Q.1: Draw the internal architecture of 8086µp and discuss the operation of EU and BIU.

Architecture of 8086:

To improve the performance by implementing the parallel processing concept the CPU of the 8086 is divided into two sections. They are Execution Unit (EU) and Bus Interface Unit (BIU).

The Execution Unit:

The Execution Unit consists of a control system, a 16-bit ALU, 16-bit Flag register and four general purpose registers (AX, BX, CX, DX), pointer registers (SP, BP) and index registers (SI, DI) of each 16-bits.

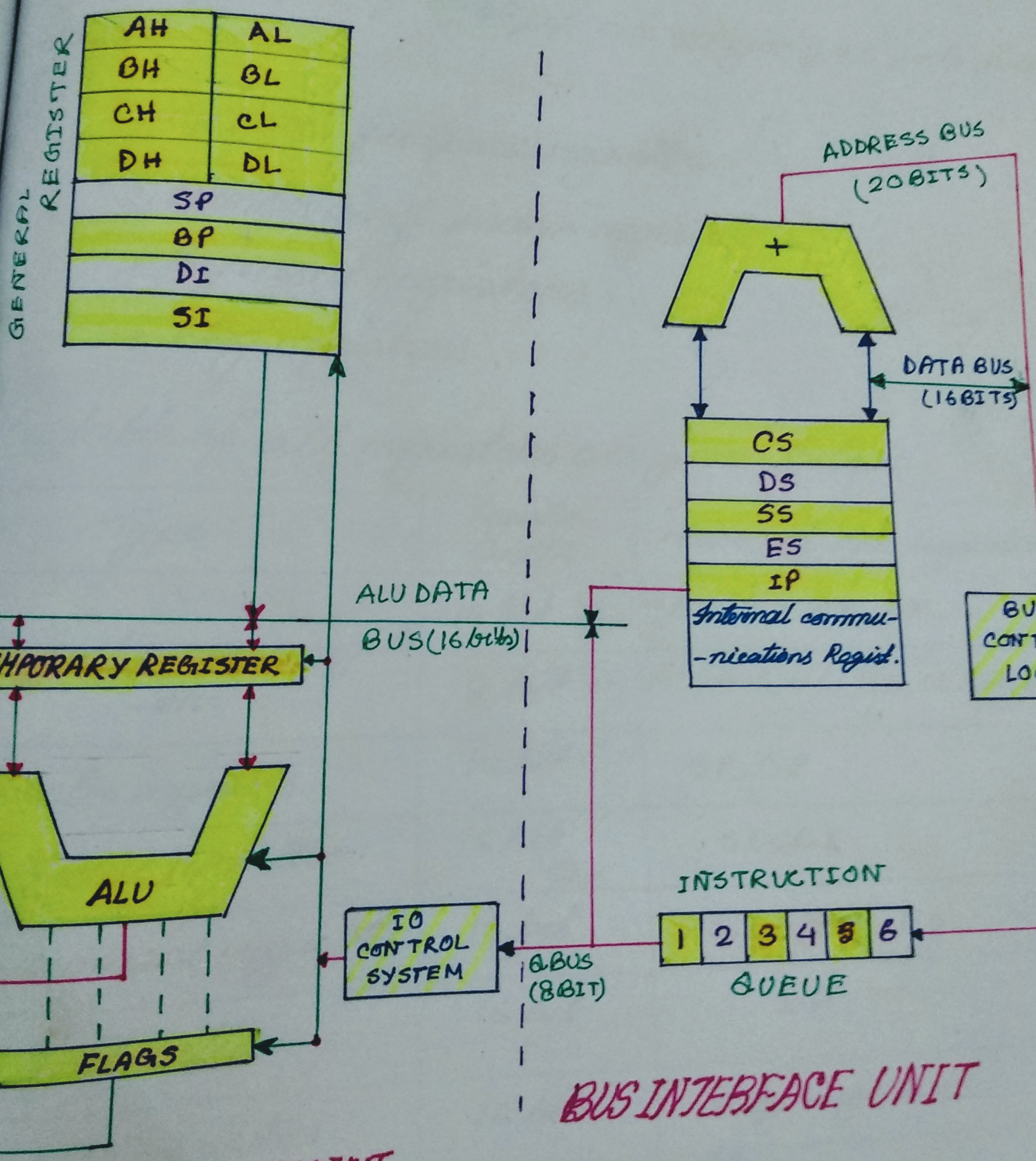
The control circuitry controls the internal operations. The decoder in the EU decodes the instructions fetched from the memory into a series of actions. The ALU can add, subtract, perform operations like logical AND, OR, NOR, increment, decrement, complement and shifting the binary numbers.

The BUS INTERFACE UNIT:

The BIU consists of a 6-byte long instruction register called Queue and four stack segment registers

(ES, CS, SS, DS), one instruction pointer (IP) and an adder circuit to calculate the 20 bit physical address of a location. It will perform all the external bus operations. They are fetching the instructions from the memory, read/write data from/into memory or port and also supporting the instruction queue etc. The BIU fetches up to six instruction bytes from the memory and stores these pre-fetched bytes in a first-in-first out register set called queue. This will increase the overall speed of microprocessor. Fetching the next instruction while the current instruction executes is called pipelining or parallel processing.

8086 ARCHITECTURE



EXECUTION UNIT

BUS INTERFACE UNIT

Q.2: Discuss the register structure of 8086 µp.

