Computer Science & Programming
Lecture 1: Computer Organization

Stephen Huang

January 23, 2023

UNIVERSITY of HOUSTON

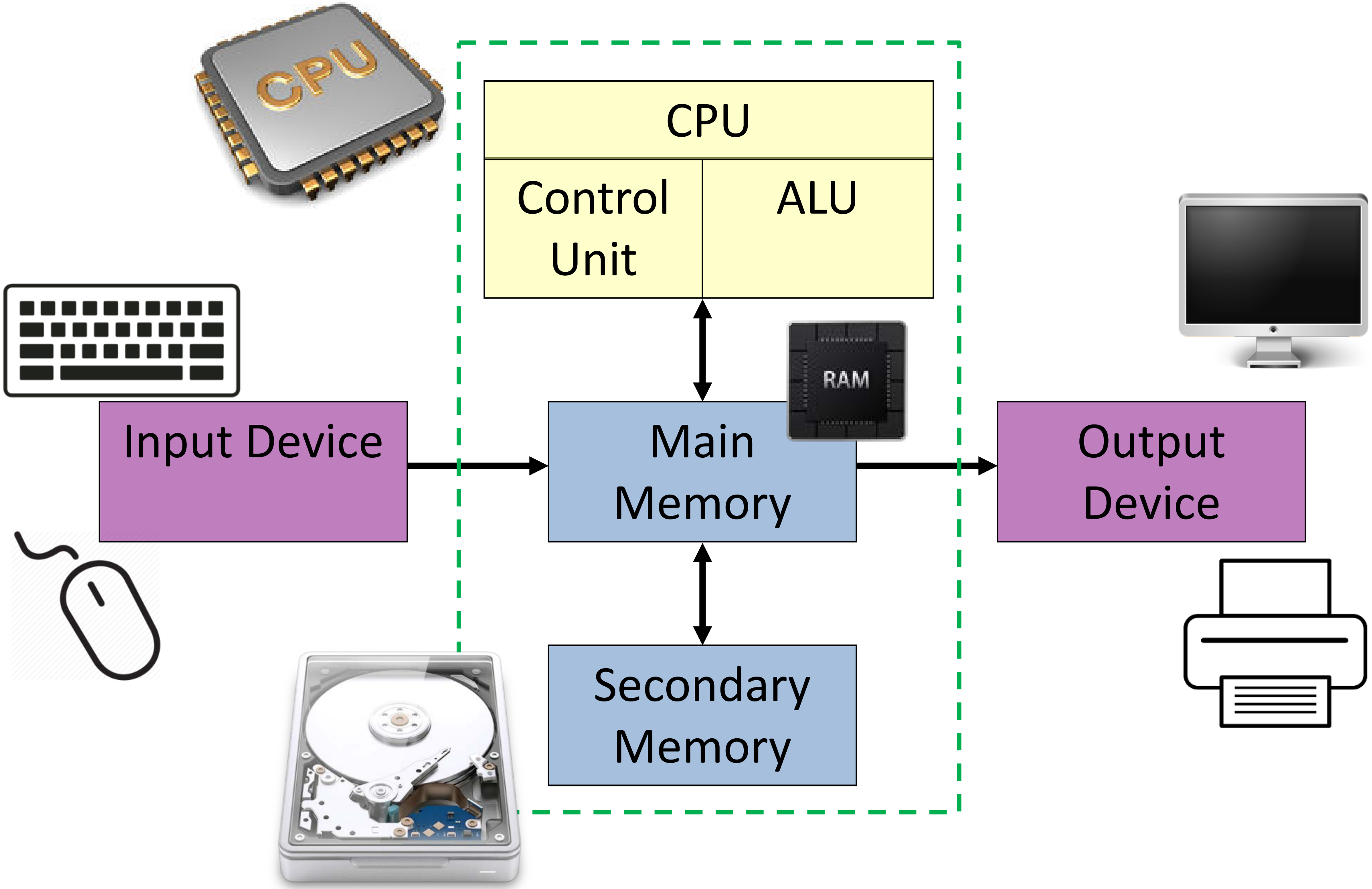
Contents

1. Computer Organization
2. Representation of Information *

1. Computer Organization

- A computer has the following functional components:
  - CPU (Central Processing Unit)
    - Control Unit
    - Arithmetic and Logic Unit (ALU)
  - Main Memory
  - Secondary Memory
  - Input Devices
  - Output Devices

