CSC 230: Introduction to Computer Architecture

Dr. Ahmad Abdullah
abdullah@uvic.ca

Department of Computer Science

University of Victoria

Spring 2023

Lectures: ECS 125, MR 10:00 – 11:20 am
Office: ECS 617, TW 10:00 – 11:00 am

00- Intro

Ahmad Abdullah, PhD
abdullah@uvic.ca
https://web.uvic.ca/~abdullah/csc230

Lectures: MR 10:00 – 11:20 am
Location: ECS 125

Course Webpage

- https://bright.uvic.ca
- Updated periodically (check before class)
    - Announcements
    - Slides
    - Assignments
    - LAB materials
    - Other materials

Introduction to Computer Architecture

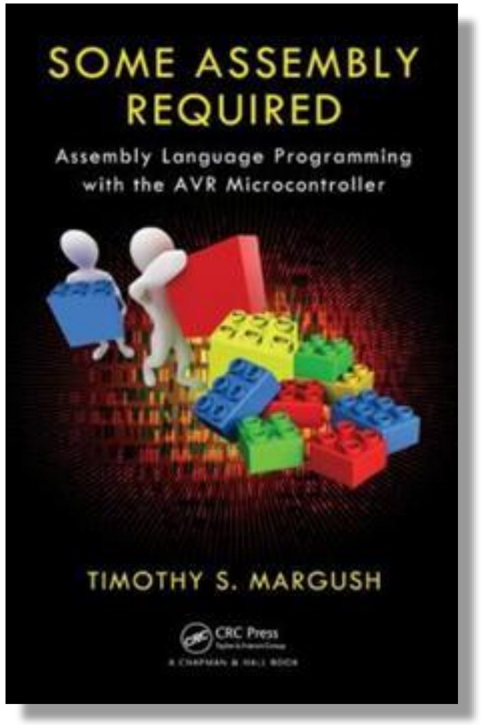
- This course aims to provide an introduction to basic principles of the architecture of modern computer systems. The course includes concepts such as CPU, memory, buses, I/O, cache, instruction sets, interrupt processing, pipelining, performance. Families of processors, CISC, RISC. Memory organization and management (including virtual memory, protection, segmentation and paging). Computer arithmetic. The use of assemblers, linkers and loaders. Assembly language programming and its interface with a high-level languages such as C langaue.

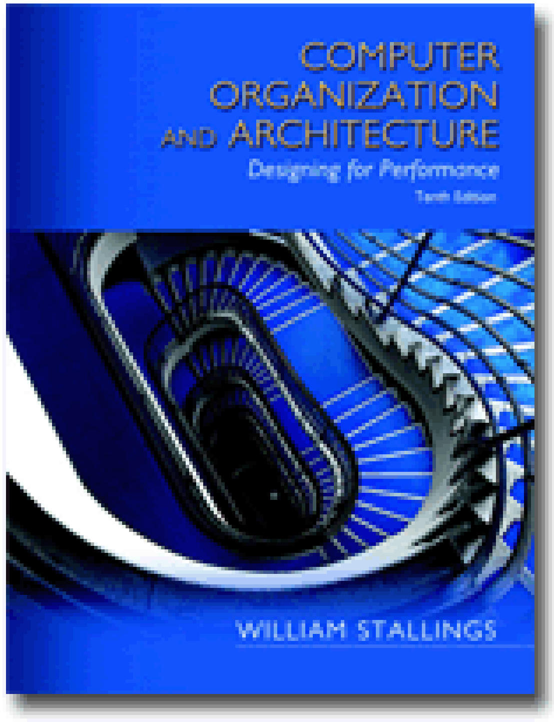
What this course covers?

- The architecture of computer systems and instruction sets
- The organization and interaction of CPU, buses, and memory
- The architecture of RISC and CISC machines
- Input / Output (I/O) and interrupt processing
- Memory organization and management (i.e., cache and virtual memory)
- Performance
- Interface with software: assemblers, linkers, and loaders
- Assembly language programming
- Interface between assembly language programming with high-level language through function call conventions
- The Atmel AVR processor as an in-depth case study

Textbook

- Recommended Textbooks
  - List provided in the course outline.




COMPUTER ORGANIZATION AND ARCHITECTURE
Designing for Performance
Tenth Edition
WILLIAM STALLINGS

Evaluation

- Official course outline:
  - https://heat.csc.uvic.ca/coview/course/2023011/CSC230
  - Link to outline also available via Brightspace site
- Four assignments (all programming, 10% each)
- Lab participation (5%)
- Two midterm exams (Thursday, February 16th; and Thursday, April 6th, 15% each)
- One final exam (scheduled by UVic admin, 25%)
- Please read the outline ASAP!
- Link to outline is provided at the Brightspace site for our course

Evaluation

Assignment/Exam	Weight	Assigned Date	Due Date
Assignment 1	10%	19-Jan	26-Jan
Assignment 2	10%	02-Feb	09-Feb
Midterm 1	15%	16-Feb	-
Assignment 3	10%	09-Mar	16-Mar
Assignment 4	10%	23-Mar	30-Mar
Midterm 2	15%	06-Apr	-
Final Exam	25%	TBD	-

Lab participation 5%.

Labs != Assignments

- First lab begins the week of January 16th, 2023
  - All labs will take place in ECS 249
  - You must be registered for a lab section
- Please: No switching of lab sections!
  - Attend the section in which you are registered
  - Enrollment numbers are high this semester, and we cannot accommodate informal lab switching
- Note that this is not “assignment completion” time!
  - Work on assignments should normally be done outside of scheduled lab-session time.
  - If you do not already have an Engineering Access Card, then please obtain one (details: https://bit.ly/3eCCWaI)

SPRING 2023 Calendar

Week	Monday	Tuesday	Wednesday	Thursday	Friday
1	09-Jan First day of classes	10-Jan	11-Jan	12-Jan	13-Jan
2	16-Jan	17-Jan	18-Jan	19-Jan	20-Jan
3	23-Jan	24-Jan	25-Jan	26-Jan Assignment #1	27-Jan
4	30-Jan	31-Jan	01-Feb	02-Feb	03-Feb
5	06-Feb	07-Feb	08-Feb	09-Feb Assignment #2	10-Feb
6	13-Feb	14-Feb	15-Feb	16-Feb Midterm #1 [15%]	17-Feb
7	20-Feb	21-Feb	22-Feb	23-Feb	24-Feb
	Family Day & Reading Break				
8	27-Feb	28-Feb	01-Mar	02-Mar	03-Mar
9	06-Mar	07-Mar	08-Mar	09-Mar	10-Mar
10	13-Mar	14-Mar	15-Mar	16-Mar Assignment #3	17-Mar
11	20-Mar	21-Mar	22-Mar	23-Mar	24-Mar
12	27-Mar	28-Mar	29-Mar	30-Mar Assignment #4	31-Mar
13	03-Apr	04-Apr	05-Apr	06-Apr Midterm #2 [15%]	07-Apr Last day of classes

Credits

- Special thanks to Professors Mike Zastre, Bird, Corless, Ganti and Jackson for letting me use their course material.

Thank you!

Class Interaction

- We will be using www.menti.com a little bit during lectures this semester
  - Informal polls
  - Non-graded quizzes and questions
  - Anonymous
  - None of this on menti.comis for course credit!
- Link and code to Mentimeter will be shown when there is something to do.
- You do not need an account with Mentimeter to complete any of this.
  - Ignore everything at www.menti.comasking you to sign up for an account!

What is a computer?

“... programmable electronic device that can store, retrieve, and process data.” [Taken from random intro-to-programming textbook.]

- A **device** that:
  - Accepts **input** (e.g., keyboard, mouse, touch screen, network interface, sensor)
  - Processes data
  - **Stores** data (usually in some form of memory)
  - **Retrieves** data (ditto)
  - Produces **output** (e.g., printed, display, actuator, network interface)

Image: http://bit.ly/2DYDzn6

Menti.com activity

- What do you think of computers?

https://www.menti.com/alcg54awpef2

Is this a computer?

Image: https://bit.ly/3xR6KFr

Parking aid
Climate control
Alarm & immobilizer
Automatic wiper control
Airbag control unit
Electric window & central locking
Anti-lock braking system
Automatic transmission
Electric motor management
Gas motor management
Battery-recharge control

etc.

