1

C How to Program

Introduction to Computers

mhd. Mazen Al Mustafa

Outline

1.1 Introduction

- *C How to Program, Fifth Edition*
  - Author : Deitel & Deitel
  - Publisher : Prentice Hall
  - Object-Oriented programming
  - C 2 Standard Edition

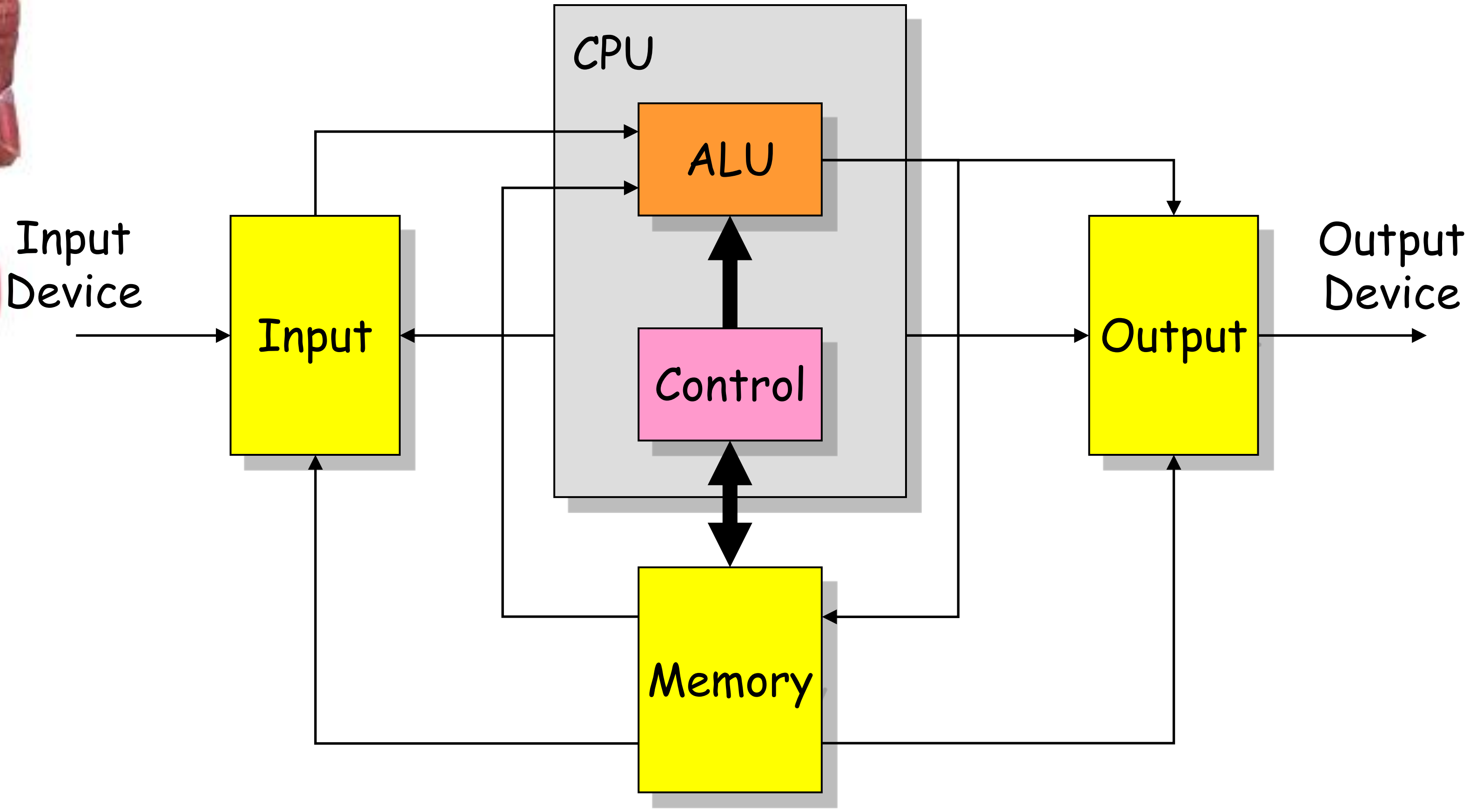
1.2 What Is a Computer?

