CSD1100

Computer Architecture And Operating Systems

Vadim Surov

Copyright © 2021 DigiPen Institute Of Technology

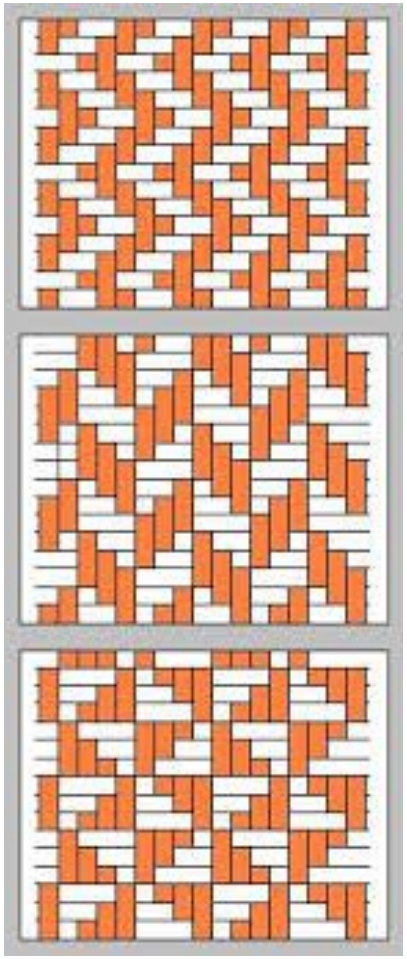
The Stored Program

- Brilliant idea:
  - Build a **machine that** can **execute** certain **instructions**.
  - Then, write down different lists of instructions (programs) to accomplish different things.
- Examples:
  - Play music using Barrel Organs
  - Music boxes
  - Mechanical Pianos

Copyright © 2021 DigiPen Institute Of Technology

The Stored Program

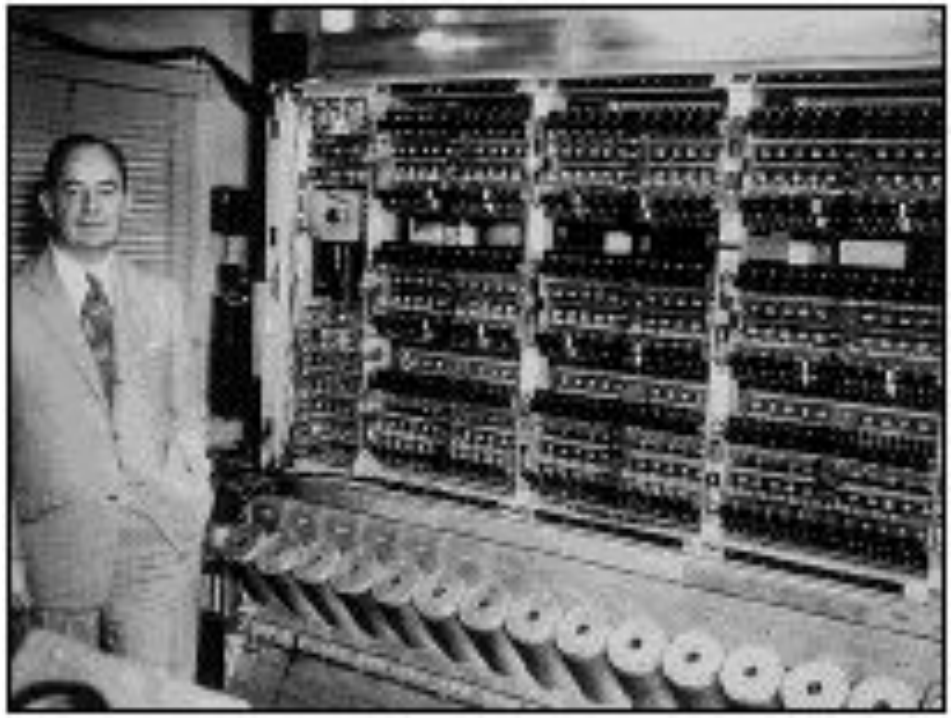
- Weaving patterns using punched cards.



Copyright © 2021 DigiPen Institute Of Technology

The Modern Computer

- Basic idea still the same in the modem computers.
- Machine instructions and data represented by sequences ‘low’ and ‘high’ signals.
- RAM is a place to store information while the computer is running.



John von Neumann

Copyright © 2021 DigiPen Institute Of Technology

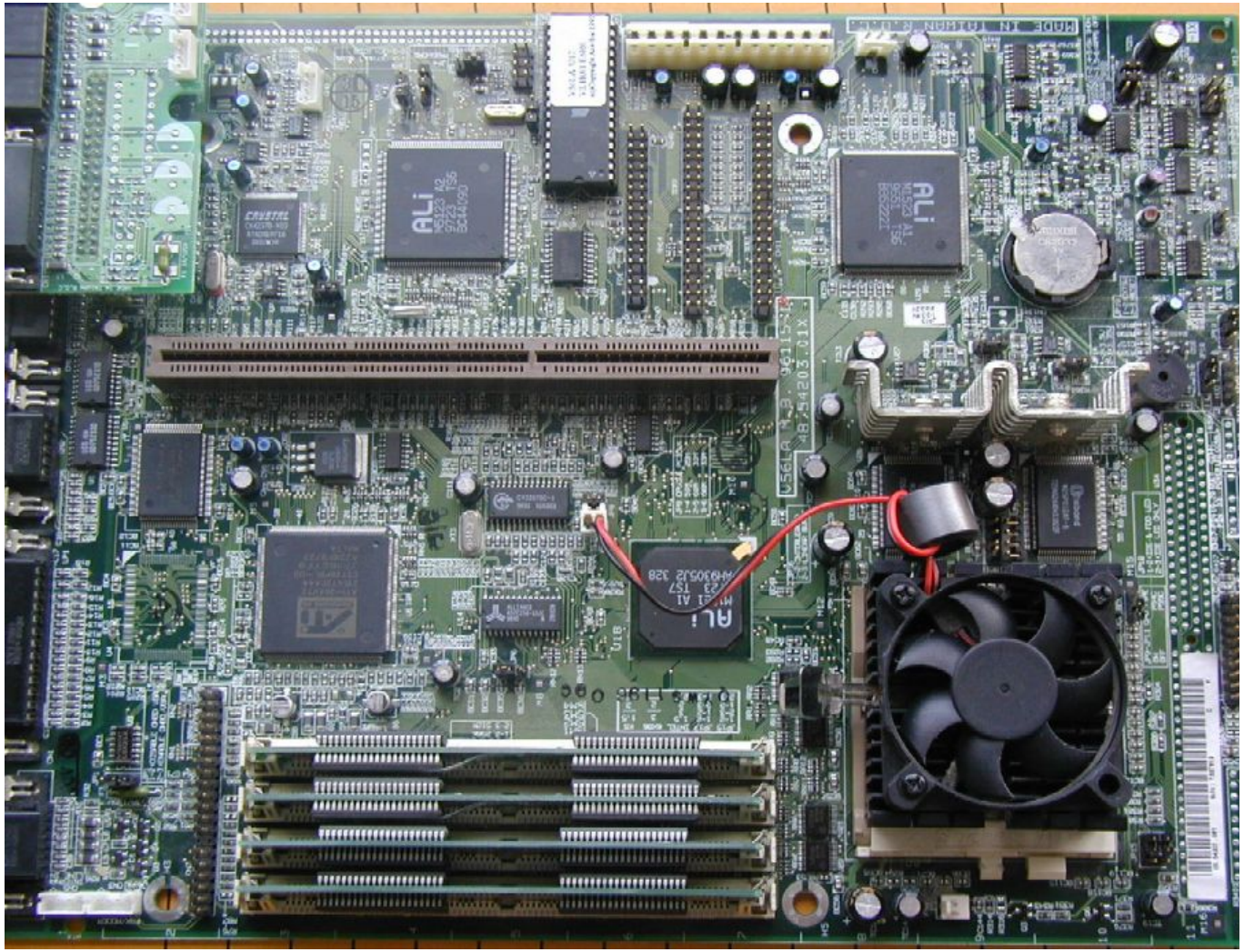
John von Neumann Model

- **The instructions themselves and the data the instructions manipulate are all stored in the same memory.**
- This innovation is crucial to modern computing.
- It even allows for the possibility of programs that change themselves as they are executed.

- With great power comes great risk …
  - Computer Viruses!

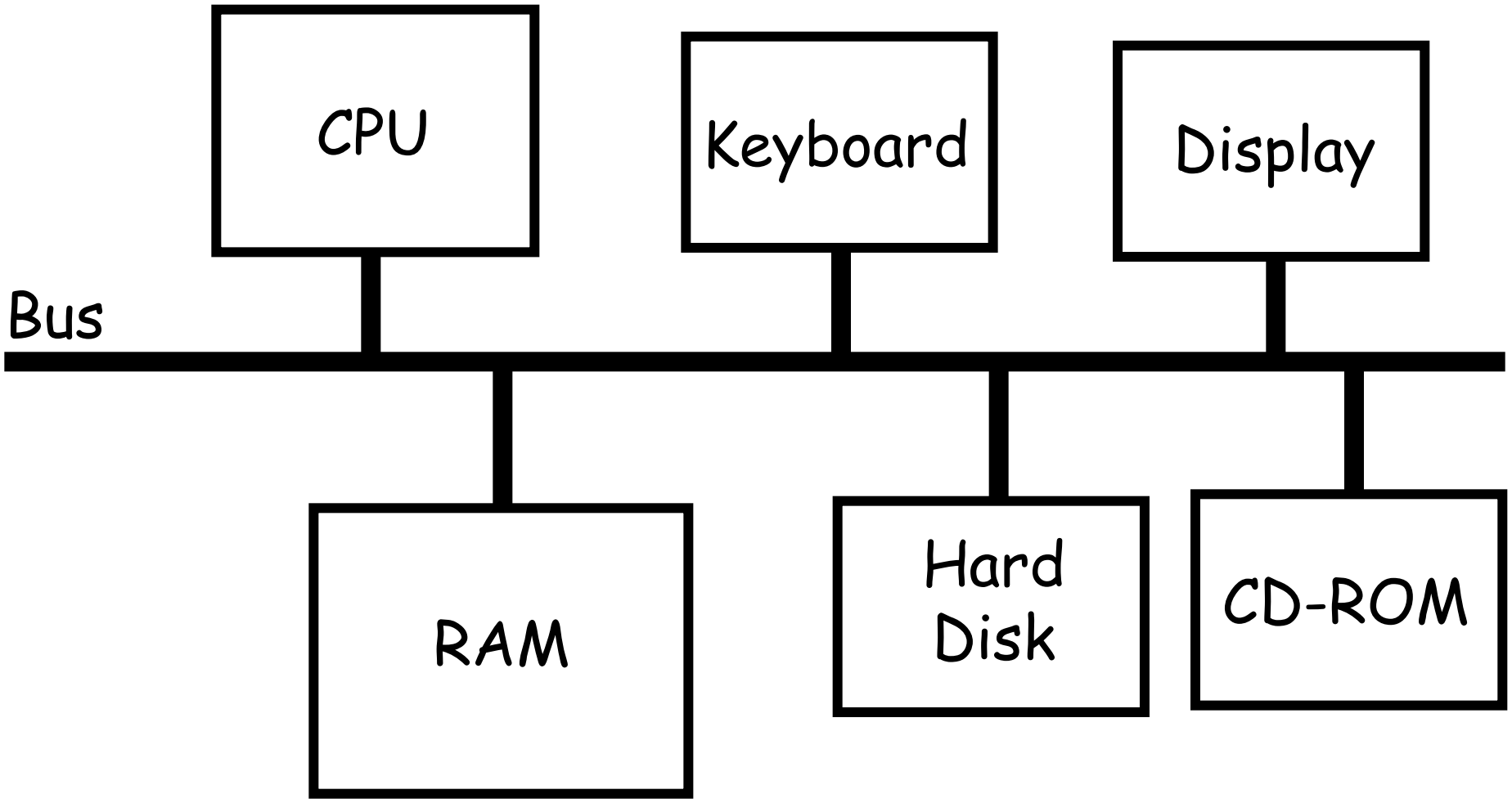
Copyright © 2021 DigiPen Institute Of Technology

