Algorithms and Programming

Chapter 1: Computer Systems, Problems, Algorithms, and Programs
Overview

High-level subjects

- Computer Systems
  - Computer Hardware and Software
- Problems, algorithms, and programs
The first widely used coding system was the Morse code.

In 1844 Samuel F. B. Morse tapped out his first telegraphic message: “What hath God wrought?”
CPU or processor: executes simple instructions manipulating values in memory
How does hardware relate to software?

• An analogy (Computer ↔ Book)
  Computer          Book
  – Hardware         pages and the ink
  – Software         paragraphs & overall theme

A computer without software is like a book full of blank pages

Software to makes the computer useful
Computer Organization

Computer hardware has five main components

- Input devices
  Allows communication to the computer
- Output devices
  Allows communication to the user
- Processor (CPU)
- Main memory
  Memory locations containing the running program
- Secondary memory
  Permanent record of data often on a disk

Ref: Savitch presentation 1
The Processor

- Typically called the **CPU** (Central Processing Unit) - “Brains” of the computer
  - Arithmetic calculations are performed using the **Arithmetic/Logical Unit (ALU)**
  - **Control unit** decodes and executes instructions (coordinates all computer operations)

- Typical capabilities of CPU include:
  - add
  - subtract
  - multiply
  - divide
  - move data from one location to another

- **Register** is a high speed memory location inside the CPU

Ref: Savitch presentation 1
Computer Memory

Main Memory (ROM, RAM) stores instructions and data while a program is running
- Long list of memory locations
  - Each contains zeros and ones
  - Can change during program execution
- Bit (Binary Digit)
  - A digit that can only be zero or one
- Byte
  - Each memory location has 8 bits
- Address
  - Number that identifies a memory location
- ASCII representation
  (will be explained later)
Larger Data Items

Some data is too large for a single byte
  – Most integers and real numbers are too large

  – Address refers to the first byte

  – Next few consecutive bytes can store the additional bits for larger data

Ref: Savitch presentation 1
Memory Locations and Bytes

byte 1
byte 2
byte 3
byte 4
byte 5
byte 6
byte 7
byte 8
byte 9

3 byte location with address 1
2 byte location with address 4
1 byte location with address 6
3 byte location with address 7

Ref: Savitch presentation 1
# Memory measurement units

<table>
<thead>
<tr>
<th>Unit Name</th>
<th>Abbrev.</th>
<th>Capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Byte</td>
<td>B</td>
<td>8 bit</td>
</tr>
<tr>
<td>Kilobyte</td>
<td>Kb</td>
<td>1024 byte</td>
</tr>
<tr>
<td>Megabyte</td>
<td>Mb</td>
<td>1024 Kilobyte</td>
</tr>
<tr>
<td>Gigabyte</td>
<td>Gb</td>
<td>1024 Megabyte</td>
</tr>
<tr>
<td>Terabyte</td>
<td>Tb</td>
<td>1024 Gigabyte</td>
</tr>
<tr>
<td>Petabyte</td>
<td>Pb</td>
<td>1024 Terabyte</td>
</tr>
<tr>
<td>Exabyte</td>
<td>Eb</td>
<td>1024 Petabyte</td>
</tr>
</tbody>
</table>

http://tr.wikipedia.org/wiki/Bayt
Bit-Byte

Converting a binary number to decimal

110111001

\[1 \times 2^7 = 1 \times 128 = 128\]
\[1 \times 2^6 = 1 \times 64 = 64\]
\[1 \times 2^5 = 1 \times 32 = 32\]
\[0 \times 2^4 = 0 \times 16 = 0\]
\[0 \times 2^3 = 0 \times 8 = 0\]
\[0 \times 2^2 = 0 \times 4 = 0\]
\[1 \times 2^1 = 1 \times 2 = 2\]
\[1 \times 2^0 = 1 \times 1 = 1\]

\[1 + 8 + 16 + 64 + 128 = 217\]
Converting a decimal number to binary

See L2_13__lec01a.pdf page 22 for more number representations. (info. only)
Octal numbers (base 8)

- Digits include
  0, 1, 2, 3, 4, 5, 6, 7

\[ 75_8 = 8^0 \times 5 + 8^1 \times 7 = 5 + 56 = 61_{10} \]

Hexadecimal (base 16)

- Digits include
  0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

- e.g. \( ABC_{16} = 10 \times 16^2 + 11 \times 16^1 + 12 \times 16^0 = 2748_{10} \)

- e.g. \( 4F3_{16} = 4 \times 16^2 + 15 \times 16^1 + 3 \times 16^0 = 1219_{10} \)
Conversion Octal to/from binary

- An octal digit is ALWAYS equal to three (3) binary digits, e.g. $5_8 = 101_2$

- Simply replace the octal digit with the corresponding binary digits and vice-versa

