Introduction to Computer Programming

Nai-Wei Lin
Department of Computer Science and Information Engineering
National Chung Cheng University

Contents

- Introduction to *computers* and *computer science*
- Basic *programming skills* in programming language C
- Basic *problem solving techniques*

2

Chapter One

Introduction to Computers

3

Computers

- *Computers* are *programmable machines* capable of *performing calculations*
- Examples of *special-purpose* computers are *calculators* and *game-playing machines*
- Examples of *general-purpose* computers are *personal computers* and *notebooks*

4

History of Computing

- Abacus – 2000 B. C.
- First mechanical calculator – Wilhelm Schickard, 1623
- Mechanical machine (addition) – Blaise Pascal, 1640
- Mechanical machine (multiplication/division) – Gottfried Leibniz, 1673
- Difference engine and analytical engine (program) – Charles Babbage, 1871; first programmer – Augusta Ada Byron

5

History of Computing

- First vacuum tube electronic computer – John Atanasoff & Clifford Barry, 1939
- Von Neumann Architecture – John von Neumann, 1946
- First transistor electronic computer – IBM 7090, 1958
- First integrated circuit electronic computer – IBM 360, 1964
- First microprocessor electronic computer – Altair 8800, 1975

6

Hardware/Software

- A *computer* consists of *hardware* and *software*
- The *hardware* consists of various *physical devices* that performs *wired* and *basic* operations
- The *software* consists of various *programs* that *coordinates* basic operations to accomplish *flexible* and *complex* tasks

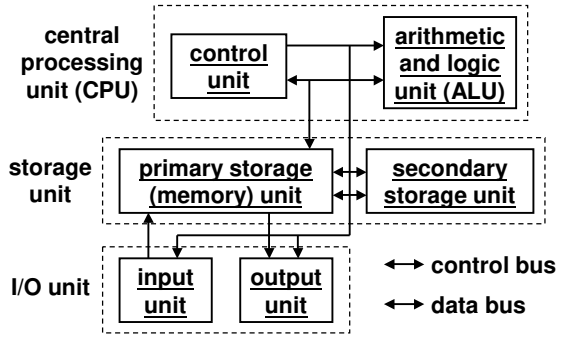
7

Programs

- A *Program* is a sequence of *instructions* (*basic operations*) to control the operation of the computer
- *Programming* is the task of *designing* programs
- *Programming languages* are *notation*s used to represent the instructions of computers

8

Hardware



9

Control Unit

- *Control unit* repeatedly *fetches instructions* from memory unit to control unit via *data bus*
- *Control unit* then *interprets* instructions, and *coordinates* the operations of other units via *control bus*

10

Arithmetic and Logic Unit

- ALU is responsible for *performing calculations* based on *arithmetic operations* such as *addition*, *subtraction*, *multiplication*, and *division*
- ALU is also responsible for *making decisions* based on *logical operations* such as *equality* (=, $\neq$) and *relation* (<, $\leq$, >, $\geq$)

11

Arithmetic and Logic Unit

- Arithmetic or logical *operations* performed by ALU are controlled by control unit via *control bus*
- *Data* on which operations are performed are transferred *from* memory unit via *data bus*
- *Data* resulted from operations are transferred *to* memory unit via *data bus*

12

Primary Storage Unit

- Memory unit *stores* both *instructions* and *data*
- It retains information that is *actively being used* by the computer
- It is the *short-term*, *rapid-access*, *low-capacity* warehouse of the compute

13

Secondary Storage Unit

- Secondary storage unit *retains* information that is *not actively being used* by the computer
- It is the *long-term*, *slow-access*, *high-capacity* warehouse of the computer
- Common secondary storage devices are *disks* and *tapes*

14

Input Unit

- Input unit *transfers* information from various *input devices* to memory unit
- It also transforms information in *human-readable* form to information in *machine-readable* form
- Common input devices are *keyboards* and *mouse devices*

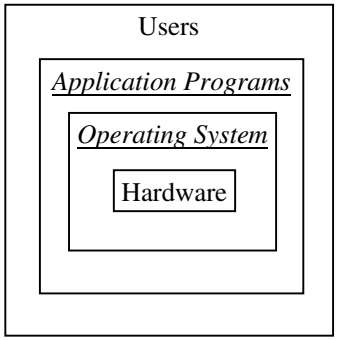
15

Output Unit

- Output unit *transfers* information from memory unit to various *output devices*
- It also transforms information in *machine-readable* form to information in *human-readable* form
- Common output devices are *screens* and *printers*

16

Software



17

Operating System

- The *operating system* provides efficient management of the *hardware* so that *application programmers* can easily use the computer *without knowing the detailed operations of the hardware*
- Common operating systems are *DOS*, *Windows 2000*, *Windows NT*, *Unix*, *Linux*

18

Functions of Operating Systems

- System administration
- Job scheduling
- Memory management
- File management
- Input and output device management

