20-09-08

16 bit microprocessor 8086

Q.1 List the features of clk generator IC 8284

ans i) It generates clk READY & RESET signal for up

2) clk frequency is 1/3rd of Crystal frequency

3) clk frequency can be generated by using external function / freq generator

4) It provides the clk frequency half of the clk o/p signal at PCLK pin

5) It provides Synchronisation signal (SYNC) which Synchronises multiple 8284 IC's in a system.

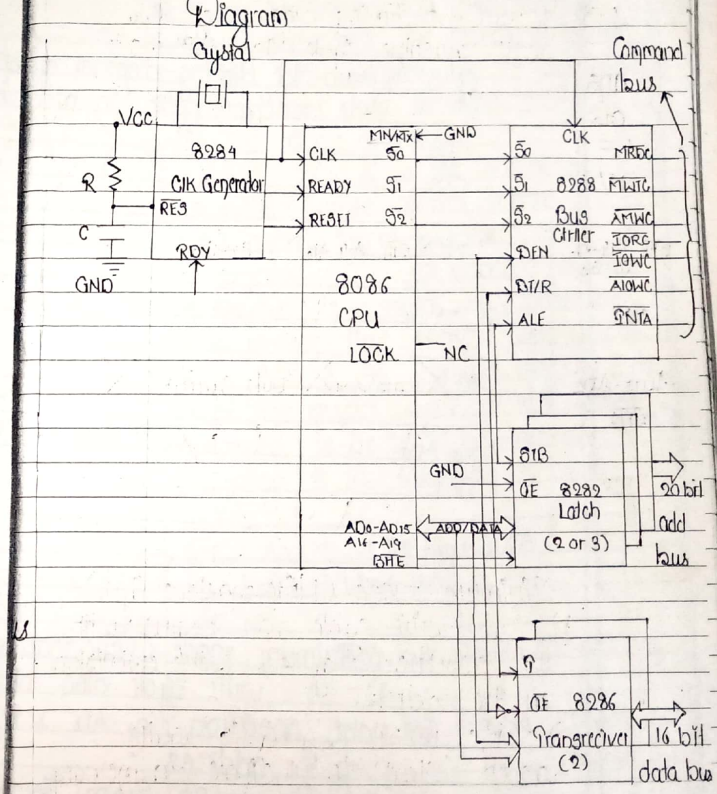
Q.2 Show interfacing of 8086 in maxm mode & explain

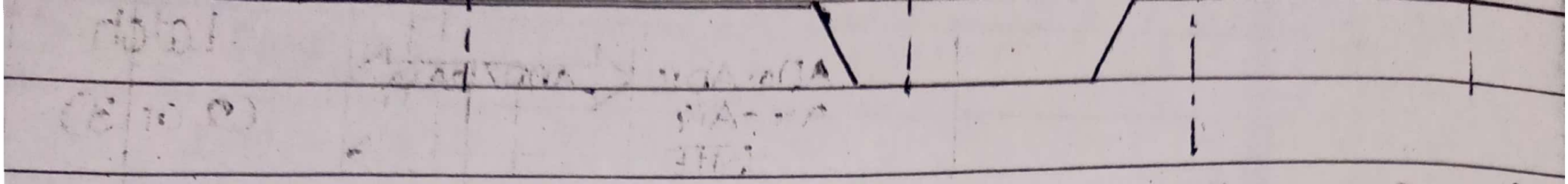
ans • The fig below shows the diagram of the 8086 sys in maxm mode

• Note the use of the bus controller IC 8288 along with the other support chips.

- Additional circuitry is required to translate the control signal. the additional circuitry converts the status signals ($\overline{S_2}$-$\overline{S_8}$) to I/O & memory transfer signals. The intel 8288 bus controller is used for this purpose. It generates the control signal req to control the data flow & for controlling latches 8282 & transceiver
- It generates the signal MRDC, MWTC, AMWC, AIOWC, IOWC signal.
- The MRDC & MWTC are memory read & memory write command signal they instruct the memory to accept or send data on data bus
- The IORC & IOWC are I/O read command & write command. the instruct the I/O device to read or write data to & from address port on data bus
- The AIOWC & AMWTC are advanced I/O write command & adv memory write command signal. they are similar IOWC & MWTC except they are activated one clock signal earlier to IOWC & MWTC signal

Diagram





Minimum mode (Write machine Cycle)

1) - A write cycle also begins with the assertion of ALE & the emission of the address

2) The M/IO signal is again asserted (Stay) to indicate a memory or I/O write operation

3) In the T_2 immediately following the address emmission the processor emmits the data to be written into the addressed location. This data remains valid untill the middle of T_4

4) During T_2, T_3 & T_W the processor asserts the writes control signal. The ($\overline{WR}$) asserts signal becomes active at the begning of T_2 as opposed to the read which is delayed somewhat into T_2 to provide time for the bus to float

5) The $\overline{BHE}$ & A_0 signals are used to select the proper byte of the memory T_0 word to be read or written according to the following

$\overline{BHE}$	A_0	CHARACTERISTICS
0	0	Whole word
0	1	Upper byte from/to odd addr
1	0	lower byte from/to even addr
1	1	None

Q 4. Draw the register structure of 8086 & explain func of each register

Ans

AH	AL
BH	BL
CH	CL
DH	DL
SP	
BP	
SI	
DI	

ES
CS
SS
DS
IP

Flag Register

The register of 8086 are divided into three groups according to their func

1) Data group:

The data group consist of 4 16-bit general purpose registers Ax, Bx, Cx & Dx. These registers can be used to store both operand & results & each of them can be access as a whole 16-bit register or two 8-bit register Seperately as follows

16-bit	8-bit high order	8-bit lower order
Ax	AH	AL
Bx	BH	BL
Cx	CH	CL
Dx	DH	DL

2) Pointer group:

The following 5 register are in a group of pointer & index register

IP (Instruction pointer)
SP (Stack pointer)
BP (Base pointer)
SI (Source index)
DI (Destination index)

i) The register IP & SP are essentially the program counter & stack pointer ~~physical address~~ of complete instruction & stack address is formed by adding the contents of these registers to the contents of code segment or stack segment register resp.

2) Base pointer is the base register for accessing the stack & may be used with other registers or displacement is part of instruction

3) The SI & DI registers are for indexing These registers together with DS & ES are used for performing string operation The actual string address is computed by adding the content of SI & DS

The actual destination string address is computed by adding the content of DI & ES.

3) Segment group:
There are 4 Segment register in 8086
i) CS (Code Segment)
The func of CS reg is to store the starting or effective address for code seg

2) DS (Data Segment)
The func of DS reg is to store the starting address of Data seg

3) ES (Extra Segment)
The func of ES reg is to store the starting address of Extra Seg

4) SS (Stack Segment):
The func of SS reg is to store the starting address of stack segment

The flag register of 8086 is as follows

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X	X	X	X	OF	DF	IF	TF	SF	ZF	X	AF	X	PF	X	CF

1) Carry flag (CF):
An addition causes these flag to be set when there is a carry out of the MSB & Substraction causes it to be set if a borrow is needed

2) Parity flag (PF):
It is set to 1 if the lower order 8 bit of the result contains an even no of 1's otherwise it is cleared

3) Zero flag (ZF):
It is set to 1 if the result is zero other wise it is cleared

4) Sign flag (SF):
The Sign flag holds the arithmatic sign of the result after arithmatic or logical operation. if the sign flag = 1 the no is

considered as -ve no & if Sign flag = 0 the result is +ve. Left most bit of the no indicates Sign of no.

5) Trap flag (TF):
If trap flag is set it initilize the program execution in single stepping mode.

6) Interrupt flag (IF):
The interrupt flag controls the operation of INTR i/p pin, if IF = 0 the interrupt pin is disabled, if IF = 1 it is enabled.

7) Direction flag (DF):
Direction flag selects either incr/decr mode for the DI & SI register during string instruction, if DF = 0 the reg are automatically incremented & if DF = 1 it is decremented.

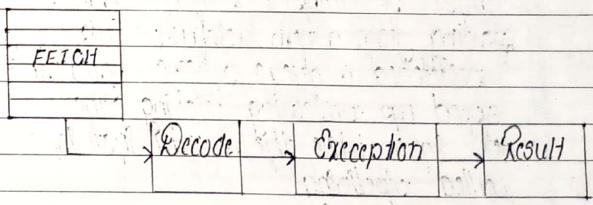
8) Over flow flag (OF):
Overflow indicates that the result has exceeded the capacity of the machine.

9) Auxiliary Carry flag (AF):
It is set if there is a carry out of bit D3 during an addition or borrow during Subtraction. So these flag is used exclusive for BCD arithmatic.

Q.5 Explain memory pointer, queue & Segmentation in 8086 µp?

ans Memory pointer: ans is same as the ans of pointer group.

Queue:



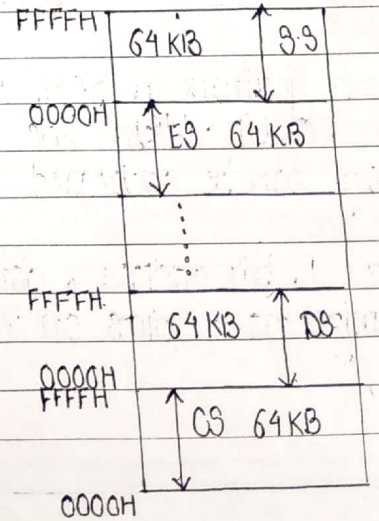
While the execution unit is decoding an instruct or executing an instruction which does not require use of buses the bus interface unit fetches upto

6 instruction bytes for the following instr. The bus interface unit stores this prefetched byte in a 1st in 1st out register set called a queue when the execution unit is ready for its nxt operation it simply reads the instr byte for the instr from the queue this is much faster than sending out an address to the sys memory & waiting for memory to send back the nxt instr byte or bit. Except in the case of jump & call instr where the queue must be dumped & then reloaded starting from a new address. This prefetched & queue scheme greatly speed up processing. fetching the next instr while the current instr is called pipelining.

Segmentation:

In 8086 the actual address line is 20 bit hence the physical memory available in 8086 is 1 MB but while programming we can use only 16 bit address bus hence by using this 16 bit address lines the max^m memory we can use is 2^{16} = 64 KB. hence to utilize total 1 MB of memory by using 16 address lines the total memory is divided into no of segments each of size 64 KB thus total 16 no of segments are available in 8086. this is called as segmentation. they are divided in 4 types as CS, DS, SS & ES.

