Microprocessor and Microcontrollers

Part-2

5

INTERFACING TECHNIQUES

Topics Covered:

1. I/O Mapped I/O
2. Memory Mapped I/O
3. Difference between Memory Mapped I/O and I/O Mapped I/O
4. Memory Device
5. Chip Select Logic

Introduction

There are two method of interfacing memory or I/O devices with the microprocessor are as follows:

a) I/O mapped I/O b) Memory mapped I/O

5.1 I/O MAPPED I/O

- ⇨ In this technique, I/O device is treated as a I/Q device and memory as memory.
- ⇨ Each I/Q device uses eight address lines.
- ⇨ If eight address lines are used to interface to generate the address of the I/O port, then 256 input and 256 output devices can be interfaced with the microprocessor.
- ⇨ The address bus of the 8085 microprocessor is 16 bit, so we can either use lower order address lines i.e. $A_0 – A_7$ or higher order address lines i.e. $A_8 – A_{15}$ to address I/O devices where the address available on $A_0 – A_7$ will be copied on the address lines $A_8 – A_{15}$.
- ⇨ In I/O mapped I/O, the complete 64 Kbytes of memory can be interfaced as all address lines can be used to address memory locations as the address space is not shared among I/O devices and memory and 256 input and /or output devices.
- ⇨ In this type, the data transfer is possible between accumulator A register and I/O devices only.
- ⇨ Address decoding is simple, as less hardware is required.
- ⇨ The separate control signals are used to access I/O devices and memory such as IOR, IOW for I/O port and MEMR,

MEMW for memory hence memory location are protected from the I/O access.

- ⇨ But in this type, arithmetic and logical operation are not possible directly.
- ⇨ Also we cannot use other register for data transfer between I/O device and microprocessor accepts A register.
- ⇨ The figure below shows interfacing I/O devices in I/O mapped I/O.

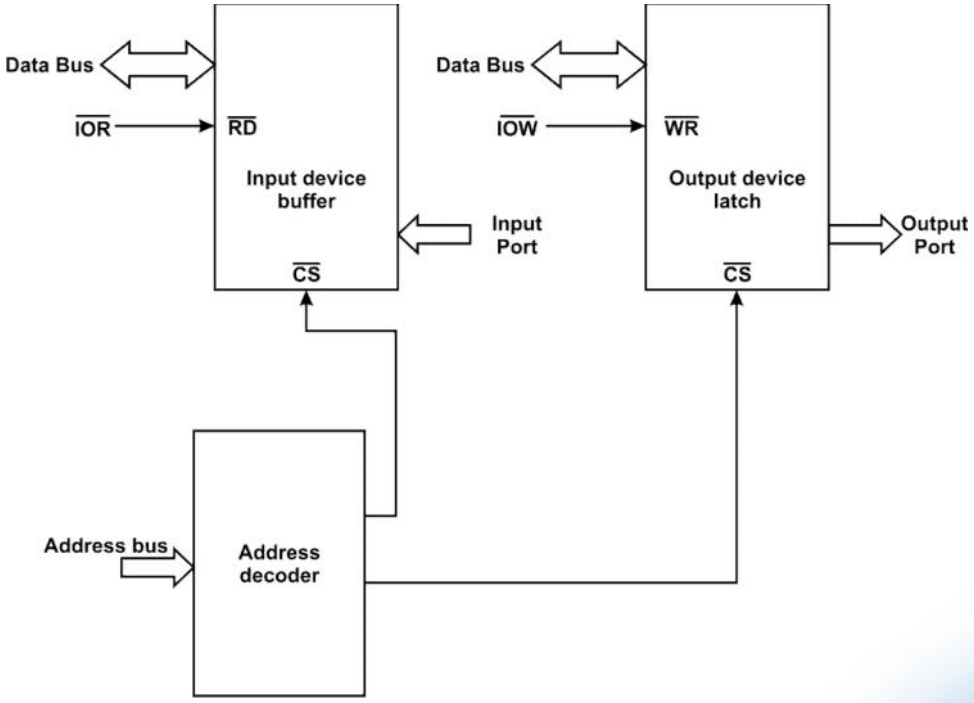

Fig.5.1 I/O mapped I/O ports

5.2 MEMORY MAPPED I/O

- ⇨ In this technique, I/O devices are treated as memory and memory as memory, hence the address of the I/O devices are as same as that of memory i.e. 16 bit for 8085 microprocessor.
- ⇨ So, the address space of the memory i.e. 64 Kbytes will be shared by the I/O devices as well as by memory.
- ⇨ All 16 address lines i.e. A_0-A_{15} is used to address memory locations as well as I/O devices.
- ⇨ The control signals MEMR and MEMW are used to access memory devices as well as I/O devices.
- ⇨ The data transfer is possible between any register of the microprocessor and I/O device or memory device.
- ⇨ Hence, all memory related instructions can be used to access devices as they are treated as memory devices.

- ⇨ Address decoding of the I/O devices and memory devices are complicated and expensive as more hardware is required.
- ⇨ The 8085 microprocessor can access either 64 K I/O ports or memory locations, hence the total numbers of the I/O ports and memory locations should not be greater than 64 K.
- ⇨ I/O devices and memory locations are distinguished by the addresses only.

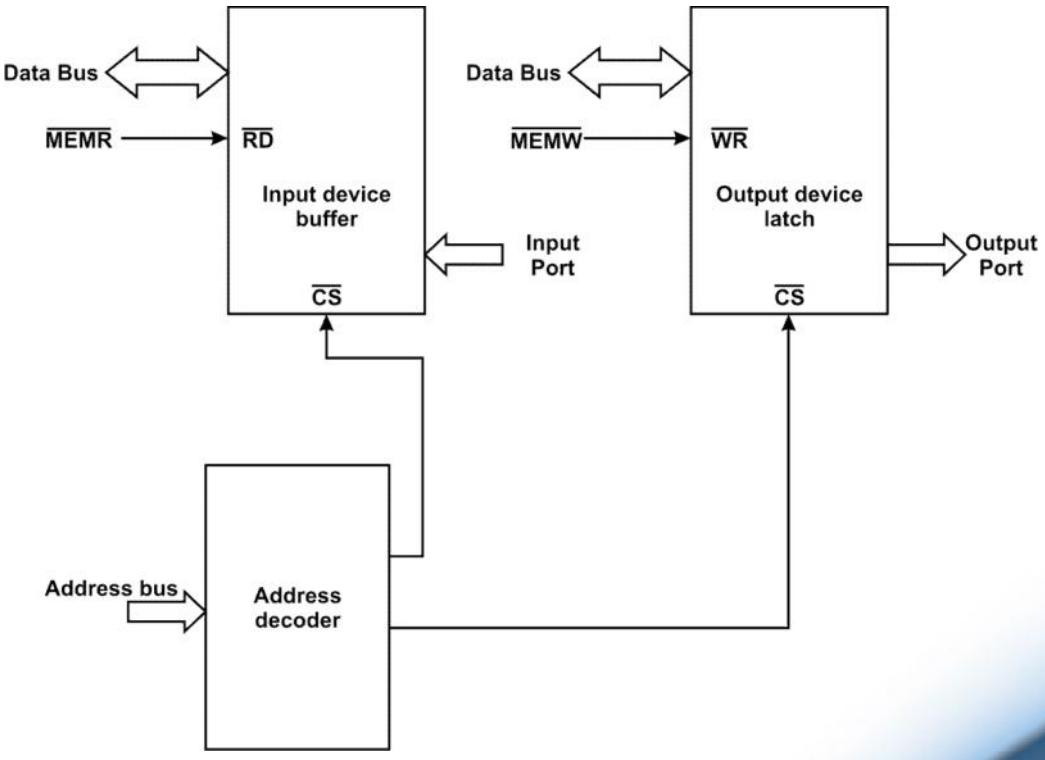

Fig. 5.2 Memory mapped I/O ports

- ⇨ Arithmetic and logical operation can be performed directly on the I/O devices.
- ⇨ Most of the memory instructions are long; hence it reduces the speed of I/O.
- ⇨ Normally, the speed of the I/O devices are very slow, hence the common interface used in memory mapped I/O will reduce the speed of memory access unnecessarily.
- ⇨ The Figure above shows interfacing of I/O devices in memory mapped I/O.

5.3 DIFFERENCE BETWEEN MEMORY MAPPED I/O AND I/O MAPPED I/O

No	I/O mapped I/O	Memory mapped I/O
1	I/O devices are treated as I/O devices and memory devices are treated as memory	I/O and memory devices are treated as memory devices.
2	Separate Control Signals for I/O devices are $\overline{IOR}$, $\overline{IOW}$ and memory devices are $\overline{MEMR}$, $\overline{MEMW}$.	Control signals for memory as well as I/O devices are $\overline{MEMR}$ and $\overline{MEMW}$.
3	IN and OUT instructions are required for I/O read and write operation.	All memory related instruction are used to Access I/O devices.
4	Data transfer is possible between I/O device and Accumulator only.	Data transfer is possible between any register and I/O devices.
5	Address decoding logic is simple.	Address decoding logic is complicated and expensive.
6	8085 can access complete 64 Kbytes of Memory and 256 of Input and 256 output devices as address space is not shared.	8085 can access 64 bytes maximum I/O devices or memory as address space is shared, so total numbers of I/O ports and memory locations should not more than 64K .
7	I/O Device address is 8 bit and memory address is 16 bit.	I/O device and memory address is 16 bit as I/O devices are treated as memory.
8	I/O devices and memory are distinguished by control signals and addresses.	I/O devices and memory are distinguished by only addresses.
9	Arithmetic and logical operations are not possible directly with I/O devices.	Arithmetic and logical operations are possible directly with I/O devices.