Register Structure:

The 14 registers of 8086 µp are categorized into four groups:

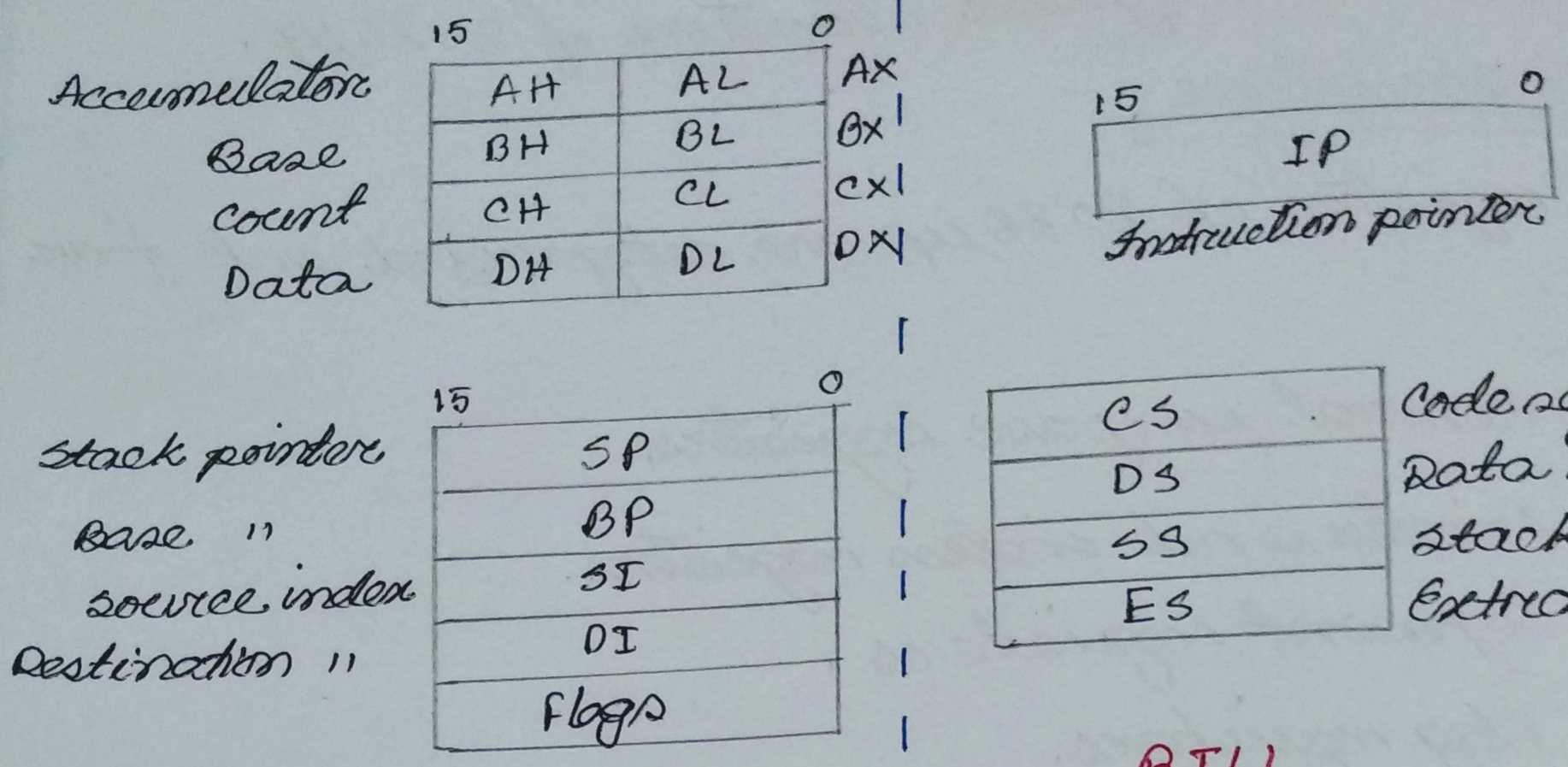
- General purpose registers.
- Pointers and index registers.
- Segment registers.
- Flag registers.

The table of all registers are given below:

No	Type	Register width	Name of the registers
1	General purpose registers	16 bit	AX, BX, CX, DX.
		8 bit	AL, AH, BL, BH, CL, CH, DL, DH.
2	Pointer registers	16 bit	SP, BP
3	Index registers	16 bit	SI, DI
4	Segment registers	16 bit	CS, DS, SS, ES.
5	Instruction	16 bit	IP
6	Flag registers	16 bit	Flag registers

The explaination of register structure are given below

There flag registers described briefly:



General purpose registers:

There are ~~for~~ 4, 16-bit general purpose registers namely AX, BX, CX and DX which are part of execution unit. These registers can be used individually for storing 16-bit data temporarily. The AL register is also called the ~~accumd~~ accumulator. The register also namely (AH, AL); (BH, BL); (CH, CL); (DA, DL). The pairs of registers can be used together to store 16-bit data words. Here L indicates the lower byte and H indicates the higher byte. X indicates the extended register. The general purpose registers are used for data manipulations.

Pointer and index register:

The other 4 registers of EU are referred to as index/ pointer registers. They are stack Pointer register, Base pointer register, source index register and destination index registers. The pointer registers contain the offset within a particular segment.

