CSC 3210

Computer Organization and Programming

Chapter 2: x86 Processor Architecture

Dr. Zulkar Nine

mnine@gsu.edu

Georgia State University

Spring 2021

X86 Processor Architecture

- One step **before using assembly language**
  - What is the selected processor **Internal architecture and capabilities**.
- What **is the underline hardware** associated with X86?
- Assembly language is **a great tool** for learning **how a computer works**.
  - It require you to have working knowledge of **computer hardware**

You should have some basic knowledge about **the processor** and the **system architecture** in order to effectively program in **the assembly language**.

Outline

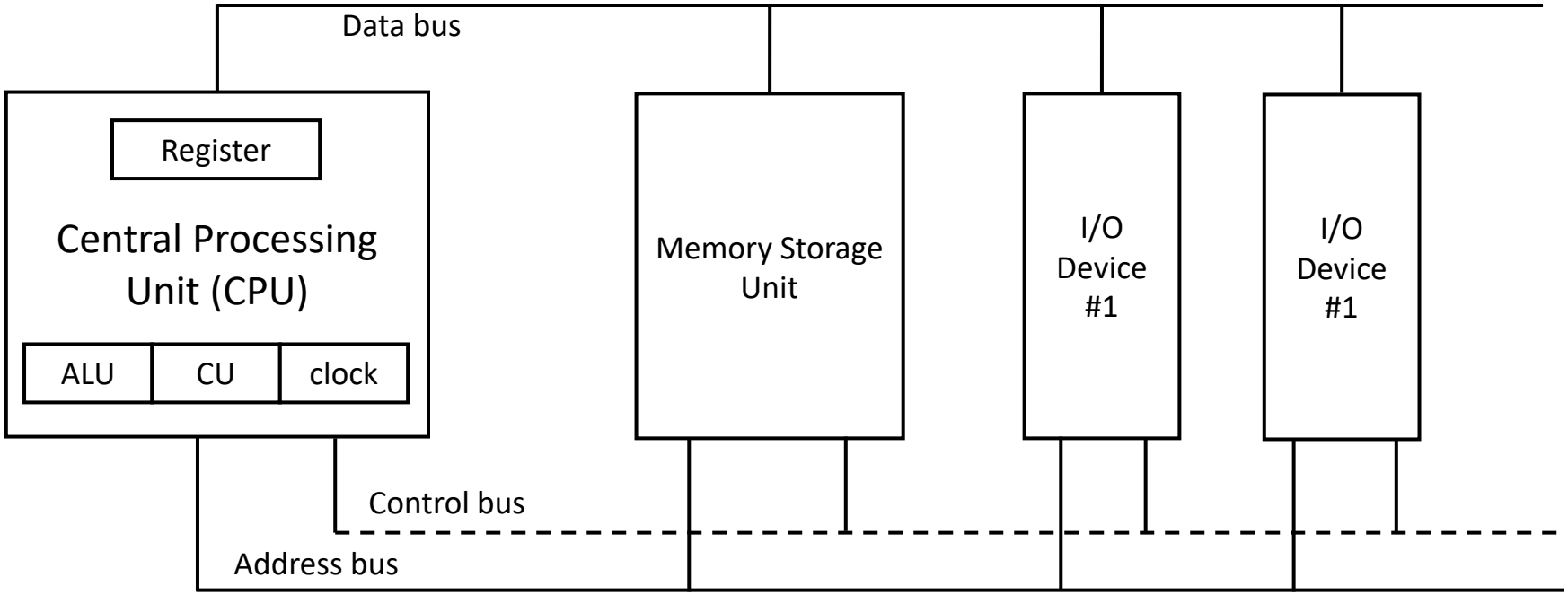
- **General Concepts**
- IA-32 Processor Architecture
- IA-32 Memory Management
- 64-bit Processors
- Components of an IA-32 Microcomputer
- Input-Output System

General Concepts

- **Basic microcomputer design**
- Instruction execution cycle
- Reading from memory
- How programs run

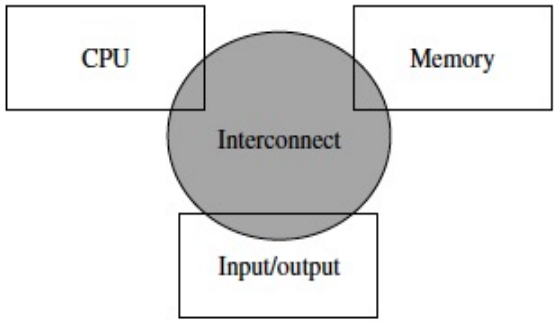
General Concepts: Basic Microcomputer Design

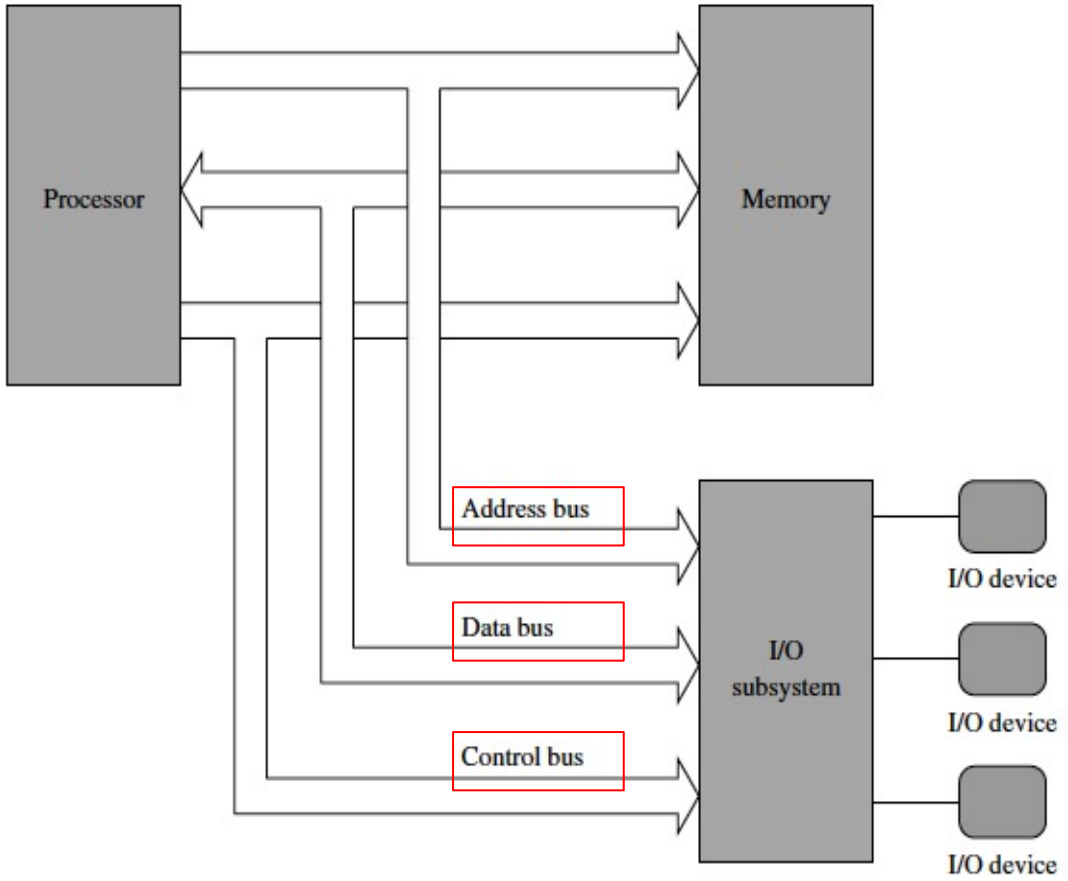
- **ALU** performs **arithmetic** and **logical** (bitwise) operations
- **Control unit (CU)** **coordinates** sequence of **execution steps**
- **Clock** synchronizes CPU **operations** with other **system components**