4. MEMORY DEVICE

- ⇨ Memory is the storage device which can be used to store monitor program, users program or users data.
- ⇨ So memory is an important component of the microprocessor based system, which will allow you to store program and data.
- ⇨ The memory consists of the thousands of memory cells arranged to store data.
- ⇨ Each memory cell is capable of storing 1 bit of the data.
- ⇨ Hence, to use memory to store programs or data of user or system, memory must be interfaced with microprocessor properly, so that it can be accessed while reading or writing data or program from/to it .
- ⇨ In the same way, input and output devices are also required to read or write data out from the microprocessor using input device such as keyboard or output device using console.
- ⇨ So, these devices must be interface properly with the microprocessor so that user can read data from input device and write data to the output device.

Memory Address

- ⇨ In computer science, a memory address is a unique identifier for a memory location at which a CPU or other device can store a piece of data for later retrieval.
- ⇨ In modern byte- addressable computers, each address identifies a single byte of storage; data too large to be stored in a single byte may reside in multiple bytes occupying a sequence of consecutive addresses.
- ⇨ Some microprocessors were designed to be word-addressable, so that the typical storage unit was actually larger than a byte.

Memory Interfacing

1. Each memory chip like RAM, ROM, EPROM, E^2PROM and DRAM have numbers of pins and these pins are used to accept different kinds of signals.
2. Normally every memory chip has pins for address, data, control signals and chip select signals.

Address Pins

- ⇨ Address pins are used to accept address from the system address bus transmitted by the microprocessor.
- ⇨ The numbers of address pins are depending upon size of the memory as shown in Table below

Table 5.1

Nos of Address lines used	Size of memory in bytes
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024 ≈ 1k
11	2048 ≈ 2k
12	4096 ≈ 4k
13	8192 ≈ 8k
14	16384 ≈ 16k
15	32768 ≈ 32k
16	65536 ≈ 64k

Data Pins

- ⇨ The size of the data bus depends on the data bits, which can be stored in memory location.
- ⇨ Slandered memory data bits stored in a memory location, available are 1, 4 and 8 bits.

$\overline{CS}$ (Chip Select) or $\overline{CE}$ (Chip Enable) Pin

- ⇨ This signal of the memory chip is ACTIVE LOW and acts as master enable pin for read or write operation.
- ⇨ Hence, for every read or write operation, this signal must be low otherwise no operation will be performed.

$\overline{WR}$ (Write Control Signal) Pin

- ⇨ This is an active low input control signal used to write data to the memory location whose address is available on address lines if chip select signal is enable.
- ⇨ This signal is available on system control bus and generated by the microprocessor or other master in the system such as DMA controller or co-processor.

$\overline{RD}$ / $\overline{OE}$ (Read / Output Enable) Pin

- ⇨ This is an active low input control signal used to read data from the memory location whose address is available on address lines if chip select signal is enable.

- ⇨ This signal is available on system control bus and generated by the microprocessor or other master in the system such as DMA controller or co-processor.
- ⇨ Beside these pin described above, some additional pin also available depending on the type of memory.
- ⇨ For example, Vpp and PGM pins are available in EPROM for programming as in normal condition EPROM is read only memory.
- ⇨ But EPROM can be programmed; the separate EPROM programming hardware is required.

Different Memory IC's Available are shown in table below

Table 5.2

Type of memory	IC number	Memory Sizes Address lines x data lines
EPROM	2716	2K X 8
EPROM	2732	4k x 8
EPROM	2764	8k X 8
SRAM	6116	2k X 8
SRAM	6264	8k x 8
SRAM	2114	1K X 4

5.5 CHIP SELECT LOGIC

Chip select logic can be developed using either combination of different gates such as AND, NAND, NOT etc. or decoders.

Using Logic Gates

- ⇨ Now take an example of interfacing of 2K of RAM with the microprocessor 8085, the 8085 is an 8 bit microprocessor. Hence all 8 lines of data bus can be directly connected after de-multiplexing to D_0-D_7 of the RAM memory. The eleven (11) address lines required to access any memory location within 2k memory, so out of 16 address lines (A_0-A_{15}) of 8085 microprocessor, the A_0-A_{10} address lines can be connected directly to generate chip select signal using NAND and NOT gates depending on the addresses required as shown in following figure.

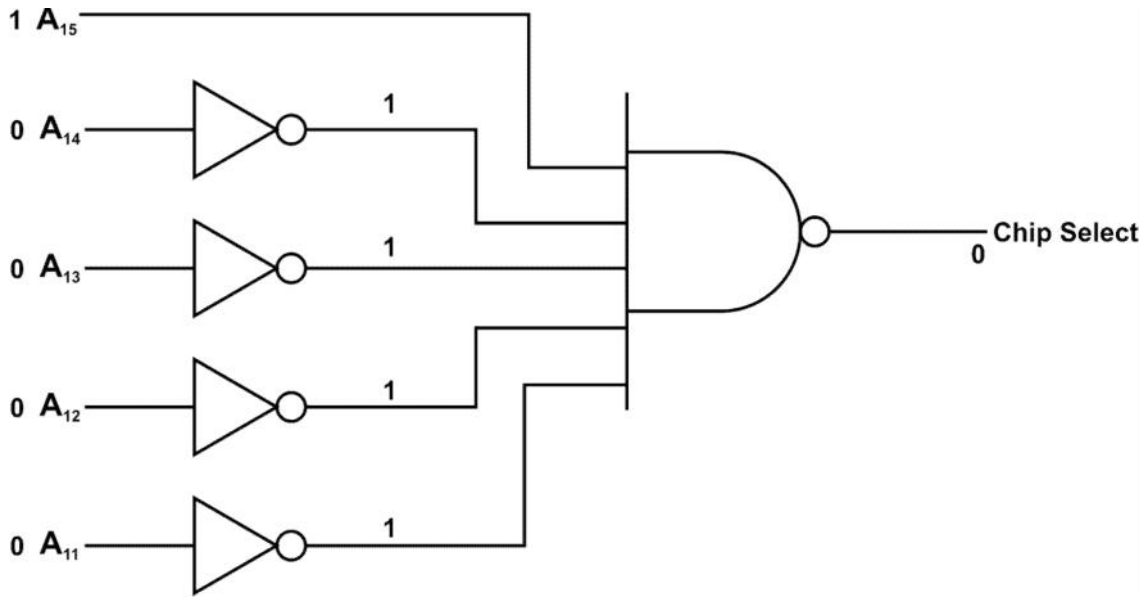

Fig 5.3 Chip select using NAND gates

- ⇨ For generation of chip select we are using NAND gate, when input to NAND gate are all logic 1, then output of NAND gate will be logic 0 and for all other combination the output will be logic 1.
- ⇨ The chip select is active low signal, hence all inputs of the NAND gates must be logic 1 to generate chip select signal as given below.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}
1	0	0	0	0

- ⇨ Hence addresses map of the 2K of RAM is given below.

A 15	A 14	A 13	A 12	A 11	A 10	A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	ADDRESS
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000H
1	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	87FFH
Used for chip select Decoder logic					Connected to A_0 – A_{10} 2 K RAM											

- ⇨ The interfacing diagram of 2K X 8 RAM is shown in following figure

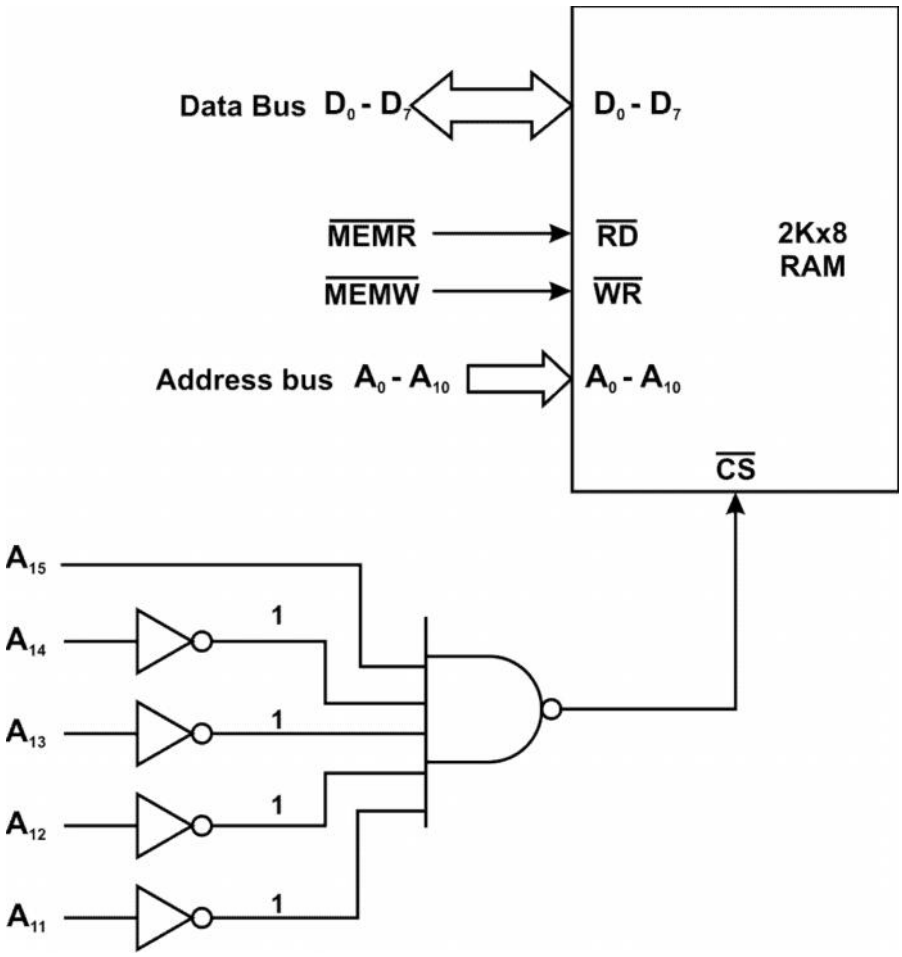

Fig. 5.4 Interfacing of 2K X 8 RAM

Using Decoder

- ⇨ Consider above example of memory interfacing, where remaining address lines i.e. A_{11}-A_{15} can be connected to the decoder. Now connect A_{11}, A_{12}, A_{13} to A, B,C inputs of 3:8 decoder 74LS138 respectively, A_{14} to G_{2A} and G_{2B} enable pins and at last A_{15} to G_1 pin of 3:8 decoder as shown in following figure.