The BP and SP registers hold the offsets within the data and stack segment respectively. The index registers are used as general purpose registers as well as for holding the offset in case of indexed based and relative indexed addressing modes. The source index register is generally used to store the offset of source data in data segment. These index registers are specifically used in string manipulations.

Segment registers:

There are four segment registers in BIU. They are 16-bit registers. They are namely,

- Code segment (CS)
- Data segment (DS)
- Extra segment (ES).
- Stack segment (SS).

The code segment register is used for addressing

for addressing the 64KB memory location in the code segment of the memory.

The DS registers point to the data segment of the 64KB memory where the data is stored.

The ES register also refers to essentially another data segment of the memory space.

The SS register is useful for addressing stack segment of memory. So, the CS, DS, SS and ES segment registers respectively contains the segment address for the code, data, stack and extra segments of the memory.

Instruction Pointer Register:

It is a 16-bit register which ~~always~~ always points to the next instruction to be executed within the currently executing code segment. So, this register contains the 16-bit offset address pointing to the next instruction code within the 64KB of the code segment area.

Q. 3: Explain the addressing modes of 8086 µp with example.

Addressing Modes:

The different ways in which a source operand is noted in an instruction are known as the addressing odes. There are 8 different addressing modes in 8086. ey are:

1. Register and immediate addressing mode:

A. Immediate addressing mode:

The addressing mode in which the data operand is part of the instruction itself is called Immediate ddressing mode. In immediate mode, 8 or 16-bit data n be specified as part of the instruction.

Example: MOV CX, 4847H
ADD AX, 2456H
MOV AL, FFH.

B. Register addressing mode:

This mode specifies the source operand, destination erand, or both to be contained in an 8086 register.

Example: MOV AX, BX.
ADD CX, DX.

2. Memory addressing mode:

A. Direct addressing mode:

The addressing mode in which the effective address of the memory location at which the data operand is stored is given in the instruction i.e. the effective address is just a 16-bit number is written directly in the instruction.

Example: MOV BX, [1354H];

MOV BL, [0400H];

B. Register indirect addressing mode:

In this mode, the EA is specified in either a pointer register or an index register. The pointer registers are

- ~~BX (Base register)~~ . SP
- BP.

The index registers are:

- SI
- DI

Example: MOV [DI], BX.

MOV AX, [BX]

C. Based addressing mode:

The offset address of the operand is given by the sum of contents of the BX or BP registers and an 8-bit or 16 bit displacement.

example: MOV DX, [BX+04].
ADD CL, [BX+08].

D. Indexed addressing mode:

In this mode the EA is calculated by adding the signed 16-bit or sign extended 8-bit displacement and the contents of SI or DI.

Example: ~~ADD CX, E~~
ADD BX, [SI+06]
ADD AL, [DI+08]

E. Based-index addressing mode:

In this mode, the EA is computed by adding a base register (BX, BP), an index register (SI, DI), and a displacement (unsigned 16-bit or sign-extended 8-bit).

Example: ADD CX, [BX+SI]
MOV AX, [BX+DI]

F. String addressing mode:

This mode uses index registers. The string instructions automatically assume SI points to the first byte or word of the source operand and DI points to the first byte or word of the destination operand.

Example: MOVS BYTE.

3. Input/Output addressing mode:

Standard I/O uses port addressing modes. For memory-mapped I/O, memory addressing modes are used. It has two types.

A. Direct addressing mode:

In direct port mode, the port number is an 8-bit immediate operand.

Example: OUT05H, AL.

B. Indirect addressing mode:

In indirect port mode, the port number is taken from DX allowing 64K 8-bit ports or 32K 16-bit ports.

Example: If $[DX] = 5040_{16}$, then IN AL, DX inputs the 8-bit content of port 5040_{16} into AL.

4. Relative addressing mode:

Instructions using this mode specify the operand as a signed 8-bit displacement relative to PC.

Example: JNC START.

5. Implied addressing mode:

Instructions using this mode have no operands.

Example: CLC which clears the carry flag to zero.

Q.4: Explain
of 8086 up.

8086 Instruct

The 8086 has
out 300 op co
rand, single
Examples of
the followi

1. Data tran

This instruc
urce operan
rand in

Example:

(a) MOV C

(b) PUSH

(c) POP

2. Input

This inst
put ore o

Example

(a) IN A

(b) OUT

Q.4: Explain the different kinds of instruction sets of 8086 µp.

8086 Instruction Set:

The 8086 has approximately 117 different instructions with about 300 op codes. The 8086 instruction set contains no operand, single operand, and two operand instructions. Examples of some of the 8086 instructions are given the following:

1. Data transfer instructions:

This instructions are used to transfer data from a source operand to a destination operand. The source operand in most of the cases remains unchanged.

Example:

(a) MOV CX, DX ; DX copies the 16-bit content of DX into CX.

(b) PUSH START[BX].

(c) POP AX.

2. Input/output instructions:

This instructions are used to ~~make~~ take data as input or output in registers.

Example:

(a) IN AX, DX.

(b) OUT DX, AL.

3. Address initialization instructions:

Example:

(a) LEA BX, 5000H and MOV BX, 5000H accomplish the same task.

(b) LDS SI, [BX].

4. Flag register instructions:

(a) PUSHF pushes the 16-bit flag register onto the stack.

(b) LAHF Loads AH with the condition codes from the low byte of the flag register.

5. Arithmatic instructions:

(a.) ADD AL, 74H.

