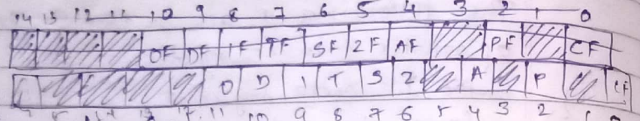
(*) The flag register of 8086 is as follows:-




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

2) Parity flag (PF):-
It is set to 1 if the lower number of the 8 bit of the result ~~is~~ contains an even number of 1's otherwise it is cleared.

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

4) Sign flag (SF):- If sign flag holds the arithmatic sign of the result after arithmatic or logical operation. If the number is 1 that is considered as negative number and if the number is 0 than it is considered as positive number.

5] Trap flag (TF):-
If the trap flag is set is ~~set~~ initialize 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 the interrup to pin is enabled.

7] Direction flag:-
Direction flag selects either increment/decr mode for the SI/DI register during string instruction, if DF=0 the ~~flag~~ register are incremented & if DF=1 the register are automatically decremented.

8) Overflow flag (OF):-
The overflow flag indicates that the result has exceeded the capacity of the machine

(A) Memory Segmentation of 8086.

i) There are two types of organization which are commonly used. They are linear addressing and segmented addressing.

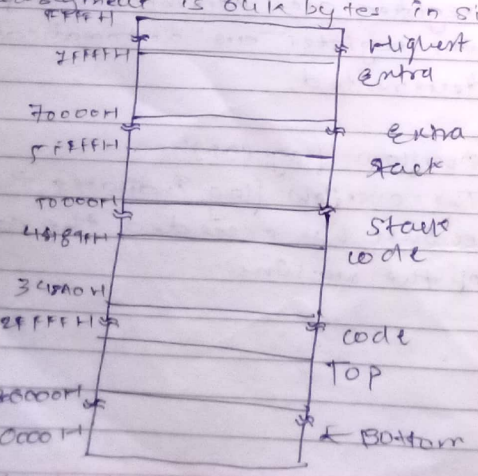
ii) In linear addressing entire memory space is available to the processor in one linear way.

iii) In segmented addressing, on the other hand the available memory is divided into "chunks" called segments.

iv) In 8086 systems, the available space is 1MB bytes.

v) Memory is divided into number of logical segments.

vi) Each segment is 64K bytes in size.




(I)

| Segment | Offset | Special Purpose |
|---|---|---|
| CS | IP | Instruction address |
| SS | SP or BP | Stack address |
| DS | BX, SI & DI 8bit/16bit | Data |
| ES | DI for string instr | string destination address |

(A) Features of 8086.

i) Provides 20 bit address line

ii) Multiplex 16 bit address & data bus ~~to minimise no of pins in IC~~.

iii) Provides 16-bit data bus.

iv) ~~It~~ 16 fourteen 16-bit registers

v) Operaters has two modes minimum and maximum mode.

vi) Supports multiprogramming.

vii) It is 16-bit microprocessor.

(3) i) Rotate register BL left four times.

→ MOV CL, 04H
ROL BL, CL

ii) Multiply AL by 05H.

→ MOV AL, 05H
MUL AL.

iii) Signed division of BL & AL.

→ IDIV BL

iv) Move 4000H in Bx register.

→ MOV BX, 4000H.

v) Load Offset 1000H in register Bx

→

vi) Rotate Bx to left 4 time through carry

(4) Maximum Mode.

ALE, INTA, M/IO, R/DT, HLDA, HOLD, DEN, WR

Signal is used for transmitting & receiving of data

i) GND:- Ground.

ii) S0, S1, S2 These pins are called status i/p pins.

iii) ALE (Address Latch Enable):- This signal is used to demultiplex the AD0-AD15 pin by A0-A15 and D0-D15.

iv) INTA (Interrupt Acknowledge):- This indicates regonition of an interrup pin.

v) M/IO:- This pin is use to distinguish memory transfer & data transfer

vi) R/DT:- This signal is used to control the data flow direction. High on this is used for transmitting the data and low is for receiving the data

