Computer Systems
Lecture 3

Overview

- Machine instructions and HLL
- Semantic gap
- Translation
- Linking
- Library files
- Interpreters
- Interpreters vs. Compilers
- Code sharing and reuse
- Dynamic libraries and dynamic linking

Machine instructions and HLL

- In the previous lecture we have seen:
  - **H**igh **L**evel Programming **L**anguages (HLLs) are more suitable for programming than the languages of machine instructions.
  - Examples: FORTRAN, COBOL, C, C++, Java, Perl,…
  - Programs in HLL still have to be translated to the machine codes. (You know why ,,,)

Semantic gap

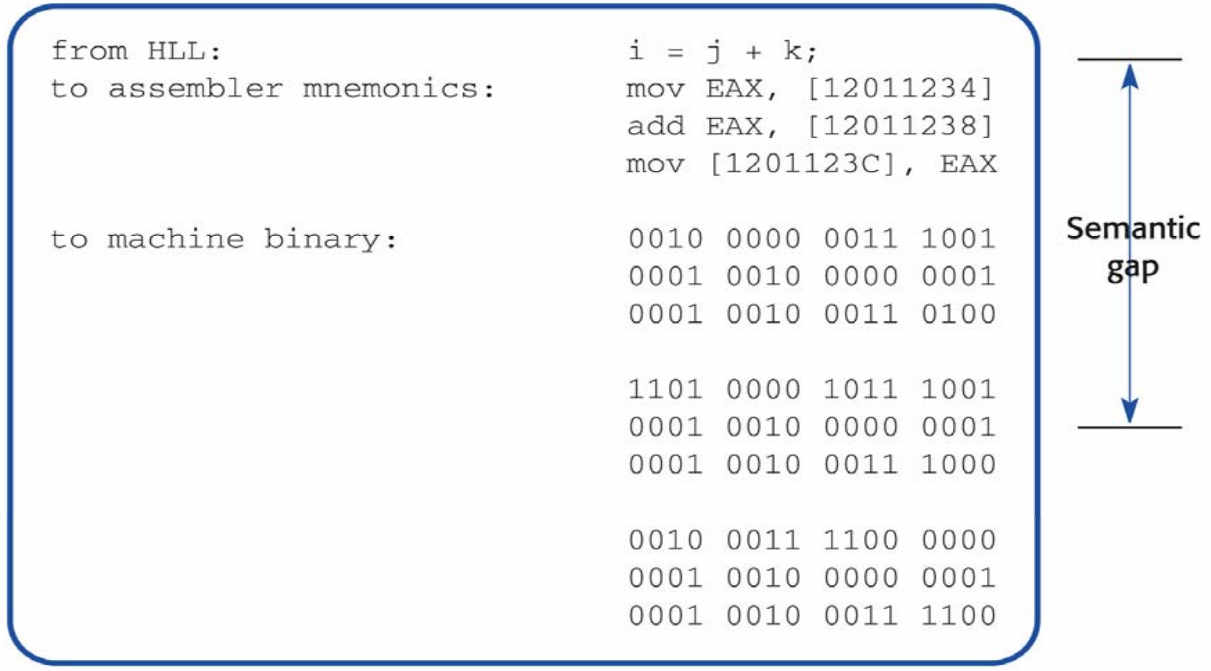

Fig. 2.3 Equivalent HLL, assembler and machine code.

- The term expresses the enormous difference between the way human languages expressing ideas and actions and the way computer instructions representing data processing activities.

Translation

- Translation is done by special programs such as:
  - **Compilers**, translating HLL instructions into machine code (sequence of instructions) before the code can be run on the machine.
  - **Assemblers,** translating mnemonic form of machine instructions (like MOV, ADD, etc) into their binary codes.
  - **Interpreters**, translating HLL instructions into machine code on-the-fly (while the program is running).

