

# Machine Called Computer

## Part 4: Data

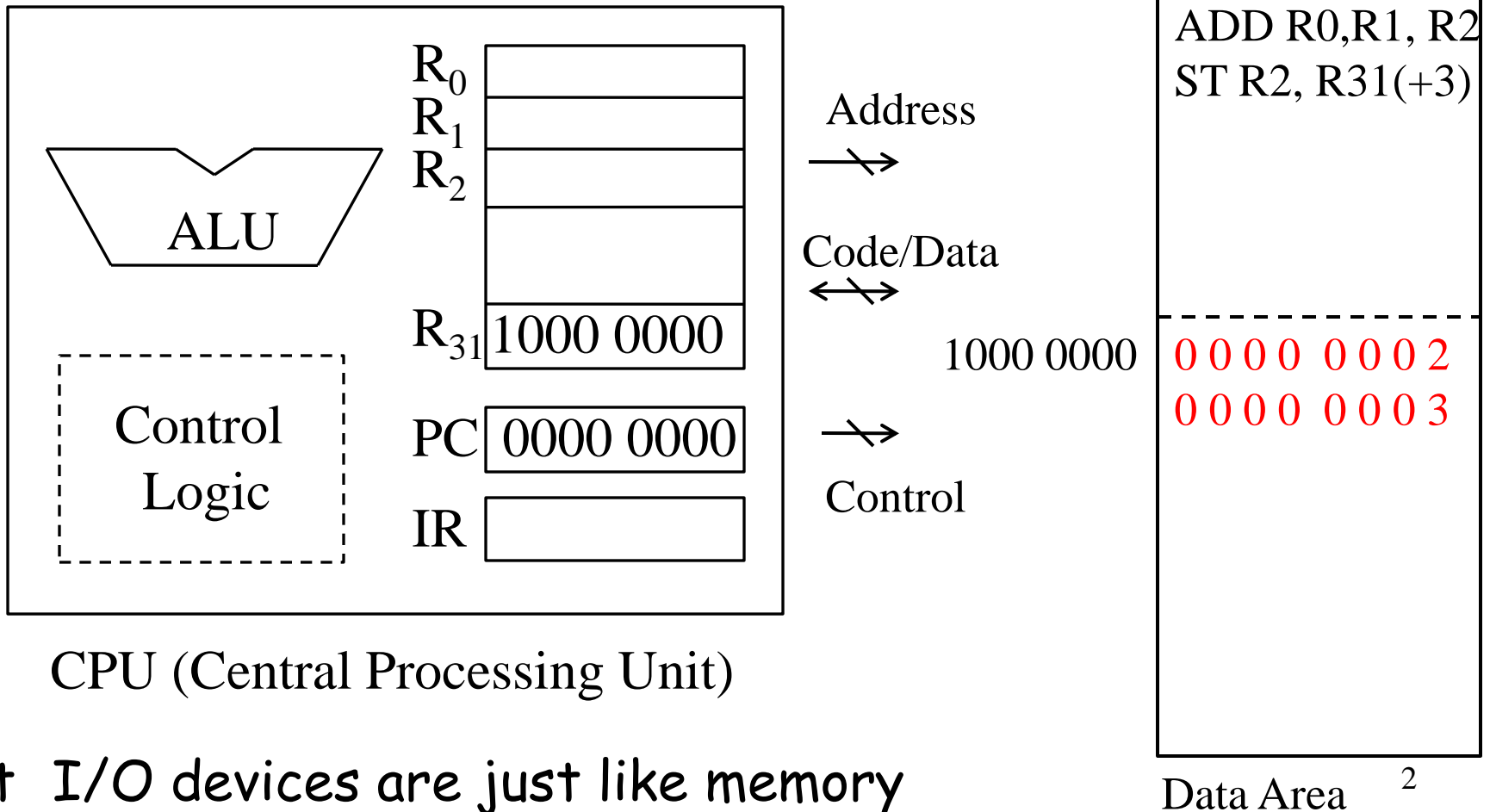
### Reference:

1. Fundamentals of Computer Science, Forouzan and Mosharraf

# Machine Called Computer

## ❑ Data in binary form

- Program and address as well



# Program vs. Data

- ❑ Computation or running programs
  - Data
    - Data representation
  - Code (or program or instructions)
    - Data manipulation
- ❑ Which is more fundamental, data or code?

# Evolution of Data

- ❑ Evolution of computer applications
- ❑ Scientific computing
  - Solve differential equations
  - Data: **numbers**
    - Integer, floating-point numbers (e.g.,  $6.02 * 10^{38}$ )
- ❑ Business computing (e.g., IBM)
  - Database
  - Data: **characters or text** (ASCII, Unicode)
- ❑ Internet applications
  - Multimedia data: **audio, image, video**

Data, program, address in binary form

- Focus on data

- Numeric data first

(meaning of data in 1945)

- Integer first

# Number Systems

## ❑ Positional number systems

- Position of symbol determine value it represent
- Decimal number system
  - $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ 
    - † Concept more than 5000 years old
- Binary number system
  - $\{0, 1\}$ 
    - † Binary digit or bit
    - † Concept more than 2000 years old

