ORGANIZATION

Information Technology

Full course data

IX orgnization 1/13/13 IT

IT FUNDAMENTALS

LEC 1

Von Neumann architecture:

1. Central processing unit (CPU):
   - Arithmetic & logical unit (ALU).
   - Control Unit (CU).
2. Main Memory (MM)
3. Input/output Units

---

1. Central Processing Unit (CPU)

1. '**controls**' what the computer does
2. responsible for performing **calculations** and data **processing**.
3. handles the **movement** of data **to** and **from** system memory.

# CPU's come in a variety of speeds which are known as **clock rates**.

# Clock rates are measured in 'Hertz'.

# Generally, the faster the clock rate, the faster the performance of the computer.

two main brands of CPU [AMD …… Intel].

# CPU is connected with memory by Buses

Buses: set of wires connecting various computer units to pass and exchange information between these units.

Can be classified by:

1. Data transfer mode:
2. According to the nature of the data:

1. Data transfer mode:

1. Serial Buses: where **one wire** is needed to transfer the data so that it is transmitted **bit by bit**.
2. Parallel Buses: where a **number of wires** are required to equal the **number of places of the word** to be passed so that these cells **move simultaneously** and in **parallel**.

2. According to the nature of the data:

1. **Data bus**: carries the data that need processing

   set of lines allocated for the exchange and transfer of **information** and data between the CPU and the main memory.

   in **both directions** between CPU & memory.

   Contain control line for write and control line for read data

2. **Address bus**: determines where data should be sent

   set of lines assigned to transfer addresses from the CPU to memory

one-way from the CPU to memory.

3. **Control bus**: determines data processing

   Regulates which direction the write and read information need to go

   set of lines allocated to transfer control signals between the CPU and memory and the transmission of control signals

   in **both directions** between CPU & memory.

Central Processing Unit

Arithmetic & logical unit (ALU).   Control Unit (CU).

An arithmetic-Logic Unit (ALU):

the part of a computer processor (CPU) that carries out arithmetic and logic operations on the operands in computer instruction words.

In some processors, the ALU is divided into two units:

arithmetic unit (AU) & logic unit (LU).

The input consists of an **instruction word** (machine instruction word) that contains an **operation code** (**op code**), one or more **operands**, and sometimes a **format code**.

Operation code:

tells the ALU what operation to perform and the operands are used in the operation.

# ALU includes storage places for input operands,

Some processors contain more than one AU

for example, one for *fixed-point operations* and another for *floating-point operations*.

# ALU directs input and output access to the processor controller, main memory (random access memory or RAM in a personal computer), and input/output devices.

# Inputs and outputs flow along an electronic path that is called a bus.

# Operands that are being added, the accumulated result (stored in an accumulator), and shifted results.

# The flow of bits and the operations performed on them in the subunits of the ALU is controlled by **gated circuits**.

# The gates in these circuits are controlled by a sequence **logic unit** that uses a particular algorithm or sequence for each operation code.

Control Unit (CU)

Before executing instruction, program instructions and data must be placed into memory from an **input** device or a **secondary storage** device

The Control Unit performs the following four steps for each instruction:

1. fetch (gets) the instruction from memory.

2.decode the instruction (decides what it means) and directs that the necessary data be moved from memory to the CU unit.

# These first two steps together are called instruction time, or I-time.

3.execute the arithmetic or logical instruction.

# the ALU is given control and performs the actual operation on the data.

4.store the result of this operation in memory or in a register.

# Steps 3 and 4 together are called execution time, or E-time.

control unit: directs memory to release the result to an output device or a secondary storage device.

# The combination of I-time and E-time is called the machine cycle.

Each **central processing unit** has an **internal clock** that produces **pulses** at a **fixed rate** to synchronize all computer operations.

NOTE↓

A *single machine-cycle instruction* may be made up of a **substantial number of *sub-instructions***, each of which must take at least ***one clock cycle***.

Each type of central processing unit is designed to understand a specific group of instructions called the ***instruction set***.

Therefore, one CPU-such as the one for a **Compaq** personal computer-cannot understand the instruction set from another CPU-say, for a **Macintosh**.

Clock Speed (processor speed)

- is the rate at which a processor can complete a processing cycle.
- typically measured in megahertz or gigahertz.
- One megahertz is equal to one million cycles per second
- One gigahertz equals one billion cycles per second.
- This means a 1.8 GHz has twice the clock speed of a 900 MHz

However, it is important to note that

a 1.8 GHz CPU is not necessarily **twice as fast** as a 900 MHz CPU.

This is because different processors often use different **architectures**.

For example, one processor may require more clock cycles to complete a multiplication instruction than another processor.

If the 1.8 GHz CPU can complete a multiplication instruction in **4 cycles,** while the 900 MHz CPU takes **6 cycles**, the 1.8 GHz processor will perform the operation **more than twice** as fast as the 900 MHz processor.

Conversely, if the 1.8 GHz processor **takes more cycles** to perform the same instruction, it will be less than **2x** as fast as the 900 MHz processor.

Other factors also contribute to the overall performance of a computer. Examples include the number of processors, the bus speed, cache size, speed of the RAM, and HDD or SSD speed.

Therefore, while the processor's clock speed is a significant indicator of how fast a computer is, it is not the only factor that matters.

LEC 2

MAIN MEMORY

Computer memory can be classified in the below given hierarchy:

1) Internal register. 2)Cache

3) RAM (Random Access Memory) 4) Hard disk

5) Magnetic tape

---

1)Internal register: **temporary storage** areas for instructions or data.

- They are **not a part of memory**
- they are special additional storage locations **offer** the advantage of speed.
- Registers work **under** the direction of the **control unit** to accept, hold, and transfer instructions or data and **perform** arithmetic or logical comparisons at **high speed**.
- Computers usually assign **special roles** to **certain registers**:
    - **Accumulator**, which collects the *result* of computations.
    - **Address register**, which keeps track of *where* a given instruction or piece of data is *stored* in memory.
    - Each storage *location* in memory is identified by an *address*.
    - **A storage register**, which *temporarily holds data* taken from or about to be *sent* to memory.
    - **General-purpose register**, is used for several functions.

2)**Cache**: a smaller memory, however, larger than internal register. used by the CPU for data which is being **accessed over and over again**. Instead of pulling it every time from the **main memory**, it is put in cache for fast access.

# Common types of caches include: is further classified to L1, L2 and L3:

a) L1 cache: It is accessed without any delay.

b) L2 cache: It takes more clock cycles to access than L1 cache.

c) L3 cache: It takes more clock cycles to access than L2 cache.

# Browser cache: when you visit a webpage, the browser may cache the HTML, images, and any CSS or JavaScript files referenced by the page. When you browse through other pages on the site that use the same images, CSS, or JavaScript, your browser will not have to re-download the files. Instead, the browser can simply load them from the cache, which is stored on your local hard drive.

# Memory cache: When an application is running, it may cache certain data in the system memory, or RAM.

3) Main memory or RAM (Random Access Memory):

It can be increased provided the operating system can handle it.

# Typical PCs these days use 8 GB of RAM.

# It is accessed slowly as compared to cache.

4) Hard disk: hardware component in a computer.