Computer Architecture - Mainboard



Copyright © 2021 DigiPen Institute Of Technology

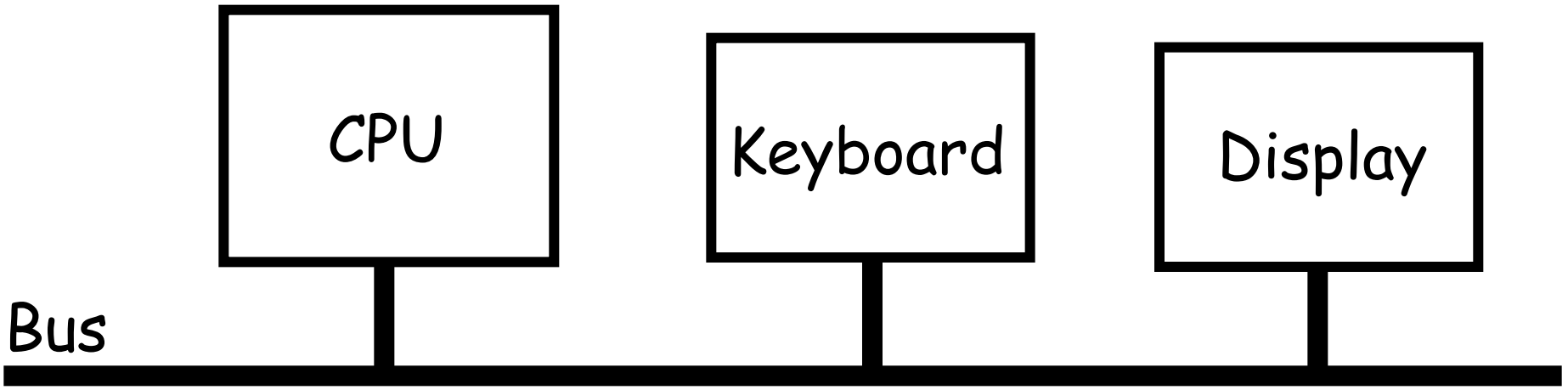
Computer Architecture



Copyright © 2021 DigiPen Institute Of Technology

The Bus

- Bus it is a simplified way for many devices to communicate to each other
  - Bus for addresses, bus for data, and bus for controls.
- Looks like a “highway” for information.
  - Actually, more like a “basket” that they all share.



Copyright © 2021 DigiPen Institute Of Technology

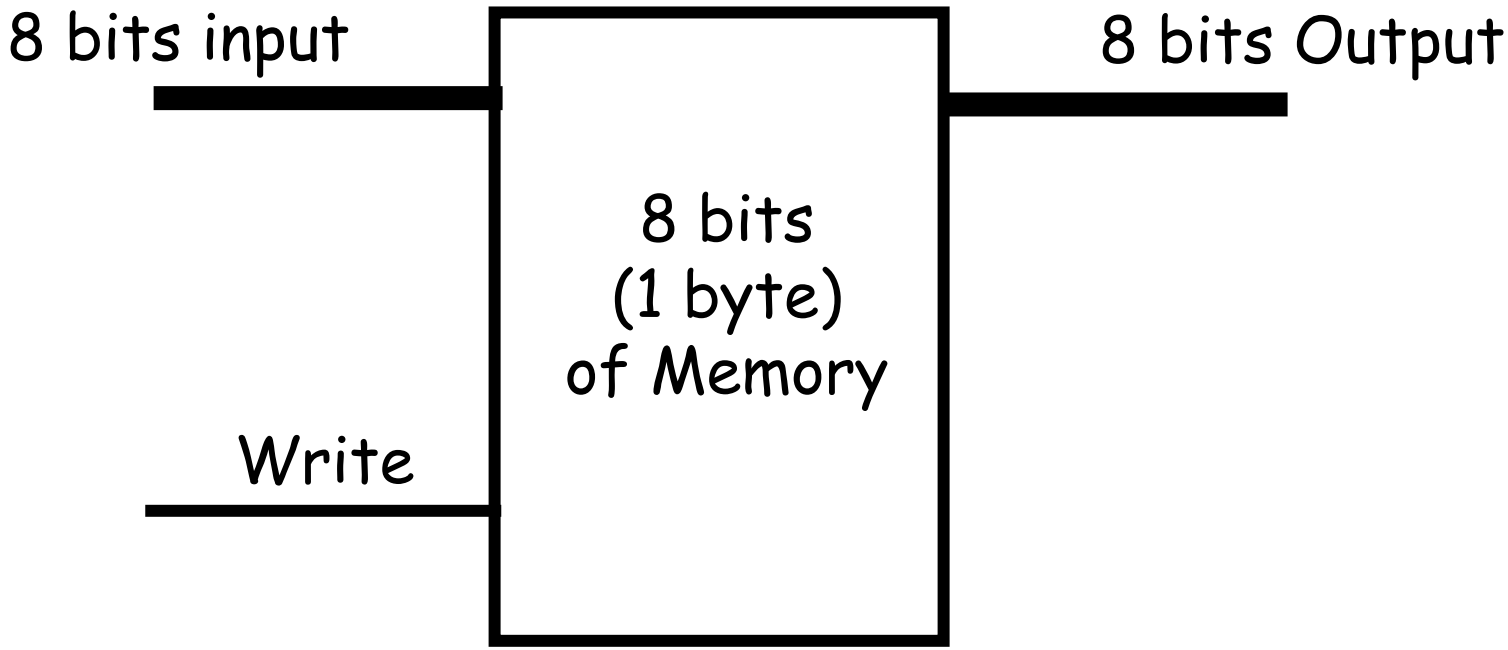
The Bus

- Suppose CPU needs to check to see if the user typed anything.
- CPU puts “Keyboard, did the user type anything?” (represented in some way) on the Bus.
- Each device (except CPU) is a State Machine that constantly checks to see what’s on the Bus.
- Keyboard notices that its name is on the Bus, and reads info. Other devices ignore the info.
- Keyboard then writes “CPU: Yes, user typed ‘a’.” to the Bus.
- At some point, CPU reads the Bus, and gets the Keyboard’s response.

Copyright © 2021 DigiPen Institute Of Technology

RAM

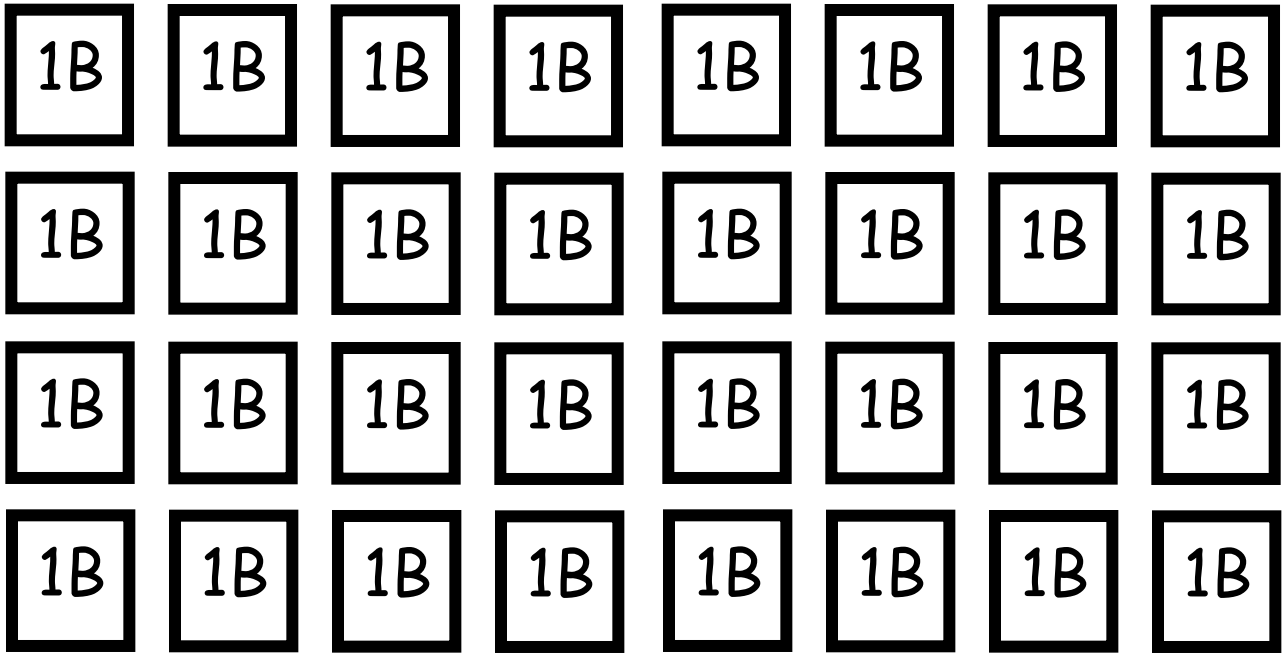
- Take a single bit of memory
- Group 8 of them together
- To have a byte of memory



Copyright © 2021 DigiPen Institute Of Technology

RAM

- We want a HUGE number of such 1 byte memory cells.



- How do we indicate which byte of memory we want to use at any given time?