<table>
<thead>
<tr>
<th>Octal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
</tr>
</tbody>
</table>
Example

$1\ 2\ 5\ 0\ 6\ 1\ 7_8 = (\ ?)_2$

Answer = 001 010 101 000 110 001 111$_2$
### Hexadecimal to/from binary

- A hexadecimal digit is equal to four (4) binary digits

<table>
<thead>
<tr>
<th>Hexadecimal</th>
<th>Binary</th>
<th>Hexadecimal</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>8</td>
<td>1000</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>9</td>
<td>1001</td>
</tr>
<tr>
<td>2</td>
<td>0010</td>
<td>A</td>
<td>1010</td>
</tr>
<tr>
<td>3</td>
<td>0011</td>
<td>B</td>
<td>1011</td>
</tr>
<tr>
<td>4</td>
<td>0100</td>
<td>C</td>
<td>1100</td>
</tr>
<tr>
<td>5</td>
<td>0101</td>
<td>D</td>
<td>1101</td>
</tr>
<tr>
<td>6</td>
<td>0110</td>
<td>E</td>
<td>1110</td>
</tr>
<tr>
<td>7</td>
<td>0111</td>
<td>F</td>
<td>1111</td>
</tr>
</tbody>
</table>
ASCII (American Standard Code for Information Interchange)

- the ASCII standard is character set and a character encoding based on the Roman alphabet, as used in modern English
- contains printable characters and control characters
- Printable characters are
  - alphabetical characters: lowercase and uppercase letter
  - numerical characters: digits from 0 to 9
  - all symbol characters: !, @, #, $, %, ^, &, *, (, ), etc.
- control (nonprintable characters) include line feed, carriage return, end of file, escape, delete, etc.
- example: int('A') gives 65, and char(65) gives 'A' – we will understand this example later
Secondary Memory

- Secondary memory
  - Stores instructions and data between sessions
  - A file stores data or instructions in secondary memory
Computer Systems - Software

- A computer program is...
  - A set of instructions for a computer to follow

- Computer software is ...
  - The collection of programs used by a computer
    - Includes:
      » Operating System (Windows, Unix, DOS)
      » Application Software (Word, Excel, Explorer)
      » Programming Languages
Computer Software

• The operating system
  – Allows us to communicate with the computer
  – Is a program
  – Allocates the computer’s resources
  – Responds to user requests to run other programs

• Common operating systems include…
  – UNIX  Linux  DOS
  – Windows  Macintosh  VMS

Ref: Savitch presentation 1
Low-level Languages

- An **assembly language** command such as

  \[ \text{ADD } X \ Y \ Z \]

  might mean add the values found at x and y in memory, and store the result in location z.

- Assembly language must be translated to machine language (zeros and ones)

  0110 1001 1010 1011

- The CPU can follow machine language
High-level Languages

- Common programming languages include ...
  C    C++    Java    Pascal    Visual Basic    FORTRAN
  COBOL    Lisp    Scheme    Ada

- These high – level languages
  - Resemble human languages
  - Are designed to be easy to read and write
  - Use more complicated instructions than the CPU can follow
  - Must be translated to zeros and ones for the CPU to execute a program
Compilers

• Translate high-level language to machine language
  – Source code
    • The original program in a high level language
  – Object code
    • The translated version in machine language

Ref: Savitch presentation 1
Linkers

• Some programs we use are already compiled
  – Their object code is available for us to use
  – For example: Input and output routines

• A Linker combines
  – The object code for the programs we write
  – The object code for the pre-compiled routines
  into
  – The machine language program the CPU can run
Word Processor (editor) Used to type in program and corrections

Source File
Format: text

Compiler
Attempts to translate program into machine code

Successful

Object File
Format: binary

Unsuccessful

Error Messages

Other Object Files
Format: binary

Linker
Resolves cross-references among object files

Executable File (load module)
Format: binary

Loader
Copies executable file into memory; initiates execution of instructions

Input data

Results
Ön işlemci programı .c ve .h uzantılı düz metin (text) halindeki kaynak dosyasını okur, ve çıktı olarak başka bir düz metin dosyası üretir. Bu yeni dosya hiçbir ön işlemci deyimi içermeyen C (veya C++) derleyicisi tarafından işlenmeye hazır bir dosyadır. İkinci adımda bu dosya ile kaynak kodları beraber derlenip sırasıyla nesne kodu ve çalıştırılabilir kod üretilir.

