EE 213 Computer Organization and Assembly Language

Week # 2, Lecture # 5

25th Dhu'l-Hijjah, 1439 A.H

5th September 2018

These slides contains materials taken from various sources. I fully acknowledge all copyrights.

Minds open...

... Laptops closed

This presentation helps in delivering the lecture.

Take notes, interact and read text book to learn and gain knowledge.

Today’s Topics

- Role of Compiler
- Role of Operating System
  - Loading of program for execution
  - Creation of Process
  - Code, data, Head, Stack areas of memory accessible to a process
- Compiling HLL programs into Machine Code
- Libraries and how they are linked with your code
- Linking: Static vs Dynamic Linking
- Coverage from Chapter # 1 and Chapter # 2 of Textbook
  - Assembly Language for x86 Processors 7th Edition (available on Slate)

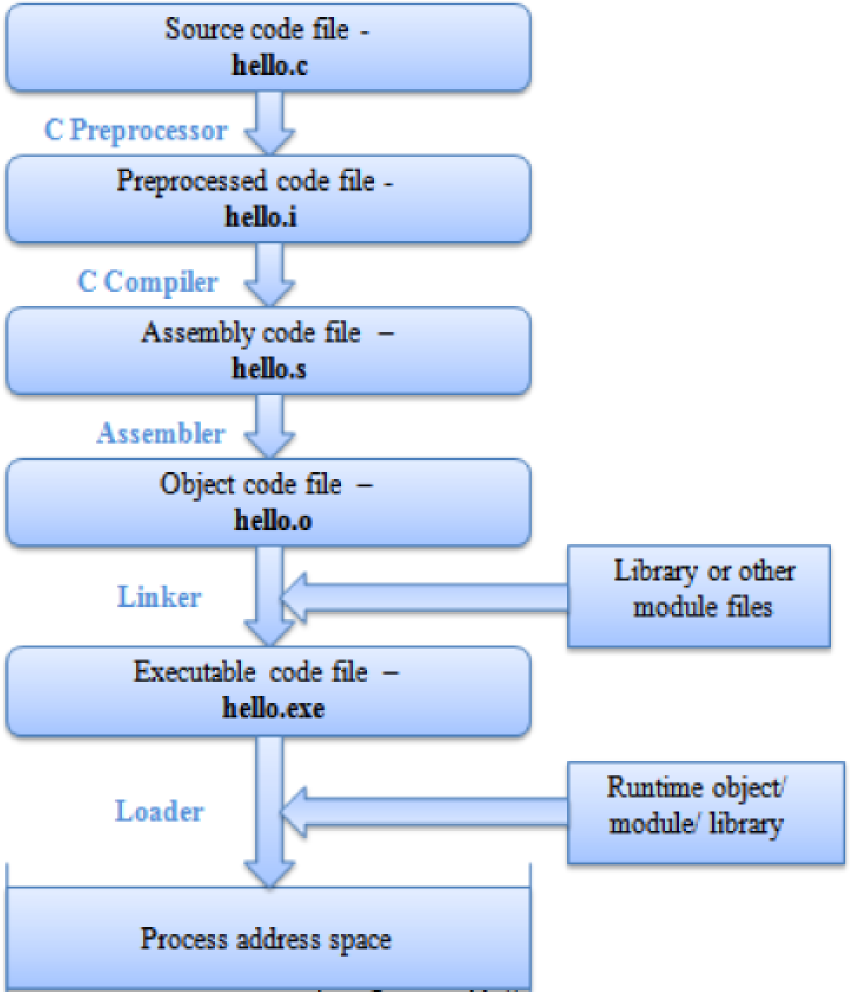
Role of Compiler

- Compiler converts high-level code into machine code (stored in .exe file) which will be executed by the processor (a complex digital circuit).
- There could be many different ways to design digital circuits. How compiler knows about the processor?
- So, there is a unique compiler for each processor. (Why?)
- Compiler read each high-level language statement and break the computation in each statement in terms of operations on data. For example, c = a + b means that there are three variable (memory locations) a, b, c and contents of a and b are added together and stored in c.
- Therefore, compiler generated code is for a specific processor. The code contains hundred of operations in specific order. The operations are in binary and act as a instruction to the processor.
- Therefore, the processor is suppose to read each instruction and execute it step-by-step and store the results internally or in memory.

