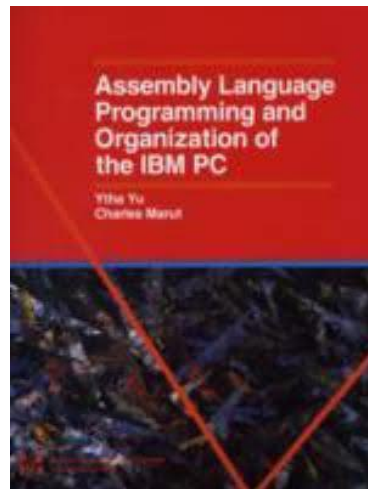
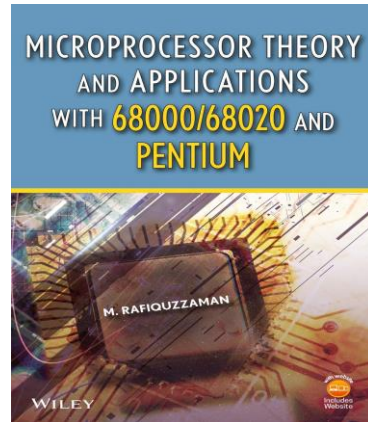
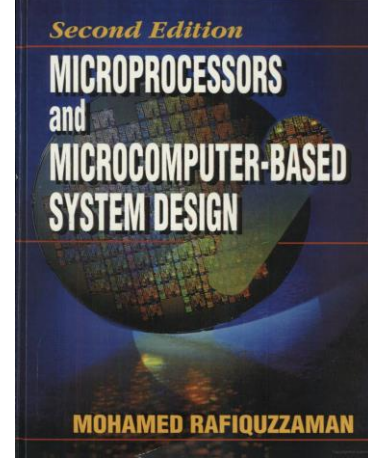


# Microprocessor and Assembly language

Presented by  
Dr. Md. Abir Hossain  
Dept. of ICT  
MBSTU

# Reference Books

1. Microprocessors and Microcomputer-based System Design – By Mohamed Rofiquzzaman
2. Microprocessor theory and applications with 68000/68020 and pentium – By Mohamed Rofiquzzaman
3. Assembly Language Programming and Organization of the IBM PC- by Ytha Yu and Charles Marut



# Microprocessor and Assembly

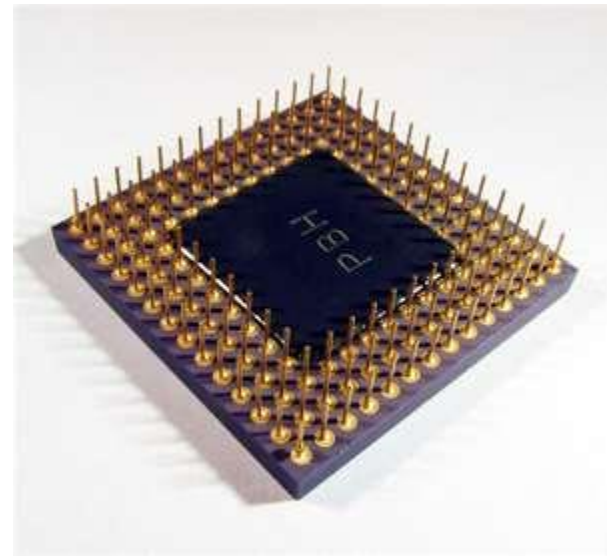
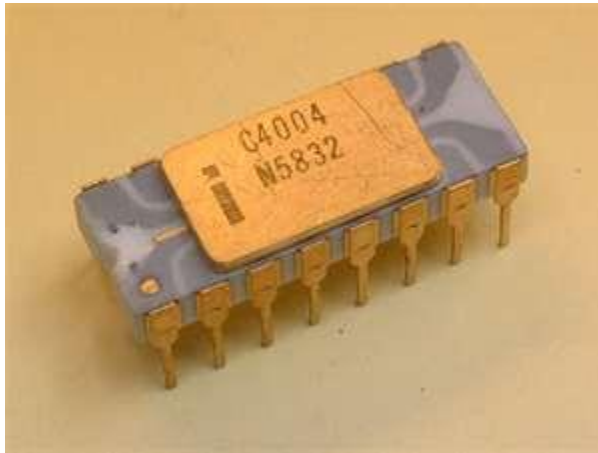


© www.cpu-world.com

# Microprocessor

- A **microprocessor** (sometimes abbreviated  $\mu\text{P}$ ) is a digital electronic component with transistors on a single **semiconductor integrated circuit (IC)**.
- A **Central processing unit (CPU)** in a **computer system** or handheld device consists of one or more microprocessors.
- A Microprocessor is essentially a set of switches. A massive set of electronic switches is superimposed onto a very small piece of silicon.
- Through the use of binary language, which consists of only two states; one and zero (on and off), these can be used to store information and perform operations on it.