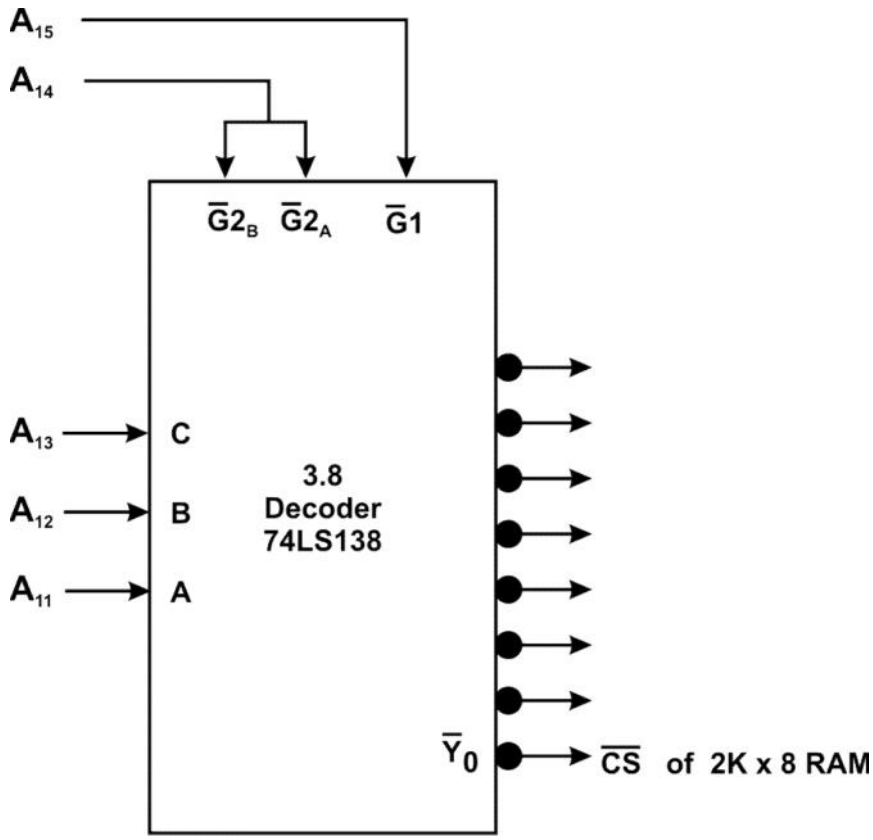

Fig. 5.5 Using decoder

⇨ After making the connection as shown above, the address map of 2K RAM will be same as specified above.

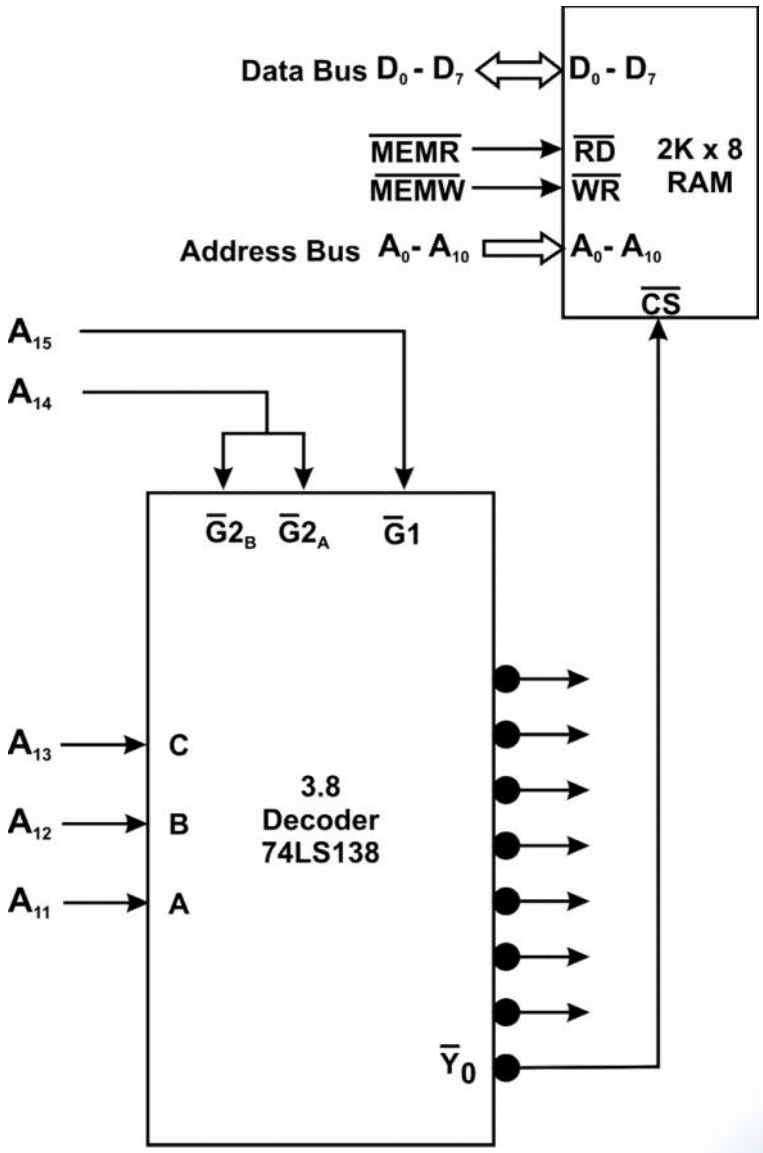

Fig. 5.6 Interfacing of 2K X 8 RAM using decoder chip select logic

⇨ Here, the advantage of using decoder is minimum hardware is required as compared using NAND gates. When we use NAND gates, other logical devices are also required as per requirement as in above examples, NOT gates are used. Hence for numbers of devices, numbers of NAND and other logical devices are required to generate chip select signals.

⇨ But when we use decoder like 3:8 (74LS138), we can generate eight chip select signals using one decoder IC, as it has eight active low output pins. The complete interfacing diagram using decoder to generate chip select signals for 2K of RAM is shown in Fig. 4.6

Exercise

1. Explain I/O Mapped I/O with diagram.
2. Explain Memory Mapped I/O with diagram.
3. Differentiate between I/O Mapped I/O and Memory Mapped I/O.
4. What is a memory device?
5. Explain Memory Interfacing.
6. Explain various data pins.
7. Explain how chip select logic can be used using GATES.
8. Explain how chip select logic can be used using decoders.

References

Computer System Architecture – M. Morris Meno, PHI, 1998

Computer Architecture and Organization - John P Hayes, McGraw Hill, 1998

Digital Computer Fundamentals – Malvino

Digital Computer Fundamentals – Thomas C Bartee, TMG

Computer Organization and Architecture – William Stallings

Microprocessor Architecture and Programming and Applications with the 8085 – R.S. Gaonkar, PRI

❖❖❖❖

6

Unit III

8085 MICROPROCESSOR PROGRAMMING MODEL

Topics Covered:

1. 8085 Programming Model
2. Instruction Classification
3. Instruction Format
4. Overview of 8085 Instruction Set

6.1 8085 PROGRAMMING MODEL

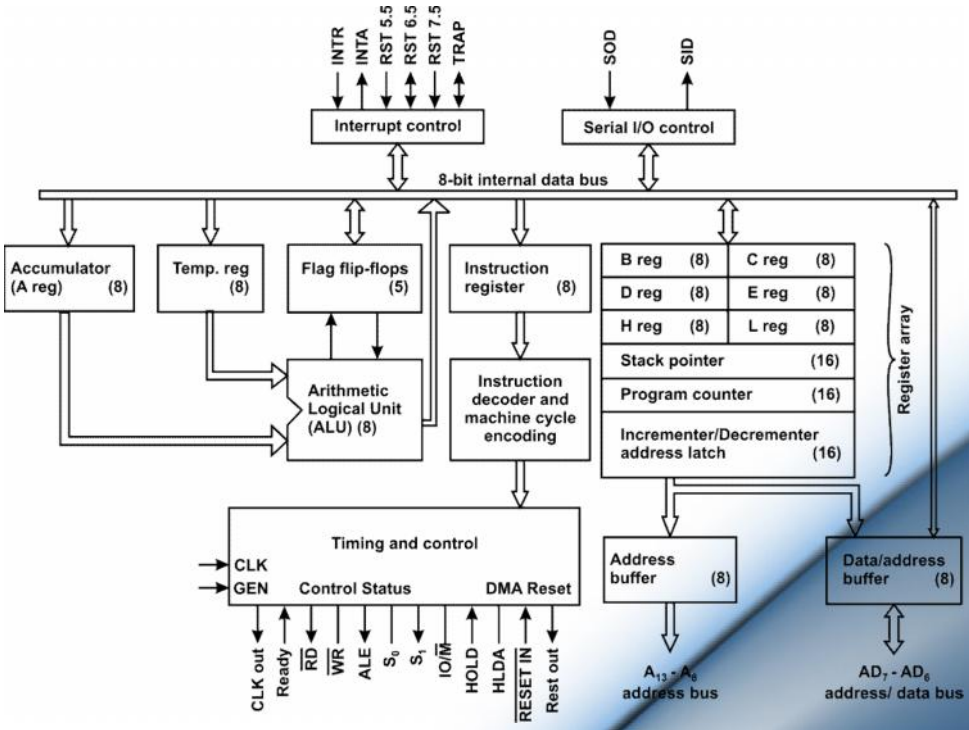

Figure 6.1 8085 programming model

- ⇨ The programming model consists of some segments of the ALU and the registers.
- ⇨ This model does not reflect the physical structure of 8085 but includes information that is critical writing assembly programs.