# Decimal Integers

$10^2$	$10^1$	$10^0$	Weight
3	2	5	Number
$N = + 3 \times 10^2 + 2 \times 10^1 + 5 \times 10^0$			Value

# Binary Integers

$2^4$	$2^3$	$2^2$	$2^1$	$2^0$	Weight
1	0	1	1	0	Number

$$N = + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \quad \text{Value}$$

- ❑ “10110” in binary: 5-bit notation
  - Bit: binary digit
  - Equivalent decimal value  $N = 16 + 4 + 2 = 22$
- ❑ As natural as decimal notation
  - Only need more digits



## Converting $22_{10}$ into Binary Integer

$$22 = 2 \times 11 \quad \text{remainder } 0$$

$$11 = 2 \times 5 \quad \text{remainder } 1$$

$$5 = 2 \times 2 \quad \text{remainder } 1$$

$$2 = 2 \times 1 \quad \text{remainder } 0$$

$$1 = 2 \times 0 \quad \text{remainder } 1$$

10110

010110

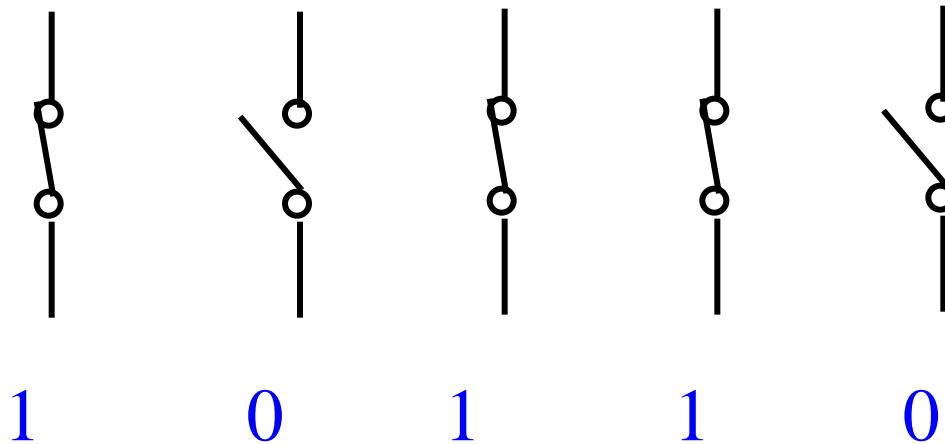
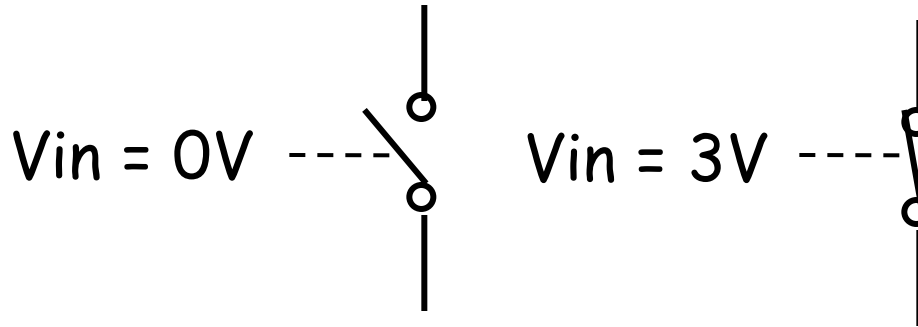
$$22 = 2 \times (2 \times (2 \times (2 \times (2 \times 0 + 1) + 0) + 1) + 1) + 0$$

1  
2  
5  
11

$$= 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0$$

# Why binary numbers in computer?

- ❑ Transistor (underlying hardware)
  - 3-terminal digital switch: two stable states (ON, OFF)



# Why do we not see binary numbers?

## ❑ Software

- Translation between human users and machine

Human Users

---

Human-friendly form

Software (OS, Application)

---

Binary form

Machine called computer

# (Unsigned) Binary Integers

- ❑ Range of expression: simply need more bits than decimal
  - 1-bit: 0 to 1
  - 2-bit: 00 to 11 (0 to 3 in decimal)
  - 4-bit: 0000 to 1111 (0 to 15 in decimal)
  - 8-bit ("byte"): 0000 0000 to 1111 1111 (0 to 255<sub>10</sub>)
  - 16-bit: 0 to  $2^{16} - 1$ 
    - 64K (65,536)
  - 32-bit: 0 to  $2^{32} - 1$ 
    - 4G (4,294,967,296)

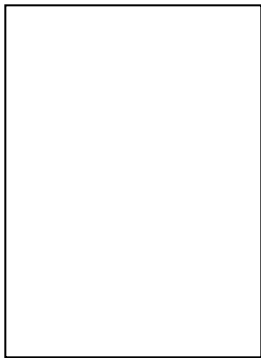
# CS-related Prefix in Metric System

			CS	CS	
yotta	Y	$10^{24}$	$2^{80}$		septillion
zetta	Z	$10^{21}$	$2^{70}$		sextillion
exa	E	$10^{18}$	$2^{60}$		quintillion
peta	P	$10^{15}$	$2^{50}$		quadrillion
tera	T	$10^{12}$	$2^{40}$		trillion
giga	G	$10^9$	$2^{30}$	1,073,741,824	billion
mega	M	$10^6$	$2^{20}$	1,048,576	million
kilo	K	$10^3$	$2^{10}$	1024	thousand