- Computer
  - Performs computations and makes logical decisions
  - Millions / billions times faster than human beings
- Computer programs
  - Sets of instructions for which computer processes data
- Hardware
  - Physical devices of computer system
- Software
  - Programs that run on computers

1.3 Computer Architecture

- Six logical units of computer system
  - Input unit
    - Mouse, keyboard
  - Output unit
    - Printer, monitor, audio speakers
  - Memory unit
    - RAM
  - Arithmetic and logic unit (ALU)
    - Performs calculations
  - Central processing unit (CPU)
    - Supervises operation of other devices
  - Secondary storage unit
    - Hard drives, floppy drives

Computer Architecture



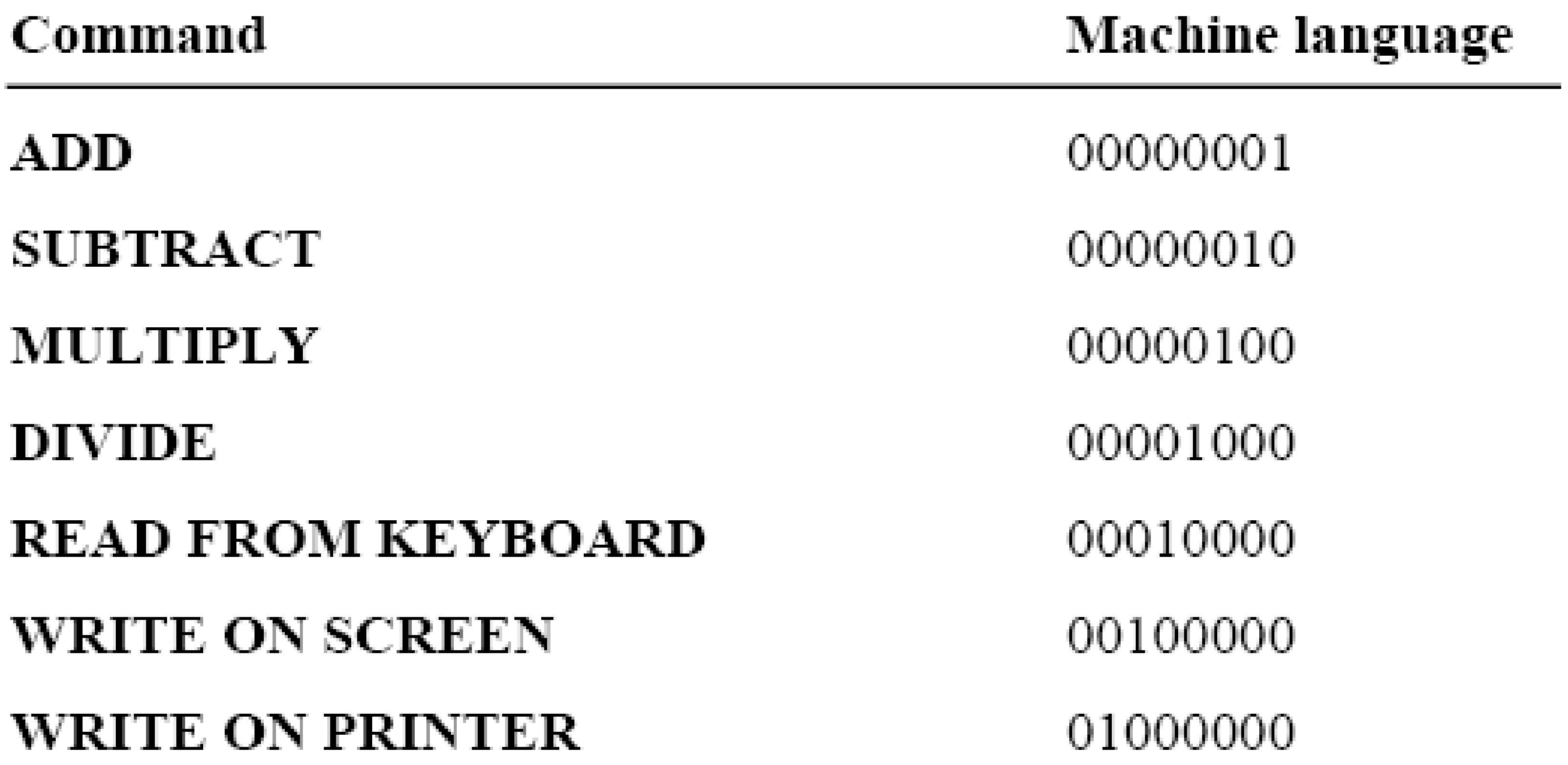
1.4 Early of Operating Systems

- Batch processing
  - One job (task) at a time
