Segment 1B

Memory Organization of Microcomputer system

2.1 Purpose of Memory:

(i) Memory unit is the integral part of any microcomputer system and its primary purpose is to hold program and data

(ii) The major design goal of memory unit is to allow it to operate at a speed close to that of the processor.

(iii) The cost factor inhibits the design of entire memory unit with single technology that guarantees high speed.

(iv) In order to seek a trade-off between the cost and operating speed, a memory system is usually designed with different technologies such as solid state, magnetic and optical.

2.2 Types of Memory:

In a broad sense, microcomputer memory can be divided into three groups:

(a) Processor Memory

(b) Primary (or main) memory

- Volatile memory
  - RAM - random access memory
    - Static RAM
    - Dynamic RAM

- Non-volatile memory
  - ROM - read only memory
  - EPROM
  - EEPROM
  - FLASH

(c) Secondary memory

- Hard disks, CD, floppy disks, tape

*Processor Memory* refers to a set of CPU registers. These registers are useful to hold temporary results when a computation is in progress. Also, there is no speed disparity between the registers and the microprocessor because they are fabricated using the same technology. The main disadvantage is the cost involved which forces the architect to include very few registers (usually 8 to 16 only) in the microprocessor.

The primary memory or, *main memory* is the memory that the CPU can access directly. Examples of main memory include RAM, ROM, etc.

The *secondary memory* cannot be addressed directly (cannot access specified memory location) by the CPU. Examples of secondary include floppy disk, hard disk, CD, etc. The information in the secondary memory must be copied to the main memory so that CPU can access it. Secondary memory is much cheaper than primary memory.
2.2.1 Types of Primary Memory:

There are two types of primary memories: volatile and nonvolatile. Volatile memory is the type of memory that will lose data when the power supply to the memory is gone. Random access memory (RAM) is one type of volatile memory. Nonvolatile memory keeps the data in the memory even it is not powered up. Nonvolatile memory includes read only memory (ROM).

Read Only Memory (ROM) is used to permanently save the program being executed. The size of program that can be written depends on the size of this memory. There are several types of ROM.

(i) **Masked ROM (MROM)** is a kind of ROM the content of which is programmed by the manufacturer. The term ‘masked’ comes from the manufacturing process, where regions of the chip are masked off before the process of photolithography. In case of a large-scale production, the price is very low.

(ii) **One time programmable ROM (OTP ROM)** enables you to download a program into it, but, as its name states, one time only. If an error is detected after downloading, the only thing you can do is to download the correct program to another chip.

(iii) **UV Erasable Programmable ROM (UV EPROM)** Both the manufacturing process and characteristics of this memory are completely identical to OTP ROM. It enables data to be erased under strong ultraviolet light. After a few minutes it is possible to download a new program into it.

(iv) **Flash Memory** This type of memory was invented in the 80s in the laboratories of INTEL and was represented as the successor to the UV EPROM. The content of this memory can be written and cleared practically an unlimited number of times. Flash memory can only be erased block (many bytes) by block.

(v) **Electrically Erasable programmable ROM (EPROM)** the contents of EEPROM may be changed during operation (similar to RAM), but remains permanently saved even after the loss of power (similar to ROM). EEPROM can be programmed and erased byte by byte.

Random Access Memory (RAM): Information stored in random access memories will be lost if the power is turned off. This property is known as volatility and hence, **RAMs** are usually called volatile memories. RAM can be made of magnetic core or by semiconductor. Semiconductor RAM’s are two types:

(i) Static RAM and
(ii) Dynamic RAM.

Static RAM: In a semiconductor memory constructed using bipolar transistors; the information is stored in the form of voltage levels in flip-flops. These voltage levels do not usually get drifted away. Such memories are called **static RAMs** because stored information remains constant for some period of time.

Since the static RAM maintains information in active circuits, power is required even when the chip is inactive or standby mode. Hence, static RAMs require large power supplies. Also each static RAM cell is about four times larger in area than an equivalent dynamic cell.

Dynamic RAM: On the other hand, semiconductor memories designed using MOS transistors, the information is held in the form of electrical charges in capacitors. Here the stored charge has the
tendency of get leaked away. These memories are referred to as *dynamic RAMs*. In order to prevent any information loss, dynamic RAMS have to be refreshed at regular intervals. Refreshing means boosting the signal level and writing it back. This activity is performed using a hardware unit called “refresh logic”.