Types of Computers

- There are some traditional **categories** ...
- ... Yet computing devices are now so ubiquitous it is easy for a computer to be "invisible"
- Personal or microcomputers
  - Desktops
  - Notebooks, Laptops
- Handheld (smartphones, PDAs)
- Mainframes
- Supercomputers
- Cloud computing environments (Amazon ECS, Microsoft Azure, Google Cloud Platform)

Types of Computers

- **Personal Computers** (PCs)
- Used for general computing tasks
- Fit the needs of most users

Images: https://bhpho.to/2lNPNsi, http://bit.ly/2CNAnLX

Types of Computers

- **Hand-held computers**
- Fit in the palm of your hand and run on batteries
- Perhaps the form of computers used directly by most people today
- (“Tablet computers” are a good crossover category crossing PCs and handhelds)

Images: http://bit.ly/2jzlvaG, http://bit.ly/2AgKqGh, http://bit.ly/2CAkUB0

Types of Computers

- **Mainframes**
- Used by companies to provide centralized storage, processing, and management for large amounts of data
- Nowadays made up of racks of blades, seen in data centres
- Important focus especially placed on fast I/O

Images: http://bit.ly/2CyqfcU, http://bit.ly/2qiD3yv

Types of Computers

- **Supercomputers**
- Largest and fastest of computers
- Meant to combine the ultimate in processing speed and data throughput
  - But focus is mostly on speed
  - Massive power consumption!
- `http://www.top500.org`

Images: http://bit.ly/2ENittc, http://bit.ly/2ClwGvV

Types of Computers

- **Cloud Computing**
- Not necessarily a different category of computer, but a particular way in which existing mainframes are used
- Location independent computing
- Shared servers provided resources
- Software and data on demand
  - (think of computing here as a “utility” like electricity or pay-per-use)

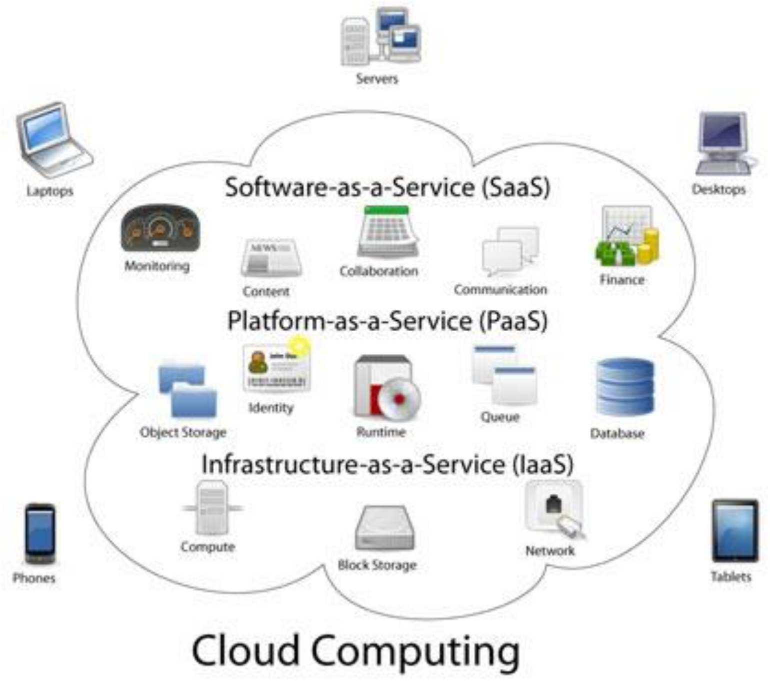
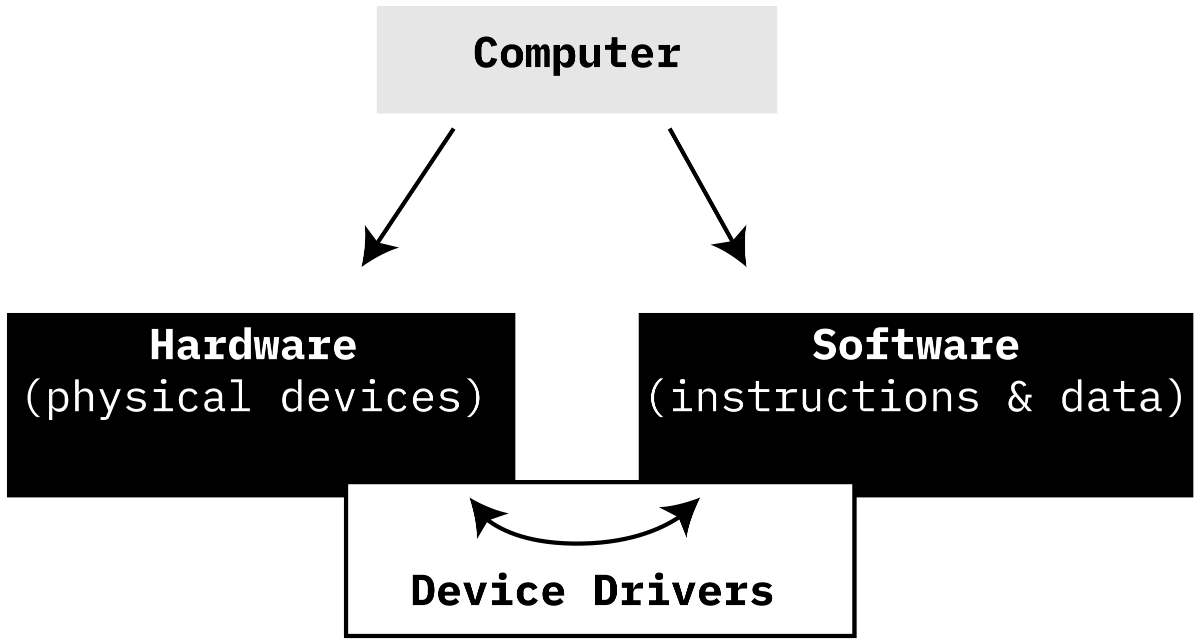


Image: http://bit.ly/2CNQv03

Components of a computer system

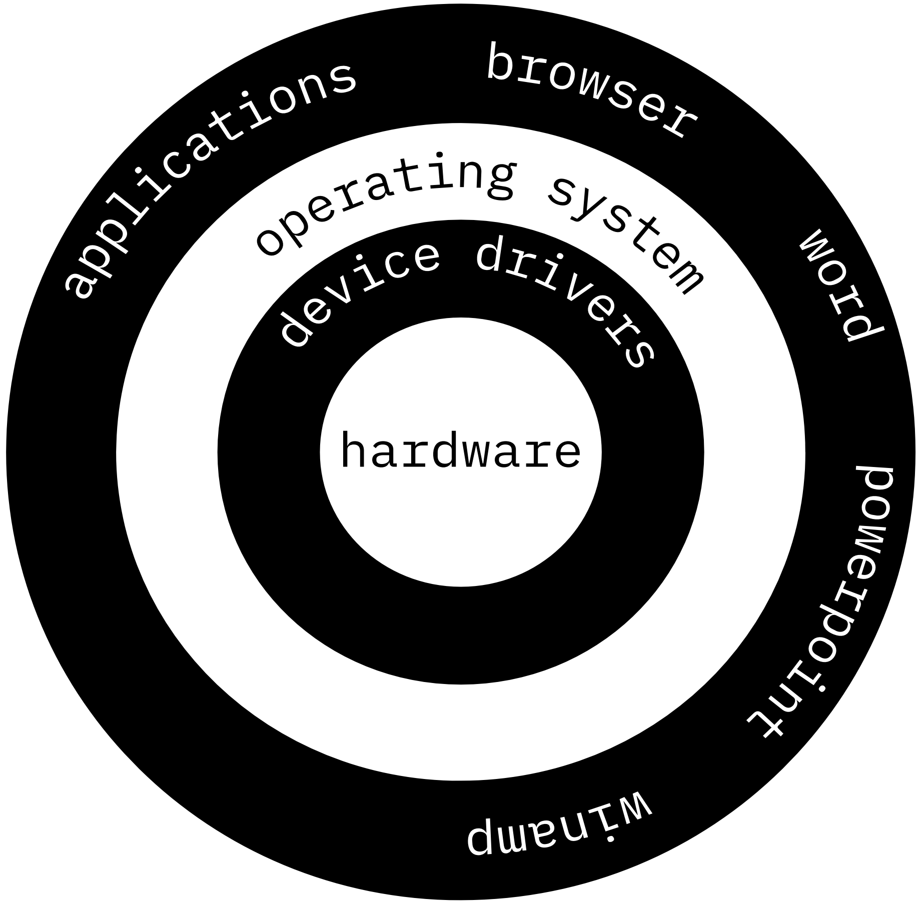


Hardware and software “communicate” through pieces of specialized code known as “device drivers”.

What is a “computer”?

- We usually also refer to a **computer system**
- **Hardware**: comprehensive term
  - Physical components...
  - ... plus the way the are interconnected (**wired**)
  - Pre-defined functionality (“**hard-wired logic**”) is very fast but hard to change
- **Software**:
  - The programs or lists of instructions
  - Thought experiment: what kinds of instructions would be needed to control the hardware properly?
  - Logic and functionality can be easily changed
- **Peripherals** (i.e., “optional”, not the core)
  - Additional components