  - Operating systems developed
    - Programs to make computers more convenient to use
    - Switch jobs easier
- Multiprogramming
  - “Simultaneous” jobs
  - Timesharing operating systems

1.5 Machine Languages, Assembly Languages and High-Level Languages

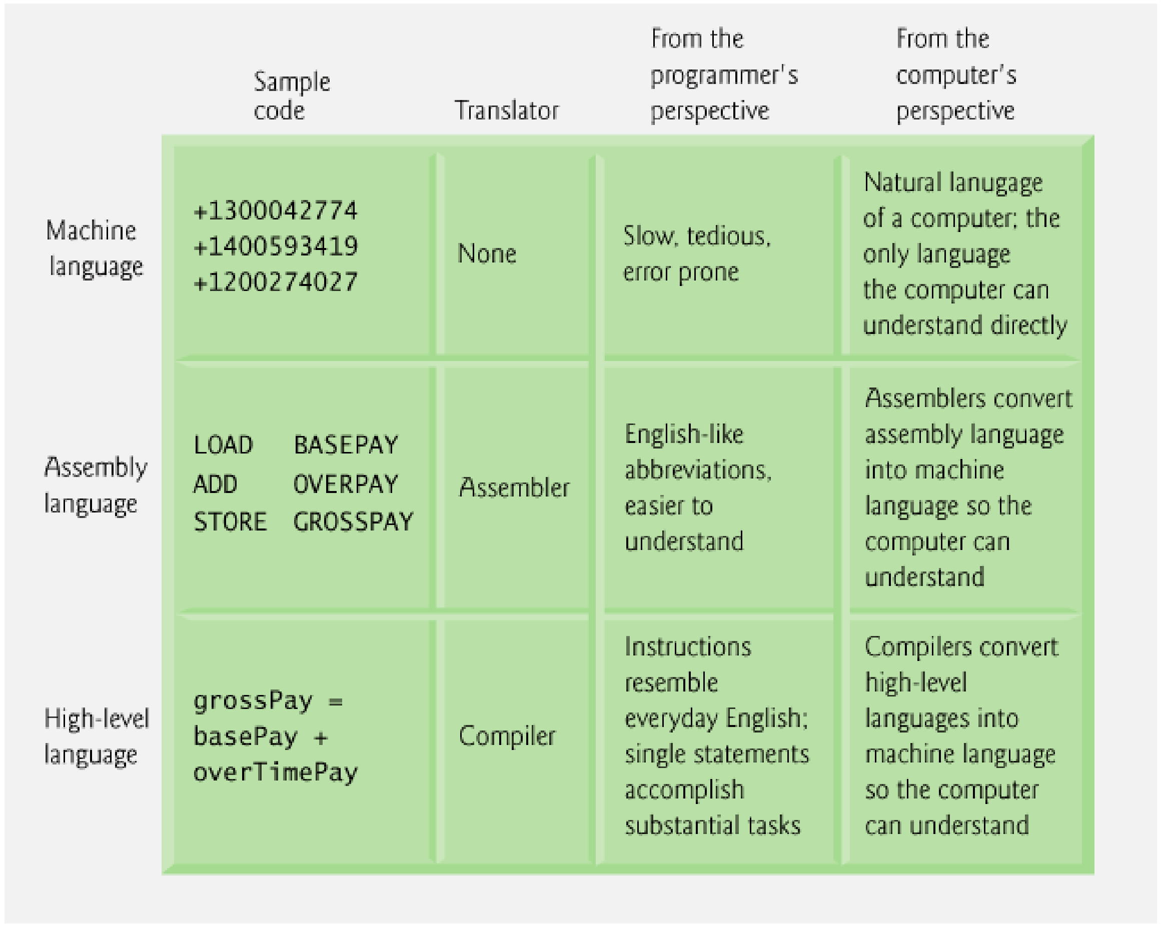
- **Machine language**
  - "Natural language" of computer component
  - Machine dependent
- **Assembly language**
  - English-like abbreviations represent computer operations
  - Translator programs convert to machine language
- **High-level language**
  - Allows for writing more "English-like" instructions
    - Contains commonly used mathematical operations
  - Compiler convert to machine language
- **Interpreter**
  - Execute high-level language programs without compilation
  - It is used in script language (Java script, VB Script ….)