- ⇨ The model includes six registers; one accumulator and one flag register as shown in following figure.
- ⇨ In addition to this it has two 16-bit registers called stack pointer and program counter

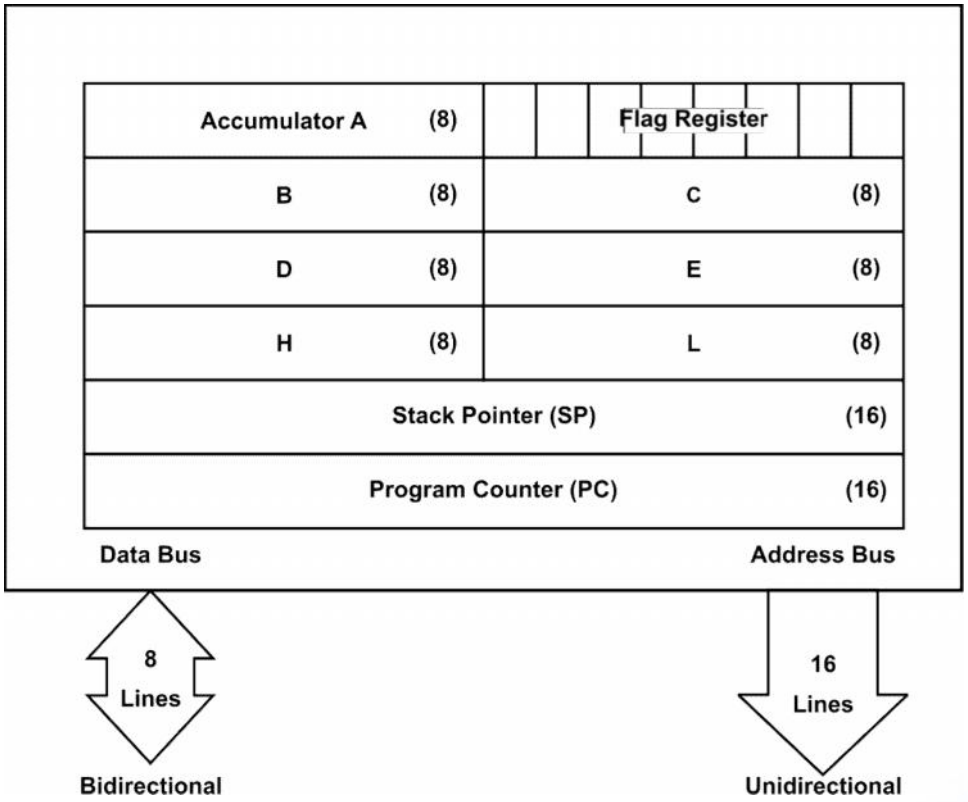

Fig 6.2 8085 Registers

Registers:

- ⇨ 8085 has six general-purpose registers to store 8-bit data.
- ⇨ These registers are B, C, D, E, H, L as shown in above figure.
- ⇨ They can be combined as register pairs – BC, DE, and HL – to perform some 16-bit operations.
- ⇨ The programmer can use these registers to store or copy data into the registers by using data copy instructions.

Accumulator:

- ⇨ The accumulator is an 8-bit register that is part of the arithmetic/logic unit (ALU).
- ⇨ This register is used to store 8-bit data and to perform arithmetic and logical operations.
- ⇨ The result of an operation is stored in the accumulator.
- ⇨ The accumulator is also identified as register A.

Flags:

⇨ The ALU includes five flip flops, which are set or reset as per the operations results in accumulator and other registers.

⇨ They are called Zero(Z) , Carry (CY), Sign(S) , Parity(P) and Auxiliary carry (AC) flags. The most commonly used flags are Zero, Carry and Sign.

⇨

S	Z		AC		P		CY
D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0

Figure 6.3 Flag Register

⇨ The flags are important in decision-making process.

⇨ E.g. the instruction JC (Jump on Carry) is implemented to change the sequence of a program when the Carry CY flag is set.

⇨ Z – Zero : The zero flag is set to 1 when the result is zero; otherwise it is reset.

⇨ CY – Carry : If an arithmetic operation results carry then CY flag is set; otherwise it is reset.

⇨ S – Sign: The sign flag is set if bit D_7 of the result =1 ; otherwise it is reset

⇨ P- Parity: If the result has an even number of 1s , the flag is set; for an odd number of 1s the flag is reset.

⇨ AC – Auxiliary Carry: In an arithmetic operation, when a carry is generated by digit D_4, the Ac flag is set. This flag is used internally for BCD operations; there is no Jump instruction associated with this flag.

Program Counter and Stack Pointer:

⇨ There are two 16-bit registers used to hold memory addresses.
⇨ The size of these registers is 16-bits because the memory addresses are 16-bits.
⇨ The MPU uses PC register to sequence the execution of the instructions.
⇨ The function of Program counter is to point to the memory location from which the next byte is to be fetched.

⇨ When a byte(machine code) is fetched the program counter is incremented by one to point to the next memory location.

⇨ The stack pointer points to the location in R/W memory. The beginning of the stack is defined by loading 16-bit address in the stack pointer. e.g. Instruction to initialize stack pointer is LXI SP,2400H

2. 8085 INSTRUCTION CLASSIFICATION

Instruction:

⇨ An instruction is a binary pattern designed inside microprocessor to perform a specific function.

⇨ Entire group of instruction is called instruction set.

⇨ 8085 instructions are functionally categorized into five types

1) Data transfer (copy) operations
2) Arithmetic operations
3) Logical operations
4) Branching operation
5) Machine control operations

Data transfer (copy) operations:

⇨ This group of instruction copies data from a location called a source to another location called destination, without modifying the contents of source.

e.g. a. Copy contents of register B into register D
b. Load register B with the data byte 35H
c. From memory location 4000H to register B
d. From input keyboard to the accumulator

Arithmetic operations:

Addition:

⇨ Any 8-bit number, or contents of a register, or the contents of memory location can be added to the contents of accumulator and sum is stored in the accumulator.

⇨ No two other 8-bit registers can be added directly (e.g. Contents of register B cannot be added directly to the contents of C) The instruction DAD is a exception; it adds 16-bit data directly in register pair.

Subtraction:

⇨ Any 8-bit number, or contents of a register, or the contents of memory location can be subtracted from the contents of accumulator and the result is stored in the accumulator.

⇨ The subtraction is performed in 2"s complement, and the result, if negative, is expressed in 2"s complement. No two other registers are subtracted directly.

Increment/Decrement:

⇨ The 8-bit contents of register or memory location can be incremented or decremented by 1.

⇨ Similarly, 16-bit contents of register pair can be incremented or decremented.

⇨ The increment/decrement differs from addition and subtraction in such a way that they can be performed on any one the register or in memory location

Logical operations:

AND,OR, Exclusive-OR:

⇨ Any 8-bit number or the contents of a register, or a memory location can be logically ANDed, ORed, or Exclusive-ORed with the contents of the accumulator. The results are stored in accumulator.

⇨ E.g To logically AND the contents of a B register with the contents of A the instruction is ANA B.

Rotate (shift):

⇨ Each bit in the accumulator is can be shifted either left or right to the next position. E.g. To rotate left each binary bit of the accumulator instruction is RLC. (Bit D_7 is placed in the position of D_0 as well as in the Carry flag.)

Compare:

⇨ Any 8-bit numbers or the contents of a register, or a memory location can be compared for equality, greater than, or less than, with the contents of accumulator.

⇨ E.g. The instruction CPI 32H compare the content of accumulator with 32H for less than, equal to or greater than.

Complement:

⇨ The content of accumulator can be complemented; all the 0s are replaced by 1s and all 1s are replaced by 0s.

⇨ E.g. the instruction is CMA to complement the content of Accumulator.

Branching operation:

Jump:

⇨ The conditional jumps are an important aspect of the decision-making process in programming.

⇨ These instructions test for certain condition (e.g. Zero /Carry/Sign etc) and alter the program sequence when condition is met. In addition to conditional jump, the instruction set includes unconditional jump. E.g. JMP 2500H

Call, Return and Restart:

⇨ These instructions change the sequence of program either by calling a subroutine or returning from a subroutine.

⇨ The conditional Call and Return instructions also can test condition flags.

Machine control operations:

⇨ These instructions control machine functions such as Halt, Interrupt or do nothing.

3. INSTRUCTION FORMAT

Instruction word size

⇨ **8085 instruction set is classified into the following three groups according to word size or byte size.**

1) 1-Byte instruction
2) 2-Byte instruction
3) 3-Byte instruction

ONE-Byte Instruction:

⇨ A 1-byte instruction includes opcode and operand in the same byte

⇨ E.g.

Task	Opcode	Operand	Binary Code	Hex Code
Copy contents of accumulator in reg. C	MOV	C, A	0100 1111	4FH
Add contents of reg. B to the contents of accumulator.	ADD	B	1000 0000	80H
Invert(Complement) each bit in the accumulator (Implicit operand)	CMA		0010 1111	2FH

TWO-Byte Instruction:

⇨ In 2-byte instruction first byte specifies the operation code and the second byte specifies the operand.

⇨ E.g.

Task	Opcode	Operand	Binary Code	Hex Code
Load an 8-bit data byte in the accumulator	MVI	A, 50H	0011 1110 (1st byte) 0101 0000 (2nd byte)	3EH 50
Load an 8-bit data byte in reg. C	MVI	C, F2H	0000 0110 (1st byte) 1111 0010 (2nd byte)	06H F2H

THREE-Byte Instruction:

⇨ In 3-byte instruction first byte specifies the operation code and the following two bytes specifies the 16-bit address

⇨ E.g.

Task	Opcode	Operand	Binary Code	Hex Code
Load contents of memory 2050H into A	LDA	2050H	0011 1010 (1st byte) 0101 0000 (2nd byte) 0010 0000 (3rd byte)	3A 50 20
Transfer the program sequence to memory location 2085H	JMP	2085	1100 1010 (1st byte) 1000 0101 (2nd byte) 0010 0000 (3rd byte)	C3 85 20

Opcode Format

⇨ To understand operation code (opcode), we need to examine how an instruction is designed into the microprocessor.