- A bit refers to one binary digit; a zero or one. In computer memory and processing this refers to the state of one switch.
- The transistors are arranged into groups in order to represent complex numbers and instructions



Intel i9 processor has around 4.2 billion transistors.

# History of Microprocessor

## First Generation $\mu$ P

- **INTEL 4004** ( 1971) – First Microprocessor
- 4-bit microprocessor
- 4 KB main memory
- 45 instructions
- PMOS technology
- Low cost
- Slow speed
- was first programmable device which was used in calculators

# History of Microprocessor

## First Generation $\mu$ P

- **INTEL 8008** (1972)
- 8-bit version of 4004
- 16 KB main memory
- 48 instructions
- PMOS technology
- Slow speed
- Low cost
- Low output currents
- Not compatible with TTL (Transistor Transistor logic)

# History of Microprocessor

## First Generation $\mu$ P

- Intel 8080 (1973)
- 8-bit microprocessor
- 64 KB main memory
- 2 microseconds clock cycle time
- 500,000 instructions/sec
- 10X faster than 8008
- NMOS technology
- Drawback was that it needed three power supplies.
- Small computers (Microcomputers) were designed in mid 1970's Using 8080 as CPU.



# History of Microprocessor

## Second Generation $\mu$ P

- Motorola 6800, 6809, Intel 8085 and Zilog Z80 (after 1973 ~ before 1978)
- 8-bit microprocessor
- NMOS technology
- Faster speed than first generation  $\mu$ P
- High density Transistor deployment.
- TTL compatible

# History of Microprocessor

## Third Generation $\mu$ P

- **INTEL 8086/8088**
- Year of introduction 1978 for 8086 and 1979 for 8088
- 16-bit microprocessors
- Data bus width of 8086 is 16 bit and 8 bit for 8088
- 1 MB main memory
- 400 nanoseconds clock cycle time
- 6 byte instruction cache for 8086 and 4 byte for 8088
- Other improvements included more registers and additional instructions
- In 1981 IBM decided to use 8088 in its personal computer

# History of Microprocessor

## Third Generation $\mu$ P

- **INTEL 80186 (1982)**
- 16-bit microprocessor-upgraded version of 8086
- 1 MB main memory
- Contained special hardware like programmable counters,
- interrupt controller etc.
- Never used in the PC
- But was ideal for systems that required a minimum of hardware .

# History of Microprocessor

## Third Generation $\mu$ P

- **INTEL 80286 (1983)**
- 16-bit high performance microprocessor with memory management & protection
- 16 MB main memory
- Few additional instructions (15) to handle extra operation
- Instruction execution time is as little as 250 ns
- Concentrates on the features needed to implement MULTITASKING

# History of Microprocessor

## Fourth Generation $\mu$ P

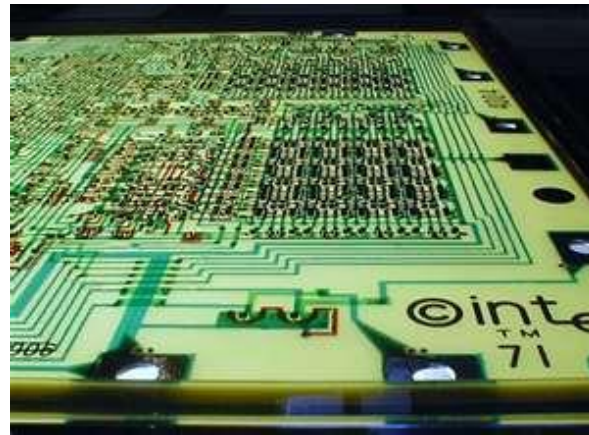
- **INTEL 80386/80486/Pentium Motorola MC68020/68030/68040/PowerPC (1980 ~ later)**
- 32-bit high performance microprocessor
- Fabricated using Low power consumption of H MOS called HCMOS technology
- Operating speed comparing to supercomputers (VAX11/750, VAX11/780)
- Both Intel(80960) and Motorola (MC88100/ PowerPC) introduced RISC (Reduced instruction set computer)  $\mu$ P
- Pipelining enhance the throughput performance of instruction execution.

# Current status of Microprocessor

- Most of today's computers are turning to 64 bit designs to handle dealing with very large amounts of data.
- This is needed especially as demand for 3D Graphics and fast video has risen. E.g. AMD Athlon, Pentium i5/i7 processors integrated with GPU (Graphics processing unit).



