Chapter 1
Computer System Overview

Eighth Edition
By William Stallings
Operating System

- Exploits the hardware resources of one or more processors
- Provides a set of services to system users
- Manages secondary memory and I/O devices
Basic Elements

- Processor
- Main Memory
- I/O Modules
- System Bus
Processor

Controls the operation of the computer

Performs the data processing functions

Referred to as the *Central Processing Unit* (CPU)
Main Memory

- Volatile
- Contents of the memory is lost when the computer is shut down
- Referred to as real memory or primary memory
I/O Modules

Moves data between the computer and external environments such as:

- storage (e.g. hard drive)
- communications equipment
- terminals
System Bus

- Provides for communication among processors, main memory, and I/O modules
Figure 1.1  Computer Components: Top-Level View

PC  =  Program counter
IR  =  Instruction register
MAR  =  Memory address register
MBR  =  Memory buffer register
I/O AR  =  Input/output address register
I/O BR  =  Input/output buffer register
Microprocessor

- Invention that brought about desktop and handheld computing
- Processor on a single chip
- Fastest general purpose processor
- Multiprocessors
- Each chip (socket) contains multiple processors (cores)
Graphical Processing Units (GPU’s)

- Provide efficient computation on arrays of data using Single-Instruction Multiple Data (SIMD) techniques
- Used for general numerical processing
- Physics simulations for games
- Computations on large spreadsheets
Digital Signal Processors (DSPs)

- Deal with streaming signals such as audio or video
- Used to be embedded in devices like modems
- Encoding/decoding speech and video (codecs)
- Support for encryption and security
System on a Chip (SoC)

- To satisfy the requirements of handheld devices, the microprocessor is giving way to the SoC

- Components such as DSPs, GPUs, codecs and main memory, in addition to the CPUs and are on the same chip
A program consists of a set of instructions stored in memory.

Two steps:
- Processor reads (fetches) instructions from memory.
- Processor executes each instruction.
Figure 1.2  Basic Instruction Cycle
Instruction Fetch and Execute

- The processor fetches the instruction from memory
- Program counter (PC) holds address of the instruction to be fetched next
  - PC is incremented after each fetch
Instruction Register (IR)

- Fetched instruction is loaded into Instruction Register (IR)
- Processor interprets the instruction and performs required action:
  - Processor-memory
  - Processor-I/O
  - Data processing
  - Control
(a) Instruction format

(b) Integer format

Program counter (PC) = Address of instruction
Instruction register (IR) = Instruction being executed
Accumulator (AC) = Temporary storage

(c) Internal CPU registers

0001 = Load AC from memory
0010 = Store AC to memory
0101 = Add to AC from memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine
<table>
<thead>
<tr>
<th>Fetch Stage</th>
<th>Execute Stage</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Memory</strong></td>
<td><strong>CPU Registers</strong></td>
</tr>
<tr>
<td>300</td>
<td>PC 300</td>
</tr>
<tr>
<td>301</td>
<td>1 9 4 0</td>
</tr>
<tr>
<td>302</td>
<td>2 9 4 1</td>
</tr>
<tr>
<td>940</td>
<td>0 0 0 3</td>
</tr>
<tr>
<td>941</td>
<td>0 0 0 2</td>
</tr>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>Step 2</strong></td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td><strong>CPU Registers</strong></td>
</tr>
<tr>
<td>300</td>
<td>PC 301</td>
</tr>
<tr>
<td>301</td>
<td>1 9 4 0</td>
</tr>
<tr>
<td>302</td>
<td>0 0 0 3</td>
</tr>
<tr>
<td>940</td>
<td>0 0 0 3</td>
</tr>
<tr>
<td>941</td>
<td>0 0 0 2</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>Step 6</strong></td>
</tr>
<tr>
<td><strong>Memory</strong></td>
<td><strong>CPU Registers</strong></td>
</tr>
<tr>
<td>300</td>
<td>PC 303</td>
</tr>
<tr>
<td>301</td>
<td>1 9 4 0</td>
</tr>
<tr>
<td>302</td>
<td>0 0 0 5</td>
</tr>
<tr>
<td>940</td>
<td>0 0 0 3</td>
</tr>
<tr>
<td>941</td>
<td>0 0 0 5</td>
</tr>
</tbody>
</table>