General Concepts: Basic Microcomputer Design

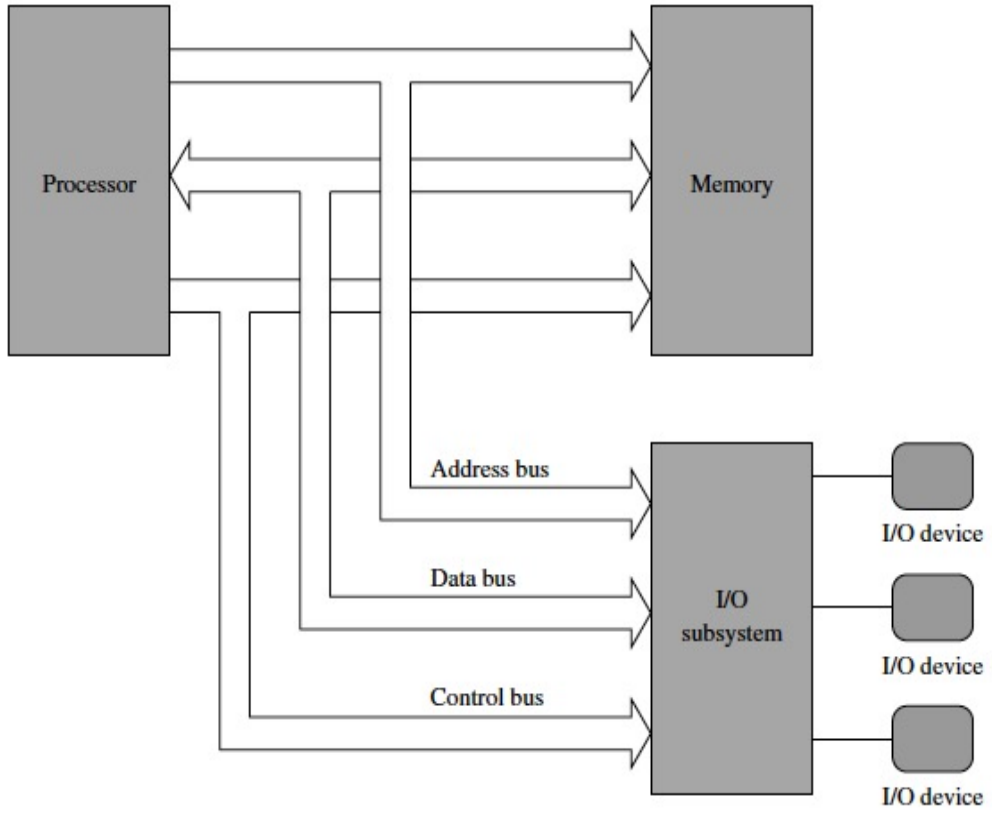
- A **bus**: a group of parallel wires that **transfer data**
  - bus types:
    - address
    - data
    - control





General Concepts: Basic Microcomputer Design

- The **Address bus** **holds the addresses** of instructions and data, when the currently executing instruction transfers data between the CPU and memory.
- The **Data bus** **transfers** **instructions** and **data** between the CPU and memory.
- The **Control bus** uses binary signals to **synchronize actions** of all devices attached to the system bus.

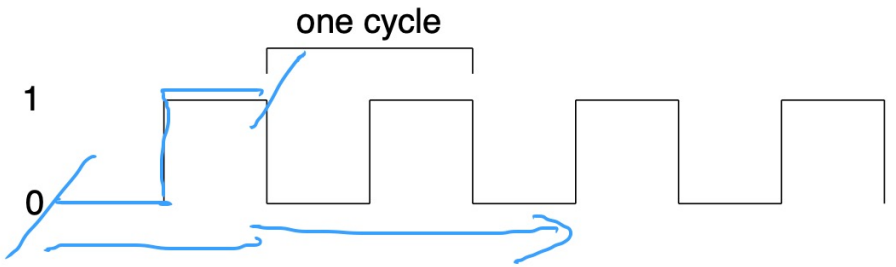


General Concepts: Clock

2.1 GHz

- The system clock provides **a timing signal** to synchronize the **operations** of the system.
  - Synchronizes all CPU and BUS **operations**
- A **clock** is a sequence of **1's** and **0's**

2.1 × 10^9 cycles per second



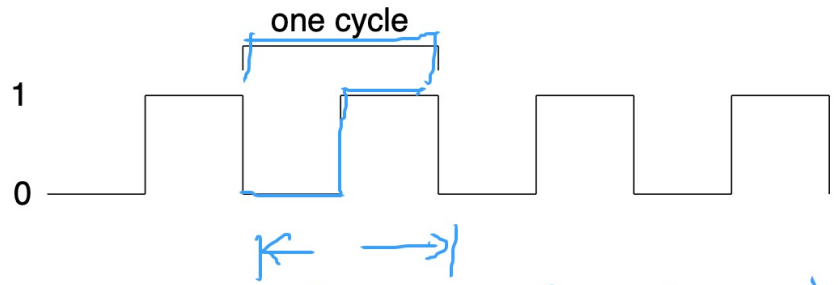
The frequency:
is the number of cycles that happens each second

1 cycle per second = 1 Hz.

8

General Concepts: Clock

- The clock frequency is measured in the number of cycles per second.
- This number is referred to as **Hertz** (Hz: the unit of frequency, defined as one cycle per second).
  - **MHz** and **GHz** represent 10^6 and 10^9 cycles per second
- The **system clock** defines **the speed** at which the system operates.



clock period = time to do 1 cycle.

= 1 / clock frequency.