⇨ This information is useful in reading user manual, in which operations codes are specified in binary formats and 8-bits are divided into various groups.

⇨ In 8085 microprocessor chip all operations, registers, and status flags are identified with specific code.

E.g. Internal registers can be identified as follows

CODE	Registers		CODE	Registers
000	B		00	BC
001	C		01	DE
010	D		10	HL
011	E		11	AF OR SP
100	H			
101	L			
111	A			
110	Reserved for memory related operation			

⇨ Some of the operation codes are identified as follows:

Function	Operation Code
Rotate each bit of accumulator to the left by one position.	00000111 = 07H (8-bit opcode)
Add the contents of a register to the accumulator	10000 SSS (5-bit opcode – 3bits reserved for a register)

⇨ E.g. The instruction of ADD B is completed by adding code of the register

Add 10000

Register B ***000***

To A Implicit

Binary instruction 10000 000 = 80H

ADD **Reg. B**

⇨ In assembly language it is expressed as

Opcode	Operand	Hex Code
ADD	B	80H

Function	Operation Code
Move (copy) contents of register Rs(source) to register Rd (Destination)	01 DDD SSS 2-bit opcode for move, Reg. Rd and Reg. Rs

⇨ Adding codes of two registers completes the instruction.

Move (copy) the content	01
To register C	001 (DDD)
From register A	111 (SSS)

Binary instruction **01 001 111 = 4FH**

Opcode operand

⇨ In assembly language it is expressed as

Opcode	Operand	Hex Code
MOV	C, A	4FH

Data Format

⇨ In 8-bit microprocessor systems, commonly used codes and data formats are
- ASCII code
- BCD code
- Signed integers
- Unsigned integers

Addressing Modes

⇨ To perform any operation, we have to give the corresponding instructions to the microprocessor.

⇨ In each instruction, programmer has to specify 3 things:
- o Operation to be performed.
- o Address of source of data.
- o Address of destination of result.

⇨ The method by which the address of source of data or the address of destination of result is given in the instruction is called Addressing Modes.

⇨ The term addressing mode refers to the way in which the operand of the instruction is specified.

Types of Addressing Modes

⇨ Intel 8085 uses the following addressing modes:

1. Direct Addressing Mode
2. Register Addressing Mode
3. Register Indirect Addressing Mode
4. Immediate Addressing Mode
5. Implicit Addressing Mode

Direct Addressing Mode

- ⇨ In this mode, the address of the operand is given in the instruction itself.
- ⇨ Eg. LDA 2500 H Load the contents of memory location 2500 H in accumulator.
  - o LDA is the operation.
  - o 2500 H is the address of source.
  - o Accumulator is the destination.

Register Addressing Mode

- ⇨ In this mode, the operand is in general purpose register.
- ⇨ Eg. MOV A, B Move the contents of register B to A.
  - o MOV is the operation.
  - o B is the source of data.

  A is the destination.

Register Indirect Addressing Mode

- ⇨ In this mode, the address of operand is specified by a register pair.
- ⇨ Mov A, M Move data from memory location specified by H-L pair to accumulator.
  - o MOV is the operation.
  - o M is the memory location specified by H-L register pair.
  - o A is the destination.

Immediate Addressing Mode

- ⇨ In this mode, the operand is specified within the instruction itself.
- ⇨ Eg. MVI A, 05H Move 05 H in accumulator.
  - o MVI is the operation.
  - o 05 H is the immediate data (source).
  - o A is the destination.

Implicit Addressing Mode

- ⇨ If address of source of data as well as address of destination of result is fixed, then there is no need to give any operand along with the instruction.
- ⇨ Eg. CMA Complement Accumulator
  - o CMA is the operation.
  - o A is the source.
  - o A is the destination.

6.4 OVERVIEW OF 8085 INSTRUCTION SET

- ⇨ The following are the notations used to describe the instructions

R= 8085 8-bit register (A, B, C, D, E, H, L)
M= Memory register (location)
Rs = Register source
Rd = Register destination (A, B, C, D, E, H, L)
Rp = Register pair (BC, DE, HL, SP)
( ) = Contents of

❖ **Data transfer instructions:**

These instruction perform six operations

1. Load an 8-bit number in register
2. Copy from register to register
3. Copy between I/O and accumulator
4. Load 16-bit number in a register
5. Copy between register and memory
6. Copy between register and stack memory

Mnemonics	*Examples*	**Operation**
MVI R, 8-bit	MVI B,4FH	Load 8-bit data in a register
MOV Rd, Rs	MOV B,A	Copy data from source register Rs to destination register Rd
LXI Rp, 16-bit	LXI B,2050H	Load 16-bit number in a register pair.
OUT 8-bit (port address)	OUT 01H	Send(write) data byte from the accumulator to an output device.
IN 8-bit (port address)	IN 07H	Accept(read) data byte from an input device and place it in accumulator.
LDA 16-bit	LDA 2050H	Copy data byte into A from memory specified by 16-bit address.
STA 16-bit	STA 2070H	Copy data byte from A into the memory specified by 16-bit address.
LDAX Rp	LDAX B	Copy the data byte into A from the memory specified by the address in the register pair.
STAX Rp	STAX D	Copy the data byte from A into the memory specified by the address in register pair.
MOV Rd, M	MOV B, M	Copy the data byte into destination register from the memory specified by the

		address in HL register.
MOV M, Rs	MOV M, C	Copy the data byte from the source register into memory specified by the address in HL register.

❖ **Arithmetic Instructions:**

The frequently used arithmetic operations are:

1. Add
2. Subtract
3. Increment (Add 1)
4. Decrement (Subtract 1)

Mnemonics	*Examples*	**Operation**
ADD R	ADD B	Add the contents of a register to the register to the contents of A
ADI 8-bit	ADI 37H	Add 8-bit data to the contents of A
ADD M	ADD M	Add the contents of memory to A; the address of memory is in HL register.
SUB R	***SUB C***	Subtract the contents of register from the contents of A.
SUI 8-bit	***SUI 7FH***	Subtract 8-bit data from the contents of A
SUB M	***SUB M***	Subtract the contents of memory from A; the address of memory is in HL register.
INR R	***INR D***	Increment the contents of register.
INR M	***INR M***	Increment the contents of memory, the address of which is in HL.
DCR R	***DCR E***	Decrement the contents of a register.
DCR M	***DCR M***	Decrement the contents of a memory, the address of which is in HL.
INX Rp	***INX H***	Increment the contents of a register pair.
DCX Rp	***DCX B***	Decrement the contents of a register pair.

❖ **Logic and Bit Manipulation Instructions:**

These instructions include the following operations:

1. AND
2. OR
3. X-OR(Exclusive OR)
4. Compare
5. Rotate Bits

Mnemonics	*Examples*	**Operation**
ANA R	ANA B	Logically AND the contents of a register with the contents of A.
Mnemonics	*Examples*	**Operation**
ANI 8-bit	ANI 2FH	Logically AND 8-bit data with the contents of A.
ANA M	ANA M	Logically AND the contents of memory with the contents of A; the address of memory is in HL register
ORA R	ORA E	Logically OR the contents of a register with the contents of A
ORI 8-bit	ORI 3FH	Logically OR 8-bit data with the contents of A
ORA M	ORA M	Logically OR the contents of memory with the contents of A; the address of memory is in HL register.
XRA R	XRA B	Exclusive OR the contents of a register with the contents of A
XRI 8-bit	XRI 6AH	Exclusive OR 8-bit data with the contents of A
XRA M	XRA M	Exclusive OR the contents of memory with the contents of A; the address of memory are in HL register.
CMP R	CMP B	Compare the contents of register with the contents of A for less than, equal to, or greater than
CPI 8-bit	CPI 4FH	Compare 8-bit data with the contents of A for less than, equal to, or greater than

❖ **Branch Instructions:**

The following instruction changes the program sequence.

Mnemonics	*Examples*	**Operation**
JMP 16-bit	JMP	Change the program sequence to the

address	2050H	specified 16-bit address.
JZ 16-bit address	JZ 2080H	Change the program sequence to the specified 16-bit address if the Zero flag is set.
JNZ 16-bit address	JNZ 2070H	Change the program sequence to the specified 16-bit address if Zero flag is reset.
JC 16-bit address	JC 2025H	Change the program sequence to the specified 16-bit address if the Carry flag is set.
JNC 16-bit address	JNC 2030H	Change the program sequence to the specified 16-bit address if the Carry flag is set.
CALL 16-bit address	CALL 2075H	Change the program sequence to the location of a subroutine.
RET	RET	Return to the calling program after completing the subroutine sequence

❖ **Machine Control Instructions:**

These instructions affect the operation of the processor

Mnemonics	*Examples*	**Operation**
HLT	HLT	Stop processing and wait
NOP	NOP	Do not perform any operation

Exercise

1. What is an instruction set?
2. Give the functional categories of 8085 microinstructions.
3. Define Opcode and operand.
4. Define the types of branching operations.
5. Define one byte/two byte/three byte instruction with one example.
6. What is the machine control operations used in 8085 microprocessor?
7. What is data transfer instructions?
8. What are the notations used in the 8085 instructions?
9. Give the classification of Instruction set.
10. Explain with the help of a diagram 8085 programming model.
11. Explain various register used in 8085 microprocessor.
12. Explain various addressing modes used in 8085 microprocessor.

