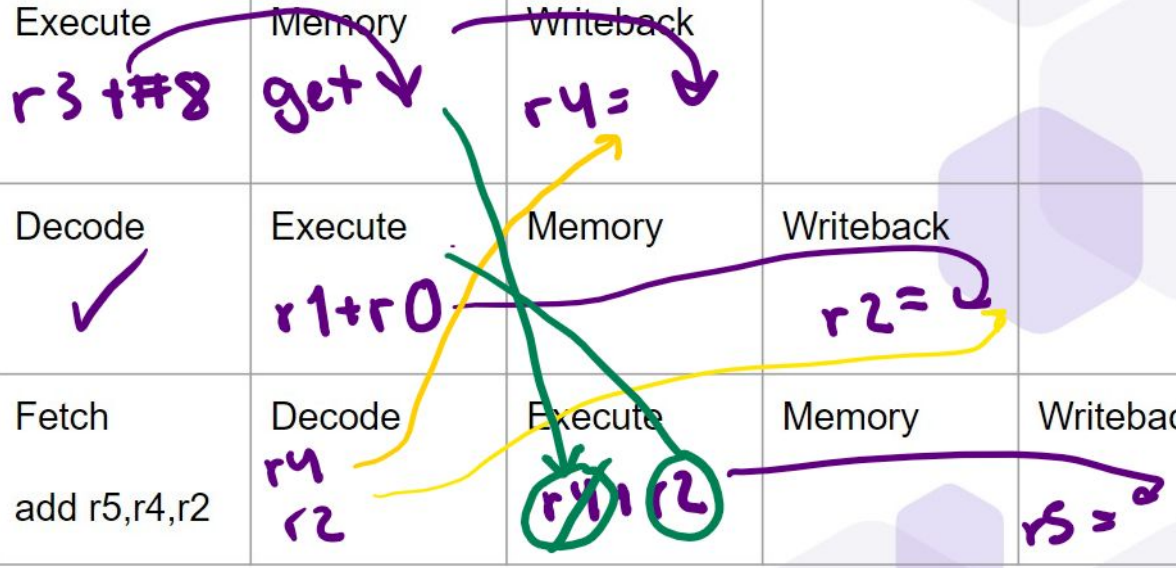
Avoiding (some) hazards using compilation

```
while (i < N) {
  int f_next = f + f_prev;
  f_prev = f;
  f = f_next;
  i += 1;
}
```

```
cmp r4, r2
ble.n   2116 <setup+0x1a>
adds    r0, r1, r3
adds    r2, #1
movs    r1, r3
movs    r3, r0
b.n 2108 <setup+0xc>
```

Pipeline forwarding

Fetch ldr r4 [r3, #8]	Decode	Execute	Memory	Writeback		
	Fetch add r2,r1,r0	Decode	Execute	Memory	Writeback	
		Fetch add r5,r4,r2	Decode	Execute	Memory	Writeback



Can you summarize the tradeoff between deep and shallow pipelines and predict which kind MCUs are more likely to have?