Fig. 1.1 Machine language Vs. Assembly language



Command	Machine language
ADD	00000001
SUBTRACT	00000010
MULTIPLY	00000100
DIVIDE	00001000
READ FROM KEYBOARD	00010000
WRITE ON SCREEN	00100000
WRITE ON PRINTER	01000000

Fig. 1.2 Comparing machine, assembly and high-level languages.



	Sample code	Translator	From the programmer's perspective	From the computer's perspective
Machine language	+1300042774 +1400593419 +1200274027	None	Slow, tedious, error prone	Natural lanugage of a computer; the only language the computer can understand directly
Assembly language	LOAD BASEPAY ADD OVERPAY STORE GROSSPAY	Assembler	English-like abbreviations, easier to understand	Assemblers convert assembly language into machine language so the computer can understand
High-level language	grossPay = basePay + overTimePay	Compiler	Instructions resemble everyday English; single statements accomplish substantial tasks	Compilers convert high-level languages into machine language so the computer can understand

C is a compiled language!!!

- Once a program is written in C, it cannot be executed without further translation on the computer.
  - source program=>machine language
- this is unlike interpreted languages where each statement in the source code is
  - translated individually and
  - executed immediately
- in compiled languages such as C all the statements are translated before being executed

Why study the C language?

- Most widely used programming language.
- Many application areas
  - real time systems
- Lots of employment prospects.
- Other languages have developed from C: C++, Visual C++, Java, C#…….
- writing C code forces you to think about programming fundamentals
- Most common example of procedural programming paradigm.

