

# Computer Programming

## Lecture 1: Introduction to Computers

**Krikamol Muandet**

Department of Mathematics  
Mahidol University  
Bangkok, Thailand

Introduction

Hardware and Software

How Computers Store Data

How a Program Works

## Introduction

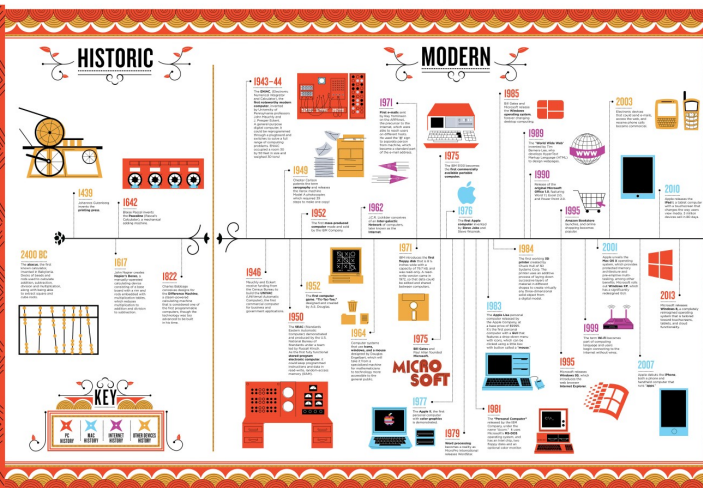
Hardware and Software

How Computers Store Data

How a Program Works

# THE EVOLUTION OF COMPUTERS

PC NINJA TRAVELS THROUGH TIME, revealing the history of how computers became our sidekicks. From sliding pebbles on a simple machine to swiping your fingers across a touchscreen, technology has transformed radically!



# General-Purpose Computers

COMPUTER

VS

CALCULATOR



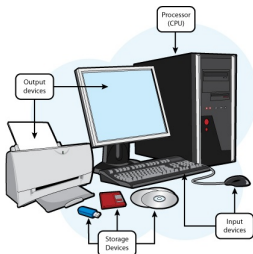
Introduction

Hardware and Software

How Computers Store Data

How a Program Works

# Hardware and Software



## Hardware

- ▶ Central processing unit (CPU)
- ▶ Main memory (RAM)
- ▶ Secondary storage devices
- ▶ Input devices
- ▶ Output devices



## Software

- ▶ Operating systems (OS)
- ▶ Utility programs
- ▶ Software development tools
- ▶ Application software

# Central Processing Unit (CPU)





# Central Processing Unit (CPU)

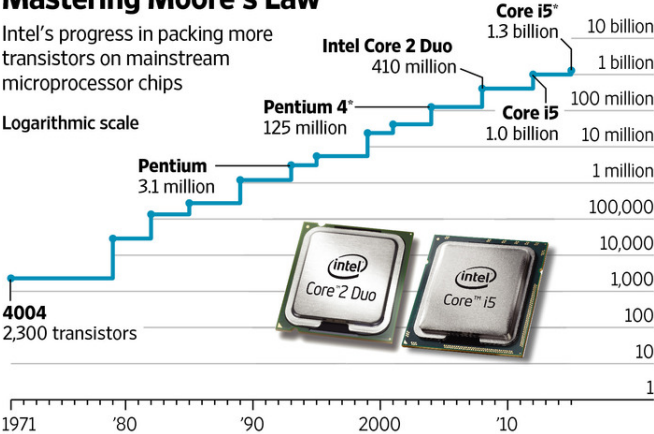


# Moore's Law

## Mastering Moore's Law

Intel's progress in packing more transistors on mainstream microprocessor chips

Logarithmic scale



\*Upgraded versions of prior models  
Source: Intel

THE WALL STREET JOURNAL.

Introduction

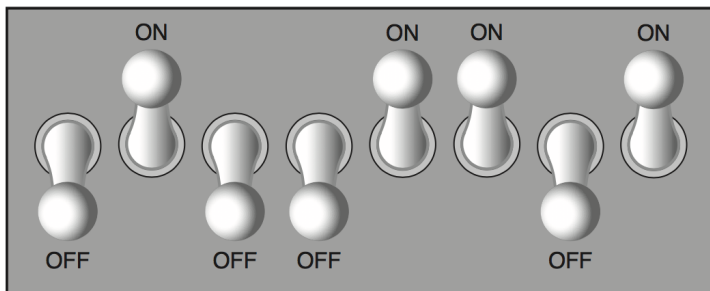
Hardware and Software

**How Computers Store Data**

How a Program Works

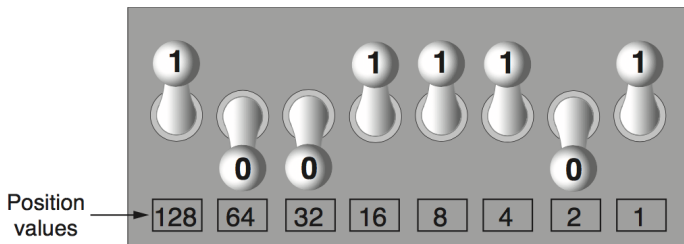
# Storing Numbers

All data stored in a computer is converted to sequences of 0s and 1s.