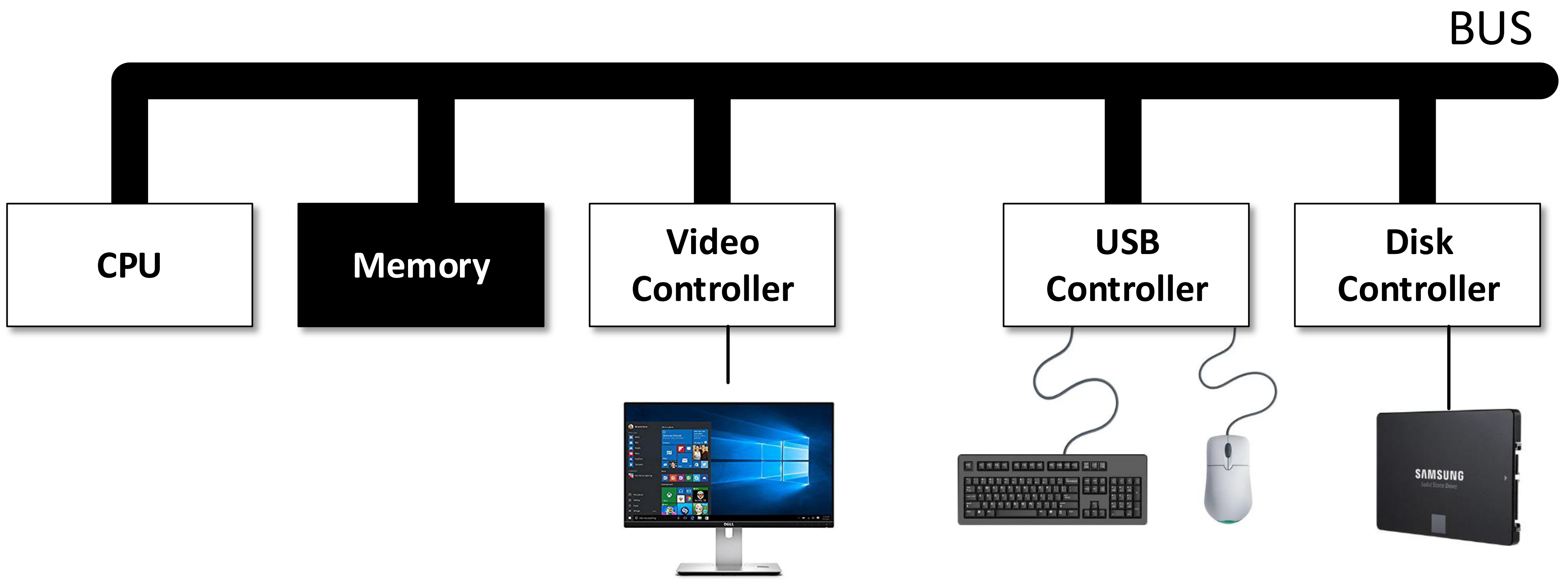
Shell model of computer system



Computer Hardware

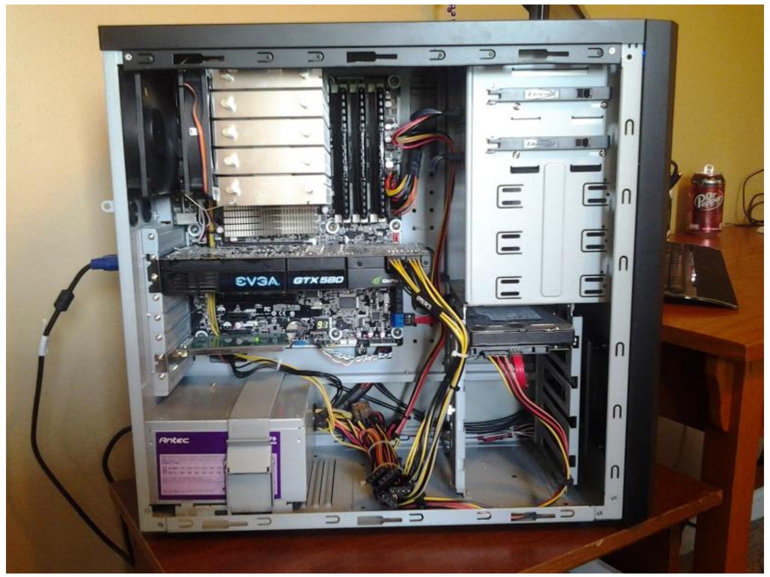
- Remember: computer here is a synonym for “computer system”
- **CPU**: Central Processing Unit
  - Also: processor; microprocessor
- **Memory**: two kinds
  - RAM (Random Access Memory)
  - ROM (Read Only Memory)
- **Storage** devices and media:
  - Hard disk
  - SSD, Flash storage
  - DVD, DVD-ROM, DVD-RW, etc.
- **Input** devices: keyboard, mouse, touchpad, button
- **Output** devices: screen, printer, network, motors
- **Communication** devices: network, infrared, Bluetooth, wifi

Computer Hardware



Key point: Devices are electrically connected using **multiple wires** known as a **bus**

Inside a computer (laptop, desktop)



Images: http://bit.ly/2EOMLMf, http://bit.ly/2qjQhuL

Modern PC architecture

- The confusing bit:
  - Some of the electrical connections are **within** chips
  - Some of the electronic connections are **between** chips
- Reminder:
  - **Input** and **output** are collectively known as I/O

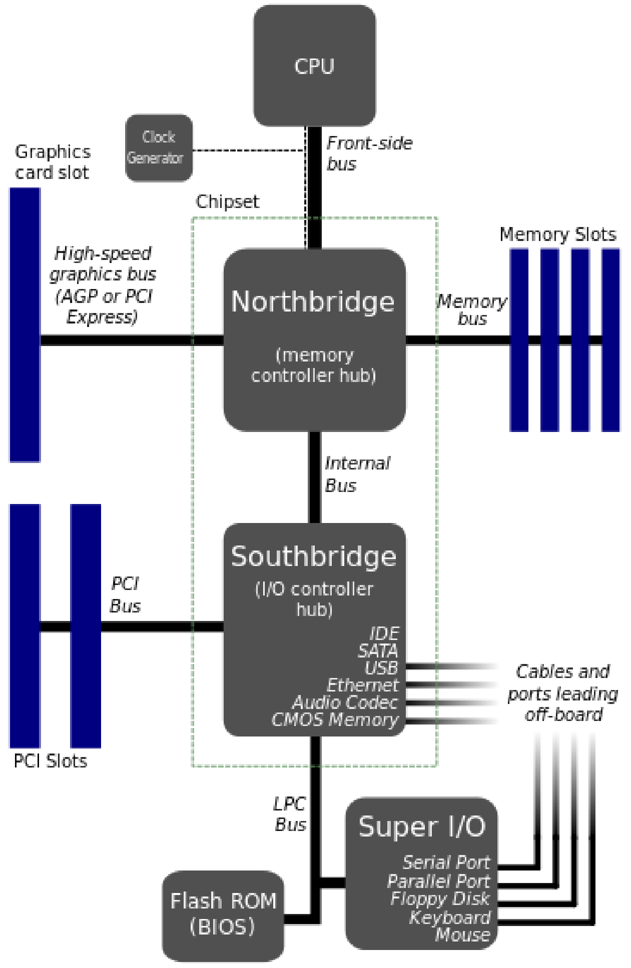


Diagram: http://bit.ly/2lUDFF1

Recent core-i7 board (2017)

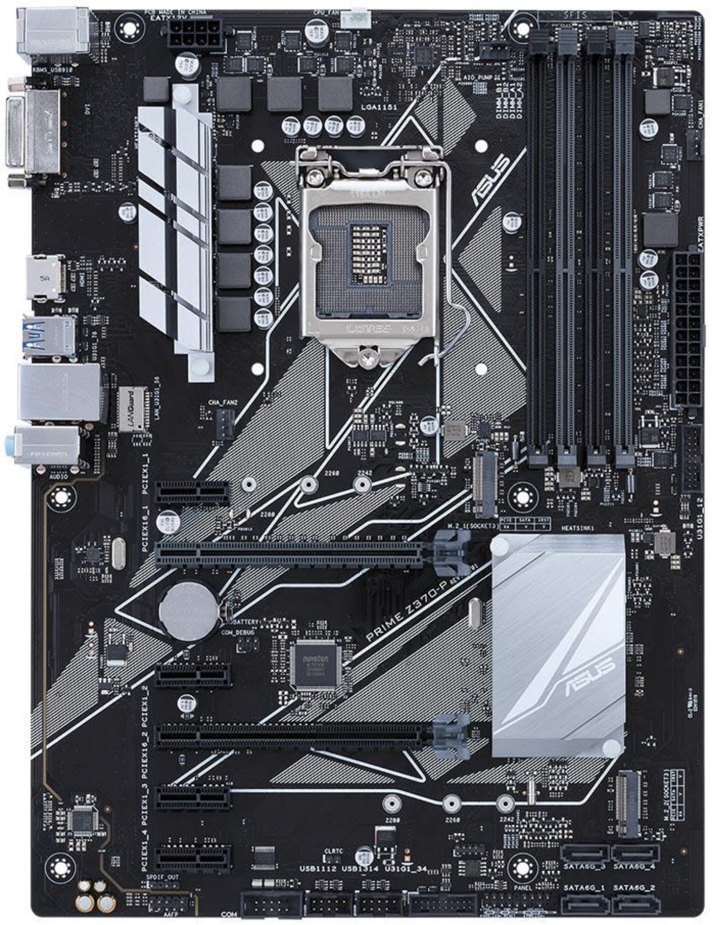


Image: http://bit.ly/2Aousu5

The good news...

- We will focus this semester on architectures less complex that current PC architectures
- Hands-on work will be with a simpler embedded-system architecture
- Note, however:
    - Complexity is usually in the service of some design aim chosen by the computer-system architectures
    - Knowing the design aims eventually helps us understand why specific decisions were made (that otherwise look complicated)

Active questioning

Imagine you are in charge of directing a group of five-year-olds to do several loads of laundry.