C ve C++ dilleri için, Ön işlemci-Derleme-Bağlama-Çalıştırma işleminin akış diyagramı.
Birden çok dosyanın derlenmesi ve birleştirilmesi.
Flow of Information During Program Execution

Input data: meter readings

Step 1

Program input

Step 2

Central processing unit

Memory

Machine language program for computing water bill

Data entered during execution

Computed results

Program output

Output results: water bill

Step 3
Software Development Method

• Clearly specify the problem
• Analyze the problem
• Design an algorithm to solve the problem
• Implement the algorithm (write the program)
• Test and verify the completed program
• Maintain and update the program
• Can you…

– List the steps in the software development method?

– List the five main components of a computer?

– Describe the work of a compiler?

– Define source code?

– Define object code?

– Describe the purpose of the operating system?
History Note

• First programmable computer
  – Designed by Charles Babbage
  – Began work in 1822
  – Not completed in Babbage’s life time

• First programmer
  – Ada Augusta, Countess of Lovelace
    • Colleague of Babbage

• First compiler
  – Grace Hopper
Key Definitions/Concepts

Problem

• Definition of task to be performed (often by a computer)

See L2_36_ProblemSolving document (info. only)

Algorithm

• A particular sequence of steps that will solve a problem
• Steps must be precise and mechanical
• The notion of an algorithm is a fundamental intellectual concept associated with computing

Program

• An algorithm expressed in a specific computer programming language (C, C++, Java, Perl, …)
# Programming vs. Cooking

<table>
<thead>
<tr>
<th>Programming</th>
<th>Cooking</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem</td>
<td>Make fudge brownies</td>
</tr>
<tr>
<td>Algorithm</td>
<td>Recipe</td>
</tr>
<tr>
<td>Program</td>
<td>Recipe written in a specific language (English, Russian, Chinese, Latvian, etc.)</td>
</tr>
</tbody>
</table>
Writing Pseudo code

- Pseudo code is a way to write an algorithm (recipe)
  ✓ Pseudo code is easier to read and write and allows the programmer to concentrate on the logic of the problem
  ✓ Pseudo code is really structured English
    • statements are written in simple English
    • each instruction is written on a separate line
    • each set of instructions is written from top to bottom, with only one entry and one exit
    • groups of statements may be formed into modules, and that group given a name
A Sample Problem

Is a given number even or odd?
Analysis

What numbers are allowed? Where does the number come from? What do “even” and “odd” mean? How is the answer to be reported? Is there any format requirements?
More Precise Problem Restatement

Given an integer number typed in from the keyboard,
   If it is even, write “even” on the screen
   If it is odd, write “odd” on the screen
An Algorithm

1. Start
2. Read in the number
3. remainder = number mod 2
4. If the remainder is 0, write “even”
   Otherwise, write “odd”
5. Stop

Let’s take two samples:
46
79
An Algorithm

1. Start
2. Read in the number
3. remainder = number mod 2
4. If the remainder is 0, write “even”
   Otherwise, write “odd”
5. Stop

Test: 234784832792543

An alternate algorithm:

If the rightmost digit is 0, 2, 4, 6, or 8, write “even”
Otherwise, write “odd”
There are six basic computer operations
1. a computer can receive information
   Ex: Get(or Read) class number from the keyboard
2. a computer can put out information
   Ex: Print(or Write) ‘Program Completed’
3. a computer can perform arithmetic operations
   - use symbols from the keyboard
4. a computer can assign a value to a variable or memory location
5. a computer can compare two variables and select one of two alternate actions
6. a computer can repeat a group of actions
Writing Pseudocode - A computer processes operations in a priority

<table>
<thead>
<tr>
<th>Priority</th>
<th>Operation</th>
<th>In a computer lang. (C)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Parenthesis</td>
<td>((...)) Inner most parenthesis processed first)</td>
</tr>
<tr>
<td>2</td>
<td>The unary operators</td>
<td>+, -, ++, --, and !.</td>
</tr>
<tr>
<td>3</td>
<td>The binary arithmetic operations</td>
<td>*, /, %</td>
</tr>
<tr>
<td>4</td>
<td>The binary arithmetic operations</td>
<td>+, -</td>
</tr>
<tr>
<td>5</td>
<td>The boolean operations (equal, not equal)</td>
<td>==, !=</td>
</tr>
<tr>
<td>6</td>
<td>The boolean operations (and)</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>7</td>
<td>The boolean operations (or)</td>
<td></td>
</tr>
</tbody>
</table>
The value of 6-4/2 is 4 not 1. (Division is done first according to the priority rules. You might test in on a calculator or Excel software for now.)

If you want to do subtraction first, you need to use the parentheses:

- (6-4)/2 is 1.
- (6-4*(5-2)) gives -6.
Example:
Write an algorithm reading two integers from the keyboard and display the sum of them on the screen.

**input:** two integers from the keyboard \((A, B)\)

**output:** sum of two integers \((\text{Total})\)

1. Start
2. Read number \(A\) from the keyboard
3. Read number \(B\) from the keyboard
4. \(\text{Total}=A+B\)
5. Display \(\text{Total}\) on the screen
6. Finish
Writing Pseudocode (cont.)