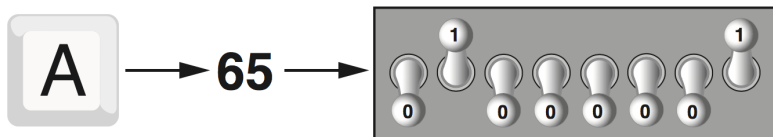
# Storing Numbers

All data stored in a computer is converted to sequences of 0s and 1s.



$$128 + 16 + 8 + 4 + 1 = 157$$

# Storing Characters

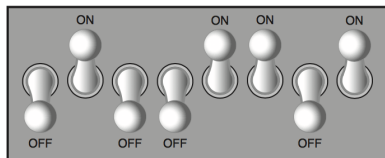


American Standard Code for Information Interchange (ASCII)

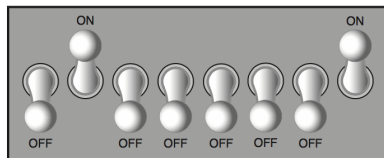
A	B	C	D	E	F	G	H	I	J	K	L	M
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
65	66	67	68	69	70	71	72	73	74	75	76	77
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕	↕
78	79	80	81	82	83	84	85	86	87	88	89	90

Table 1: ASCII encodings of English capital letters.

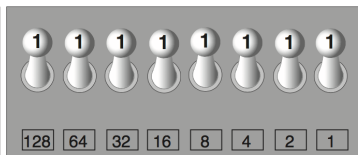
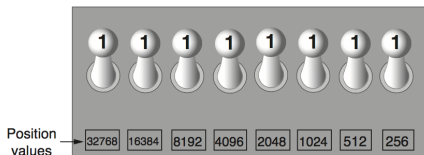
# String Characters



The number 77 stored in a byte.



The letter A stored in a byte.



$$32768 + 16384 + 8192 + 4096 + 2048 + 1024 + 512 + 256 + 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 65535$$

Introduction

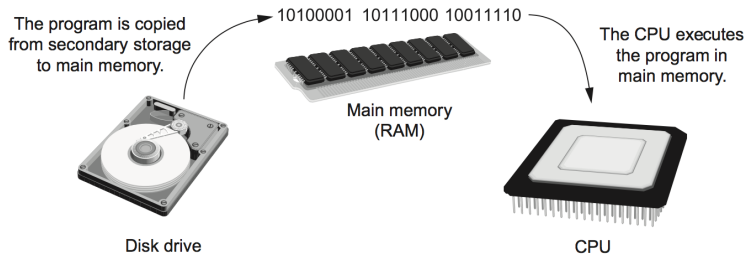
Hardware and Software

How Computers Store Data

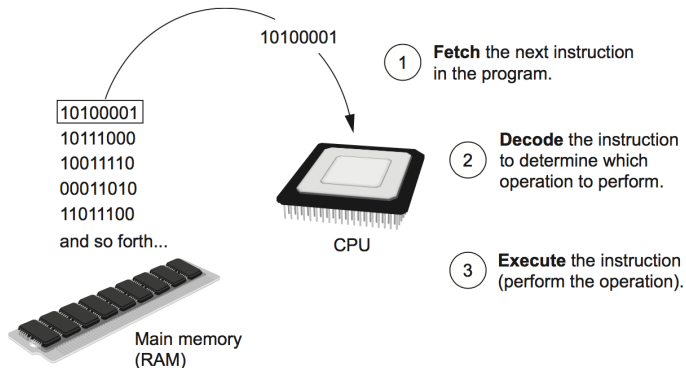
How a Program Works



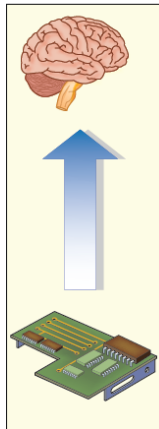
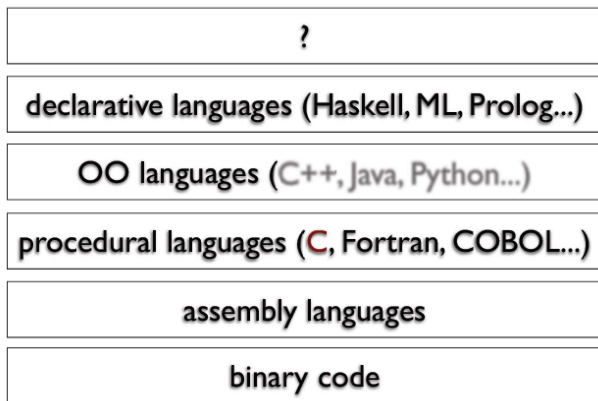
# Fetch-Decode-Execute Cycle