www.shutterstock.com - 51164518



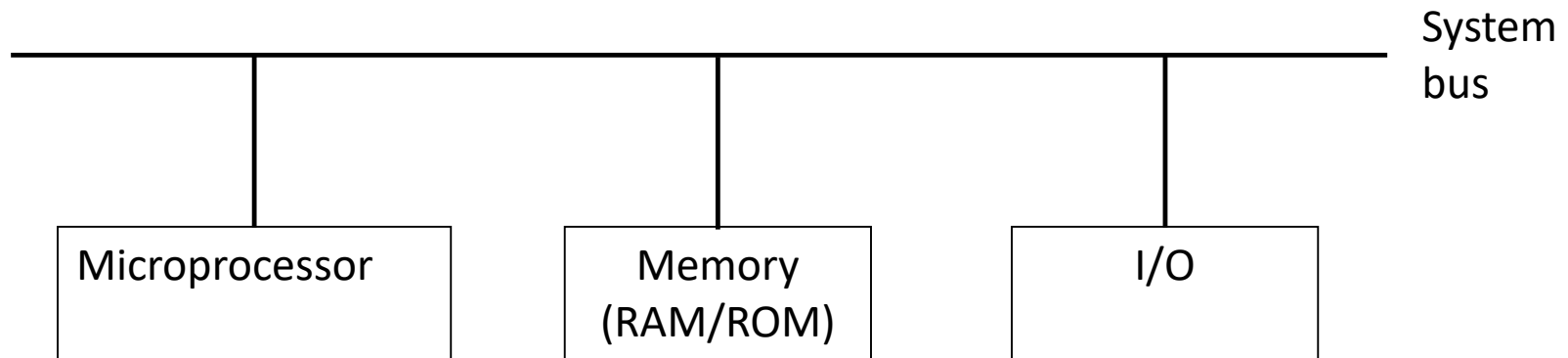
- Microprocessors are classified into different types on the basis of the bit of operation. Based on bit of operation at a time, the following are the types of microprocessors:
  - ==> 4 bit. e.g. Intel 4004
  - ==> 8 bit. e.g. Intel 8085, 8088, Zilog Z80, Z180
  - ==> 16 bit. e.g. Intel 8086, 80186, 80286, 80386,
  - ==> 32 bit. e.g. Intel Pentium, Celeron, AMD Sempron
  - ==> 64 bit. e.g. AMD Athlon, intel core i5/i7.
- Based on the instruction set microprocessors are classified into:
  - RISC — Reduced Instruction Set Computing. These types of processors are commonly used in ovens, air conditioners, etc.
  - CISC — Complex Instruction Set Computing. The types of processors are used in desktops, laptops and servers.

# Microprocessor Data Types

- Unsigned(positive) and Signed Binary numbers(Both positive and negative binary number)
- Binary Coded Decimal (BCD)
  - Four digits ( $0000_2 \sim 1001_2$ )
- ASCII (American Standard Code for Information Interchange)
  - Represents alphanumeric and symbols in  $\mu P$ 's memory
- Floating point number
  - Contain three component
    - Sign (Positive or negative)
    - Exponent (power)
    - Mantissa
    - $-2.5 \times 10^{-2}$

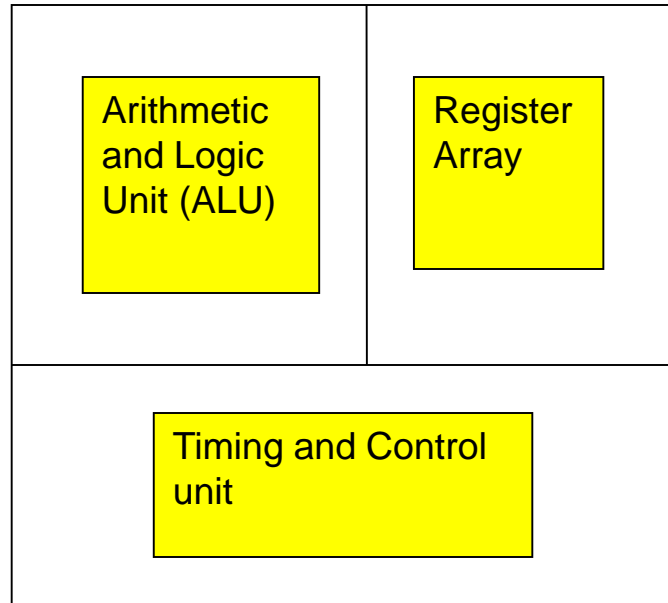


# Microprocessor connected with Bus



- The system bus contains three bus
  - Address bus
  - Data bus
  - Control bus
- The buses connect the  $\mu$ P to each memory and I/O to transfer information.
- In address bus, information transfer in one direction ( $\mu$ P  $\rightarrow$  memory and I/O)
- Data bus is a bidirectional bus
- Control bus dealing instruction both unidirectional and bidirectional