Figure 1.4 Example of Program Execution (contents of memory and registers in hexadecimal)
Interrupts

- Interrupt the normal sequencing of the processor

- Provided to improve processor utilization
  - most I/O devices are slower than the processor
  - processor must pause to wait for device
  - wasteful use of the processor
### Table 1.1 Classes of Interrupts

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program</td>
<td>Generated by some condition that occurs as a result of an instruction execution, such as arithmetic overflow, division by zero, attempt to execute an illegal machine instruction, and reference outside a user's allowed memory space.</td>
</tr>
<tr>
<td>Timer</td>
<td>Generated by a timer within the processor. This allows the operating system to perform certain functions on a regular basis.</td>
</tr>
<tr>
<td>I/O</td>
<td>Generated by an I/O controller, to signal normal completion of an operation or to signal a variety of error conditions.</td>
</tr>
<tr>
<td>Hardware failure</td>
<td>Generated by a failure, such as power failure or memory parity error.</td>
</tr>
</tbody>
</table>
Flow of Control Without Interrupts

Figure 1.5a

(a) No interrupts

(b) Interrupts; short I/O wait

(c) Interrupts; long I/O wait
Figure 1.5b
Short I/O Wait
Figure 1.5c

Long I/O Wait
Figure 1.6 Transfer of Control via Interrupts
Figure 1.7 Instruction Cycle with Interrupts
Figure 1.8 Program Timing: Short I/O Wait
Figure 1.9  Program Timing: Long I/O Wait

(a) Without interrupts

(b) With interrupts
Device controller or other system hardware issues an interrupt

Processor finishes execution of current instruction

Processor signals acknowledgment of interrupt

Processor pushes PSW and PC onto control stack

Processor loads new PC value based on interrupt

Save remainder of process state information

Process interrupt

Restore process state information

Restore old PSW and PC

Figure 1.10  Simple Interrupt Processing
Figure 1.11 Changes in Memory and Registers for an Interrupt

(a) Interrupt occurs after instruction at location $N$

(b) Return from interrupt
Multiple Interrupts

An interrupt occurs while another interrupt is being processed

- e.g. receiving data from a communications line and printing results at the same time

Two approaches:

- disable interrupts while an interrupt is being processed
- use a priority scheme
Figure 1.12 Transfer of Control with Multiple Interrupts

(a) Sequential interrupt processing

(b) Nested interrupt processing
Figure 1.13 Example Time Sequence of Multiple Interrupts
Memory Hierarchy

- Major constraints in memory
  - amount
  - speed
  - expense

- Memory must be able to keep up with the processor

- Cost of memory must be reasonable in relationship to the other components
Memory Relationships

Faster access time = greater cost per bit

Greater capacity = smaller cost per bit

Greater capacity = slower access speed
Going down the hierarchy:

- decreasing cost per bit
- increasing capacity
- increasing access time
- decreasing frequency of access to the memory by the processor
Figure 1.15  Performance of a Simple Two-Level Memory
Principle of Locality

- Memory references by the processor tend to cluster.
- Data is organized so that the percentage of accesses to each successively lower level is substantially less than that of the level above.
- Can be applied across more than two levels of memory.
Secondary Memory

Also referred to as auxiliary memory

- external
- nonvolatile
- used to store program and data files
Cache Memory

- Invisible to the OS
- Interacts with other memory management hardware
- Processor must access memory at least once per instruction cycle
- Processor execution is limited by memory cycle time
- Exploit the principle of locality with a small, fast memory
Figure 1.16 Cache and Main Memory

(a) Single cache

(b) Three-level cache organization
Figure 1.17 Cache/Main-Memory Structure
Figure 1.18  Cache Read Operation

START

Receive address RA from CPU

Is block containing RA in cache?  