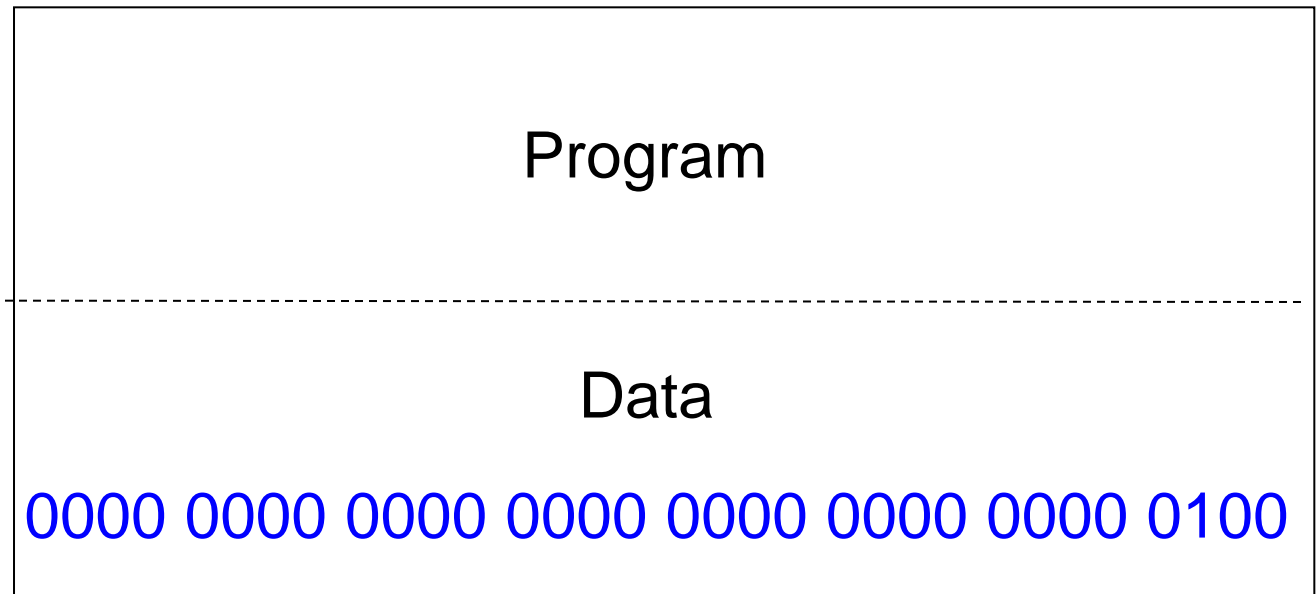
# 32-bit Computer

- ❑ Memory: many slots to store data

CPU



Memory: 32-bit wide main memory



<byte addressing>  
address      8bit  
                 (char

) -

가

.

I/O: Monitor, keyboard, LAN-Internet, ...

# Hexadecimal Notation

- ❑ 0000 0000 0000 0000 0000 0000 0000 0100 (32-bit binary)
  - 00000004 in hex (8 hex digits)
  - Pure mnemonic (for easy human recognition)

Binary	Hex	Binary	Hex	Binary	Hex	Binary	Hex
0000	0	0100	4	1000	8	1100	C, c
0001	1	0101	5	1001	9	1101	D, d
0010	2	0110	6	1010	A, a	1110	E, e
0011	3	0111	7	1011	B, b	1111	F, f

# Memory Model

- ❑ What's stored in a single memory address
  - Bit, byte or what?

0000 0000

0000 0001

0000 0002

0000 0003

.

.

.




# Memory Model - 32-Bit Machine

- 32-bit access, 4열 종대
  - Addresses for "char", "short int", "int"?
  - Address for instruction?

0000 0000	3	2	1	0
0000 0004				
0000 0008				
0000 000C				
.				
.				
.				

# Built-in Data Types in C

## ❑ Integer

- `short` or `long`, `signed` or `unsigned`
- Size is machine dependent (but `int` is at least 16 bits)

## ❑ Character (`char`)

- Single byte capable of holding one character
- Signed or unsigned; in a sense, small integer

## ❑ Floating point number (`float`, `double`)

- Single-/double-precision (IEEE 754; 32/64 bits)

