


CC2511 Week 1: Lecture 2

“People who are more than casually interested in computers should have at least some idea of what the underlying hardware is like. Otherwise the programs they write will be pretty weird.”

-- Donald Knuth

Today: Computer Architecture

We want to run an **application** using **semiconductor physics**. How do we go from electrons and holes all the way up to computer applications?

- Application / Algorithms
 - Programming Language
 - Operating System
 - **Instruction Set Architecture (“Machine code”)**
 - **Registers / Arithmetic + Logic**
 - Digital circuits
 - Transistors and other devices
 - Solid State Physics
- 

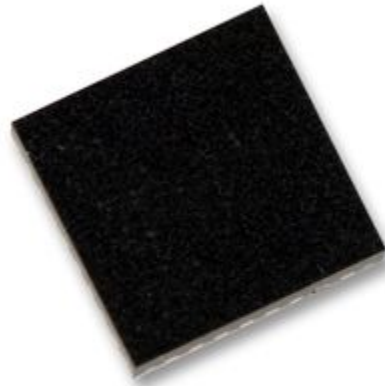
Computer architecture

- If this is your first time programming microcontrollers, then you're going to be working much “**closer to the metal**” than ever before.
- Microcontrollers are slow and limited. For example: the K20D50M board has:
 - 50 MHz clock (compared with several GHz on a PC)
 - 16 KB random access memory (compared with gigabytes on a PC)
 - 128 KB permanent storage (compared with terabytes on a PC)
- **Microcontroller software needs to be efficient!**

Why are microcontrollers limited?

- Why do we make microcontrollers with such limited capabilities?
- Microprocessors are cheap.

FREESCALE SEMICONDUCTOR - MK20DX128VFM5 - MCU, 32BIT, CORTEX-M4, 50MHZ, QFN-32




*Image is for illustrative purposes only.
Please refer to product description*

Manufacturer: FREESCALE SEMICONDUCTOR

Order Code: 2133567

Manufacturer Part No: MK20DX128VFM5

 [Technical Data Sheet \(1.70MB\) EN](#)

Product Information

- MCU, 32BIT, CORTEX-M4, 50MHZ, QFN-32
- **Architecture:** ARM Cortex-M4
- **Program Memory Size:** 128KB
- **RAM Memory Size:** 16KB
- **CPU Speed:** 50MHz
- **No. of I/O's:** 20



Sp

Availability

20 in stock for same/next working day delivery
490 deliver in 6 - 7 working days from our UK warehouse

[see cut-off times](#)

► [Check more stock...](#)

Price For: 1 Each

Minimum Order Quantity: 1

Order Multiple: 1

Unit Price: \$4.01

Qty

 **Buy**

Price

Qty	List Price
1 - 24	\$4.01
25+	\$3.83

More : [Get Quote](#)

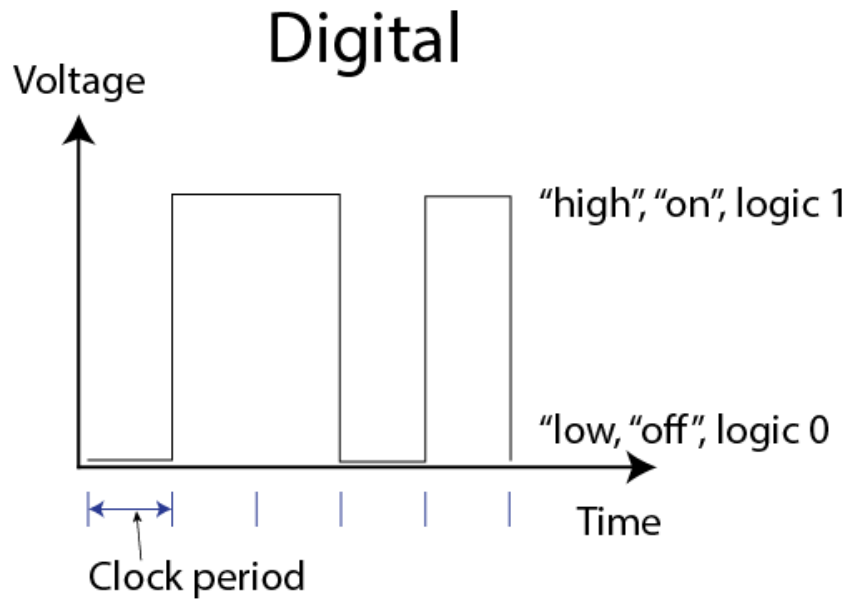
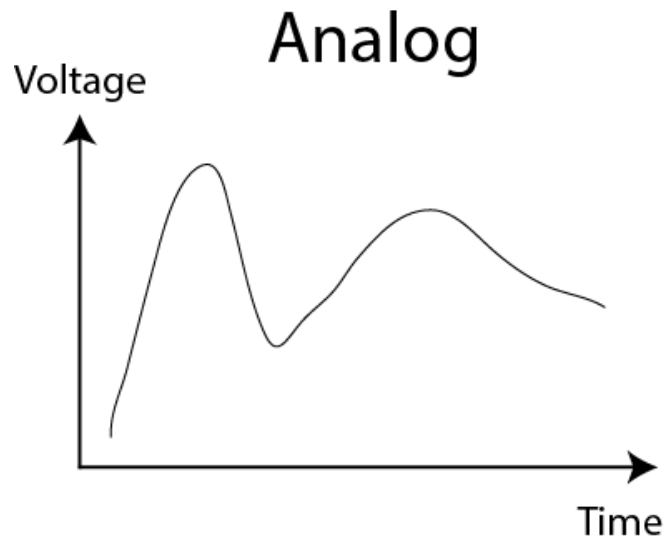
How can we write efficient software?

Microprocessors are generally programmed in **low level languages**.

- Very common choices are: assembly code and C.
- To be effective with these, we must understand the inner workings of the machine.

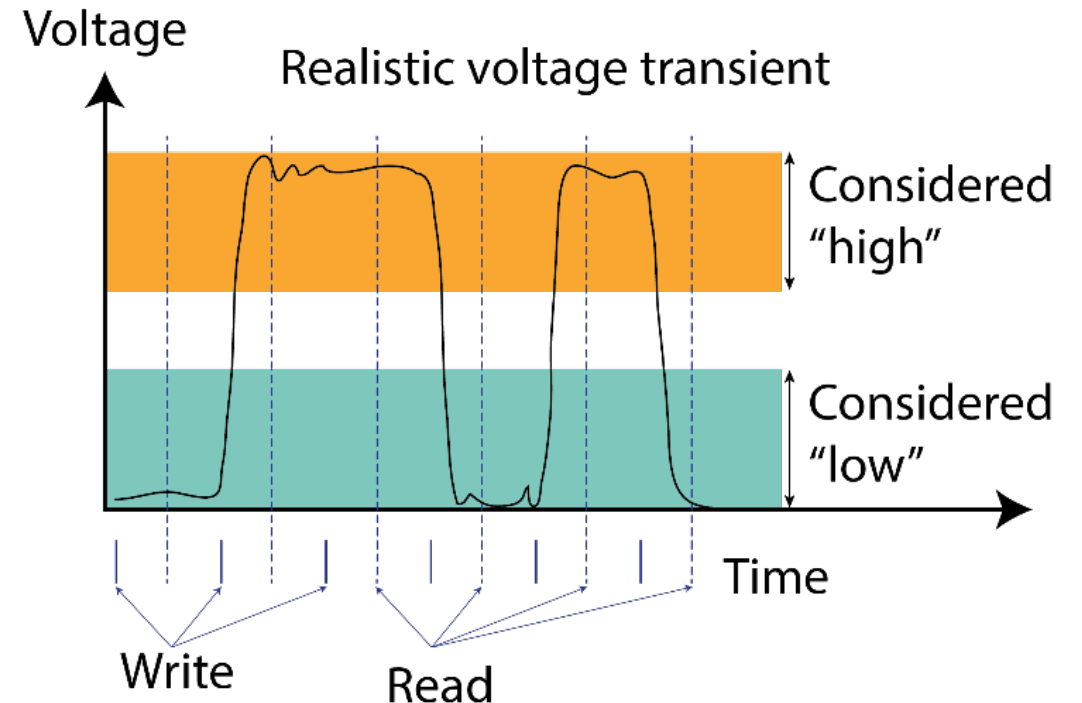
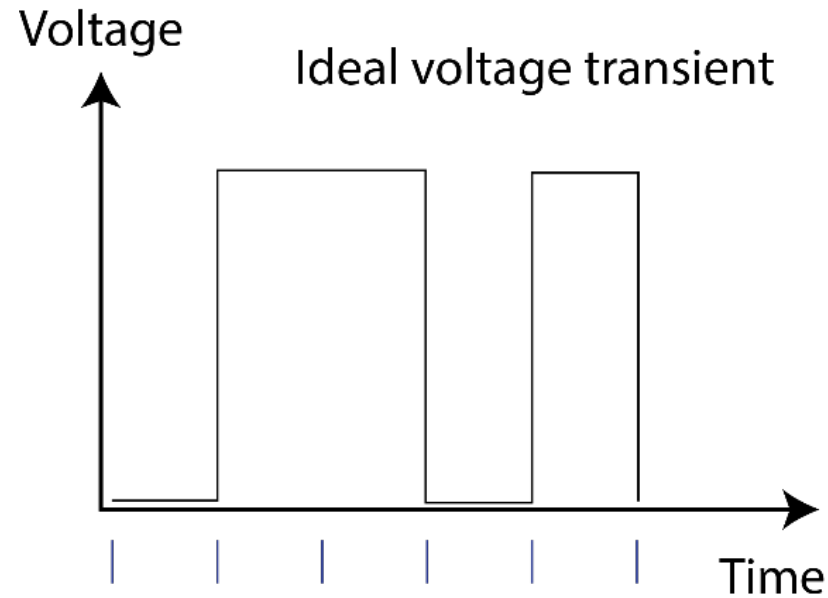
Fundamentals of digital electronics