Components of a Computer



CPU

Control Unit

ALU

Input Device

Main Memory

Output Device

Secondary Memory

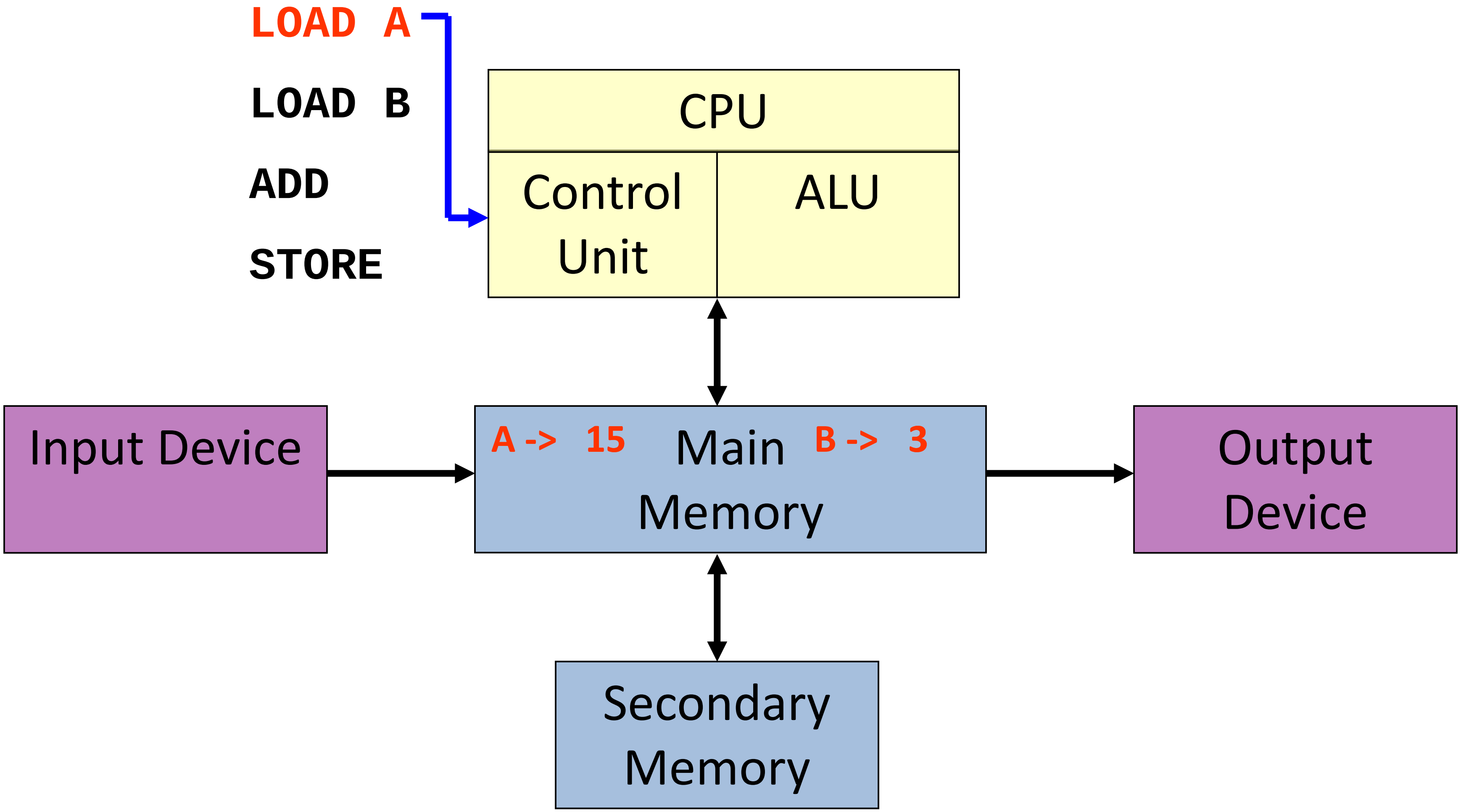
The Control Unit

- The control unit of the CPU controls all operations of the computer.
- It works in a cycle. Each cycle
  - fetches the next instruction of the program currently being executed,
  - interprets (decodes) the instruction to determine what should be done,
  - executes the instruction.
- This cycle is sometimes summarized as fetch/decode/execute.

CPU: Control Unit

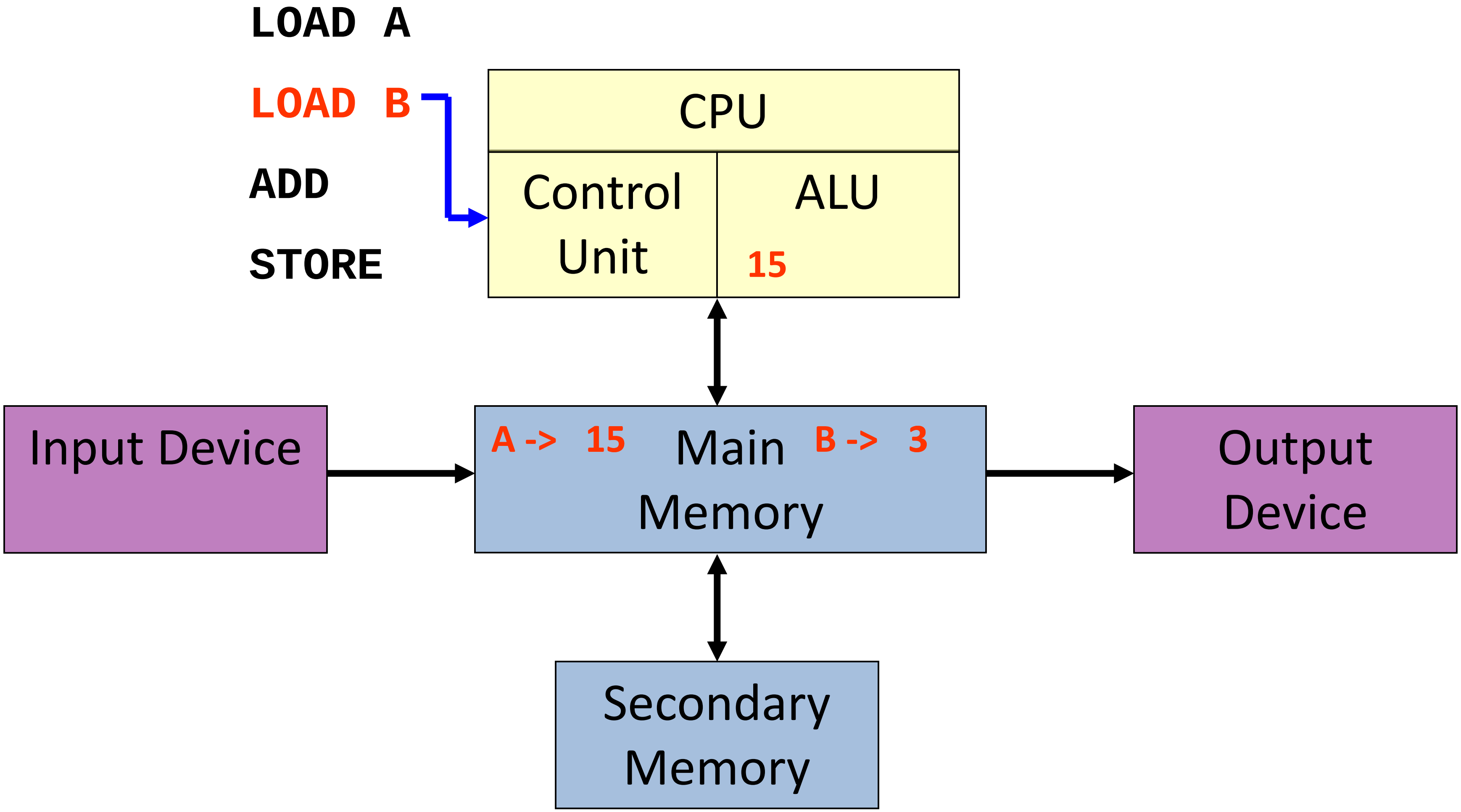
- To add two numbers, the control unit does the following:
  - Load the two numbers to ALU from the memory
  - Perform the addition in ALU
  - Copy the result to some specified memory cell