# Inside the Microprocessor



**ALU** – Performs all arithmetic and logical operations

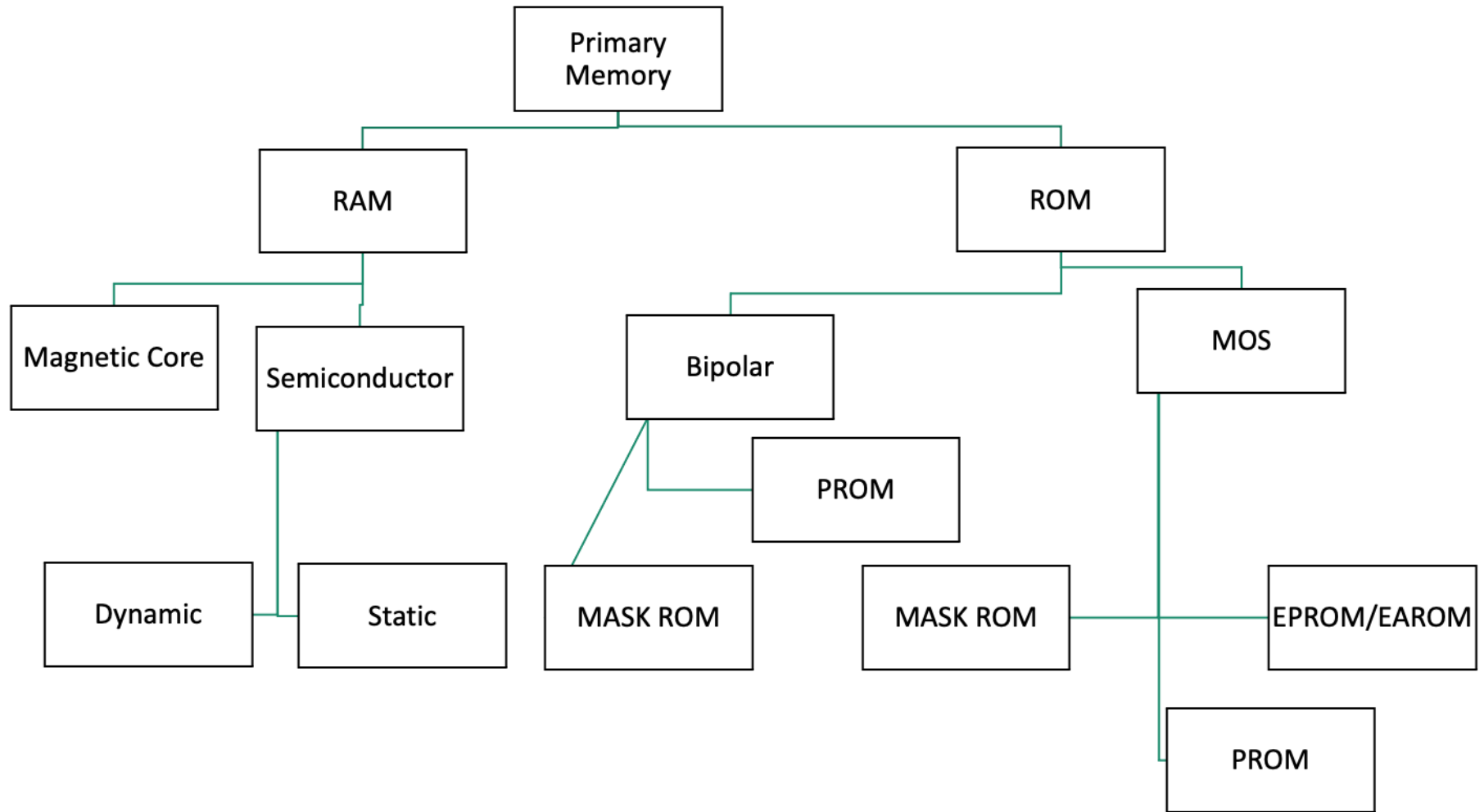
**Register array** – Holds the data temporarily for processing

**Control Unit** – It supervises/monitors all the operations carried out in the computer

# $\mu$ P`s Memory

- Logically divided into three groups
  - Processor Memory
    - $\mu$ P`s registers. It hold temporary results of an operation.
  - Primary/Main memory
    - A storage area where all programs are executed.
  - Secondary Memory
    - Storage device whre data stored after the operation.

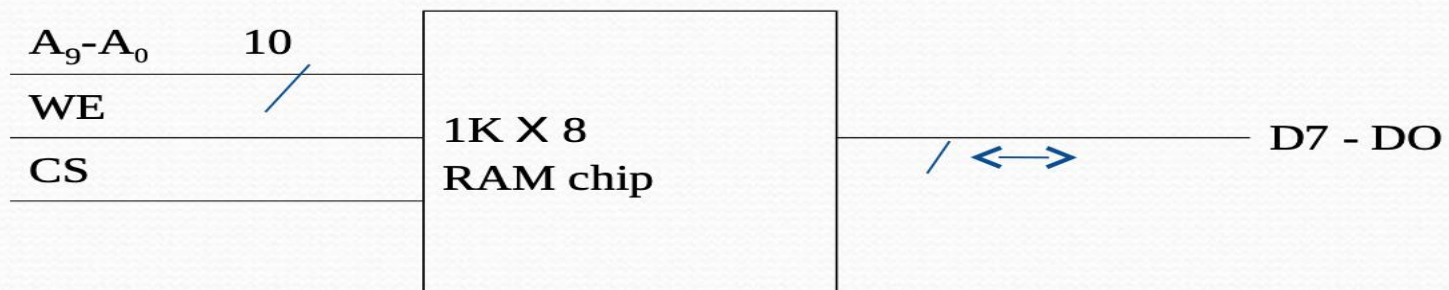
# Primary memory subcategories



# Main Memory Array Design

- In many application, a memory of large capacity is often realized by interconnecting several small-size memory blocks.
- 3 types of techniques used for designing the main memory.
  - Linear decoding
  - Full decoding /partial decoding &
  - Memory decoding using PALs.