- Microcontrollers are **digital electronics**.
- Internal voltage levels change periodically according to a **system clock**.



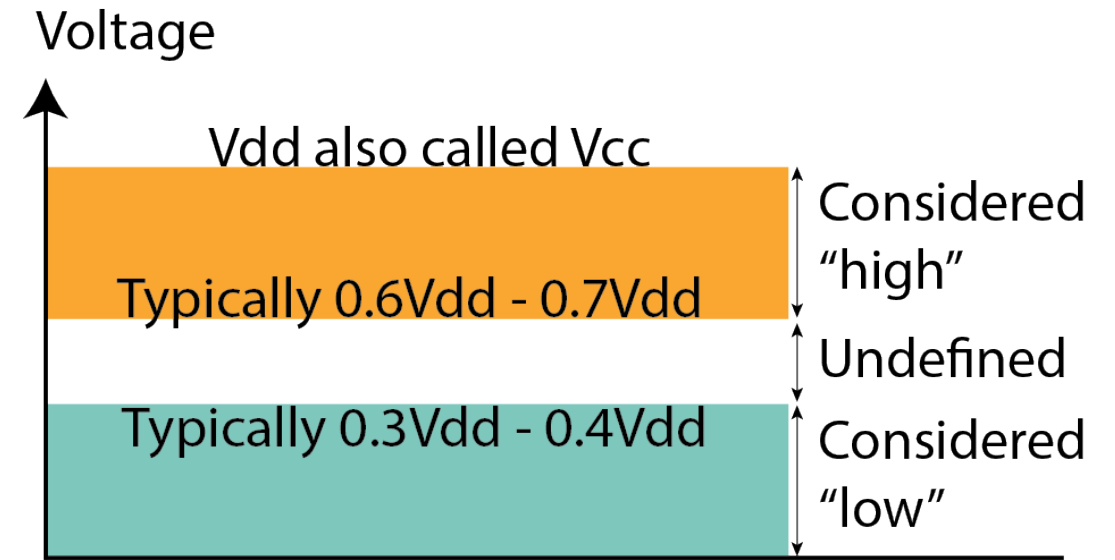
What are the advantages of digital electronics (compared with analog)?

- Digital is noise tolerant.
- Often digital signals can be exactly reproduced without error.



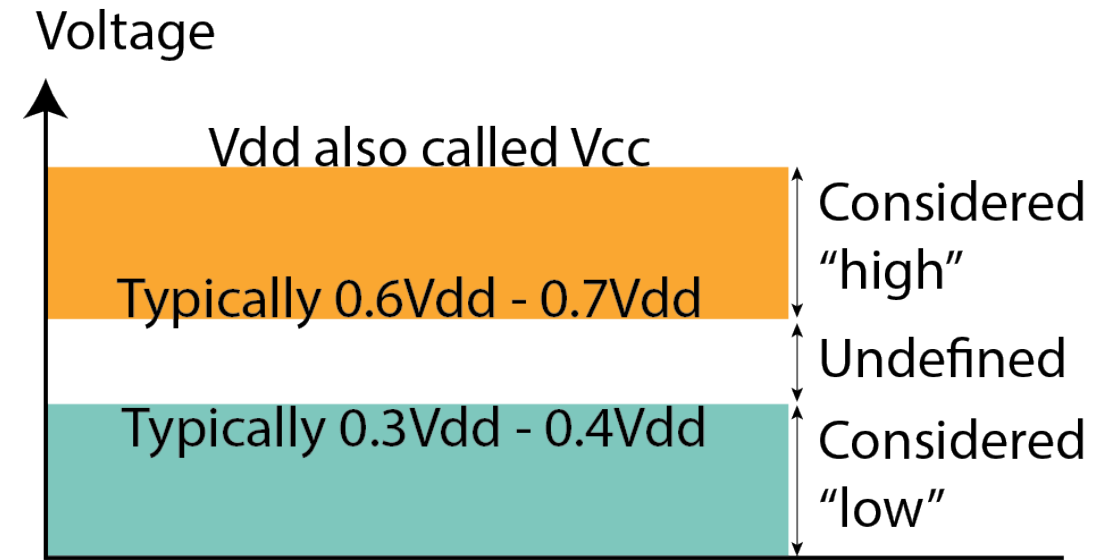
Terminology: voltages

- The digital supply voltage is called **Vdd**.
 - Origin of the name: “voltage on the drain” of the field effect transistors.
- Older terminology used interchangeably by most people: **Vcc**.
 - Origin of the name: “voltage on the collector” of bipolar junction transistors.



Terminology: voltages

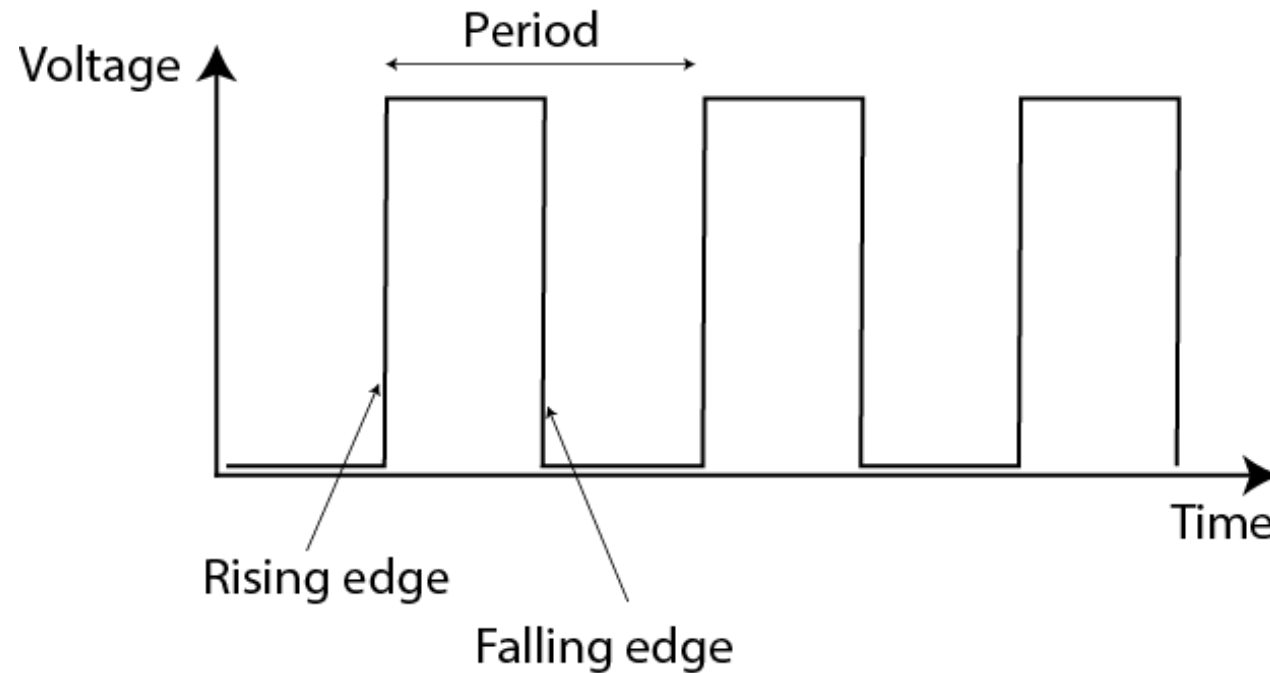
- Digital systems have large **noise margins** since the range of voltages considered high/low is quite large.
- Refer to the datasheets for each part to find the precise voltages that are considered as logic high or logic low.