References

Computer System Architecture – M. Morris Meno, PHI, 1998

Computer Architecture and Organization - John P Hayes, McGraw Hill, 1998

Digital Computer Fundamentals – Malvino

Digital Computer Fundamentals – Thomas C Bartee, TMG

Computer Organization and Architecture – William Stallings

Microprocessor Architecture and Programming and Applications with the 8085 – R.S. Gaonkar, PRI

❖❖❖❖

7

8085 PROGRAMS

List of Programs covered:

1. To add two 8-bit data
2. To add two 8-bit data present in the memory
3. To add two 16-bit data
4. To subtract two 16-bit data
5. To add two 2-digit BCD data
6. To add two 4-digit BCD data
7. To multiply two numbers of 8-bit data
8. To multiply two numbers of 16-bit data
9. To divide two numbers of 8-bit data
10. To add an array of data
11. To search smallest data in the array
12. To search largest data in the array
13. To sort an array of data in ascending order
14. To sort an array of data in descending order
15. To find the square root of an 8-bit binary number
16. To convert 2 digit BCD to binary number
17. To convert 8-bit binary number to BCD
18. To convert 8-bit binary to ASCII
19. To convert ASCII code to binary value
20. Transfer a block of data from one location to another

7.1 TO ADD TWO 8-BIT DATA

```
LDA 0000
MOV  B,A
LDA 0001
MVI  C,00
ADD B
```

```
        JNC AHEAD
        INR C
[LAB1]  STA 0002
        MOV A,C
        STA 0003
        HLT
```

Here, 0000 -> 1st Operand 0001 -> 2nd Operand

0003 -> Sum 0004 -> Carry

7.2 TO ADD TWO 8-BIT DATA PRESENT IN THE MEMORY

```
        LXI H,0000
        MVI C,00
        MOV A,M
        INX H
        ADD M
        JNC LAB1
        INR C
[LAB1]  INX H
        MOV M,A
        INX H
        MOV M,C
        HLT
```

Here, 0000 -> 1st Operand 0001 -> 2nd Operand

0003 -> Sum 0004 -> Carry

7.3 TO ADD TWO 16-BIT DATA

```
        LHLD 0000
        XCHG
        LHLD 0002
        XRA A
        DAD D
```

```
        JNC LAB1
        INR A
[LAB1]  SHLD 0004
        STA 0006
        HLT
```

Here, 0000 & 0001 -> 1st Operand 0002& 0003 -> 2nd Operand
0004 & 0005 -> Sum 0006 -> Carry

---

4. TO SUBTRACT TWO 16-BIT DATA

```
        LDA 0002
        MOV B,A
        LDA 0000
        SUB B
        STA 0004
        LDA 0003
        MOV  B,A
        LDA  0001
        SBB
        STA 0005
        HLT
```

Here, 0000 & 0001 -> 1st Operand 0002& 0003 -> 2nd Operand
0004 & 0005 -> Subtracted Result

---

5. TO ADD TWO 2-DIGIT BCD DATA

```
        LDA 0000
        MOV B,A
        LDA 0001
        MIV C,00
        ADD D
        DAA
```

```
        JNC LAB1
        INR C
[LAB1]  STA 0002
        MOV A,C
        STA 0003
        HLT
```

Here, 0000 -> 1st Operand 0001 -> 2nd Operand

0002 -> Sum 0003 -> Carry

7.6 TO ADD TWO 4-DIGIT BCD DATA

```
        LDA 0000
        MOV B,A
        LDA 0002
        MVI C,00
        ADD B
        DAA
        STA 0004
        LDA 0001
        MOV B,A
        LDA 0003
        ADC B
        DAA
        STA 0005
        JNC GO
        INR C
[GO]    MOV A,C
        STA 0006
        HLT
```

Here, 0000 & 0001 -> 1st Operand 0002& 0003 -> 2nd Operand

0004 & 0005 -> Sum 0006 -> Carry

7.7 TO MULTIPLY TWO NUMBERS OF 8-BIT DATA

```
        LXI H,00
        MVI C,00
        XRA A
        MOV B,M
        INX H
        MOV D,M
[REPT]  ADD D
        JNC GO
        INR C
[GO]    DCR B
        JNZ REPT
        INX H
        MOV M,A
        INX H
        MOV M,C
        HLT
```

Here, 0000 -> 1st Operand 0001 -> 2nd Operand

0002 -> Sum 0003 -> Carry

7.8 TO MULTIPLY TWO NUMBERS OF 16-BIT DATA

```
        LHLD 0000
        SPHL
        LHLD 0002
        XCHG
        LXI H,0000
        LXI B,0000
[NEXT]  DAD SP
        JNC AHED
        INX B
[AHED]  DCX D
        MOV A,E
        ORA D
```

```
        JNZ NEXT
        SHLD 0004
        MOV L,C
        MOV H,B
        SHLD 0006
        HLT
```

Here, 0000 & 0001 -> 1st Operand 0002& 0003 -> 2nd Operand

0004 -> 1st byte of Product

0005 -> 2nd byte of Product

0006 -> 3rd byte of Product

0007 -> 4th byte of Product

7.9 TO DIVIDE TWO NUMBERS OF 8-BIT DATA

```
        LDA 0001
        MOV  B,A
        LDA 0000
        MVI C,00
[AGO]   CMP B
        JC STORE
        SUB B
        INR C
        JMP AGO
[STO]   STA 0003
        MOV A,C
        STA 0002
        HLT
```

Here, 0000 -> Dividend 0001 -> Divisor

0002 -> Quotient 0003 -> Remainder

10. TO ADD AN ARRAY OF DATA

```
        LXI H,0000
        MOV B,M
        MVI C,00
        XRA A
[REPT]  INX H
        ADD M
        JNC AH1
        INR C
 [AH1]  DCR B
        JNZ REPT
        STA  1000
        MOV  A,C
        STA  1001
        HLT
```

Here, 0000 -> No. of data to be added (n). Data are taken from next „n" consecutive memory locations.

1000 -> 1st byte of sum

1001 -> 2nd byte of sum

11. TO SEARCH SMALLEST DATA IN THE ARRAY

```
        LXI H,0000
        MOV B,M
        INX H
        MOV A,M
        DCR B
[LOOP]  INX H
        CMP M
        JC AHED
        MOV A,M
 [AHD]  DCR B
```

```
        JNZ LOOP
        STA 1000
        HLT
```

Here, 0000 -> No. of data(n). Data are taken from next 'n' consecutive memory locations.

1000 ->Smallest element

7.12 TO SEARCH LARGEST DATA IN THE ARRAY

```
        LXI H,0000
        MOV B,M
        INX H
        MOV A,M
        DCR B
[LOOP]  INX H
        CMP M
        JNC AHED
        MOV A,M
[AHED]  DCR B
        JNZ LOOP
        STA 1000
        HLT
```

Here, 0000 -> No. of data(n). Data are taken from next 'n' consecutive memory locations.

1000 ->Largest element

7.13 TO SORT AN ARRAY OF DATA IN ASCENDING ORDER

```
        LDA 0000
        MOV  B,A
        DCR B
[LOP2]  LXI H,1000
        MOV C,M
```

```
        DCR C
[LOP1]  INX H
        MOV A,M
        INX H
        CMP M
        JC AHD1
        MOV D,M
        MOV M,A
        DCX H
        MOV M,D
        JMP AHED
[AHD1]  DCX H
[AHED]  DCR C
        JNZ LOP1
        DCR B
        JNZ LOP2
        HLT
```

Here, 0000 -> No. of data (n). Data are taken from next „n" consecutive memory locations.

Sorted data present in the same memory location.

7.14 TO SORT AN ARRAY OF DATA IN DESCENDING ORDER

```
        LDA 0000
        MOV  B,A
        DCR B
[LOP2]  LXI H,1000
        MOV C,M
        DCR C
[LOP1]  INX H
        MOV A,M
        INX H
        CMP M
```

```
        JNC AHD1
        MOV D,M
        MOV M,A
        DCX H
        MOV M,D
        JMP AHED
[AHD1]  DCX H
[AHED]  DCR C
        JNZ LOP1
        DCR B
        JNZ LOP2
        HLT
```

Here, 0000 -> No. of data (n). Data are taken from next „n" consecutive memory locations.

Sorted data present in the same memory location.

7.15 TO FIND THE SQUARE ROOT OF AN 8-BIT BINARY NUMBER

```
        LDA 0000
        MOV B,A
        MVI C,02
        CALL DIV
[REPT]  MOV E,D
        MOV A,M
        MOV C,D
        CALL DIV
        MOV A,D
        ADD E
        MVI C,02
        CALL DIV
        MOV A,E
        CMP D
        JNZ REPT
```

```
        STA 0001
        HLT
[DIV]   MOV D,00
[NEXT]  SUB C
        INR D
        CMP C
        JNC NEXT
        RET
```

Here, 0000 -> data 0001 -> Square root of data

7.16 TO CONVERT 2 DIGIT BCD TO BINARY NUMBER

```
        LDA 0000
        MOV E,A
        ANI F0
        RLC
        RLC
        RLC RLC
        MOV B,A
        XRA A
        MVI C,0A
        ADD B
[REPT]  DCR C
        JNZ REPT
        MOV B,A
        MOV A,E
        ANI 0F
        ADD B
        STA 1000
        HLT
```

Here, 0000 -> BCD data 1000 -> Binary data (result)

7.17 TO CONVERT 8-BIT BINARY NUMBER TO BCD

```
        MVI   E,00
        MOV   D,E
        LDA 0000
[HUND]  CPI 64
        JC TEN
        SUI 64
        INR E
        JMP HUND
 [TEN]  CPI 0A
        JC UNIT
        SUI 0A
        INR D
        JMP TEN
[UNIT]  MOV C,A
        MOV A,D
        RLC
        RLC
        RLC
        RLC
        ADD C
        STA  2500
        MOV   A,E
        STA  2501
        HLT
```

Here, 0000 -> Binary data

2500 -> Ten's and Units's digit

2501 -> Hundred's digit

7.18 TO CONVERT 8-BIT BINARY TO ASCII

```
        LDA 0000
        MOV B,A
```

```
        ANI 0F
        CALL CODE
        STA 0001
        MOV A,B
        ANI F0
        RLC
        RLC
        RLC
        RLC
        CALL CODE
        STA 0002
        HLT
[CODE]  CPI 0A
        JC SKIP
        ADI 07
[SKIP]  ADI 30
        RET
```

Here, 000 ->Hexa data

0001 -> ASCII Code of LSB of data

0002 -> ASCII Code of MSB of data

7.19 TO CONVERT ASCII CODE TO BINARY VALUE

```
        LXI H,0000
        MOV D,M
        LXI B,1000
[LOOP]  INX H
        MOV A,M
        CALL BIN
        STAX B
        INX B
        DCR D
        JNZ LOOP
        HLT
```

```
[BIN]  SUI 30
       CPI 0A
       RC
       SUI 07
       RET
```

Here, 0000 -> No. of data (n). Data (ASCII) are taken from next „n" consecutive memory locations.
(HL register pair points to source memory)
1000 -> Result starting from this memory location
(BC register pair points to destination memory)

7.20 TRANSFER A BLOCK OF DATA FROM ONE LOCATION TO ANOTHER

```
       MVI  D, 0AH
       LXI  H, D000H
       LXI  B, D100H
NEXT:  MOV A, M
       STAX  B
       INX  H
       INX  B
       DCR  D
       JNZ  NEXT
       HLT
```

Here, before the execution, the ten data bytes must be stored from memory location D000H. After the execution, the contents of source block will be transferred to destination block starting from D100H.