**Differences between static and dynamic RAMs:**

<table>
<thead>
<tr>
<th>Static RAM</th>
<th>Dynamic RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>This semiconductor memory is constructed using bipolar transistors</td>
<td>This semiconductor memory is constructed using MOS transistors</td>
</tr>
<tr>
<td>Information is stored in the form of voltage levels in flip-flops</td>
<td>Information is stored in the form of electrical charges in capacitors</td>
</tr>
<tr>
<td>These voltage levels do not get drifted away</td>
<td>Has tendency of leakage</td>
</tr>
<tr>
<td>No refresh logic is needed</td>
<td>Refresh logic is necessary since leakage of electrical charges</td>
</tr>
<tr>
<td>Power is required even when the chip is in standby mode.</td>
<td>Refresh login is inbuilt, so draws less power comparatively.</td>
</tr>
<tr>
<td>Four time larger in size compared to an equivalent dynamic cell</td>
<td>Four times as many bits as a static RAM chip.</td>
</tr>
</tbody>
</table>

2.3 Main Memory Array Design:

2.3.1 Addressing Memory:

Using 3-bit we can address $2^3 = 8$ distinct memory location. So, n number of bits can address $2^n$ memory locations.

1K Byte = 1024 Byte.

$2^{10} = 1024$

So, for addressing 1KB of memory we need at least 10 bits. And the addresses will be-

0 to 1023 (decimal) or, 000H to 3FF (HEX) or, 0000 0000 0000 to 0011 1111 1111 B (Binary)

![1 Kbyte Memory Diagram](image)

Similarly, for addressing 64 KB of memory we need at least 16 bits.
How many bits are necessary to address 1MB memory?

1MB = 1024 Kbyte = 1,048,576 Byte = $2^{20}$ Byte

So, we need 20 bits to address 1 MByte memory. Addresses will be-
0 to 1,048,575 (Decimal)
00000H to FFFFFH (Hex)
0000 0000 0000 0000 0000 to 1111 1111 1111 1111 1111 B (Binary)

N.B. 8086 has 20-bit address bus. So, it can addresses 1 MB memory.

### 2.3.2 Main Memory Array Design:

In many applications, a memory of large size capacity is often realized by interconnecting several small size memory blocks. There are two kinds of techniques used for designing the main memory in such cases. They are:

1. **Linear Decoding.**
2. **Fully Decoding.**

To realize the interconnecting technique first we should know about a small memory block. Let us consider the block diagram of a typical RAM IC.
The capacity of this chip is 1Kbytes.
They are organized in the form of 1024 words with 8 bits/word.
Each word has a unique address and is specified on 10-bit address lines A9 – A0.
The inputs and outputs are routed through the 8-bit bidirectional data bus (D7–D0). The operation of this chip is governed by two control inputs: \( \overline{WE} \) (Write Enable) and \( \overline{CS} \) (Chip Select).

The following truth table describes the operation of this chip:

<table>
<thead>
<tr>
<th>( \overline{CS} )</th>
<th>( \overline{WE} )</th>
<th>MODE</th>
<th>Status (D7 – D0)</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>X</td>
<td>Not selected</td>
<td>High Impedance</td>
<td>Standby</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>Write</td>
<td>Input Bus</td>
<td>Active</td>
</tr>
<tr>
<td>L</td>
<td>H</td>
<td>Read</td>
<td>Output Bus</td>
<td>Active</td>
</tr>
</tbody>
</table>

When \( \overline{CS} \) is high, chip is not selected at all, hence D7 to D0 are driven to high impedance state.
When \( \overline{CS} = 0 \) and \( \overline{WE} = 0 \), data on lines D7 – D0 are written into the word addressed by A0 through A9.
When \( \overline{CS} = 0 \) and \( \overline{WE} = 1 \), the contents of memory word whose address is on A9 – A0 will appear on lines D7 – D0.