Example: Fetch/Decode/Execute



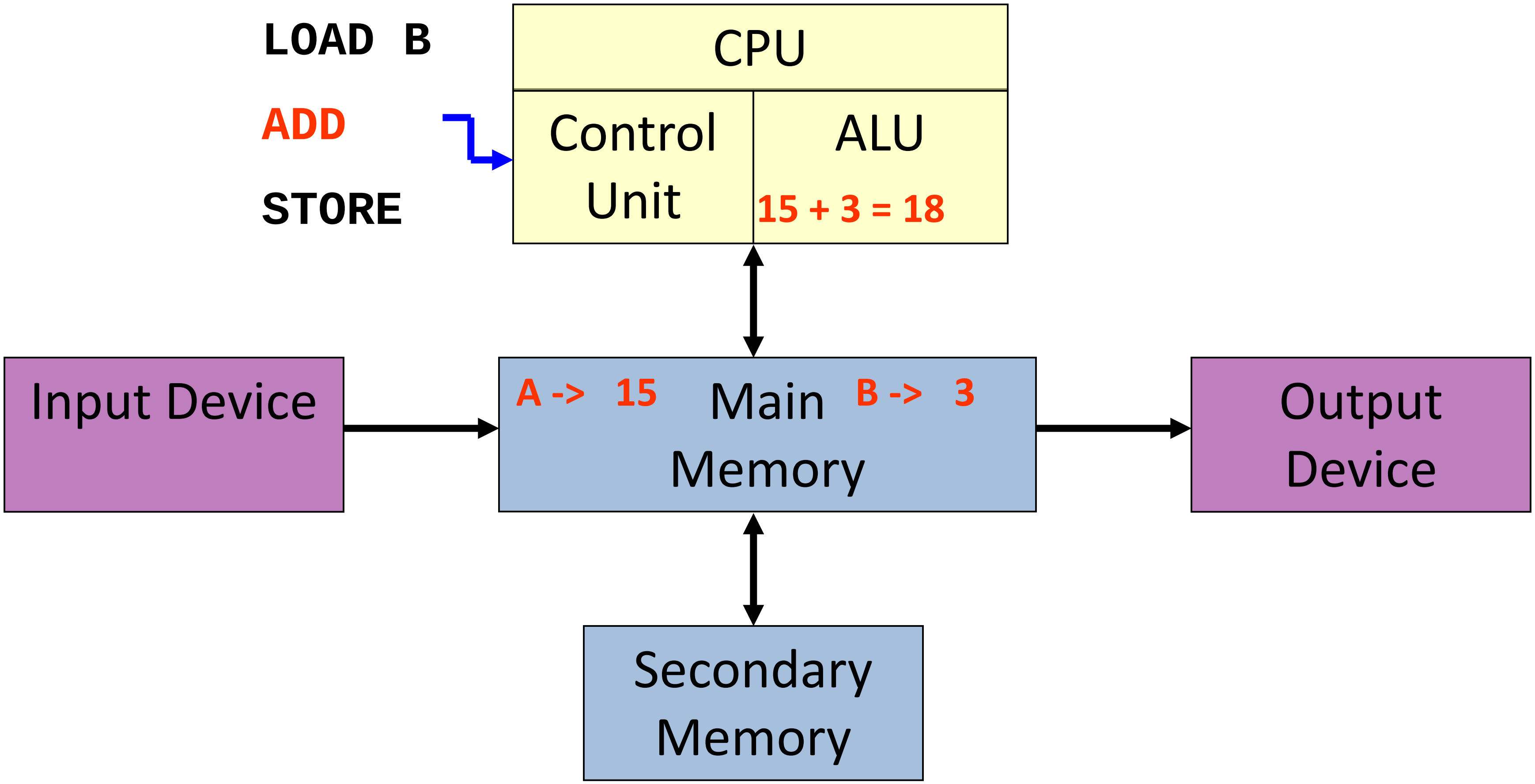
UNIVERSITY of HOUSTON

Example: Fetch/Decode/Execute



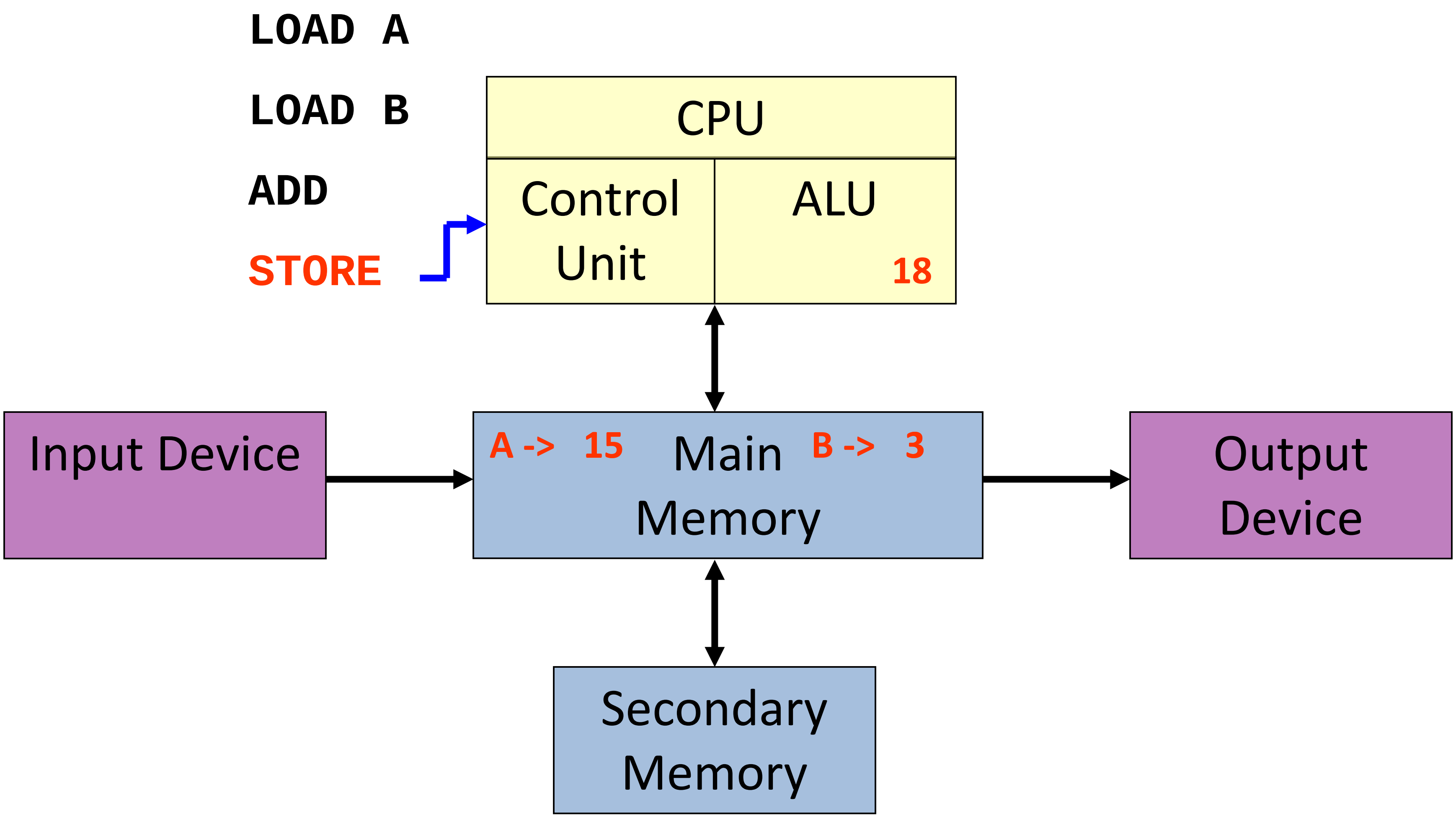
Example: Fetch/Decode/Execute

```
LOAD A
LOAD B
ADD
STORE
```



