Ch 7 Assembly language level

Intro definitions:
- translator, source language, target language
- object program, executable
- assembly language, assembler, compiler, pseudo-instructions, assembler directives
- macro definition, macro call, macro expansion, formal parameters, actual parameters

Assembly statements (one per line) typically contain: label field (optional), operation, operands, and comments (optional).

Macros - simple text replacement (copying), expanded during assembly (not program exec), can have parameters

7.3 The assembly process

This section has a direct impact on our Program #4. We'll discuss more details there.

Definitions:
- two-pass translator, forward reference problem, instruction location counter (ILC)
- associative memory, hash table, binary search

Tables kept/used by Pass 1 = symbol table, pseudo-instruction table, opcode table, literal table
The tables are used in Pass 2 (duh)

Typical errors: Symbol used but not defined, symbol defined more than once, illegal opcode, missing operands, too many operands, invalid token, bad pseudo-instruction, etc.

7.4 Linking and loading

Linker - generates an executable binary program from a collection of independently translated object modules. *(Figure 7-12 below)*

The bogey - understand the linking example laid out in Figure 7-13 and 7-14.

Relocation problem - the merging of each object file's separate address space into a single linear address space
External reference - (usually) a call to another, outside procedure in an object file

Binding time - when symbolic names are turned into actual addresses: 1) when the program is written, 2) when translated, 3) when linked, 4) when loaded, 5) when a base registers is used for addressing, 6) when an instruction is executed (dynamic!).

Dynamic linking - links each procedure at the time it is first called. (Figure 7-17 example)

Windows = **Dynamic Link Library (DLL)**

- Contain procedures and data
- Loaded into memory when called and accessed by multiple processes at the same time
- Advantages: save space in memory, disk, easier to update,
- Examples: Windows system libraries, graphics libraries, fonts, etc.

Unix = **Shared library**

implicit linking - static link to import library that provides glue to DLL's
explicit linking - run-time binding to library

Figure 7-17: Dynamic linking	Figure 7-18: DLL example