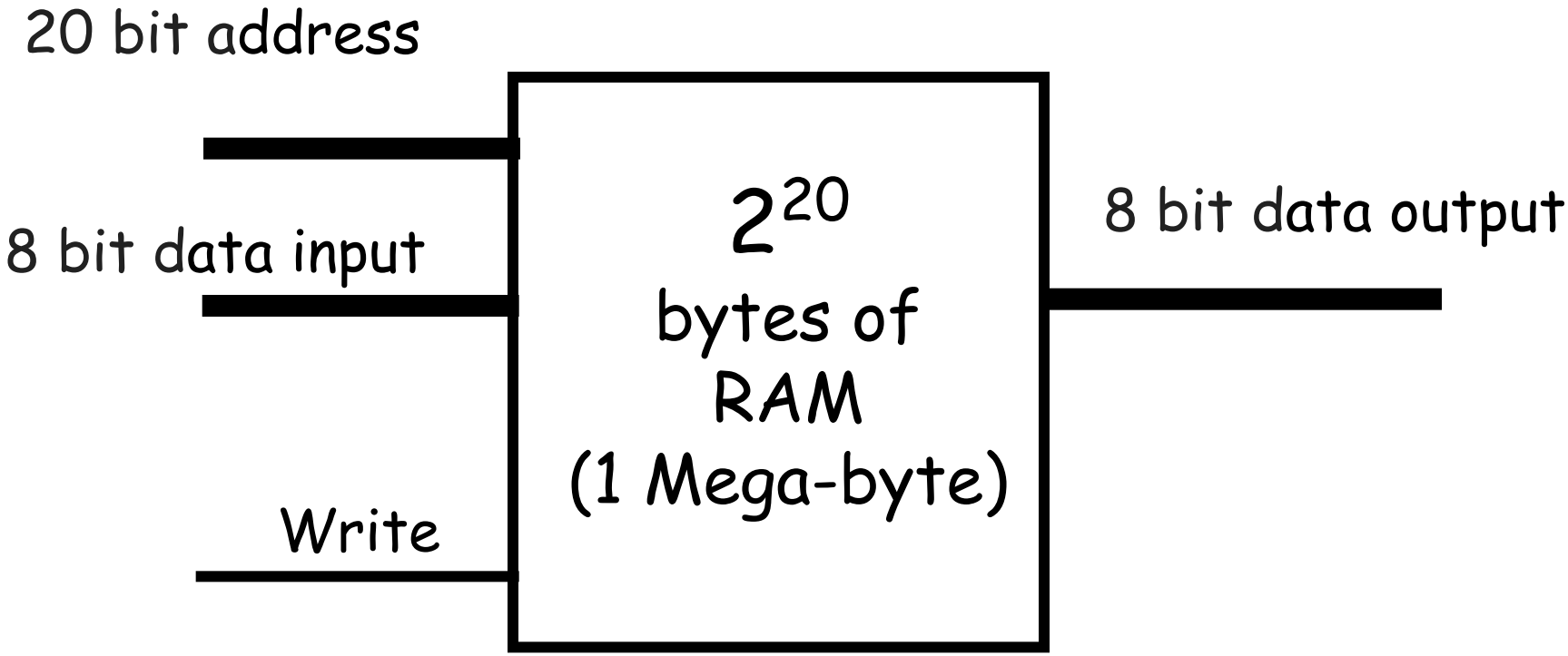
Copyright © 2021 DigiPen Institute Of Technology

RAM

- Answer: Assign each byte an address.
- We can number all the bytes in binary.
- Each byte's assigned number is called the byte's address in memory.
- Then, we can ask the RAM:
  - What is byte number 011010101010?
- Or tell the RAM:
  - Write "01101100" into memory at address 011010101010

Copyright © 2021 DigiPen Institute Of Technology

RAM



Copyright © 2021 DigiPen Institute Of Technology

Inside the CPU

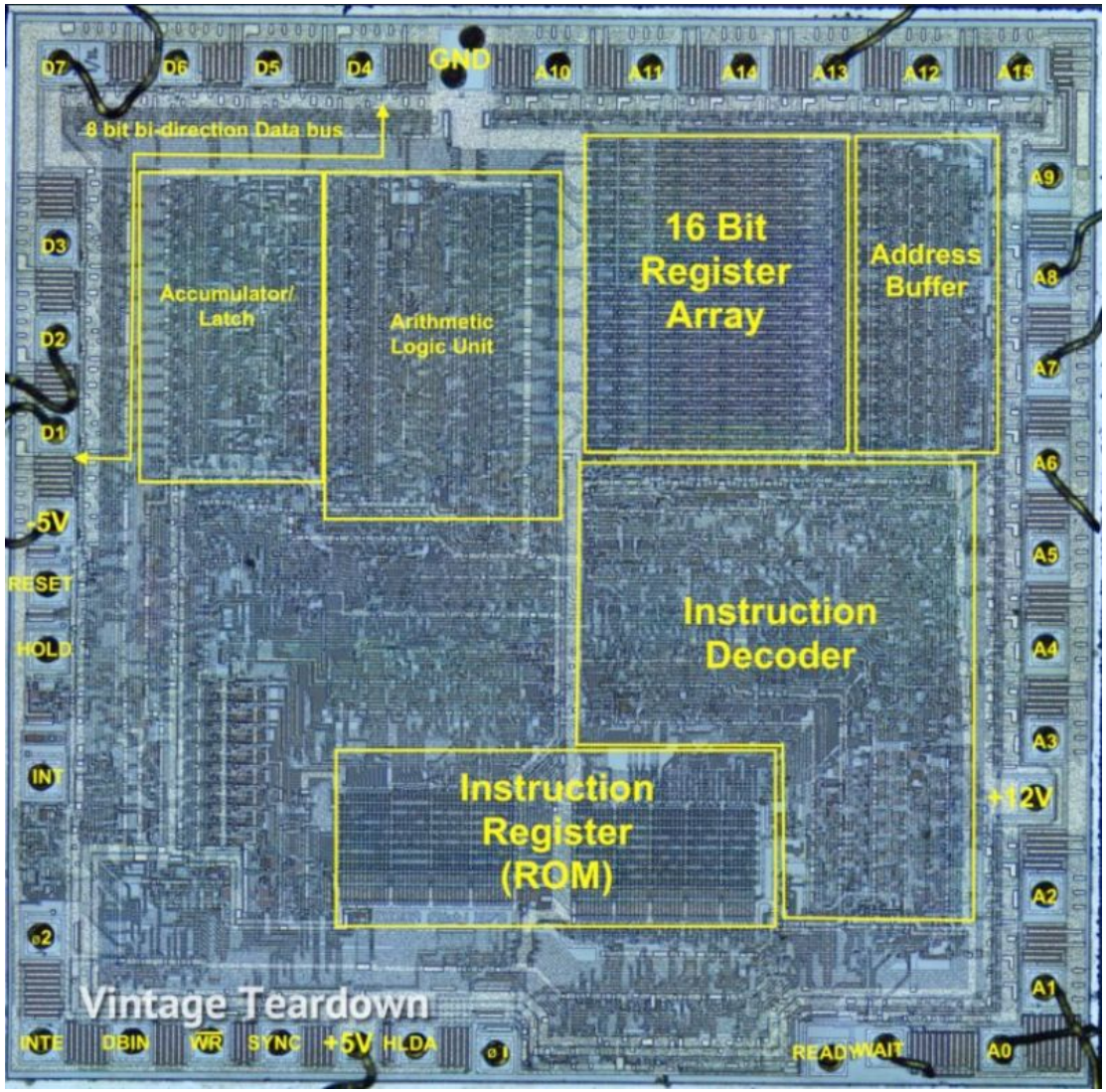
- The CPU is the brain of the computer.
- It is the part that actually executes the instructions.

- Let’s take a look inside.



Copyright © 2021 DigiPen Institute Of Technology

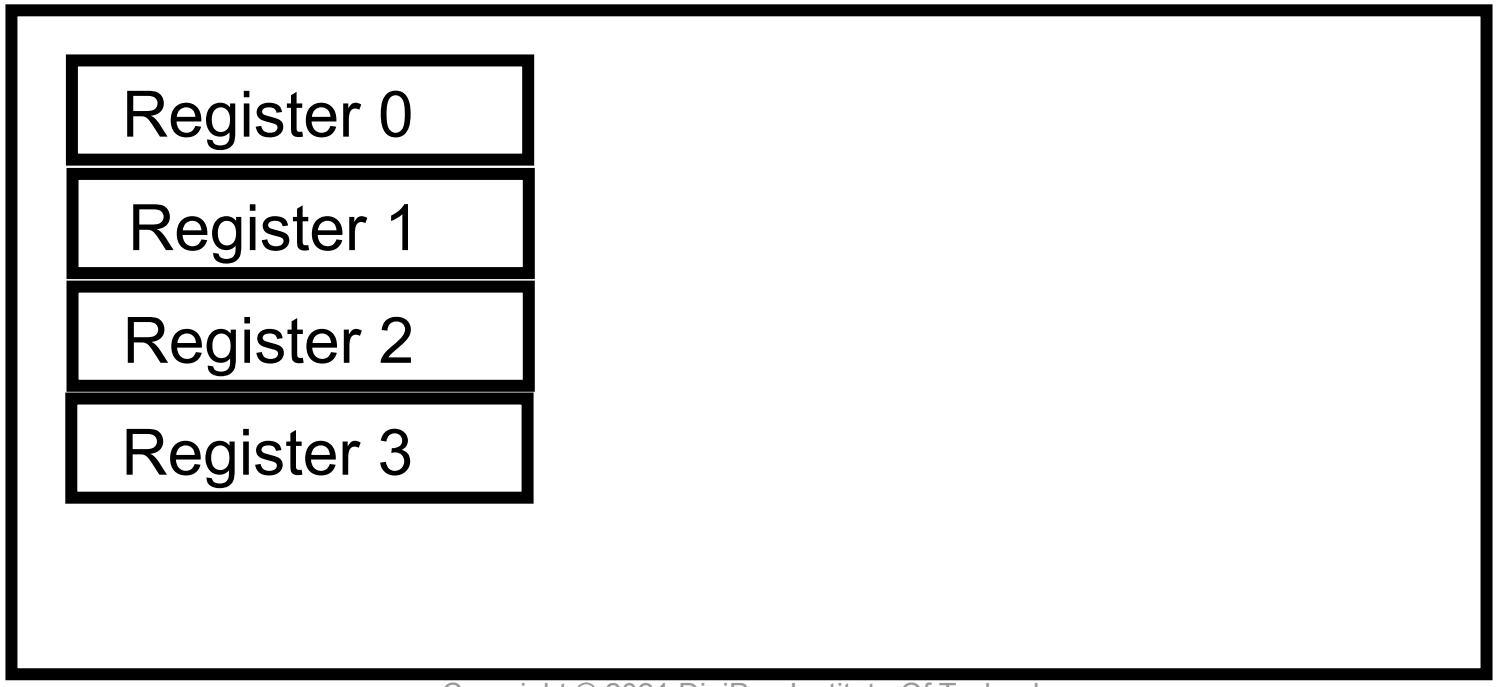
Inside the CPU



Copyright © 2021 DigiPen Institute Of Technology

Inside the CPU: Registers

- Temporary Memory.
- Computer “Loads” data from RAM to registers, performs operations on data in registers, and “stores” results from registers back to RAM