3.1 GHz.

3.1×10^9 cycle — 1 sec

1 " — $\frac{1}{3.1 \times 10^9}$ sec

= 3 ns

General Concepts: Clock

- **Ex: transfer of data** from a memory location to X86 (Pentium) takes **three clock cycles**.
- The **clock period** is defined as the length of time taken by one clock cycle .

$$\text{Clock period} = \frac{1}{\text{Clock frequency}}$$

For example, a clock frequency of 1 GHz yields a clock period of

$$\frac{1}{1 \times 10^9} = 1 \text{ ns}$$

- If it takes three clock cycles to execute an instruction, it takes 3 X 1 ns = 3 ns.
- Machine **(clock) cycle** measures time of a single operation
- Clock is used to trigger events

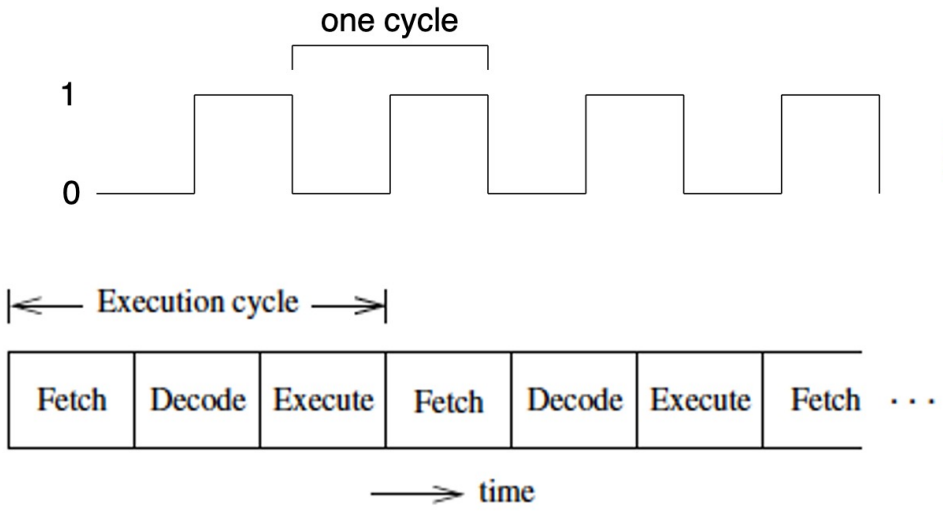
General Concepts: Clock

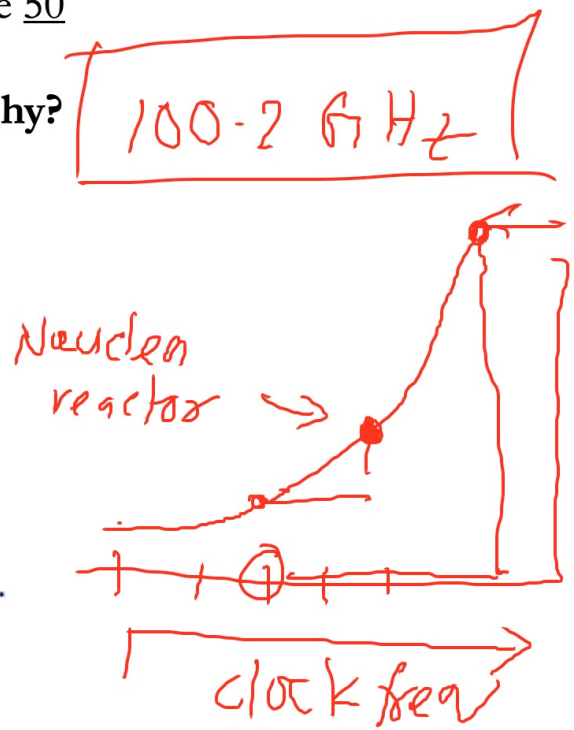
2.1 GHz

10.2 GHz

- A **machine instruction** requires one clock cycle to execute, few require 50 clocks
- Instructions require memory access: Empty clock cycle, **wait states, Why?**
  - CPU, system bus, and memory circuits

100.2 GHz





12

Clock per Instruction (CPI)

- Is an effective average.
- It is the average number of clocks required by the instructions in a program.
- In a program 60% instructions takes 4 clock cycles and the rest of the instructions takes 1 clock cycles.
- CPI = 0.6 * 4 + 0.4 * 1 = 2.8 clocks per instruction.

Million Instructions Per Second

- **Step 1:** Perform Divide operation between no. of instructions and Execution time.
- **Step 2:** Perform Divide operation between that variable and 1 million for finding millions of instructions per second.
- For example,
  - if a computer completed 2 million instructions in 0.10 seconds
  - 2 million/0.10 = 20 million.
  - No of MISP=20 million/1 million
  - =20

An Example

- An instruction on average takes 4 clock cycles to execute. A program with these instructions take 5 seconds to run on a 1.2 GHz processor. How many instructions the program have?

1.5×10^9 instructions

1.2 GHz.

1 second $\longrightarrow$ 1.2×10^9 cycles

5 seconds $\longrightarrow$ $5 \times 1.2 \times 10^9$ cycles

4 cycles $\longrightarrow$ 1 instruction

$5 \times 1.2 \times 10^9$ cycle $\longrightarrow$ $\frac{5 \times 1.2 \times 10^9}{4}$