Translation – compilers and assemblers

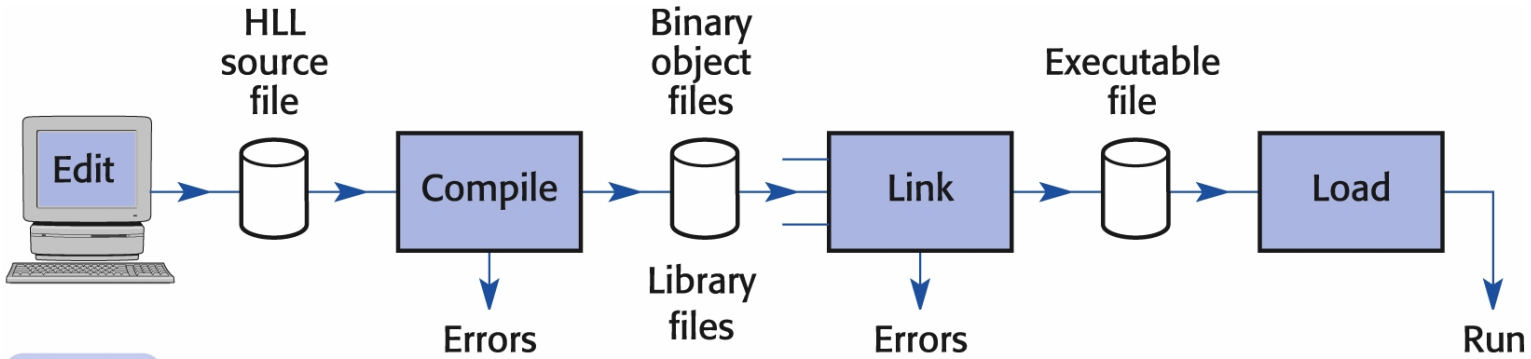

Fig. 2.4 How executable code is produced using compilers.

- What do we do with compile-time errors?
- Linking: resolving external references.
- What do we do with link-time errors?
- What do we do with load-time errors?

Linking

- Big programs usually are divided into several separate parts or **modules.** (Why?)
- Each module has to be designed, coded and compiled.
- There are frequent occasions when code in one module needs to reference data or subroutines in another module.
- (See the example in the following page.)

Linking (cont.)

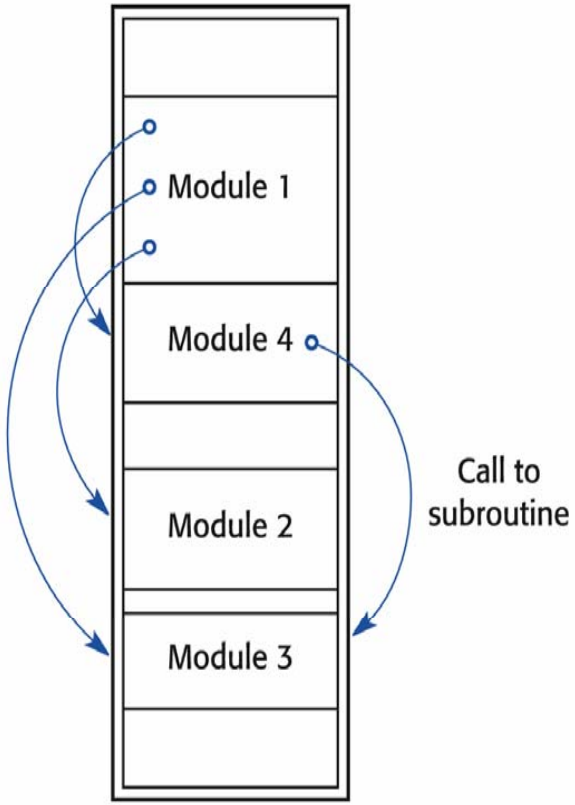

Fig. 2.5 Modules with external references.

Linking (cont.)

- A compiler can translate a module into binary codes, but it cannot resolve those references to other modules. (Why?)
- Those *external references* remain symbolic after the compilation, until the linker gets to work.
  - The linker is to join together all the binary parts.
  - The linker will report errors if it cannot find the module or code referred to by those external references.

Library files

- Library files.
  - Translated object code.
  - Provide many functions for programmers, but are only usable if linked into your code. (Why?)
  - In Unix:
    - Directories `/lib` and `/usr/lib/`.
  - In Windows:
    - DLL files.

Output log of a building process by Microsoft Visual C++ 6.0

Deleting intermediate files and output files for project 'voronoi - Win32 Release'.

--------------------Configuration: voronoi - Win32 Release--------------------

Compiling resources...

Compiling...