Exercise:

1. To find GCD of two numbers
2. To find LCM of two numbers
3. To swap block of data.

References

Computer System Architecture – M. Morris Meno, PHI, 1998

Computer Architecture and Organization - John P Hayes, McGraw Hill, 1998

Digital Computer Fundamentals – Malvino

Digital Computer Fundamentals – Thomas C Bartee, TMG

Computer Organization and Architecture – William Stallings

Microprocessor Architecture and Programming and Applications with the 8085 – R.S.Gaonkar, PRI

❖❖❖❖

8

INTRODUCTION TO MODERN DAY COMPUTER SYSTEMS

Topics Covered:

1. Introduction
2. Hardware
3. Processor
4. Bus System
5. PCI (Peripheral Component Interface) Bus

1. INTRODUCTION

- ⇨ A computer system is made up of both hardware and software.
- ⇨ Software is another term for computer program.
- ⇨ Software controls the computer and makes it do useful work. Without software a computer is useless, like to a car without someone to drive it.
- ⇨ Hardware refers to the physical components that make up a computer system.
- ⇨ These include the computer's processor, memory, monitor, keyboard, mouse, disk drive, printer and so on.

2. HARDWARE

- ⇨ The hardware of a computer system is made up of a number of electronic devices connected together. Figures 8.1 and 8.1A are block diagrams of a typical computer system.

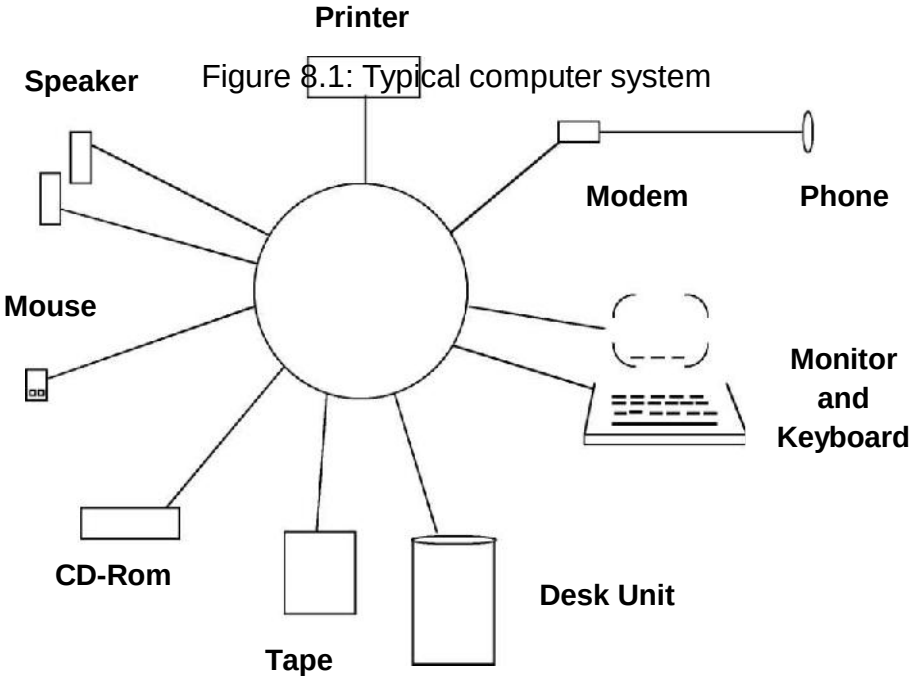

Figure 8.1: Typical computer system

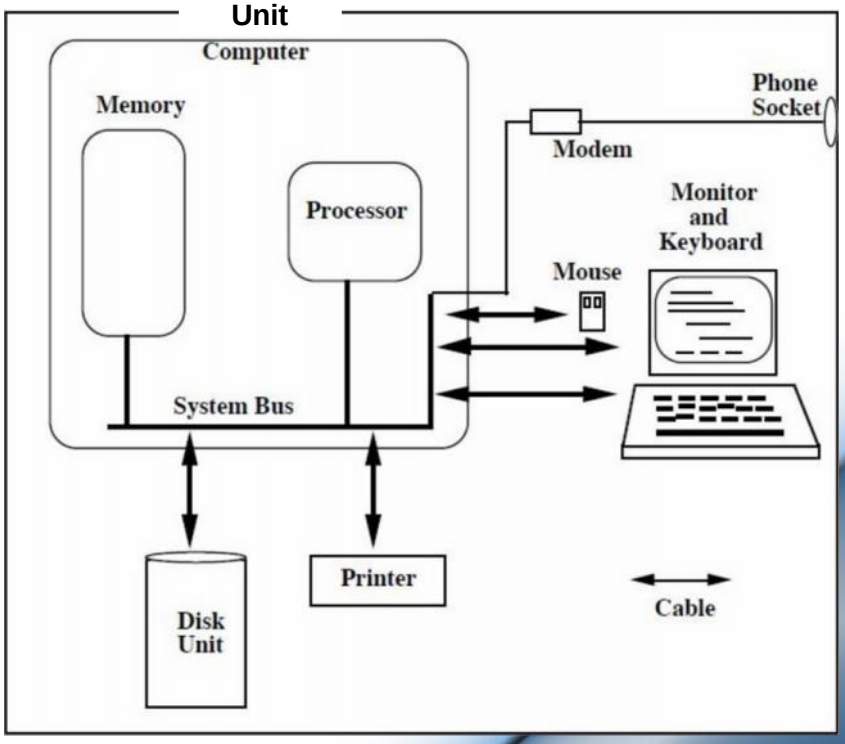

Figure 8.1A: Typical computer system: Processor and Memory (RAM)

- ⇨ A computer has two major internal components, namely its processor and its memory.
- ⇨ There will also be a power supply unit (not shown) to provide power for the system.

- ⇨ The term device is used to describe any piece of hardware that we connect to a computer such as a keyboard, monitor, disk drive, printer and so on.
- ⇨ Such devices are also sometimes described as peripheral devices or simply peripherals.
- ⇨ They may be classified as input/output (I/O) devices and storage devices.
- ⇨ As the name suggests, I/O devices are responsible for communicating with the computer, providing input for the computer to process and arranging to display output for computer users.
- ⇨ The keyboard and mouse are commonly used input devices. The monitor is the commonest output device, followed by the printer for hardcopy (permanent) output.
- ⇨ Storage devices are used to store information in a computer system.
- ⇨ The memory is used to store information inside the computer while the computer is switched on.
- ⇨ Disk storage is the commonest form of external storage, followed by the tape storage.
- ⇨ External storage devices can store information indefinitely or more realistically, for some number of years.
- ⇨ A very important component of a computer system is the system bus. This is used to transfer information between all system components.
- ⇨ It is crucial to understand that all information is represented inside a computer system in binary form i.e. using the binary numbers 1 and 0.
- ⇨ The hardware of a computer system has no other way of representing information.
- ⇨ Thus when you press a key on a computer's keyboard, a binary number (code) which represents the symbol on that key is transmitted to the computer and not the symbol itself, for example, 'A', displayed on the key.
- ⇨ Similarly, when a computer transmits a character to be displayed on the monitor, it is the binary code representing that character that is sent to the monitor.
- ⇨ The monitor hardware takes this binary code and displays the corresponding symbol on the screen.
- ⇨ To reiterate, all information is transmitted and manipulated inside a computer system in the form of binary numbers.

- A binary digit (1 or 0) is called a bit and a group of 8 bits is called a byte.
- When describing storage capacity, the byte and multiples of bytes are the units used. A kilobyte (Kb) is 2^{10} (1024) bytes, a megabyte (Mb) is 2^{20} bytes (1024Kb), a gigabyte (Gb) is 2^{30} bytes (1024Mb) and a terabyte (Tb) is 2^{40} bytes (1024Gb).
- When describing transmission speeds, the number of bits per second (bps) is the unit used.
- A typical modem can handle speeds of up to 56 Kbps i.e 56 kilo bits per second or approx 56,000 bps.

8.3 THE PROCESSOR

- The processor as its name suggests is the unit that does the work of the computer system i.e. it executes computer programs.
- Software is composed of instructions, which are executed (obeyed) by the processor.
- These instructions tell the processor when and what to read from a keyboard; what to display on a screen; what to store and retrieve from a disk drive and so on.
- A computer program is a set of such instructions that carries out a meaningful task. It is worth remembering at this stage that the processor can only perform a limited range of operations.
- It can do arithmetic, compare numbers and perform input/output (read information and display or store it).
- It is instructive to bear in mind that all computer programs are constructed from sequences of instructions based on such primitive operations.
- The processor itself is made up of a number of components such as the arithmetic logic unit (ALU) and the control unit (CU).
- The ALU carries out arithmetic operations (e.g. addition and subtraction) and logical operations (e.g. and, or xor) while the CU controls the execution of instructions.
- Traditionally, the processor is referred to as the central processing unit or CPU.
- With the advent of microprocessors, the term MPU or microprocessor unit is also used.