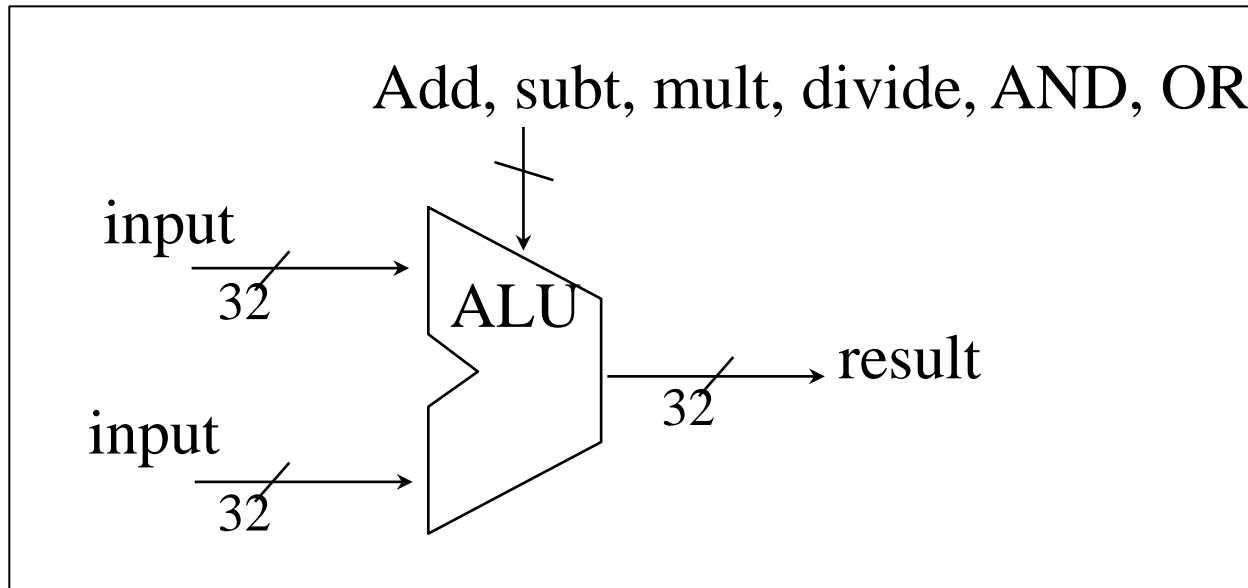
† Boolean, string (implicit in C)

# Binary integer

- Arithmetic:  $+$ ,  $-$ ,  $*$ ,  $/$ 
  - ALU design
- Negative integer
- Real number

# Operation on Numbers and ALU

- ❑ Representation of numbers: binary
- ❑ Operation on numbers
  - ALU (Arithmetic and Logic Unit) in CPU



# Add Binary Numbers

- ❑ Same with decimal add
  - Limitation of 8-bit addition
  - Speed of addition?

[illegible]

# Multiply Binary Numbers

❑ Same with decimal multiply

- Multiply: many shifts and adds (result: double precision)
  - Speed?

- Divide

$$\begin{array}{r} 1100 \\ 1001 \\ \hline 1100 \\ 0000 \\ 0000 \\ 1100 \\ \hline 1101100 \end{array} \quad \begin{array}{l} = 12_{10} \\ = 9_{10} \\ \\ \\ \\ = 108_{10} \end{array}$$

# Binary Representation and ALU

## □ How to represent negative integers

Unsigned binary

$$000 = +0$$

$$001 = +1$$

$$010 = +2$$

$$011 = +3$$

$$100 = +4$$

$$101 = +5$$

$$110 = +6$$

$$111 = +7$$

Two's complement

$$000 = +0$$

$$001 = +1$$

$$010 = +2$$

$$011 = +3$$

$$100 = -4$$

$$101 = -3$$

$$110 = -2$$

$$111 = -1$$

# Binary Rational Numbers (유한소수)

## ❑ Fixed-point representation

$2^2$	$2^1$	$2^0$	$2^{-1}$	$2^{-2}$	Weight
1	0	1	.	1	Number

$$N = + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} \quad \text{Value}$$

❑ Equivalent decimal value  $N = 4 + 1 + 0.5 + 0.25 = 5.75$

❑ ALU not aware of the binary point

- Software perform translation

† Irrational numbers (무한소수): truncation, lose accuracy

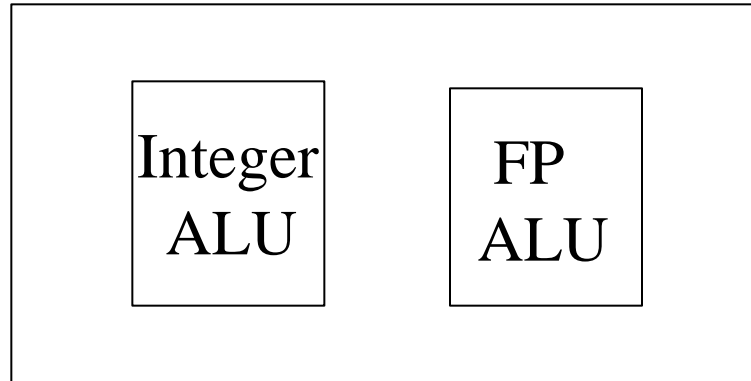


# Floating-Point Numbers

- ❑ What is floating-point representation?
  - Very big and small numbers (i.e., scientific numbers)
    - $1.011 \times 2^{-97}$ ,  $1.101 \times 2^{68}$
    - Need too many bits: store mantissa and exponent
  - Arithmetic becomes quite complex
  - Quite different from binary arithmetic
- ❑ Two types of applications: integer and floating-point
  - Two types of ALU: integer ALU and FP ALU
  - Does your PC have both?
    - What about your smartphone?