19

Evolution of Operating Systems

- Simple batch systems
- Multiprogrammed batch systems
- Time-sharing systems
- Parallel systems
- Distributed systems

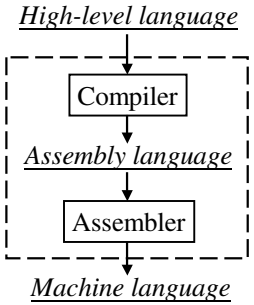
20

Application Programs

- An *application program* allows *users* to use a computer to solve problems in a specific domain *without knowing the details of the computer*
- Common application programs are *MS Word*, *MS Excel*, *MS Access*, *MS Internet Explorer*, *Netscape Communicator*

21

Programming Languages



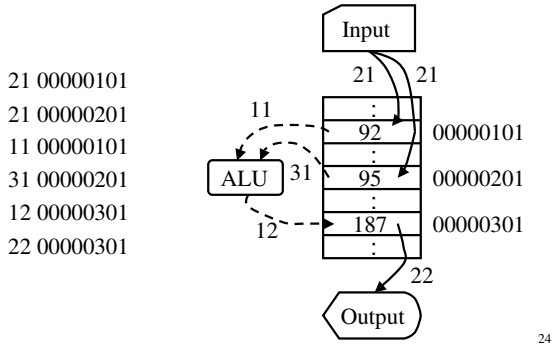
22

Machine Languages

- A computer can *directly understand only* its own *machine language*
- A program denoted by a machine language consists of *strings of numbers* (1's and 0's)
- Machine languages are *machine-dependent*

23

An Example



24

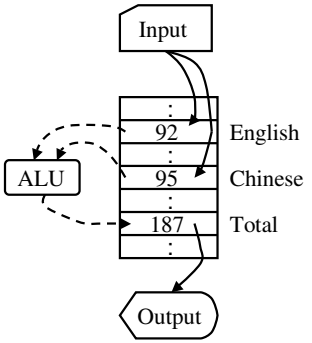
Assembly Languages

- The *assembly language* of a computer is a *mnemonic* representation of its machine language and is also *machine-dependent*
- An *assembler* converts assembly language programs into machine language programs
- Each assembly instruction is usually converted into *one* machine instruction

25

An Example

```
INPUT   English
INPUT   Chinese
LOAD    English
ADD     Chinese
STORE   Total
OUTPUT  Total
```



26

An Example

```
INPUT   English          21 00000101
INPUT   Chinese          21 00000201
LOAD    English    ⇨     11 00000101
ADD     Chinese          31 00000201
STORE   Total            12 00000301
OUTPUT  Total            22 00000301
```

27

High-Level Languages

- A *high-level language* uses English-like notations and commonly used mathematical notations
- A *standardized* high-level language is *machine-independent* or *portable*

28

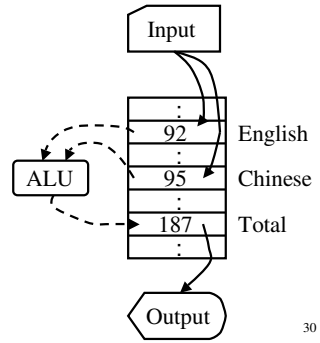
Compilers

- A *compiler* translates high-level language programs into assembly language programs or machine language programs
- Each high-level instruction is usually translated into *several* assembly instructions

29

An Example

```
Input(English);
Input(Chinese);
Total = English + Chinese;
Output(Total);
```



30

An Example

```
Input(English);                 INPUT    English
Input(Chinese);                 INPUT    Chinese
Total = English + Chinese;  ⇨   LOAD     English
                                ADD      Chinese
                                STORE    Total
Output(Total);                  OUTPUT   Total
```

31

Computer Science

- *Computer science* is more concerned with the *software* or the science of *problem solving*
- *Computer engineering* is more concerned with the *hardware* or the engineering of *computing machines*
- *Hardware costs* have been *declining* dramatically; *software costs* have been *rising* steadily

32

Algorithms

- An *algorithm* is an *abstract strategy* for solving a problem
- Solving a problem by computer consists of *designing an algorithm* and *expressing the algorithm as a program*

33

An Example:Finding the GCD

	M	N	R
1	369	27	
	369 =	27 x	13 + 18
2~3	27 =	18 x	1 + 9
	18 =	9 x	2 + 0
4		9	

34

An Example:Finding the GCD

- Store *M* and *N* the value of the larger and smaller of the two input values, respectively
- Divide *M* by *N*, and store the remainder *R*
- If *R* is not 0, then store *M* the value of *N*, store *N* the value of *R*, and return to step 2
- Otherwise, the GCD is the current value of *N*

35

An Example:Finding the GCD

```
Input(M);
Input(N);
Do {
   Q = M / N;   R = M % N;
   M = N;  N = R;
} While (R != 0);
Output(N);
```

36