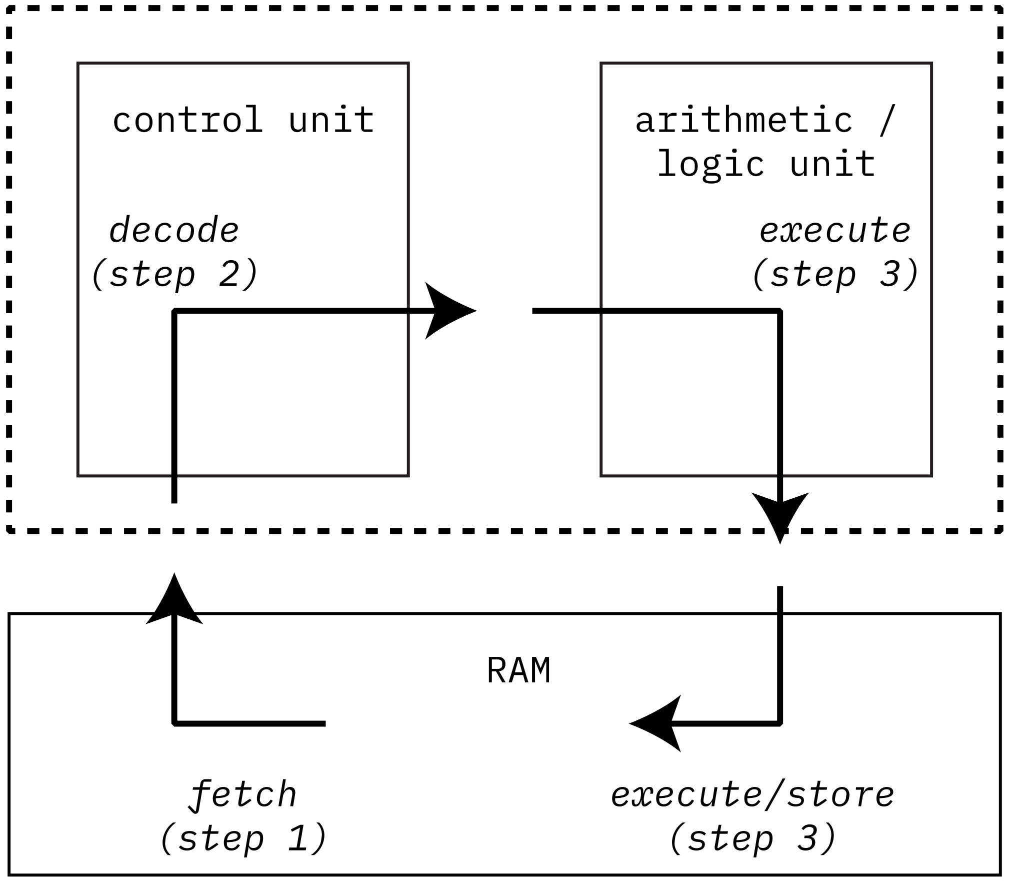
You are in the laundry room with the kids (and the washer and dryer), but only the kids are allowed to touch the machines.

What kind of instructions would you give them to do three loads?

Basic Processor Architecture

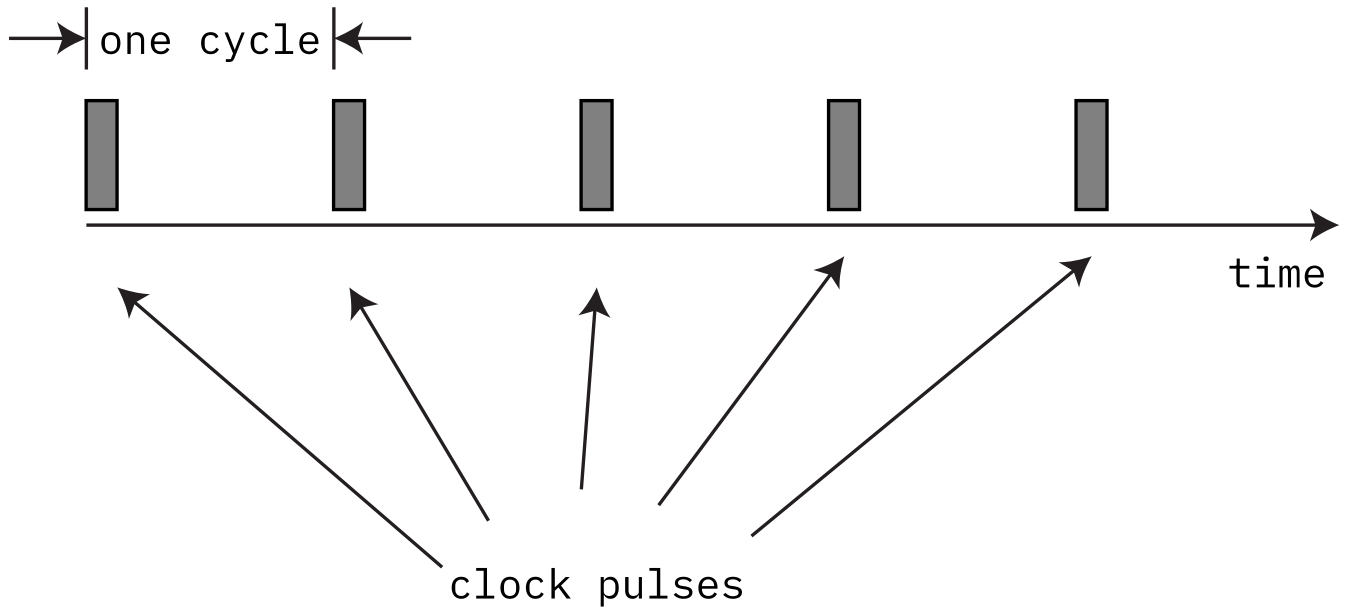
- The computers in which we are interested are also known as **stored-program processors**
- Every general purpose processor is designed on the **fetch-decode-execute** cycle
  - A machine instruction is **fetched** from program memory...
  - ... the instruction is then **decoded** to determine its meaning ...
  - ... and then the actions indicated through decoding are **executed**
- This cycle is driven by a series of **clock pulses** (timing signals)
  - Goes on forever (or until power is turned off)
  - Each pulse marks the start of a fetch-decode-execute cycle

CPU Execution Cycle



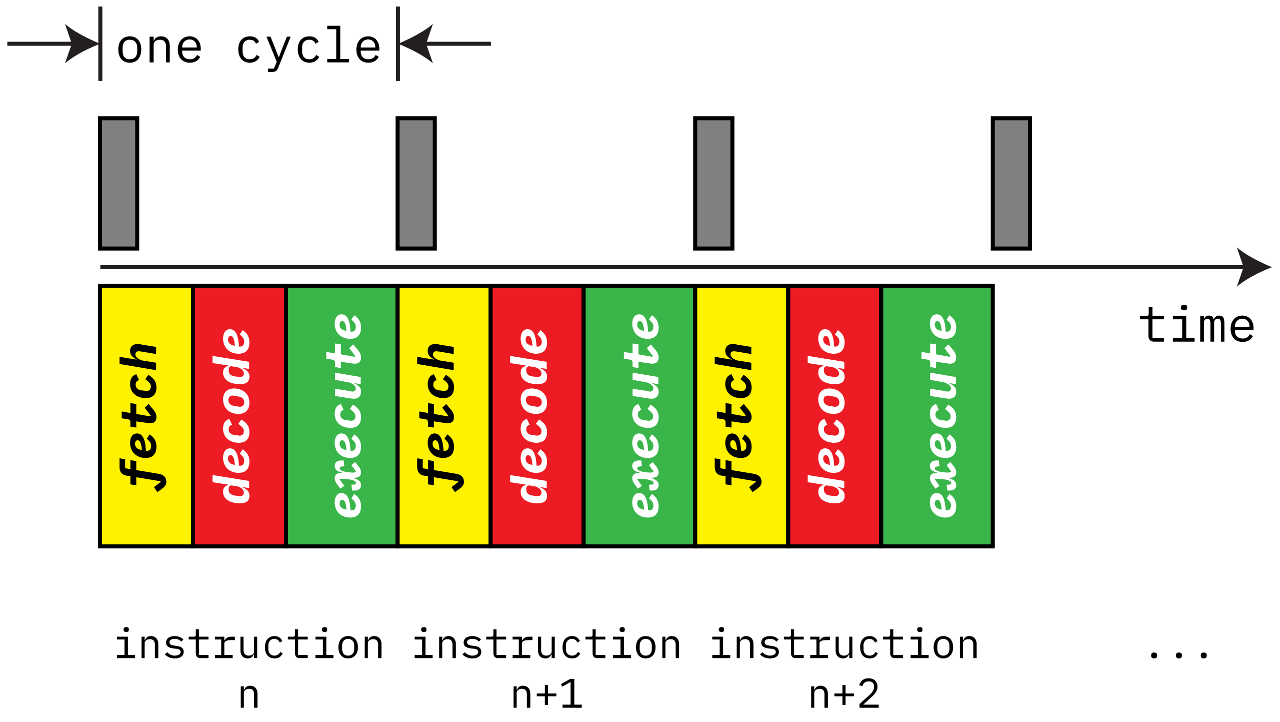
Timing (Clock)

- Nearly all computers depend upon some sort of **timing signal** or **clock**
  - Clock is a periodic timing signal whose period is called a cycle



Timing + fetch/decode/execute

- One architectural approach is to fit the fetch/decode/execute cycle in one clock cycle



Timing (Clock)