(b) ADD AX, BX.

(c) INC BX.

(d) SUB AL, CH.

6. Logical, shift and rotate instructions:

(a) TEST BL, 3.

(b) SHL DX, 1 .

7. String instructions:

(a) LODS BYTE.

(b) LODS WORD.

8. Unconditional transfers instructions:

(a) JMP FAR BEGIN.

(b) CALL DWORD PTR[BX].

9. Conditional transfer instructions:

(a) JC : Jump if carry flag set.

(b) JS : Jump if sign flag set.

10. Loop instructions:

(a) LOOP NEXT.

(b) LOOP BEGIN.

11. Interrupt instructions:

(a) INT n is a software interrupt instruction.

(b) IRET pops IP, CS and flags from the stack.

12. Processor or control instructions:

(a) STD sets direction flag to one.

(b) CLI clears the IF flag.

Q.6: Discuss about memory interfacing of 8086µp.

Memory interfacing of 8086µp:

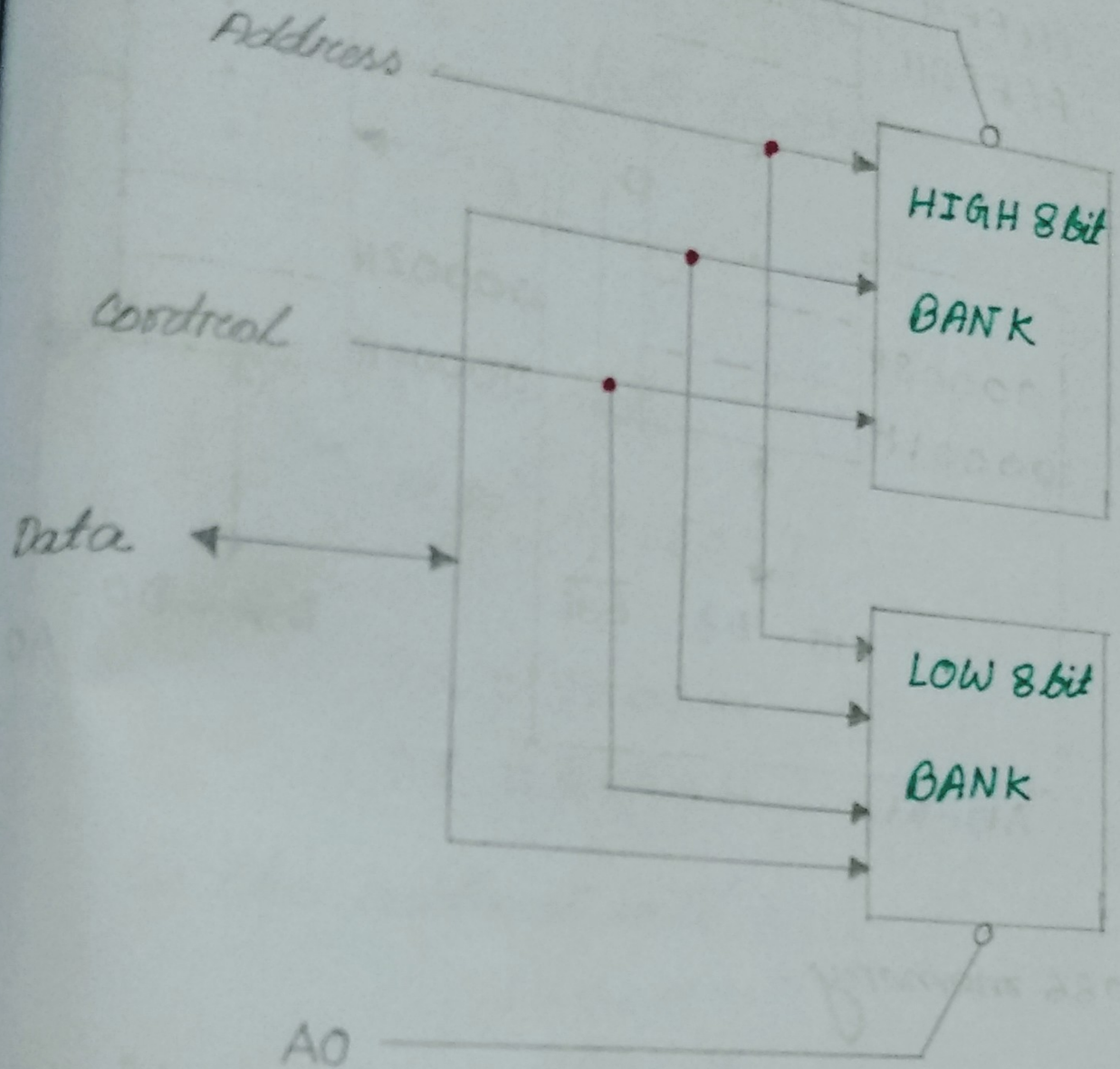


Fig: 8086 memory array.