=

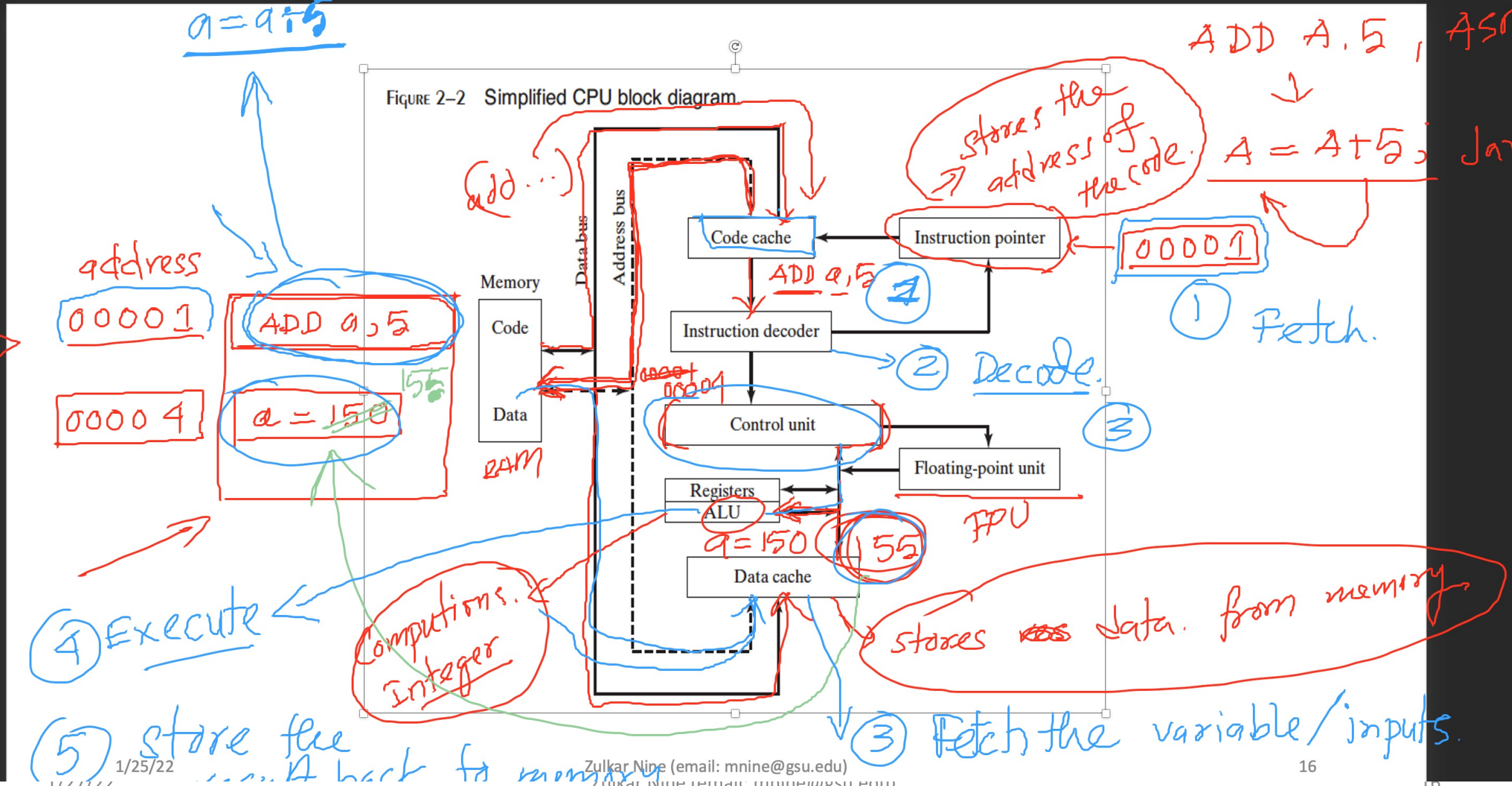


FIGURE 2–2 Simplified CPU block diagram
Memory
Code
Data
Data bus
Address bus
Code cache
Instruction pointer
Instruction decoder
Control unit
Floating-point unit
Registers
ALU
Data cache
address
00001
00004
ADD a,5
a = 150
ADD A,5
A = A+5;
ASM
Java
Fetch.
Decode.
FPU
Execute
Computations. Integer
stores data. from memory
Fetch the variable/inputs.

General Concepts

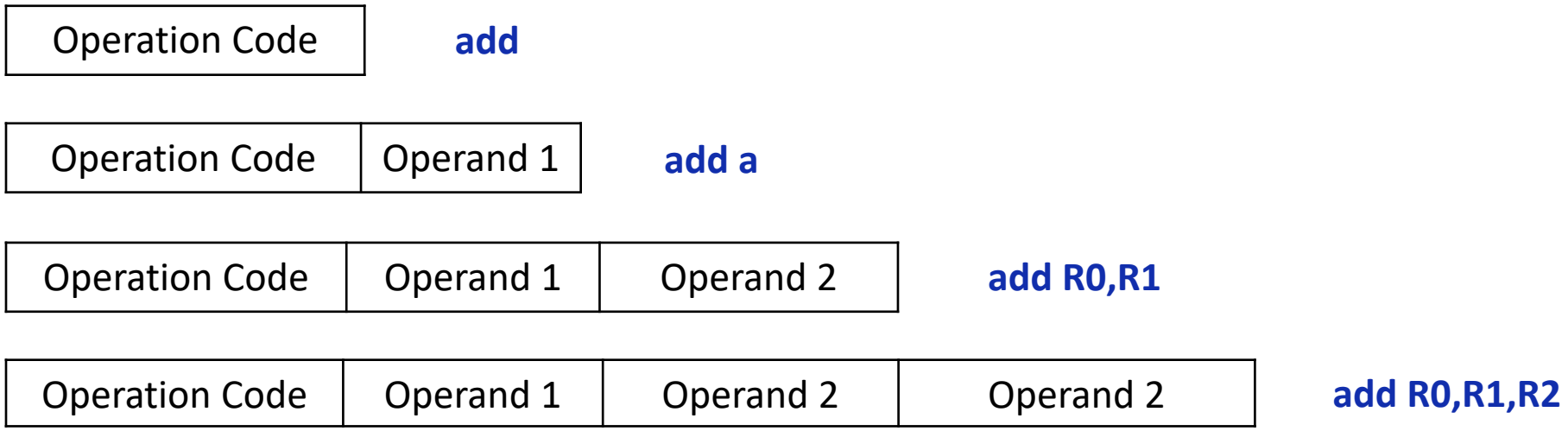
- Basic microcomputer design
- **Instruction execution cycle**
- Reading from memory
- How programs run

Instruction Execution Cycle

- An instruction is a **binary pattern designed inside a microprocessor** to perform a specific function.
- The entire group of instructions that a microprocessor supports is called **Instruction Set**.
- 8086 has more than **20,000** instructions.
- **Classification of Instruction Set**
  - Data Transfer Instructions: mov, push, pop,…
  - Arithmetic Instructions: add, sub, inc …
  - Bit Manipulation Instructions: and, or, xor, ….
  - Program Execution Transfer Instructions: jmp, call, ret, ….
  - String Instructions: cmps, movs, rep, …
  - Processor Control Instructions: stc, clc, wait…

Instruction Execution Cycle

- **Instruction format**
  - An instruction consists of an **opcode**, usually with some additional information like where operands come from, and where results go.

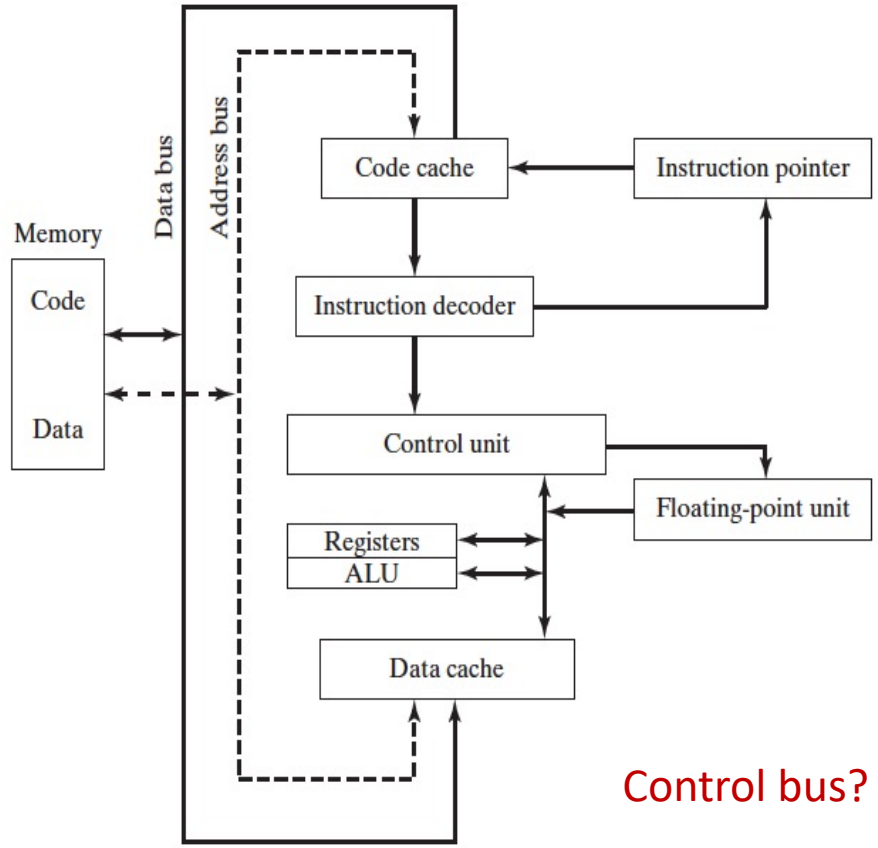


  - An **operand** can be register, memory location or immediate (ex. mov 5,R0) operand