UNIVERSITY of HOUSTON

Example: Fetch/Decode/Execute

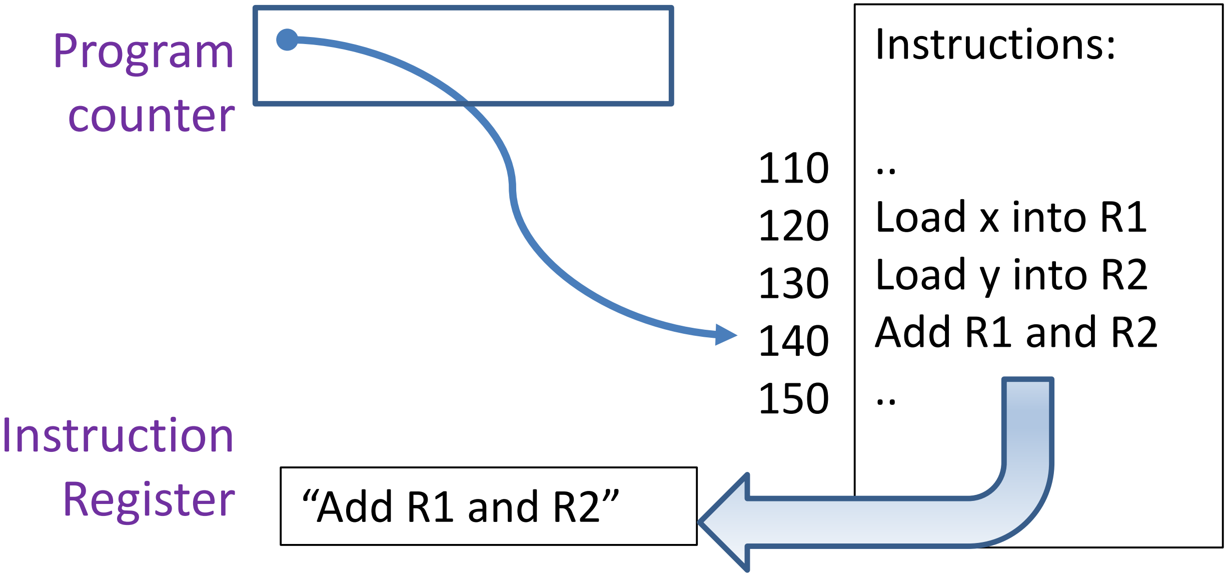


ALU

- The arithmetic and logic unit, ALU**,** is responsible for performing
  - Arithmetic calculations involving addition, subtraction, multiplication, and division and
  - Logical operations such as the test i < n.
- The ALU uses arithmetic registers to store the numbers involved in a calculation or logical operation.

The Control Unit

- The control unit (ALU) uses
  - a program counter to store the address of the next instruction to be fetched and
  - an instruction register to keep the current instruction being decoded and executed.



Program counter

Instructions:

110	..
120	Load x into R1
130	Load y into R2
140	Add R1 and R2
150	..

Instruction Register

"Add R1 and R2"

The Control Unit

- The faster this basic cycle can be performed, the faster the computer can execute a program.
- The speed of a computer is usually stated as some number of
  - Megahertz (million cycles per second) or
  - Gigahertz (billion cycles per second).

Intel's 12th Gen i9 processor can hit 5.5 GHz on up to two cores.

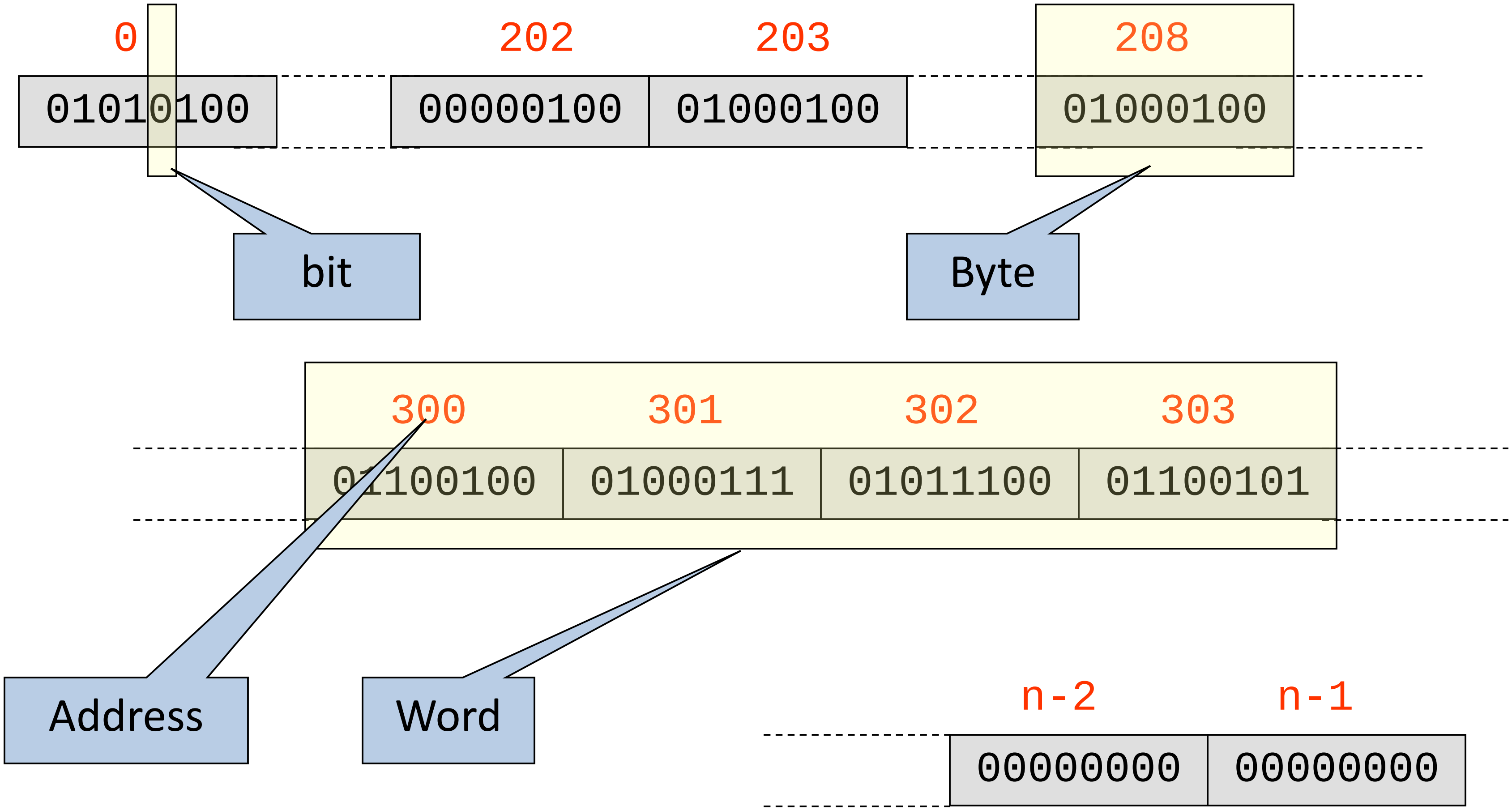
The Main Memory

- Main memory consists of memory cells, each of which can store a sequence of binary digits (bits) that represent
  - an instruction, or
  - a data value.
- The memory necessary to store a single character is called a byte.
- Each byte consists of 8 bits.