Cortex-M0+

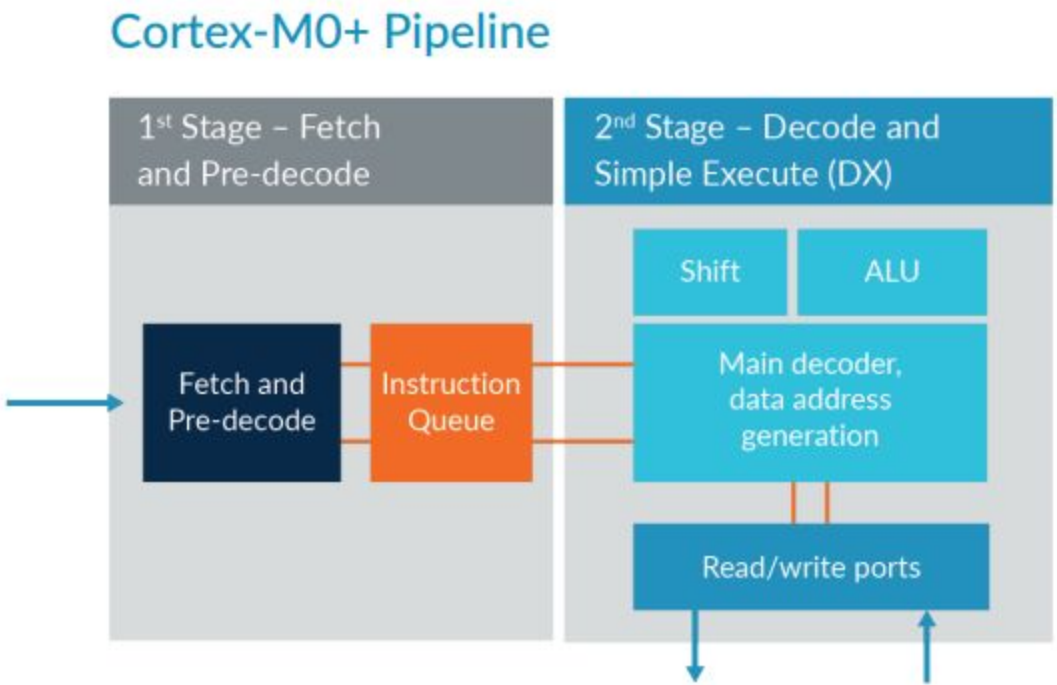


Image source

Memory

Information stored in memory:

- Code
- Program data
- Stack
- Heap (dynamically allocated data)
- Register file

```
Disassembly of section .data:

20000000 <N>:
20000000:       0000000c
```

Types of memory

Volatile - Gets erased when power gets turned off

- RAM (DRAM, SRAM)

Non-volatile - Persists when power gets turned off

- Flash
- ROM (sometimes rewritable, like EEPROM)

Specifications

x-bit processor:

- Data registers, data busses, *words* are that size

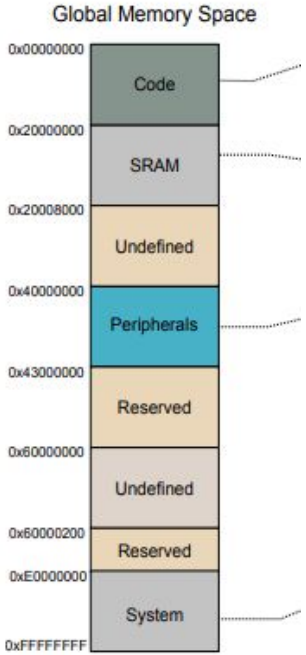
memory address may not be that size

- Common for 8-bit CPUs to have 16-bit addresses (**why?**)
- **What are the implications for atomicity?**

Harvard Architecture - code has separate memory space from data (common in MCUs)

- vs. Von Neumann - shared memory space (SAM D21 is Von Neumann)