# Data is kept *permanently* in this memory.

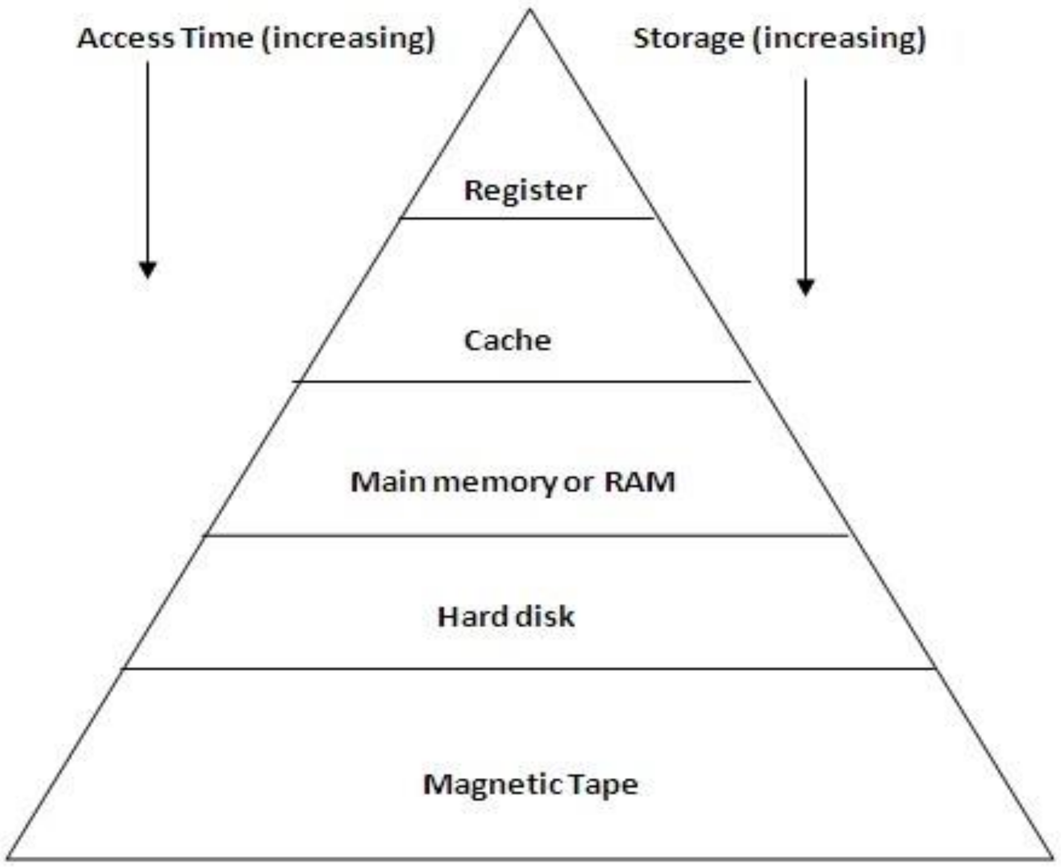
# Memory from hard disk is not *directly accessed* by the CPU, hence it is slower.

# As compared with RAM, hard disk is cheaper *per bit*.

5) Magnetic tape: used for *backing up large data*.

# When the system needs to access a tape, it is first mounted to access the data. When the data is accessed, it is then unmounted.

# The memory access time is slower in magnetic tape and it usually takes few minutes to access a tape.



RAM (Random Access Memory)

used to **temporarily store** information that is **currently in use** by the computer.

include anything from word documents to videos.

# RAM is a **fast** memory.

# Data can be written to and read from RAM very quickly.

# RAM is generally measured in GB (Gigabytes).

# RAM is Volatile Memory.

This means that information stored in RAM is deleted as soon as the computer is turned off.

# How the information is stored in memory?

- Memory is divided into **locations**.
- Each location has its own **address**.
- When information stored in memory is needed, it is accessed directly through its own address.
- If there is **no** special address for each location, then to find the information, it must search **all** locations for finding the required information.
- This search is done in an **organized** manner, i.e.
- searching the first location, then the second, third, and so on.

In contrast,

with other **direct-access data storage** media such as hard disks, CD-RWs, DVD-RWs and the older magnetic tapes and drum memory, the **time** required to read and write data items **varies** significantly depending on their **physical locations** on the **recording medium**, due to mechanical limitations such as **media rotation speeds** and **arm movement**.

# RAM chips are an **integrated circuit** made up of millions of transistors and capacitors that make up the BIT, which stores the number or **code**, and in turn it retains the **value** of the information.

The **more RAM** you have installed in your computer -- the faster it can perform. You can open and use **more programs** at the **same time** without slowing the computer down.

Common types of RAM

- **DRAM (Dynamic RAM):**

  It is dynamic because it is in the process of saving information that needs to be refreshed several times per second.

- **SRAM (Static RAM):**

  Similar to the DRAM. Except that it does not need to be refreshed and this makes it very fast but the SRAM chipset are larger and more expensive and therefore used in Cache Memory.

O **A DIMM (dual in-line memory module)**

is a double SIMM (**single in-line memory module**)?

Like a SIMM,

a DIMM is a module that **contains** one or several random-access memory (RAM) **chips** on a small circuit board with pins that connect it to the computer **motherboard**.

ROM (Read Only Memory)

# ROM is used to **permanently store instructions** that tell the computer how to boot (**startup**). It also loads the operating system (e.g. Windows).

(These instructions are known as the **BIOS** (Basic input/output system) or the boot program.)

# Information stored in ROM is known as **READ ONLY**.

This **means** that the **contents** of ROM **cannot be** altered or added to by the user.

# ROM is **fast** memory. Data stored in ROM can be accessed and read very quickly.

# ROM is **Non-Volatile** memory. This means that stored information is **not lost** when the computer loses power.

Examples:

1-**DVD/CD ROMS** bought in stores containing pre-recorded music and movie files. These are played back at home but **cannot be altered**.

2-**ROM in printers** which is used to store different **font types**.

# ROM is a small chip on your motherboard that looks after your BIOS.

# The code in ROM and Data in Hard disk is always there until you remove it,

# The original purpose of such ROM was to help the CPU find the hard disc or floppy in order to be able to load enough new code to understand the disc structure, and thus **load** an entire Disk Operating System (DOS). This process was called **booting**,

# Common types of ROM

1-PROM (Programmable ROM):

is a piece of memory that can be *programmed only once*? Once the information is written on it, it cannot be erased or changed. (Like: CD)

2-EPROM (Erasable PROM):

is the same as PROM, but information in this memory *can be erased* using ultraviolet light. These **rays are routed to a special** probe **located** on the memory for a certain period of time, which results in the scanning of all the information and thus can reprogram the memory with other information.

3-EEPROM (Electrically Erasable PROM):

This memory is now used in most modern motherboards to save the BIOS. This type of memory can be erased and **re-programmed**. The presence of Flash BIOS in the motherboard specification means that it uses EEPROM.

LEC 3

Machine Level Representation of Data

- An important issue in the construction and maintenance of information systems is the *amount of storage required*.
- This chapter presents basic concepts and calculations pertaining to the most common data types.

