System Software

SYSTEM SOFTWARE: AN OVERVIEW
ASSEMBLERS AND ASSEMBLY LANGUAGE
OPERATING SYSTEMS
IMPLEMENTING SYSTEM SOFTWARE
System Software: An Overview

• “Naked” computer hard to deal with, e.g.,
  1. Write machine language program.
  2. Load program into memory starting at address 0.
  3. Load 0 into PC and start execution.

• Need virtual machine interface, which does the following:
  • Hides details of machine operation.
  • Does not require in-depth knowledge of machine internals.
  • Provides easy access to system resources.
  • Prevents accidental or intentional damage to hardware, programs, and data.

• Create virtual machine and associated interface with system software.
System Software: An Overview (Cont’d)

Figure 6.1 The Role of System Software
System Software: An Overview (Cont’d)

- System software provided by **Operating System (OS)**.
- Many types of system software in an OS, e.g.,
  - **Graphical User Interface (GUI)**: Access system services.
  - **Language services**: Allow programming in high-level languages, e.g., text editor, assembler, loader, compiler, debugger.
  - **Memory manager**: Allocate memory for programs and data and retrieve memory after use.
  - **Information manager**: Organize program and data files for easy access, e.g., folders, directories.
  - **I/O system manager**: Access I/O devices.
  - **Scheduler**: Manage multiple active programs.
System Software: An Overview (Cont’d)

OS dramatically simplifies creation of software, e.g.,

1. Write **source program** \( P \) in high-level programming language using a text editor.
2. Use an information manager to store \( P \) as a file in a directory.
3. Use a compiler and an assembler to translate \( P \) into an equivalent machine language program \( M \).
4. Use scheduler to load, schedule, and run \( M \) (with scheduler calling memory manager and loader).
5. Use I/O system manager to display output on screen.
6. If necessary, use debugger to isolate and text editor to correct program errors.
Assemblers and Assembly Language

• An assembly language is the human-friendly version of a machine language, courtesy of several features:
  
  • Symbolic op-codes, e.g., ADD, COMPARE;
  • Symbolic memory addresses and labels, e.g., IND, ONE, AFTERLOOP; and
  • **Pseudo-ops** which specify extra assembler directives, e.g., .DATA, .BEGIN, .END.

• An assembler converts an assembly language source program into a machine language **object program**; a loader then places the instructions in that object program in the specified memory addresses.
Assemblers and Assembly Language (Cont’d)

Figure 6.3
The Continuum of Programming Languages
Assemblers and Assembly Language:
Example Assembly Language Code

set $A$ to the value of $B + C$

```
LOAD  B
ADD   C
STORE A
...
A: .DATA 1
B: .DATA 2
C: .DATA 3
```
Assemblers and Assembly Language: Example Assembly Language Code (Cont’d)

if $A > B$ then
    set $C$ to the value of $A$
else
    set $C$ to the value of $B$

LOAD B
COMPARE A
JUMPGT IFPART
LOAD B
STORE C
JUMP ENDF

IFPART:
LOAD A
STORE C

ENDIF:

A: .DATA 1
B: .DATA 2
C: .DATA 3
set IND to 0
while IND ≤ MAXIND do
  ⟨LOOPBODY⟩
  set IND to IND + 1

CLEAR IND
LOOPSTART: LOAD MAXIND
COMPARE IND
JUMPGT LOOPEND
⟨LOOPBODY⟩
INCREMENT IND
JUMP LOOPSTART
LOOPEND:  ⋯
  ⋯
IND: .DATA 0
MAXIND: .DATA 25
Assemblers and Assembly Language: An Assembly Language Program

Consider the following algorithm for computing and printing the sum of all values in a \(-1\)-terminated list:

<table>
<thead>
<tr>
<th>Step</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Set $SUM$ to 0</td>
</tr>
<tr>
<td>2.</td>
<td>Read the first list value into $CURVAL$</td>
</tr>
<tr>
<td>3.</td>
<td>while ($CURVAL \neq -1$) do</td>
</tr>
<tr>
<td>4.</td>
<td>Set $SUM$ to $SUM + CURVAL$</td>
</tr>
<tr>
<td>5.</td>
<td>Read the next list value into $CURVAL$</td>
</tr>
<tr>
<td>6.</td>
<td>Print the value of $SUM$</td>
</tr>
<tr>
<td>7.</td>
<td>Stop</td>
</tr>
</tbody>
</table>

