EE 107 Intro. to ECET with software development, Spring 2020

Department of Electrical and Computer Engineering & Technology, MNSU
Dr. Qun Zhang

Handout 1 Background and Chapters 1 and 2 of book

Key points covered

0. Our EE/CE programs
   Sciences (Scientific methods) vs. engineering design vs. technology applications
   ECET areas / faculty expertise (see department posters in the ECET area TN 242)

1. What will be covered in the course

   a. Concepts associated with computation
   b. Computational thinking and problem solving techniques
   c. The C programming language + a little Java programming language & OOP
   d. The basics of programming
      … with digital circuit and other ECE examples throughout

2. Abstraction in ECE

   Nature; physical laws (Scientific methods; Maxwell's equations); Circuit models (Ohm's law + K's laws – workhorses of circuit simulators);

   ➔ **route 1**: amps (Shockley, Bardeen and Brattain – you will see these names again in your electronics courses); digital; combinational circuits; clocked; ISA; languages; software systems;


Shockley


Bardeen


Brattain

   ➔ **route 2**: Amp: op amps; analog systems; toasters etc. (*ECET has Drs. Wu and Zhang in the area of microwave/RF high speed analog circuit*)

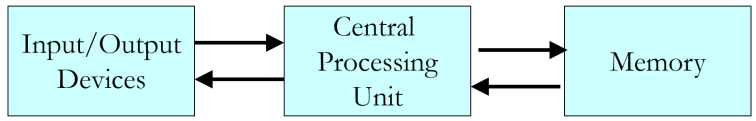
   Recap: The abstraction of combinational circuits: what we have learned? Name them; review them.

3. Question: how it goes from there to a hardware computer and software systems? Answers: 1). theory of computation (Turing machine); sequential circuits (memory); Von Neumann Architecture (implementing Turing machine); ISA (come back to this shortly); programming languages; system software and application software systems. 2). Semiconductor IC/VLSI/microelectronics (Kilby, Noyce) (*ECET has a µElectronics Lab – Drs. Khaliq and Hamari*)

Turing Von Neumann Kilby Noyce

Remark: A modern computer is an electronic machine that takes in data and instructions (input) works with the data (processing) puts out information (output)

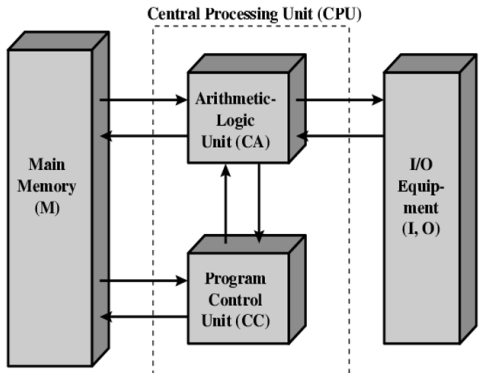
a. Computers are made of HARDWARE (physical equipment) and SOFTWARE (the instructions that tell the computer what to do: system software and application software)



Hardware piece

sw's

Remark: differences between a fixed program computer vs microprocessor/general purpose computer (stored-program computer) (Van Neumann architecture)



Programming could be facilitated if program could be represented in a form suitable for storing in memory alongside the data. Then a computer could get its instructions by reading from memory, and a program could be set or modified by setting the values of a portion of memory (proposed and applied to IAS (inst. adv. study, princeton univ.) computer by Von Neumann et al.)

Question: how abstract instructions run?

4. Program Design (computational thinking; algorithmic thinking; modeling/problem solving)

    a. computational thinking is a set of problem-solving methods that involve expressing problems and their solutions in ways that a computer could execute (will come back to this often)
        - declarative vs. imperative knowledge
        - AI: reflex based → state based → variable based → logic based
    b. Algorithmic thinking is a way of getting to a solution through the clear definition of the (a bounded number of) steps needed
        - Smart algorithms leads to high computational efficiency
    c. Combined we need to develop skills of systematical problem solving and enhance our power of abstraction.
    d. Question: can a computer store the number e? can it compute $\sqrt{2}$ or $\sin(\pi)$?

5. The hypothetical universal Turing machine
   All modern programming languages are Turing complete
   Each of them has a set of primitive constructs, a syntax, a static semantics, and semantics.