Bits, bytes and words

- A **bit** is the smallest unit of memory, and is basically a **switch**. It can be in one of two states, "0" or "1". These states are sometimes referenced as "off and on", or "no and yes"; but these are simply alternate designations for the same concept.
- Given that **each bit** is capable of holding **two possible** values, the number of possible different combinations of values that can be stored in n bits is $\mathbf{2^n}$. For example:
- 1 bit can hold $2 = 2^1$ possible values (0 or 1)
- 2 bits can hold $2 \times 2 = 2^2 = 4$ possible values (00, 01, 10, 11)
- 3 bits can hold $2 \times 2 \times 2 = 2^3 = 8$ possible values (000, 001, 010, 011, 100, 101, 110, or 111)
- 4 bits can hold $2 \times 2 \times 2 \times 2 = 2^4 = 16$ possible values

- 5 bits can hold $2 \times 2 \times 2 \times 2 \times 2 = 2^5 = 32$ possible values
- 6 bits can hold $2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^6 = 64$ possible values
- 7 bits can hold $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^7 = 128$ possible values
- 8 bits can hold $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 = 2^8 = 256$ possible values
- n bits can hold 2^n possible values.

Bits vs. Bytes

- A byte is simply 8 bits of memory or storage. This is the smallest amount of memory that standard computer processors can manipulate in a single operation.
- If you determine the number of bits of memory that are required, and divide by 8, you will get the number of bytes of memory that are required. Similar, to convert from bytes to bits, you must multiply by 8.

Word

- Many standard kinds of data occupy either 1, 2, 4, or 8 bytes, which happen to be the data sizes that today's typical processor chips are designed to manipulate most efficiently.
- A **word** is basically the number of bits a particular computer's CPU can deal with.

1 byte = 8 bits

- o A single character of text (for most character sets). Thus, an MS Access field with datatype Text and field width n consumes n bytes. Example: Text(40) consumes 40 bytes.
- o A whole number from –128 to +127 of from 0 to 255, is a Number/Byte datatype.
- Yes/No fields also consume 1 byte. In principle, you only need a single bit, but one byte is the minimum size for a field.

2 bytes = 16 bits, or two bytes

- A whole number between about –32,000 and +32,000; is a Number/Integer datatype, often also called a “short” integer
- A single character from a large Asian character set

4 bytes = 32 bits

- Can hold a whole number between roughly –2 billion to +2 billion, is a Number/Long Integer datatype
- A “single precision” **floating-point** number. “Floating point” is basically scientific notation, although the computer’s internal representation uses powers of 2 instead of powers of 10.
- The Number/Single datatype, is about of 6 decimal digits of accuracy.

8 bytes = 64 bits

- Can hold a “double precision” floating-point number with the equivalent of about 15 digits of accuracy, is a Number/Double datatype, and is the most common way of storing numbers that can **contain fractions**.
- Really massive whole numbers is essentially the way to store the following datatypes
- Date/Time
- Currency.

Byte-Oriented Memory Organization

- When you buy a *15 GB hard drive*, however, you might well get **15 decimal gigabytes**, so when the drive is *formatted*, your **computer's operating system** might *state its size as* **13.97 binary GB**.
- You haven't lost 1 GB; the **size** was measured using two *different systems*.

Computer Memory & Data Representation

0 Computers store everything as binary digits.

0 So, how can we encode numbers, images, sound, text ??

Alphanumeric Data

- Data represented by letters, digits, and sometimes by special characters and the space character
- Alphanumeric data are represented by assigning a unique binary code to represent each character.
- Each character entered from a keyboard is converted into a binary code.
- Character code sets contain two types of characters:
  - Printable (normal characters)
  - Non-printable. Characters used as control codes: CTRL G (beep)

Alphanumeric Codes

There are 3 main coding methods in use:

- ASCII
- Unicode
- EBCDIC (Extended Binary Coded Decimal Interchange Code).

Non-numeric data

Each character uses 7 bits e.g.

'A' is represented by '1000001' (65 in decimal)

'a' is represented by '1100001' (90 in decimal)

'1' is represented by '0110001' (49 in decimal)

'9' is represented by '0111001' (57 in decimal)

'?' is represented by '0111111' (63 in decimal)

Delete is represented by '1111111' (127 in decimal)

---

2 main character sets in use:

- ASCII (American Standard Code for Information Interchange)
- Unicode

ASCII

- Suitable for most Western languages,
- Uses **8 bits** to represent each character,
- Characters are numbered 0-255, Characters 0-32 are "unprintable", Uppercase and lowercase letters have different codes, Letters and digits are in order.

Unicode

- Can represent a greater number of languages,
- Uses **16 bits** to represent each character, 65535 possible characters,
- First 256 characters = ASCII characters, Java uses Unicode, Audio, video, and pictures.

Programming languages and data types *page 39*

All languages allow programmer to specify data as belonging to particular data types.

- Char
- Boolean
- Integer
- Real
- Strings
- Arrays

Data Type Selection

- Integers– *signed* or *unsigned* as appropriate.
- Floating point for large numbers, fractions, or approximations in measurement.
- Boolean for flags.

Integers

- Integers can be represented in 8-bit, 16-bit, 32-bit or 64-bit.
- An 8-bit *unsigned* integer has a range of 0 to 255, while an 8-bit *signed* integer has a range of -128 to 127 - both representing 256 distinct numbers.

- You, as the programmer, need to decide on the bit-length and representation scheme for your integers, depending on your application's requirements.
- Suppose storing a small quantity from 0 up to 200, you might choose the *8-bit unsigned* integer scheme as there is no negative numbers involved.
- Usually, a fixed size (such as *one byte for numbers* up to *255*, two bytes for numbers up to *65535* or four bytes for numbers up to 4294967295) is used to hold such a number.

Value Type

- The value types directly contain data.
- Some examples are int, char, and float, which stores numbers, alphabets, and floating point numbers, respectively.
- When you *declare* an int type, the system *allocates* memory to store the value.

Type	Represents	Range	Default Value
bool	Boolean value	True or False	False
byte	8-bit unsigned integer	0 to 255	0
char	16-bit Unicode character	U +0000 to U +ffff	'\0'
decimal	128-bit precise decimal values with 28-29 significant digits	(-7.9 x 10^{28} to 7.9 x 10^{28}) / $10^{0 \text{ to } 28}$	0.0M

double	**64-bit double-precision floating point type**	**(+/-)5.0 x 10^{-324} to (+/-)1.7 x 10^{308}**	**0.0D**
float	32-bit single-precision floating point type	-3.4 x 10^{38} to + 3.4 x 10^{38}	0.0F
int	32-bit signed integer type	-2,147,483,648 to 2,147,483,647	0
long	64-bit signed integer type	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	0L
double	64-bit double-precision floating point type	(+/-)5.0 x 10^{-324} to (+/-)1.7 x 10^{308}	0.0D

Long	**64-bit signed integer type**	**-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807**	**0L**
sbyte	8-bit signed integer type	-128 to 127	0
short	16-bit signed integer type	-32,768 to 32,767	0
uint	32-bit unsigned integer type	0 to 4,294,967,295	0
ulong	64-bit unsigned integer type	0 to 18,446,744,073,709,551,615	0

Size of a type or a variable

- Use the *sizeof* method, to get the exact size of a type or a variable on a particular platform.
- *sizeof* (type) yields the storage size of the object or type **in bytes**.

```
using System;
namespace DataTypeApplication
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.WriteLine("Size of int: {0}",                );
         Console.ReadLine();
      }
   }
}
```

Size of a type or a variable

- When the above code is compiled and executed, it produces the following result:
- Size of int: 4

Data compression

- What if you wanted to compress a file?
- This means turning it into a representation that takes less space.