Copyright © 2021 DigiPen Institute Of Technology

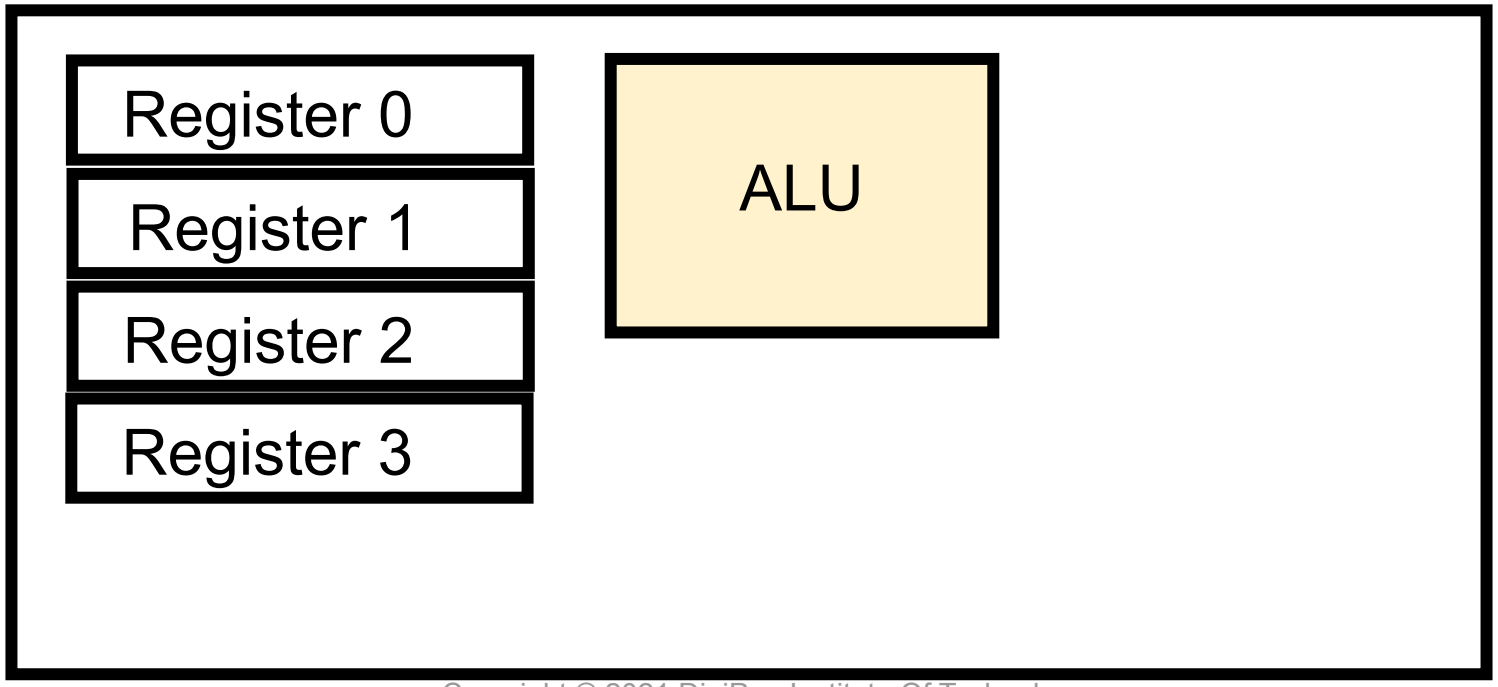
Inside the CPU: General Purpose Registers

- GPR - registers that can be used to hold data for the purpose of computation.
  - For example, if we have a statement in C like “ a = b+c+z;”, where a, b, c and z are locations in memory.
  - Usually, the ALU can only perform additions for 2 values at a time.
  - That means that the intermediate results of b+c needs to be saved somewhere.
  - The usual practice is to use the GPRs instead of memory locations to store these intermediate results for quick access purposes.

Copyright © 2021 DigiPen Institute Of Technology

Inside the CPU: ALU

- For doing basic Arithmetic / Logic
- Operations on values stored in the Registers



Copyright © 2021 DigiPen Institute Of Technology

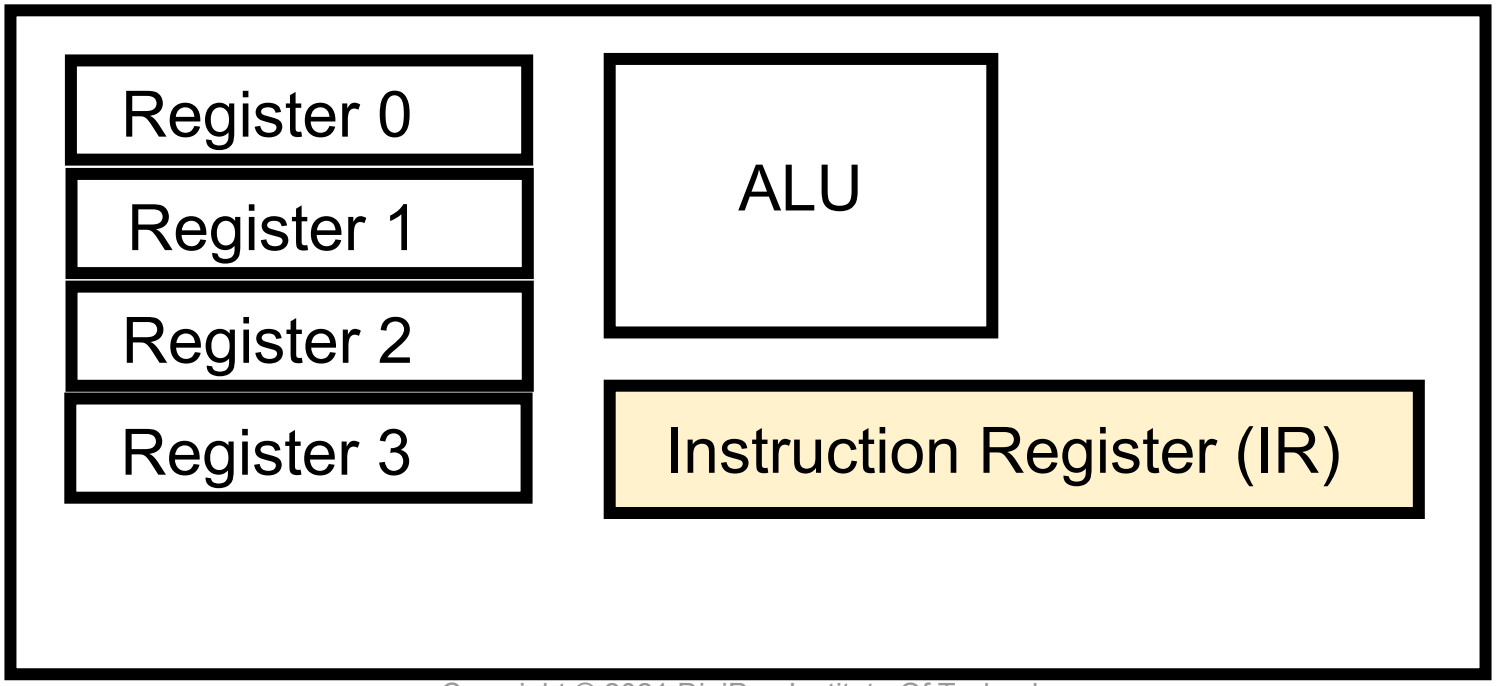
Inside the CPU: ALU

- Instructions performed in the ALU can be divided into two categories depending on the location of operands:
  - Register-Register or Register-Memory instructions allow memory words to be fetched into registers, where they can be used as ALU inputs, perform some operations on them, and then store the result back in a register.
  - A Memory-Memory instruction fetches its operands from the memory into the ALU inputs registers, performs its operation, and then writes the result back into memory.

Copyright © 2021 DigiPen Institute Of Technology

Inside the CPU: Instruction Register (IR)

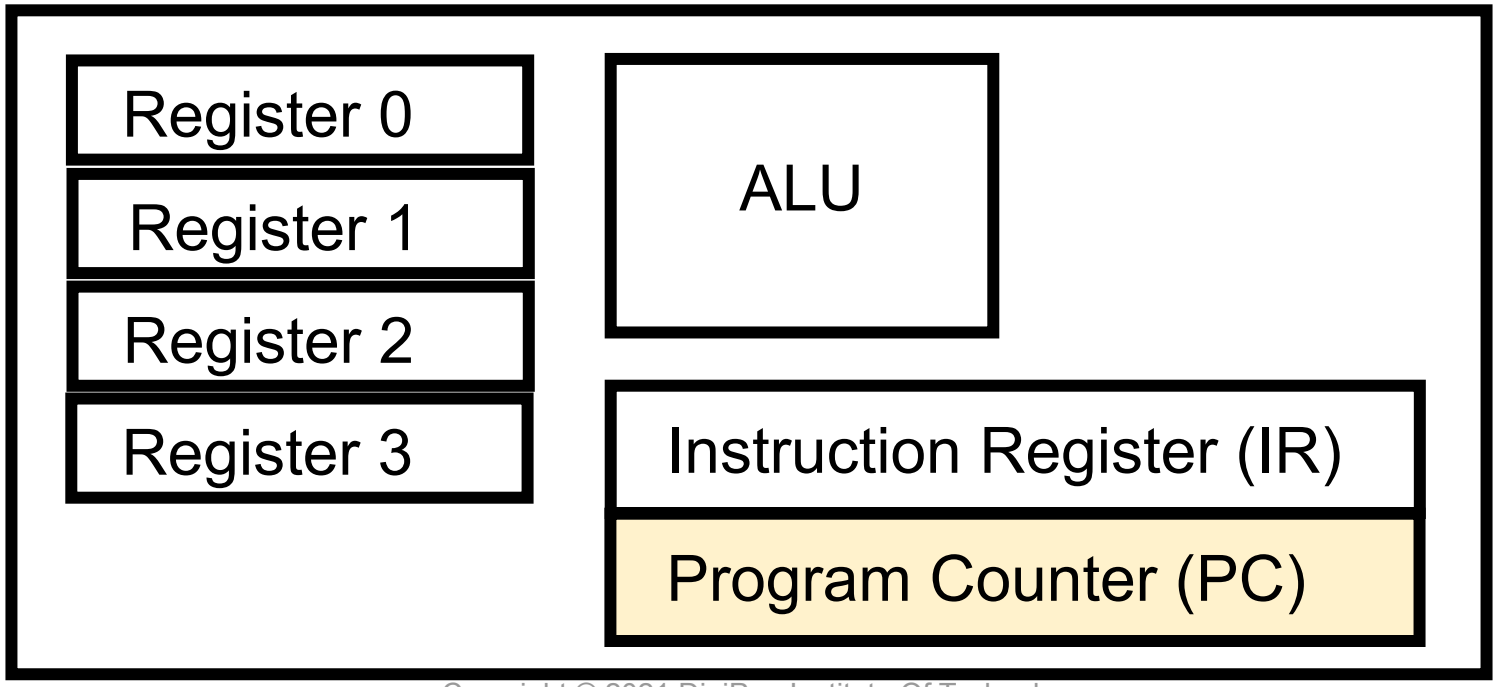
- Holds the instruction currently being decoded and executed
- Can be more IR than 1, bcs decoding on several instructions can be done in parallel.