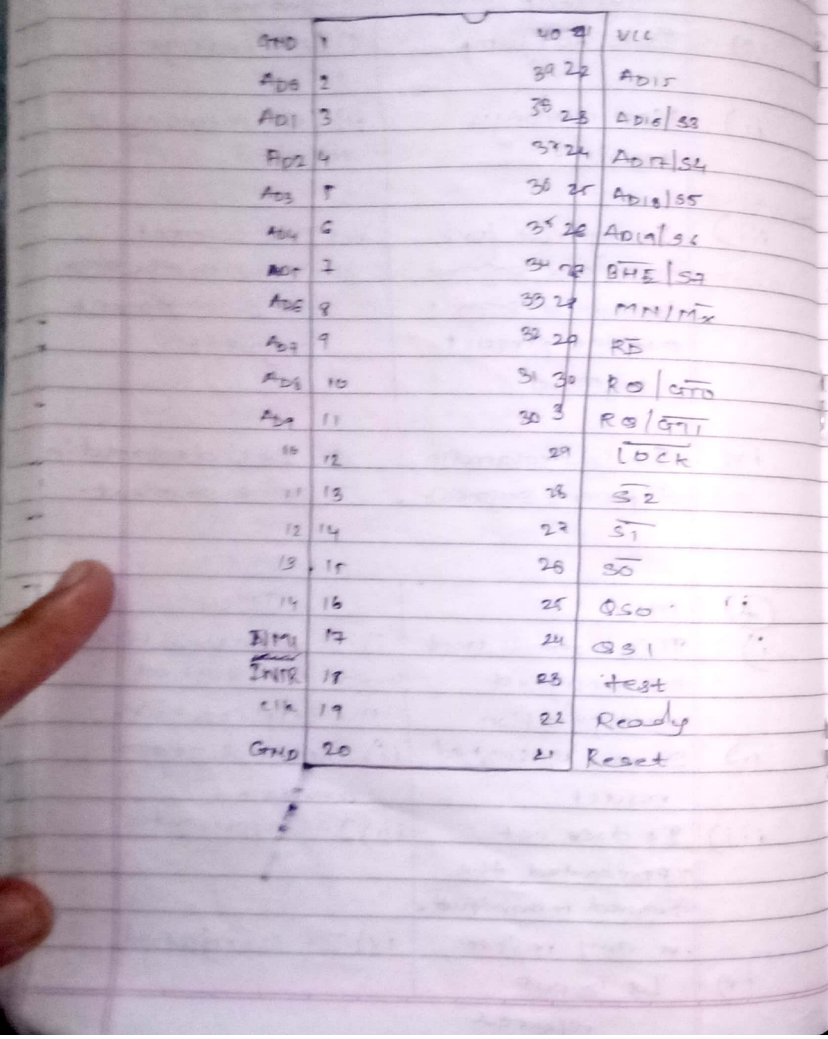
| Minimum Mode | Maximum Mode. |
|---|---|
| i) MN/MX is connected to vcc (+5 v) | i) MN/MX is connected to GND (to 0). |
| ii) It is single processor system configuration | ii) It is a multi processor system configuration |
| iii) In 8086, here itself generates system control. | iii) Here external device is been used. |
| iv) Minimum signal mode are from 24 to 30. INTA, ALE, DEN, M/IO, DT/R, WR, HLDA & HLD respectively. | iv) Maximum mode $\overline{QS_1}$, $\overline{QS_0}$, $\overline{SO}$, $\overline{S_1}$, $\overline{S_2}$, LOCK, RQ/$\overline{GT_1}$, RQ/$\overline{GT_0}$ respectively. |

12/04/19

| (A) | Inter segment | Intra segment |
|---|---|---|
| i) | It is far jump | i) It is near jump. |
| ii) | More memory is required | ii) less memory is required. |
| iii) | It segment jump from one segment to another segment outside present segment | iii) This segment jumps with the segment 0 in present segment |
| iv) | It is declared in different segment | iv) It is declared in same segment |

| (B) | MUL | IMUL. |
|---|---|---|
| i) | This instn is used for unsigned multiplication | i) This instruction is used for signed multiplication. |
| ii) | Shows unsigned result | ii) Shows signed result. |
| iii) | It does not represented the signed magnitude in any register | iii) It represented. |
| iv) | It is not rejected. | iv) It is rejected. |

Pin diagram




(A) Addressing modes:-

1) Immediate addressing mode:- Here the data can be of 8 bit or 16 bit in the form of successive byte or word.

eg. MOV AL, 16H

2) Direct:-

In Direct addressing mode the 16 bit data is directly specified in the instruction as a part of it.