Picture data - unit size

- is represented in "raster" or "bitmap" format – a rectangular array of dots, each with its own color.
- "dots" or "pixels" are identical.
- "dots" is associated with dots-per-inch (dpi), is a measure of resolution for scanners and printers, while "pixels" is associated with the working resolution of a computer monitor.
- The **memory requirement** for a single dot or pixel depends upon the *level of color* or *shade* resolution desired in the picture.
- For *black-and-white* pictures: Line Art (black and white only) 1 bit/pixel
- 16 shade grayscale 4 bits/pixel
- 64 shade grayscale 6 bits/pixel
- 256 shade grayscale 8 bits/pixel
- For *color pictures* 16 color (basic EGA (Enhanced Graphics Adapter) color) 4 bits/pixel

- 256 color (Basic VGA (Video Graphics Adapter) or 8 bit color) 8 bits/pixel
- 16 bit color (65,536 colors)
- **Colors** are represented by **numbers**
- • For black and white: a *number* indicating how bright the dot is.
- • For color, *three number* indicating how bright the dot is (red, blue, and green).

Sound data - unit size

- air pressure variation using a sequence of numbers.
- (for humans vibrations ranging from 20 Hz (cycles per second) to over 20 kHz.
- Most CD-Quality recordings have range of upper 22.05 kHz represented as a 16-bit number.

Video data - unit size

- a series of pictures called frames.
- Video requires a *rapid sequence of pictures* (typically 24 frames per second) to provide *realistic animation.*

LEC 4

Picture data - unit size

is represented in **"raster"** or **"bitmap"** format – a rectangular array of **dots**, each with its own color.

"dots" or "pixels" are **identical**.

"dots" is associated with dots-per-inch (dpi), is a measure of resolution for scanners and printers, while "pixels" is associated with the working resolution of a computer monitor.

The **memory requirement** for a single dot or pixel depends upon the *level of* **color** or shade resolution desired in the picture.

For **black-and-white** pictures: Line Art (black and white only) 1 bit/pixel

- 16 shade grayscale 4 bits/pixel
- 32 shade grayscale 5 bits/pixel
- 64 shade grayscale 6 bits/pixel
- 256 shade grayscale 8 bits/pixel

For **color** pictures 16 color (basic EGA (Enhanced Graphics Adapter) color) 4 bits/pixel

256 color (Basic VGA (Video Graphics Adapter) or 8 bit color) 8 bits/pixel

16 bit color (65,536 colors)

Colors are represented by **numbers**

• For black and white: a **number** indicating how bright the dot is.

• For color, three number indicating how bright the dot is (red, blue, and green).

Sound data - unit size

air pressure variation using a sequence of numbers.

(for humans vibrations ranging from 20 Hz (cycles per second) to over 20 kHz.

Most CD-Quality recordings have range of upper 22.05 kHz represented as a 16-bit number.

Video data - unit size

a series of pictures called **frames**.

Video requires a rapid sequence of pictures (typically **24 frames** per second) to provide realistic animation.

Determining how many "units" make up a file

- Alphanumeric files - number of units.
- Picture files - number of units.
- sound files - number of units.
- Video files - number of units.
- Final calculations, and the effect of **compression** on file size.

Alphanumeric files - number of units

The number of characters **in a file** can be determined from existing data.

As an example, the number of characters in a **book** can be determined by multiplying:

Characters/book = (characters/line) × (lines/page) × (pages/book)

An **80-page book** with **50 lines** per page and **80 characters per line** would have

(80 characters / line) × (50 lines / page) × (80 pages / book) = 320,000 characters

Remember that *spaces* are characters, so half lines, half pages and blank pages need to be included.

Pictures files - number of units

In picture files the number of dots or pixels make up the number of "units".

Picture files are more likely to be specified by their **length and width**.

To determine the number of units in this type of file, you simply multiply the length by the width.

An example would be the **maximum resolution** of the camera.

i.e. Olympus camera can produce picture files with a resolution of 2048 by 1536 pixels.

(2048 pixels wide) × (1536 pixels long) = 3,145,728 pixels total.

An example would be the **maximum resolution** of the camera.

i.e. Olympus camera can produce picture files with a resolution of 2048 by 1536 pixels.

(2048 pixels wide) × (1536 pixels long) = 3,145,728 pixels total.

As an example, a **4-inch long by 6-inch** wide photo, which is scanned at a **resolution of 600 dpi**, will contain:

(4 inch long) × (600 dpi) = 2400 dots long

(6 inch wide) × (600 dpi) = 3600 dots wide

(2400 dots long) × (3600 dots wide) = 8,640,000 dots total.

Sound files - number of units

The number of units in sound files is based upon time.

Samples are normally specified as samples per second, so you must also know the *total length of the recording*.

The number of samples can then be calculated from the relationship:

Total number of samples = (samples rate per second) × (total time in seconds)

Digital sampling requires a minimum of two samples per Hz. Since the upper range of human hearing is near 20,000 Hz, CD-Quality recordings range near 22.05 kHz

The sampling rate for CD audio is standardized at 44,100 samples per second, and 16 bits per sample.

CD audio (containing data in files) is not compressed.

Compression techniques, such as MP3 format, *compromise slightly on sound quality* to obtain a much smaller file size.

Video files – number of units

The number of units in a video file is based upon time.

Samples are normally specified as frames per second (fps), thus the *total time* of the video file must also be known.

The number of frames can then be calculated from the relationship:

total number of frames = (frame rate per second) × (total time in seconds).

Sampling of analog video requires a minimum of 10 fps to provide low quality video, and at least 24 fps for high quality. 30 fps is quite common.

Final calculations, and the effect of compression on file size

As stated initially, file size is the product of "unit" size and *number of units that make up the file*.

In the case of alphanumeric and picture files, this calculation determines the uncompressed file size.

For sound and video files, if the sample or frame rate is provided, the calculation determines the compressed file size.

Compression is a process in which file size is reduced while maintaining all critical file components.

There are many different compression techniques, most of which are optimized using specific "compression ratio".

Compression ratio is described as:

Compression ratio = (Original file size) / (Compressed file size)

Alphanumeric files - final size and compression

For alphanumeric files, the memory requirement calculation is:

Memory requirement = (Memory / character) × (characters / file).

Assuming a 7-bit character set, the memory requirement for a book containing 320,000 characters:

(7 bits / character) × (320,000 characters / file) = 2,240,000 bits.

Memory requirement in bytes:

2,240,000 bits × (1 byte / 8 bits) = 280,000 bytes

Picture files - final size and compression

Memory requirement = (Memory / dot or pixel) × (dots or pixels / file).

Assuming a 16-shade grayscale resolution, the memory requirement for the scanned picture containing 8,640,000 pixels in bytes:

(4 bits / pixel) x (8,640,000 pixels / picture) × (1 byte / 8 bits) = 4,320,000 bytes = 4.32 MB

The above result is for an uncompressed picture file, commonly called a bitmap file.

Sound files - final size and compression

Memory requirement = (Memory / sound sample) × (sound samples / file).

Certain compression techniques minimize storage requirements by saving only the differences from one sample to the next.

Video files - final size and compression

Memory requirement = (Memory / video frame) × (video frame / file).

Memory requirement = (Memory / video frame) × (video frames / sec) × (seconds / file)

Example

Suppose you have a table that stores the USA counties about 3100, for each county, its name (up to 40 characters in 8-bit ASCII), its state (a two-letter code), its population, and its median income (both as 32-bit numbers). How much space would the whole database take in binary-style KB?