Copyright © 2021 DigiPen Institute Of Technology

Inside the CPU: Program Counter (PC) / Instruction Pointer

- To hold the address of the current instruction in RAM



Copyright © 2021 DigiPen Institute Of Technology

Other Registers

- Status register
  - Status register also referred to as the condition code or flag, is used to make decisions depending on the value of flags. Those flags are set depending on the result of the last operation.
  - The most important status register flags are:
    - Zero flag: ZF
    - Overflow flag: OF
    - Sign flag: SF
    - Carry flag: CF

Copyright © 2021 DigiPen Institute Of Technology

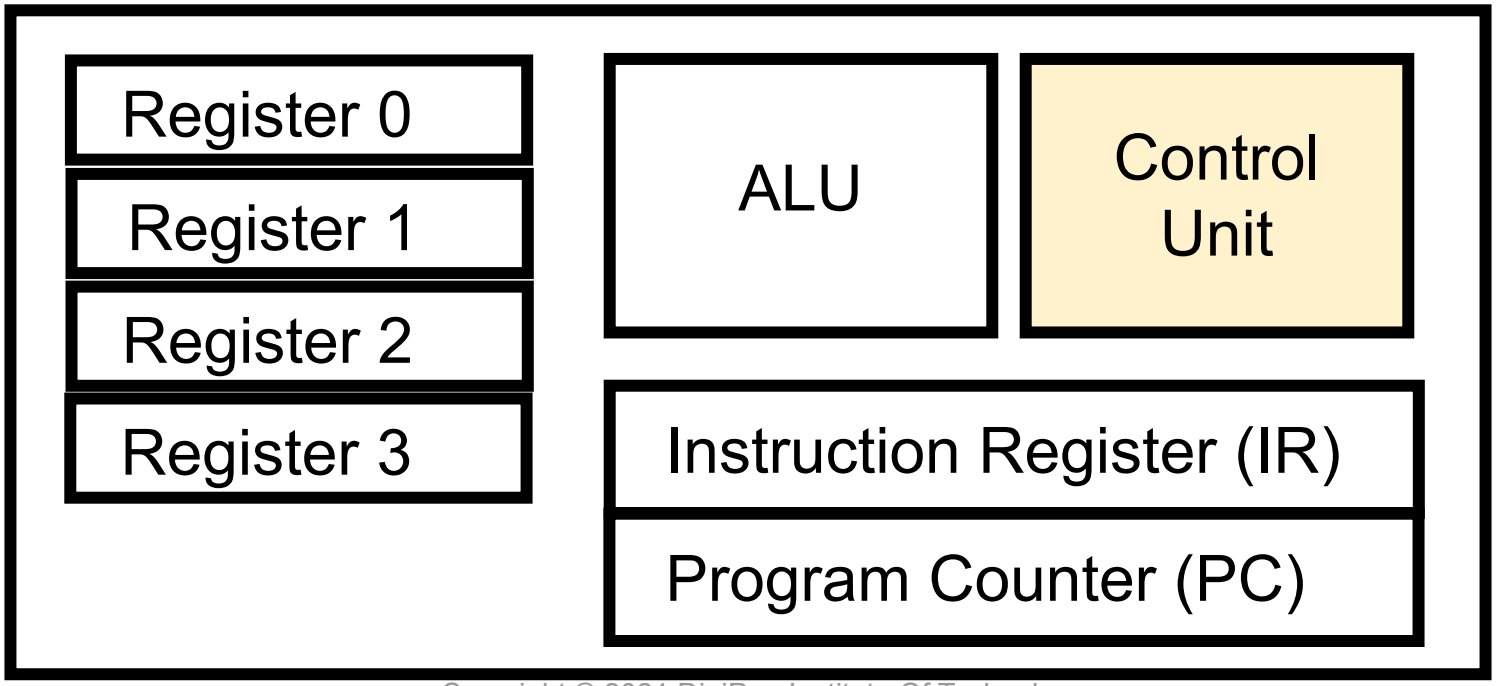
Other Registers

- Memory Address Register (MAR)
  - The memory address register (MAR) holds the address of the memory where data should be stored.
- Memory Data Register (MDR)
  - The memory data register (MDR) stores the results of an instruction.

Copyright © 2021 DigiPen Institute Of Technology

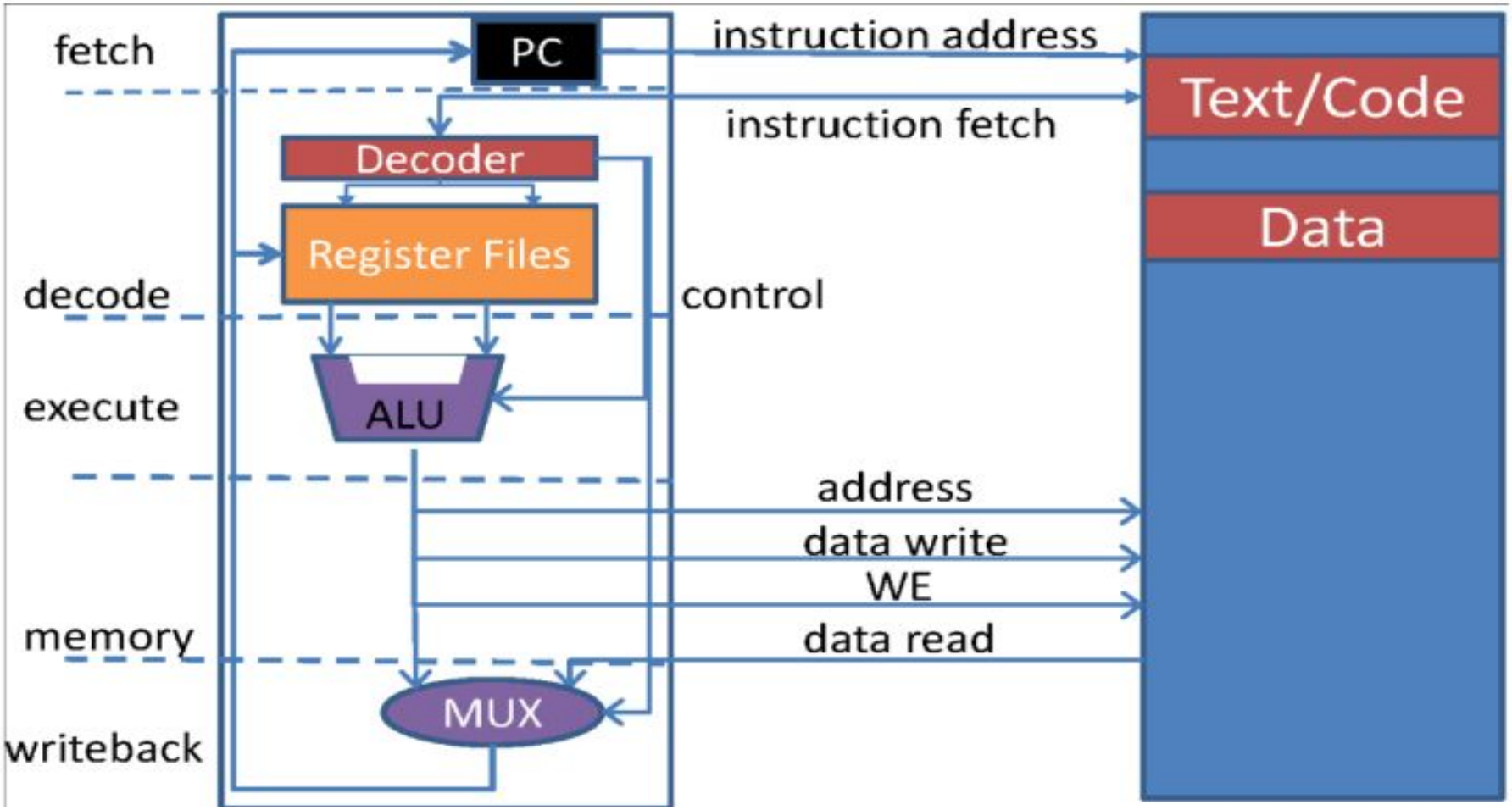
Inside the CPU: The Control Unit

- It all comes down to the Control Unit.
- CU makes sure that everything is running smoothly and in order.



Copyright © 2021 DigiPen Institute Of Technology

Datapath



- Multiplexer(MUX) - a device that selects one of several digital input signals and forwards the selected input into a single line

Copyright © 2021 DigiPen Institute Of Technology

Fetch-Decode-Execute-Writeback

- Fetch Stage
  - The value stored in the PC is the address of the instruction we want to fetch. That instruction is fetched into the instruction register which is an input into the decoder.
- Decode Stage
  - The decoder extracts the following from the instruction stored in the instruction register
    - Operands involved for the current operation,
    - What operation the ALU needs to do
    - Desired effect of the operation.

Copyright © 2021 DigiPen Institute Of Technology

Fetch-Decode-**Execute**-Writeback

- Execute Stage
  - The ALU performs the operation based on the inputs given from the register file (collection of registers that can be accessed by programmers) and the decoder.
  - The control line of the decoder indicates the desired operation (addition/subtraction/etc) while the input from the register files are the operands of the operation.