Code seg holds the prog instr code, The data seg holds data for prog. The Extra seg is an extra data seg, the stack seg is used to store interrupt & sub routine return address



Q.6 Mention default seg base & offset pair reg. Mention the use of all seg base register ?

ans The default seg base & offset pair reg are given as follows

Segment	offset	Special Purpose
CS	IP	Instruction address
SS	SP or BP	Stack address
DS	Bx, SI & DI 8bit or 16 bit no	Data
ES	DI for string instr	String destination address

Q.7 List any 8 features of 8086 ?

ans 1) Provide 20 bit address line so 1 MB of memory can be addressed

2) Multiplex 16 bit address & data bus to minimize no of pins on IC

3) Operating clock freq are 5 MHz, 8 MHz & 10 MHz

4) Arithmetic operation can be performed on 8 bit or 16 bit Signed data including multiplication & division

5) Can operate in Single processor & multi processor config

6) The instr set is powerful, flexible & can be programmed in high level language like C

7) Provides 256 type of vectored s/w interr

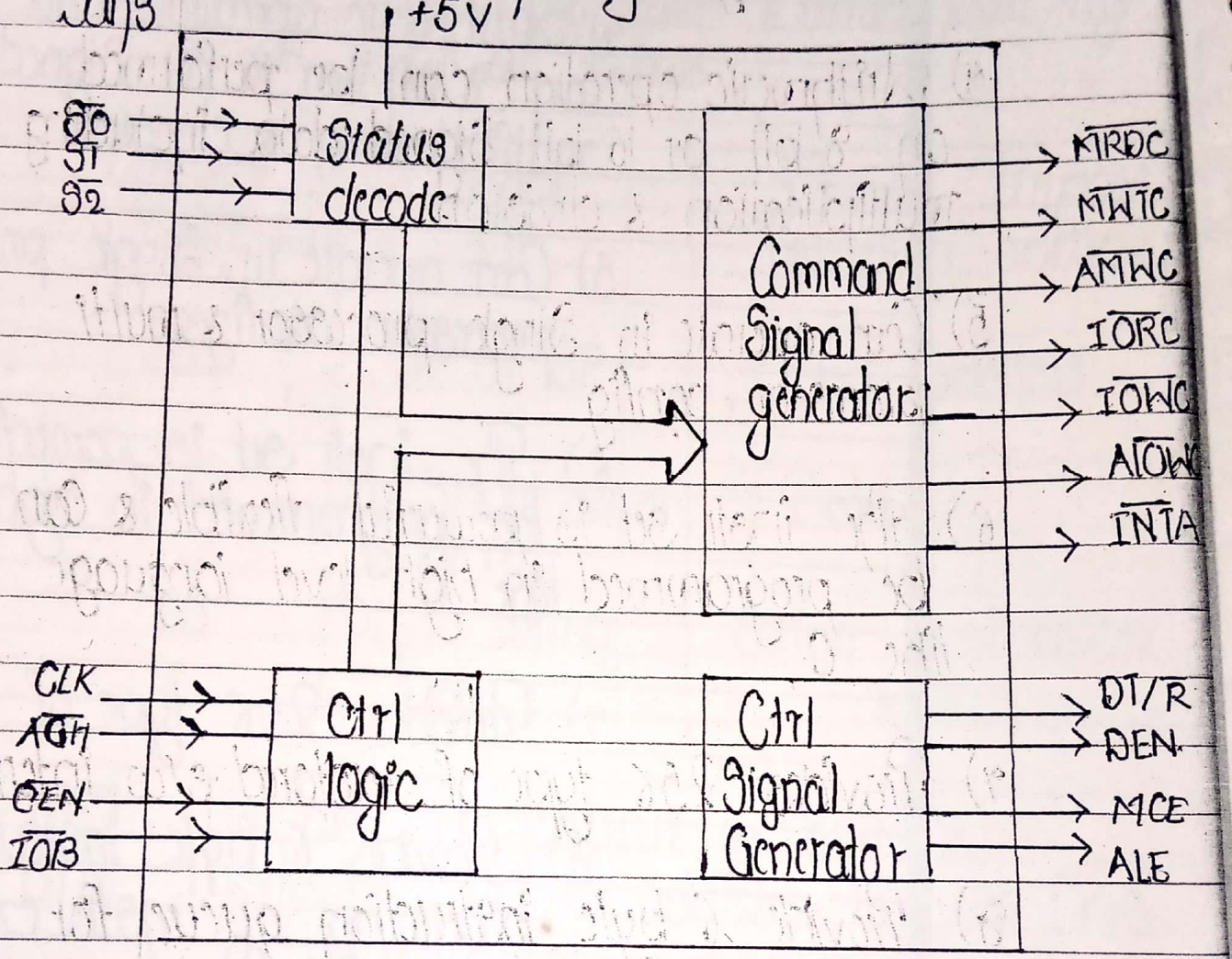
8) Provide 6 byte instruction queue for pipelining of instr execution

9) Operates in max^m & min^m mode

10) Supports multiprogramming

Q.8 Describe the function of bus controller 8288 with diagram.

