COP 3223
C programming Language

Dr. Nihan Cicekli

University of Central Florida

Department of Computer Science

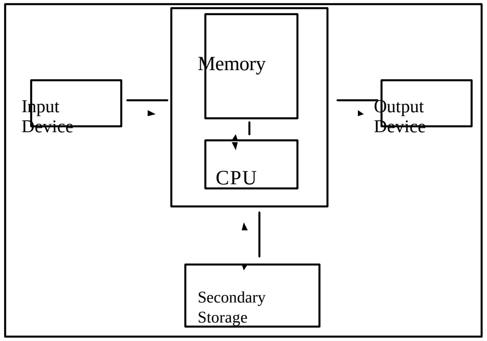
http://www.cs.ucf.edu/courses/cop3223/nihan

Spring 2003

What Is a Computer?

- Computer
  - Device capable of performing computations and making logical decisions
  - Computers process data under the control of sets of instructions called computer programs
- Hardware
  - Various devices comprising a computer
  - Keyboard, screen, mouse, disks, memory, CD-ROM, printer, and processing units
- Software
  - Programs that run on a computer
  - Microsoft Windows 2000, Microsoft Office 2000, Netscape Navigator, Internet Explorer

Conceptual Structure of a Computer System



Input Device

Memory

CPU

Output Device

Secondary Storage

Memory

- Store information (data + instructions)
- A sequence of memory cells.
  - a byte is 8 bits
  - a bit is the smallest unit (0 or 1)
- Store, retrieve, update
  - changing the pattern of 0 and 1s in memory cells
  - copying these patterns into some internal registers
- Stored information in memory is volatile.

CPU (Central Processing Unit)

- Process and manipulate information stored in memory.
- It can be divided into two units: CU (Control Unit) and ALU (Arithmetic Logic Unit)
- CU coordinates activities of the computer and controls other devices of computer.
- ALU processes arithmetical and logical instructions.

Input and Output Devices

- Provide the interface between the user and the computer.
- Input devices are used to enter instructions or data by the user.
- Output devices are used to give results of computations.
- Input Devices: keyboard, mouse
- Output Devices: monitor, printer

Secondary Storage

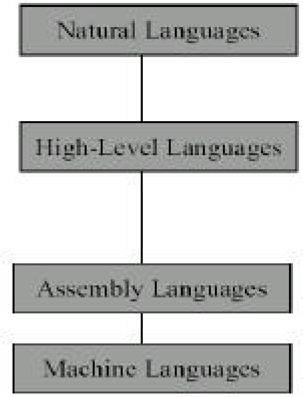
- Computers have limited main memory and information stored in main memory is volatile. i.e. when a computer is switched off, information in its main memory disappears.
- There are additional data storage units, called *secondary storage devices*.
- Data stored in these secondary storage devices are permanent, i.e. data does not disappear when you switch off the computer.
- Some secondary storage units:
  - Floppy Disks, Hard Disks, Tape Drive, Optic Disk (CD Drive)

Our focus is software.

- Software allows the user to communicate with the hardware.
- *Programming* is the process by which computer software is created.
  - – Programmers are humans that create software.
- Don’t need to know a great deal about computer hardware in order to write software.
  - – Analogous to driving a car.

Computer Languages

- Software is written using a computer language (or programming language).
- Computers understand only sequences of numbers.
  - – In particular, sequences of 0’s and 1’s.
- Special languages allow people to communicate with computers since they are not capable of understanding human languages.
  - – Examples include C, Pascal, and Java.



Computer Languages (Cont.)

Three types of programming languages

1. Machine languages
   - Strings of numbers giving machine specific instructions
   - Example:

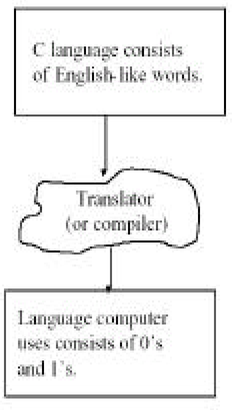
```
00010011010000011110100
```

2. Assembly languages
   - English-like abbreviations representing elementary computer operations (translated via assemblers)
   - Example:

```
LOAD   BASEPAY
ADD    OVERPAY
STORE  GROSSPAY
```

Computer Languages (Cont.)

3. High-level languages
   - Codes similar to everyday English
   - Use mathematical notations (translated via compilers)
   - Example:
     `grossPay = basePay + overTimePay;`



C language consists of English-like words.

Translator (or compiler)

Language computer uses consists of 0's and 1's.

We Can Write Programs to

- Search a telephone directory
- Play chess
- Send a rocket to outer space
- and so on ...