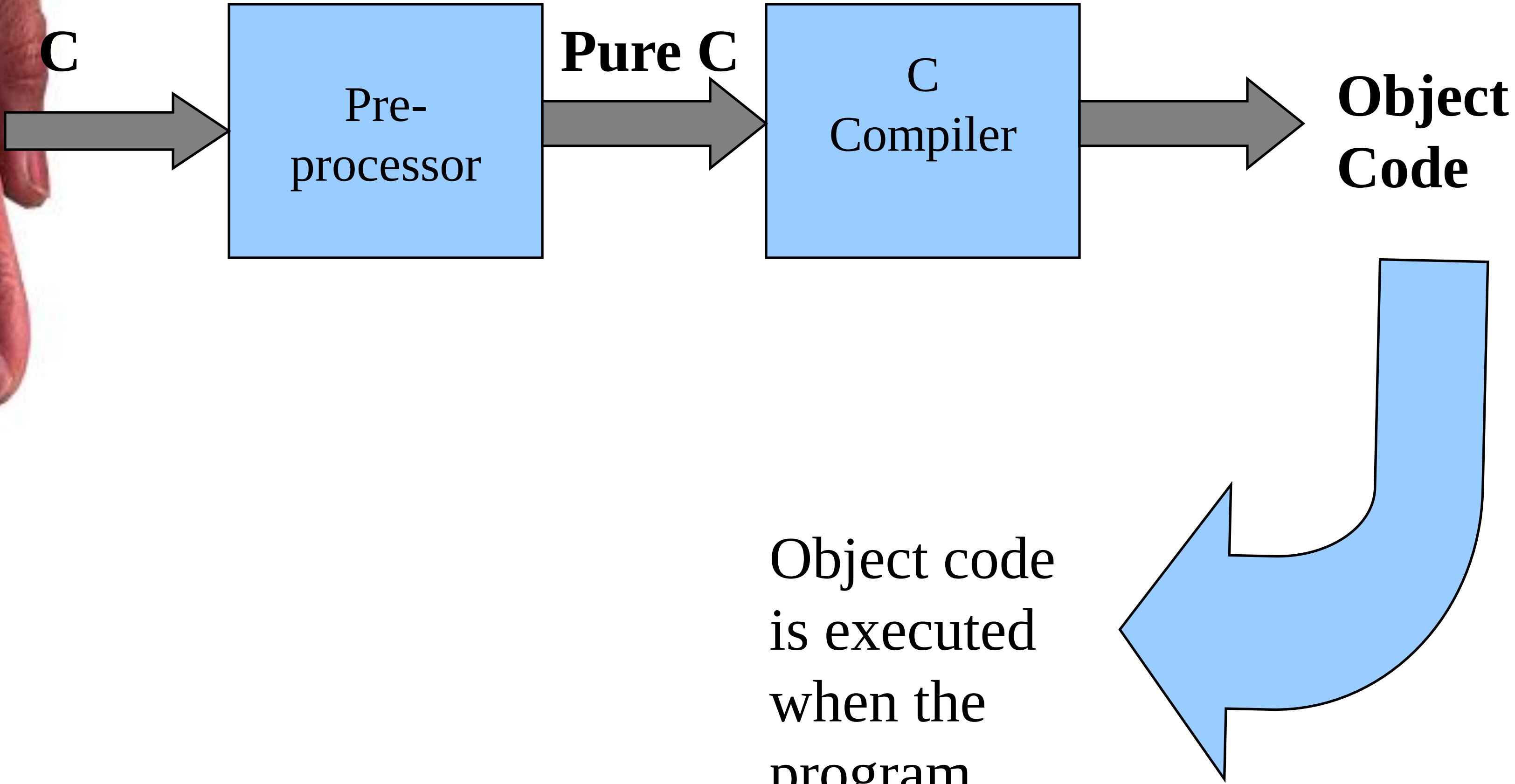
Basics of C Environment

- C systems consist of 3 parts
  - Environment
  - Language
  - C Standard Library