Let’s implement this algorithm in assembly language.
Assemblers and Assembly Language:
An Assembly Language Program (Cont’d)

Step 1
SUM: .DATA 0
CURVAL: .DATA 0
ENDVAL: .DATA -1
.END

Step 2
.IN CURVAL

Step 3
LOOPSTART: LOAD ENDVAL
COMPARE CURVAL
JUMPEQ LOOPEND

Step 4
LOAD SUM
ADD CURVAL
STORE SUM

Step 5
.IN CURVAL
JUMP LOOPSTART

Step 6
LOOPEND: OUT SUM

Step 7
HALT
Assemblers and Assembly Language: The Big Picture

**Figure 6.4**
The Translation/Loading/Execution Process (Assembly --> M.C.)
Operating Systems

Major duties of an operating system:

- **User Interface**: Accept system commands from user and, if these commands are valid, schedule appropriate system software to execute command.

- **System Security and Protection**: Determine valid users and valid activities and accesses for users using usernames, passwords, and access control lists.

- **Efficient Management of Resources**: Optimize processor use by maintaining Running (active program), Ready (programs ready to execute), and Waiting (programs waiting on I/O requests) queues.

- **Safe Use of Resources**: Prevent deadlock (two or more users have partial required resources) using resolution algorithms and protocols.
Implementing System Software: Compilers

Grace Hopper (1906–1992)

- A compiler translates a program in a high-level programming language into a behaviorally equivalent program in a lower-level programming language.
- First compilers developed by Grace Hopper in early 1950s.
- Compilers can be cascaded, e.g., high-level language $\Rightarrow$ medium-level language $\Rightarrow$ assembly language $\Rightarrow$ machine language.
Implementing System Software: Programming Languages

- **FORTRAN** (FORmula TRANslation) created by Backus team at IBM in 1957; designed for scientific computation.
- **COBOL** (CCommon Business-Oriented Language) created by industry / government committee in 1959.
Implementing System Software: Programming Languages (Cont’d)

- BASIC (Beginner’s All-purpose Symbolic Instruction Code) created by Thomas Kurtz (1928–) and John Kemeney (1926-1992) at Dartmouth College in 1964.
- Designed as a programming language for everyone.
Implementing System Software: Operating Systems

- OS only possible after sufficient computer memory available for system software starting around 1955.
- Three OS generations to date:
     Run multiple programs in sequence with aid of Job Control Language (JCL).
     Run multiple programs in apparent parallel by swapping programs in and out of the control unit.
- Future OS will incorporate multimedia user interfaces (e.g., voice / gesture-based) and fully distributed execution.
Doug Engelbart (1925-2013)  
Computer Mouse (1965)  

- Engelbart and colleagues develop graphical user interface (GUI) and computer mouse at Stanford starting in 1963.
Implementing System Software: User Interfaces (Cont’d)

“The Mother of All Demos” (1968)
Implementing System Software: User Interfaces (Cont’d)

Xerox Alto (1973) [$25K (est)]

- Alto was first modern GUI-driven PC; also incorporated local-area networking and laserjet printers (WYSIWYG).

Xerox Star (1981) [$75K]

- Star intended for use in large corporations.
Implementing System Software: User Interfaces (Cont’d)
Implementing System Software: User Interfaces (Cont’d)

- Starting in 1979, Steve Jobs re-creates GUI-based functionality at Apple in the Lisa and Macintosh PCs.
- Part of Macintosh application and OS development subcontracted to Microsoft starting in 1981.

Apple Macintosh (1984) [$2,500]
Implementing System Software: User Interfaces (Cont’d)

• Microsoft releases Windows v1.0 in 1985; legally emulated portions of Lisa and Mac look.

• Microsoft releases Windows v2.0 in late 1987; is not only much faster but (now illegally) identical to Mac look.


• By late 1980s, Windows has 90% market-share in GUI-based PC computing.
... And If You Liked This ...

- MUN Computer Science courses on this area:
  - COMP 2001: Object-oriented Programming and HCI
  - COMP 2003: Operating Systems
  - COMP 3300: Interactive Technologies
  - COMP 4712: Compiler Construction

- MUN Computer Science professors teaching courses / doing research in in this area:
  - Ed Brown
  - Rod Byrne
  - Oscar Meruvia-Pastor