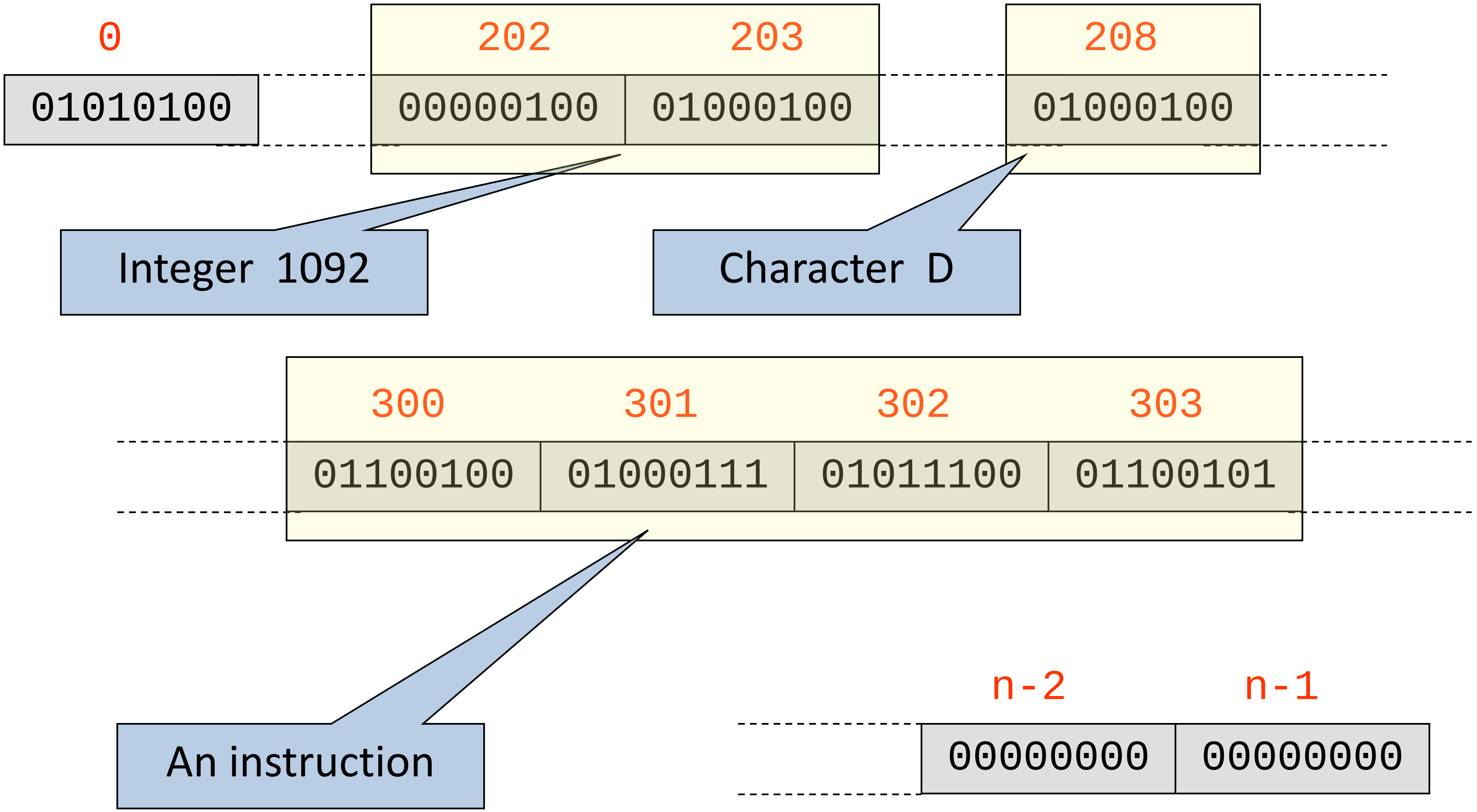
Memory Address

- The bytes of memory are numbered sequentially from 0 to n-1, where n is the number of bytes in the computer's main memory.
- This number, referred to as the address of the byte, serves to identify the memory location.
- Memory addresses are also used in some instructions. The information stored in memory can be sensed (read) by the computer change.
- However, if the computer stores new information in memory, it destroys the old information in it.

Main Memory


0
01010100
202
00000100
203
01000100
208
01000100
bit
Byte
300
01100100
301
01000111
302
01011100
303
01100101
Address
Word
n-2
00000000
n-1
00000000

Main Memory



0

01010100

202 203

00000100 01000100

208

01000100

Integer 1092

Character D

300 301 302 303

01100100 01000111 01011100 01100101

An instruction

n-2 n-1

00000000 00000000

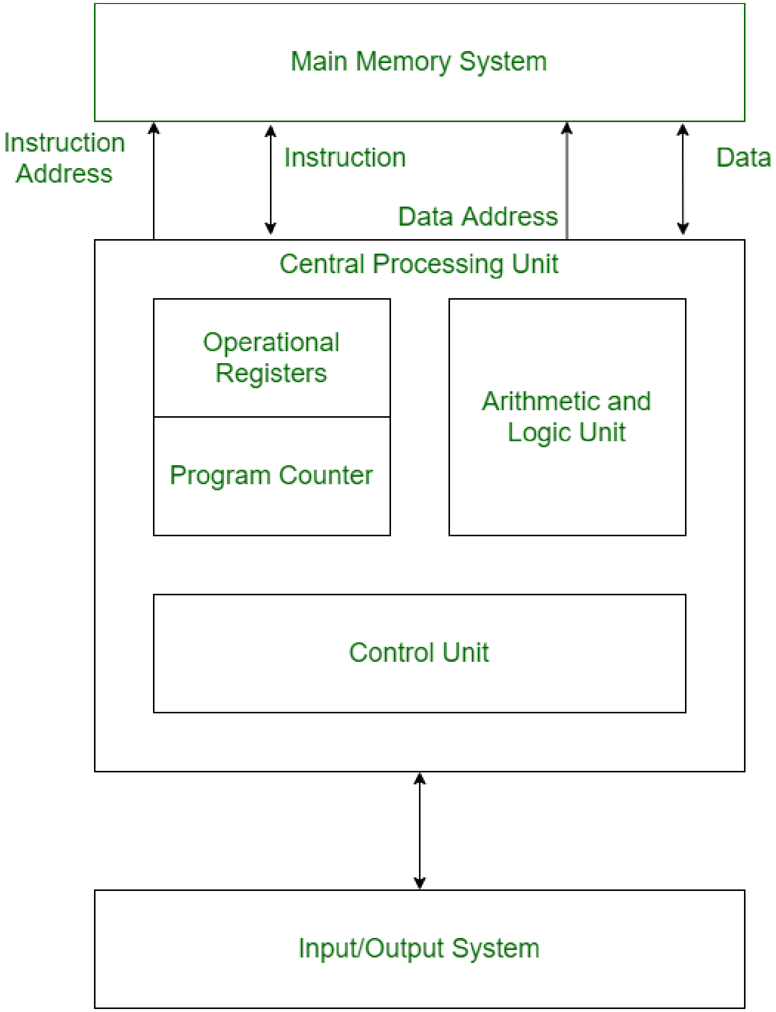
Addresses

- High-level programming languages like C++ or Python use variable names to refer to values stored in memory.
- This makes referring to stored values much more manageable than remembering an address.
- For example, if a programmer in C++ declares a variable, say ***i***, to be of type integer, then the compiler associates this name with some bytes of memory, say bytes 202-203 in the above picture.