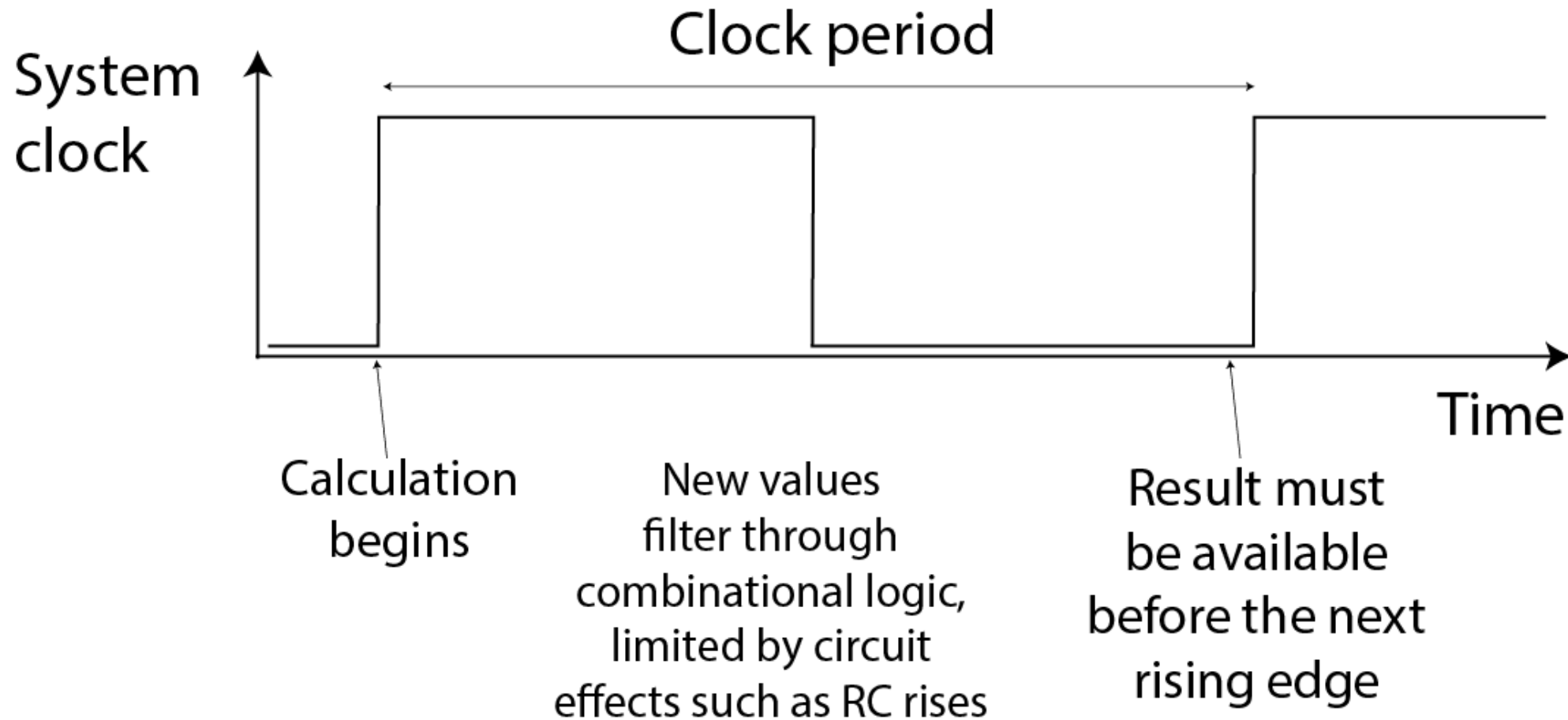
V_{IH}	Input high voltage <ul style="list-style-type: none">• $2.7\text{ V} \leq V_{DD} \leq 3.6\text{ V}$• $1.7\text{ V} \leq V_{DD} \leq 2.7\text{ V}$	$0.7 \times V_{DD}$	—	V	
		$0.75 \times V_{DD}$	—	V	
V_{IL}	Input low voltage <ul style="list-style-type: none">• $2.7\text{ V} \leq V_{DD} \leq 3.6\text{ V}$• $1.7\text{ V} \leq V_{DD} \leq 2.7\text{ V}$	—	$0.35 \times V_{DD}$	V	
		—	$0.3 \times V_{DD}$	V	

Clocked electronics

- Microprocessors operate on the basis of a system clock.
- A microcontroller clocked at 50 MHz has a period of $1/(50 \text{ MHz}) = 20\text{ns}$.

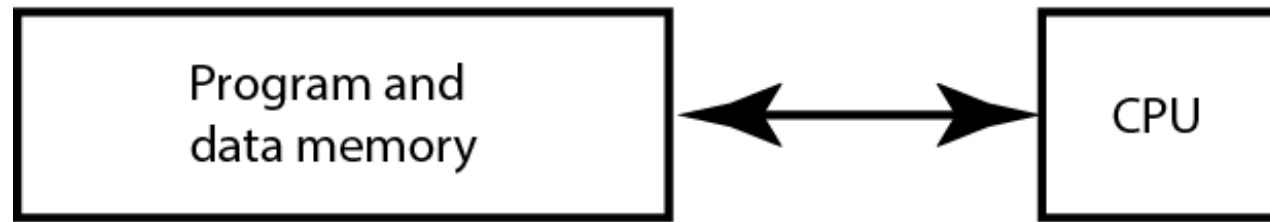


Calculations are organised around clock cycles

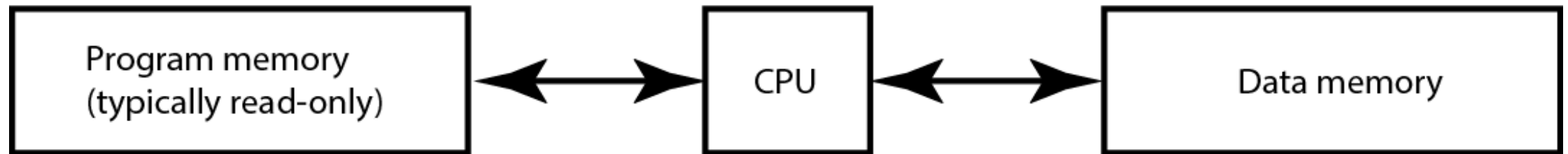


Two architectures

von Neumann architecture:

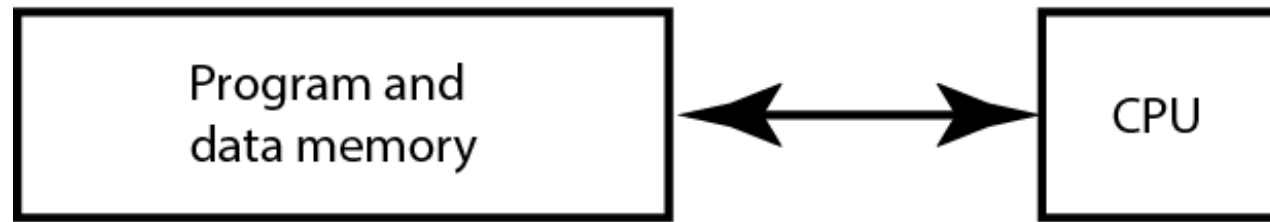


Harvard architecture:

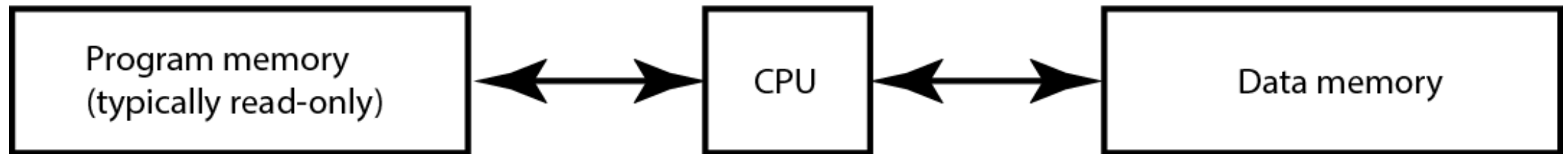


Discussion: advantages and disadvantages of each architecture

von Neumann architecture:



Harvard architecture:



Von Neumann vs Harvard

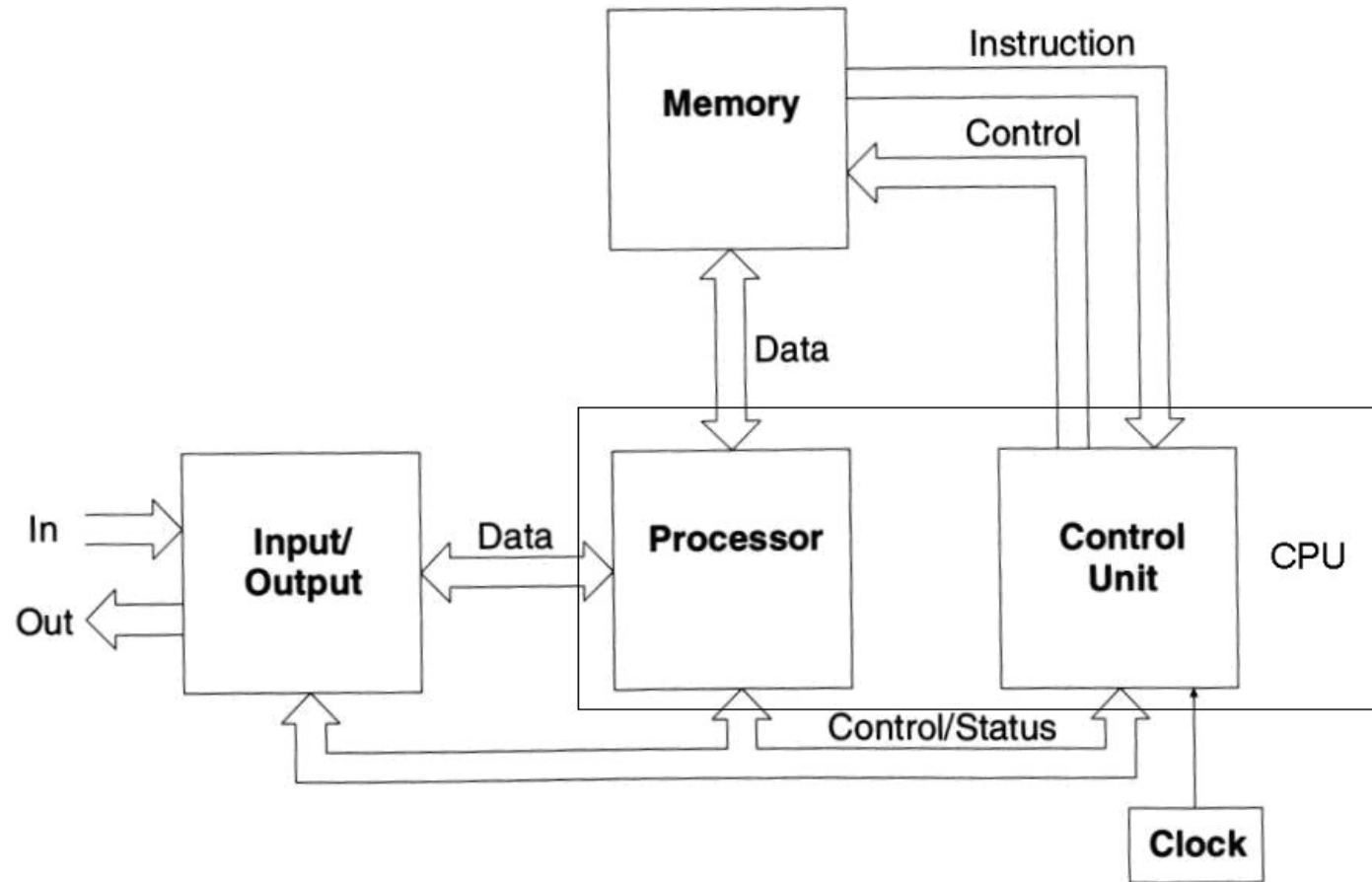
Von Neumann: (combined code and data memory)

- Typically used for general purpose computing.
- e.g. Intel and AMD desktop processors (x86 architecture).
- Programs can self-modify and the computer can dynamically load new programs. (“Code is data”)

Harvard: (separate code and data memory)

- High performance because separate memory banks can be read simultaneously.
- Usually code modification or runtime loading of new code is not possible / requires special effort.