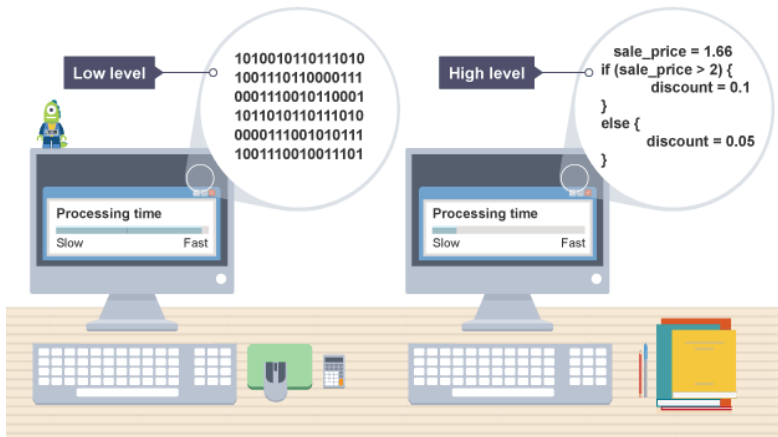
# Fetch-Decode-Execute Cycle



# Programming Languages



# Programming Languages



# Programming Languages

## Machine code

```
89 F8 A9 01 00 00 00 75 06 6B C0 03 FF C0 C3 C1 E0 02 83 E8 03 C3
```

# Programming Languages

## Machine code

```
89 F8 A9 01 00 00 00 75 06 6B C0 03 FF C0 C3 C1 E0 02 83 E8 03 C3
```

## Assembly language

```
.globl f
.text
f:
    mov    %edi, %eax    # Put first parameter into eax register
    test   $1, %eax      # Examine least significant bit
    jnz    odd           # If it's not a zero, jump to odd
    imul   $3, %eax       # It's even, so multiply it by 3
    inc    %eax           # and add 1
    ret                                # and return it
odd:
    shl    $2, %eax       # It's odd, so multiply by 4
    sub    $3, %eax       # and subtract 3
    ret                                # and return it
```



# Programming Languages

## Machine code

```
89 F8 A9 01 00 00 00 75 06 6B C0 03 FF C0 C3 C1 E0 02 83 E8 03 C3
```

## Assembly language

```
.globl f
.text
f:
    mov    %edi, %eax    # Put first parameter into eax register
    test   $1, %eax      # Examine least significant bit
    jnz    odd           # If it's not a zero, jump to odd
    imul   $3, %eax       # It's even, so multiply it by 3
    inc    %eax           # and add 1
    ret                                # and return it
odd:
    shl    $2, %eax       # It's odd, so multiply by 4
    sub    $3, %eax       # and subtract 3
    ret                                # and return it
```

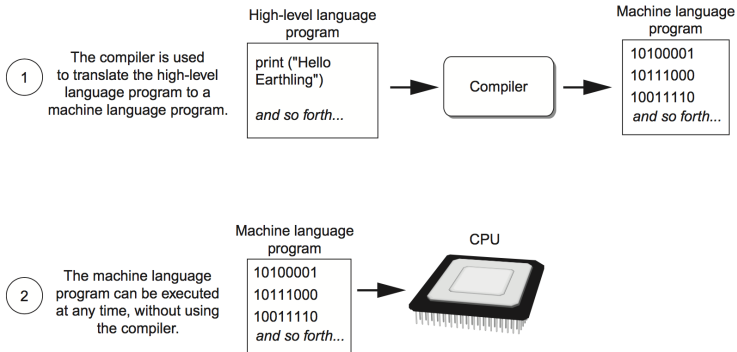


## High-level language (Python)

```
def f(n):
    return 3 * n + 1 if n % 2 == 0 else 4 * n - 3
```

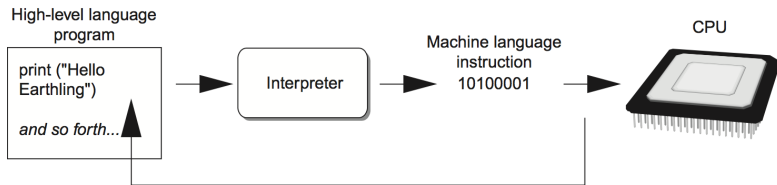


# Compiler vs Interpreter





# Compiler vs Interpreter



The interpreter translates each high-level instruction to its equivalent machine language instructions and immediately executes them.

This process is repeated for each high-level instruction.