Role of Operating System

- Compiler makes executable file on disk.
- Operating System (OS) reads code from disk and load it into memory.
- OS later create a process to execute the program on the processor.
- Processor (e.g. Intel Core i7, AMD, IBM, NVIDIA) executes code in memory by reading inputs: keyboard or data files on disk, etc. and generating outputs: Display, Ports (network, printer, etc.), disk, or other connected devices.
- Therefore, OS give users a user-friendly computing environment where multiple programs execute together facilitating the computer user.
- However, in this course, we are interested in understanding:
  - (40%) How internal digital circuits of a processor are organized to execute machine-code? (No circuit diagrams only block diagram of processor organization)
  - (60%) How processor perform execution steps using the internal organization when it executes each machine code instruction?

Compiling HLL programs into Machine Code



Source code file - **hello.c**

C Preprocessor

Preprocessed code file - **hello.i**

C Compiler

Assembly code file – **hello.s**

Assembler

Object code file – **hello.o**

Linker

Library or other module files

Executable code file – **hello.exe**

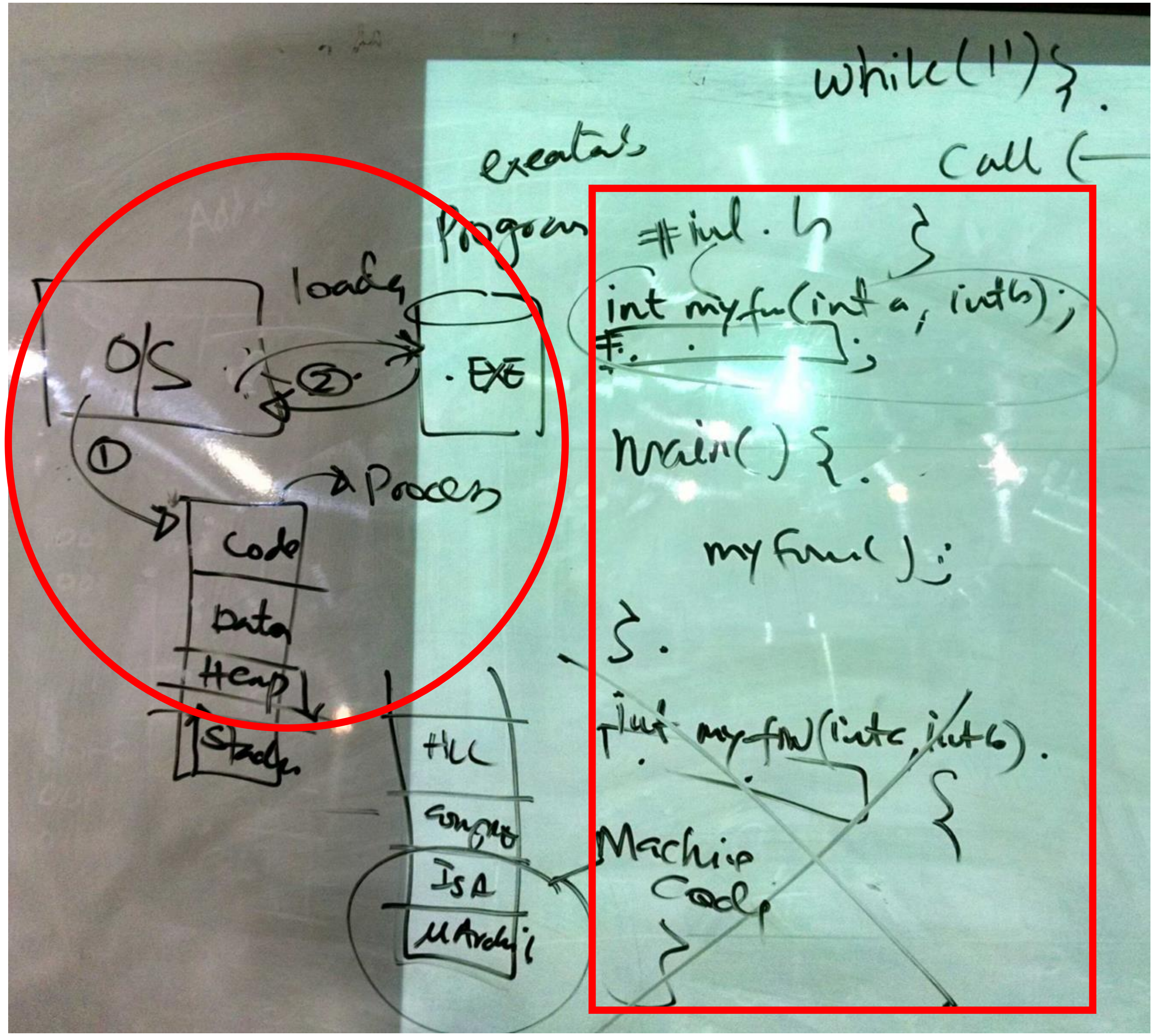
Loader

Runtime object/ module/ library

Process address space

- Object code files contain unresolved reference to external functions which are in libraries.
- The linker takes all object code files search given libraries to get executable code and create a unified executable (.EXE) file.