Copyright © 2021 DigiPen Institute Of Technology

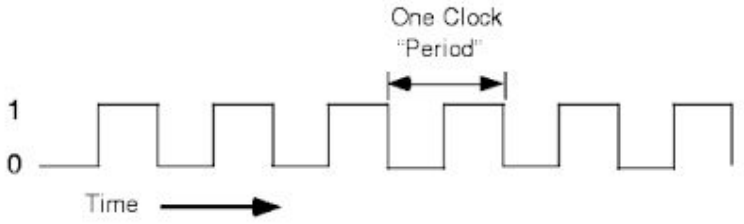
Fetch-Decode-Execute-**Writeback**

- Writeback Stage
  - Again, the input from the decoder determines what happens during this stage.
  - Depending on the instruction that is decoded
    - the result of the ALU may be written back into memory,
    - a value may be loaded onto the register from the memory or
    - the result of the ALU may be written back into the register file immediately.

Copyright © 2021 DigiPen Institute Of Technology

System Clock

- Every computer contains an **system** (AKA internal) **clock** that regulates the rate at which instructions are executed and synchronizes all the various computer components
- The clock signal consists of clock cycles (every clock cycle is called the clock period) that are generally represented by a square wave.



Copyright © 2021 DigiPen Institute Of Technology

System Clock

- Now in order to specify the clock speed or cycle, we use the **clock frequency** that is calculated using the inverse of the clock period (the duration of time of one cycle).
- We measure the clock frequency in Hz and it represents one cycle/second.
- Top CPU: Intel Core i7-8700K
  - Base clock: 3.7GHz
  - Boost clock: 4.7GHz
- GHz = 10^9 clock cycles per second.

Copyright © 2021 DigiPen Institute Of Technology

I/O Devices

- Man readable
  - Printers, graphical terminals, screen, keyboard, mouse, pointing devices,…
- Machine readable
  - Disks, tapes, sensors, controllers, actuators
- Communication
  - Modems

- So different, so need **a uniform approach to I/O as seen from the user and from the operating system point of view**

Copyright © 2021 DigiPen Institute Of Technology

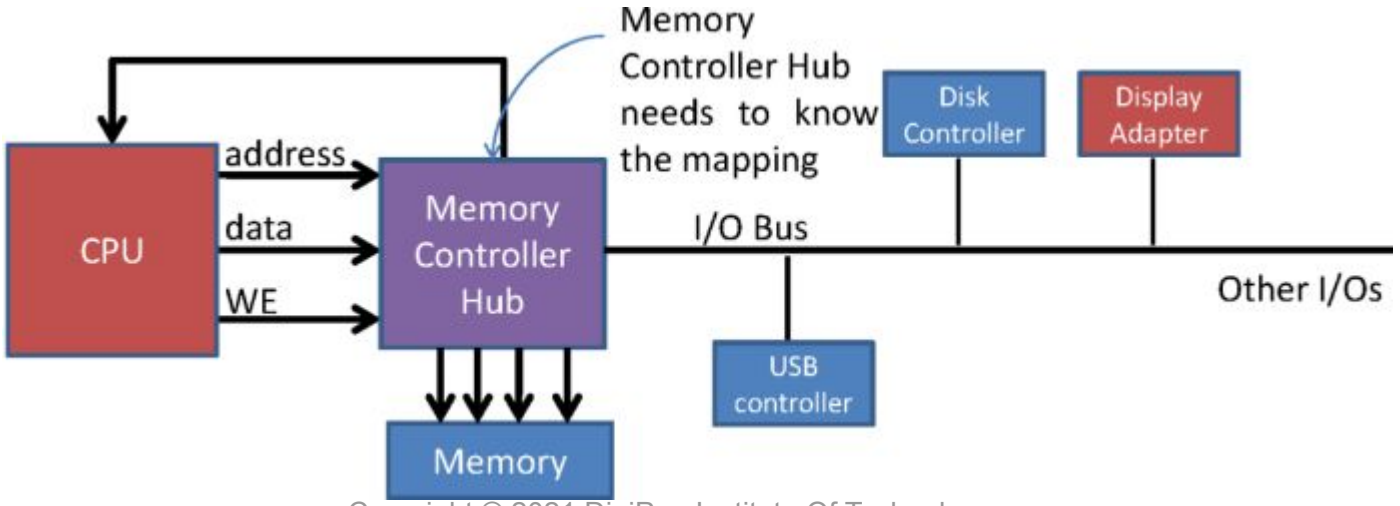
I/O Devices. **Special Instructions**

- Besides arithmetic and memory instructions, we include special instructions for interacting with I/O.
- Can be done using dedicated I/O processors (channels) which execute their own instructions

Copyright © 2021 DigiPen Institute Of Technology

I/O Devices. **Memory-Mapped Instructions**

- Idea is to uses the same address space to address both memory and I/O devices to make some special addresses correspond to specific I/Os.
- This mapping is dependent on the motherboard and the computer architecture and is achieved through using a **memory controller hub** which is like a postman who delivers letters to the correct household.



Copyright © 2021 DigiPen Institute Of Technology

I/O Devices. **Memory-Mapped Instructions**

- Thus when the addresses are those mapped to the RAM, the RAM is accessed. Otherwise, the information is passed onto the I/O bus for the specific I/O device.
- This way, from the CPU point of view, **communication with I/O is just like reading or writing to memory**.

A sample system memory map

Address range (hexadecimal)	Size	Device
0000–7FFF	32 KiB	RAM
8000–80FF	256 bytes	General-purpose I/O
9000–90FF	256 bytes	Sound controller
A000–A7FF	2 KiB	Video controller/text-mapped display RAM
C000–FFFF	16 KiB	ROM

Copyright © 2021 DigiPen Institute Of Technology

I/O Devices. **Polling**

- Polling can be used to inform CPU of an I/O event happening
- The CPU executes instructions to keep on checking whether a desired I/O operation is done or not.
- Pros:
  - Simple
- Cons
  - Wastes CPU cycles that can be used to do other useful tasks.

Copyright © 2021 DigiPen Institute Of Technology

I/O Devices. **Interrupts**

- Enable the I/O devices to signal the CPU electronically when a task is completed.
- Pros
  - Increased CPU utilization
- Cons
  - Complex procedures need to be put in place in order to handle interrupts.

Copyright © 2021 DigiPen Institute Of Technology

Programs and Instructions

- Programs are made up of instructions
- CPU executes one instruction every clock cycle
  - Modern CPUs do more, but we ignore that
- Specifying a program and its instructions:
  - Lowest level: Machine language
  - Intermediate level: Assembly language
  - Typically today: High-level programming language

Copyright © 2021 DigiPen Institute Of Technology

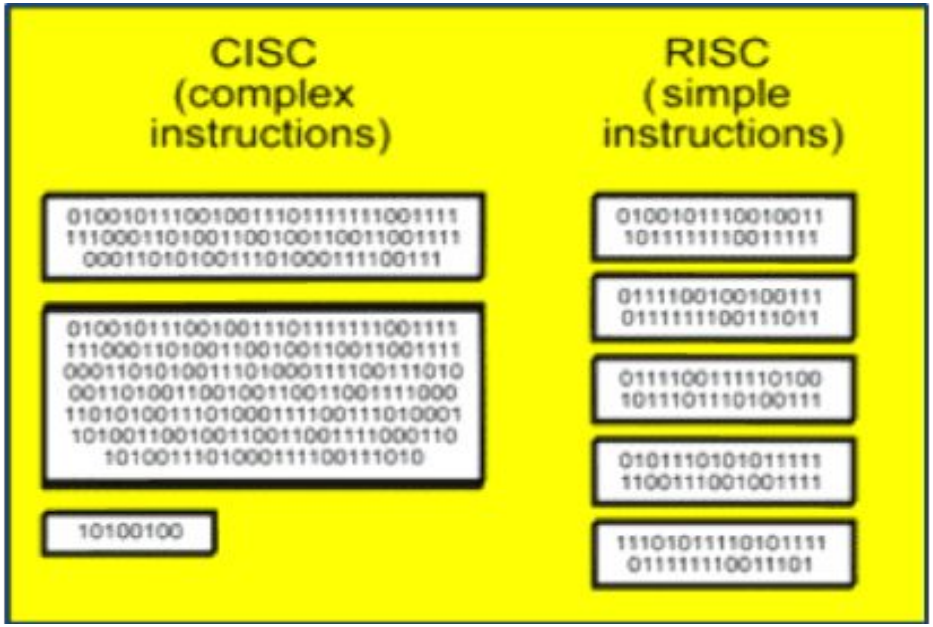
Programs and Instructions

- High-level programs: each statement translates to many instructions
  - int c = a + b;
- Assembly language: specify each machine instruction, using mnemonic form
  - add rax, a
- Machine language: specify each machine instruction, using bit patterns
  - 1101101000001110011

Copyright © 2021 DigiPen Institute Of Technology

Complex vs Simple Instructions

- Processors can be
  - Complex Instruction Set Computer (CISC)
  - Reduced Instruction Set Computer (RISC)



- RISC utilizes small and highly optimized set of instructions.
- The main intent of the CISC processor architecture is to complete task by using less number of assembly lines

Copyright © 2021 DigiPen Institute Of Technology

Typical Assembly Instructions

- Some common assembly instructions include:
  - “Load” – Load a value from RAM into one of the registers
  - “Add” – Add the contents of two registers and put the result in a third register
  - “Compare” – If the value in a specified register is larger than the value in a second register, put a “0” in Register r0
  - “Jump” – If the value in Register r0 is “0”, change IP to the value in a given register