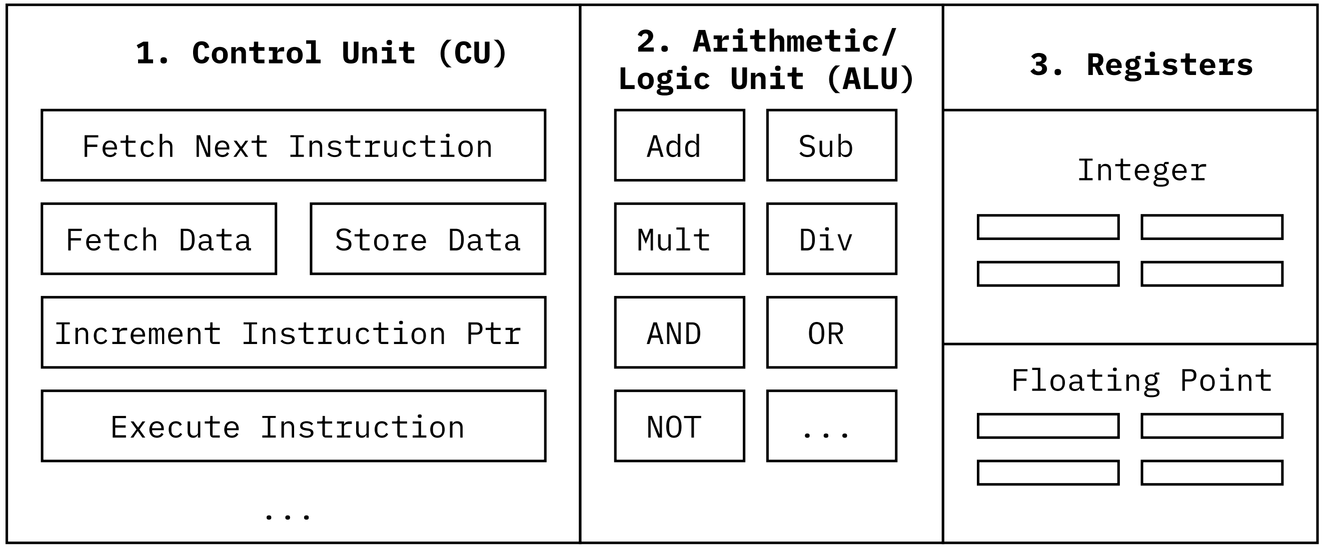
- The **numbers of cycles in a single second** is called the cycle’s **frequency**
  - Units are **Hertz** (Hz); in older literature may see **cps**
- Each processor operates at a certain frequency
  - Also known as the **processor speed**
  - 1000 Hz = 1 kilohertz = 10^3 Hz = 1 KHz
  - 1,000,000 cycles per second = 1 megahertz
    = 10^6 Hz 1 MHz
  - 1,000,000,000 cycles per second = 1 gigahertz
    = 10^9 Hz = 1 GHz

Central Processing Unit (CPU)

- This is the coordinator of every computer system
  - Sometimes called the “brain”, but that can be very misleading!
- **CPU** (or **processor**):
  - Responsible for **controlling** most aspects of computer system
  - Processes data in small units
  - Performs arithmetic and logical operations
  - Stores and retrieves data from memory
  - Communicates with and/or controls peripheral devices
- Each family of CPUs has its own **machine language**

CPU internals

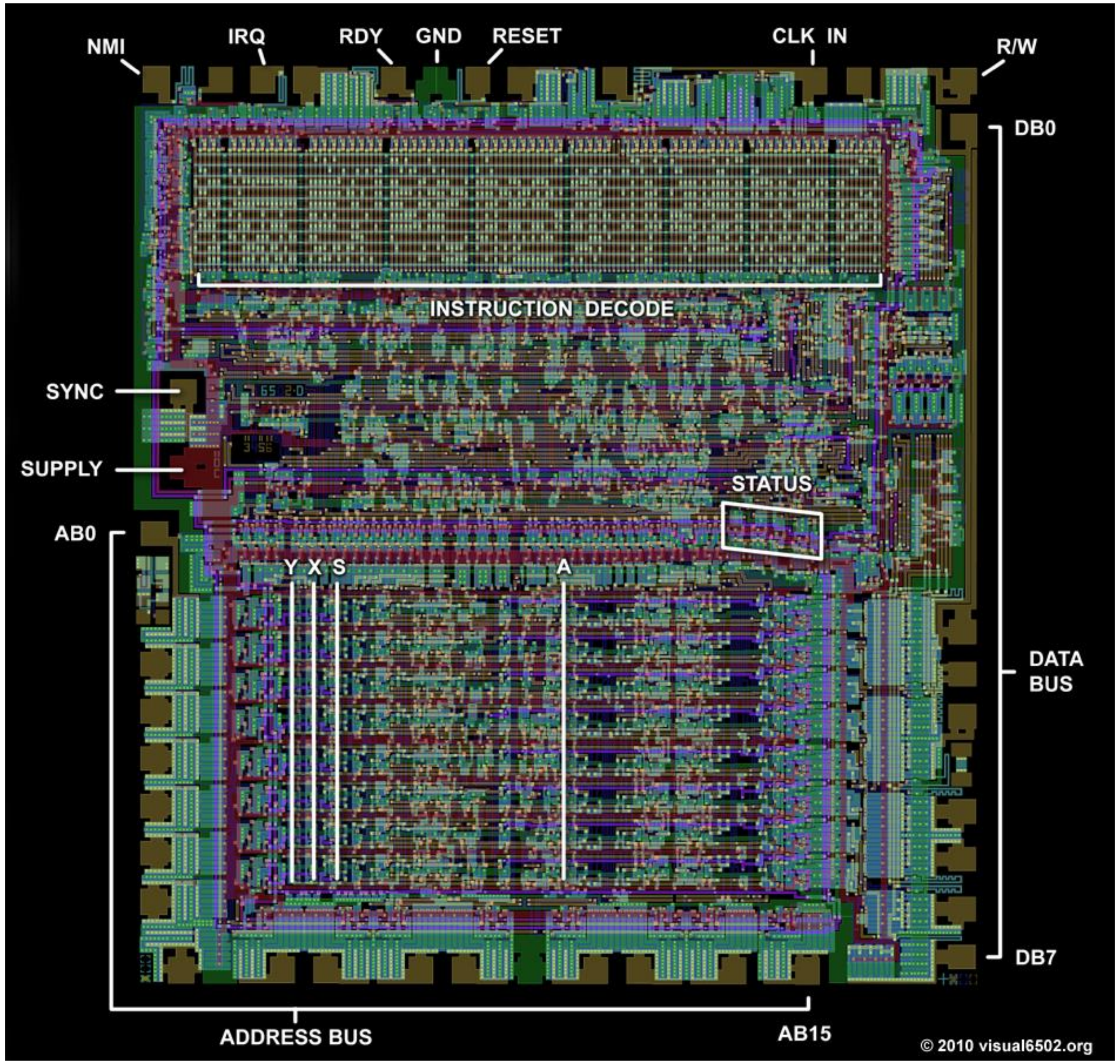
- CPU consists of three main parts:
  - Control unit (CU, the main coordinator)
  - Arithmetic Logic Unit (ALU)
  - Registers (very fast temporary memory)



MOS Technology 6502 (from 1976)

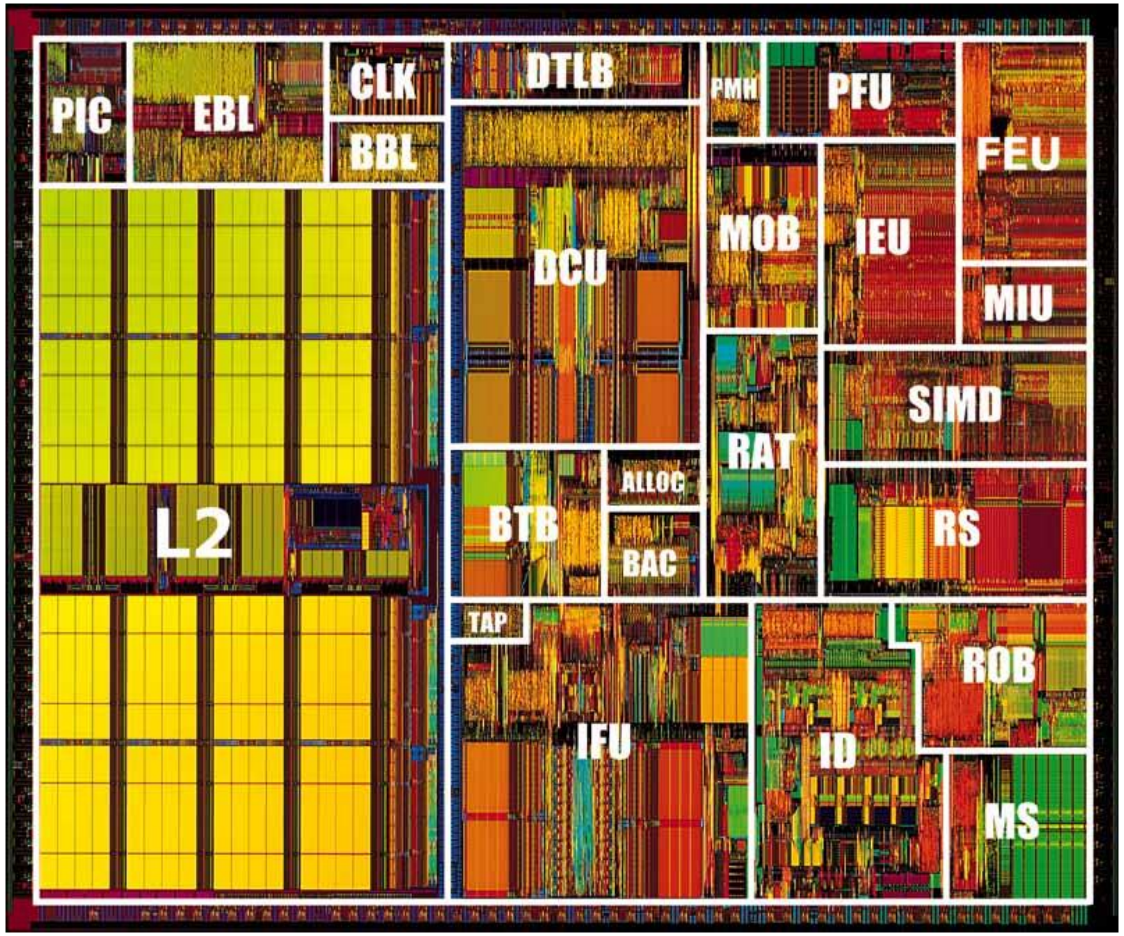
Outside: https://bit.ly/2Hhgj78

Inside: https://bit.ly/2WDNrwE



Pentium III (from 2003)

Outside: http://bit.ly/2Cr3dki



Inside (plus meanings of acronyms):
http://bit.ly/2lRINd4

1. Control Unit (CU)

- Decides what CPU components are to be used next
- Responsible for overall guidance on fetching, decoding, and executing instructions
  - In reality, it causes data to flow to the CPU components that can complete the operation
  - Instructs ALU to perform **add**, **multiply**, **AND**, etc.
  - Possibly branch (i.e., choose amongst several possible flows of control)
- Issues logic signals for routing data between registers and internal data pathways
  - Load data from main memory (RAM) into register
  - Store data from register into main memory
- In all of this, system clock synchronizes activities!