Solution

Name 40 Characters $\rightarrow$ 40 bytes

State 2 Characters $\rightarrow$ 2 bytes

Population 32 bit integer $\rightarrow$ 4 bytes (32 bits/8 bits/byte = 4 bytes)

Median Income 32 bit integer $\rightarrow$ 4 bytes

40+ 2+4+4 = 50 bytes

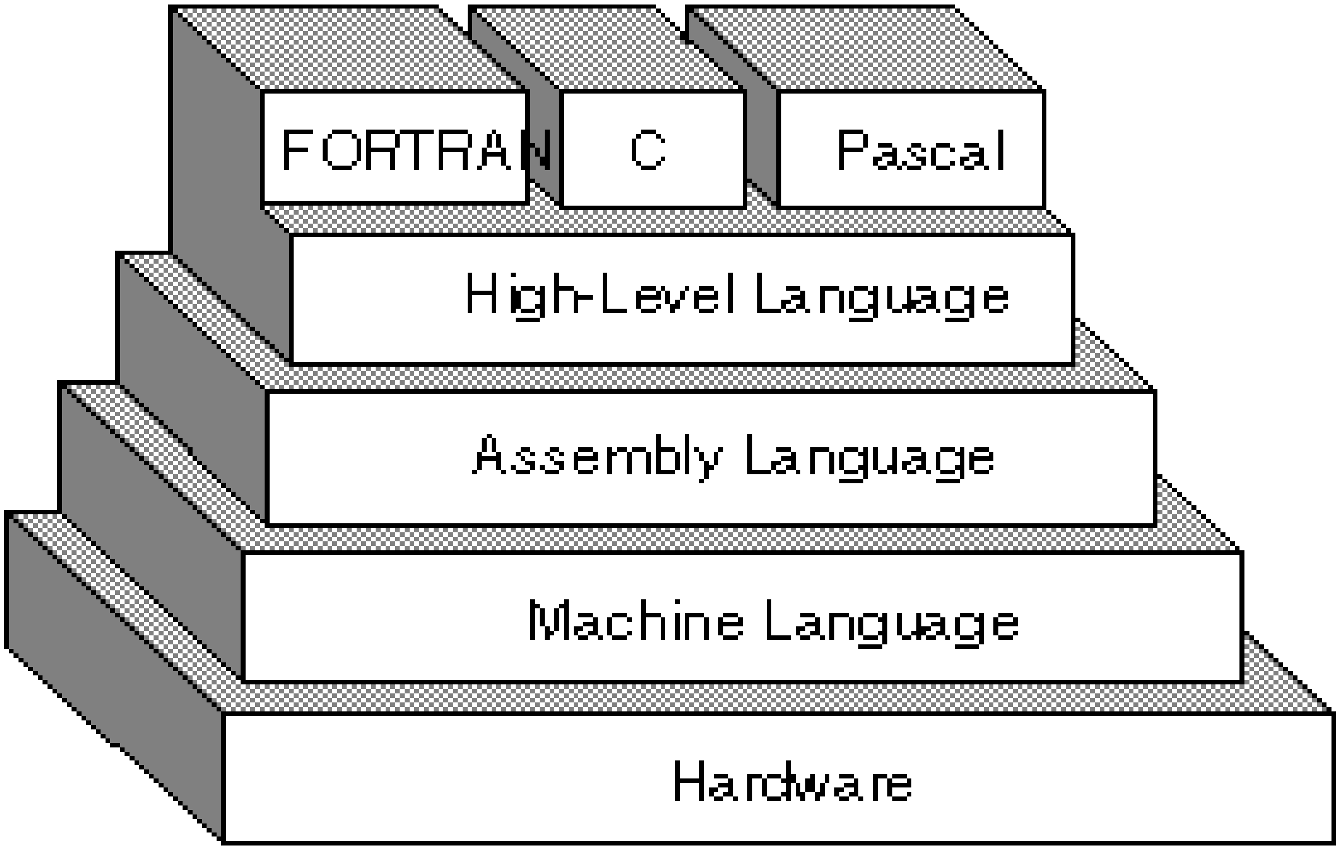
If there are 3100 rows, that means

3100 rows × 50 bytes/row = 155,000 bytes for the whole table.

Finally, 155,000 bytes / 1024 bytes/binary KB ≈ 151 KB.

LEC 5

Assembly Programming

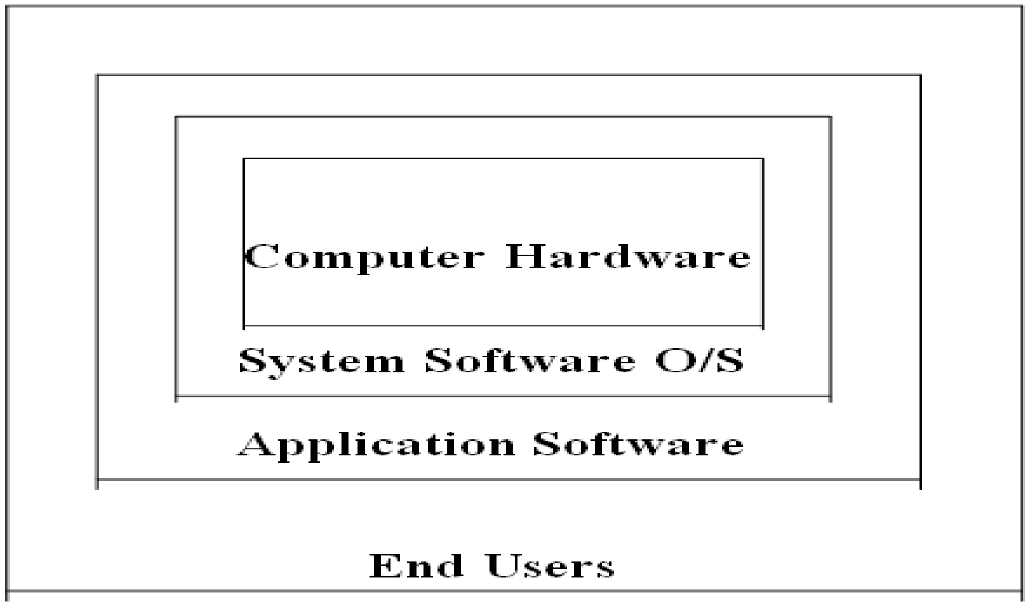


Assembly language:

- Is a **low-level programming language** for specific to a particular computer architecture
- In contrast to most **high-level programming languages**, which are generally **portable across multiple systems**.
- Assembly language is converted **into executable machine code** by a utility program referred to as an assembler like **NASM, MASM**, etc.

What is Assembly Language?

- Each PC has a microprocessor that manages arithmetical, logical, and control activities.
- Each family of processors has its own set of instructions for handling various operations
- such as getting input from keyboard, displaying on screen and performing various jobs.
- These set of instructions are called 'machine language instructions'.
- A processor understands only machine language instructions, which are strings of 1's and 0's.
- However, machine language is too obscure and complex for using in software development.
- So, the low-level assembly language is designed for a specific family of processors that represents various instructions in symbolic code and a more understandable form.



Multilevel Machine

- The most important system software is the operating system,
- O/S is an integrated system of programs that manages the system resources, and provides various support services
- such as executing the application programs of users.

Programming Languages Categories

- Machine Language (First-generation Language)
- Use binary coded instruction.
- Assembly Language (Second-generation Language)
- Use symbols to represent operation codes and storage locations.
- High-Level Languages (Third-generation Languages)
- Use statements that closely resemble human language or standard notation of mathematics.
- Fourth-generation Languages
- Use nonprocedural programming languages.