StdAfx.cpp

Compiling...

…

DialogConfirmationUniform.cpp

DialogDistribution.cpp

E:\voronoi\voronoiDoc.h(255) : warning C4244: 'return' : conversion from 'double' to 'float',
possible loss of data

…

Generating Code...

Linking...

voronoi.exe - 0 error(s), 21 warning(s)

Interpreters – alternative way of running HLL programs

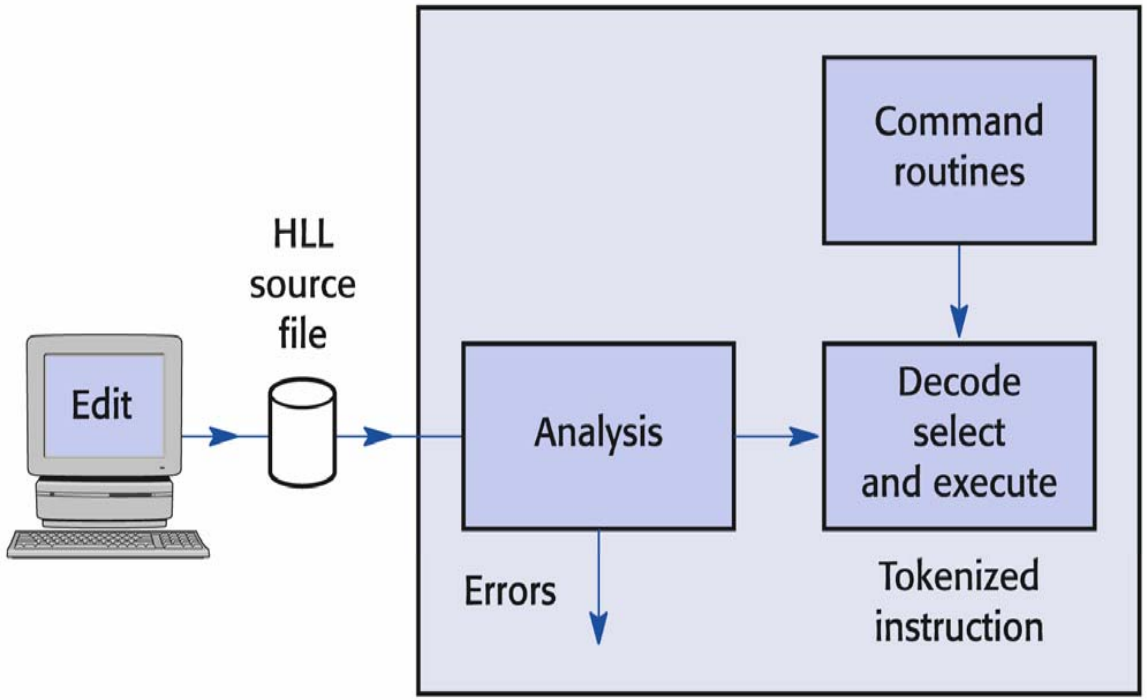

Fig. 2.6 Using an interpreter to translate and execute a program.

A Sample Java Program

```
class Motto {
public static void main(String[] args)
   { System.out.println("Java is not
   everything!");
   }
}
```

Interpreters

- Interpreters.
  - Such as used with BASIC and Java.
- Instructions are converted into an *intermediate form*, consisting of *tokens*.
  - In Java, tokens such as: static, boolean, file, string, void, return
- Tokens are then passed to the decoder, which selects appropriate routines for execution.

Interpreters (cont.)

- **Compilers**.
  - Take a program and translate it as a whole into machine code.
  - The processes of translation and execution are separate.
- **Interpreters**.
  - Take an instruction, one at a time, translate and execute it.
  - The processes of translation and execution are interlaced.