Instruction Execution Cycle

- Predefined sequence of steps to execute a machine instruction
- Simple IEC: **Fetch**, **Decode**, **Execute**
  - **Fetch**
  - **Decode**
  - **Fetch operands** (not always needed?)
    - Address calculation?
  - **Execute**
    - Update few status flags: zero, carry, overflow
  - **Store output** (not always needed?)

FIGURE 2–2 Simplified CPU block diagram.



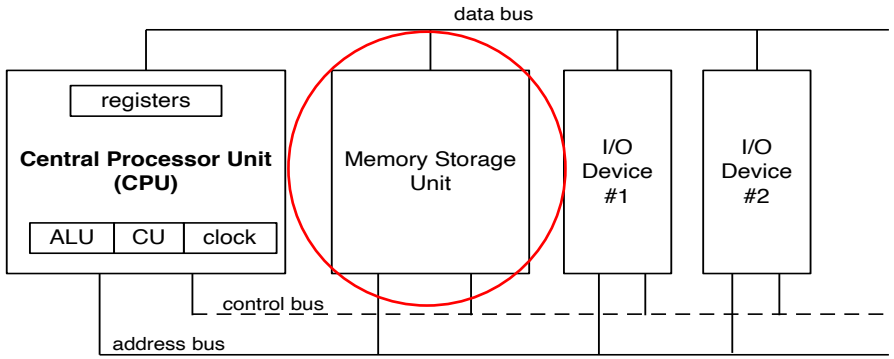
Control bus?

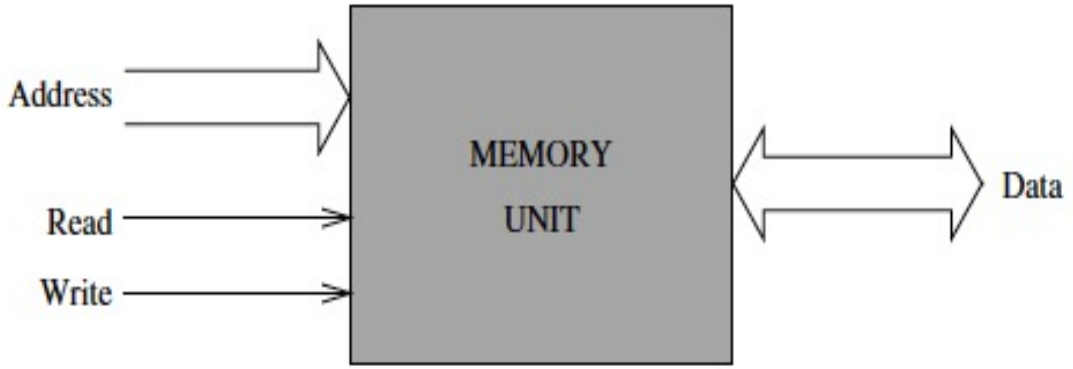
General Concepts

- Basic microcomputer design
- Instruction execution cycle
- **Reading from memory**
- How programs run

Memory

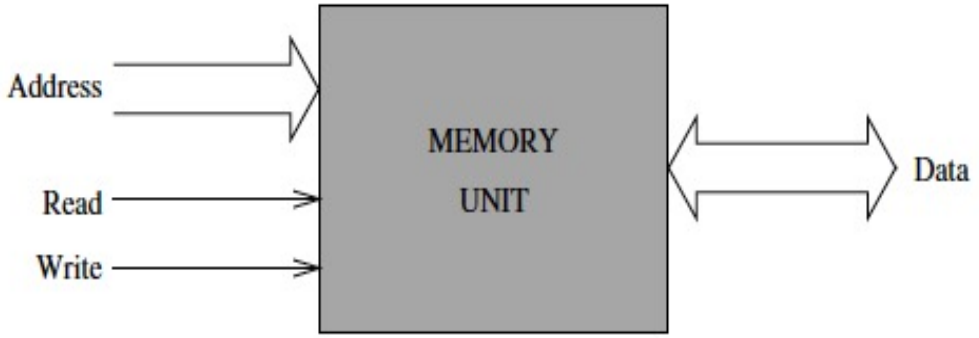
- The memory unit supports <u>two fundamental operations</u>: **read and write** .
  - The **read operation** reads a previously stored data
  - The **write operation** stores a value in memory.



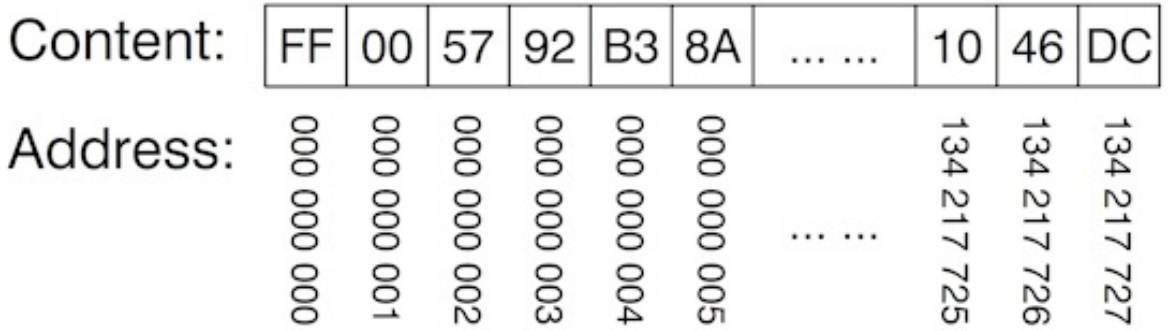


- Both of these operations require an **address** in memory from which to read a value or to which to write a value.