6. What is C Program Language? Its inventions and impact.


Dennis Ritchie (also inventor of Unix; Turing award)

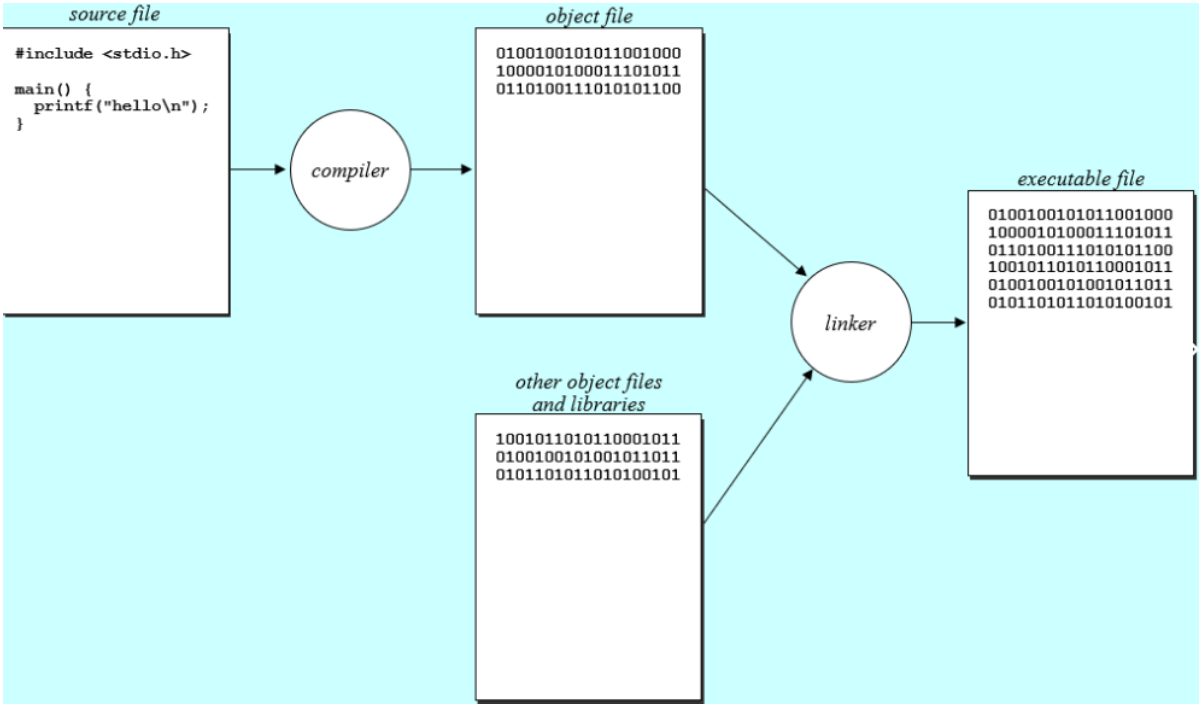
7. C as a high level language, a good language to start to learn programming [can access memory (good for microprocessor/embedded programming) and has high efficiency (good for scientific and engineering programming)]. It forms the core of Java/C++. When you are good at programming with C, you know 2/3 of Java/C++ programming (C has 1/3 for computational thinking/procedural programming, 1/3 for language syntax/semantics, the rest of the 1/3 unique to java/C++ is called object oriented programming).

8. C's use for system level programs, simulation programs, and also in embedded systems (Discussion of cyber physical systems (CPS): if you are interested to build CPS modeling/simulation software tools and have a high GPA and have extra time besides course work, please let me know and we can talk about some research projects).

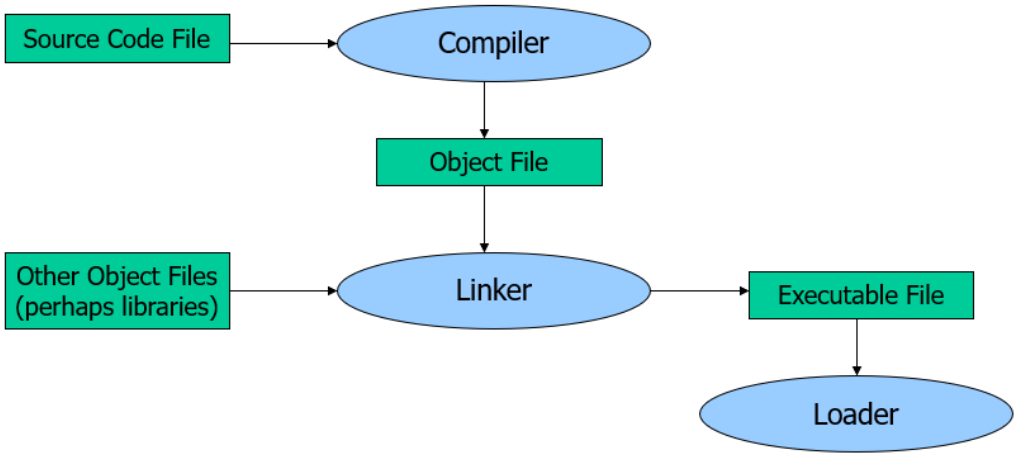
   C vs. C++/C#/Objective C vs. Java vs. R vs. Matlab vs. Python vs. Julia