Ans



The 8288 bus driller is a 20 pin bipolar IC it is used in the maxm mode config of 8086. The 8288 bus driller accepts the $\overline{S0}$, $\overline{S1}$ & $\overline{S2}$ o/p of 8086 & generates the command ctrl & timing

20-09-08

16 bit microprocessor 8086

Q.1 List the features of clk generator IC 8284

ans i) It generates clk READY & RESET signal for µp

2) clk frequency is 1/3rd of Crystal frequency

3) clk frequency can be generated by using external function / freq generator

4) It provides the clk frequency half of the clk o/p signal at PCLK pin

5) It provides Synchronisation signal (SYNC) which Synchronises multiple 8284 IC's in a system.

Q.2 Show interfacing of 8086 in maxm mode & explain

ans • The fig below shows the diagram of the 8086 sys in maxm mode

• Note the use of the bus controller IC 8288 along with the other support chips

signal at its o/p it also provides the bipolar bus drive capability & optimize the sys performance

i) Vcc :- +5V Supply

2) GND :- Ground

3) $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$:- Status i/p pins these pins are the status i/p pins from the 8086 processor the 8288 decodes these i/p's to generate command & ctrl signals at the appropriate time. when the pins are not in use they are all high

4) CLK :- This is a clk signal from 8284 clk generator & serves to establish when cmd & ctrl signals are generated

5) ALE :- (Address latch enable) This signal is active high & latching occurs on the falling transition. ALE is intended to use with transparent D type latches.

6) DEN:- (Data Enable) This signal serves to enable data transceiver on to either local or 16 data bus this signal is active high

7) DT/$\overline{R}$:- (Data transmit / receive) These signal establishes the direction of data flow through the transceiv & high on these line indicates transmit & low indicates receive.

8) $\overline{AEN}$ (address enable):- It enables cmd o/p of the 8288 bus driver at least 105 ns after it becomes active low. $\overline{AEN}$ going inactive immediately tristate the cmd o/p drivers. $\overline{AEN}$ does not affect the IO cmd lines if the 8288 is in the IO bus mode

9) CEN (Cmd enable):- When this signa is low all 8288 cmd o/p & the DEN & PDEN ctrl o/p are forced

Q.9 List importance of max^m mode of 8086 based sys

ans i) If MN/MX pin is connected to a logic 0, then the processor operates in the max^m mode

2) Max^m mode config is a bit complex & it is preferred for the multiprocessor environment, that means when more than one processor are being used simultaneously in same sys

3) In this mode pin 24 to 31 will have func describe by the mnemonics written next to the pins

4) For pins 26, 27 & 28 now works as $\overline{S0}$, $\overline{S1}$ & $\overline{S2}$ resply. They are ctrl bus signal

5) The ctrl signal in the encoded form is sent out by the 8086 in the maxm mode on these lines. the extendl bus ctrller decodes these signals to devlop ctrl signal req for the operation of two or more processor sharing the same bus

Q10 List significant diff betwn minm & maxm mode operation of 8086. describe under what situation maxm mode oper is useful?

ans i) The mode of oper is maxm mode minm mode can be ctrled by using pin 33 MN/$\overline{MX}$. The 8yo sys operates in the min mode by dyault pwr on

1) If MN/$\overline{MX}$ pin is grounded, the sys operates in max mode.
2) When only one CPU is to be used in a uP sys. It is used in min^m mode oper^n. In the min^m mode CPU issues ctrl signals that are req by the memory & I/O devices. For a uP sys the CPU operates in max^m mode. In max^m mode the ctrl signals are issued by the intel 8288 bus ctriller
3) The pin 24 to 31 have alternate fun when the CPU operates in max^m & min^m mode. The pins used in min^m mode are M/$\overline{IO}$, $\overline{INTA}$, ALE, HOLD, HLDA, DT/$\overline{R}$, DEN The pins used in max^m modes are $\overline{S2}$, $\overline{S1}$, $\overline{S0}$, $\overline{LOCK}$, QS, QS$_0$, RQ, $\overline{GT_0}$ $\overline{GT_1}$
4) Min^m mode oper^n is less expensive because all the memory & IO ctrl signals are generated by the uP itself. In max^m mode we req external bus ctrl like 8288 that generates the ctrl signal.

Min^m mode read cycle.

1) In min^m mode, the MN/MX pin is strapped to V_{CC}.

2) The read cycle begins in T1 with the assertion of the addr latch enable (ALE) signal. The trailing edge of this signal is used to latch the addr info which is valid on the local bus at this time into 8288 / 8283 latch.

3) The BHE & A0 signal addr the low, high or both bytes

4) From T1 to T4 the M/IO signal indicates a memory or IO operation. At T2 addr is removed from the local bus & the bus goes to a high impedance state

5) The read ctrl signal is also asserted at the T2 the (RD) signal causes the addressed device to enable it's data bus drive to the local bus.

6) Some time later valid data will be available on the bus & the addressed device will drive the READY & HIGH.

7) When the processor returns the read signal to a high level addressed device will again 3 state it's bus drive.

8) If a transceiver (8286 / 8287) is required to buffer the 8086 local bus, signal DT/R & DEN are provided by the 8086.