eg. MOV AX, [1000H]

3) Register:-

In Register addressing mode, the data stored in the register and it is refered using the particular register

eg:- mov AX, BX

MOV AL, BL

4) Indexed:

In Indexed addressing mode offset of the operand is stored in one of the Indexed register.

eg) MOV AX, [SI]

5) Implied addressing mode:-

No address is required because the address of the operand is implied in the instruction itself

eg) STD, STC, NOA

Register Indirect:
Operand is determined in an indirect way of using offset address.
eg- MOV AX, [BX]

i) Text editor:-
The first step in the development process is to write an assembly language program. The assembly language program can be written with an ordinary word such as star, edit and so on.
c:\tasm\bin\> edit prg.asm.

ii) Assembler.
- Assembler translates the source file that was created using the editor into machine language such as binary or machine code.
- The assembler then reads the source file of our program from the disk where we have saved the editing.
- The assemblers generated two files or floppy floppy or hard disk during these two two passes.
- The first file is called object file
- The second file is called assembly list file.

the command on command prompt performing this operation is given below:
c:\tasm\bin\> tasm prg

iii) Linker.
The linker is generally used to join together a several programs into one large program.
When writing a large program, it is usually much more efficient to divide them into small modules. Each module can be individually tested, written and debugged. And then this small modules then can be joined together and will get come into a large functioning program.
The linker also provides a link map which contains the address information about the link files.

command prompt
c:\tasm\bin\> tlink prg.obj

iv) Debugger:-

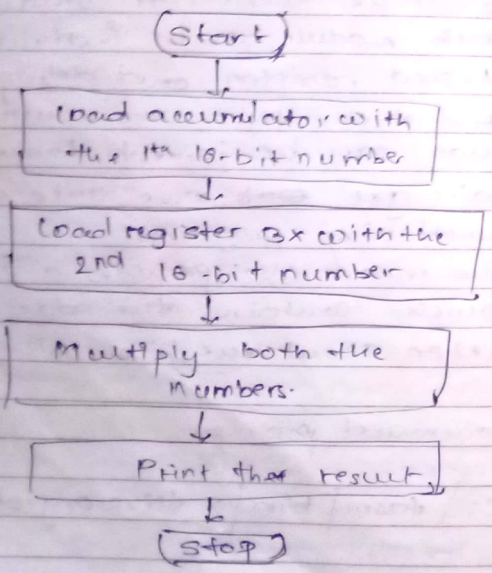
The debugger is a program which allows us to load the object program into system memory, execute the program and debug it.

command prompt:-

C:[team]bin>td prg.exe.

(A) Draw the flowchart for multiplication of two 16 bit numbers.

→




(A) Procedures:- A procedure is a group of instructions that usually performs one task, it is a reusable section of the software in memory once, but used as often as necessary.

Syntax:-

Procedure-Name PROC

---

Procedure statements

Procedure-name ENDP

(A) What is the role TEST instruction in Assembly Language Programming

→ The TEST instruction performs the AND operation. The difference is that the AND instruction changes the destination operation, while the TEST does not changes. The TEST instruction only affects the ~~ft~~ flag register, which indicates the result of the test

eg:- TEST AL, BL

TEST AX, BX

(A) List the major steps in developing an Assembly language program

→ Major steps in developing an Assembly language program are:-

i) Specifying the problem.
ii) Designing the problem solution.
iii) Coding
iv) Debugging.

(B) What is stack? State its significance.

→ The stack is a section of memory you set aside for storing return addresses. The stack is also used to save the contents of registers for the calling program while a procedure executes.

Significance is are:-

i) PUSH instruction.
ii) POP instruction.

(A) What is the use of REP in string related instructions?

→ REP is a prefix which is written before one of the string instructions. This instructions repeat until the specified condition exists.

(A) What is pipelining? How it improves the processing speed?

→ i) To speed up program execution, the BIU fetches six instruction bytes ahead of time from the memory.

ii) These prefetched instruction bytes are held for the execution unit in groups of registers called queue.