Copyright © 2021 DigiPen Institute Of Technology

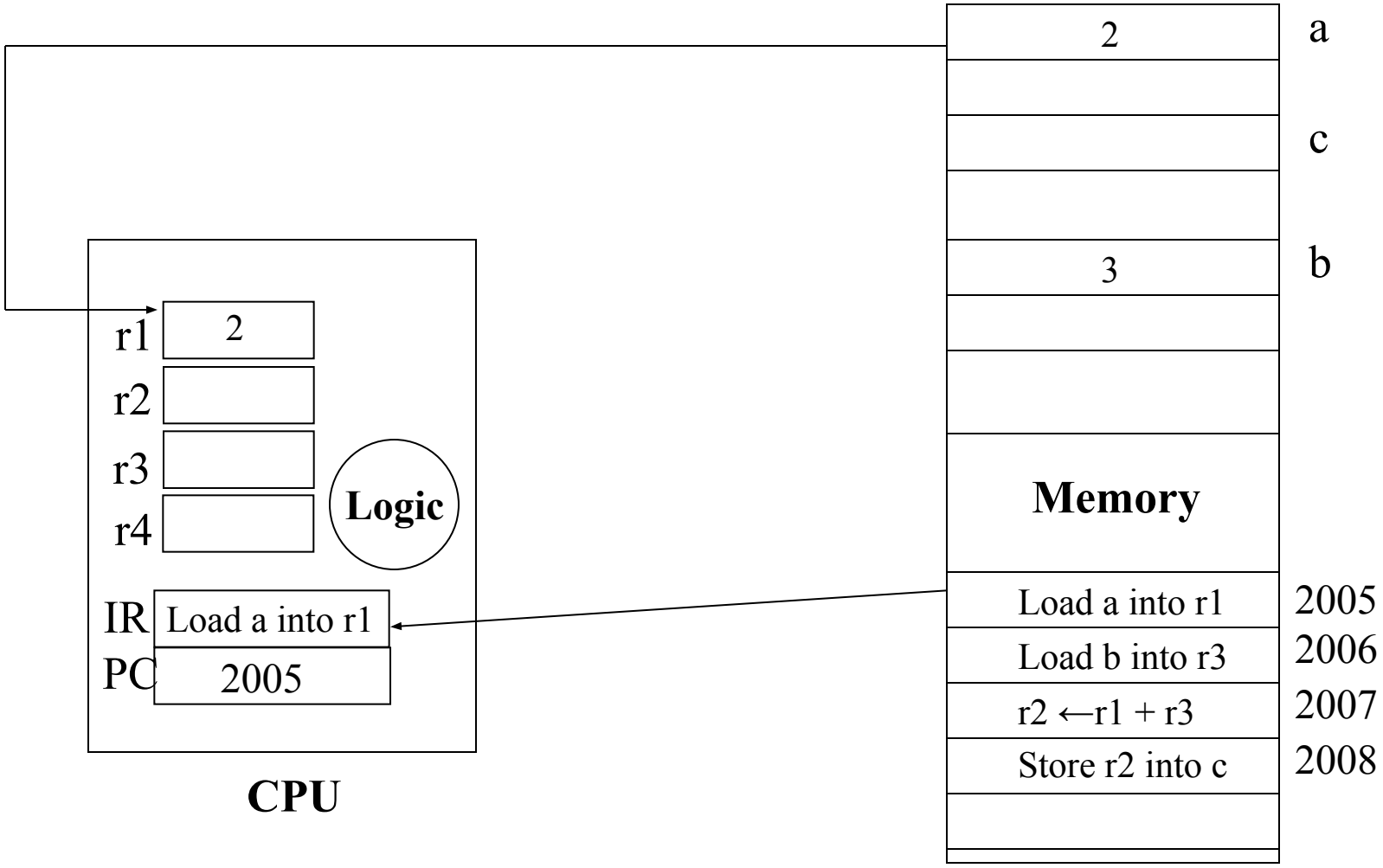
Example: A Simple Program

- Want to add values of variables a and b (assumed to be in memory), and put the result in variable c in memory

- Instructions in program
  - Load a into register r1
  - Load b into register r3
  - r2 ← r1 + r3
  - Store r2 in c

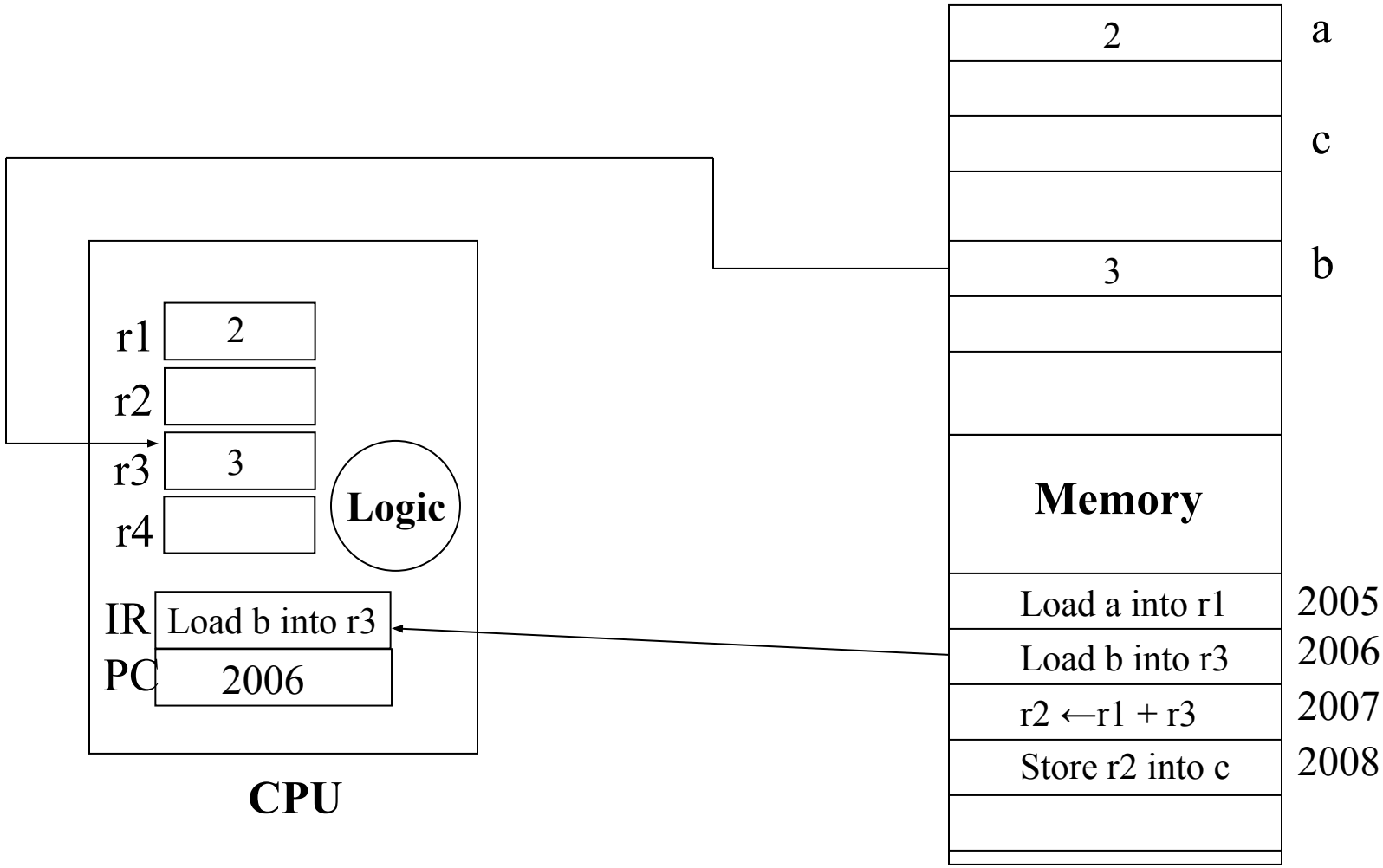
Copyright © 2021 DigiPen Institute Of Technology

A Simple Program: Running



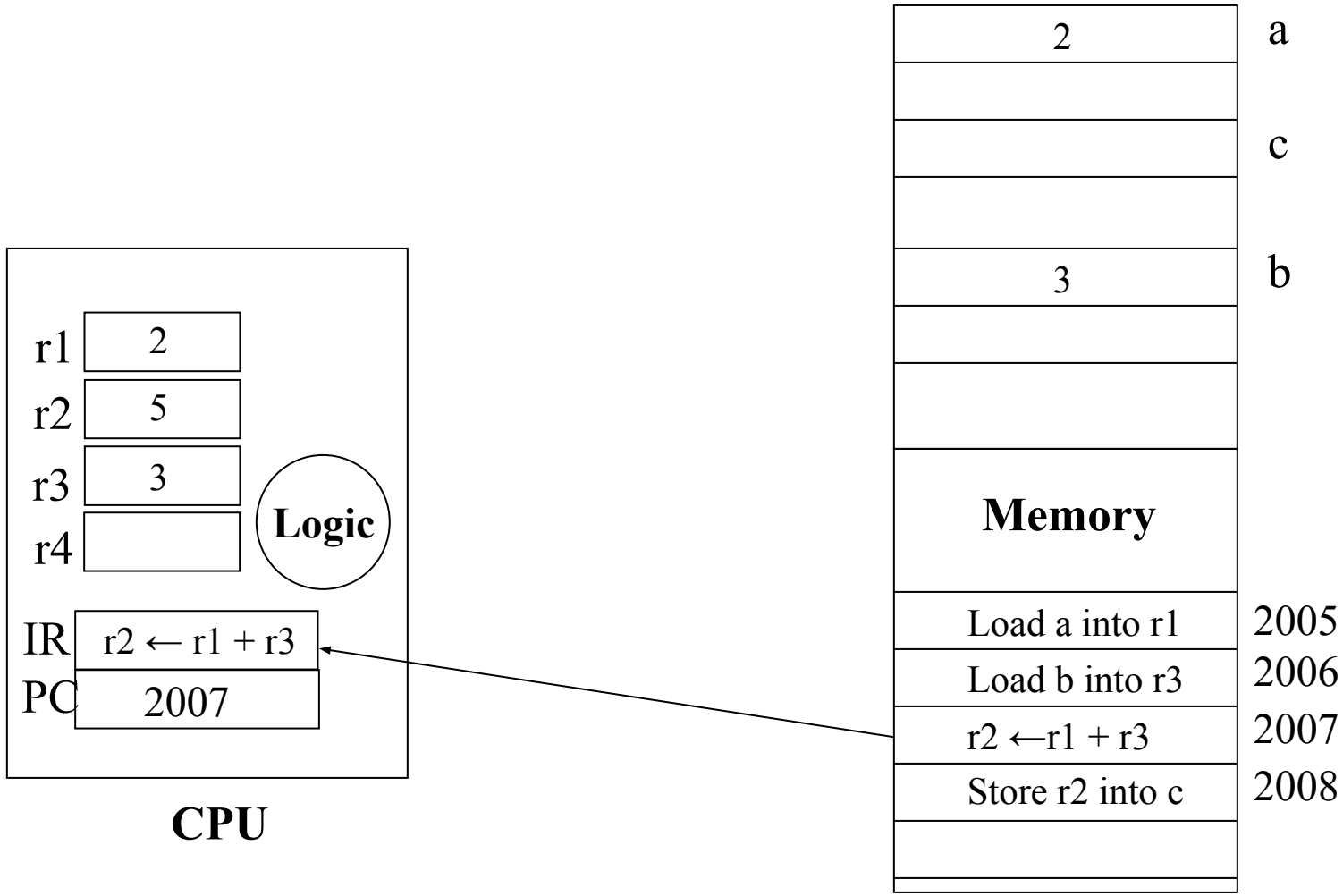
Copyright © 2021 DigiPen Institute Of Technology