Q12 Define logical address & physical address. Explain the addr generation process in 8086 if DS = 3458H & SI = 13DCH, calculate physical addr

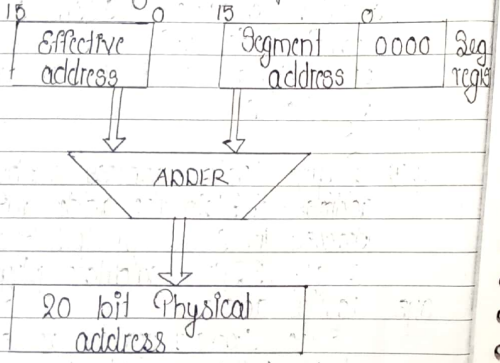
ans Logical address :-

The Segment register are used to hold 16 bit of starting address of a memory seg. this add is called as logical address

Physical address :-

The address outputted by bus interface unit is 20 bit & is calculated by inserting a zero at the end of offset address & adding it in logical address. The 20 bit address is called as physical address

Address generation process :



The segment register of 8086 act as a base register. They point out the starting address of various memory segment. The registers that can be used for addressing Bx, IP, SP, BP, SI, DI are only 16 bit wide & therefore effective address has only 16 bit. On other hand the address put on the address bus called the physical address that contains 20 bit. The extra 4 bit are obtained by adding the effective address to the content of one of the seg register

eg : if CS = 123AH & IP = 341BH then the nxt instr will be fetched from.

341B
123A0
157BB

DS = 3458H & SI = 13DCH

13DC
34580
36 01 E

34580
13DC
3595C ... 20 bit physical address

Q14 List four advantages of Segmented memo

ans i) Allow the memory capacity to be 1 MB even though the addresses associated with the individual instr are only 16 bit wide.

2) Allow the instr, data or stack portion of a prog to be more than 64 Kb long by using more than one code or stack segment

3) Facilitate the use of Seperate memor for prog, it's data & the stack

4) Permit a prog &/or it's data to be put into diff areas of memory, each time the program is executed

INSTRUCTION SET

Q.1 Select instruction for each of the following

i) Rotate register BL left 4 times

ans: ROL BL, 04H

2) Multiply AL by 05

ans: MOV BL, 05H
MUL BL

3) Signed division of BL & AL

ans:
IDIV BL

4) Move 1000H to Register BX

ans: MOV BX, 1000H

Q.2 List addressing modes of 8086 with example

ans i) Immediate addressing mode:
The [data] datum (Operand) is either 8 bit or 16 bit long & is the part of Instruction
Datum

eg: MOV AL, 22H
MOV BH, 15H
MOV AX, 1000H

Eg:
8086 routin to convert ASCII code byte to EBCDI equivalent
: ASCII code byte in AL at start
: EBCDI code in AL at end
MOV BX: OFFSET EBCDI_TABLE
: Point Bx at start of EBCDIC table in DS
XLAT : Replace ASCII to AL with EBCDIC from table

ii) XCHG:
The XCHG instr exchanges the content of a register with the contents of another register or with content of memory location. The XCHG cannot directly exchange the content of 2 mem location. A mem location can be specified as the source or as destination by any of the 24 addressing mode. The source & destination must both be words or they must both be byte. The segment register cannot be used in the instr. No flags are affected by this instr.

eg:
XCHG AX, DX : Exchange words in AX with words in DX

Q5 Which instruction are used for stack access? illustrate the use of any one?

Ans The instr used for stack access are as follows
1) PUSH
2) PUSHF
3) POP
4) POPF

PUSH:
The push instr decrements the stack ptr by 2 & copies a word from a specified source to the location in the stack segment where the stack ptr then points. The source of word can be a general purpose reg, segment reg or memory. The stack seg reg & the stack ptr must be initialized before the instr can be used. Push can be used to save data on the stack: so that it will not be destroyed by a procedu It can also be used to put data on the stack so that the procedure can access it

eg
Push Bx: Decrement stck ptr by 2, Copy Bx to stack

Q.6 Name different type of jump inst used in 8086 assembly language prog. Give diff btwn inter Seg & intra Seg type of jump. describe each with eg

ans. Name	Function	Condition
JA / JNBE	Jump if above / Jump if not below or equal	CF = 0 ZF = 0
JAE / JNB	Jump if above or equal is not below	CF = 0 ZF = 1
JB / JC	Jump if below / Jump if not above or equal / Jump if Carry	CF = 1 ZF = 0
JBE / JNA	Jump if below or equal / Jump if not above	CF = 1 ZF = 1
JG / JNLG	Jump if greater / jump if not less than or equal	CF = 0 ZF = 0
JLE / JNG	Jump if less than or equal / Jump if not equal	ZF = 1 SF ≠ O
JL / JNGE	Jump if less than / Jump if not greater than or equal	SF ≠ O

jump inst used prog. Give intra Seg type eg

	Condition
	CF = 0 ZF = 0
	CF = 0 ZF = 1
	CF = 1 ZF = 0
	CF = 1 ZF = 1
	CF = 0 ZF = 0
	ZF = 1 SF ≠ O
	SF ≠ O

Eg:-

Inter Segment	Intra Segment
1) This seg jump from 1 seg to another or it jump outside of the present Segment	This seg jump within one seg or it jump in present Segment
2) It is far jump	It is near jump
3) More memory is req	less memory is req
4) It is declared in diff Seg then the seg where the main prog is stored	It is declared in same Seg where the main prog is stored
5) CALL & RET are inter seg	Jump uncondil are Intraseg

Eg:-

```
Code Segment
:
Jump label
:
ENDS
Code Segment
:
LABEL
:
ENDS
```

```
Code Segment
:
LABEL
Jump LABEL
:
:
ENDS
```

Q.7 Write instr. for following
1) Complementing Carry flag
2) Set interrupt flag
3) Swap the lower nibble & upper nibble content of DL reg

ans
1) Complementing Carry flag
CMC (Complement Carry)