Libraries and how they are linked with your code

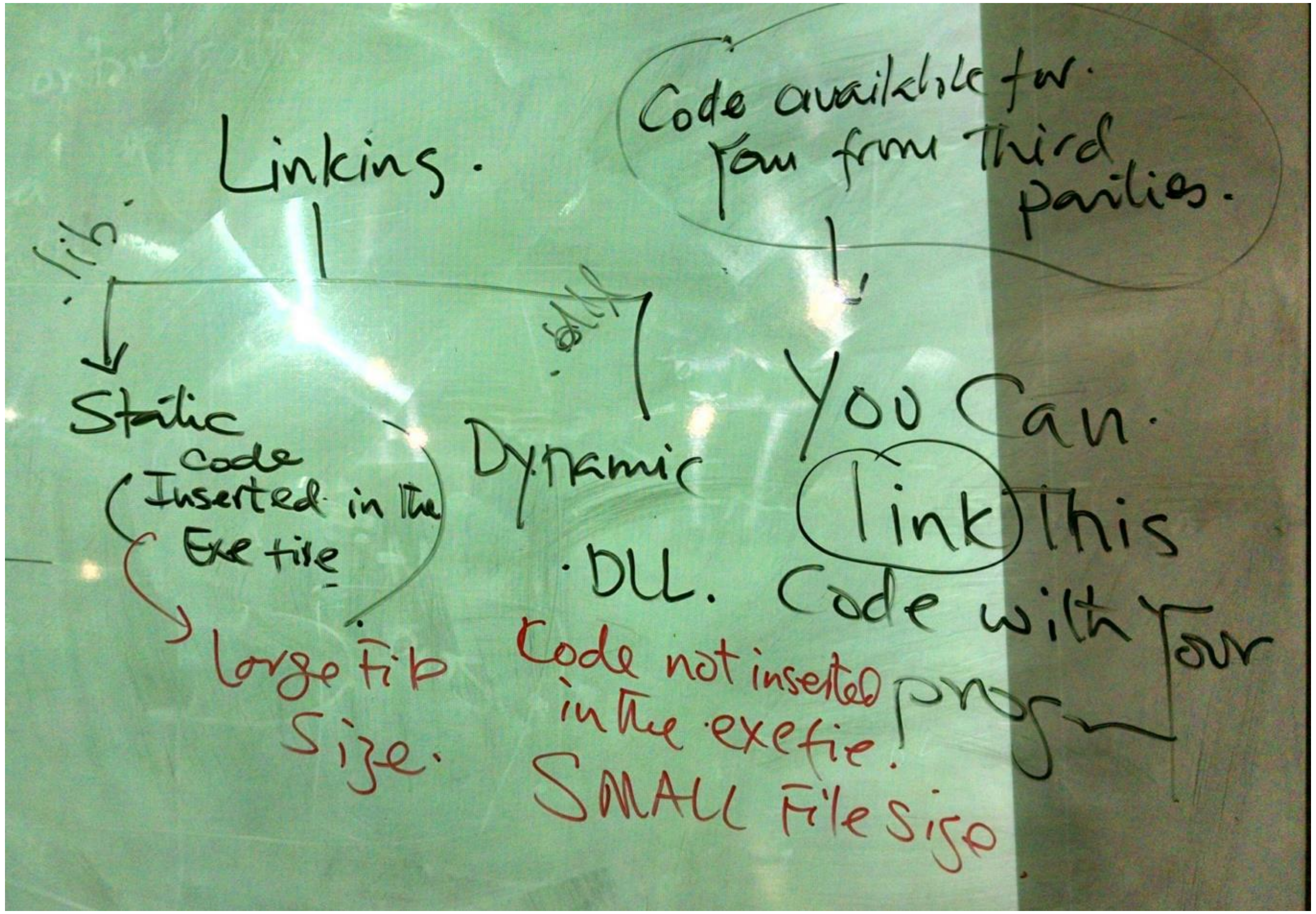


- How OS create a process and load the executable program from the file system.