Memory

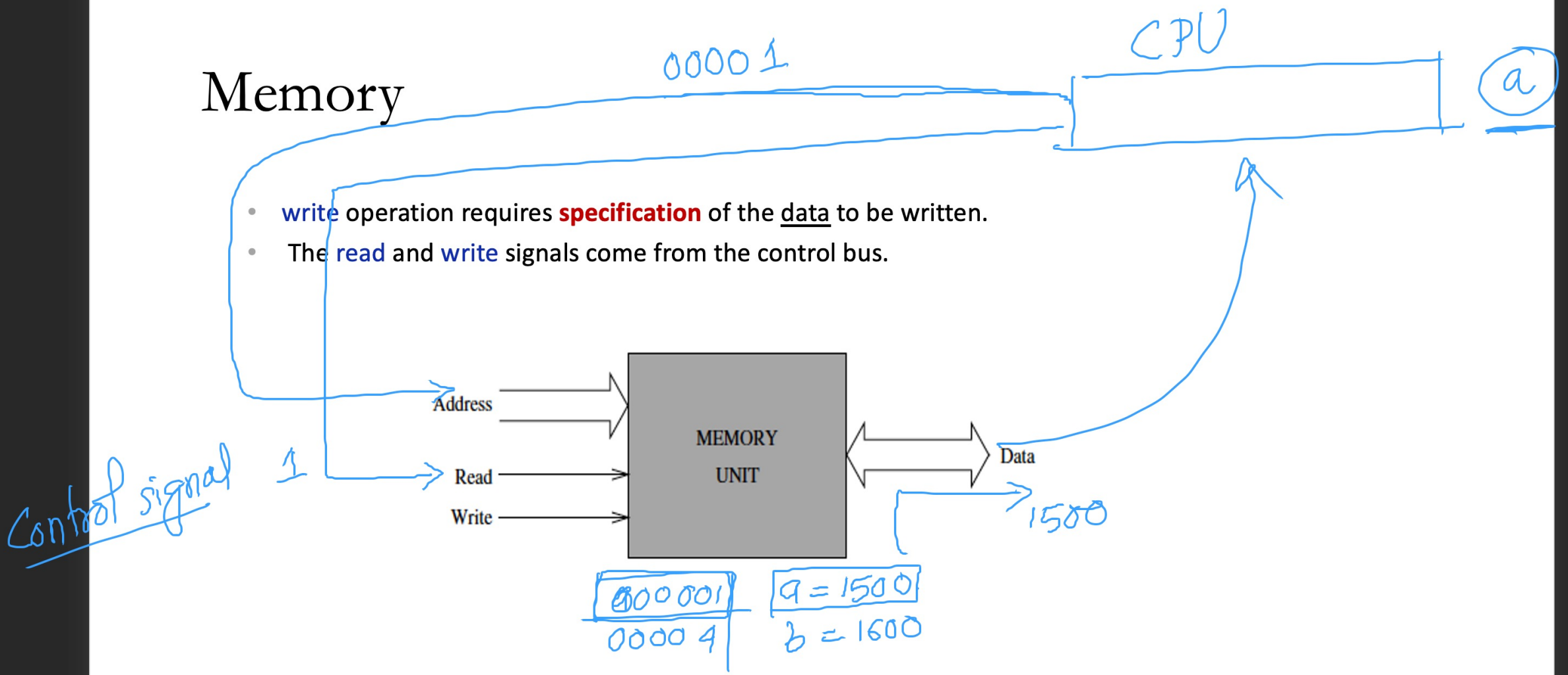


Memory (RAM) as an array of bytes



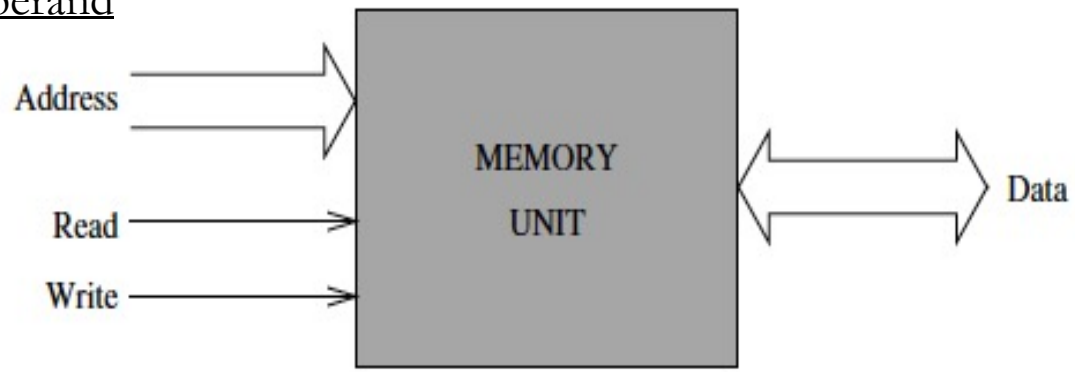
Memory

- write operation requires **specification** of the data to be written.
- The read and write signals come from the control bus.



Memory: **Reading from Memory**

- **Multiple machine cycle** are required when **reading** from memory, Why?
  - Because **it** responds much more slowly than the CPU.
- Steps in a typical **read cycle**
  1. **Place** the address of the value you want to read on the address bus.
  2. **Assert** (changing the value of) the processor's **RD** (read) pin.
  3. **Wait** one clock cycle for the memory chips to respond.
  4. **Copy** the data from the data bus into the destination operand



Address ↑ where to write

Memory: **Writing to Memory**

CPU

address of

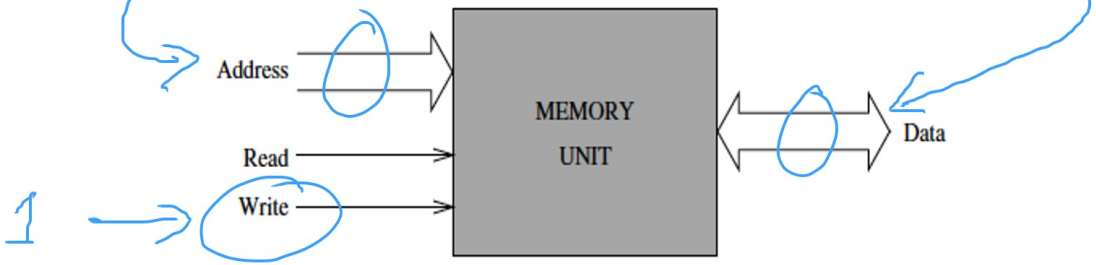
What to write.

→ Data.