Consider the block diagram of typical static RAM Chip as shown below. The capacity of this chip 8192 bits are organized as 1024 words with 8 bits / word. Each word has a unique address and this is specified on 10 bit address lines A9 - A0 .



**Fig: Typical Static RAM chip**

# Main Memory Array Design

- The inputs and outputs are routed through the 8 bit bidirectional data lines D7 through D0.
- The operation of this chip is governed by the two control units: WE and CS
- The truth table describes the operation of this chip.

Table: Truth table for 1K X 8 static RAM

CS	<u>WE</u>	MODE	Status of D7-D0	Power
L	X	Not selected	High impedance	Standby
H	L	Write	Acts as input bus	Active
H	H	Read	Acts as output bus	Active

H- High, L- Low, X- don't care

# Main Memory Array Design

## Linear Decoding

- uses the unused address lines of the  $\mu$ P as chip selects for the memory chips.
- This method is used for small systems.
- A simple way to connect an 8 bit microprocessor to a 6k RAM system using linear decoding is shown as follows.
- In this approach, the address lines A9 though A0 of the microprocessor are used as a common input to each 1K x 8 RAM chip.
- The remaining 6 high – order lines are use to select one of the 6 RAM chips.
- For Example,  
If A15 A14 A13 A12 A11 A10 = 000010  
Then chip 2(RAM 2) is selected.