2. Arithmetic/Logic Unit

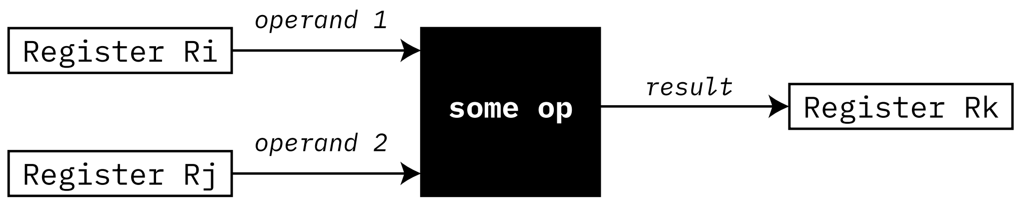
- Contains circuitry for integer, floating point, and logic operations
- Note CU is responsible for getting the data to and from the ALU
  - Depending on signals it receives, ALU performs actions
- Arithmetic op examples:
  - Add, subtract, multiply, divide, square root, cosine
- Logical op examples:
  - Compare two numbers to see which is greater
  - Check whether a number is equal to zero
  - Check whether a number is negative

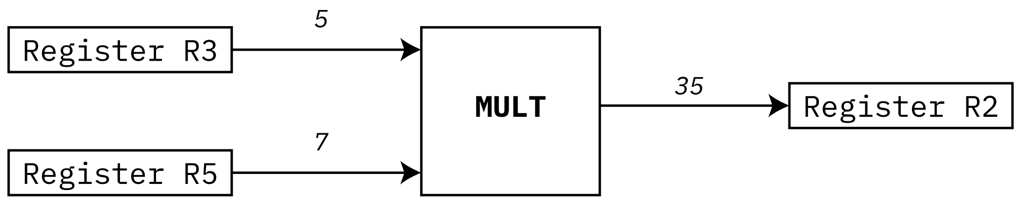
3. Registers

- Memory-like locations inside the CPU
- Designed for use with instructions currently being performed
- Hold operands used by current ALU operations
  - Also hold results of those operations
- Example:
  - CPU is adding two numbers
  - One operand is in some register, other operand is in a different register
  - Addition is performed, with result stored in (perhaps!) yet another register
- Registers are expensive (far fewer in number than normal memory, at most a few hundred).

How registers are used

- Every arithmetic or logical operation has one or more operands...
- ... And one result.
- Functional black-box performs operation
- Result always goes to a register





CPU execution cycle

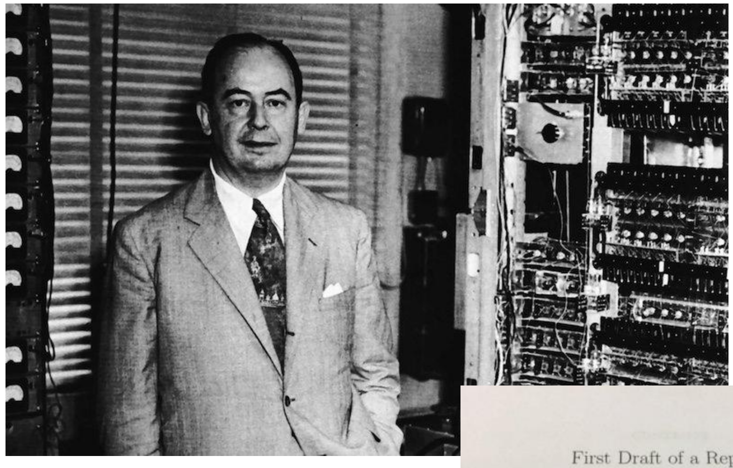
- Memory normally organized:
  - as a sequence of **bytes**, or ...
  - ... as a sequence **words** ...
  - ... but in either case, is accessed by an address.
- Processor keeps track of next instruction’s address
  - That is, the next address to be **fetched**
  - Address normally stored in the **program counter** (PC) or **instruction pointer** (IP)
- After fetch, PC / IP is updated to address of next instruction
  - Normally this update is in the form of an increment by some fixed amount
  - This update may be overwritten when control flow changes...

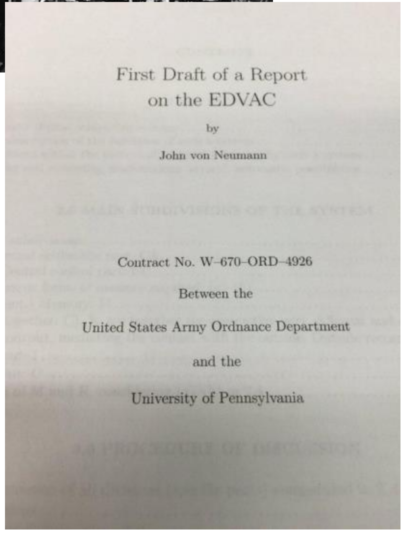
A bit more about memory

- Memory is used for at least two things:
  - contains code of running program
  - contains data (i.e., variables, constants) for running program
- **Von Neumann architecture**
  - Same memory is used for code and data
  - (also called a Princeton architecture)
- **Harvard architecture**
  - Code is in one memory system
  - Data is in a separate, different memory system

Von Neumann architecture

- Named after **John von Neumann** (1903 – 1957), a Hungarian-American mathematician.
- While doing consulting work for the University of Pennsylvania in the 1940s, Neumann proposed storing program instructions in same physical location as data to be processed.
- He wrote up his ideas in a very influential report.
- von Neumann was actually at the Institute for Advanced Studies at the time.
  - Hence “Princeton” Architecture
  - More commonly called the **Stored Program Computer**





First Draft of a Report
on the EDVAC
by
John von Neumann

Contract No. W–670–ORD–4926
Between the
United States Army Ordnance Department
and the
University of Pennsylvania

Images: http://bit.ly/2qVkAYX, http://bit.ly/2D0tjii

Harvard architecture

- Named after Harvard University who collaborated with IBM during World War 2 in order to create an electrical-mechanical computer
  - von Neumann worked on the programming team.
- By keep instructions apart from data ...
  - ... different pathways / buses were used for each ...
  - ... potentially increasing performance.
- Fetch and decode of one instruction happens simultaneously with execution of another instruction

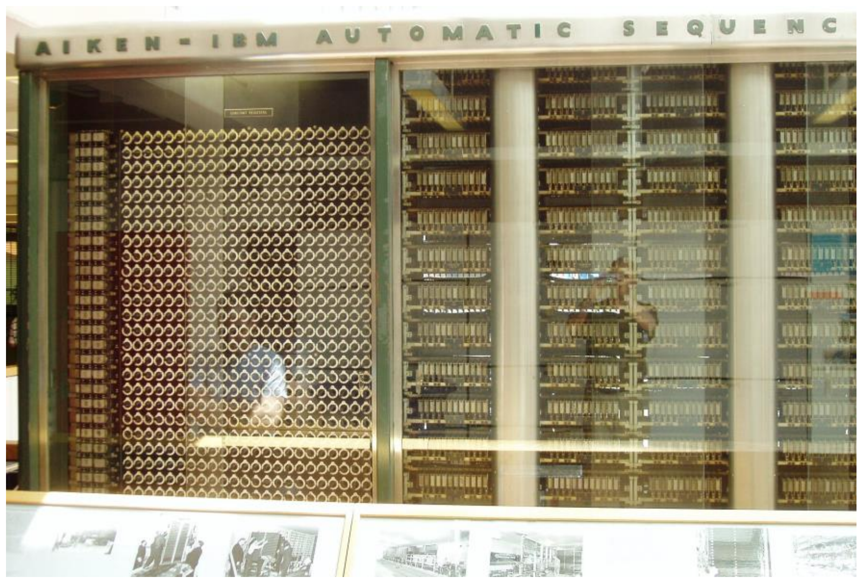
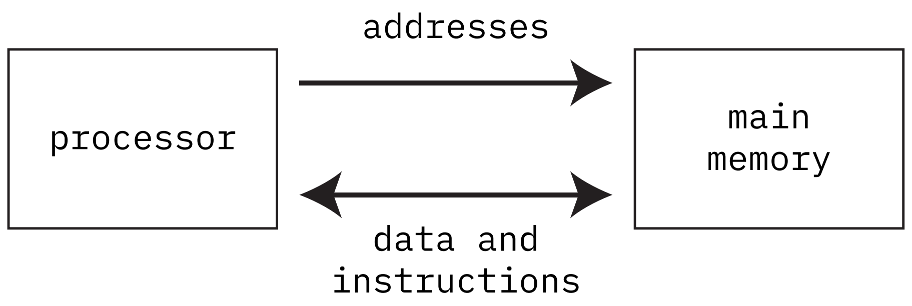