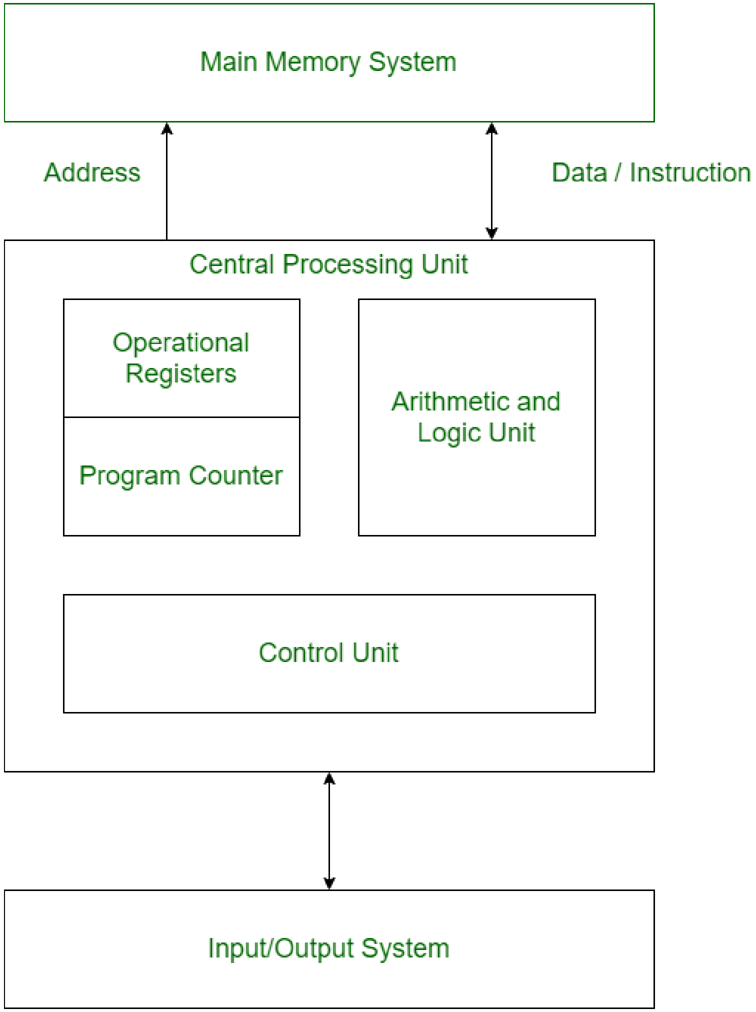
Addresses

- Then instead of referring to its address, we can use a meaningful variable name in the program.
- However, some C++ instructions that use addresses (pointers), and you need to be familiar with the notion of a memory address.
- Luckily, there are no pointers in Python.

Von Neumann Architecture



Harvard Architecture

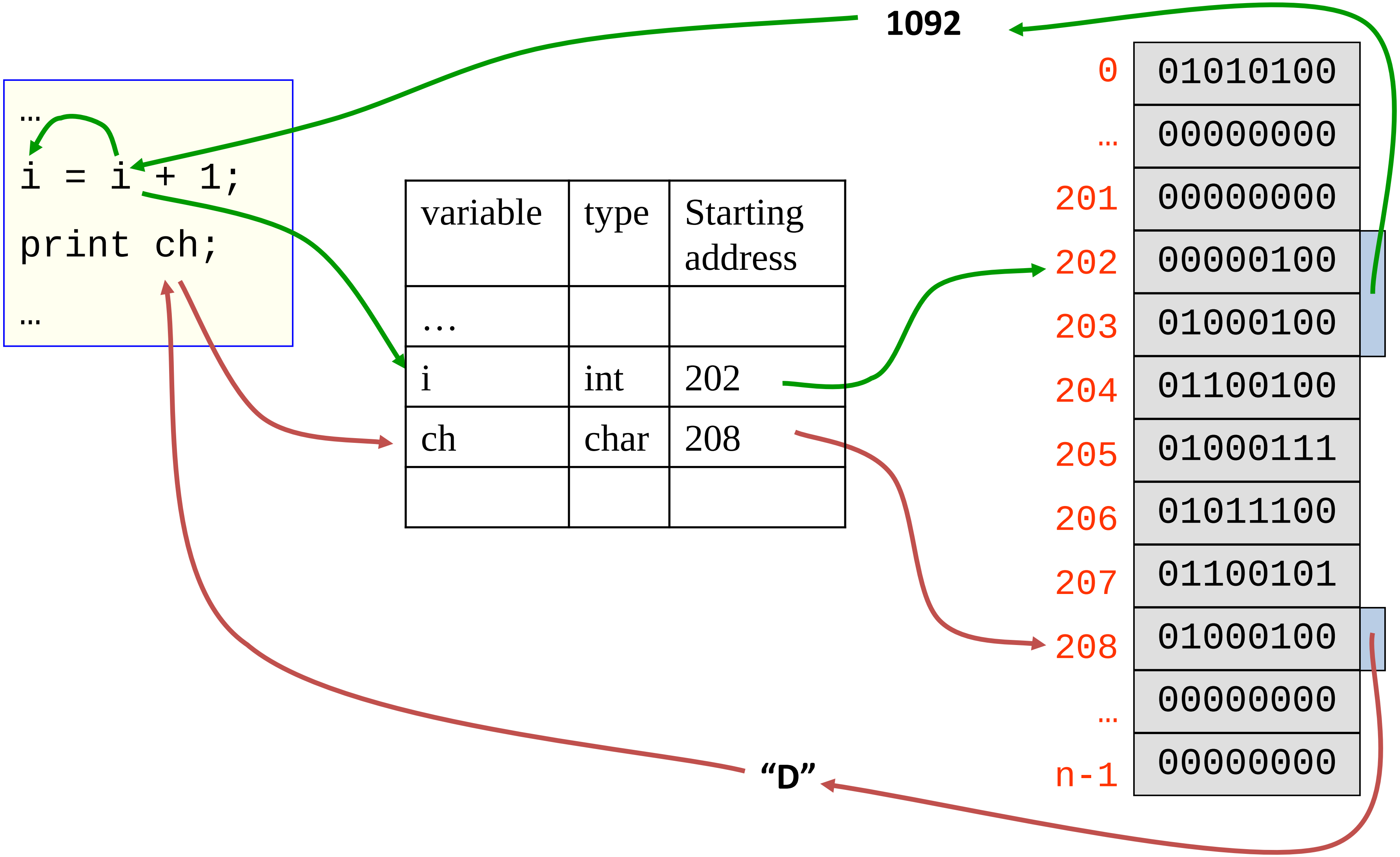


Von Neumann Architecture

Von Neumann Architecture

- John von Neumann made modern computing possible by bridging the gap between instructions and data.