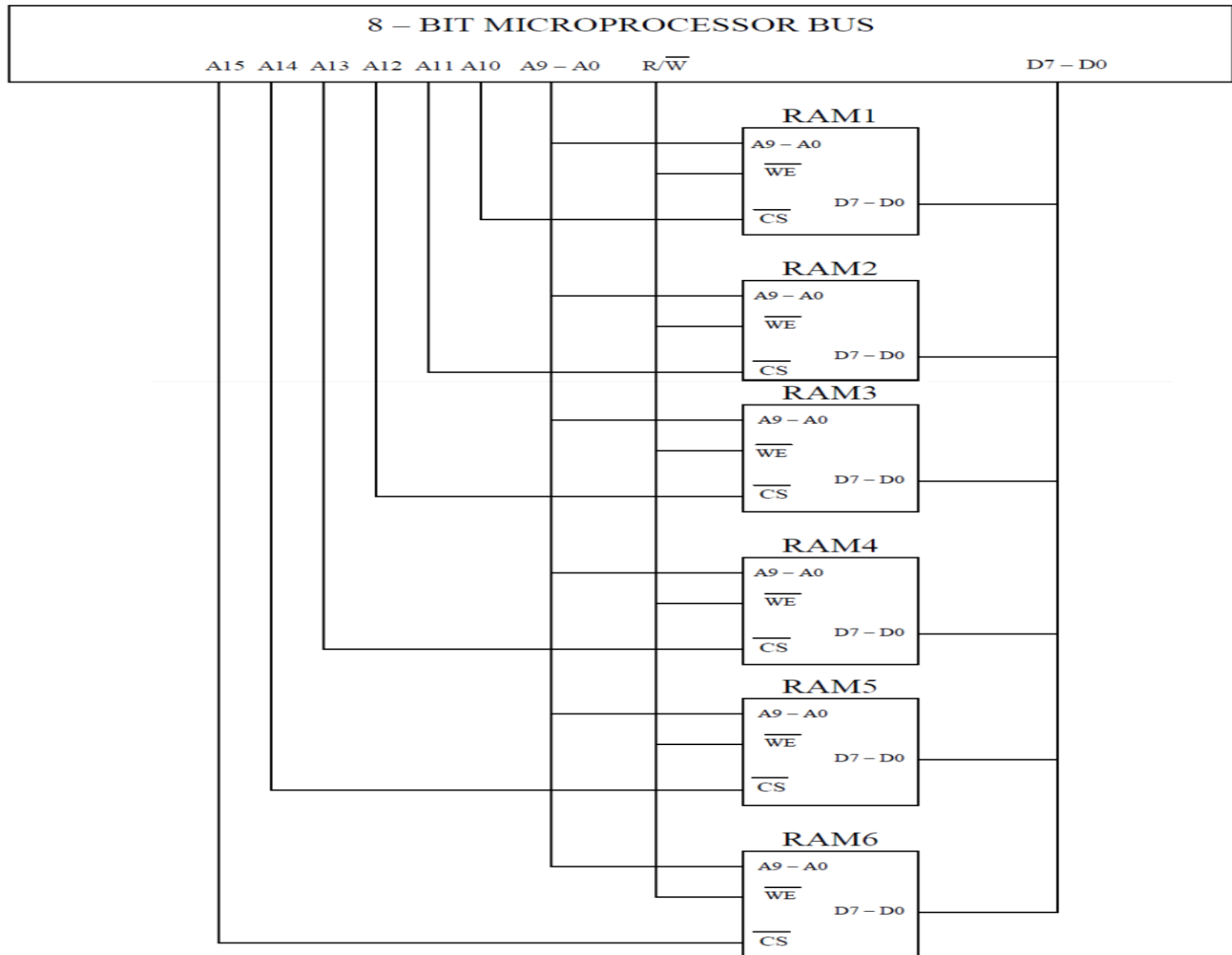


Fig. Linear Decoding



# Main Memory Array Design

## Linear Decoding

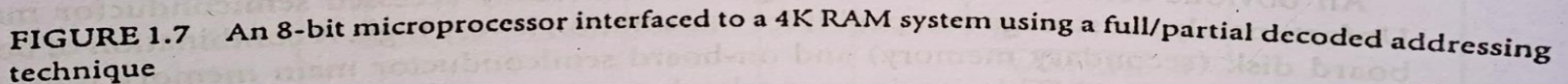
- **Advantages:** The principal advantage of this method is that it does not require any **decoding hardware**.
- **Disadvantage:**
  - 64K bytes of RAM space, interface only 6 K (Wastage address space).
  - Address map is not contiguous, it is sparsely distributed.
- If both A11 & A12 high at the same time, bus conflict occurs.
- If all unused address lines are not utilized as CS for memory, then unused pins don't care (Can be 0 or 1).

# Main Memory Array Design

## Full/partial Decoding

- Difficulties (Bus conflict & sparse address) are eliminated by the use of the full / partial decoded addressing technique.
- Consider the figure in the next slide.
- Here **2- to- 4 decoder** is used and **interface** the 8 bit microprocessor with **4K bytes of RAM**.
- In particular, the four combinations of the lines A11 & A10 select the RAM chips as follow:

A11	A10	Device Selected
0	0	RAM chip 0
0	1	RAM chip 1
1	0	RAM chip 2
1	1	RAM chip 3



**FIGURE 1.7** An 8-bit microprocessor interfaced to a 4K RAM system using a full/partial decoded addressing technique

# Main Memory Array Design

## Memory Decoding using PAL

- Programmable Array Logic (PAL) is similar to a ROM in concept except that it does not provide full decoding of the input lines.
- A PAL provide a partial sum of products which can be obtained via programming.
- It saves a lot of space on the board.
- PAL chip contain a fused programmable AND array and a fixed OR array.

# Main Memory Array Design

## Memory Decoding using PAL

Binary Address Pattern A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0	Device Selected	Address Assignment in Hex
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 . . . . . 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1	RAM CHIP 0	0400 to 07FF
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 . . . . . 0 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1	RAM CHIP 1	0800 to 0BFF
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 . . . . . 0 0 0 1 0 0 1 1 1 1 1 1 1 1 1 1	RAM CHIP 2	1000 to 13FF
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . . . 0 0 1 0 0 0 1 1 1 1 1 1 1 1 1 1	RAM CHIP 3	2000 to 23FF
0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . . . 0 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1	RAM CHIP 4	4000 to 43FF
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 . . . . . 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1	RAM CHIP 5	8000 to 83FF

# Memory Management Concept

- Due to the massive amount of information that must be saved in most systems, the mass storage is often a disk.
- If each access to a disk (hard disk), system throughput reduce to unacceptable levels.
- Obvious solution: large and fast locally accessed semiconductor Memory but
- Unfortunately storage cost per bit is very high.
- A combination of both off board disk (secondary memory) and on board semiconductor must be designed into a system.
- This requires a mechanism to manage the two way flow of information between the primary & secondary media.



# Memory Management Concept

- This mechanism must be able to transfer blocks of data efficiently, keep track of block usage, and replace them in a non arbitrary way.
- An O/S must have resource protection from corruption or abuse by others.
- Users must be able to protect areas of code from each other, while maintaining the ability to communicate and share other areas of code.
- All these requirements indicate the need for a device, located between microprocessor and memory, to control access perform address mapping, and act as an interface between logical (program memory) and microprocessor Physical (memory) address spaces.

# Memory Management Concept

- Since this device **must manage memory use**, it is called the memory management unit (MMU).
- Typical 32-bit microprocessor (Motorola 68030 & Intel 80386) include on chip MMU.
- The MMU reduces the burden of the memory management function of the O/S.
- The basic function provided by the MMU are
  - Address translation &
  - Protection



# Memory Management Unit(MMU)

- The MMU translates these logical addresses to physical addresses provided by the memory chips.
- The MMU can perform address translation in one of two ways:
  - Using substitution technique
  - By adding an offset to each logical address to obtain the corresponding physical address.