Figure shows a general block diagram of an 8086 memory array. In figure, the 16-bit word memory is partitioned into high and low 8-bit banks on the upper and lower halves of the data bus selected by $\overline{BHE}$ and A0. Here the two banks space is 512KB for each bank. These banks can be selected by $\overline{BHE}$ and A0 as follows:

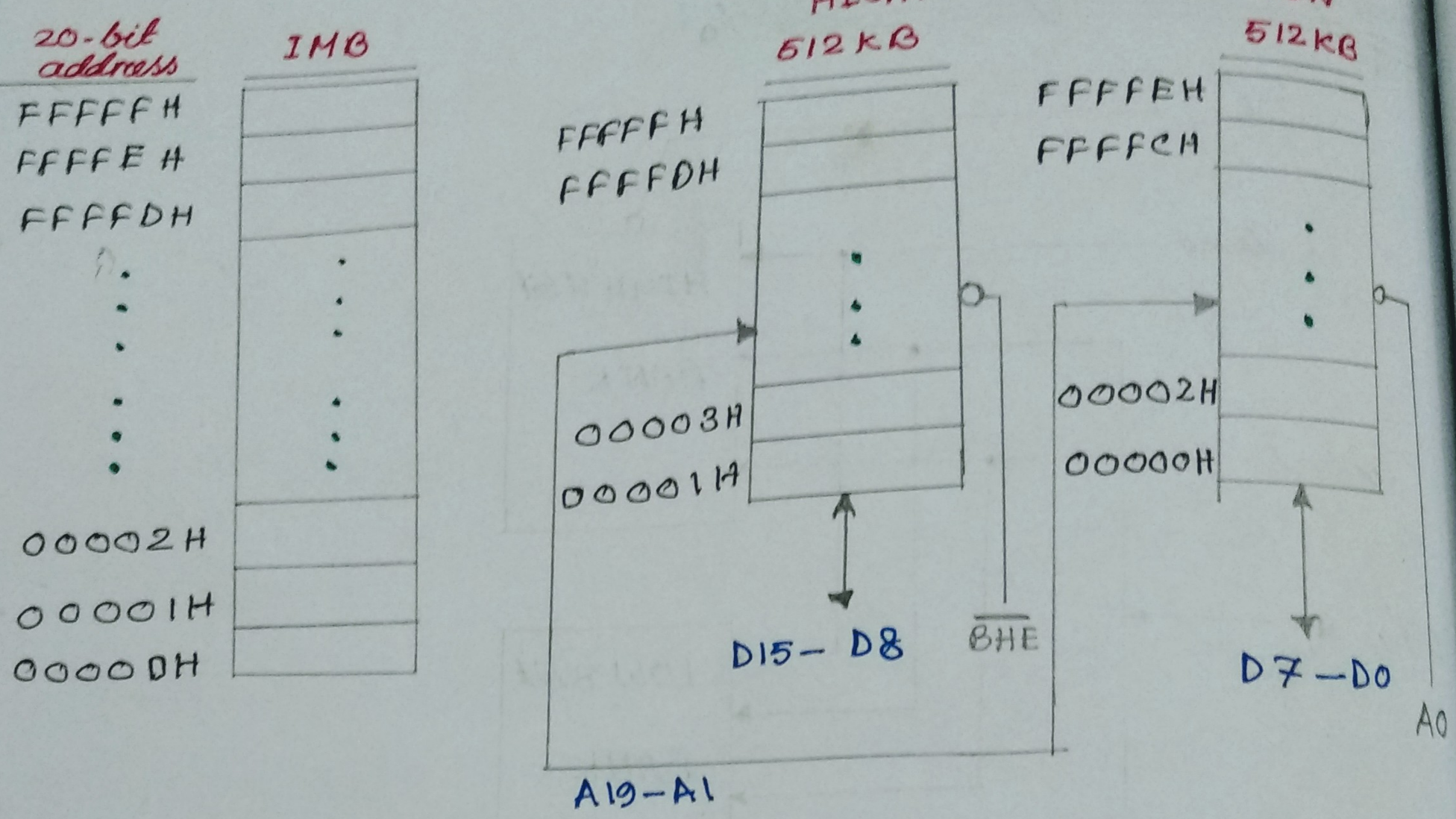


Fig: 8086 memory.

One bank is connected to D7–D0 and contains all even addressed bytes (A0=0). The other bank is connected to D15–D8 and contains odd-addressed bytes (A0=1). A particular byte in each bank is addressed by A19–A1. The even-addressed bank is enabled by LOW A0 and data bytes transferred over D7–D0 lines. The 8086 outputs HIGH on $\overline{BHE}$ and thus disables the odd-addressed bank. The 8086 LOW on $\overline{BHE}$ to select the odd-addressed bank and HIGH on A0 to disable the even-addressed bank. This directs the data transfer to the appropriate half of the

data bus. The table of $\overline{BHE}$ and A0 is:

$\overline{BHE}$	A0	
0	0	Byte transferred
0	1	Both bytes
1	0	Byte to/from odd address via $D_{15}-D_8$
1	1	Byte to/from even address via D_7-D_0
		None

Example: Suppose,

MOV AX, [SI]

where,

DS = 1200H

SI = 0044H.

So the address will,

12000H
0044H
12044H, which is even. So the data will go to LOW bank. And if the address will 12045H then the data will go HIGH bank. For HIGH bank that time the $\overline{BHE}=0$ and A0 = 1. and for low bank $\overline{BHE}=1$ and A0 = 0.