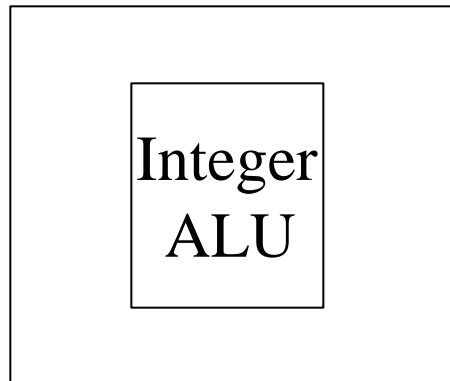
# Integer and FP ALUs

- ❑ Processor for general-purpose computers:



Pentium

- ❑ Processors for embedded systems can be like:



Smaller  
Cheaper  
Low-power

# Non-Positional Number System (skip)

## ❑ Roman numerals

- {I, V, X, L, C, D, M}

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

## ❑ Need specific rules to determine value

- IV:  $5 - 1 = 4$
- XVIII:  $10 + 5 + 1 + 1 + 1 = 18$
- XIX:  $10 + 10 - 1 = 19$
- LXXII:  $50 + 10 + 10 + 1 + 1 = 72$

## ❑ Easy to build ALU?

# Computer Arithmetic

1. ALU – ALU
2. ISA – instruction set
3. data center

- ❑ Can you imagine
  - Many algorithms for addition,
    - Multiplication, division, FP operations as well
- ❑ Focus of building computer in 1945
  - Faster ALU
    - How to represent numbers
- ❑ Computer arithmetic
  - Matured in 1950s and 1960s
- ❑ Today, can buy ALU as IP (Intellectual Property)

Computation:

More than arithmetic

Different types of data:

- Numbers
- Text
- Audio, image, video

# Different Types of Data

- ❑ The term "multimedia"
  - Text, audio, image, video
  - All in binary pattern

Data Type	Program	Representation
Numbers	Math. routine	All in binary pattern
Text	Text editor	
Audio	Voice recorder	
Image	Image recorder	
Video	Video recorder	

# ASCII Code

ASCII code chart, image file in Wikipedia:

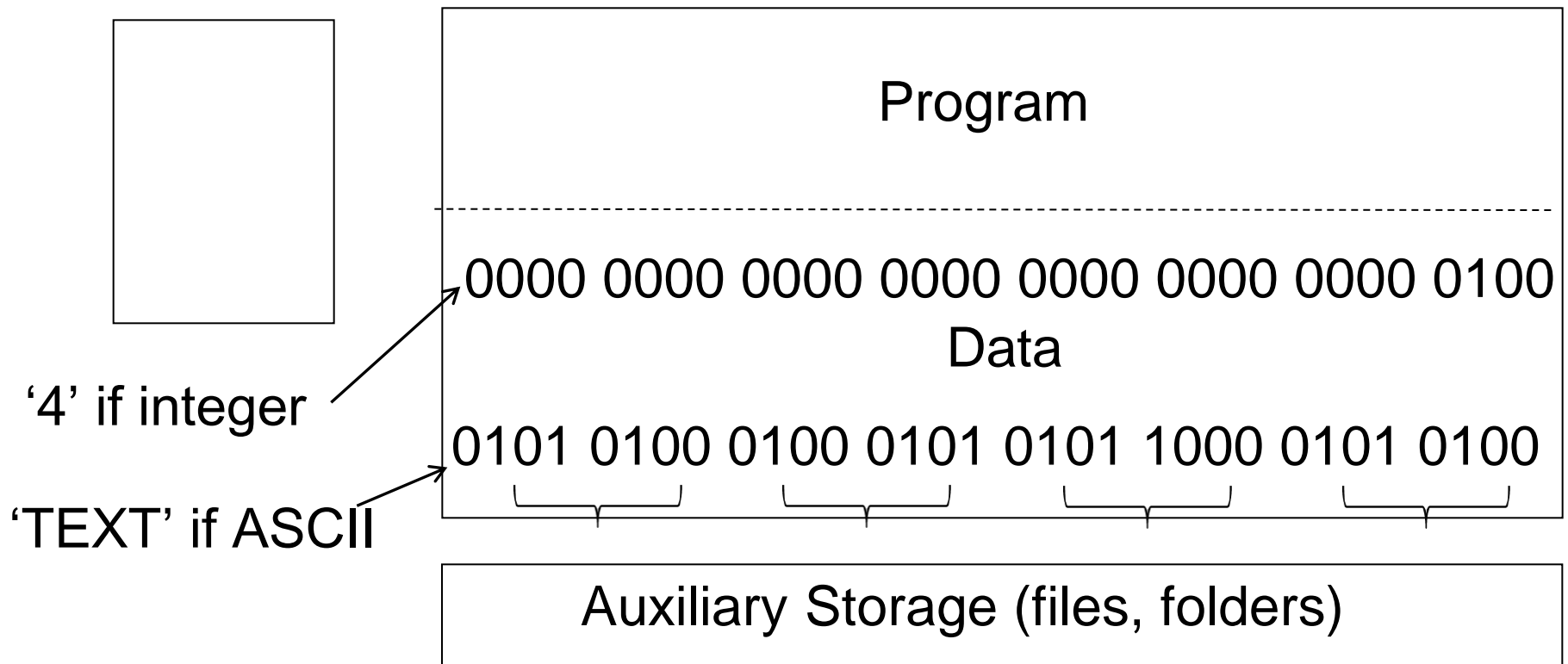
[http://en.wikipedia.org/wiki/File:ASCII\\_Code\\_Chart-Quick\\_ref\\_card.jpg](http://en.wikipedia.org/wiki/File:ASCII_Code_Chart-Quick_ref_card.jpg)

# 32-bit Computer

- Memory: many slots to store data

CPU

Memory: 32-bit wide main memory



I/O: Monitor/keyboard, LAN-Internet, ...