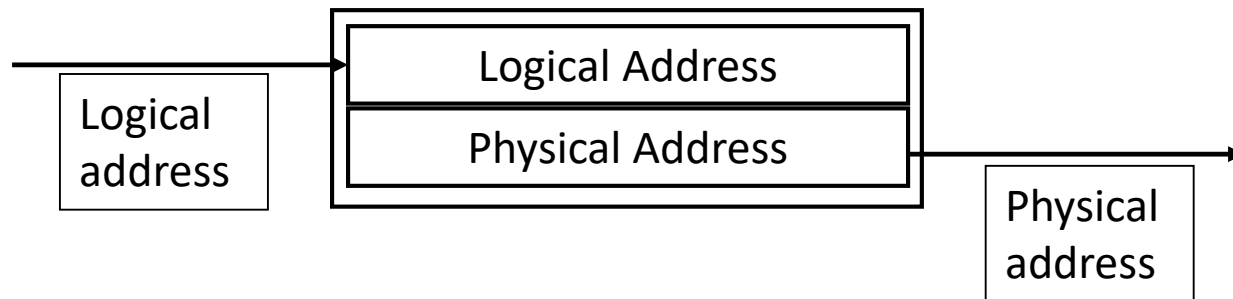


Fig: Address translation using substitution technique

# Memory Management Unit(MMU)

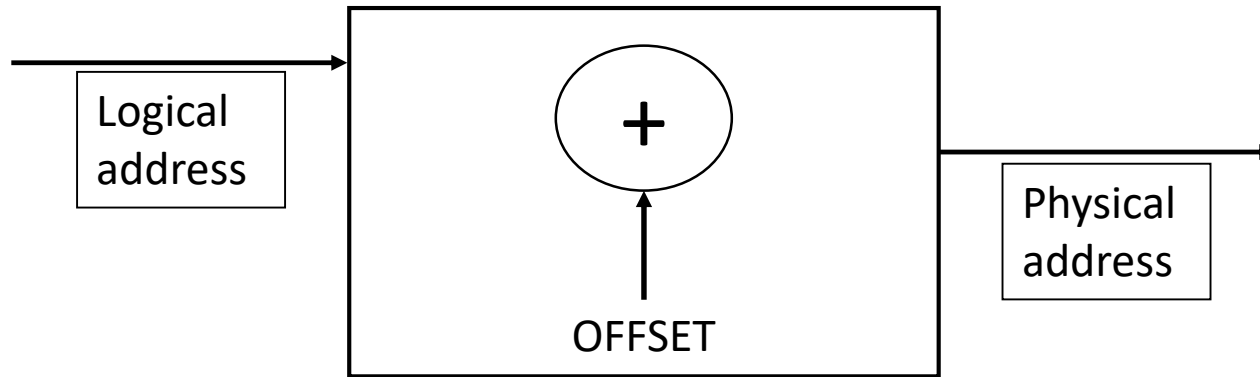
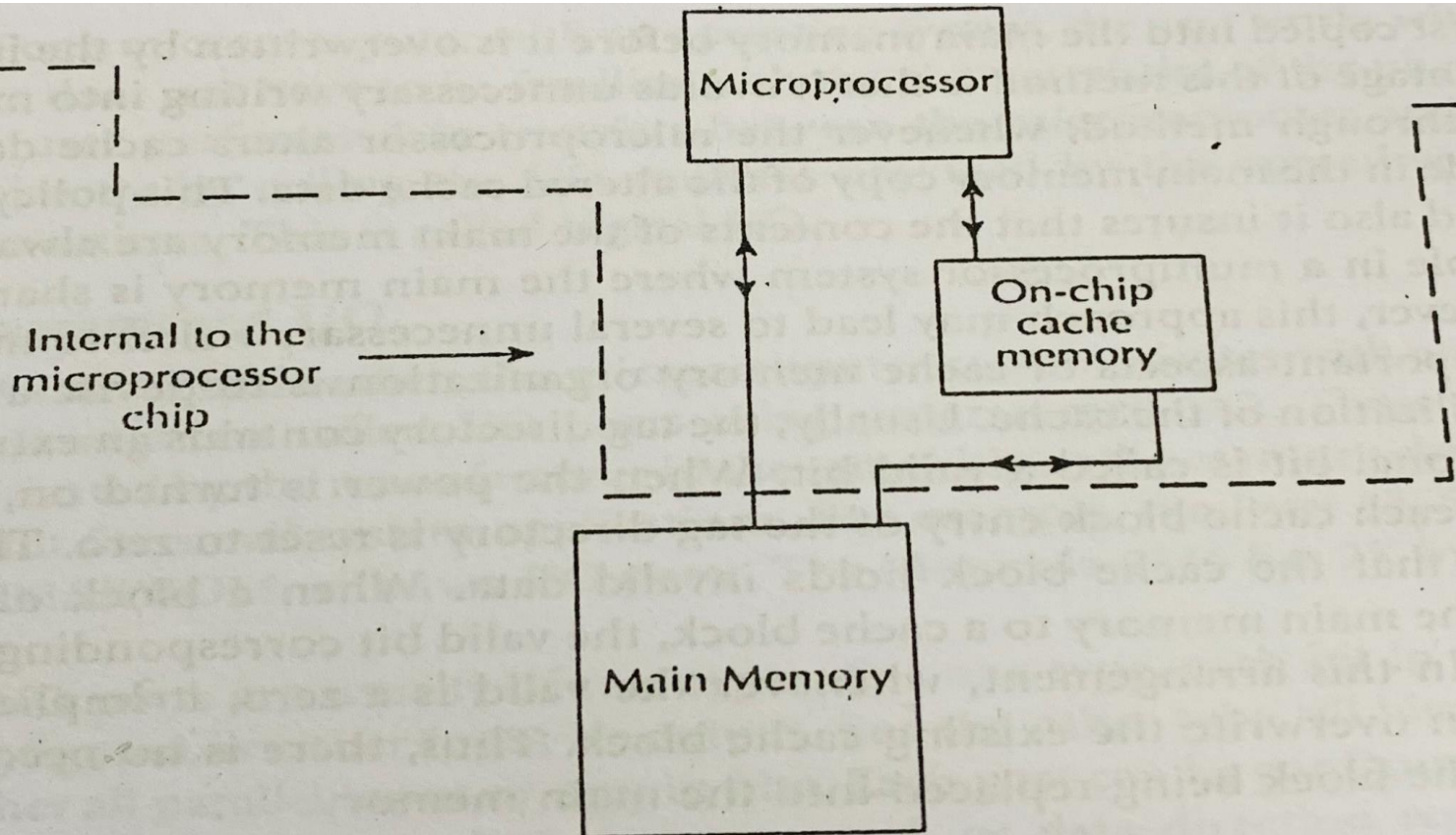