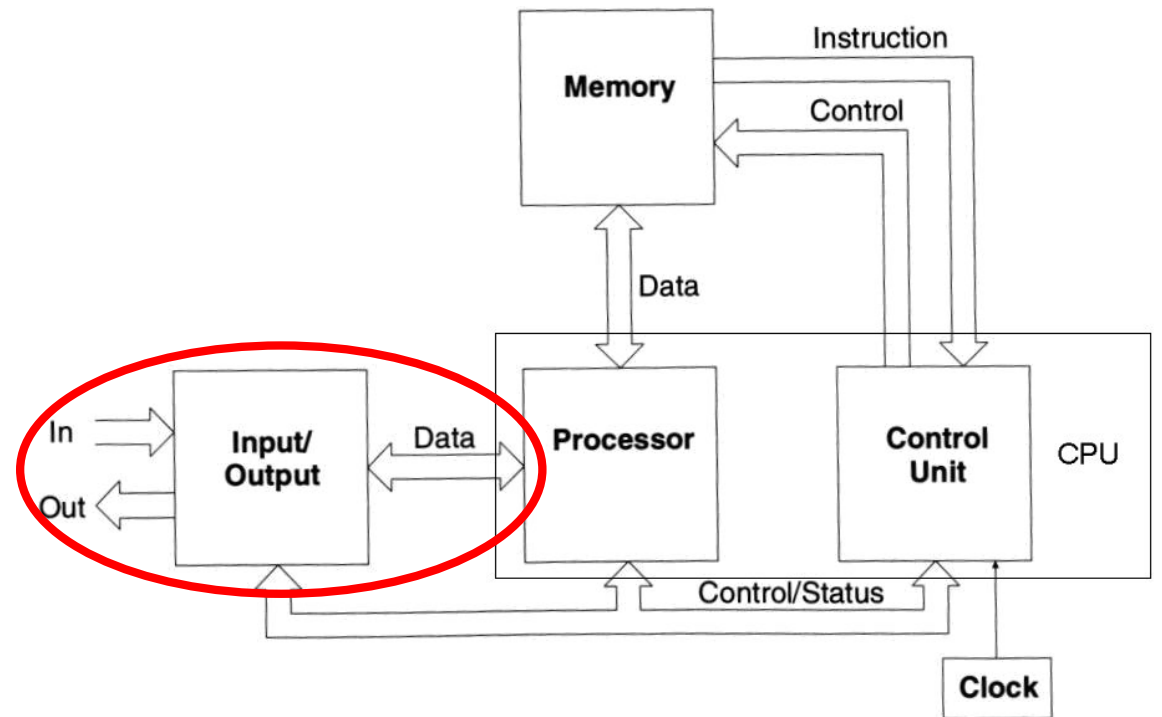
Von Neumann Architecture



Input/Output

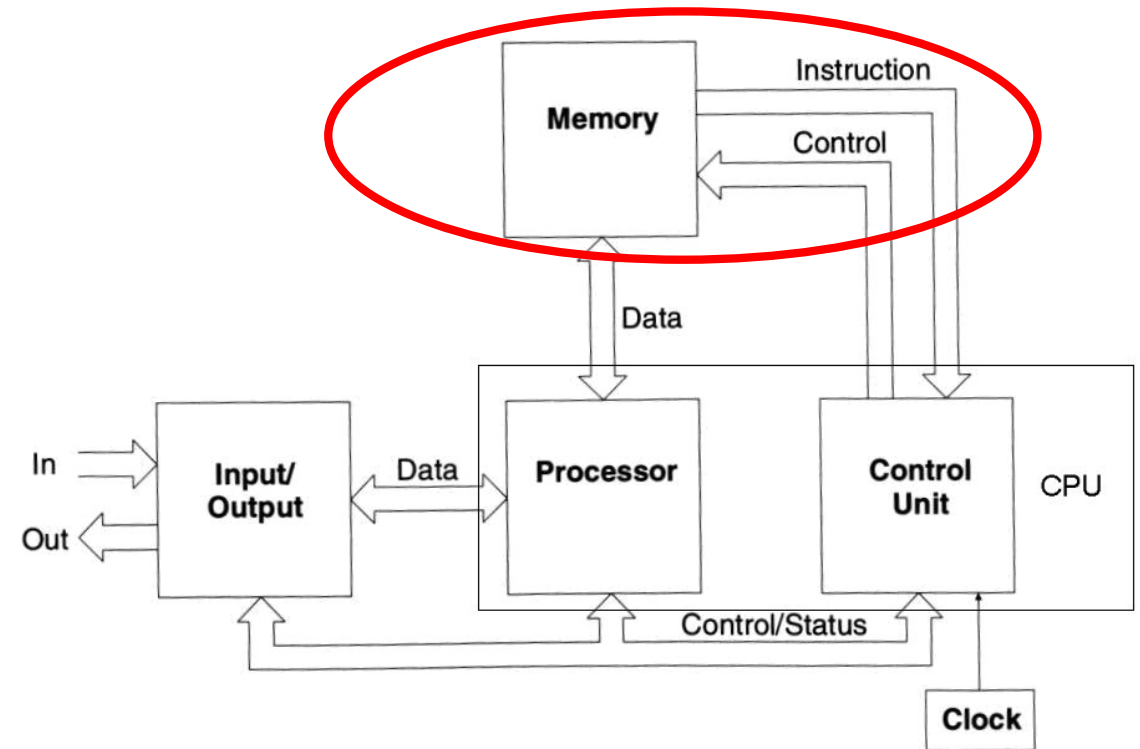
Input/Output could be:

- External storage (e.g. hard drive, USB memory stick)
- Network or Internet connection
- Keyboard/Mouse
- Sensors
- LEDs
- Motors



Memory

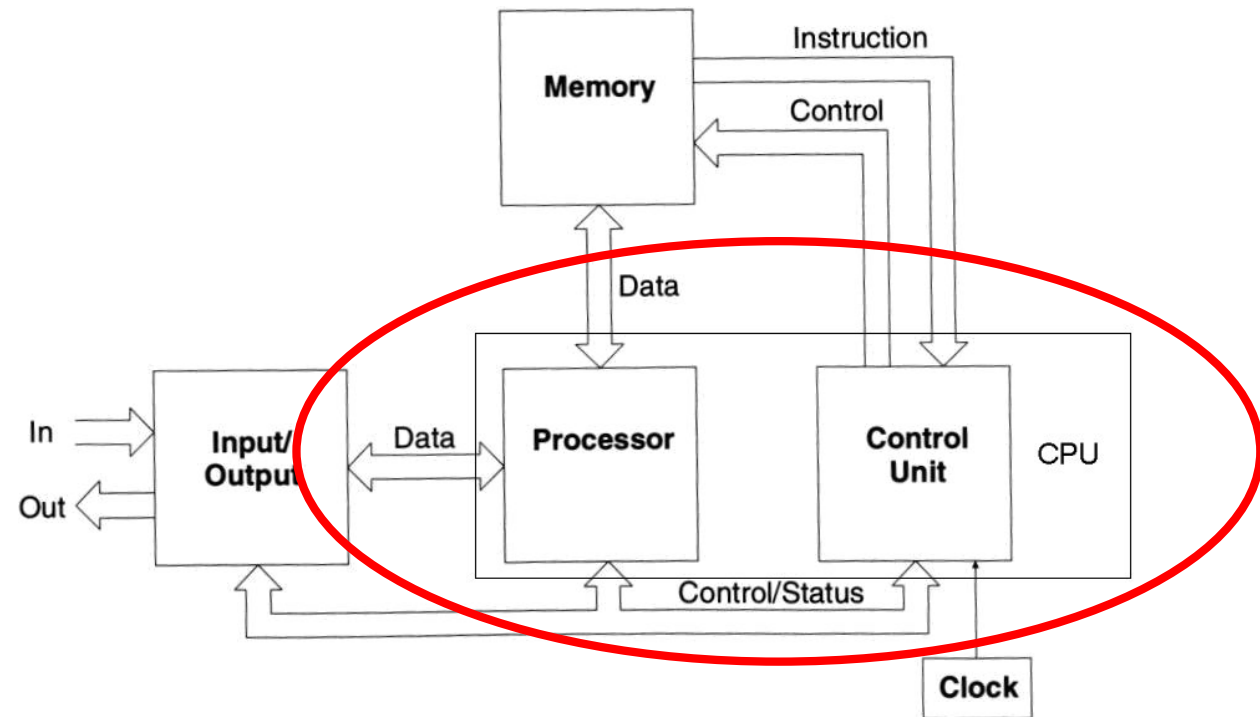
- Memory stores data at particular addresses
- Random Access Memory (RAM) stores **variables** and **computer code**
- On microprocessors, flash memory is often used for computer code.



CPU

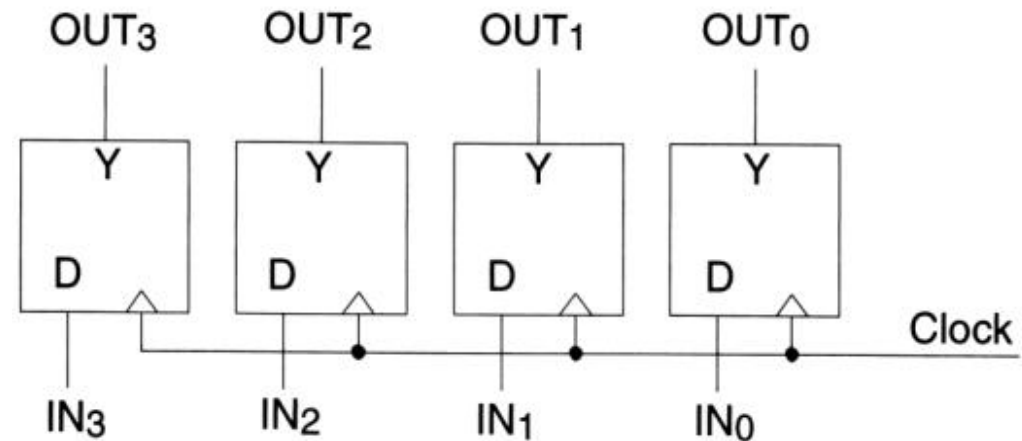
CPU (central processing unit)

- Performs calculations and makes decisions by following instructions
- Instructions are read from memory and then executed in the order they appear.