Memory Addresses



```
…
i = i + 1;
print ch;
…
```

variable	type	Starting address
…		
i	int	202
ch	char	208

0	01010100
…	00000000
201	00000000
202	00000100
203	01000100
204	01100100
205	01000111
206	01011100
207	01100101
208	01000100
…	00000000
n-1	00000000

UNIVERSITY of HOUSTON

Secondary Memory

- Secondary memory provides permanent and large-scale storage of information.
- The most common secondary storage devices are magnetic disks that record information in a magnetic form.
  - Floppy disks
  - Hard disks
  - CD-ROMs
  - Flash memory

Input Devices

- The input devices allow information to be inputted into the computer.
- The most common input devices are the keyboard and mouse.

Output Devices

- The output devices allow information to be outputted from the computer to the user.
- The most common output devices are the monitor and printer.

2. Representation of Information

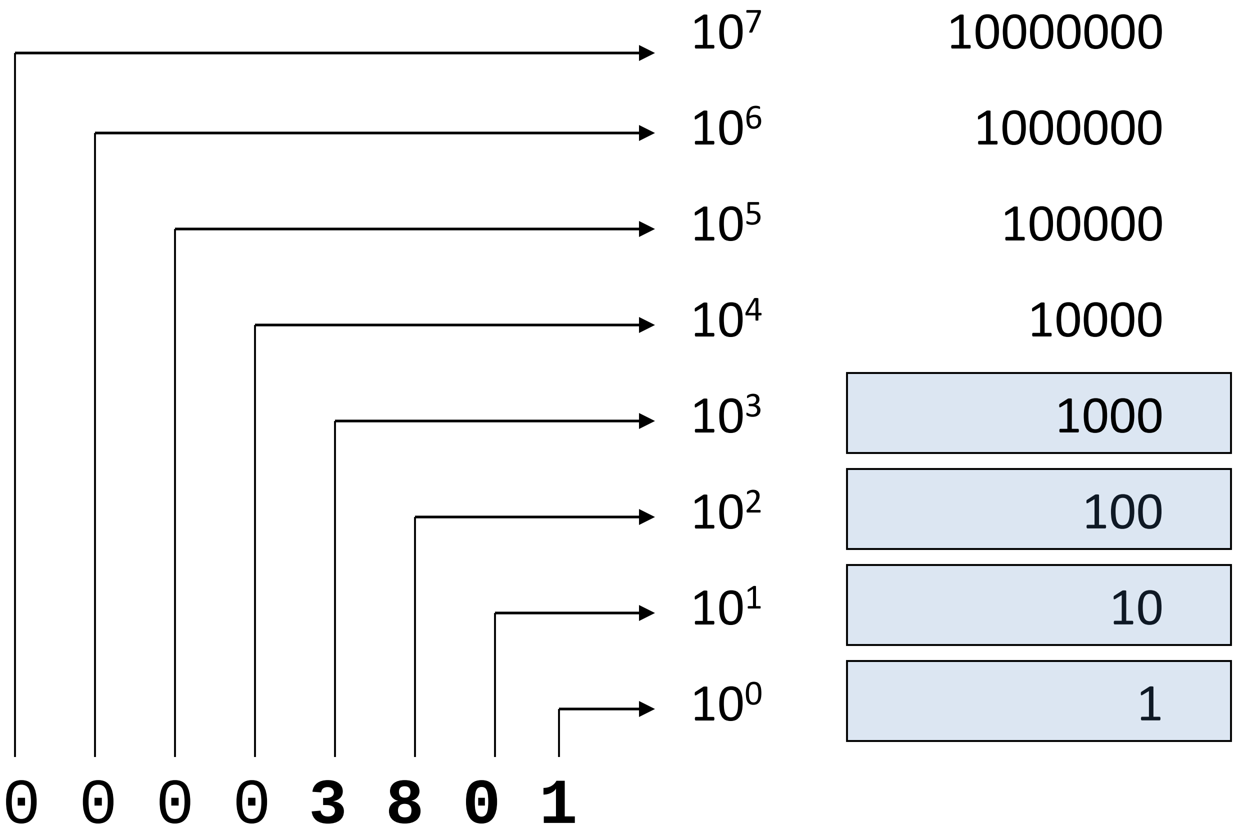
- All information stored in a computer's memory is binary, i.e., a sequence of binary digits (bits).
- This may include:
  - Numbers (integer, floating-point numbers, etc.)
  - Boolean Values (True, False)
  - Strings
  - Addresses (memory)
  - Instructions

Numbers

- We are used to writing numbers in decimal.
  - Decimal (integer) numbers are written as a sequence of decimal digits, 0-9.
  - The position of a digit determines what it stands for.
  - The rightmost digit is thought of as multiplied by 1, the next digit is thought of as multiplied by 10, the next by 100, etc.
  - In other words, each digit from right to left is multiplied by the next power of 10.

Number Systems

For example, the decimal integer 3801 stands for the number 1*1 + 0*10 + 8*100 + 3*1000.