Now let us discuss linear decoding and fully decoding:

**Linear Decoding**

This technique uses the unused address lines of the microprocessor as chip selects for the memory chip.
Figure: Linear Decoding
A simple way to connect an 8-bit microprocessor to a 6K RAM system using linear decoding is shown in figure, where-

- Address lines A9 through A0 of the microprocessor used as common input to address lines of all memory chips.
- The data lines of microprocessor are connected to data lines of all memory chips.
- The remaining address lines (A10-A15) are used to select one of the chips (\( \overline{CS} \)) at a time. For example, 000001 selects chip 1, 100000 selects chip 6 etc.
- \( R/\overline{W} \) from microprocessor is connected to \( \overline{WE} \) of all RAM Chips. This line selects whether data is writing to memory or reading from memory.

**Advantage:**

Linear decoding does not need any decoding circuit (Hardware).

**Disadvantages:**

a) Although there is an address bus of 16-bits wide, we could connect only 6Kb of RAM. This idea clearly wasted address space.

b) Address map is not contiguous. It is sparsely distributed.

c) Conflicts occur if two of the select lines (suppose A11 and A10) become active at the same time.

d) If all unused address lines are not used as chip selectors, then these unused lines become don’t cares. This results in foldback, meaning a memory location will have its image in memory map. For example, if A15 is don’t care, then address 0000H is same as address 8000H. It wastes memory space.

**Fully Decoding**

The problems of bus conflict and sparse address distribution are eliminated by the use of fully decoding address technique.

Consider an example where we interface 4Kb of RAM to an 8 – bit microprocessor. The RAM chips are available in the form of 1K X 8.

First we have to write memory map to identify the address lines to be given to decoder logic:

<table>
<thead>
<tr>
<th>A15</th>
<th>A14</th>
<th>A13</th>
<th>A12</th>
<th>A11</th>
<th>A10</th>
<th>A9</th>
<th>A8</th>
<th>A7</th>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
<th>A0</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>000000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>03FF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>040000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>07FF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>080000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0BFF</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0C0000</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0FFFFF</td>
</tr>
</tbody>
</table>
If we observe A10 and A11, 2-to-4 decoder would be an obvious choice for CS signals. We can write the truth table as follows:

<table>
<thead>
<tr>
<th>A11</th>
<th>A10</th>
<th>Device selected</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>RAM chip1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>RAM chip2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>RAM chip3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>RAM chip4</td>
</tr>
</tbody>
</table>

Figure: Fully Decoding
The above hardware makes sure that the memory system enables only when the lines A15-A10 will be high. If any line becomes low memory system disable and by this way fold bask is removed. Above that, address space is not wasted since the unused lines (A15-A10) can be used in future by making use of higher decoder.

2.3.3 The 3-to-8 Line Decoder (74LS138)

One of the most common integrated circuit decoders found in many microprocessor-based systems is the 74LS138 3-to-8 line decoder. Following figure shows pin diagram and truth table of 74LS138 decoder.

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Enable</th>
<th>Select</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>G2A</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>G2B</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>G1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The truth table shows that only one of the eight outputs ever goes low at any time. For any of the decoders outputs to go low, the three enable inputs (G2A, G2B, and G1) must all be active. To be active, the G2A and G2B inputs must both be low (logic 0) and G1 must be high (logic 1). Once the 74LS138 is enabled, the address inputs (C, B and A) select which output pin goes low. Imagine EPROM CE inputs connected to the eight outputs of the decoder. Figure below shows a circuit that uses eight 2764 EPROMs for a 64KX8 section of memory in an 8086 or 8088 microprocessor based system. The addressed selected in this circuit are F0000H-FFFFFH.
Problem: Interface 4KB memory to 8085 microprocessor with starting address A000H using 74LS138 decoder.

Solution:
A0-A11 address lines are directly connected to address bus of memory chip. A12-A15 are used for generating chip select signal for memory chip.
A15 line is use for enabling 74x138 decoder chip. A12, A13, A14 lines are connected to 74LS138 chip as inputs. When these lines are 010 output should be ‘0’. This is provided at O2 pin of 74LS138 chip.
The range of address is found as follows-

<table>
<thead>
<tr>
<th>A15 A14 A13 A12</th>
<th>A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>1 0 1 0</td>
<td>1 1 1 1 1 1 1 1 1 1 1 1</td>
</tr>
</tbody>
</table>

**Problem:** Find the range of address for RAM1 and RAM2 in the following figure.