2) Set interrupt flag
STI (Set interrupt)

3) Swap lower & upper nibble content of DL reg
MOV AL, DL
ROL AL, 04H

Q.8 With suitable eg describe the use of DAA instruction of 8086.

ans DAA:
This instr is used to make sure the result of adding two packed BCD number is adjusted to be a legal BCD number the result of the addition must be in AL for DAA to work correctly if the lower nibble in AL after addition is greater than 9 or AF was set by the addition than the DAA instr will add 6 to the lower nibble in AL if the result in the upper nibble of AL is greater than 9 or if the carry flag was set by the addition or correction then DAA instr will add 64 to AL

eg
AL = 0101 1001 = 59 BCD
BL = 0011 0101 = 35 BCD
ADD AL, BL AL = 1000 1110 = 8EH
DAA: add 0110 because 1110 > 9
AL = 1001 0100 = 94 BCD
The DAA instr updates AF, CF, PF & ZF
OF is undefined after a DAA instr

Q.9 Describe with eg two comparing instr of 8086 up

ans CMPS:
A string is a series of the same type of data item in sequential memory location
The CMPS instr can be used to compare a byte in one string with a byte in

another string or to compare a word in one string to word in another string. SI is used to hold the offset of a byte or a word in the source string & DI is used to hold the offset of a byte or a word in the string. the comparision is done by subtracting the byte or word pointed to or by DI from the byte or word pointed by SI. The AF, CF, OF, PF, SF & ZF flags are affected by comparision. after the comparision SI & DI will automatically be incremented or decremented to pt the next element in the string.

CMP:
this instr compares a byte from the specific source with a byte from the specified source with a word from the specified destination the source can be an immediate number, a reg or a memory location specified by one of the 24 addressing modes. The destination can be a reg or memory. Loc however

word in
string.
of a byte
& DI is
byte or a
rision is
de or word
the byte
nc AF, CF,
ffected by
rision SI
ncremented
next element

Q10
ans

Q11

om the Specific
specified
e specified
be an
a memory
of the
he destination
Loc however

the source & destination cannot both be mem location in the same instr. the byte or word from the destination byte or word. the source & destination are not changed but the flags are set to indicate the result of the comparision. AF, OF, SF, ZF, PF & CF are updated by this instr

eg:
CMP BX, 0100H
CMP 0100

Q10 Explain LEA instruction.
ans LEA
this instruction determines the offset of the variable or memory location named as source & put the offset in the indicated 16 bit reg. LEA changes no flag.
eg:
LEA BX, Prices : load BX with offset of prices DS

Q11 Compare the following 8086 instr
1) AND & TEST
2) AAA & DAA

ans:

AND	TEST
1) It AND the corresponding bit of two operand	It AND the operand to update flags
2) After the AND both the corresponding result of two number is stored in destination word	This flags are affected but don't store the result of Corresponding two operands

AAA	DAA
1) ASCII adjust for addition	Decimal adjust after BCD addition
2) It is only for ASCII addition	It is only for BCD addition

EST
the operand
date flags
lags are affected
don't store the
t of Corresponding
operands

DAA
mal adjust after
addition
only for
addition

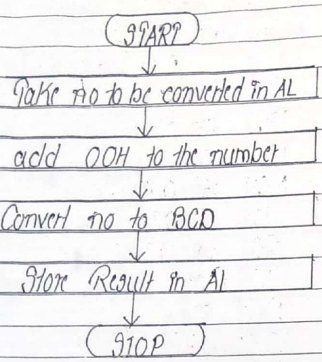
Q Distinguish betwn 8085 & 8086 up.

ans

	8085	8086
1)	It is an 8 bit up	It is 16 bit up
2)	It has 8 bit data bus	It has 16 bit data bus
3)	It has 8 bit flag register	It has 16 bit flag register
4)	It has 16 bit address bus	It has 20 bit address bus
5)	Provides 2 level interrupt	Provides 1 level interrupt
6)	64 KB of addr space	1 MB of addr space
7)	Doesn't req extra clk generator	It req external clk generator
8)	Max clk freq is 3 MHz	Clk freq is 5 MHz

Q.17 Write an ALP to Convert hex to BCD

ans

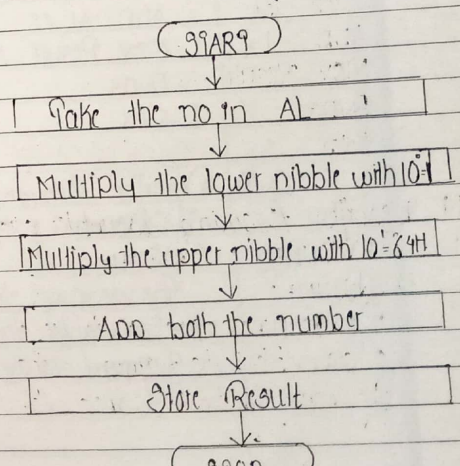



```
Assume CS: Code
       DS: Data
Code Segment
MOV AX, data
MOV DS, AX
MOV AL, 00H
MOV CL, 08H
UP:  RCL DI, 08
     JC DOWN
A:   DEC CL
     JNZ UP
```

```
     MOV RESULT, AL
     JMP XYZ
DOWN: ADD AL, 01H
     JMP A
     ENDS
Data Segment
Result db (?)
ENDS
END
```