1.6 Typical C Program Development Environment

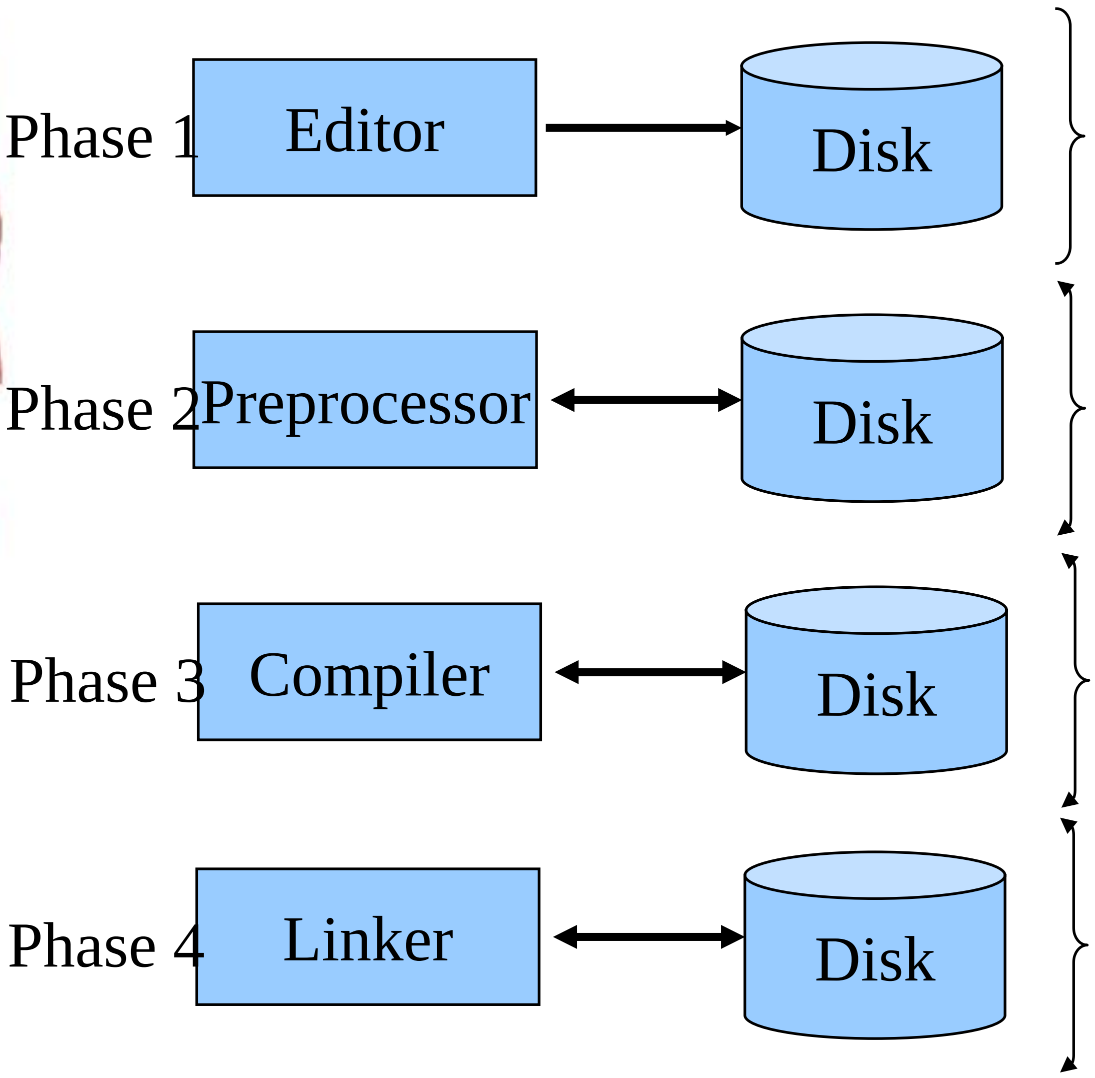
- C programs normally undergo six phases
  - **Edit**
    - Programmer writes program (and stores program on disk)
  - **Preprocessing**
    - Obeys special commands called **preprocessor directive**
  - **Compile**
    - Compiler creates ***Object codes*** from program
  - **Link**
  - Linker links the *Object codes with the code* for the missing functions to produce an **executable image**
  - **Load**
    - Class loader stores byte *Object codes* in memory
  - **Execute**
    - Interpreter translates *Object codes* into **machine language**