A Simple Program: Running



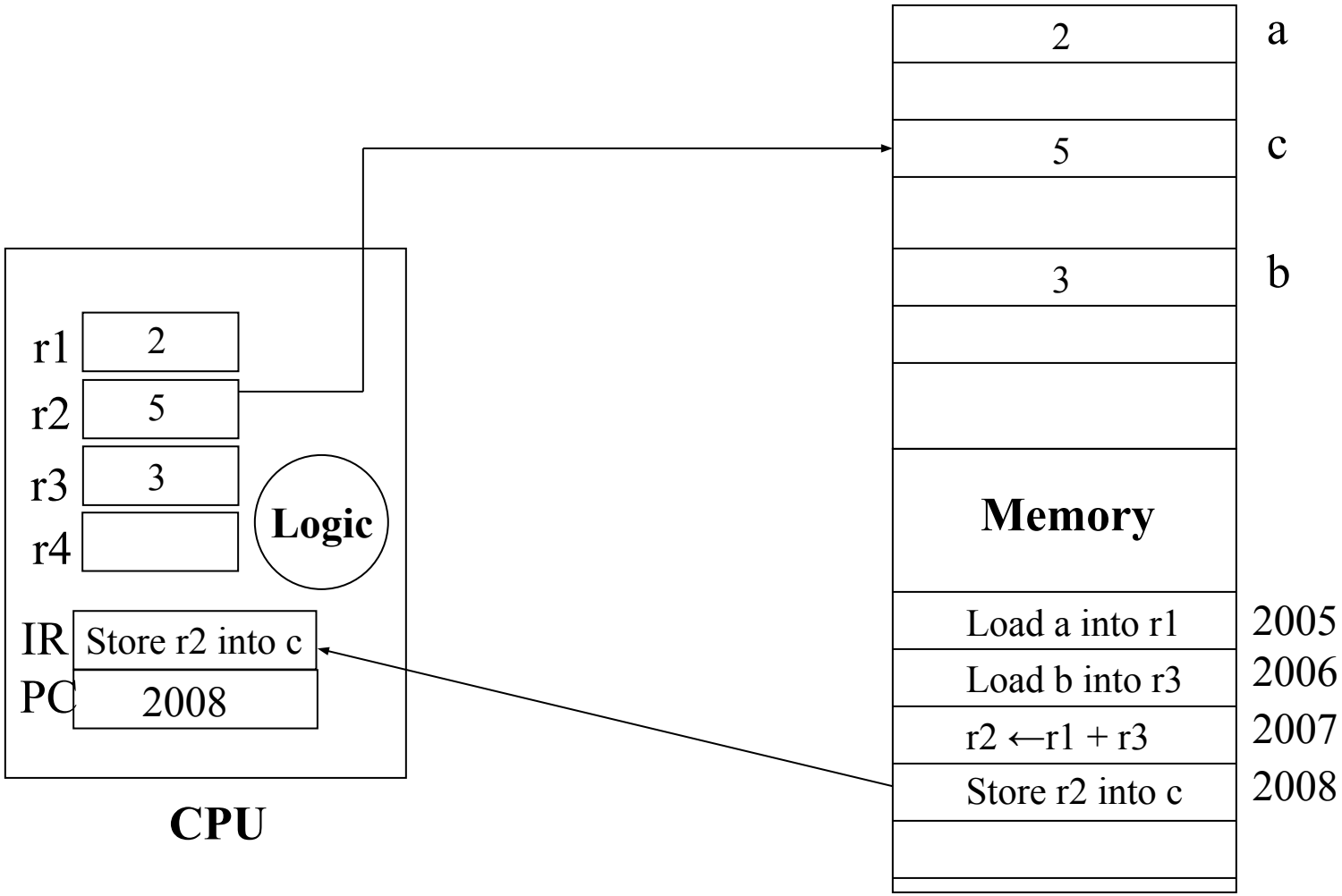
Copyright © 2021 DigiPen Institute Of Technology

A Simple Program: Running



Copyright © 2021 DigiPen Institute Of Technology

A Simple Program: Running



Copyright © 2021 DigiPen Institute Of Technology

Program execution

- So how does a program execute?
- Naïve answer:
  - load the program into memory and point the PC to the start of the program.

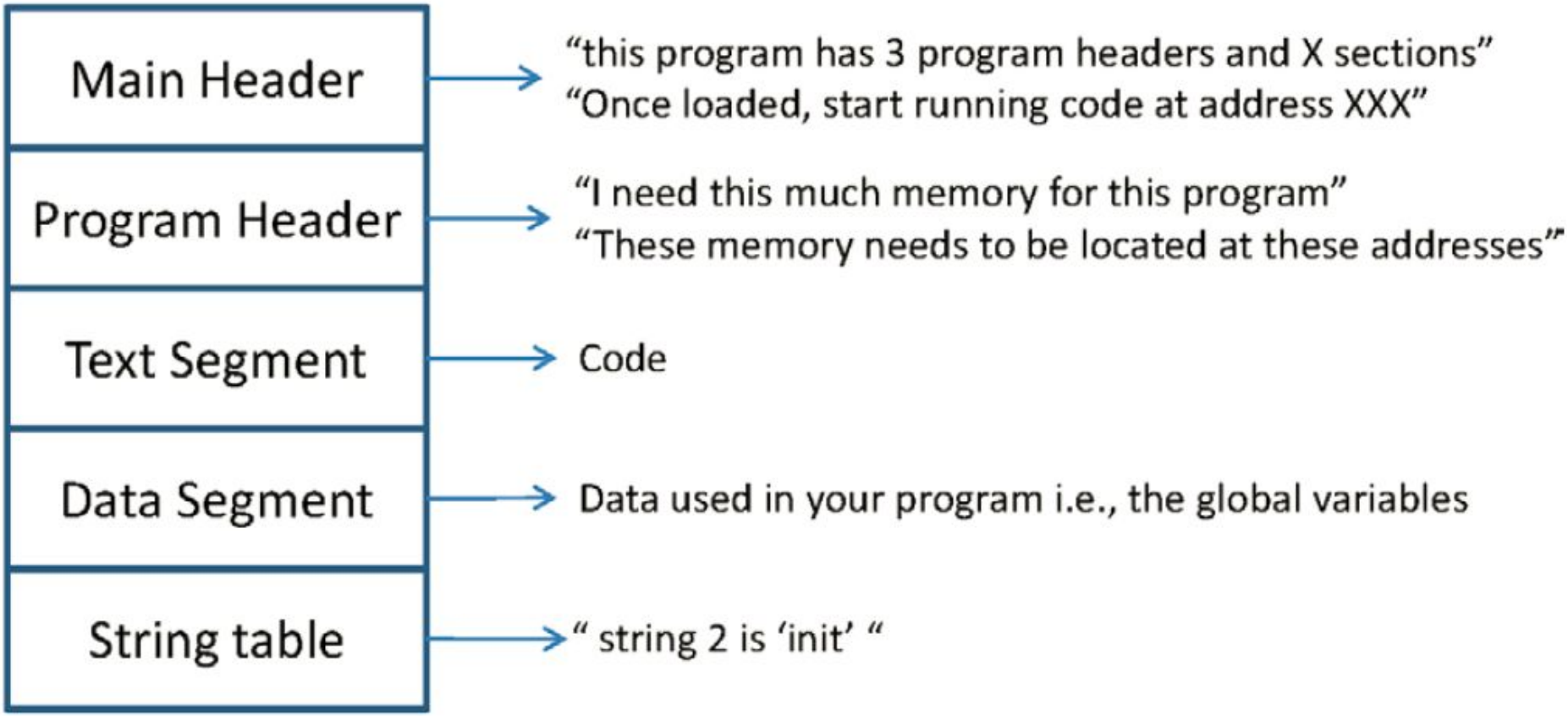
Copyright © 2021 DigiPen Institute Of Technology

Loading a Program

- Need a program called a **loader**
  - Able to read the executable format
  - Copy the text and data segments into the correct memory addresses.
  - Allocate space for Stack and heap for the new running program
  - Set the Program Counter value to the address of the starting instruction of the loaded program.
  - The newly loaded program runs

Copyright © 2021 DigiPen Institute Of Technology

Layout of an executable



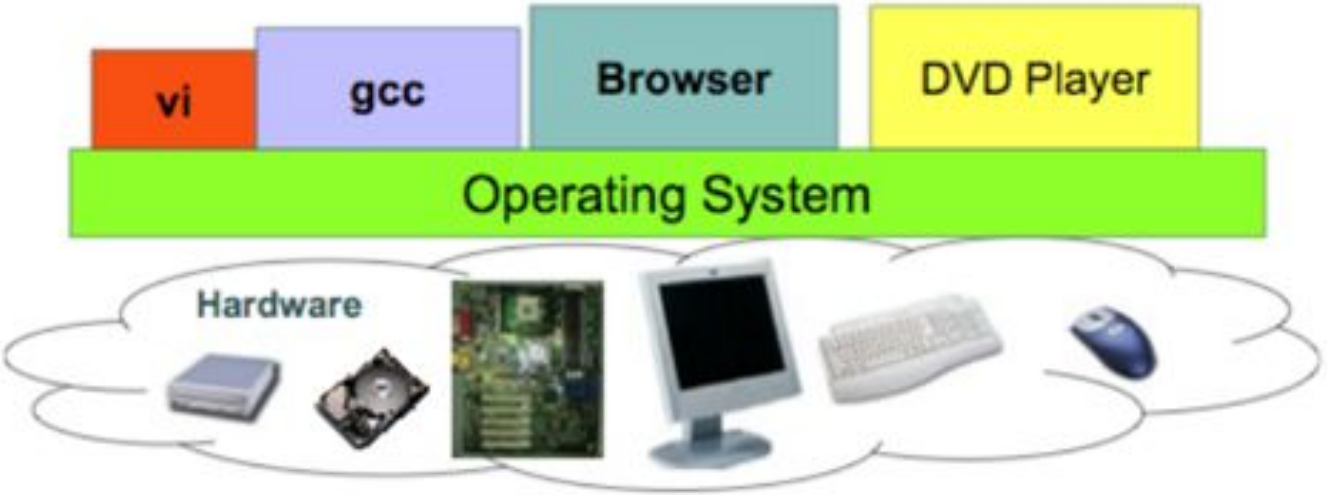
Copyright © 2021 DigiPen Institute Of Technology

Loader

- Chicken and egg problem
- Who loads the loader?
- Need to talk about the boot sequence:
  1. Power-On
  2. Run Basic I/O System (BIOS)
  3. BIOS load and run Master Boot Record (MBR) in HDD/SSD
  4. MBR loads an OS Boot Loader (BL)
  5. BL loads and run OS

Copyright © 2021 DigiPen Institute Of Technology

What is an OS?



- OS is a key part of a computer system, an interface between your hardware and software.
- OS understands how computers work under the hood
- Combine languages, hardwares, data structures, and algorithms.

Copyright © 2021 DigiPen Institute Of Technology

What does an OS do?

- Manages execution of programs/applications
- Abstraction for programmers
  - API removes need for low-level details
- Portability
- Resource Management
  - Virtualization
- Security

Copyright © 2021 DigiPen Institute Of Technology