- Steps in a typical **write cycle:**

1. **Place** the address of the location to be written on the address bus,
2. **Place** the data to be written on the data bus,
3. **Assert** (changing the value of) the processor's **WR** (write) pin.
4. **Wait** for the memory to store the data at the addressed location

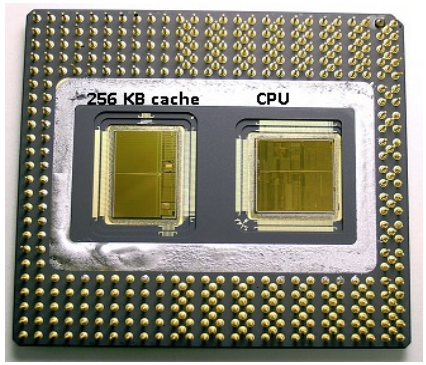
1600 → 001001
Data Address

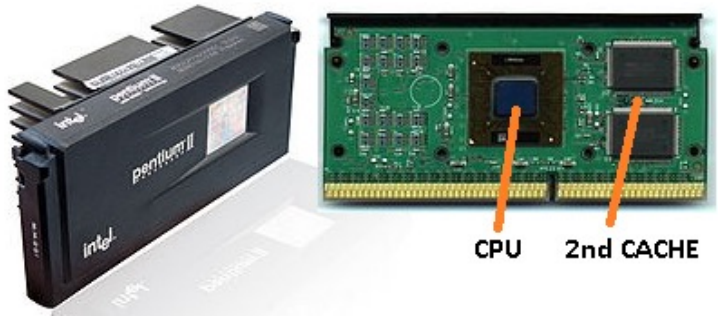


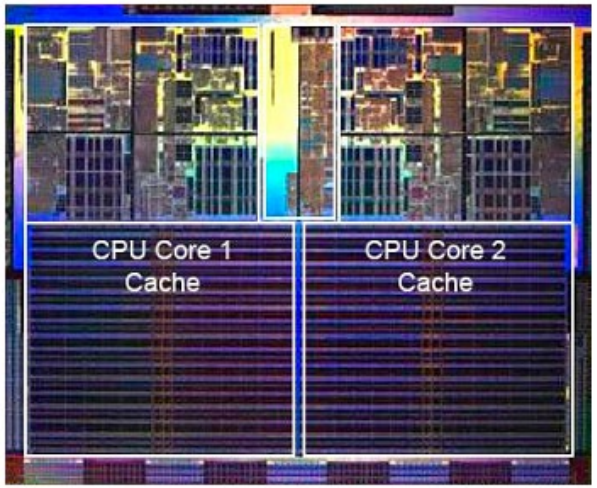
001001 → 1600

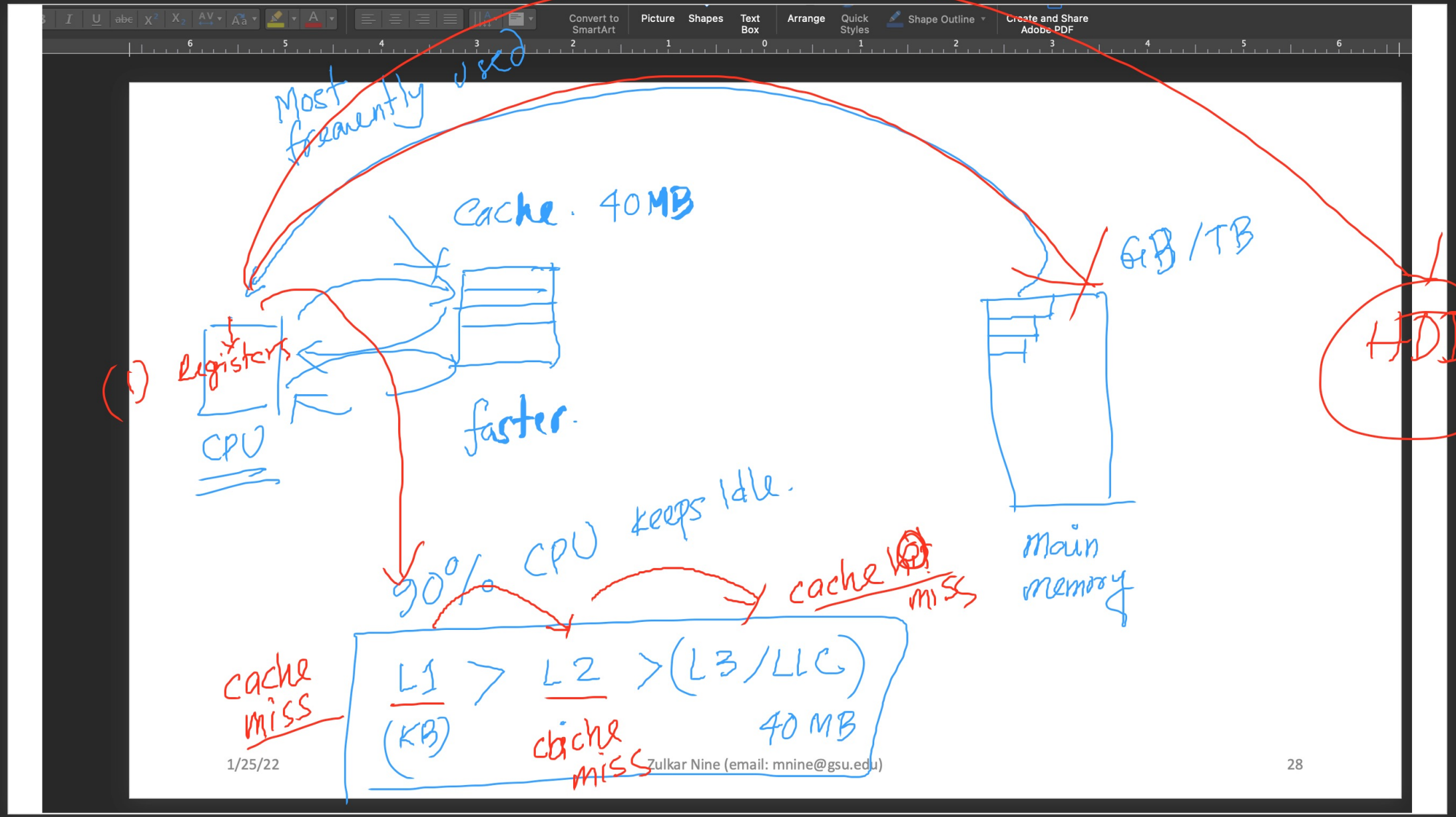
Reading from Memory: Cache Memory

- In practice, **instructions and data** are **not fetched**, most of the time, from the **main memory**.
- There is a **high-speed cache memory** that provides
  - faster access to **instructions** and **data** than the main memory.