The Pre-Processor



Object code is executed when the program runs

Basics of C Environment



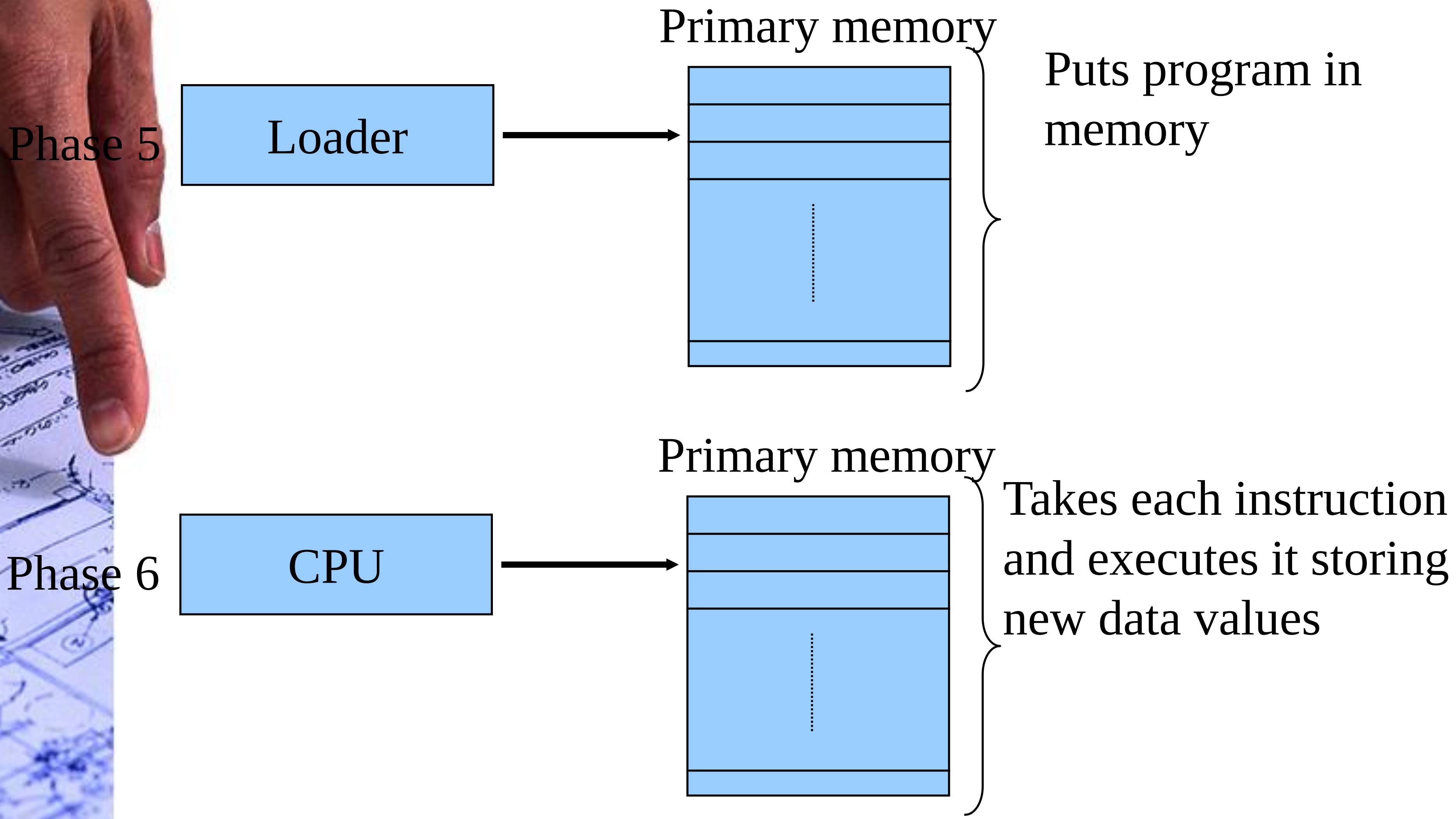
Program edited in Editor and stored on disk

Preprocessor program processes the code

Creates object code and stores on disk

Links object code with libraries and stores on disk

Basics of C Environment



1.7 Programming Errors

- Syntax Errors
  - Detected by the compiler
- Runtime Errors
  - Causes the program to abort
- Logic Errors
  - Produces incorrect result