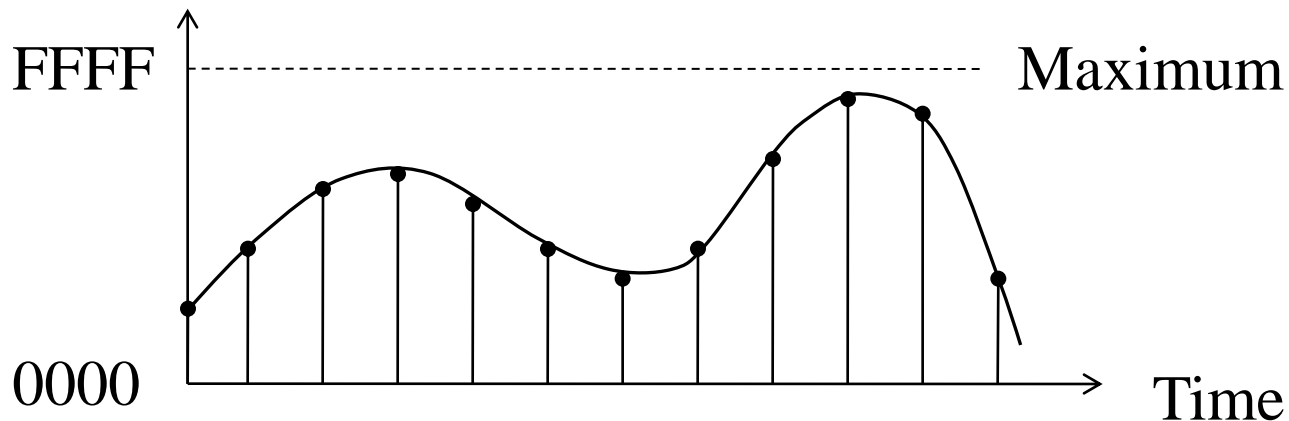


# Representing Text

- ❑ Each character is assigned a unique bit pattern
  - ASCII
    - 7-bit to represent 128 symbols in English text
  - ISO: a number of 8-bit extensions to ASCII
    - To accommodate different language groups
  - Unicode (e.g., Java): 32-bit characters
    - Represent alphabets in world's languages plus symbols
      - † "Internationalization"
    - Characters encoded in 1 byte to 4 bytes

# Storing Audio

- ❑ Speech or music represented by sequence
- ❑ Dominant standard: MP3 (short for MPEG Layer 3)
  - 44100 samples per second
    - Sampling theorem
  - 16 bits per sample: 0000 to FFFF in hexadecimal



# Storing Images

## ❑ JPEG standard

- True color: 24 bits per pixel (R,G,B 각각 8 bits)
- Compress to reduce size

Color	Red	Green	Blue	Color	Red	Green	Blue
Black	0	0	0	Yellow	255	255	0
Red	255	0	0	Cyan	0	255	255
Green	0	255	0	Magenta	255	0	255
Blue	0	0	255	White	255	255	255

## ❑ HD 1080: $(1920 \times 1080) \times 3 \text{ bytes} \approx 6 \text{ MB}$

# Display Resolution

Image file showing display resolution:

[http://en.wikipedia.org/wiki/File:Vector\\_Video\\_Standards4.svg](http://en.wikipedia.org/wiki/File:Vector_Video_Standards4.svg)

# Storing Video

- ❑ Images (or frames) over time
  - e.g., 24 frames per second
- ❑ MPEG (Moving Picture Experts Group) standard
  - Lossy compression
    - † MPEG-2
    - † MP3 (MPEG audio layer 3)
- † Lossless compression
  - ALZip

All in Binary Form:

Address, (Data and Program)