Advantages of Assembly Language

Makes one aware of :

- How to programs OS interface,
- How data is represented in memory and other external devices;

- How the processor accesses and executes instruction;
- How instructions access and process data;
- How a program accesses external devices.

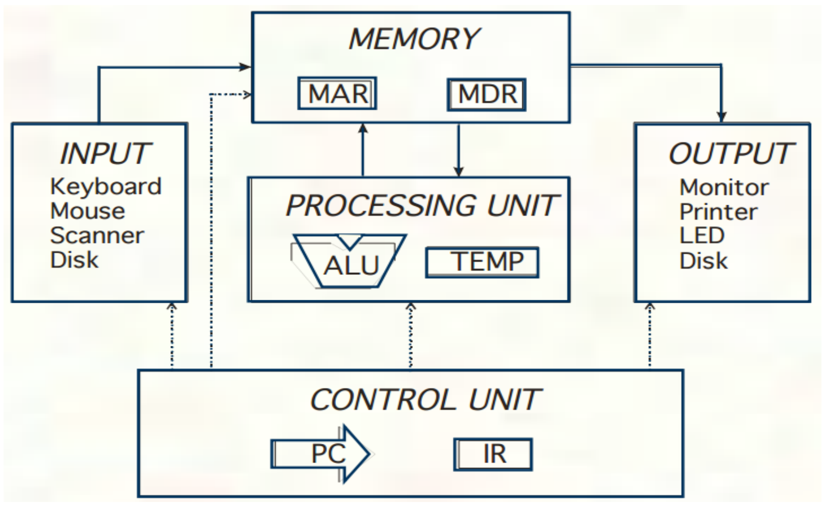
Other advantages:

- Requires less memory and execution time;
- Allows hardware-specific complex jobs in an easier way;
- Suitable for time-critical jobs;
- Assembly uses **mnemonic sequences** instead of numeric operation codes and can use **symbolic labels** instead of manually calculating offsets.

Von Neumann model

The basic structure of "von Neumann machine" (or model):

- A memory, containing instructions and data.
- A processing unit, for performing arithmetic and logical operations.
- A control unit, for interpreting instructions.



Memory: $2^k \times m$ array of stored bits.

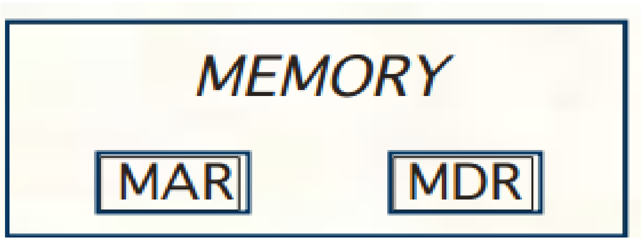
Address: unique (k-bit) identifier of *location*.

Contents: m-bit *value* stored in location.

Basic Operations: **LOAD**, read a value from a memory location and **STORE**, write a value to a memory location.

How processing unit get data to/from memory?

- **MAR**: Memory Address Register
- **MDR**: Memory Data Register



To LOAD a location (A)

1. Write the **address** (A) into the MAR.

2. Send a **"read"** signal to the memory.

3. Read the **data** from MDR.

To STORE a value (X) to a location (A)

1. Write the **data** (X) to the MDR.

2. Write the **address** (A) into the MAR.

3. Send a **"write"** signal to the memory.

Processing Unit

- Functional Units
- ALU = Arithmetic and Logic Unit could have many functional units.
- Special-purpose (multiply, square root, …)
- Performs ADD, AND, NOT

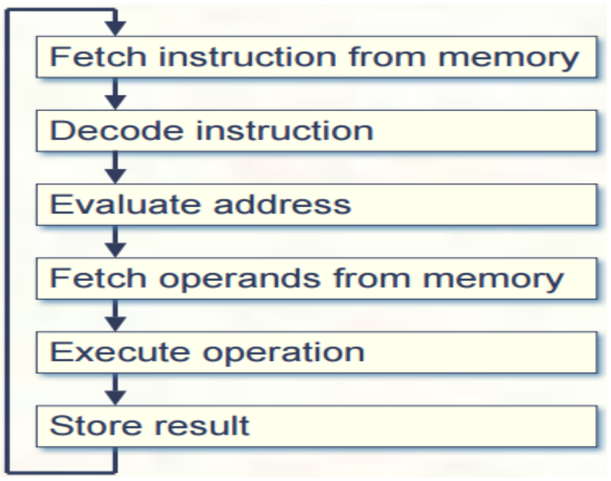
Input and Output

- Input and Output Devices *for getting data* into and out of computer memory.
- Each device has its *own interface*, usually a set of *registers* supports keyboard (input) and monitor (output)
- keyboard: data register (KBDR) and status register (KBSR)
- Monitor (Display): Data Register (DDR) and Status Register (DSR).
- Some devices *provide both* input and output disk, network Program that *controls access to a device* is usually called a driver.

Control Unit

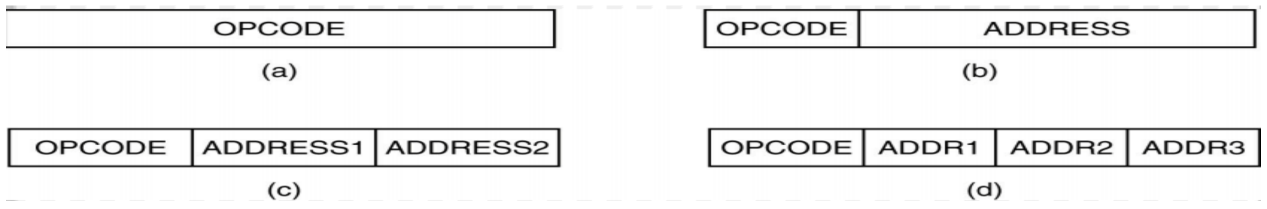
- *Orchestrates execution* of the program.
- Instruction Register (IR) contains the current instruction. Program Counter (PC) contains the address of the next instruction to be executed.
- Control unit: **reads** an instruction from memory the instruction's address is in the PC interprets the instruction, **generating** signals that tell the other components what to do, an **instruction** may take *many machine cycles* to complete.

Instruction Processing



- The instruction is the *fundamental unit of work.*
- Specifies two things:
- **Opcode or "field"**: (operation code) operation to be performed
- **Operands or "constant value"** : data/locations to be used for operation, provide supplemental information required for the operation.
- An instruction is encoded as *a sequence of bits.*

Instruction Format



Four common instruction formats:

(a) Zero-address instruction. (b) One-address instruction

(c) Two-address instruction. (d) Three-address instruction

Instruction Processing

Fetch **Decode** **Execute**

Fetch:

- The first step
- involves retrieving an instruction (which is represented by a number or sequence of numbers) from program memory.
- The instruction's location (address) in program memory is determined by a *program counter* (PC), which stores a number that identifies the address of the *next instruction* to be fetched.
- After an instruction is fetched, the PC is **incremented** by the length of the instruction so that it will *contain the address* of the next instruction in the sequence.

Decode:

- This step, performed by the **instruction decoder**,
- The instruction is **converted into signals** that control other parts of the CPU.
- The way in which the instruction is interpreted is defined by the CPU's **Instruction Set Architecture** (ISA).
- Often, one group of bits (that is, a "field") within the instruction, called the **opcode**, indicates *which operation* is to be performed,
- While the remaining fields usually *provide supplemental information* required for the operation, such as the **operands**.