Memory layout of SAMD 21 chip



Peripherals

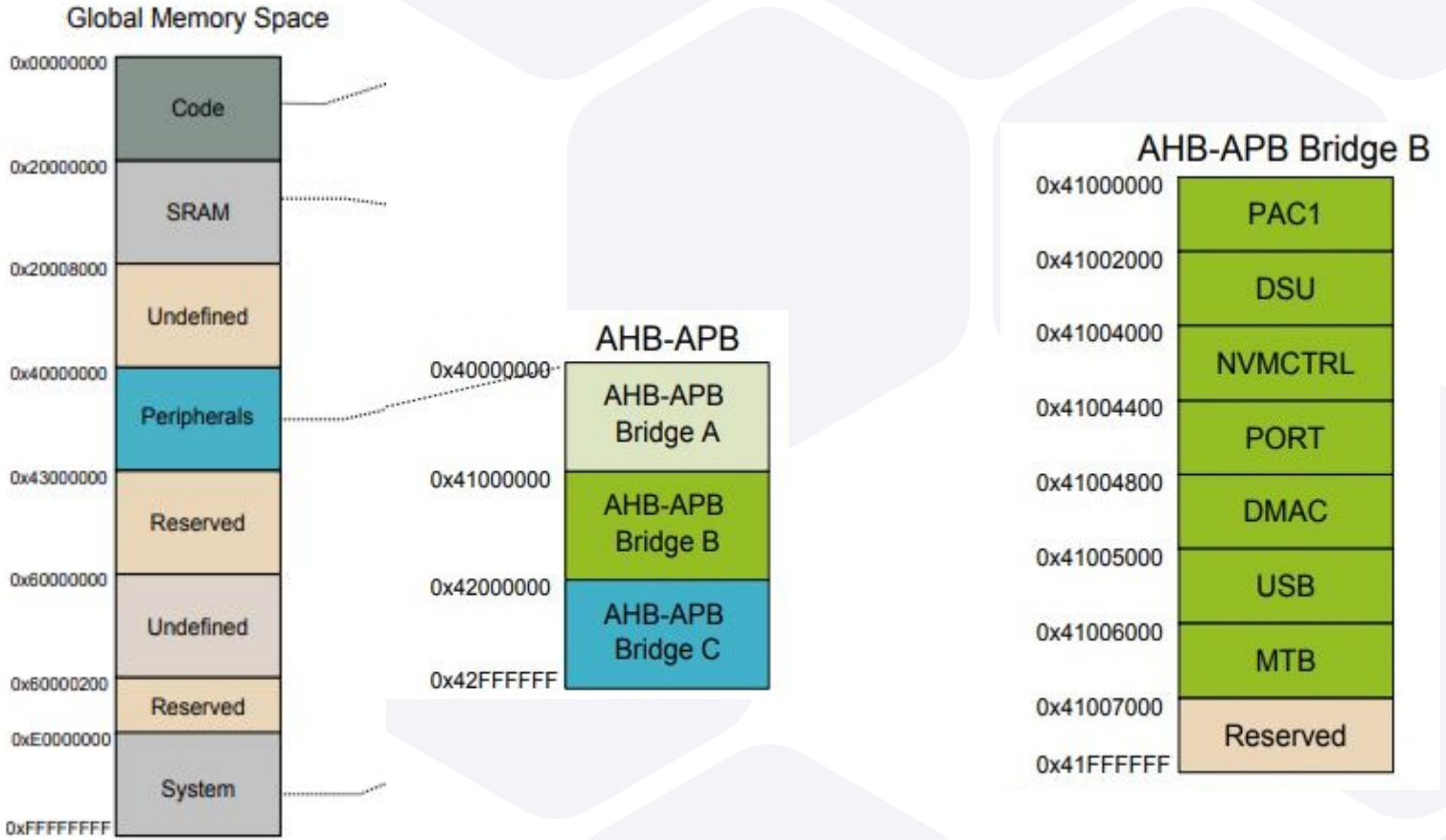
Timers, ADCs, GPIO, etc

Controlled by special registers (**different** from CPU registers!)

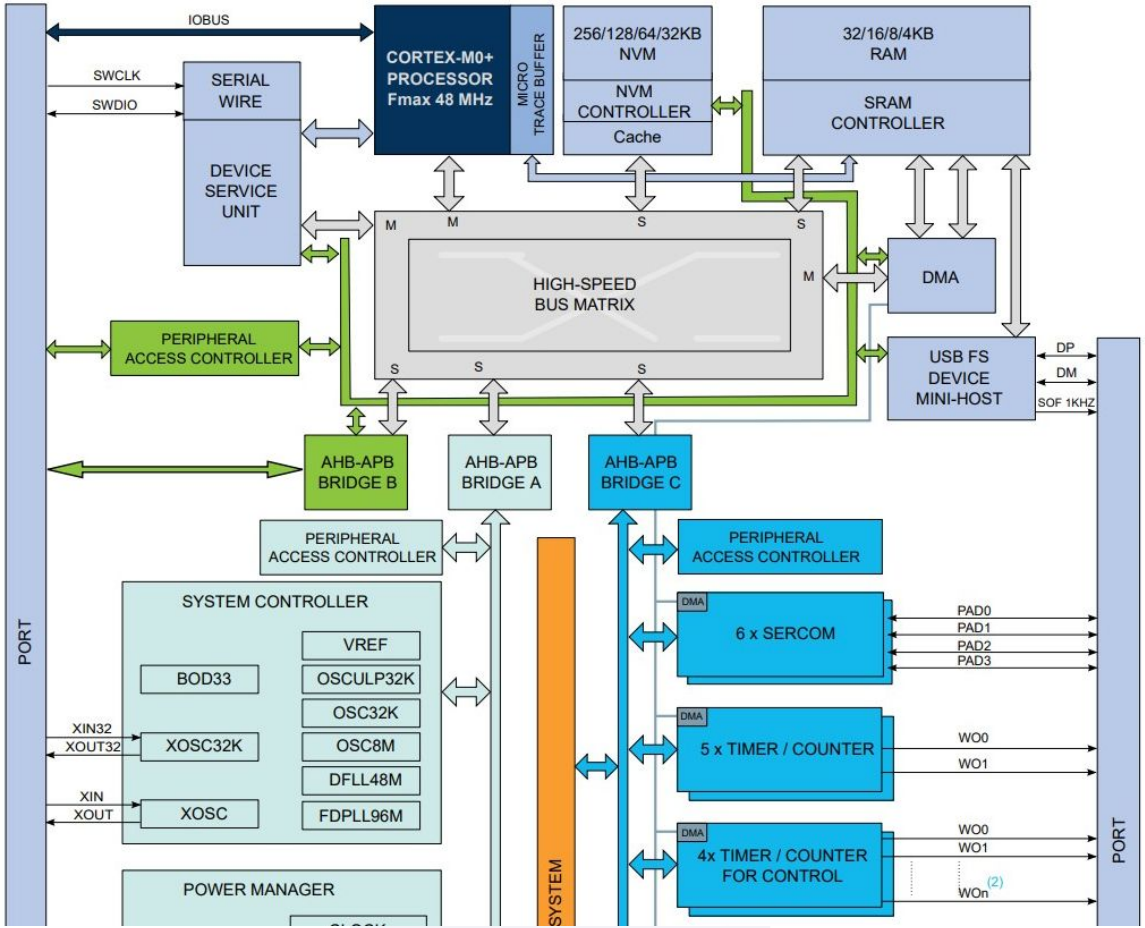
- You will see this in lab!