C program compilation, linking & execution

- C language source code → compiler (program) → assembly language → assembler → machine code
- Once we have machine code:
- machine code → linking and loading (program) → program code execution (program)

Java

Java source code → compiler (program) → Java "byte codes" → Java interpreter (program)

Java: compilers and interpreters

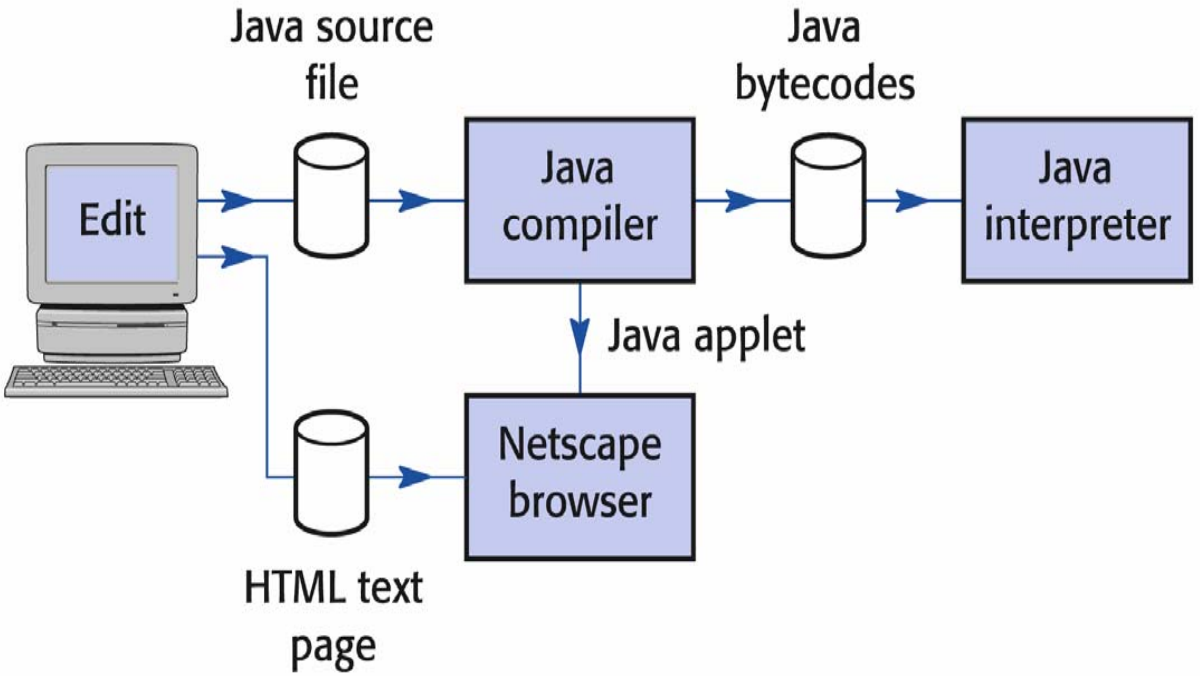

Fig. 2.7 Java uses compilers and interpreters.

Interpreters vs. Compilers

- Execution of compiled code is much faster than execution of interpreted code. (Why?)
  - But this disadvantage of interpretation is not a big problem for most applications.
- Interpreters are more suitable for rapid prototyping and for other situations when a program is frequently modified.
  - Interpreters are more accurate in terms of error reporting. (Why?)
  - Interpretation can provide uniform execution environment across several diverse computers. (Portable)

Interpreters as Virtual Machines

- Interpreters are somewhat similar to the computer hardware (CPU)
  - take one instruction at a time and execute it.

- Because of that sometimes they are referred to as a **virtual machine**
  - Example: JVM, Java Virtual Machine

Code sharing and reuse

- The problem.
  - How to **reuse** existing proven software when developing new systems?

- Three solutions.
  - Source-level subroutines and macro libraries.
  - Pre-translated re-locatable binary libraries.
  - Dynamic libraries and dynamic linking.

How code can be shared

- A summary diagram.