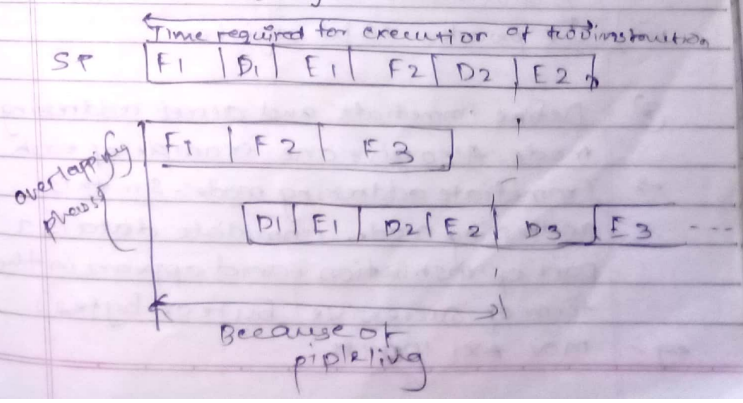
iii) The BIU continues the process as long as the process does not get filled.

iv) In case of JUMP and CALL instructions, instruction already fetched in queue for no use.

v) Features of fetching the next instruction while the current instruction is executing It is called as pipelining.

vi) The queue operates the principle of First in first out (FIFO).

Diagram.



## Summer - 17

(A) State the function of following pins of 8086 microprocessor.

i) INTR :- It is a level triggered, and non-vectored interrupted. When INTR occurs the microprocessor generates interrupt acknowledge ~~meet~~ signal $\overline{INTA}$.

ii) $\overline{INTA}$ :- It is an active low acknowledgement signal for INTR.

(A) List any four features of 8086.

→ i) It is a 16 bit microprocessor.

ii) It has 16 bit databus.

iii) It has 20 bit address bus.

iv) It mainly have 40 pins.

v) It has multiplexed pin AD0 - AD15 and A16 - A19.

vi) It has two modes named as minimum & maximum.

vii) It provides 16 bit register.

(A) Define immediate and direct addressing mode. Also give one example of each.

→ Immediate addressing mode :- In this addressing mode, immediate data is a part of instruction and appears in the form of successive byte or bytes.

eg :- MOV AX, 1004H

ii) Direct addressing mode :- In direct addressing mode, 16 bit (offset) address is directly specified as a part of it.

eg :- MOV CL, [1004H].

(A) List the program development steps for assembly language programming.

→ i) Defining the problem.

ii) Algorithm

iii) Flowchart.

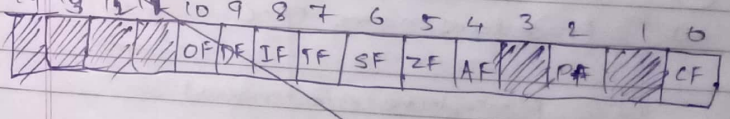
iv) Initializing checklist.

v) choosing instructions.

vi) Converting algorithms to assembly language program.

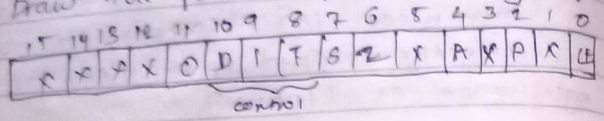
(A) Draw the format of 8086 flag register

→



(A) Draw the format of 8086 flag register

→




(A) Give the steps in physical address generation in 8086 microprocessor.

→ i) Segment register carry 16 bit data, which is also know as base address.

ii) BIU append CBIU Interface unit) four 0000 bits to LSB of the base address. This address becomes 20-bit address ~~bit~~.

iii) Any base / pointer or index register carries ~~offset~~ 16 bit offset.

~~+ve~~

(A) Give the syntax for defining a procedure.

→ Procedure_name PROC

- - - -

Procedure Statement

Procedure_name ENDP.

(A) i) copy 1000H to register BX

→ MOV BX, 1000H

ii) Rotate register BL left four times

→ MOV CL, 04H

ROL BL, CL.

(A) State the function of Assembler and Debugger.

→ i) Assembler :-

- Assembler is a program which translates assembly language program to ~~the~~ correct binary code.
- It also generates the object file with extention .obj.
- It also displays syntax error in program, if any.
- It can also be used in ~~produces~~ .list and .crf files.

ii) Debugger:-

- Debugger is a program that allows the execution of program in ~~sing~~ single step mode under the control of user.
- The error in program can be located and corrected using a debugger.
- Debugger generates ~~exe~~ file.