# Meaning of Address (반복)

- ❑ Unique identifier for locations

Address from CPU

16 memory locations

0000

0001

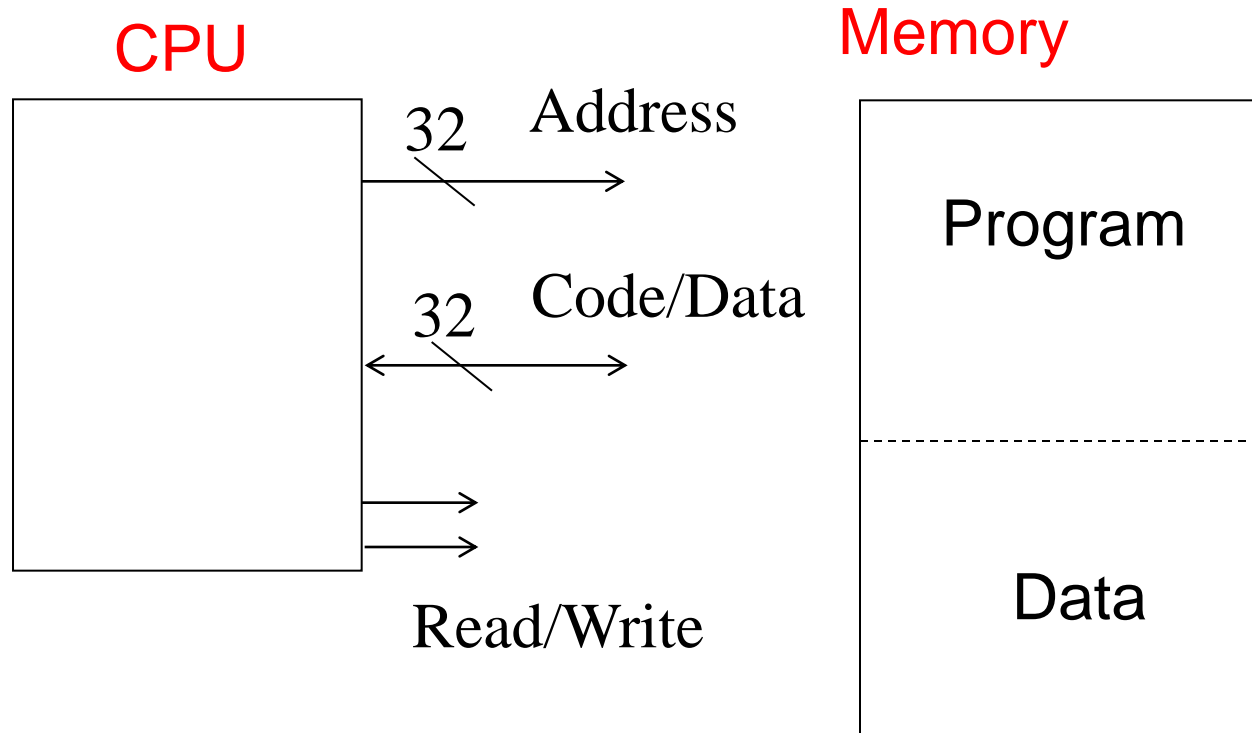
1111


# Number of Address Bits (반복)

- ❑  $256 = 2^8$  memory locations: 8-bit address
- ❑  $64K = 2^{16}$  memory locations: 16-bit address
  - 8-bit microprocessor
- ❑  $4G = 2^{32}$  memory locations: 32-bit address
  - 32-bit processor



# 32-bit Computer (반복)



I/O device 0 (e.g., disk)

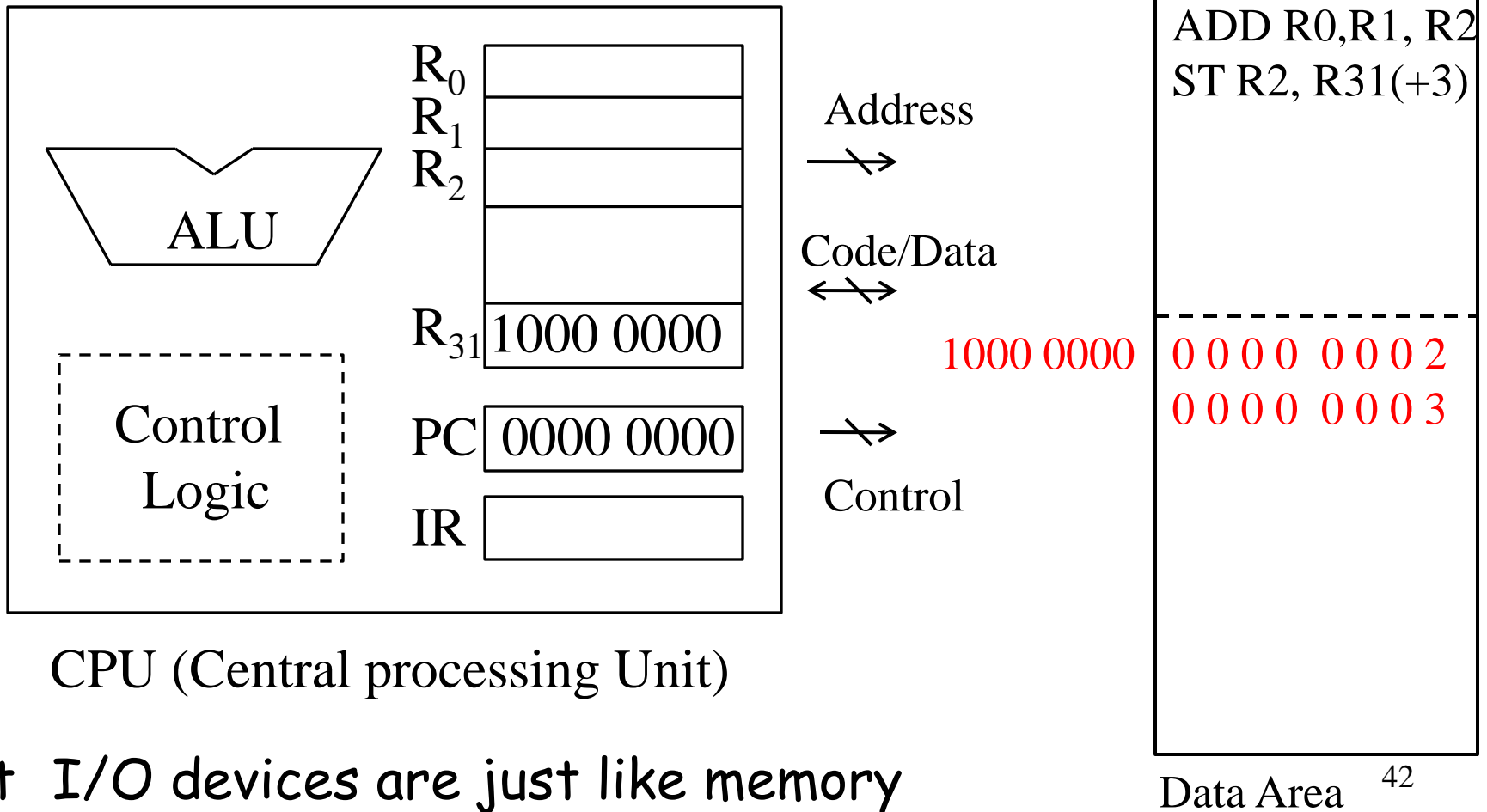
I/O device 1 (e.g., monitor)