"Memory-mapped": from CPU perspective, just like writing to any other memory address

- From MCU perspective, need controller hardware to configure/send data to the right place

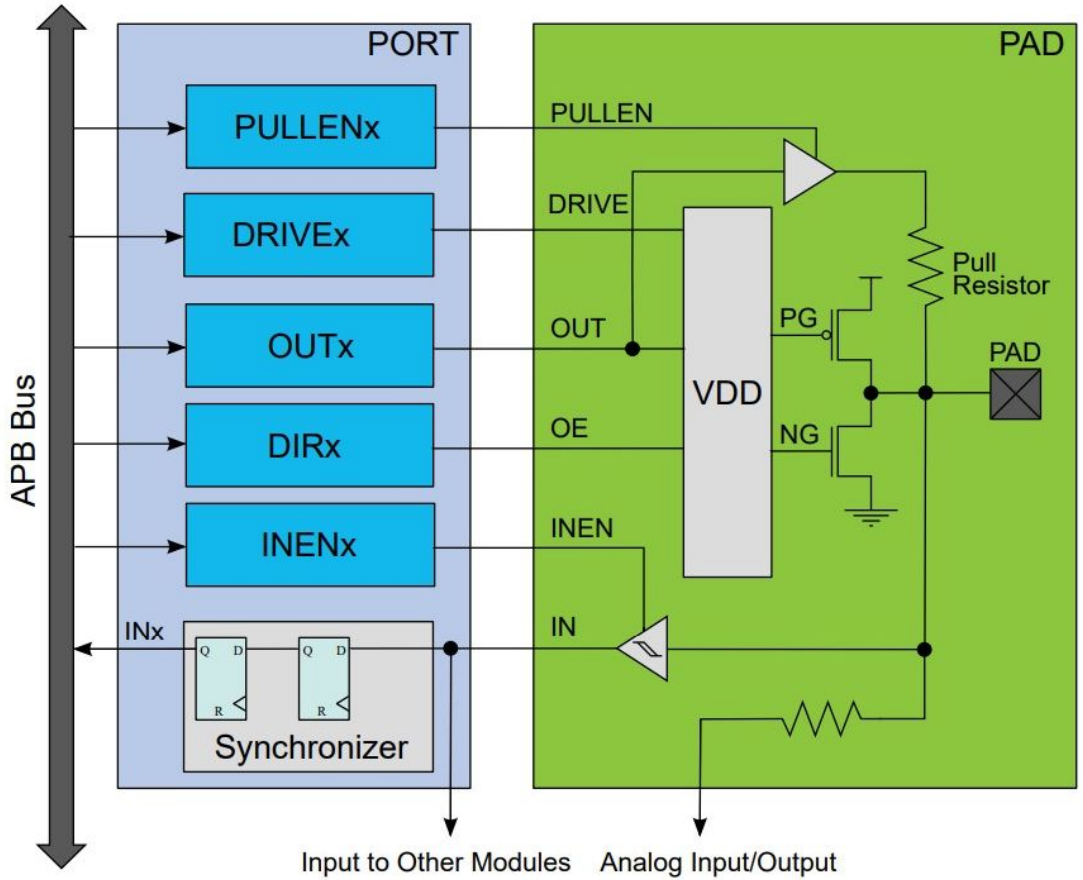

Global Memory Space
0x00000000
Code
0x20000000
SRAM
0x20008000
Undefined
0x40000000
Peripherals
0x43000000
Reserved
0x60000000
Undefined
0x60000200
Reserved
0xE0000000
System
0xFFFFFFFF
AHB-APB
0x40000000
AHB-APB Bridge A
0x41000000
AHB-APB Bridge B
0x42000000
AHB-APB Bridge C
0x42FFFFFF
AHB-APB Bridge B
0x41000000
PAC1
0x41002000
DSU
0x41004000
NVMCTRL
0x41004400
PORT
0x41004800
DMAC
0x41005000
USB
0x41006000
MTB
0x41007000
Reserved
0x41FFFFFF

4. Block Diagram



23.6 Functional Description

Figure 23-2. Overview of the PORT



How information gets onto an MCU

Bootloader

- Firmware on the board that can interface with the computer
- Copies memory on upload

Hardware programmer

- Special piece of hardware that connects to pins directly and transfers using a protocol

Why do we need to know about the type and layout of memory on an MCU?