Execute:

- Depending on the CPU architecture, this may consist of a **single action** or a **sequence of actions**.
- During each action, various parts of the CPU are *electrically connected* so they can *perform* all or part of the desired operation and then the action is *completed*, typically *in response* to a clock pulse.
- Very often the *results* are written to an *internal CPU register* for quick access by subsequent instructions.
- The ALU is configured to perform an *addition* operation so that the *sum of its operand* inputs will appear at its *output*, and the ALU output is *connected to storage* (e.g., a register or memory) that will *receive* the sum.

STORE RESULT:

- Write results to destination. (register or memory)
- Examples: result of ADD is placed in destination register result of memory load is placed in destination register for store instruction,
- data is stored to memory write *address* to **MAR**, *data* to **MDR** assert WRITE signal to memory.

Summaries

FETCH

- Load next instruction (at address stored in PC) from *memory* into *Instruction Register* (IR).
- Copy contents of PC into *MAR*. Send "read" signal to memory.
- Copy contents of *MDR* into IR.
- Then *increment* PC, to the next instruction in sequence. PC becomes PC+1.

DECODE

- identify the opcode.
- The first no. of bits of instruction.
- A decoder asserts a control line corresponding to the desired opcode.
- Depending on opcode, identify other operands from the remaining bits.

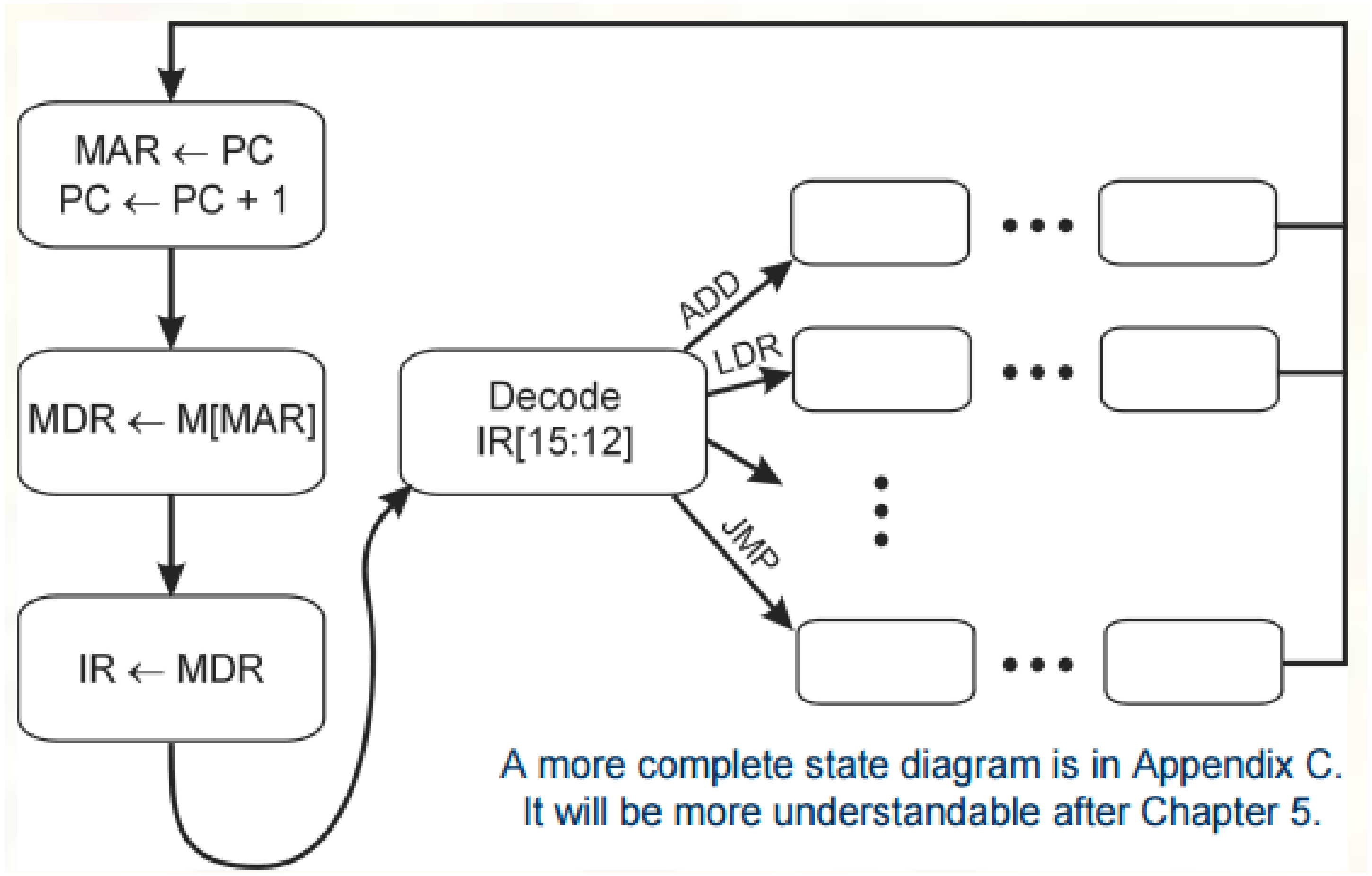
EXECUTE

- When the clock pulse occurs, the sum will be transferred to storage.
- CP, the Clock pulse is used to *synchronize the timing* of hardware components. The speed of the computer's processor, measured in MHz or GHz, refers to its number of clock pulse cycles per second.

STORE

- Write results to destination. (register or memory)
- Examples: result of ADD is placed in destination register result of memory load is placed in destination register for store instruction,
- data is stored to memory write *address* to MAR, *data* to MDR assert WRITE signal to memory.

Control Unit State Diagram


MAR ← PC
PC ← PC + 1
MDR ← M[MAR]
IR ← MDR
Decode
IR[15:12]
ADD
LDR
JMP
A more complete state diagram is in Appendix C.
It will be more understandable after Chapter 5.

LEC 7

Virtual Machines

- A virtual machine (VM) is an operating system (OS) or application environment that is installed on software, which imitates dedicated hardware.
- The end user has the same experience on a virtual machine as they would have on dedicated hardware.
- Virtual machines allow you to run other operating systems within your current operating system –as if they're just another program on your computer.

A System

- A set of principles and concepts and all of the components which obey these principles.

An Operating System

- A system of software which manages the hardware resources of the system to provide a base for its users' programmatic computing needs.
- The operating system acts as an interface or intermediary between the user of a computer and the computer hardware.
- Other definition are below.

Processes

- The OS supports operations on processes: Create a process; Delete a process; Suspend a process; Resume a process; Inter-process communication; Inter-process synchronization; Create/delete a subprocess.

- OS allocate memory space for programs; de-allocate memory space when needed; maintain the mappings from virtual to physical memory; decide how much memory to allocate to each process, and when a process should be removed from memory (policies).

Parts of an operating system page 108

- Kernel
- Device Drivers
- User Interface
- System Utilities
- Shell

Kernel

- handles:
- Loading / Unloading applications from memory
- Scheduling tasks to run on the CPU
- Memory management
- File management
- Data security

Device Drivers

- Every piece of hardware that makes up the computer or connected to it, will have a device driver that allows the operating system to control and communicate with it.