- ❑  $4G = 2^{32}$  memory and I/O locations
- ❑ Given address, enable corresponding location

# Machine Called Computer

## ❑ Data, address in binary form

- Program as well



# Summary

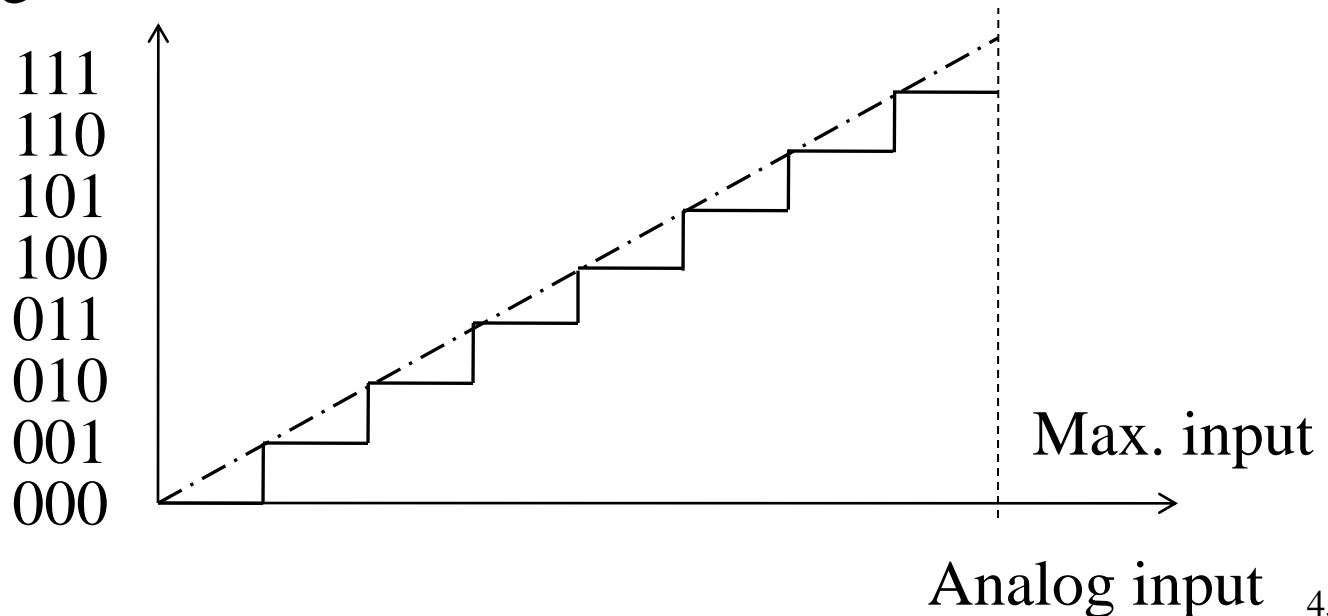
- ❑ Look into “data” first (data and program in computer)
  - Numeric data for computation
    - Binary numbers (binary integers)
      - † Add, subtract, multiply, divide (ALU)
    - Rational and irrational numbers
    - Floating-point numbers
  - Business computing: text data
  - Multimedia data: audio, image, video
- ❑ (Memory and I/O) “address”

To Think About

# Why "Digital" Computer?

- ❑ Nature: continuous differential equation (i.e., analog)
- ❑ Quantization error
  - Can reduce it with more bits
- ❑ Processing and communication error

Digitized info.



# Error Detection and Correction

## ❑ Single-bit parity and error detection

0100 1001  
 data parity (odd or even)

## ❑ Two-dimensional bit parity

data	10101	0	10101	0	01101	0
	10010	1	11010	1	01010	1
	11100	0	11100	0	11100	0
	00100	0	00100	0	00100	0
		parity				
			1-bit error, correctable		4-bit error, undetectable	

† Notion of redundancy (in Korean, English)