(A) EDB :- i) This is used to define byte type variable.

ii) This can be used to define single byte or multiple byte

- DW (Define word) :

i) This is used to define 16-bit word type variable.

ii) This can be used to define single word or multiple word.

- DD (Define double).

i) This is used to define double word (32-bit) type variable.

ii) This can be used to define double word or multiple double word.

- DQ (Define quad word).

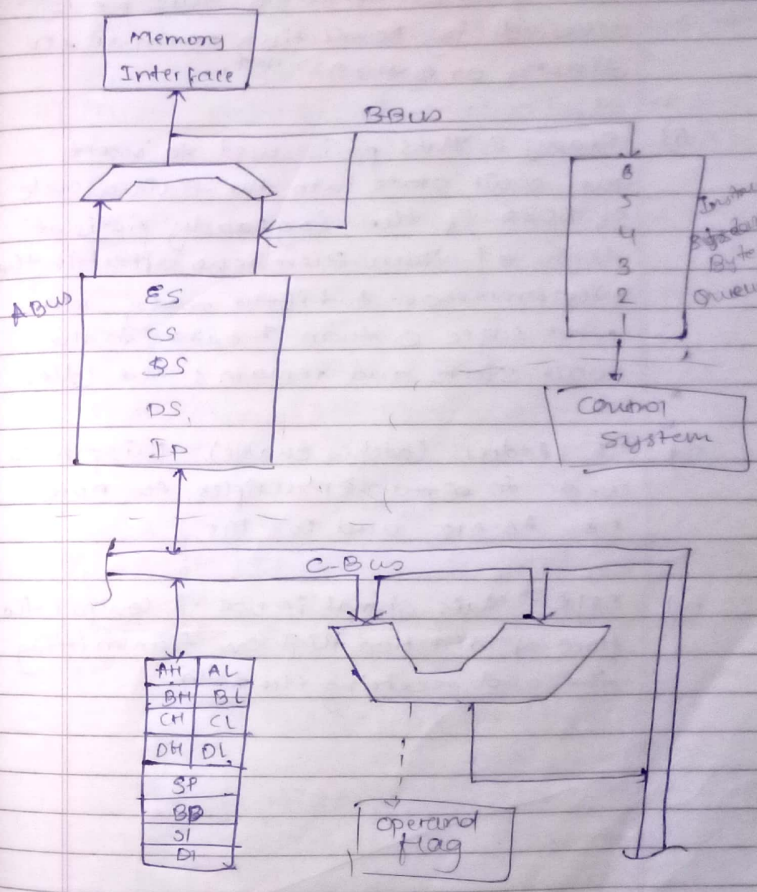
i) To define (64-bit) type variable.

(3)

| Re-entrant Procedure | Recursive Procedure |
|---|---|
| i) A procedure is said to be re-entranted, if it can be interrupted, re-entered, used without losing or corrupting over anything. | i) Recursive procedure is a procedure which calls itself. |
| ii) | |

(4) Draw the architecture of 8086.

->



(A) i) MN/$\overline{MX}$:- The MN/$\overline{MX}$ pin is used to select Minimum mode or maximum mode operation of 8086. This pin is achieved by connecting either +5V directly or to the ground.

ii) Ready :- This pin is used to insert the wait state into the timing cycle of 8086. If the Ready pin is at logic 1 then there is no affect in the microprocessor. And if the ready pin is at logic 0 then it enters to the wait state and remains the idle.

iii) $\overline{ALE}$ (Address Latches Enable):- This pin is used to demultiplex AD0-AD15 i.e A0-A15 and D0-D15

iv) DT/$\overline{R}$:- This signal is use to control the flow of direction. High on transmitting the and receiving the data.

(A) Explain the concept of pipeline in 8086 microprocessor with diagram.

