Introduction to Computing

Objectives

- Understand basic computer technology.
- Become familiar with the software development process.
- Learn about machine language, assembly language, and higher level languages.
- Understand the purpose of assemblers, compilers, and interpreters.
What Is a Computer?

- *Computer* – a machine that processes information under the control of a program.
- *Hardware* – a computer’s electronic and mechanical components.
- *Software* – a computer’s control programs.
- *General purpose computer* – is able to change its control program.
- *Special purpose computer* – has a fixed control program; a calculator or watch.

Computer Hardware (Input/Output)

- **Input devices**
  - Bring data into the computer.
  - Examples: Keyboard, mouse, scanner.
- **Output devices**
  - Produce information in human understandable form.
  - Examples: Printers, monitors, speakers.
Computer Hardware (Memory)

- Primary or main memory
  - Scratch pad: Temporarily stores data and programs during processing.
  - Volatile: Data is lost when power is off.
  - Fast: Completely electronic.
- Secondary storage
  - Nonvolatile storage of data and programs
  - Examples: Disk drives, Compact Disks (CDs).
  - Slow: Requires mechanical motion.

Computer Hardware (Processor)

- Central Processing Unit (CPU)
  - The microprocessor, such as Intel Pentium.
  - Controlled by software.
  - Issues signals to control hardware components.
- Arithmetic-Logic Unit (ALU)
  - Part of the CPU.
  - Performs arithmetic (+), relational (>) and logic (&) operations.
Types of Software

- **Application Software**
  - Programs that perform a particular task or provide a particular service.
  - Examples: Word processor, spreadsheet, computer games, email program.

- **System Software**
  - Programs that perform basic operations.
  - Examples: The *Operating System* lets you run programs, save your work in files, manage folders, etc.

Computer Science and Computer Engineering

- Historically, university computing departments were founded either by electrical engineers or by mathematicians.
- Electrical engineers were interested mainly in developing new hardware and called their departments Computer Engineering.
- Mathematicians were mainly interested in developing new software and called their departments Computer Science.
- Later, business school faculty developed MIS departments to study how computers could be used to effectively manage businesses.
The Software Development Process (Software Lifecycle)

- Analysis (7% of development cost)
  - Domain and problem understanding
  - Requirements definition and specification
- Design (6% of cost)
- Programming (5% of cost)
- Testing and Integration (15% of cost)
- Maintenance (67% of cost)
  - Fixing bugs and security problems
  - Improving efficiency
  - Adding new features

What is Programming?

- Computer program
  - A set of instructions that directs the computer's behavior.
- Computer programming
  - The art and science of designing and writing computer programs.
- Challenges
  - Size and complexity of modern programs is enormous.
  - Windows Vista is ~50 million SLOC.
Computer Hardware

- A computer is essentially an enormous collection of interconnected off/on switches.
- All information used by the computer (data and instructions) must be represented internally as a sequence of off/on states.
- This two-valued encoding is called binary.
- Normally we write 0 and 1 to represent off and on, respectively.
  - Each 0 or 1 in memory is called a bit.
  - A byte is a sequence of 8 bits.

Encoding of Natural Numbers
(In Computer Science, zero is included.)

- What is the obvious encoding using only zeros and ones?

<table>
<thead>
<tr>
<th>Binary (base 2)</th>
<th>0000</th>
<th>0001</th>
<th>0010</th>
<th>0011</th>
<th>0100</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

- What numbers can be represented using 4 bits?
- How many distinct numbers can be represented with 4 bits?
- With 8 bits?
- With $n$ bits?
Addition Circuit

- 3 inputs and 2 outputs.
- The outputs depend on the number of inputs that are on (1). It doesn’t matter which inputs are on.

<table>
<thead>
<tr>
<th>On Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
</tr>
</tbody>
</table>

Addition Circuit

7 + 6

7 + 6 = 13
0111 + 0110 = 1101
Binary Encoding of Other Numbers

- **Negative Integers**
  - Essentially, use one bit to represent the sign.
  - In practice, computers use two's-complement representation. To invert the sign of a number, invert all the bits, then add one:

    \[
    \begin{align*}
    000\ldots00101 &= 5 \\
    111\ldots11010 + 1 &= 111\ldots11011 = -5
    \end{align*}
    \]

  - Why? Because the arithmetic circuits work for both positive and negative numbers without any modification.

- **Floating point numbers, e.g. \(-2.438 \times 10^{13}\)**
  - Use a bit for the sign, some bits for the mantissa and some bits for the exponent.

Binary Encoding of Text

- No obvious way to encode text characters (letters, punctuation marks, digits, and other symbols), so it is decided arbitrarily.

- The most common encoding is ASCII (American Standard Code for Information Interchange):

  \[
  \begin{array}{c|c}
  \text{Character} & \text{Binary} \\
  \hline
  A & 01000001 \quad = 65 \\
  a & 01100001 \quad = 97 \\
  6 & 00110011 \quad = 51 \\
  \end{array}
  \]

- What does a text editor do? What is stored in a text file?
Arithmetic with Text?

- The hardware doesn’t know if a sequence of bits represents a character or a number.
- In the ASCII code, upper case letters are sequential, lower case letters are sequential, and digits are sequential.
- In most programming languages the expression (‘A’ + 1) has the same value as ‘B’.

Machine Language

- Instructions are also represented internally in binary. For example, the addition circuit might be activated when a certain combination of switches are on.
- Highly simplified machine language instruction format:

```
001  XXXX  YYYY  ZZZZ
```

ADD  Number stored at location xxxx  To number stored at location yyyy  And store result in location zzzz
Machine Language Programming

- Sample Program
  001100101110010
  010110110010110
  100011010011101

- Programming in machine language is difficult and error prone!
- Different CPU’s use different languages, so a program written in machine language is not portable between different processors.

Assembly Language

- A shorthand for machine language. There is a different assembly language for every machine language, i.e. assembly language is not portable.
- Uses symbols for instructions and memory addresses.
- Each assembly language instruction corresponds to one machine language instruction.

  ADD A B C
  MUL C C D ; D = (A+B) x (A+B)

- Instructions are written with a text editor.
- An assembler is a program that translates assembly language into machine language.
**High-Level Languages**

- Portable and easier to use than assembly language.
- Include instructions that don’t correspond to a single machine language instruction, e.g. 
  \[ D = (A+B) \times 2 \]  
  *Fortran*
- A compiler is similar to an assembler, but is more complex since there is not a one-to-one correspondence between high-level and machine instructions.
- An interpreter translates one high-level instruction at a time, executes the machine code, then continues with the next instruction.

**Compiled vs. Interpreted Languages**

- Developers of compiled programs distribute the compiled object (machine) code. A different object code file is required for each platform.
- Developers of interpreted programs distribute the source code. Only one version is required for all platforms.
- Running a program in an interpreted language requires the user to have an interpreter installed.
- Interpreted code runs more slowly, since the translation to machine code happens while the program is running.
Java is Both Compiled and Interpreted

- Java source code is compiled to produce byte-codes (machine code for a virtual machine).
- The java virtual machine (JVM) is a byte-code interpreter.
- The byte-codes are portable. The same version of the compiled program will run on any platform with a JVM installed.
- The JVM instruction set is very close to typical real machines, so the byte-code interpretation is very fast.