- Hundreds of device drivers pre-installed with the operating system, and the right ones for that particular computer set-up is loaded on boot-up.
- A device driver for Windows is different from the device driver for Linux.

User interface

- Graphical User Interface (GUI)
- directing what you see on the screen (via the device driver) and reacting to your key presses and other inputs.

System Utilities

- the basic facilities that run in the background without user interaction. For example, Print spool services; Cryptographic; password management; File management services.

Shell

- The interface to the operating system.
- The outermost layer of the operating system. Incorporate a programming language to control processes and files,
- allowing communication with the operating system via a control language, letting the user control the peripherals without knowing the characteristics of the hardware used.

EVOLUTION OF OPERATING SYSTEMS

- Serial Processing
- Simple Batch Systems
- Multiprogrammed Batch Systems

- Time-Sharing Systems
- Personal-Computer Systems (PCs)
- Parallel Systems
- Asymmetric multiprocessing model
- Distributed Systems
- Hard real-time systems

Serial Processing

- The programmer interacted directly with computer hardware i.e., no operating system.
- These machines were run with a console consisting of display lights, and a printer.
- If an error occur indicated by lights. Programmers examine the registers and main memory to determine error. If the program is success, then output will appear on the printer.

Simple Batch Systems

- To speed up processing, jobs with similar needs are batched together and run as a group.

Multiprogrammed Batch Systems

- Jobs run sequentially, on a first-come, first-served basis.
- However when several jobs are on a direct-access device, job scheduling is run as multiprogramming to increases CPU utilization.

Time-Sharing Systems

- or multitasking is a logical extension of multiprogramming. That is processors time is shared among multiple users simultaneously.
- The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is in Multiprogrammed batch systems its objective is maximize processor use, whereas in Time-Sharing Systems its objective is minimize response time.

Personal-Computer Systems (PCs)

- is dedicated to a single user.
- Micro computers are considerably smaller and less expensive than mainframe computers.
- For e.g., MS-DOS, Microsoft Windows and Apple Macintosh.

Parallel Systems

- Multiprocessor systems have more than one processor.
- The advantages :
- Throughput (Number of jobs to finish in a time period)
- Save money by sharing peripherals.
- Increase reliability
- Fault-tolerant (Failure of one processor will not halt the system).
- Symmetric multiprocessing model: Windows 7 and Windows Server 2008.
- Each processor runs an identical job (copy) of the operating system, and these copies communicate.

Asymmetric multiprocessing model

- Each processor is assigned a specific task.
- A master processor controls the system.
- Sun's operating system SunOS version 4 is a asymmetric model.

Distributed Systems

- Distributed systems distribute computation among several processors.
- Instead, each processor has its own local memory.

Real-time Systems

- are used when there are rigid time requirements on the operation of a processor or the flow of data and real-time systems can be used as a control device in a dedicated application.
- has well-defined, fixed time constraints.
- E.g., Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, and home-applicance controllers.

Real-time Systems types

O Hard real-time systems.

- Hard real-time systems guarantee that critical tasks complete on time.

O Soft real-time systems.

- Soft real time systems are less restrictive. Critical real-time task gets priority over other tasks and retains the priority until it completes. e.g., Multimedia, virtual reality.

EVOLUTION OF OPERATING SYSTEMS

- Serial Processing
- Simple Batch Systems
- Multiprogrammed Batch Systems
- Time-Sharing Systems
- Personal-Computer Systems (PCs)
- Parallel Systems
- Asymmetric multiprocessing model
- Distributed Systems
- Hard real-time systems

Operating Systems

- the core of any modern technological advancement.
- Operating Systems enable other programmers to do their job easier, as they do all the low-level operations such as interfacing with the hardware.

Operating Systems evolution of computers

- 1956, This early OS was primarily designed to automatically switch to the next job once its current job was completed. It was used on about fourty IBM 704 mainframes.
- 1966, IBM developed a few unsuccessful mainframe Operating Systems until it finally released DOS/360 and its successors, which put IBM in the driver seat for both the hardware and OS industries.

- 1981, MS-DOS: Developed by Microsoft for the IBM PC's. It was the first widely available Operating Systems for home users.
- In 1985, Microsoft released Microsoft Windows, Microsoft Windows allowed users a graphical user interface (GUI), which rapidly spread Microsoft's product.
- 1993, Windows NT.
- 1995, Windows95.
- 1993:Windows NT,
- 1995:Windows95,
- 2000:Windows 2000,Windows ME,
- 2001:Windows XP,
- 2003:Windows Server.
- 2007:Windows Vista,
- 2008:Windows Server,
- 2009:Windows 7,
- 2012:Windows 8

User Interfaces

- The User Interface is the interaction between the User and the Machine, letting the user send commands with the expected results.
- Two forms of the Interface User are the Command Line Interface and the Graphical User Interface.

Command Line Interface (CLI)

- The more primitive User Interface, the user would type in a line of command or a single word followed by pressing the Return key.
- Following exact command, informing the user of its progress.
- For example, to type a command to print a document.
- Type the statement Print, then the name of the Printer and finally the name of the file to be printed:
- print [/d:Printer] [Drive:][Path] FileName [...]

Command Line Interface (CLI)

Pros

- CLIs can run on simple hardware with limited resources.
- Some commands may be simpler to perform in a CLI than in a GUI.

Cons

- Learning the commands may make it not suitable for a new user.
- Some commands may be harder to perform in a CLI than in a GUI.

Graphical User Interface (GUI)

- The GUI is made up of Windows which will display as an array of usually colourful tiny blocks called Pixels, and a group of pixels is a called an image. The maximum number of colours that can be used are called the Graphics.
- This User interface will display a Main window (also known as the Desktop) which then can be accompanied by more windows.

- GUI is the most common of the User interfaces, and the most user friendly, made up of event driven software.
- This type of UI is best for users lacking in computer coding skills, since GUI only needs you to make use of the mouse's events.

Pros

- Simple and intuitive for new users.
- commands may be simpler to perform in a GUI than in a CLI.

Cons

- GUIs require system resources and more complex hardware than CLIs.
- Some commands may be harder to perform in a GUI than in a CLI.

Main responsibilities of O/S

- 1.An operating system is responsible for initializing a computers hardware.
- 2.An operating system is responsible for interaction with user and computer hardware as well.
- 3.An operating system is responsible for time-sharing for different programs.
- 4.An operating system is responsible for multitasking as run more than two programs at a time.
- 5.An operating system is responsible for multiprocessing as an operating system can deal with more than two programs at a time.
- 6.An operating system is responsible for multithreading.
- 7.An operating system is responsible for error messaging between user and computer hardware.

Classification of Operating systems

- Multi-user: Allows two or more users to run programs at the same time.
- Multiprocessing : Supports running a program on more than one CPU.
- Multitasking : Allows more than one program to run concurrently.
- Multithreading : Allows different parts of a single program to run concurrently.
- Real time: Responds to input instantly. General-purpose operating systems, such as DOS and UNIX, are not real-time.

Network operating system

- includes special functions for connecting computers and devices into a local-area network (LAN).
- Some operating systems, such as UNIX and the Mac OS, have networking functions built in.
- a mobile OS, is an operating system that is specifically designed to run on mobile devices such as mobile phones, smartphones, tablet computers and other handheld devices.
- The mobile operating system is the software platform on top of which other programs, called application programs, can run on mobile devices.