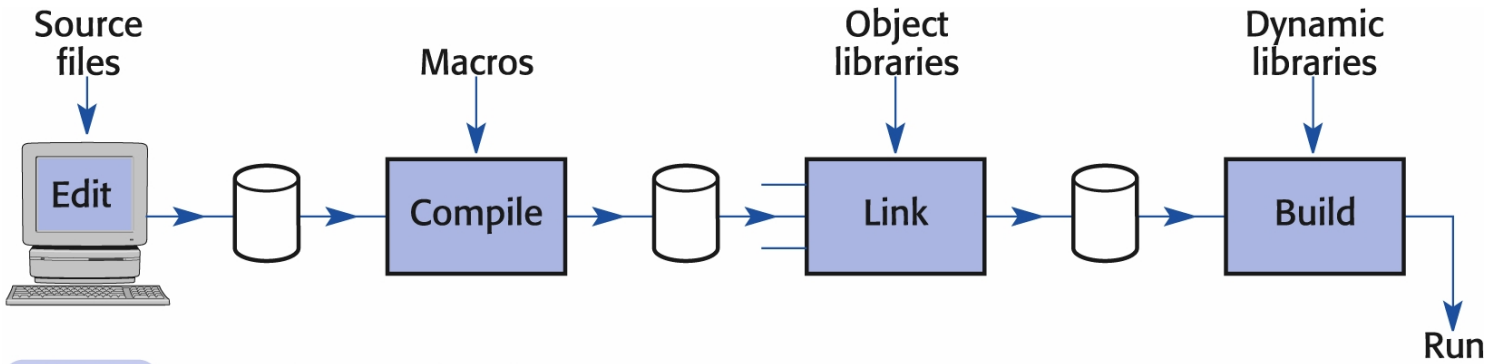

Fig. 2.8 How code can be shared.

Macro example

```
macro Print_strings (s1, s2)
printf(“%s”, s1);
printf(“ %s”, s2);
End macro
```

```
main()
{
string a, b;
 …
Print_strings (a, b);
…
Print_strings(“Hello”,”world”);
…
Print_strings(“Game”,”over”);
…
}
```

Expanded macro

```
main()
{
string a, b;
 …
Print_strings (a, b);
…
Print_strings(“Hello”,”world”);
…
Print_strings(“Game”,”over”);
…
}
```

```
main()
{
string a, b;
 …
printf (“%s”, a);
printf ( “%s”, b);
…
printf (“%s”, “Hello”);
printf (“ %s”, ”world”);
…
printf (“%s”, “Game”);
printf (“ %s”, ”over”);
…
}
```

Exercises

- Expand the following macros
  - Print_strings(“Hello my”, “little boy”);
  - Print_strings(“Game is not”,”over yet …”);

Source-level subroutines and macro libraries

- Intention.
  - Take copies of the library routines.
  - Edit them into your new code.
  - Translate the whole together.
- Disadvantages.
  - Who owned the code?
  - Who should maintain it?

Pre-translated relocatable binary libraries

- Intention.
  - Libraries are pre-translated into relocatable binary code.
  - Can be linked into your new code, but not altered.
- Acceptance.
  - Successful, and still essential for all software development undertaken today.
- Disadvantage.
  - Each program is to have a private copy of the subroutines, wasting valuable memory space, and swapping time, in a multitasking system.
- ‘Relocatable’ means?

Dynamic libraries and dynamic linking

- Intention.
  - Load a program which uses “public” routines already loaded into memory.
  - The memory-resident libraries are mapped, through the memory management system to control access and avoid multiple code copies.
  - Mapped into what?
- Acceptance.
  - Successful through Microsoft’s ActiveX standard.

Q&A

- Name 4 examples of HLLs.
- Translation fills in the semantic gap in computer systems. (True of false?)
- Name 3 different ways of translation. Identify the crucial role of translation under each.
- When compile-time errors occur, what do we do?
- What are the purposes of linking?

Q&A

- ‘Loading’ is carried out before ‘linking’ after a program is compiled. (True or false?)
- Program modules can be compiled separately. (True or false?)
- A compiler can translate a module into binary codes, but it cannot resolve those references to other modules. This occurs when …?
- Library files are usable if linked into your program code. (True or false?)

Q&A

- Interpreters typically convert program code into what?
- What is the output of program compilation?
- Name 2 scenarios where interpreters are more useful than compilers.
- Interpreters are sometimes called as virtual machine because … ?
- Mention 3 approaches to code sharing.
- Name 2 disadvantages (or issues) of macro libraries.

Q&A

- Libraries can be linked into your program code, but not altered. (True or false?)
- What are the disadvantages of program execution with pre-translated program library?
- There exists a de-facto standard of dynamic libraries and dynamic linking. (True or false?)

Readings

- [Wil06] Chapter 2, sections 2.4-2.7.
- Wikipedia.
  - http://en.wikipedia.org/wiki/Compiler
  - http://en.wikipedia.org/wiki/Interpreted_language