   Remark: on GUI programming and ease of use of software vs. internal complexity

9. Compilers and compiling programs (Fig. 1.1)
   Another chart:



   Another chart

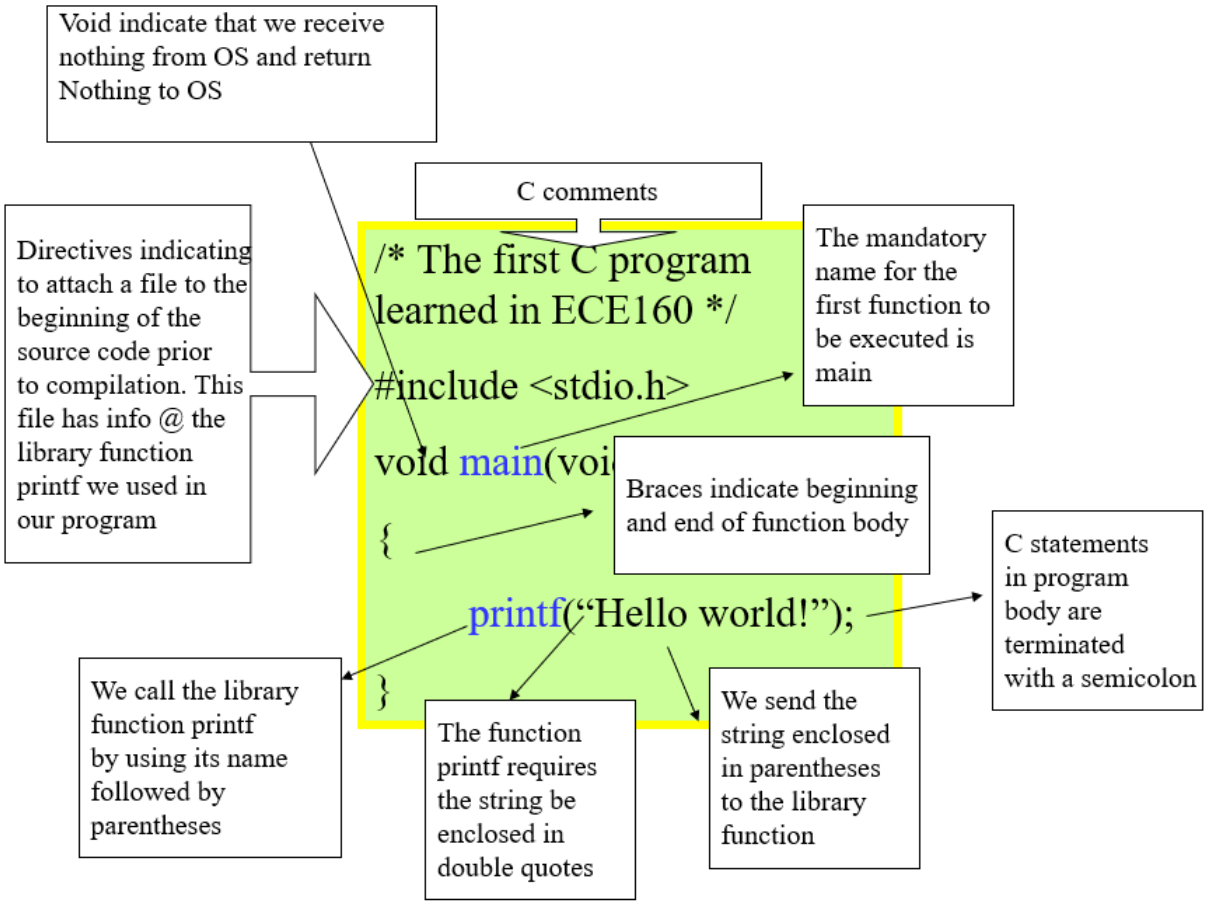


   Discussion: What is machine language? How exactly compilers work?
   Discussion: Difference between a compilation and an interpretation

10. The IDEs

11. Cygwin

12. Some basics on compiling and running C programs with hello world example



13. Understanding your first program: Programs 2.2 and 2.3

```
#include <stdio.h>

int main (void)
{
   printf ("Programming is fun.\n");
   printf ("And programming in C is even more fun.\n");

   return 0;
}
```

```
#include <stdio.h>

int main (void)
{
   printf ("Testing...\n..1\n...2\n....3\n");
```

```
    return 0;
}
```

14. Displaying the value of variables: Programs 2.4 and 2.5

```
#include <stdio.h>

int main (void)
{
    int sum;

    sum = 50 + 25;
    printf ("The sum of 50 and 25 is %i\n", sum);

    return 0;
}
```

```
#include <stdio.h>

int main (void)
{
    int  value1, value2, sum;

    value1 = 50;
    value2 = 25;
    sum = value1 + value2;
    printf ("The sum of %i and %i is %i\n", value1, value2, sum);

    return 0;
}
```

15. Comments: Program 2.6.

```
/* This program adds two integer values
   and displays the results            */

#include <stdio.h>

int main (void)
{
   // Declare variables
   int  value1, value2, sum;
```

```
    // Assign values and calculate their sum
    value1 = 50;
    value2 = 25;
    sum = value1 + value2;

    // Display the result
    printf ("The sum of %i and %i is %i\n", value1, value2, sum);

    return 0;
}
```