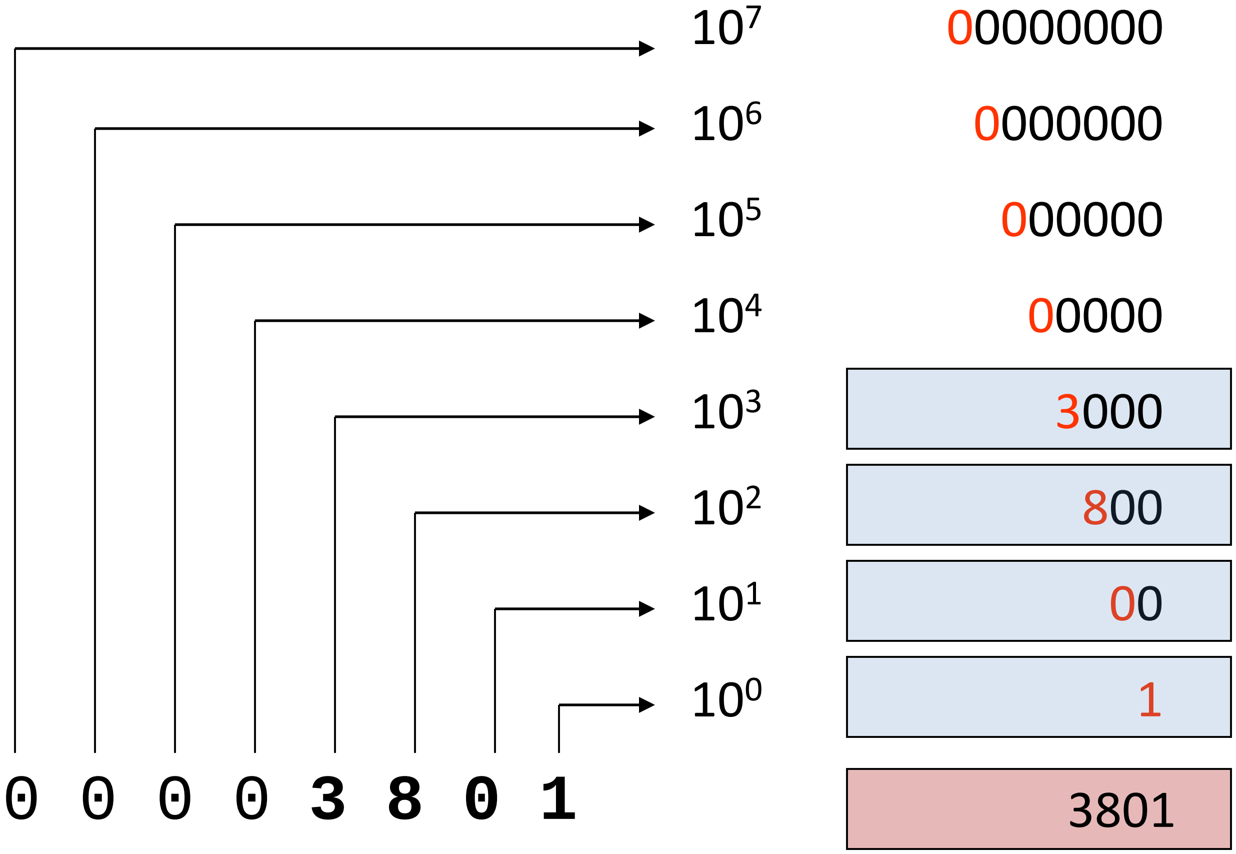


10^7	10000000
10^6	1000000
10^5	100000
10^4	10000
10^3	1000
10^2	100
10^1	10
10^0	1

`0 0 0 0 3 8 0 1`

Example: Decimal Number

For example, the decimal integer 3851 stands for the number 1*1 + 0*10 + 8*100 + 3*1000.



10^7 00000000

10^6 0000000

10^5 000000

10^4 00000

10^3 3000

10^2 800

10^1 00

10^0 1

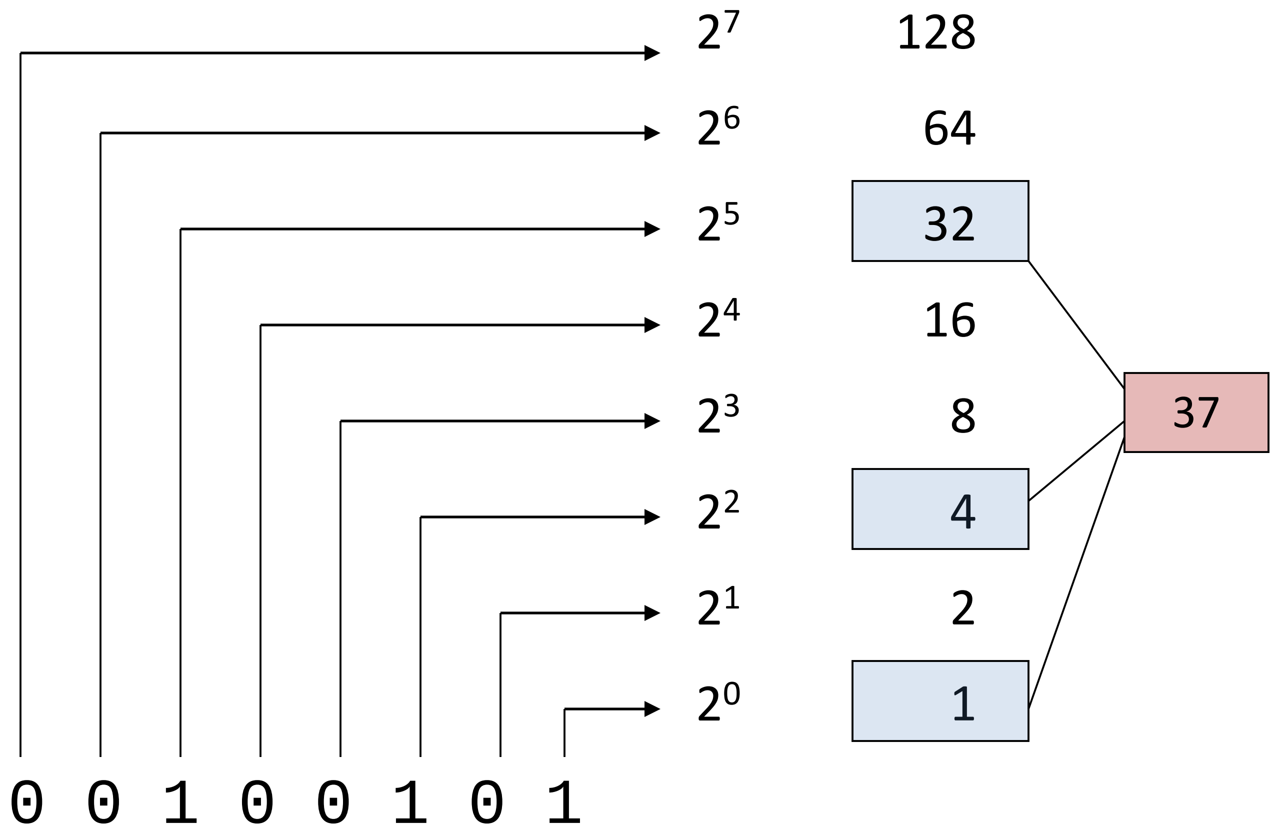
0 0 0 0 3 8 0 1

3801

Binary Numbers

- The binary number system represents an integer by a sequence of bits, 0, 1.
- The rightmost bit is thought of as multiplied by 1, the next bit from the right is multiplied by 2, the next by 4, etc.
- For example, the binary integer 100101 stands for the number 1*1 + 0*2 + 1*4 + 0*8 + 0*16 + 1*32 which is 37 in decimal.
- In a computer that stores integers in 2 bytes (or 16 bits), this number would be stored as 00000000 00100101.

Example: Binary Number



UNIVERSITY of HOUSTON

Decimal to Binary

- How do you convert a decimal integer into binary?
- One method is to divide the decimal number by 2 repeatedly.
- The remainder of the division is the next bit from right to left.
- The quotient is then used in the subsequent division.
  - The binary representation of the decimal number 37 is 00100101.
  - If this is stored in 2 bytes, then the leading bits are all 0, giving the result 00000000 00100101.

Decimal to Binary

Number (Divide by 2)	Quotient	Remainder
37/2	18	1
18/2	9	0
9/2	4	1
4/2	2	0
2/2	1	0
1/2	0	1
0	0	0
0	0	0

Stop

00100101

Summary

- Computer organization.
- Memory address.
- Binary numbers.
  - Converting between binary and decimal
- Variables.

Expected Outcomes

- Take a decimal number and convert it to a binary number.
- Take a binary number and convert it to a decimal number.

- Note: We only showed non-negative integer numbers in this lecture.