Q.18 Draw flowchart & write an ALP to convert BCD to HEX

ans



```
Assume CS: Code
Assume DS: Data
Code Segment
MOV AX, Data
MOV DS, AX
MOV AL, 56H
MOV BL, 01H
MUL BL
AAM
MOV CL, AL
MOV AL, AH
MOV BL, 0AH
MUL BL
ADD AL, CL
MOV Result, AL
ENDS
END
```

Q.19 Describe following assembler directive
1) ASSUME & 2) EVEN

ans 1) ASSUME:
The assume directive is used to inform the assembler, the names of the logical segment to be assumed for different

r directive

used to inform
of the logical
for different

Segments used in the prog in assembly language prog each segment given the name code. Data seg given the name Data

2) EVEN:
The assembler, while creating the assembling procedure of any prog, init a loc counter & goes on updating it as the assembling proceeds.
The EVEN directive updates the location counter to the next even address if the current location counter contents are not even, & assign the following routine or variable or constant to that address.

Page No.
Date

Q.12 Explain the assembler directive DD, DUP, DW & ENDS

ans * DD: Define doubleword

The DD directive is used to declare a variable of type double word or to reserve memory location which can be access as type double word.

* ENDS: End Segment

This directive is used with the name of a segment to indicate the end of that logical segment. ENDS is used with the SEGMENT directive to 'bracket' a logical seg containing instr of data

eg Code Segment : Start of logical segment containing code.

Code ENDS : End of seg named END

* DW : Define word

The DW directive is used to tell the assembler to define a variable of type word or to reserve storage location of type word in memory

eg:
THREE_LITTLE_WORDS DW
1234H, 3456H, 5678H;
declare array of 3 word & init with
specified value

* DUP:
Syntax: 10 DUP (0)
This directive allocate the no of memory
location. the no of mem location is
counted by multiplying the decimal
no given before the DUP directive
with the data directive type
eg:
MSG db 10 DUP (0) - allocate 10x1
byte = 10
mem loc.
MSG dw 100 DUP (0) - allocate
100x2 byte
= 200 mem loc.

Q.13 Write an ALP to multiply two 16 bit
BCD no

ans

DW
H;
& init with

no of memory
location is.
the decimal
DUP directive
type

- allocate 10x1
byte = 10
mem loc.
- allocate
100x2 byte
= 200 mem loc.

ply two 16 bit

(START)
↓
Store 1st no in AX register
↓
Store 2nd no in BX register
↓
Multiply AX with BX
↓
Convert the no in BCD
↓
Store the Result
↓
(STOP)

```
Assume CS: Code
       DS: Data
Code Segment
MOV AX, Data
MOV DS, AX
MOV AX, num1
MOV BX, num2
MUL BX
MOV CL, 00H
ADD AL, CL
```

```
DAA
MOV Result 1, AL
MOV AL, DL
ADD AL, CL
DAA
MOV Result 2, AL
MOV AL, DH
ADD AL, CL
DAA
MOV Result 3, AL
Ends
End
Data Segment
Num 1 dw (1234H)
Num 2 dw (5678H)
Result db (?)
Result 1 db (?)
Result 2 db (?)
Result 3 db (?)
Ends
End
```

Q.14 What are assembly directive? Write abt any 1 in brief?

ans Assembly directive are the instruction which gives direction to the assembler

DW:

The DW directive is used to tell the assembler to define a variable of type word or to reserve storage location of type word in memory

eg

```
THREE_LITLE_WORDS
1234H, 3456H, 5678H
declare array of 3 word & init with
specified value.
```

Q.15 Using assembly directive write and ALP to multiply two unsigned 16 bit number? State diff btwn MUL & IMUL instruction

ans

```
Assume CS: Code
       DS: Data
Code Segment
MOV AX, data
MOV ds, AX
```

MOV AX, num 1
MOV BX, num 2
MUL BX
MOV Result; AX
ends
Data Segment
num 1 dw 0008H
num 2 dw 0002H
Result dw (?)
ends
end

MUL	IMUL
i) This instr is used for unsigned multiplication	This instr is used for signed multi
2) Shows unsigned result	Shows signed result
3) It does not repr^nt the signed magnitude in any register	It repr^nt the signed magnitude
4) Signed flag is not rejected	4) Sign flag is rejected

QT6
ans
UP:
A:
DOWN:

QT6 Write an ALP to count the number of one's in register DL
ans Assume CS: Code
DS: Data
Code Segment
MOV AX, data
MOV DS, AX
MOV AL, 00H
MOV CL, 08H
UP: RCL DL, 01
JC Down
A: DEC CL
JNZ UP
MOV Result, AL
JMP XYZ
DOWN: ADD AL, 01H
JMP A
Ends
Data Segment
Result db (?)
ends
end

```
Assume CS: Code
Assume DS: Data
Code Segment
MOV AX, Data
MOV DS, AX
MOV AL, 56H
MOV BL, 01H
MUL BL
AAM
MOV CL, AL
MOV AL, AH
MOV BL, 0AH
MUL BL
ADD AL, CL
MOV Result, AL
ENDS
END
```

Q.19 Describe following assembler directive
i) ASSUME & ii) EVEN

Ans i) ASSUME:
The assume directive is used to inform the assembler, the names of the logical segment to be assumed for different

directive

used to inform
of the logical
for different

Segments used in the prog. In assembly language prog each segment given the name code. Data Seg given the name Data

ii) EVEN:
The assembler, while creating the assembling procedure of any prog. init a loc counter & goes on updating it as the assembling proceeds.
The EVEN directive updates the location counter to the next even address if the current location counter contents are not even, & assign the following routine or variable or constant to that address.