No

Access main memory for block containing RA

Allocate cache slot for main memory block

Load main memory block into cache slot

DONE

Yes

Fetch RA word and deliver to CPU

Allocate cache slot for main memory block

Deliver RA word to CPU

RA - read address
Main categories are:

- cache size
- block size
- mapping function
- replacement algorithm
- write policy
- number of cache levels

Cache Design
Cache Size

small caches have significant impact on performance

Block Size

the unit of data exchanged between cache and main memory
Mapping Function

* Determines which cache location the block will occupy

Two constraints affect design:

- when one block is read in, another may have to be replaced
- the more flexible the mapping function, the more complex is the circuitry required to search the cache
Replacement Algorithm

- Least Recently Used (LRU) Algorithm
  - effective strategy is to replace a block that has been in the cache the longest with no references to it
  - hardware mechanisms are needed to identify the least recently used block
    - chooses which block to replace when a new block is to be loaded into the cache
Write Policy

Dictates when the memory write operation takes place

• can occur every time the block is updated
• can occur when the block is replaced
  • minimizes write operations
  • leaves main memory in an obsolete state
I/O Techniques

* When the processor encounters an instruction relating to I/O, it executes that instruction by issuing a command to the appropriate I/O module

Three techniques are possible for I/O operations:

- Programmed I/O
- Interrupt-Driven I/O
- Direct Memory Access (DMA)
Programmed I/O

- The I/O module performs the requested action then sets the appropriate bits in the I/O status register.
- The processor periodically checks the status of the I/O module until it determines the instruction is complete.
- With programmed I/O the performance level of the entire system is severely degraded.
Processor issues an I/O command to a module and then goes on to do some other useful work.

The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor.

The processor executes the data transfer and then resumes its former processing.

More efficient than Programmed I/O but still requires active intervention of the processor to transfer data between memory and an I/O module.
Interrupt-Driven I/O

Drawbacks

- Transfer rate is limited by the speed with which the processor can test and service a device
- The processor is tied up in managing an I/O transfer
  - a number of instructions must be executed for each I/O transfer
Direct Memory Access (DMA)

* Performed by a separate module on the system bus or incorporated into an I/O module

When the processor wishes to read or write data it issues a command to the DMA module containing:

- whether a read or write is requested
- the address of the I/O device involved
- the starting location in memory to read/write
- the number of words to be read/written
Direct Memory Access

- Transfers the entire block of data directly to and from memory without going through the processor
  - processor is involved only at the beginning and end of the transfer
  - processor executes more slowly during a transfer when processor access to the bus is required

- More efficient than interrupt-driven or programmed I/O
A stand-alone computer system with the following characteristics:
- two or more similar processors of comparable capability
- processors share the same main memory and are interconnected by a bus or other internal connection scheme
- processors share access to I/O devices
- all processors can perform the same functions
- the system is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels
SMP Advantages

Performance
• a system with multiple processors will yield greater performance if work can be done in parallel

Availability
• the failure of a single processor does not halt the machine

Scaling
• vendors can offer a range of products with different price and performance characteristics

Incremental Growth
• an additional processor can be added to enhance performance
Figure 1.19  Symmetric Multiprocessor Organization
Multicore Computer

- Also known as a chip multiprocessor
- Combines two or more processors (cores) on a single piece of silicon (die)
  - each core consists of all of the components of an independent processor
- In addition, multicore chips also include L2 cache and in some cases L3 cache
Figure 1.20  Intel Core i7-990X Block Diagram
Summary

- Basic Elements
- Evolution of the microprocessor
- Instruction execution
- Interrupts
  - Interrupts and the instruction cycle
  - Interrupt processing
  - Multiple interrupts
- The memory hierarchy

- Cache memory
  - Motivation
  - Cache principles
  - Cache design

- Direct memory access

- Multiprocessor and multicore organization
  - Symmetric multiprocessors
  - Multicore computers