Operating System

“the software that controls the hardware”

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
    volatile
    referred to as real memory or primary memory

I/O modules
    secondary memory devices
    communications equipment
    terminals

System bus
    communication among processors, memory, and I/O modules
Processor

Two internal registers
- Memory address register (MAR)
  - Specifies the address for the next read or write
- Memory buffer register (MBR)
  - Contains data written into memory or receives data read from memory
- I/O address register
- I/O buffer register

Top-Level Components

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
User-visible Processor Registers

Enable programmer to minimize main-memory references by optimizing register use.

May be referenced by machine language
Available to all programs - application programs and system programs

Data registers
Address registers
Index
Segment pointer
Stack pointer

Address Registers
Index
Involves adding an index to a base value to get an address
Segment pointer
When memory is divided into segments, memory is referenced by a segment and an offset
Stack pointer
Points to top of stack

Control and Status Processor Registers

Used by processor to control operating of the processor
Used by privileged operating-system routines to control the execution of programs

Program Counter (PC)
Contains the address of an instruction to be fetched

Instruction Register (IR)
Contains the instruction most recently fetched

Program Status Word (PSW)
Condition codes
Interrupt enable/disable
Supervisor/user mode

Condition Codes or Flags
Bits set by the processor hardware as a result of operations

Examples
Positive result
Negative result
Zero
Overflow
Instruction Execution

Two steps
- Processor reads (fetches) instructions from memory
- Processor executes each instruction

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

Instruction Register

Fetched instruction is placed in the instruction register

Categories
- Processor-memory
  - Transfer data between processor and memory
- Processor-I/O
  - Data transferred to or from a peripheral device
- Data processing
  - Arithmetic or logic operation on data
- Control
  - Alter sequence of execution
Characteristics of a Hypothetical Machine

(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 AC to Memory

(d) Partial list of opcodes

Figure 1.3 Characteristics of a Hypothetical Machine

Example of Program Execution

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

Interrupt the normal sequencing of the processor

Most I/O devices are slower than the processor
Processor must pause to wait for device

<table>
<thead>
<tr>
<th>Classes of Interrupts</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Program</strong></td>
</tr>
<tr>
<td><strong>Timer</strong></td>
</tr>
<tr>
<td><strong>I/O</strong></td>
</tr>
<tr>
<td><strong>Hardware failure</strong></td>
</tr>
</tbody>
</table>

Interrupt Handler

Program to service a particular I/O device

Generally part of the operating system

Suspends the normal sequence of execution

Figure 1.6  Transfer of Control via Interrupts
**Program Flow of Control**

1. **No Interrupts**
   - Processor checks for interrupts.
   - If no interrupts, fetch the next instruction for the current program.

2. **Interrupt, short I/O wait**
   - If an interrupt is pending, suspend execution of the current program, and execute the interrupt-handler routine.

**Interrupt Cycle**

- Processor checks for interrupts.
- If no interrupts, fetch the next instruction for the current program.
- If an interrupt is pending, suspend execution of the current program, and execute the interrupt-handler routine.

**Figure 1.7 Instruction Cycle with Interrupts**
Timing Diagram Based on I/O Wait

Figure 1.8 Program Timing: Short I/O Wait

Figure 1.9 Program Timing: Long I/O Wait

Simple Interrupt Processing

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

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

(b) Return from interrupt

---

### Multiprogramming

Processor has more than one program to execute

The sequence the programs are executed depend on their relative priority and whether they are waiting for I/O

After an interrupt handler completes, control may not return to the program that was executing at the time of the interrupt
**Memory Hierarchy**

Going down the hierarchy:
- Decreasing cost per bit
- Increasing capacity
- Increasing access time
- Decreasing frequency of access of the memory by the processor
- Locality of reference

**Memory Classes**

**Secondary Memory**
- Nonvolatile
- Auxiliary memory
- Used to store program and data files

**Disk Cache**
- A portion of main memory used as a buffer to temporarily hold data for the disk
- Disk writes are clustered
- Some data written out may be referenced again. The data are retrieved rapidly from the software cache instead of slowly from disk

**Cache Memory**
- Invisible to operating system
- Increase the speed of memory
- Processor speed is faster than memory speed
- Exploit the principle of locality
**Cache Memory**

Contains a copy of a portion of main memory.
Processor first checks cache.
If not found in cache, the block of memory containing the needed information is moved to the cache and delivered to the processor.

![Diagram of Cache and Main Memory](image)

*Figure 1.16 Cache and Main Memory*

---

**Cache/Main Memory System**

![Diagram of Cache/Main Memory System](image)

*Figure 1.17 Cache/Main Memory System*
Cache Read Operation

START
Receive address RA from CPU

Is block containing RA in cache?
No
Access main memory for block containing RA

Yes
Fetch RA word and deliver to CPU

Load main memory block into cache slot

DONE

RA - read address

Allocate cache slot for main memory block

Done

Figure 1.18 Cache Read Operation

Cache Design

Cache size
Small caches have a significant impact on performance

Block size
The unit of data exchanged between cache and main memory
Larger block size more hits until probability of using newly fetched data becomes less than the probability of reusing data that have to be moved out of cache

Mapping function
Determines which cache location the block will occupy

Replacement algorithm
Determines which block to replace
Least-Recently-Used (LRU) algorithm

Write policy
When the memory write operation takes place
Can occur every time block is updated
Can occur only when block is replaced
Minimizes memory write operations
Leaves main memory in an obsolete state
### Programmed I/O

I/O module performs the action, not the processor

Sets appropriate bits in the I/O status register

No interrupts occur

Processor checks status until operation is complete

### Interrupt-Driven I/O

Processor is interrupted when I/O module ready to exchange data

Processor saves context of program executing and begins executing interrupt-handler

No needless waiting

Consumes a lot of processor time because every word read or written passes through the processor
Direct Memory Access (DMA)

I/O exchanges occur directly with memory

Processor grants I/O module authority to read from or write to memory

Relieves the processor responsibility for the exchange

Direct Memory Access

Transfers a block of data directly to or from memory

An interrupt is sent when the transfer is complete

Processor continues with other work

Diagram:

- Issue Read
- Block command
- I/O module
- Do something else
- Interrupt
- DMA module
- DMA → CPU
- Next instruction
- (e) Direct memory access