Registers

- A CPU holds the values it is currently using in **registers**.
- Registers are digital logic elements that store a given value.
- Registers are directly wired into the CPU's arithmetic and logic units.
 - Therefore, the values stored in registers are immediately available for computation



Any given CPU architecture will have a small number of registers (typical range is 1 – 32 general purpose registers).

The instruction pointer

- The **instruction pointer** is a special register that the CPU uses to keep track of which part of the program it will execute next.
- Also called the **program counter**.
- It gives the address in memory of the next instruction to execute.

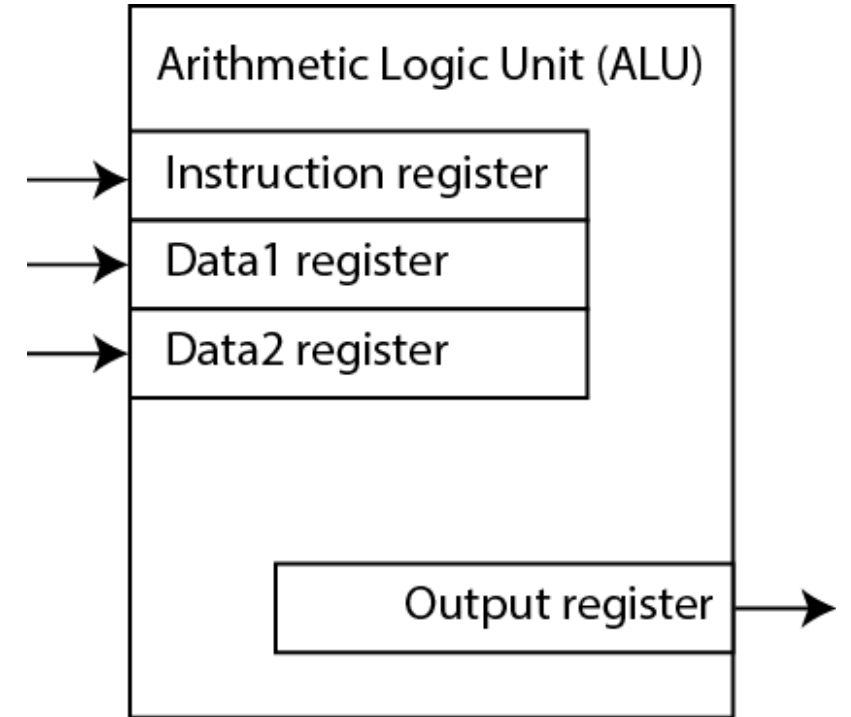
Machine Instructions

- An instruction is a single line of machine code (also called assembly code).
- Examples of possible instructions:
 - **Load** memory from address X into register A
 - **Add** registers A and B, placing the result into register A
 - **Store** the data in register A into address Y.
- Instructions are encoded into a numeric format called the **op-code** or operation code.

Arithmetic Logic Unit (ALU)

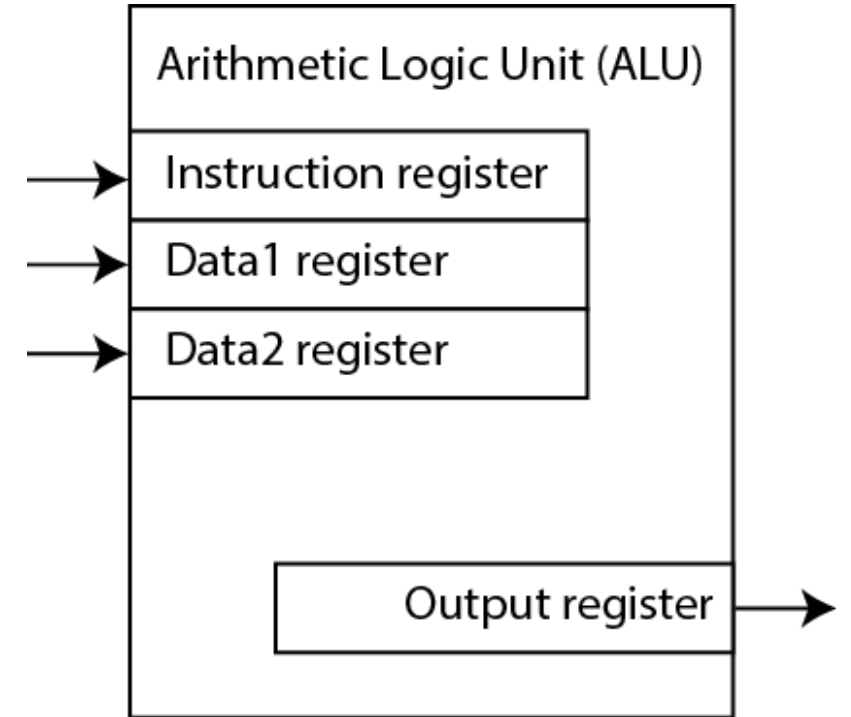
The arithmetic logic unit (ALU) performs:

- Arithmetic such as add and subtract
- Logic such as “is this value equal to zero?”



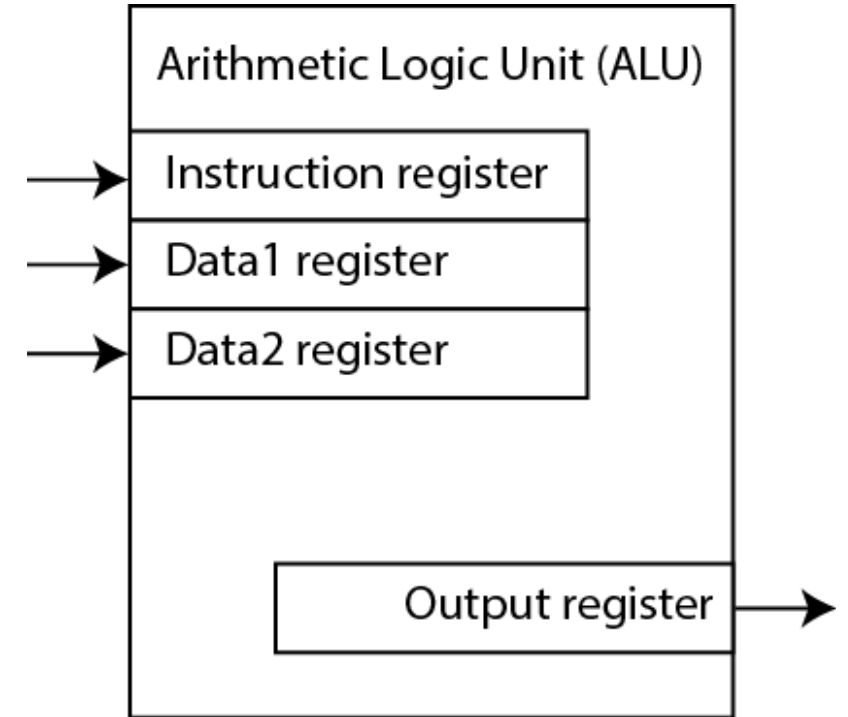
Arithmetic Logic Unit (ALU)

- The task to be performed in the next clock cycle is loaded into the **instruction register**.
- The **op-code** tells the ALU which operation it should perform.



Arithmetic Logic Unit (ALU)

```
“Pseudo-VHDL”:  
if (clock is rising edge) then  
  if (IR=1) then  
    OR <= Data1 + Data2  
  elsif (IR=2) then  
    OR <= Data1 - Data2  
  else ...  
    -- other opcodes here  
  end if  
end if
```



Word size

- The **word size** of a processor is the number of bits in its general purpose registers.
- Typical word sizes are 8 bits, 16 bits, 32 bits, and 64 bits.
- We are using 32 bit microprocessors
 - Therefore: All memory addresses and the default native integer are both 32 bits long.

Important note about terminology

- The **word size** of a CPU is the width of its data busses and the number of bits in its registers.
- A “**word**” in the C++ language is often 16 bits because the terminology became established when most processors actually had 16 bit words.
- To ease the transition from 16 bit machines to 32 bit machines, the new 32 bit integers were called “double words” or “long ints”.