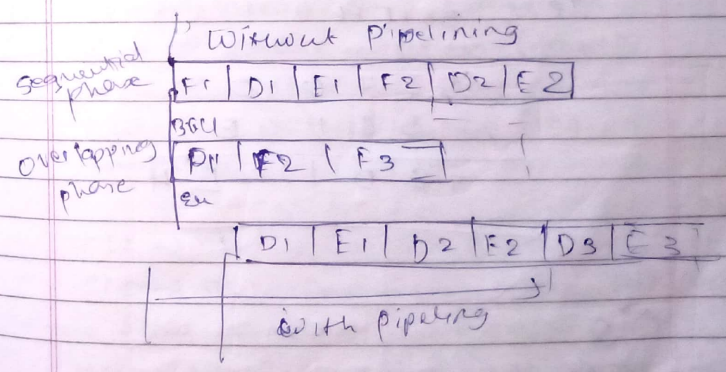
→i) To speed up program execution, the BIU fetches six instructions bytes ahead of time from the memory.

ii) These prefetched instruction bytes are held for the execution unit of group called as queue.

iii) Pipelining follows the principle of first in and first out i-e FIFO.

iv) The BIU continues the process as long as the queue is not full.

v) In case of JUMP and CALL instructions, instructions get fetched already for no used.



(3) Calculate the physical address for given:

i) DS = 73A2H
SI = 3216H

```
  73A20
   3216
  ------
  7AC36
```

```
  73A20
   3216
  ------
  7AC36H
```

ii) CS = 7370H
IP = 561EH

```
  73700
   561E
  ------
  78D1EH
```

CLC
CLD
STC
STD

(5) State the function of the following register of 8086 microprocessor.

->) General Purpose Registers of 8086

- AX
- BX
- CX
- DX
- SB
- BP
- SI
- DI
- IP

Segment Register
- CS
- DS
- ES
- SS

(4) List and explain any four process control instruction with their function.

i) CLC → clear carry flage
CLD → clear direction flag
STC → set carry flag
STD → set direction flag.
~~WAIT~~ WAIT →

CLC → This instruction clear carry flag. CF ← 0
CLD → This instruction clear Direction flag DF ← 0
STC → This instruction set carry flag CF ← 1
STD → This instruction set Direction flag. DF ← 1.

(*) Identify the addressing mode for following instructions:

i) MOV AX, 2034H
→ Immediate addressing mode.

ii) MOV AL, [6000H]
→ Direct Addressing mode.

iii) ADD AL, CL
→ Register addressing mode.

iv) MOV AX, 50H [BX] [SI]
→ Relative Base addressing mode.

(*) XLAT
→ i) XLAT replaces a byte in AL register with a byte from 8-256 byte lookup table beginning with [BX]
ii) AL is used as offset in this table.
iii) Flags are not affected.
iv) operation - AL ← [BX+AL]

ii) XCHG
This instruction exchanges the register from the another register or memory location.

(*) Define MACRO with its Syntax. Also give two advantages of it.

→ MACRO is a group of instructions. The macro assembler generates the code in program where each time macro is 'called'. MACRO can be defined by MACRO and ENDM as assembler directives.

Advantages of MACRO
i) To speed up the execution of the program.
ii) It reduces the length of the program.

| (*) Near Procedure | Far Procedure. |
|---|---|
| i) It is also called as Inter-segment | i) It is also called as Intra segment |
| ii) less stack segment is required | ii) More stack segment is required. |
| iii) The near procedure can replace old IP to New IP | iii) The far procedure can replace the old DS: IP pairs with new DS. IP |
| iv) It has the same code segment | iv) It has the different code segment |

Winter 17 MIC

(A) Syntax :-

macro-name MACRO [arg1, arg2 - argN]

(A)

| Near Procedure | Far Procedure |
|---|---|
| i) It is also called Intra segment | i) It is also called as Inter segment |
| ii) Less stack location is required | ii) More stack location is required. |
| iii) The near procedure replace a the old IP with the new IP | iii) The far procedure replace the old CS:IP pairs with the new CS:IP. |
| iv) It has same code segment | iv) It has different code segment |

(A) INC :-

i) This instruction is used to increament the operand specified.

Syntax

INC operand.

- Operand can be either register or memory locations.
- Operand : operand+1
- Immediate cannot be an operant.

(A) LOOP

This instruction is used to ~~Specify~~ series of instruction in many times. The number of times the instruction sequence is to be repeated is loaded in cx. Each time loop is executed, the cx is decremented by 1

Syntax :-

LOOP label name

(A) Re-entrant procedure :-

In some situation ~~the~~ it may happen ~~that~~ Procedure 1 is called in main program Procedure 2 is called in procedure 1 and Procedure 1 is called again in Procedure 2 If the should not occure than we have a procedure name as re-entrant. re-entrant procedure.

In this procedure it is, it can be interrupted used and re-entered with losing or writing anything.