- Header files included in your program to supply the definitions and declarations you need to invoke system calls and libraries.

- See textbook chapter # 2 for more details.

Linking: Static vs Dynamic Linking

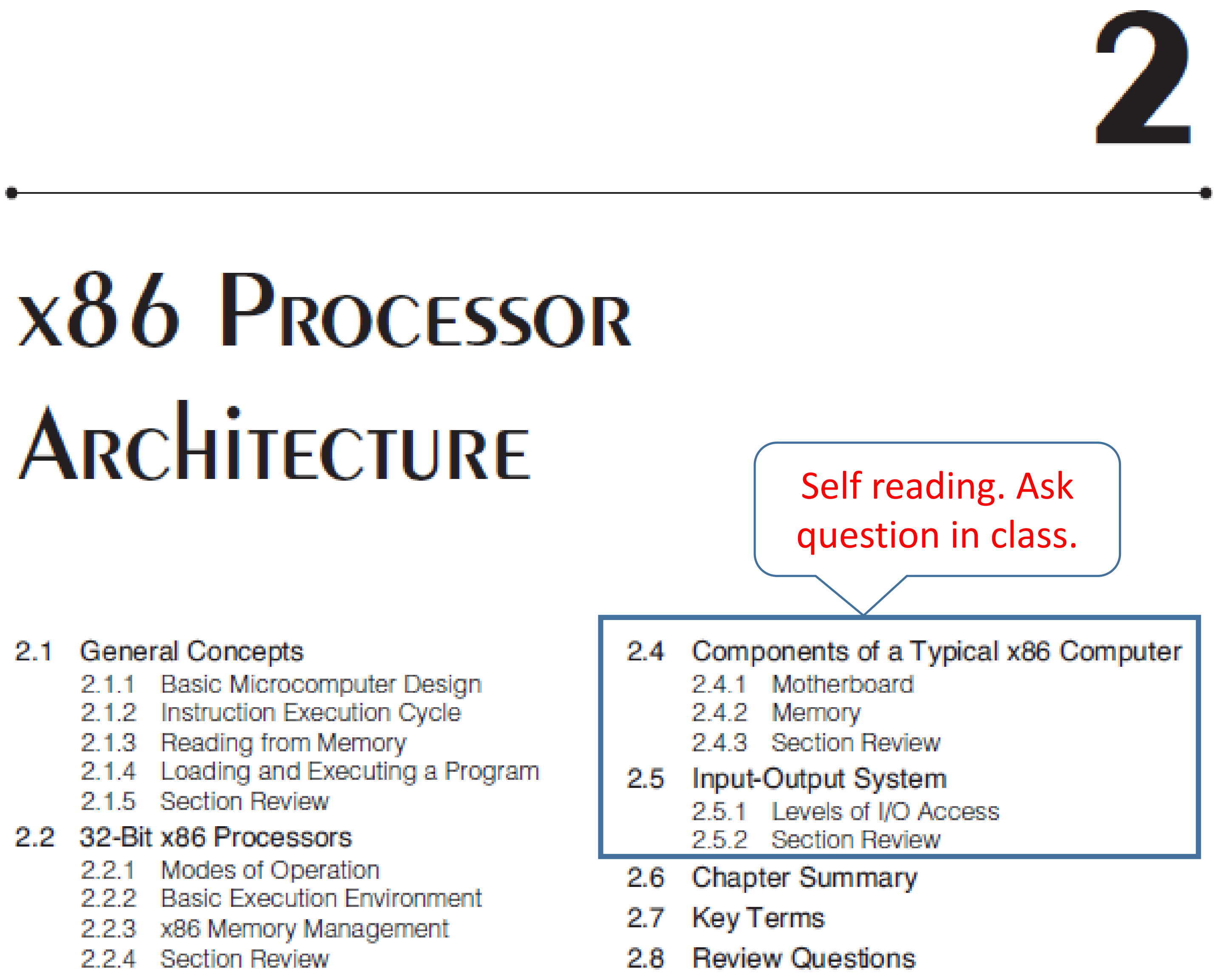


- Linking can be static linking (compile type) or dynamic linking (runtime using dynamic link libraries)

- Libraries contain executable code where the coder doesn’t want to reveal the source code. However, you can linked executable code into your program.

1

Basic Concepts



2

x86 Processor Architecture

Self reading. Ask question in class.