Other registers

- Microprocessors often contain a variety of peripherals for connecting with the outside world.
 - For example: analog to digital converters, communication protocols, ...
- These are controlled by special registers.
 - For example, the analog to digital converter reads its instructions from a particular register.
- The processor maps these control and data registers into memory locations. They can be accessed like variables.
- This practice is called “memory mapped hardware”.

Registers

- **CPU registers** or **system registers** are directly accessible by arithmetic and logic operations.
- **Peripheral registers** are special parts of system memory that control other parts of the microcontroller (e.g. the voltage on a pin).

Memory and addressing

- Memory is addressed in units of bytes (i.e. 8 bits)
- For efficiency reasons, memory access often must be **aligned** to word size boundaries
 - On 32 bit machines, addresses should be multiples of 4.

Address	0x0	0x1	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC
Value	Byte	Byte	Short int		Int (32 bits)				Int (32 bits)				

Memory layout

- The memory layout for the microprocessor is documented in its **Reference Manual**

31.3 Register Definition

This section describes the ADC registers.

ADC memory map

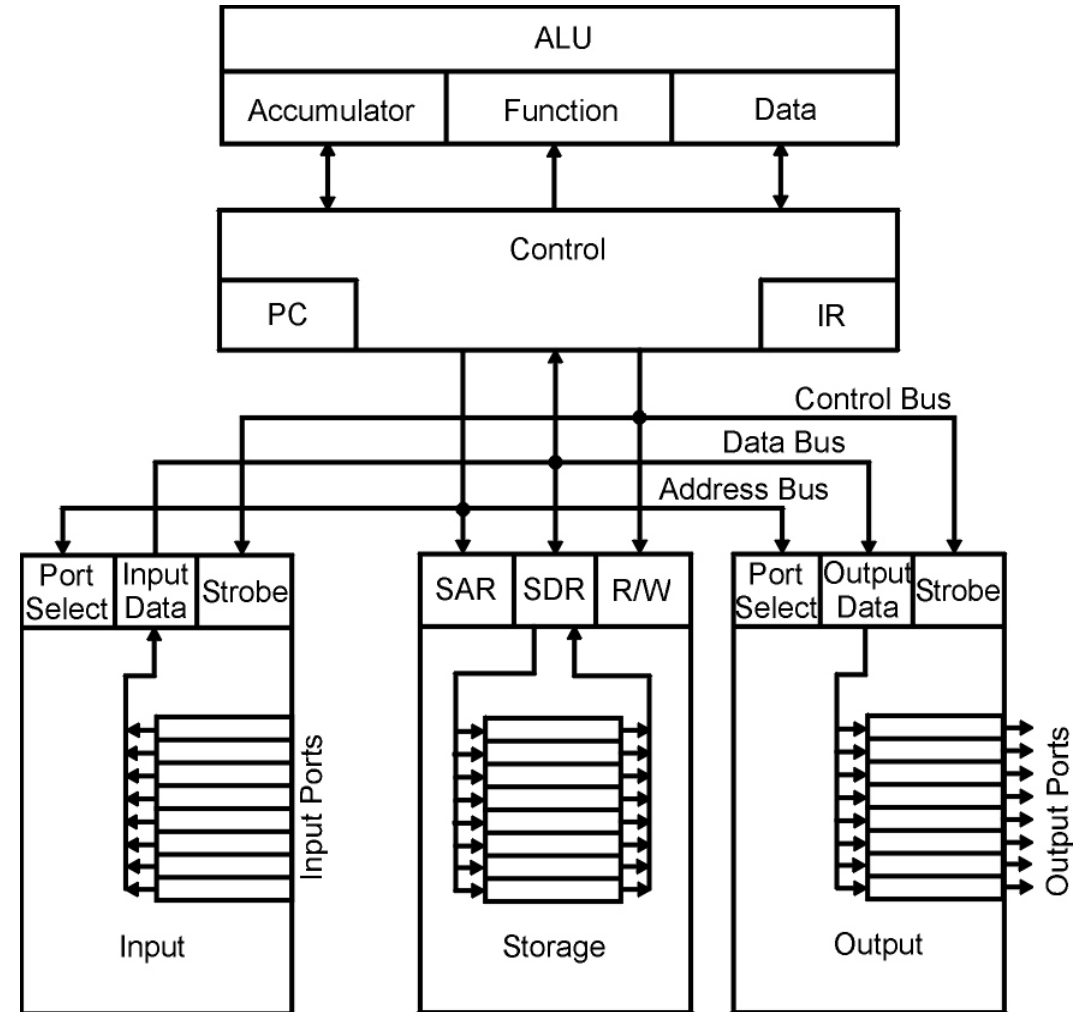
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4003_B000	ADC status and control registers 1 (ADC0_SC1A)	32	R/W	0000_001Fh	31.3.1/587
4003_B004	ADC status and control registers 1 (ADC0_SC1B)	32	R/W	0000_001Fh	31.3.1/587
4003_B008	ADC configuration register 1 (ADC0_CFG1)	32	R/W	0000_0000h	31.3.2/590

Table continues on the next page...

A simple model CPU

Let's consider a very simple model CPU to introduce the concepts.

(Real machines are not necessarily built like this!)



Memory hierarchy

- Memory access is extremely slow in comparison with register access.
- Use small amounts of faster memory to **cache** frequently-used items

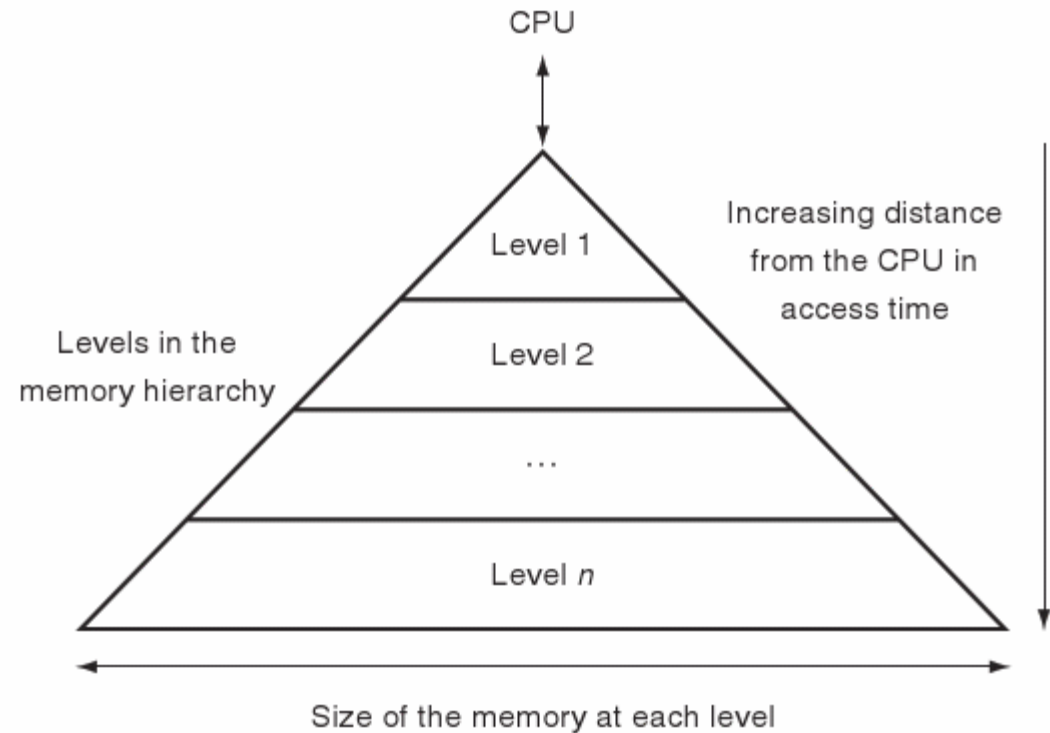
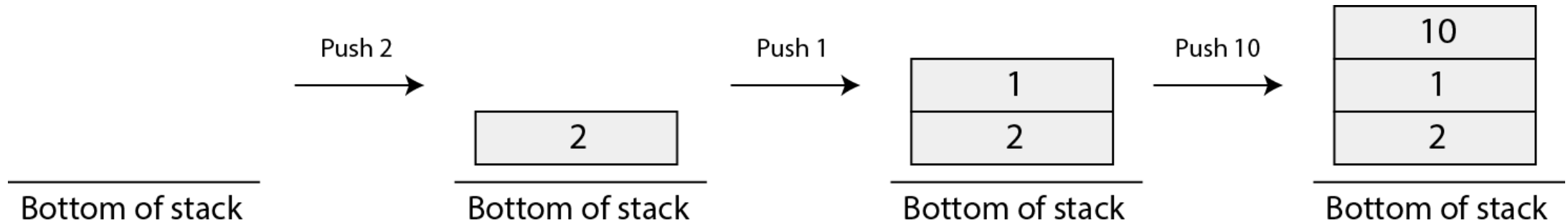


Image from D.A. Patterson and J. L. Hennessy, *Computer Organization and Design*, 3rd edition.