In class example:
Write an algorithm
- reading the price of a merchandise from the keyboard
- calculating the KDV (18%)
- and printing KDV on the screen
In class example solution:

**input:** price  
**output:** kdv

1. Start  
2. Read price from the keyboard  
3. kdv=price*0.18  
4. Write kdv on the screen  
5. Stop
When designing an algorithm, a programmer must introduce some unique names which represents variables or objects in the problem. Names should be meaningful.
Flowchart is a type of **diagram** that represents an **algorithm** or **process**, showing the steps as boxes of various kinds, and their order by connecting these with arrows. This diagrammatic **representation** can give a step-by-step solution to a given **problem**. Process operations are represented in these boxes, and arrows connecting them represent flow of control.

Flowcharts are used in analyzing, designing, documentating or managing a process or program in various fields.
Flowcharts building blocks

Lamp doesn't work

Lamp plugged in?

No

Plug in lamp

Yes

Bulb burned out?

Yes

Replace bulb

No

Repair lamp
Flowcharts building blocks

**Example:**
A flowchart for computing the factorial of N (10!) where N! = (1*2*3*4*5*6*7*8*9*10).

**Symbols**
A typical flowchart from older basic computer science textbooks may have the following kinds of symbols:

- **Start and end symbols**
  Represented as circles, ovals or rounded rectangles, usually containing the word "Start" or "End", or another phrase signaling the start or end of a process, such as "submit inquiry" or "receive product".

- **Arrows**
  Showing "flow of control". An arrow coming from one symbol and ending at another symbol represents that control passes to the symbol the arrow points to.

http://en.wikipedia.org/wiki/Flowcharts
Flowcharts building blocks (cont.)

**Input/Output**
Represented as a parallelogram. Examples: Get X from the user; display X.

**Generic processing steps**
Represented as rectangles. Examples: "Add 1 to X"; "replace identified part"; "save changes" or similar.

**Conditional or decision**
Represented as a diamond (rhombus) showing where a decision is necessary, commonly a Yes/No question or True/False test. The conditional symbol is peculiar in that it has two arrows coming out of it, usually from the bottom point and right point, one corresponding to Yes or True, and one corresponding to No or False. (The arrows should always be labeled.) More than two arrows can be used, but this is normally a clear indicator that a complex decision is being taken, in which case it may need to be broken-down further or replaced with the "pre-defined process" symbol.
Flowcharts building blocks (cont.)

Output
A rectangle with a curved bottom represents a document or report.

Labeled connectors
This shape means the flow continues on another page. A letter or page number in the shape tells you where to go.

You can draw these shapes in a MS Word document by selecting these from Ekle-Şekiller-Akış çizelgesi.
Next, a C Program

Now that have an algorithm, we would like to write a C program to carry it out.

But first, what is a program? In fact, what is a computer?
What is a Program?

The CPU executes instructions one after the other.

Such a sequence of instructions is called a “program”

Without a program, the computer is just useless hardware

Complex programs may contain millions of instructions
A "Hello world" program is a computer program that outputs "Hello, world" on a display device.

```c
#include <stdio.h>
int main()
{
    printf("Hello world.");
}
```
Memory

Memory is a collection of locations called **variables**

Each variable has

- A **name** (an **identifier**)
- A **type** (the kind of information it can contain)

Basic types include

- **int** (integers – whole numbers: 17, -42)
- **double** (floating-point numbers with optional fraction and/or exponent: 3.14159, 6.02e23)
- **char** (character data: ‘a’, ‘?’, ‘N’, ‘ ’, ‘9’)


C Program Skeleton

/* <program description> */
/* <pseudocode section> */

#include <stdio.h>  /* standard input output library */

int main ()
{
  /* <variable declaration section> */
  /* <program statement section> */
  /* exit with successful execution status */
  return (0);
}

/* exit with successful execution status */
A "Hello world" program is a computer program that outputs "Hello, world" on a display device.

```c
#include <stdio.h>
int main()
{
  printf("Hello world.");
}
```
The Program in C (part I)

/* read a number and report whether it is even or odd */
#include <stdio.h>

int main () {
    int num; /* input number */
    int rem; /* remainder after division by 2 */

    /* get number from user */
    printf("Please enter a number: ");
    scanf("%d", &num);
/* calculate remainder and report even or odd */
rem = num % 2;
if (rem == 0) {
    printf("even\n");
} else {
    printf("odd\n");
}
/* terminate program */
return 0;

Remember: Don’t sweat the details!!! (for now)
Note: Code is saved as L2-61.c
Sample Execution

```
"C:\evenodd\Debug\evenodd.exe"

Please enter a number: 17
odd
Press any key to continue
```