History of C

- C
  – Developed by Denis M. Ritchie at AT&T Bell Labs in 1972 as a systems programming language
  – Used to develop UNIX
  – Used to write modern operating systems
  – Hardware independent (portable)
- Standardization
  – Many slight variations of C existed, and were incompatible
  – Committee formed to create a "unambiguous, machine-independent" definition
  – Standard created in 1989, updated in 1999

The C Standard Library

- C programs consist of pieces/modules called functions
  - A programmer can create his own functions
    - Advantage: the programmer knows exactly how it works
    - Disadvantage: time consuming
  - Programmers will often use the C library functions
    - Use these as building blocks
  - Avoid re-inventing the wheel
    - If a pre-made function exists, generally best to use it rather than write your own
    - Library functions carefully written, efficient, and portable

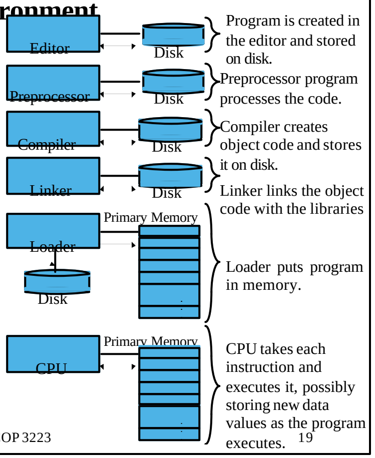
Other High-level Languages

- C++
  - Superset of C, and provides object-oriented capabilities
- Java
  - Create web pages with dynamic and interactive content
- Fortran
  - Used for scientific and engineering applications
- Cobol
  - Used to manipulate large amounts of data
- Pascal
  - Intended for academic use

Basics of a Typical C Program Development Environment

- Phases of C Programs:

1. *Edit*
2. *Preprocess*
3. *Compile*
4. *Link*
5. *Load*
6. *Execute*



Program is created in the editor and stored on disk.

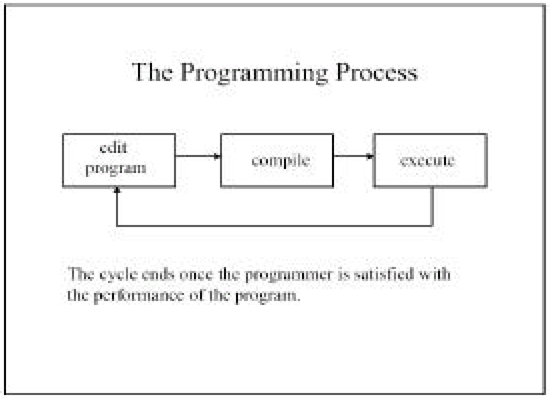
Preprocessor program processes the code.

Compiler creates object code and stores it on disk.

Linker links the object code with the libraries

Loader puts program in memory.

CPU takes each instruction and executes it, possibly storing new data values as the program executes.



The Programming Process

edit program
compile
execute

The cycle ends once the programmer is satisfied with the performance of the program.

Let’s Learn C

- C programming language
  - – Structured and disciplined approach to program design
- You cannot learn the C language by reading it.
  - – You must experiment with the programs discussed in the lecture and textbook. In other words, type the programs into the computer and see what happens.

A Simple C Program

```
/* The traditional first program in honor of Dennis
   Ritchie who invented C at Bell Labs in 1972. */

#include <stdio.h>
int main(void)
{
    printf(“Hello, world!\n”);
    return 0;
}
```

```
Hello, world!
```

A Simple C Program: Printing a Line of Text

- Comments
  - – Text surrounded by **`/*`** and **`*/`** is ignored by computer
  - – Used to describe program
- **`#include <stdio.h>`**
  - – Preprocessor directive
    - Tells computer to load contents of a certain file
  - – **`<stdio.h>`** allows standard input/output operations

A Simple C Program: Printing a Line of Text

- **`int main(void)`**
  - – C programs contain one or more functions, exactly one of which must be **`main`**
  - – Parenthesis used to indicate a function
  - – **`int`** means that **`main`** "returns" an integer value
  - – **`void`** indicates that the function takes no arguments
- Braces (**`{`** and **`}`**) indicate a block
  - – The bodies of all functions must be contained in braces

A Simple C Program: Printing a Line of Text

- **`printf("Hello, world!\n" );`**
  - Instructs computer to perform an action
    - Specifically, prints the string of characters within quotes (`" "`)
  - Entire line called a statement
    - All statements must end with a semicolon (**`;`**)
  - Escape character (**`\`**)
    - Indicates that **`printf`** should do something out of the ordinary
    - **`\n`** is the newline character

A Simple C Program: Printing a Line of Text

- **`return 0;`**
  - A way to exit a function
  - **`return 0`**, in this case, means that the program terminated normally
- Right brace **`}`**
  - Indicates end of **`main`** has been reached

Another Simple C Program

```
/* Printing on one line with two printf
   statements */

#include <stdio.h>

int main (void )
{
    printf(“Welcome ”);
    printf(“to C!\n”);
    return 0;
}
```

```
Welcome to C!
```

Another Simple C Program

```
/* Printing multiple lines with a single printf */

#include <stdio.h>

int main (void)
{
  printf(“Welcome\nto\nC!\n”);
  return 0;
}
```

```
Welcome
to
C!
```

Some common escape sequences

- \n Newline.
- \t Horizontal tab.
- \r Carriage return.
- \\ Backslash.
- \” Double quote.