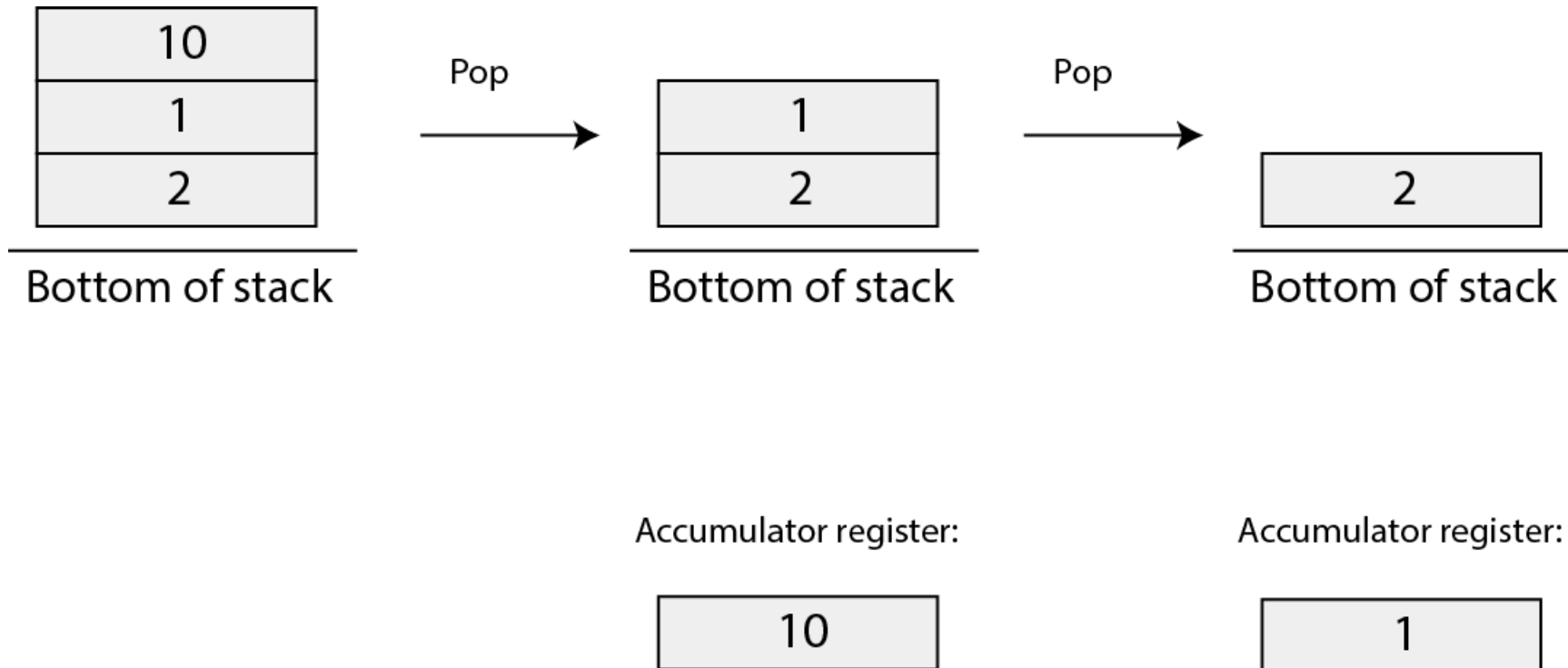
The stack

- Almost every CPU design includes the concept of a **stack**.
- A stack provides a hierarchy where new values are added “on top” of old values.
- Values are added onto the **top of the stack**.

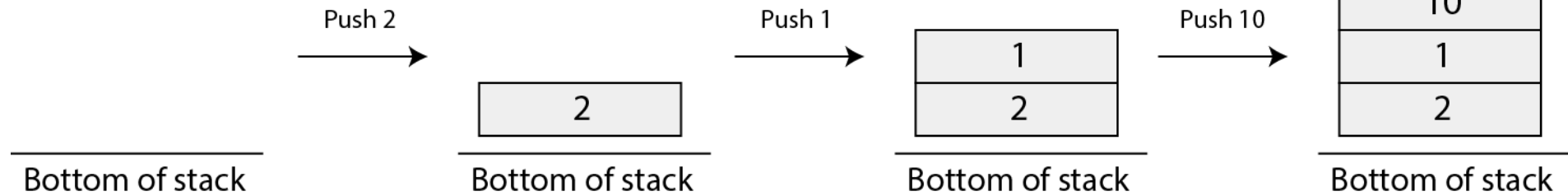


Removing items from the stack

- Items are removed from the stack in the **reverse order** to which they were added.

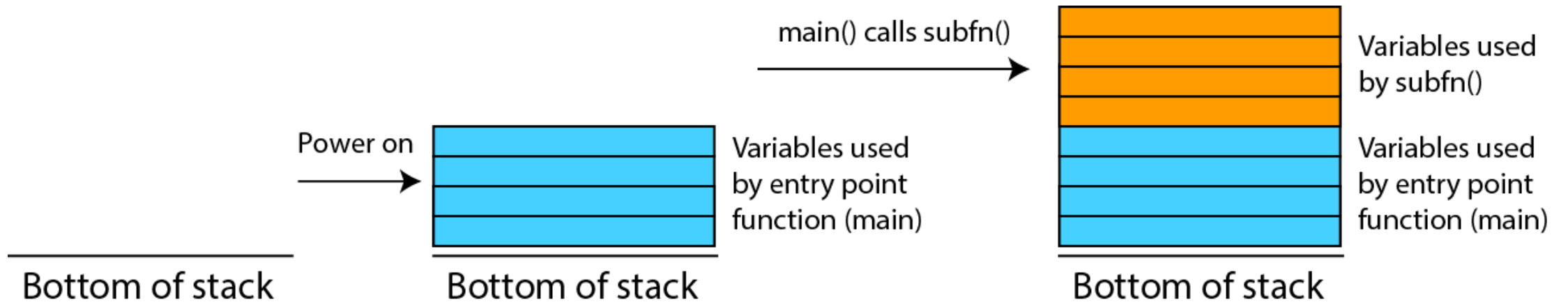


Why are stacks useful?



Stack layout for function calls

- **Function calls** add their own variables to the top of the stack



Summary: functional description of CPU architecture

- The **instruction pointer** (or **program counter**) points to the next instruction to be executed
- Each instruction is encoded into an **op-code**
- The op-code is loaded into the **instruction register** which signals the respective parts of the CPU to perform the appropriate actions
 - e.g. the **arithmetic logic unit (ALU)** manipulates registers to perform arithmetic and logic operations
- The instruction pointer is updated with the next address and the process repeats.

Summary: memory-mapped hardware

- Peripherals are components that interact with the world outside the microprocessor, such as:
 - Digital input/output (I/O)
 - Serial communications
 - Analog to digital converters (ADC)
- These peripherals have their own registers which are mapped into specific memory locations.
- The processor controls the peripherals by writing into these memory-mapped registers.