Fig: Address translation using offset technique

- Substitution → faster than offset method.
- Offset method has the advantage of mapping a logical address to physical address by the offset value.

# Cache Memory

- Computer system performance can be significantly improved by introducing a small, expensive, but fast memory between the  $\mu$ P and main memory.
- This memory is called Cache memory.



# Cache Memory

- The relationship between the cache and main memory blocks is established using mapping technique.
- There are three mapping techniques widely used.
  - Direct Mapping
  - Fully-Associative mapping
  - Set-associative mapping
- In direct mapping, the main memory address is divided into two fields
  - Index field and
  - Tag field
- The number of bits in the index field of the main memory is equal to the number of address bits required to access the cache memory.

# Cache Memory

## Direct Mapping

- Assume that the main memory address is  $M$  bits wide and the cache memory address is  $N$  bits wide.
- Then the index field requires  $N$  bits and tag field is  $(M-N)$  bits wide.
- Drawbacks of direct mapping
  - Numerous misses may occur if two or more words with addresses having the same index but different tags are accessed several times

# Cache Memory

## Fully Associative Mapping

- Fully associative mapping is the fastest cache memory mapping that utilizes an associative memory.
- Each associative memory content contains main memory address and its content(data).
- When a  $\mu$ P request a main memory address, it compared associatively with all address in the associative memory.
- If there is a match, the corresponding data word is read from associative cache memory.
- If a miss occurs, the main memory is accessed and the address and its corresponding data are written to the associative cache memory.
- If the cache is full, then certain page replacement algorithms as FIFO are used to replace the cache.

# Cache Memory

## Set-Associative Mapping

- A combination of direct and fully associative mapping.
- Each cache word stores two or more main memory words using the same index address.
- Each main memory word consists of a tag and its data word.
- An index with two or more tags and data words forms a set.
- When the  $\mu P$  make a memory request, the index of the main memory is used as the cache address.
- The tag filed of the main memory address is then compared associatively with all tags stored under the index.
- If a match occurs, the data word is read. But
- If a miss occurs, the data word along with its tag is read from main memory and also written into the cache.

# Input/ Output

- Used for interchange data and information from  $\mu$ P to other unit.
- Three ways of transferring data between  $\mu$ P and physical I/O device
  - Programmed I/O
  - Interrupt driven I/O
  - Direct Memory Access (DMA)
- The microcomputer executes a program to communicate with an external device via a register called programmed I/O.
- An external device request the  $\mu$ P to transfer data by activating a signal on the  $\mu$ P `s interrupt line during interrupt I/O. The  $\mu$ P then called interrupt-service routine to carry out the function.
- Data transfer between the  $\mu$ P`s memory and an external device occurs without  $\mu$ P`s involvement is called Direct Memory Access (DMA)



*Thank You*