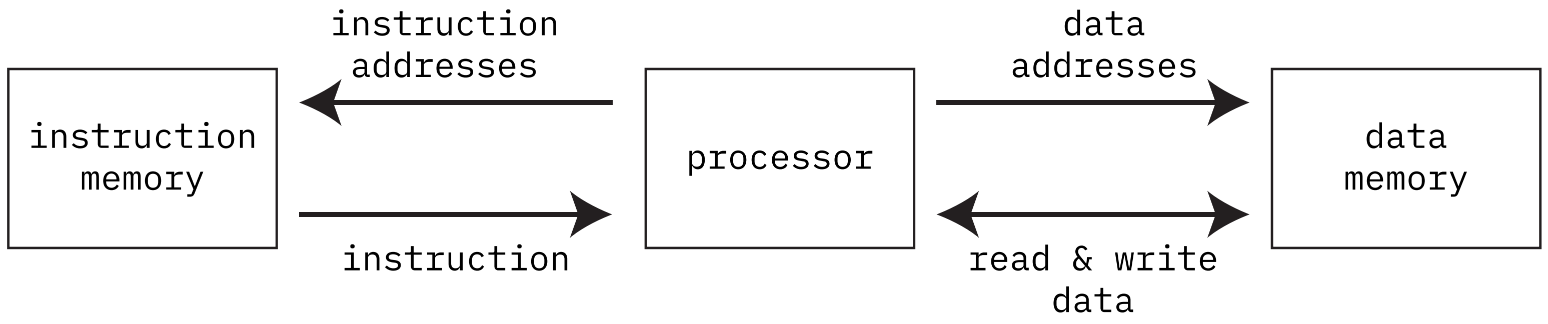
Image: http://bit.ly/2D2jLDA

Comparison

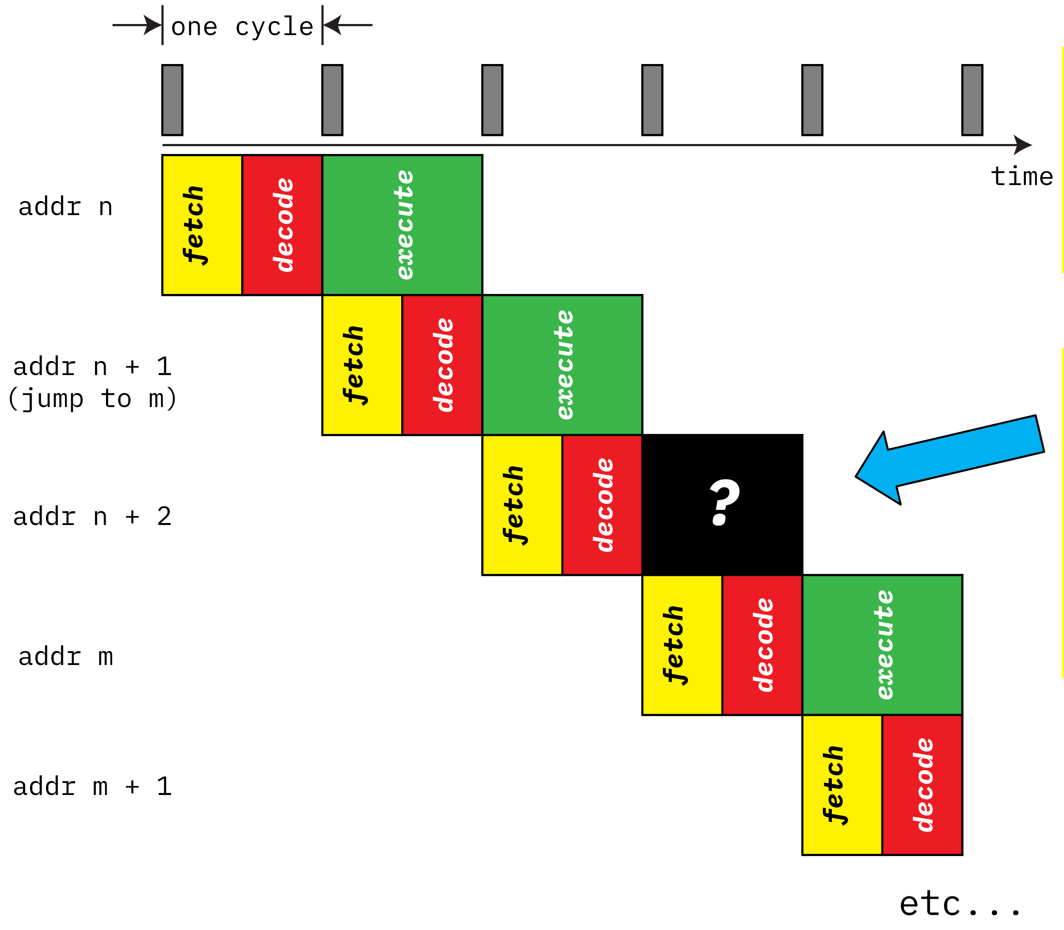
von Neumann / Princeton



Harvard



Fetch-decode-execute modification



In any one cycle, there may be one instruction being fetched & decoded, and one instruction being executed.

When control-flow of a program changes from strict sequencing (i.e., a branch, jump, etc.) then work for some fetch/decode is discarded.

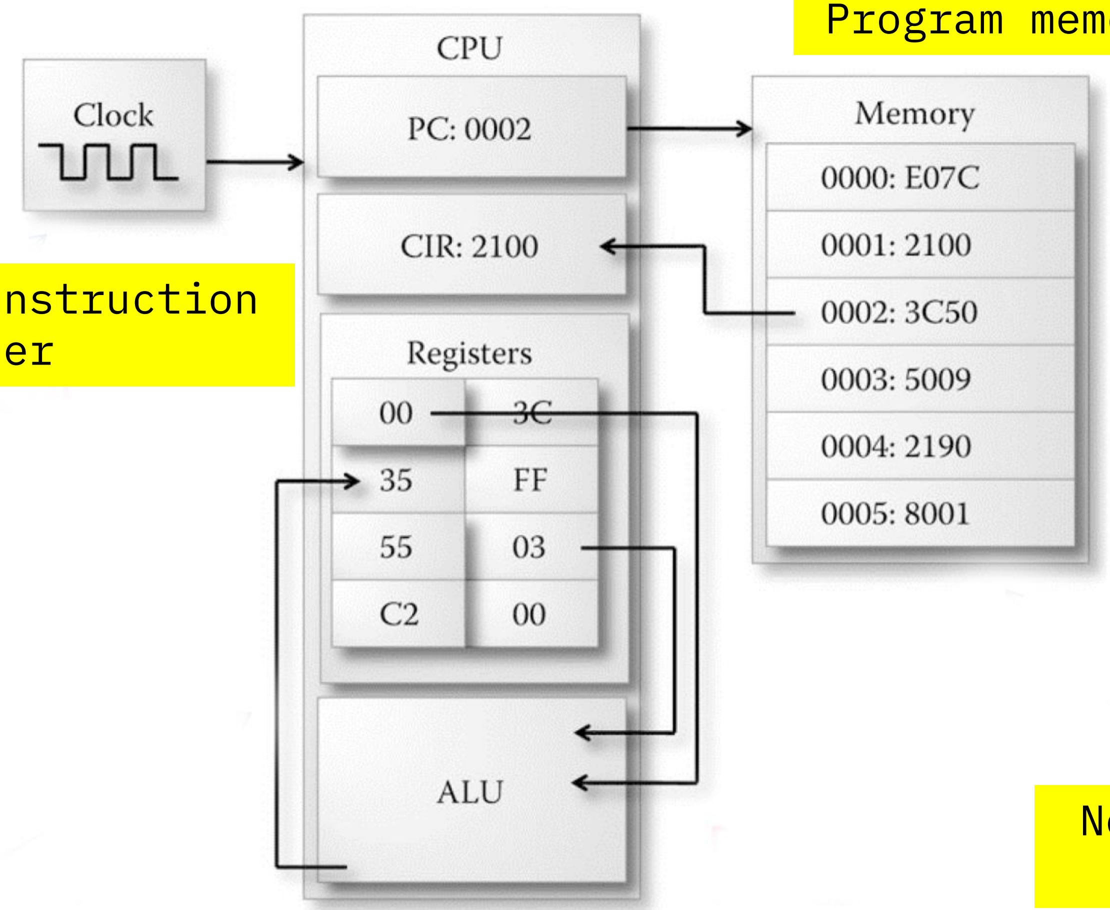
Basic Components (AVR)