Q. 7: Discuss the function of FLAG registers bit.

FLAG registers of 8086μp:

Flag register in EU is of 16-bit and is shown in fig:

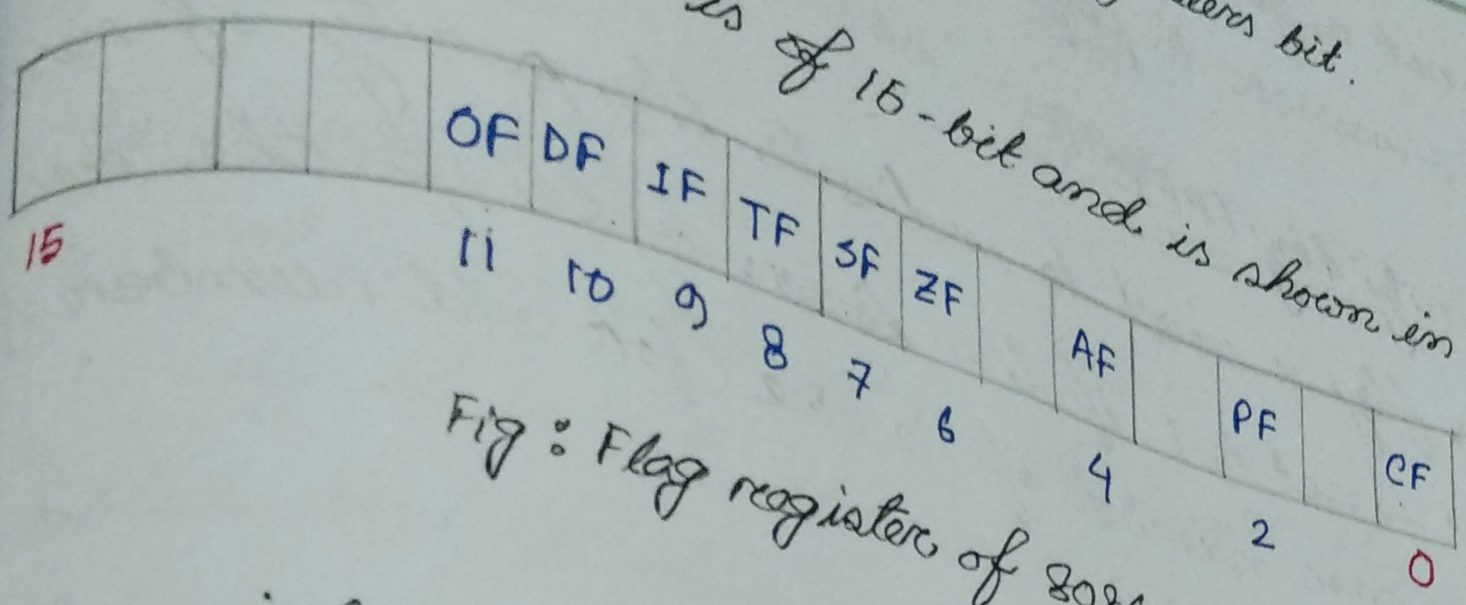


Fig: Flag register of 8086μp.

Flag register determines the current state of the processor. 8086μp has 9 flags and they are divided into 2 categories:

1. Conditional flag:

Conditional flags represent result of last arithmatic or logical instruction executed. They are:

a. Carry Flag (CF):

The flag which indicates an overflow condition for unsigned integer arithmatic is called CF. It is also used in multiple precision arithmatic.

b. Auxiliary Flag (AF):

If an operation performed in ALU generates a carry from lower nibble (i.e $D_0 - D_3$) to upper nibble ($D_4 - D_7$), flag is set i.e. carry given by D_3 to D_4 is AF flag.

This is not a general-purpose flag, it is used internally by the processor to perform Binary to BCD conversion.

C. Parity Flag (PF):

This flag is used to indicate the parity of result the flag is set to 1, if the lower byte of the result contains even number of 1's else (for odd number of 1's) set to zero.

D. Zero Flag (ZF):

It is set; if the result of arithmetic operation is zero else it is reset.

E. Sign Flag (SF):

This flag is set, when the result of any computation is negative.

F. Overflow Flag (OF):

It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine.

2. Control Flags:

control flags are set or reset deliberately to control the operations of the execution unit. control flags are

A. Trap flag (TP):

- It is used to single step control.
- When trap flag is set, program can be run in single step mode.

B. Interrupt Flag (IF):

- It is an interrupt enable/disable flag.
- If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.

C. Direction Flag (DF):

- It is used in string operation.
- If it is set, the string bytes are accessed from higher memory address to lower memory address.

Q.8: List the 8086 minimum and maximum mode signals. How are these modes selected?

MINIMUM AND MAXIMUM MODE:

The 8086 can be configured in either minimum or maximum mode using the MN/$\overline{MX}$ input pin. The signals of minimum modes are:

- DT/$\overline{R}$ (Data transmit/Receive):

It signals the direction of data transfer over the bus. Logic 1 indicates that the bus is in the transmit mode. Logic 0 signals that the bus is in the receive mode.

- $\overline{DEN}$ (Data enable):

It is provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver.

- $\overline{DEN}$:

It is active LOW during each memory and I/O access and for $\overline{INTA}$ cycles.

- ALE (Address Latch Enable):

It is an output signal provided by the 8086 and can be used to demultiplexed the AD0-AD15 into A0-A15 and D0-D15 at the falling edge of ALE. The 8086 ALE signal is similar to the 8085 ALE.

- M/$\overline{IO}$ (Memory/IO):

It tells external circuitry whether a memory or I/O

transfer is taking place over the bus. Logic 1 signals a memory operation and logic 0 signals an I/O operation.