(A) Two manipulation instructions.

i) AND, OR, EXOR, NOT

1MB = 00000 H — FFFFF H

1111 1111 1111 1111 1111

$A_0 - A_{19}$

(4) State the control signal generated by $S_0, S_1, S_2$ with their function of 8086.

→

| | $S_2$ | $S_1$ | $S_0$ | Function |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | Interrupt acknowledge |
| 1 | 0 | 0 | 1 | Read I/O |
| 2 | 0 | 1 | 0 | write I/O |
| 3 | 0 | 1 | 1 | Halt |
| 4 | 1 | 0 | 0 | Code access |
| 5 | 1 | 0 | 1 | Read memory |
| 6 | 1 | 1 | 0 | write memory |
| 7 | 1 | 1 | 1 | Inactive. |

FFFFF

64 Kbytes — 16 Segments

Segment

Code Segment
Stack
Data
Extra

1MB

$2^{20}$ = 1MB

64 Kbytes

+5V

0 V

Registers

D15 D0

0000 0000 0000 0000

1111 1111 1111 1111

$2^{16}$ = 64 Kbytes

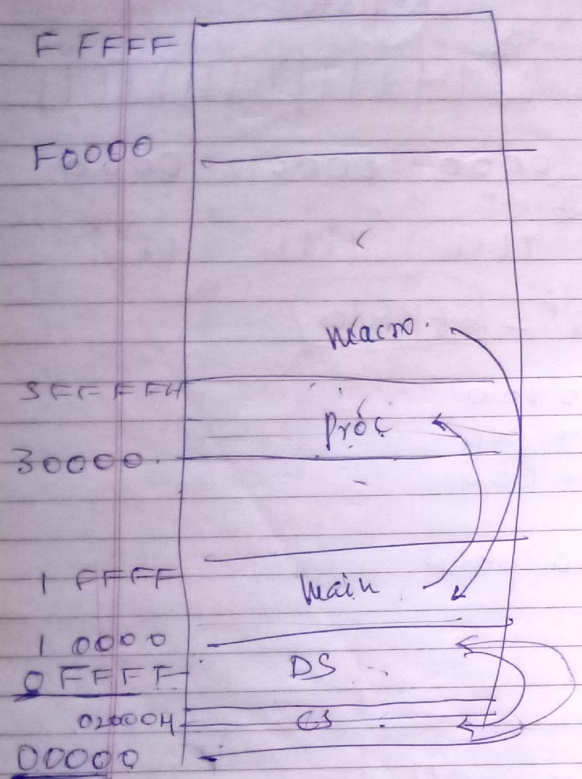
1000 000
— 64 000
946 000

main program

JMP, Call

RET

20 bit




Segment registers.

CS ⇒ I.P
DS ⇒ BP
ES ⇒ DI
SS ⇒ CI

CS : 2000H
IP : 0000H

| | |
|---|---|
| 00000 | 30000 |
| + 2000 | 2000 |
| 02000H | 32000H |

CS: Harshala
CS: XYZ

Assume CS: ~~code~~, DS: Data

XYZ
Code Segment

Mnemonics

MOV AX, Data
MOV DS, AX.
MOV AX, num1
MOV BX, num2
ADD AX, BX.
MOV Result, AX.

opcodes.

Code ENDS
Data Segment
num1 DW 1111H
num2 DW 2222H
~~Hit~~ Result DW ?

Data ENDS

0011 0011 0011 0011

num db 05, 4, 3, 2, 1, 6, 7, 8, 9.

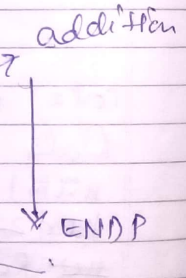
05 − 04 = 01

4 − 5 = −1

Addition  Subtraction.

Main program.
data Segment
num1 dw 4444H
num2 dw 2222H
sum dw ?
difference dw ?
data ENDS

main segment.
mov ax, data
mov ds, ax
~~jmp proc~~ call Addition.




Addition. ~~segment proc~~ Macro num1, num2
mov ax, num1
mov bx, num2
Add AX, BX.
Endm

Subtraction Proc
MOV AX, num1
mov BX, num2
Sub AX, BX.
Endp.

Main segment
MOV ax, data
MOV DS, ax.
addition( 10, 20
MOV AX, BX
addition 40, 30