PC: Program Counter

Program memory

Timing signal

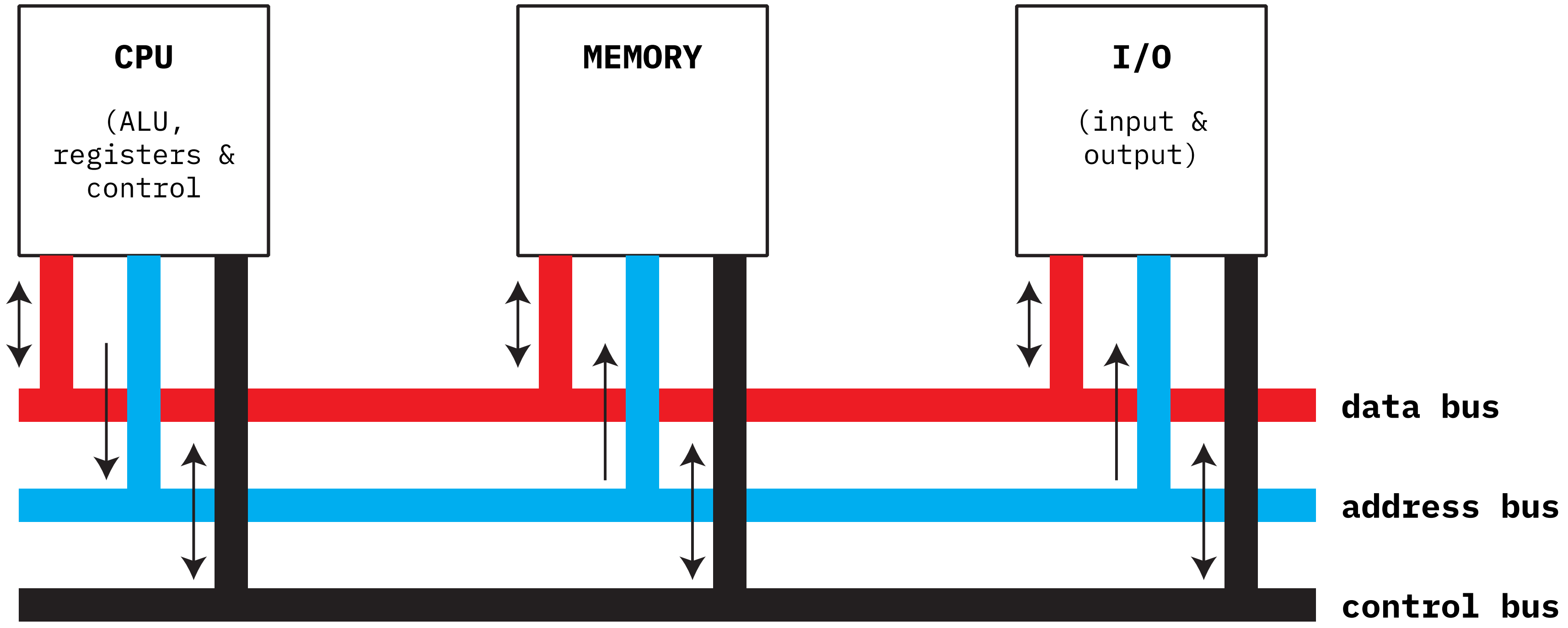
CIR: Current Instruction Register



Not shown here: Data memory

Processor

System Bus Model



For most of the course, we will look at architectural concepts from the point of view of the Harvard model.

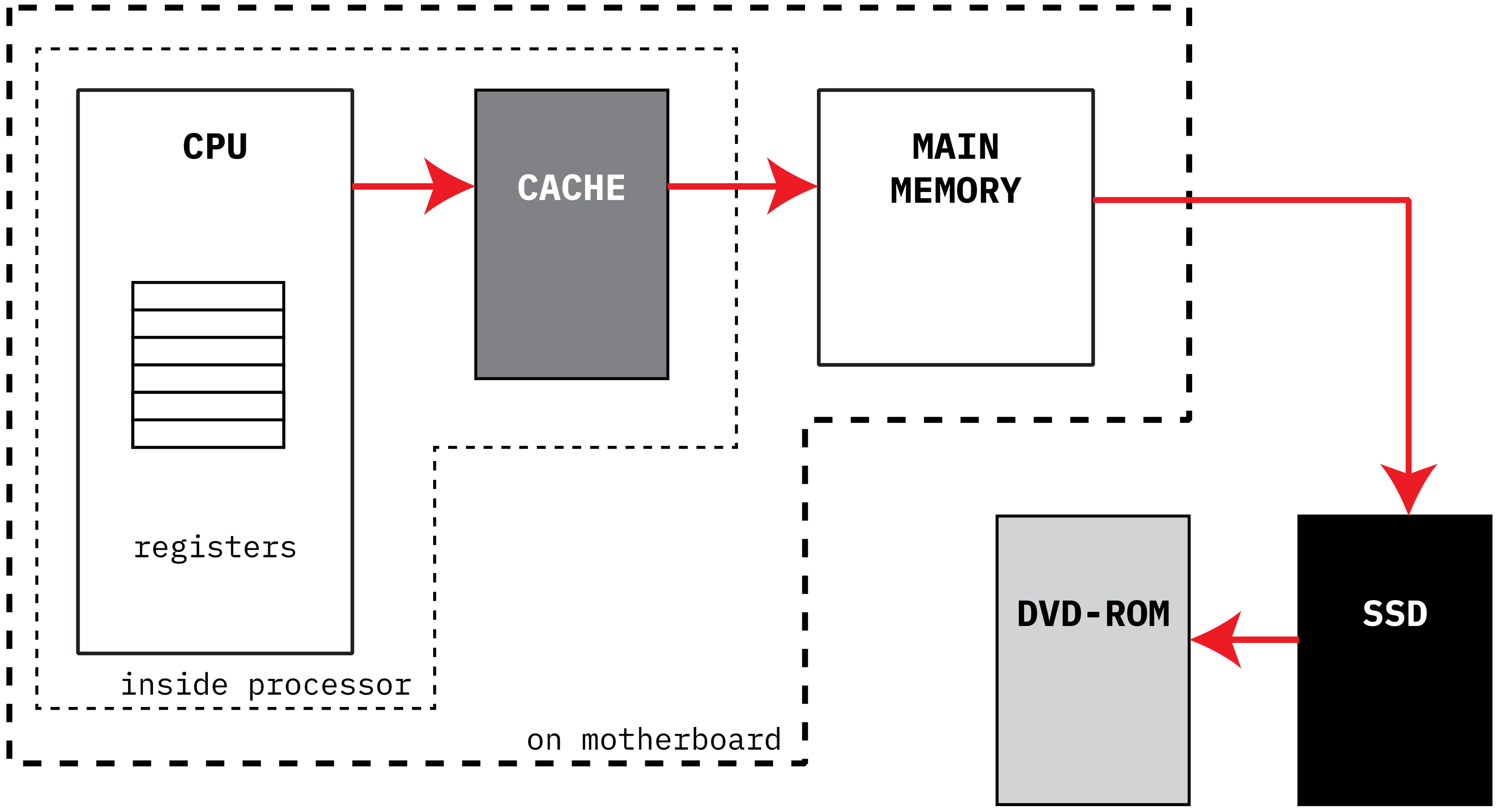
System Bus Functionality

- Bus is used as a “bit highway”
  - For transferring data, address, and control signals between devices (bus = large # of parallel “wires”)
- Example of use: CPU needs to READ data from another unit.
  1. CPU places **address** of needed data on **address bus**
  2. The requirement for a read is **signaled** by CPU on the **control lines**
  3. The correct device (memory? input device?) **responds** to what it sees on address bus and control lines
  4. Device **places data onto data bus**
  5. CPU has been **listening** to the **data bus** and control lines; when it recognizes data is present, it **pulls it off the data bus** (and perhaps into a **register**)

Storage / Memory Hierarchy

- Memory hierarchy refers to the levels of fast and slow speed memories use in PC architectures
  - System cleverly moves data to and from levels
  - This is done to improve both runtime performance and system flexibility
  - Two main categories of storage
- **Primary** (fast memory, but small in size, and expensive)
  - registers
  - cache
  - main memory
- **Secondary** (slower memory, very large capacity, much less expensive)
  - hard disk, SSD, memory keys
  - other kinds of removable media (e.g., DVD-ROM)

Red arrows: towards slower memory



Registers

- These are inside the CPU
- Architecture usually determines:
  - the number of general-purpose registers
  - how large each register is (i.e., number of bits stored in a register)
  - the manner in which a register can be used by an instruction
- Our processor for the course has 32 **general-purpose** (GP) registers
  - each is 8-bits large (i.e., each GP register stores a byte)

Primary Storage

- This is where data and instructions reside when used by a currently-executing program
- Typically **volatile**
  - That is: contents of primary storage disappears when computer is powered off
- Normally two categories of primary storage.
- In this course will eventually examine:
  - cache (L1, L2)
  - main memory (random-access memory, or RAM)

An aside: ROM vs RAM

- ROM: Read-Only Memory
  - Non-volatile
  - CPU can look at any location, and get the same memory speed, but cannot modify
- RAM:
  - CPU can look and change arbitrary values (i.e., data at addresses) and in any sequence without a time penalty
  - Quite often **RAM**, **main memory**, and **memory** are used as synonyms

Secondary storage

- Normally where data and instructions are stored outside of primary memory
- Important: This memory is also **non-volatile**
- Much, much slower than RAM...
    - ... but therefore also much cheaper, and with a higher memory capacity
- Other than some SSDs and installed hard drives, this storage is separate from the computer itself, perhaps even portable

I/O: input/output devices

- We normally say I/O as shorthand for “input/output”
- Input devices:
  - transfer data from some device into computer
  - keyboard, mouse, scanner, microphone, touchpad, joystick
- Output devices:
  - transfer data out of computer
  - monitor, printer, speakers
- Note:
  - Although strictly speaking with perform I/O when using secondary storage, we usually distinguish such devices from more general I/O.

Summary

- We are interested here in computer architectures...
- ... and the way the fit into larger computer systems
- Central to a computer is the CPU and what it encompasses
  - ALU
  - Registers
  - Control data
- CPU’s operation organized around the the fetch-decode-execute cycle
  - Each clock tick starts a cycle
- Larger computer system access via the system bus

Any Questions?