- $\overline{WR}$ (write):

It is switched to logic 0 to signal external devices that valid output data are on the bus.

- INTA (Interrupt Acknowledge):

For interrupt Acknowledge cycles, the 8086 outputs LOW on this pin.

- HOLD (Input), HLDA (output):

A HIGH on the HOLD pin indicates that another master is requesting to take over the system bus. The processor receiving the HOLD request will output HLDA high as an acknowledgment. At the same time the processor tristates the system bus. Upon receipt of LOW on the HOLD pin, the processor places LOW on the HLDA pin.

- CLK (Input):

It provides the basic timing for the 8086.

In the maximum mode, some of the 8086 pi
in the minimum mode are redefined. For exampl
pins HOLD, HLDA, $\overline{WR}$, M/$\overline{IO}$, DT/$\overline{R}$, $\overline{DEN}$, ALE and $\overline{INTA}$

the minimum mode are redefined as $\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$, $\overline{LOCK}$, $\overline{S2}$, $\overline{S1}$, $\overline{S0}$, QS0 and QS1 respectively.

These signals describe in below:

- $\overline{S0}, \overline{S1}, \overline{S2}$:

In maximum mode, the 8288 bus controller decodes the status information from $\overline{S0}, \overline{S1}, \overline{S2}$ to generate bus timing and control signals required for a bus cycle. they are decoded as follows:

$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	
0	0	0	Interrupt Acknowledge.
0	0	1	Read I/O port.
0	1	0	Write I/O port.
0	1	1	Halt.
1	0	0	Code access.
1	0	1	Read memory.
1	1	0	Write memory.
1	1	1	Inactive.

- $\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$:

These Request/Grant pins are used by other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. Each pin is bidirectional, with $\overline{RQ}/\overline{GT0}$ having higher priority than $\overline{RQ}/\overline{GT1}$. The $\overline{RQ}/\overline{GT0}$ or $\overline{RQ}/\overline{GT1}$ function of the 8086 works as follows:

→ A pulse from another local bus master indicates a local bus request to the 8086.

→ At the end of 8086 current bus cycle, a pulse from the 8086 to the requesting master indicates that the 8086 has relinquished the system bus and tristated the outputs. Then the new bus master subsequently relinquishes control of the system bus by sending a LOW on $\overline{RQ}/GT0$ or $\overline{RQ}/GT1$ pins. The 8086 then regains bus control.

• $\overline{LOCK}$:

The lock signal is meant to be output (Logic 0) whenever the processor wants to lock out the other processors from using the bus.

• QS1, QS0 :

The 2-bit queue status code QS0 and QS1 tells the external circuitry what type of information was removed from the queue during the previous clock cycle.

QS1	QS0	
0	0	No operation.
0	1	First byte of opcode from queue
1	0	empty the queue.
1	1	Subsequent byte from queue.

Q.9: What happens to the contents of AX, BX, DX registers after execution of the following 8086 instruction sequence:

(a) MOV AX, 51D
MOV BX, 10D
DIV BX.

(b) MOV AX, 5559D
MOV BX, 66D
MUL BX.

Ans: (a) Here AX is divided by BX and after the execution AX carries the quotient and DX carries the remainder. So the contents of these registers:

	H	L
AX	00	05
BX	00	0A
CX	00	00
DX	00	01

(b) Here AX is multiplied by BX and after the execution AX carries the result. But we know AX is a 16-bit register; where 5559D × 66D = 5902EH which is 20 bit. So the most significant 16 bits of this product will be in DX and remains are in AX. So the contents of these registors are showed in the given chart where we see the contents of AX, BX, CX and DX registors:

	H	L
AX	99	2E
BX	00	42
CX	00	00
DX	00	05

Q.10: If $[CS] = 456A_{16}$ and $[IP] = 1620_{16}$ then compute the 20 bit physical address.

Ans: Here,

The physical address,

$$\begin{array}{r} 456A0H \\ 1620H \\ \hline 46CC0H. \end{array}$$

So the physical address (20bit) = 46CC0H.

Ans: 46CC0H.

Q.12: Draw and explain the control word format of 8255 PPI.

control word:

The control word format of 8255PPI are described below:

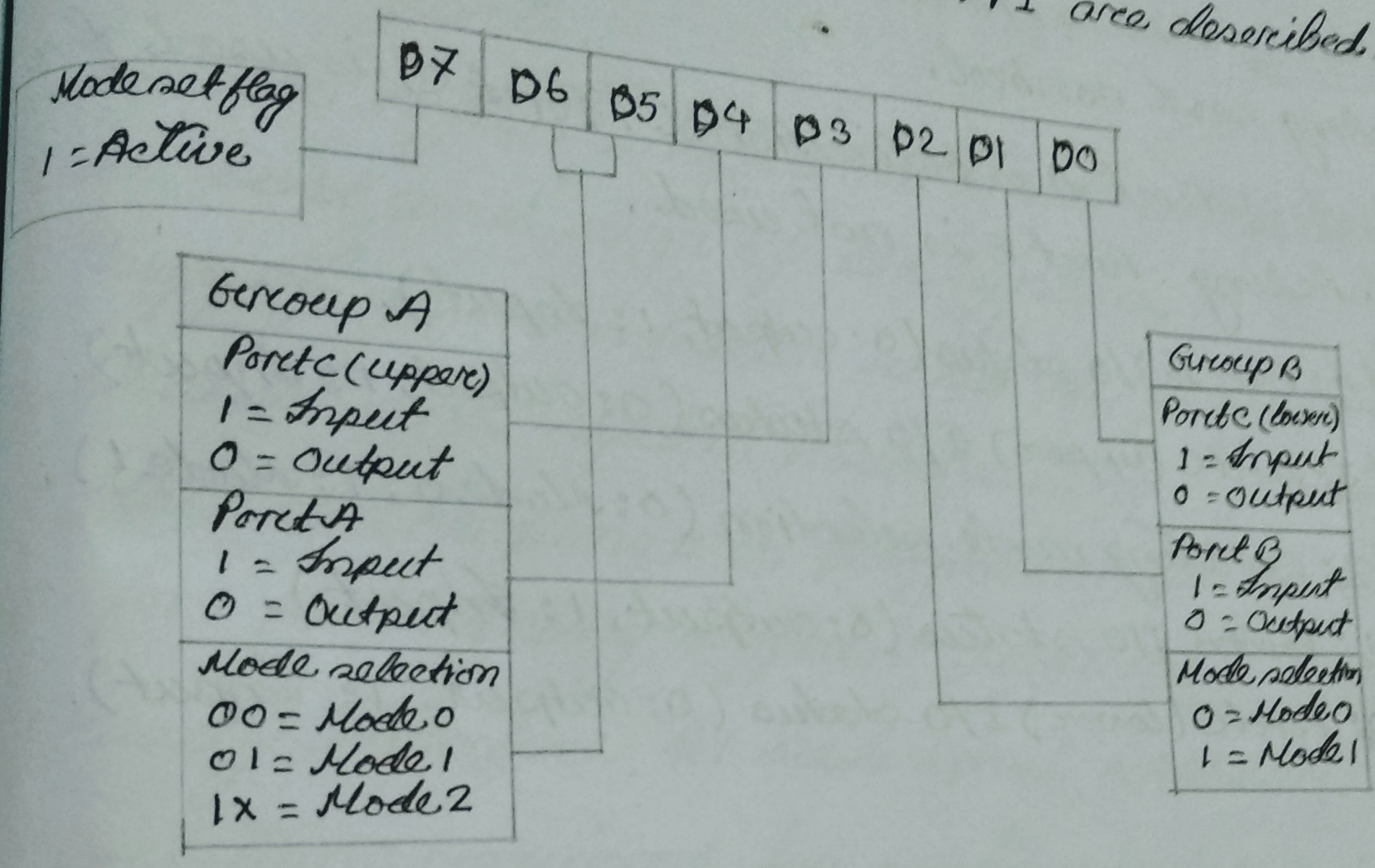


Fig: control word of 8255 PPI

When bit 7 is set, the command byte operates in the following way:

Bit 7: Control byte function (1 = active).

Bit 5 to 6: operating mode,

Bit 6	Bit 5	Mode
0	0	Mode 0
0	1	Mode 1
0	X	Mode 2
1		

Mode 0: Ports A and B opperate as inputs ore outputs.
Port C is divided into two 4-bit groups, which can be operated as inputs ore outputs.

Mode 1: Same as Mode 0 but Port C is used fore hand-shaking and control.

Mode 2: Port A is bidirectional. Port C is used for handshaking. Port B is not used.

Bit 4: Port A I/O status (0: output, 1: Input).

Bit 3: Port C (upper) I/O status (0: output, 1: Input).

Bit 2: operating mode selection (0: Mode 0, 1: Mode 1).

Bit 1: Port B I/O status (0: output, 1: Input).

Bit 0: Port C (lower) I/O status (0: output, 1: Input).

Q.17: Write down the advantages of Microcontroller.

Advantages of Microcontroller:

- Microcontroller acts as a microcomputer without any digital parts.
- As the higher integration inside microcontroller reduce cost and size of the system.
- Using of microcontroller is simple easy for troubleshoot and system maintaining.
- Most of the pins are programmable by the user for performing different functions.
- Easily interface RI additional RAM, ROM, I/O ports.
- Low time required for performing operations.

For displaying 'MEC'.

```
org 100h

MSG DB 'MEC $'

MOV AX,@DATA
MOV DS,AX

LEA DX,MSG

MOV AH,9
INT 21H

MOV AH,4CH
INT 21H

Ret
```

Adding two decimals and display result in decimal.

```
ORG 100H

MOV AX,500D
MOV BX,180D

ADD AX,BX

MOV CX,0
MOV BX,10D

LOOP1:
  MOV DX,0
  DIV BX
  PUSH DX
  INC CX
  CMP AX,0
  JNZ LOOP1

MOV AH,2

LOOP2:
  POP DX
  ADD DL,48D
  INT 21H
  DEC CX
  JNZ LOOP2

ret
```

Converting lower case to upper case.

```
org 100h

MOV AH,1
INT 21H

MOV DL,AL
SUB DL,20H

MOV AH,2
INT 21H

ret
```

a = 61H
A = 41H

Converting upper case to lower case.

```
org 100h

MOV AH,1
INT 21H

MOV DL,AL
ADD DL,20H

MOV AH,2
INT 21H

ret
```

Multiplication of two numbers.

```
org 100h

MOV AX,51D
MOV BX,10D
MUL BX

ret
```

Division of two numbers.

```
org 100h

MOV AX,51D
MOV BX,10D
DIV BX

ret
```

SOEB.EEE.MEC