IC150 Lecture 2

Timothy A. Gonsalves

Review

Computers:

- almost everywhere these days
- banks, shops, railway reservations, internet/web
- engineering applications

  VLSI chip design, machine design (CAD/CAM), structural analysis, process control etc etc
- doing without computers - unimaginable

Computer Software:

- collection of instructions to the computer

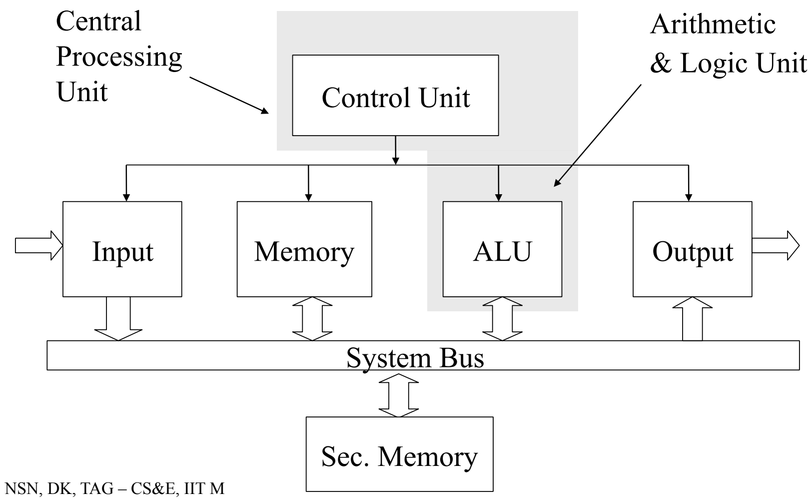
Software

Very critical component in a computer application

Considerable complexity

- large collection of programs
- subdivided into modules with specific purposes
- developed by a team of individuals
- involves - system design, choice of algorithms, choice of data structures, language of implementation, testing, maintenance

Building Blocks (Computer Architecture)



The Blocks, Their Functions

To be useful, a programme must take inputs from the outside world and give back its output

- **Input unit**

  Takes inputs from the external world via variety of input devices:
  keyboard, mouse, touchscreen
  temperature sensors, odometers, wireless devices etc.

- **Output Unit**

  Sends information (after retrieving, processing) to output devices:
  monitor/display, speaker
  projectors, switches, relays, *gearbox* etc.

More (try *more filename* on your Unix/Linux machine)

- **Memory**

  Place where information is stored.

Primary memory

Electronic devices, used for temporary storage. Characterized by speedy response (ns).

Secondary Memory – Devices for long-term storage.
Contain mechanical components, magnetic storage media – floppies, hard disks.
Compact Disks use optical technology.
Used to store user data (programs, inputs, results etc.), also used extensively during computation.
Low-cost, high capacity but slow (ms).

Some More (Commands are in */bin, /usr/bin*. Use *ls*)

- **System Bus**

  Essentially a set of wires, used by the
  other units to communicate with each other.
  transfers data at a very high rate

- **ALU** – Arithmetic and Logic Unit

  Processes data - add, subtract, multiply, …
  Decides – eg. after comparing two values

Finally (check *man cp, man mv, man ls, man –k* search string)

- **Control Unit**

  Controls the operation of the other units.
  Controls the interaction between the other units.

Control Unit + ALU is called the CPU

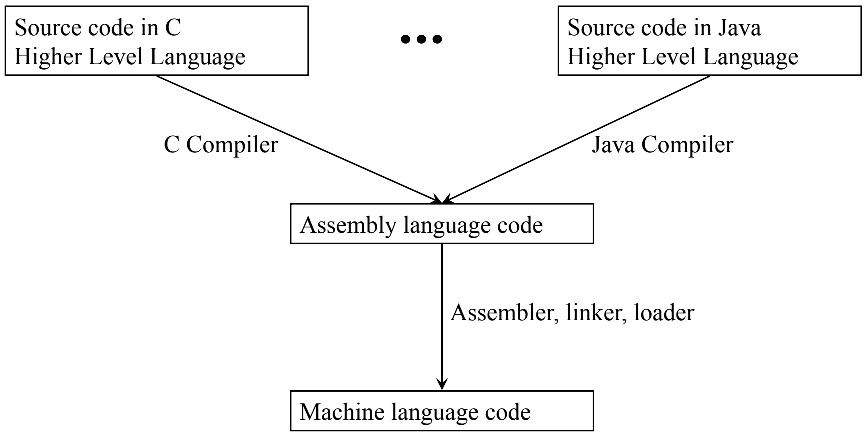
The CPU (editors *emacs, vi, gedit* used to create text)

- Can *fetch* an instruction from memory
- *Execute* the instruction
- *Store* the result in memory
- An instruction has the following structure:
  Operation, operands
- Source operand and destination operand may be the same
- A simple operation

  add a, b *Adds the contents of memory locations a and b and stores the result in location a*

Compilers

Human friendly languages → source code



Machine understandable language

Assembly language

- An x86/IA-32 processor can execute the following binary instruction as expressed in machine language:

  Binary: 10110000 01100001

  Asm: mov al, 061h

  HLL: al = 97;
  - Move the hexadecimal value 61 (97 decimal) into the processor register named "al".
  - assembly language representation is easier to remember (*mnemonic*)

From Wikipedia

Higher Level Languages

- Higher level statement = many assembly instructions
- For example "X = Y + Z" could require the following sequence
  - Fetch into R1 contents of Y
  - Fetch into R2 contents of Z
  - Add contents of R1 and R2 and store it in R1
  - Move contents of R1 into location named X
- HLLs can be at many levels

Programs = solutions

- A program is a sequence of instructions
  - *This is from the perspective of the machine or the compiler!*
- A program is a (frozen) solution
  - *From the perspective of a human a program is a representation of a solution* devised by the human. *Once frozen (or written and compiled) it can be executed by the computer – much faster, and as many times as you want.*

Programming = Problem Solving

- Software development involves the following
  - A study of the problem (requirements analysis)
  - A description of the desired solution (specification)
  - Devising an actual solution (design)
  - Writing the program (coding)
  - Testing
- The critical part is the solution design:
  - Must work out the steps to solve the problem
  - Analyse the steps
  - Code them into a programming language

The C programming language

C is

- a general-purpose imperative language
- used extensively in the development of UNIX
- has compact syntax, modern control flow and data structures and a rich set of operators
- extremely effective and expressive
- not a "very high level" nor a "big" language
- useful for embedded programming
- extensive collections of library functions

Origins of C

- Developed by Dennis Ritchie at Bell Labs
  - first implemented on DEC PDP-11 in 1972
- Based on two existing languages
  - BCPL and B languages
  - BCPL: Martin Richards, 1967 - systems programming
  - B: Ken Thomson, 1970 - early versions of UNIX

  The C Programming Language, Kernighan & Ritchie, 1978
- ANSI C: a standard adopted in 1990
  - unambiguous, machine-independent definition of C

  The C Programming Language (2nd edition) Kernighan & Ritchie, 1988

A tiny program

```
/* A first program in C */
#include <stdio.h>
main( )
{
  printf("Hello, World! \n");
}
```

A comment

Library of standard input output functions

Every C program starts execution with this function.

Statement & terminator

Escape sequence - newline

Body of the function - enclosed in braces

printf - a function from C Standard library stdio.h
- prints a char string on the standard output

Developing and running a C program

Typically six phases:

1. Edit: the program is created and stored on disk
   – Emacs, vi and gedit are popular editors on Linux
   – usually part of IDE on Windows

2a. Preprocess: handles # directives
   – include other files, macro expansions etc

2b. Compile: translates the program
   – into machine language code or object code
   – stores on disk

Other phases

2c. Link: combines
   – the object code of the program
   – object code of library functions and other functions

   creates an executable image with no "holes"

3a. Load:
   – transfers the executable image to the memory

3b. Execute:
   – computer carries out the instructions of the program

Programming Basics (*emacs* for programs)

- A variable – changes value during the execution of a program.
- A variable has a name, e.g. – *name, value, speed, revsPerSec* etc.
- Always referred to by its name
- Note: physical address changes from one run of the program to another.

Variables and Constants

Names

- made up of letters, digits and '_'
  - case sensitive: *classSize* and *classsize* are different
  - maximum size: 31 chars
- first character must be a letter
- choose meaningful and self-documenting names
  - MAX_PILLAR_RADIUS — a constant
  - pillarRadius — a variable
- keywords are reserved, cannot be used as names:
  - if, for, else, float, …

Assignments and variables

- The value of a variable is modified due to an assignment
- The LHS is the variable to be modified and the RHS is the value to be assigned
- So RHS is evaluated first and then assignment performed
- a = 1
- a = c
- a = MAX_PILLAR_RADIUS
- a = a*b + d/e   *not a mathematical equation*

Variable Declaration

- Need to declare variables.
- A declaration: *type variablename;*
- Types: *int, float, char*
- *int x;* contents of the location corresponding to x is treated as an integer. Number of bytes assigned to a variable depends on its type.
- Assigning types helps write more correct programs. Automatic type checking can catch errors like *integer = char +char;*

A more useful C program

Another simple C program

```
#include<stdio.h>
main()        // Find the square of a given number
{
   int n;     // the given number
   int sq;    // the square of n

   scanf("%d", &n);
   sq = n * n;
   printf("Square of %d = %d\n", n, sq);
}
```

A comment

A function from stdio.h

A Variable has a Type

Another simple C program: find size of variables

```
#include<stdio.h>
main()
{
 int i;
 char c;
 float f;
 printf(“int, char, float use %u, %u and %u bytes\n”,
                              sizeof(i), sizeof(c), sizeof(f));
}
```

A function from stdio.h

Exercise

- Type the above program using the *Emacs* or *gedit* editor
- Compile it using *gcc*
- Run the *executeable* file

If you already know C:

- Write a program that reads the coefficients of a quadratic and prints out its roots