Q.2 Define macros with eg?

ans When the repeated group of instruction is too short or not appropriate to be written as a procedure we use a macro A macro is a group of instr we bracket & give a name to the start of our prog Each time we call the macro in our prog the assembler will insert defined group of instr in place of the 'Call'. In other word the macro call is like shorthand expression which tell's the assembler Every time you see a macro name in a prog replace it with the group of instr defined at the start of macro the prog. An impt pt is that the assembler

generates machine code for the group of instr each time the macro is called. replacing the macro with the instr it represents is commonly called expanding the macros since the generated machine code are right in-line with the rest of the prog the processor does not have to go off to a proc & return. Therefore using a macro avoids the overhead time involved in calling & returning from a procedure. A dis-adv of generating in-line code each time a macro is called is that this will make the prog take up more memory than using a proce.

Eg:

```
PUSH_ALL MACRO
PUSHP
PUSH AX
PUSH BX
PUSH CX
PUSH DX
PUSH BP
PUSH SI
PUSH DI
```

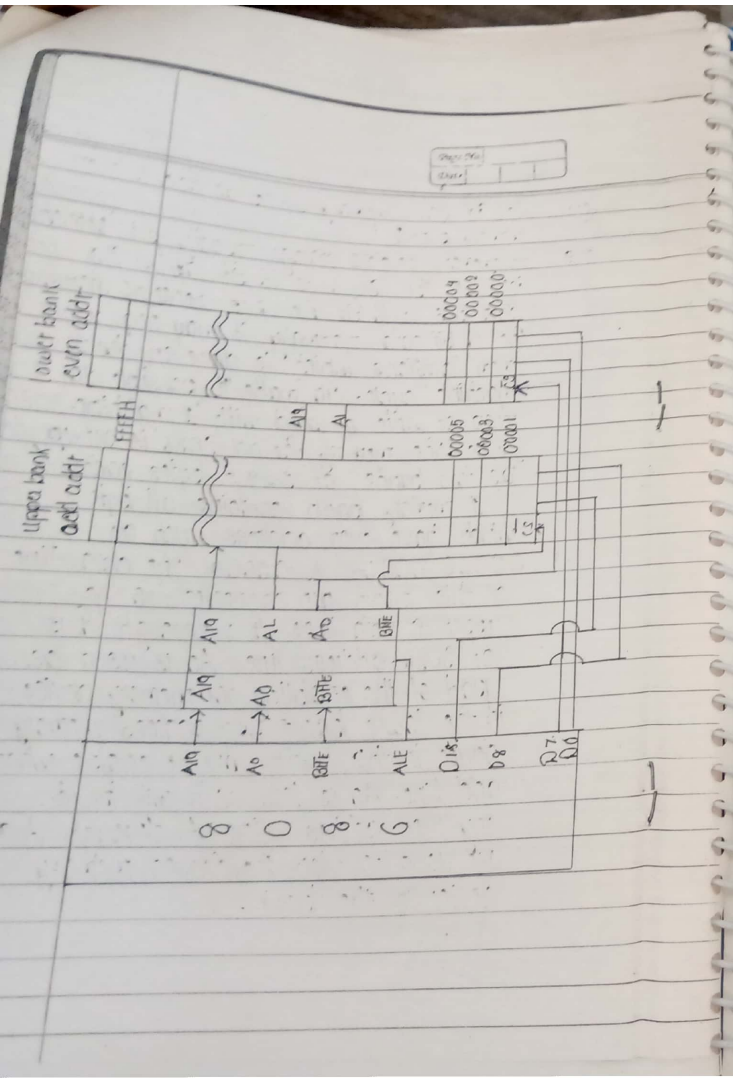
using
dress.

Page No.
Date

Q.2 Why is 8086 memory set up as odd & even memory bank?

ans In memory the data storage capacity per location is 8 bit but 8086 is 16 bit up. ie it's data bus is 16 bit D0 - D15 hence to store the 16 bit data normally it may req 2 mem locations which is done in two machine cycle. To make it possible to read or write a word with 1 machine cycle. The memory for an 8086 is set up as two banks of 524 K bytes each.

1 memory bank contains all byte which have even address such as 00000, 00002, & 00004. The data line of this bank are connected to the upper 8 data lines D8 through D15 of 8086. address line A0 is used as part of the enabling for memory device in the lower bank. An address memory device in this bank will be enable when add line. A0 is low as it will be for any even address line. A1 through A19 are used to select the desired memory device in the bank.



Q.3 Difference btwn Memory mapped I/O & I/O mapped I/O

ans 1) I/O & memory devices are treated as memory devices

2) Control signal for memory as well as I/O devices are MEMR & MEMW

3) All memory related instr are used to access I/O devices

4) Data transfer is possible btwn any register & I/O devices

5) Address decoding logic is complicated & expensive

6) 8086 can access 1 MB max I/O devices of memory as address space is shared so total no of I/O port & mem location should not be more than 1 MB

7) I/O devices & mem address is 20 bit as I/O device are treated as memory