- A microprocessor is simply a processor contained on a single silicon chip.
- In addition to the ALU and CU, the processor has a small number (usually less than 100) of storage locations to store information that is currently being processed.
- These locations are called registers and depending on the processor, a register may typically store 8, 16, 32 or 64 bits.
- The register size of a particular processor allows us to classify the processor. Processors with a register size of n-bits are called n-bit processors, so that processors with 8-bit registers are called 8-bit processors, similarly there are 16-bit, 32-bit and 64-bit processors.
- An n-bit processor is said to have an n-bit word size so a 32-bit processor has a 32-bit word size.
- The greater the number of bits the more powerful the processor is, since it will be able to process a larger unit of information in a single operation.
- For example, a 32-bit processor will be able to add two 32-bit numbers in a single operation whereas an 8-bit processor will only be able to add two 8-bit numbers in a single operation.
- An n-bit processor will usually be capable of transferring n-bits to or from memory in a single operation. This number of bits is also referred to as the memory word size.
- So, while a byte refers to an 8-bit quantity, a word can mean 8, 16, 32, 64 or some other number of bits.
- On some machines a word is taken to mean a 16-bit quantity and the term long word is used to refer to a 32-bit quantity.
- An alternative method of classifying a processor is to use the width of the data bus, in which case an n-bit processor describes one operating with a data bus of n-bits.
- This means that the CPU can transfer n-bits to another device in a single operation.
- Using this classification, the Intel 8088 microprocessor is an 8-bit processor since it uses an 8-bit data bus, although its CPU registers are in fact 16-bit registers. Similarly the Motorola 68000 is classified as a 16-bit processor, even though its CPU registers are 32-bit registers. Sometimes a combination of the two classifications is used where the 8088 might be described as 8/16-bit processor and the Motorola 68000 as a 16/32-bit processor.

- The data bus width is very important in a computer system, since it determines the amount of information that can be transferred to or from the CPU, in a single operation.
- This means, for example, that the Motorola 68000 would have to transfer two 16-bit items to the CPU to fill a 32-bit register, since the data bus width is 16-bits.
- I/O devices and memory operate at very slow speeds compared to the speed of the CPU. As a result, the CPU is frequently delayed by these slower devices, waiting for information to be transferred along the data bus.
- So, the more information we can transfer in a single operation, between an I/O devices and the CPU, the less time the CPU will spend waiting for information to process. This in turns means that we should strive to have the data bus as wide as possible.
- An important component not shown in Figure 8.1 is the CPU clock.
- The clock controls the rate at which activities are carried out by the CPU.
- It generate a stream of cycles or ticks and an action can only be carried out on the occurrence of a clock tick.
- Obviously, the more cycles per second, the more actions that the CPU can carry out.
- The speed of the clock is measured in millions of cycles per second.
- One cycle per second is one Hertz (Hz), a kilohertz (KHz) is 1000Hz, a megahertz (MHz) is 1000 KHz and a gigahertz is 1000 MHz. Currently, PCs are being marketed with clock rates range from 2 to 4 GHz and the rate continues to increase.

Bus System

- The processor must be able to communicate with all devices.
- They are connected together by a communications channel called a bus.
- A bus is composed of a set of communication lines or wires.
- A simple bus configuration is shown Figure 8.2.
- We refer to this bus as the system bus as it connects the various components in a computer system.

- ⇨ Internally, the CPU has a CPU bus for transferring information between its components (e.g. the control unit, the ALU and the registers).

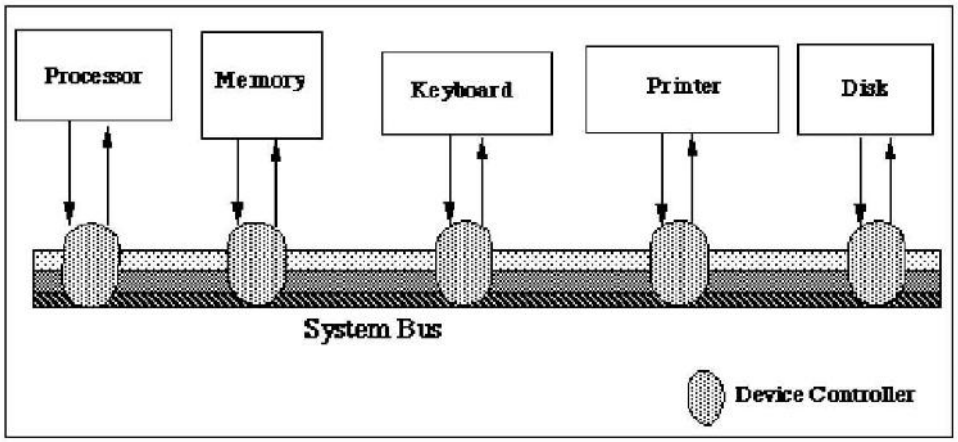

Figure 8.2: The system bus: the processor communicates with all devices via the system bus

- ⇨ Information is transferred from one device to another on the bus.
- ⇨ For example, information keyed in at the keyboard is passed along the bus to the processor.
- ⇨ The processor executes programs made up of instructions, which are stored in the computer"s memory.
- ⇨ These instructions are transferred to the processor using the bus.
- ⇨ As indicated in Figure 8.2, the lines of the bus may be classified into 3 groups.
- ⇨ One group of lines, the data lines, is used to carry the actual data along the bus from one device to another.
- ⇨ A second group of lines, the address lines, allow the CPU to specify where the data is going to or coming from i.e. which memory location is to be accessed or which I/O device is to be used.
- ⇨ The third group of lines, the control lines, carries control signals that allow the CPU control the transfer of information along the bus.
- ⇨ For example, the CPU must be able to indicate whether information is to be transferred from memory or to memory; it must be able to signal when to start the transfer and so on. We will refer to these groups of lines as separate buses, so we refer to the data bus, address bus and control bus as separate entities.

- It is important to realise that a computer system may have a number of separate bus systems so that information can be transferred between more than one pair of components at the same time.
- For example, it is common to have one bus for communicating between memory and the CPU at high speeds.
- This bus is called a CPU-memory bus.
- In addition, this bus would be connected to a second I/O bus via a bus adapter, as illustrated in Figure 8.3.
- This second bus would be used for the slower I/O devices.

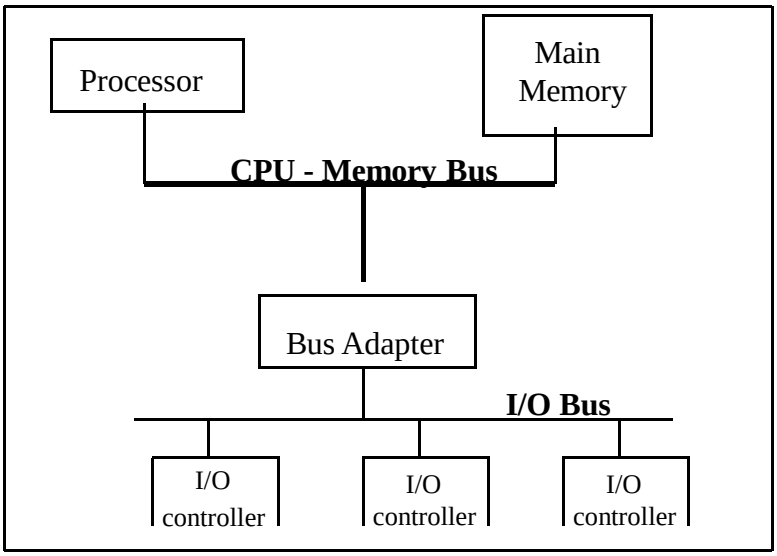

Figure 8.3: CPU-memory bus and I/O bus

- This allows the processor more efficient access to memory, as the CPU-memory bus can operate at very high speeds.
- These high speeds are only possible, if the physical bus length is quite short.
- Thus, by providing a second I/O bus to accommodate the various I/O devices that may be connected to the computer, the length of the CPU-memory bus can be kept shorter than it would be if the I/O devices were to be directly attached to a single system bus.
- On the other hand, to keep the cost of a computer system low, a single bus running at a slower speed, may be used to connect all devices to the CPU.
- In order to attach any device to a computer, it must be connected to the computer's bus system.

- ⇨ This means that we need a unit that connects the device to the bus.
- ⇨ The terms device controller and device interface are used to refer to such a unit.
- ⇨ So, for example, a disk controller would be used to connect a disk drive to the system bus and the term I/O controller refers to the controller for any I/O device to be connected to the bus system.
- ⇨ A computer system will have some standard interfaces such as a serial interface, which can be used with a number of different I/O devices.
- ⇨ The serial interface, for example, can be used to attach a printer, a mouse or a modem (device for communications over a telephone line) to the computer.

8.4 PCI (PERIPHERAL COMPONENT INTERFACE) BUS

- ⇨ The PCI bus was developed in the early 1990‟s by a group of companies with a goal to advance the interface allowing OEM‟s or users to upgrade the I/O (Input-Output) of personal computers.
- ⇨ The PCI bus has proven a huge success and has been adopted in almost every PC and Server since.
- ⇨ The latest advancement of the PCI bus is PCI-X. PCI-X is a 64-bit parallel interface that runs at 133 MHz enabling 1GB/s (8 GB/s) of bandwidth.
- ⇨ Though other advancements are in the works, including DDR, or the PCI bus, they are perceived as falling short.
- ⇨ They are too expensive (too many pins in the 64-bit versions) for the PC industry to implement in the mass volumes of PCs and that they do not offer sufficient bandwidth and advanced feature set required for the servers of the future.
- ⇨ Many would argue that there is no need to advance the bandwidth of PCI on PCs since few I/O cards are taxing the 250 to 500 MB/s bandwidth that is currently available. This is not the case for Servers.
- ⇨ High performance servers are frequently equipped with clustering, communication and storage I/O cards that together tax the bandwidth of the PCI-X bus.

Thank You