Chapter 2 The OS, the Computer, and User Programs

- Fundamental Principles of OS Operation
- The Computer
- OS Interaction with the Computer and User Programs

Fundamental Principles of OS Operation

- The *kernel* of the OS is the collection of routines that form the core of the operating system
  - Implements *control functions*
  - Offers a set of services to user programs
  - Exists in memory during operation of the OS
- An *interrupt* diverts the CPU to execution of the kernel code
  - A *software interrupt* is used by programs to communicate their requests to the kernel
Fundamental Principles of OS Operation (continued)

• CPU has two modes of operation:
  – *Kernel mode*
    • CPU can execute all instructions
    • Kernel operates with CPU in this mode so that it can control operations of the computer
  – *User mode*
    • CPU cannot execute instructions that could interfere with other programs or with the OS if used indiscriminately; these are called *privileged instructions*.
    • CPU is put in this mode to execute user programs

The Computer

• The CPU
• Memory Management Unit (MMU)
• Memory Hierarchy
• Input/Output
• Interrupts
The CPU

- Two features of the CPU are visible to user programs or the OS:
  - General-purpose registers (GPRs)
    - Also called program-accessible registers
    - Hold data, addresses, index values, or the stack pointer during execution of a program
  - Control registers
    - Contain information that controls or influences operation of the CPU
    - Set of control registers is called the program status word (PSW)
    - An individual control register is referred to as a field of the PSW
The CPU (continued)

- CPU can operate in two modes:
  - *Kernel mode*
    - Can execute privileged instructions
    - OS puts CPU in kernel mode when it is executing instructions in the kernel
  - *User mode*
    - Cannot execute privileged instructions
    - OS puts CPU in user mode while executing user programs
- *Mode* (M) field of PSW contains 0 if CPU is in kernel mode and 1 if it is in user mode
State of the CPU

- GPRs and PSW contain the information needed to know what the CPU is doing
  - State of the CPU
- Kernel saves state of CPU when it takes away the CPU from program
  - When program is to be resumed, it reloads the saved CPU state into GPRs and PSW

Example 2.1: State of the CPU

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>PSW</th>
<th>registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>0142</td>
<td>MOVE A, ALPHA</td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>0146</td>
<td>COMPARE A, 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0150</td>
<td>BEQ NEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0192</td>
<td>NEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0210</td>
<td>ALPHA DCL_CONST 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.3 (a) Listing of an assembly language program showing address assigned to each instruction or data; (b) state of the CPU after executing the COMPARE instruction.
Memory Management Unit (MMU)

- *Virtual memory* is an illusion of memory that may be larger than the real memory of a computer
  - Implemented using noncontiguous memory allocation and the MMU
    - CPU passes the address of data or instruction used in current instruction to MMU
      - It is called the *logical address*
    - MMU *translates* logical address to *physical address*

Memory Hierarchy

- The *memory hierarchy* provides a large and fast memory, at a low cost
  - It is an arrangement of several memory units with different access speeds and sizes
    - The CPU accesses only the fastest memory; i.e., the *cache*
    - If a required byte is not present in the memory being accessed, it is loaded there from a slower memory
Memory Hierarchy (continued)

- When CPU performs a cache lookup, a cache hit or miss may occur
  - Hit ratio \( h \) of the cache is the fraction of bytes accessed by the CPU that score a hit in the cache
    
    \[
    t_{\text{ema}} = h \times t_{\text{cache}} + (1 - h) \times (t_{\text{tra}} + t_{\text{cache}}) \\
    = t_{\text{cache}} + (1 - h) \times t_{\text{tra}}
    \]

    where
    
    \( t_{\text{ema}} \) = effective memory access time,  
    \( t_{\text{cache}} \) = access time of cache, and  
    \( t_{\text{tra}} \) = time taken to transfer a cache block from memory to cache.
Memory Hierarchy (continued)

• Operation of memory is analogous to operation of a cache
  – Blocks of bytes (pages) are transferred from disk to memory or from memory to disk
  – But,
    • Memory management and transfer of blocks between memory and disk are performed by software
    • In the cache, the transfer is performed by hardware
• Memory hierarchy comprising MMU, memory, and the disk is called the virtual memory

Memory Hierarchy (continued)

• Memory protection is implemented by checking whether a memory address used by a program lies outside the memory area allocated to it
• Two control registers used: base and size (also called limit)
  – Address of first byte = <base>
  – Address of last byte = <base> +<size> – 1
• Base and size registers contained in the memory protection information (MPI) field of PSW
• Instructions to load values into the base and size registers are privileged instructions
Example 2.2: Basics of Memory Protection

- Execution of Load instruction causes protection violation

![Diagram](image)

**Figure 2.5** Memory protection using the base and size registers.