General Form of a Simple C Program

Preprocessing directives
`int main(void)`
`{`
  declarations
  statements
`}`

The use of #define and #include

`#include <`*filename*`>`
e.g. stdio.h, math.h, string.h, stdlib.h
(usually found in */usr/include/*)

```
#define PI 3.14159
#define MAX 100
```

e.g. `printf(“Pi = %f ”, PI)` is equivalent to
`printf(“Pi = %f ”, 3.14159)`

Addition Program

```
/* This programs adds the two integers that it reads */
#include <stdio.h>

int main (void)
{
   int num1, num2, sum;              /* declarations */

   printf(“Enter first integer.\n”); /* prompt */
   scanf(“%d”, &num1);               /* read an integer*/

   printf(“Enter second integer.\n”);/* prompt */
   scanf(“%d”, &num2);               /* read an integer */

   sum = num1 + num2;                /* assignment of sum */

   printf(“Sum is %d.\n”, sum);      /* print sum */
   return 0;                         /* program ended
}                                       successfully */
```

Sample Runs

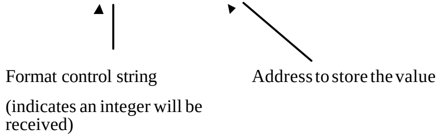
```
Enter first integer.
45
Enter second integer.
15
Sum is 60.
```

```
Enter first integer.
30
Enter second integer.
12
Sum is 42.
```

Dissection of the Addition Program

- **`num1`**, **`num2`**, and **`sum`** are variables.

  The declaration specifies that these variables hold integer values.
- **`scanf("%d", &num1);`**



Format control string

(indicates an integer will be received)

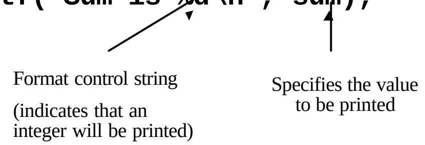
Address to store the value

Dissection of the Program (cont.)

- **`sum = num1 + num2;`**

– calculates the sum of variables **num1** and **num2**, and assigns the result to variable **sum** using the assignment operator =

- **`printf("Sum is %d\n", sum);`**



Format control string

(indicates that an integer will be printed)

Specifies the value to be printed

Memory Concepts

- Variables
  - Variable names correspond to locations in the computer's memory
  - Every variable has a name, a type, a size and a value
  - Whenever a new value is placed into a variable (through **`scanf`**, for example), it replaces (and destroys) the previous value
  - Reading variables from memory does not change them
- A visual representation

Write a single C statement to accomplish each of the following.

- Declare variables **c, thisVariable, q76354,** and **number** to be of type int.

```
int c, thisVariable, q76354, number;
```

- Prompt the user to enter an integer. End your message with a colon followed by a space and leave the cursor positioned after the space.

```
printf(“Please enter an integer. ”);
```

- Read an integer from the keyboard and store the value entered in integer variable a.

```
scanf(“%d”,&a);
```

Write a single C statement to accomplish each of the following.

- Print the message **“This is a C program.”** on one line.

```
printf(“This is a C program.\n”);
```

- Print the message **“This is a C program.”** with each word on a separate line.

```
printf(“This\nis\na\nC\nprogram.\n”);
```

Identify and correct the errors in each of the following statements.

- **`scanf(“d”, value);`**

```
scanf(“%d”, &value);
```

- **`num1 + num2 = sum;`**

```
sum = num1 + num2;
```

- **`printf(“%d + %d is \n”, x,y,x+y)`**

```
printf(“%d + %d is %d \n”, x,y,x+y);
```

Variables of other types

```
#include <stdio.h>

int main()
{
  char c;
  float x, y;

  c = ‘A’;
  printf(“%c\n”, c);
  x = 1.0;
  y = 2.0;
  printf(“The sum of x and y is %f.\n”, x+y);
  return 0;
}
```

Initialization

- When variables are declared they may also be initialized.

```
char c ='A';
int i =1;
float z = 1.75;
int length =10, width = 5;
```

The Use of `printf()`

- `printf("%d %3d%7d ***\n", 1, 2, 3);`

will print

```
1•••2••••••3•***
_
```

- `printf("Get set: %s %d %f %c%c***", "one",2,3.33,'G', 'N');`

will print

```
Get set: one 2 3.330000 GN***_
```

- `printf("%.1f %.2f%7.3f",4.52,1.0, 6.0);`

will print

```
4.5•1.00••6.000_
```

The use of `scanf()`

- `scanf("%c%c%c%d", &first,&mid,&last,&age);`

Input:

```
ABC19
```

- `scanf("%f",&average);`

Input:

```
65.9
```

- `printf` returns the number of characters printed
- `scanf` returns the number of successful conversions

Problem Solving

- Write a C program to read three integers and print their average.