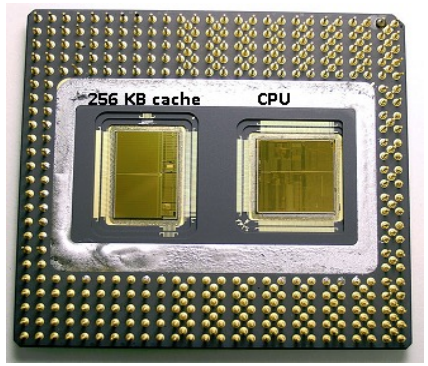




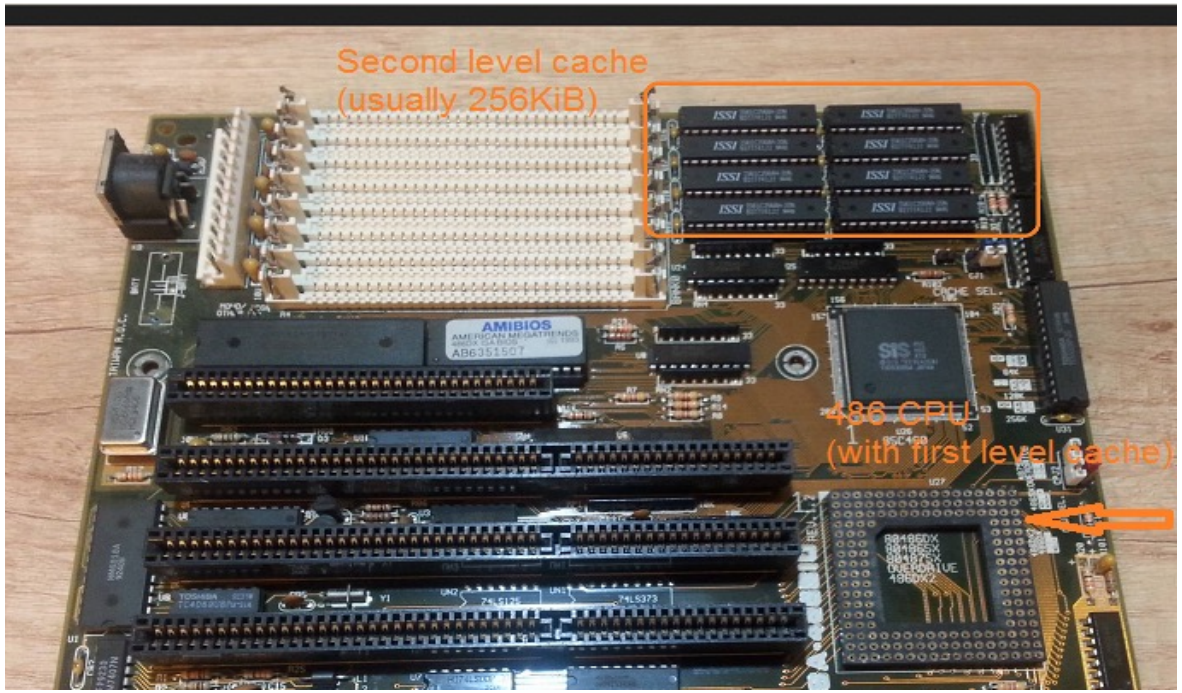



Most frequently used
cache. 40MB
Registers
CPU
faster.
GB/TB
HDD
Main memory
90% CPU keeps idle.
cache hit
miss
cache miss
L1 > L2 > (L3/LLC)
(KB)
cache miss
40 MB
1/25/22
Zulkar Nine (email: mnine@gsu.edu)
28

Reading from Memory: Cache Memory



- **Level-1 cache**: inside the CPU



- **Level-2 cache**: outside the CPU (attached to CPU by high speed data bus)

Reading from Memory: Cache Memory

- **Cache hit**: when data to be read is already in cache memory
- **Cache miss**: when data to be read is not in cache memory.

General Concepts

- Basic microcomputer design
- Instruction execution cycle
- Reading from memory
- **How programs run**

How a Program Runs

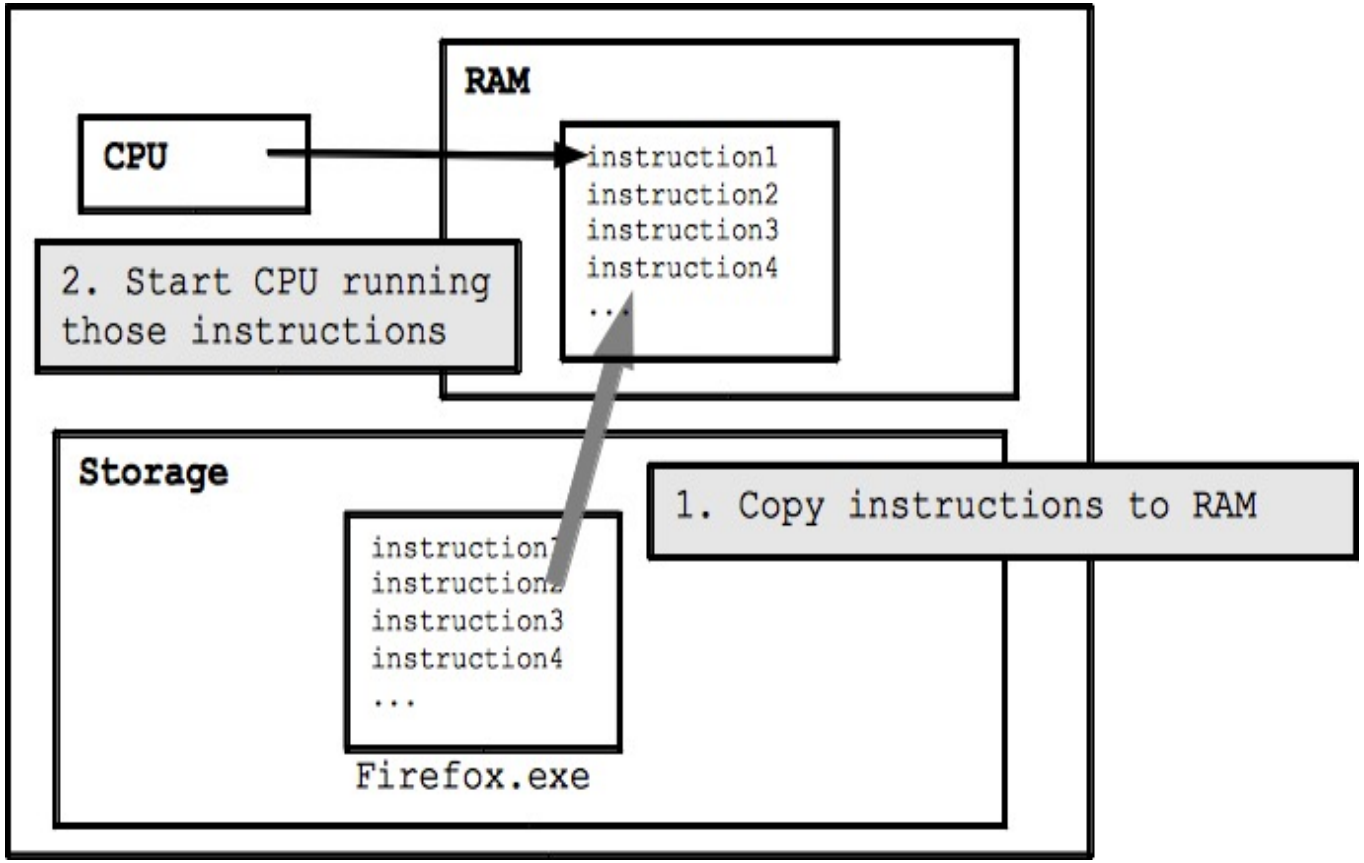
Program loaded: Hard disk drive

loader

RAM

OS points to program **entry point**

CPU



Think Again?

- Why does memory access take more machine cycles than register access?
- What are the three basic steps in the instruction execution cycle?
- Which two additional steps are required in the instruction execution cycle when a memory operand is used?