

## Serial Communication

- i) Data is transmitted bit after the bit in a single line.
- ii) Data congestion take place.
- iii) Low speed transmission.
- iv) Implementation is not an easy task.
- v) less expensive.
- vi) No crosstalk problem.
- vii) bandwidth of serial wires is much higher.
- viii) work effectively at high frequencies.

## Parallel communication

- i) through group of lines
- ii) No.
- iii) High
- iv) easily implemented in hardware.
- v) more expensive
- vi) having.
- vii) is much lower.
- viii) hard to run at high frequencies.

\*\* Why port 0 is connected to a 10 k $\Omega$  pull up register?

- A pull up resistor is connected to the port to ensure a defined logic level (1) when no external device is driving the port. The 10k $\Omega$  value of the pull-up resistor is typically chosen to provide a balance between ensuring a defined logic level and not consuming too much power.

## DUAL Role of Port 0

- \* Data(D): Port 0 pins P0.0 to P0.3 are used for data transfer either as input or output.

Upper(U): P0.4 to P0.7 are used for upper address decoding and external memory access.

Address(A): P0.4 to P0.7 are used for address decoding for external memory access.

LOWER(L): P0.6 is used as low byte select for external memory.

## DULC Role of Port 2

Besides working as I/O, Port P2 is also used to provide 16-bit address bus for external memory along with port 0. Port P2 is also designed as (A8-A15) while Port 0 provides the lower 8 bits via (A0-A7). In other words we can say that when an 8082 is connected to an external memory (ROM) which can be maximum upto 64kB and this is possible by 16-bit address bus because we know  $2^{16} = 64\text{ kB}$ . Port 2 is used for the upper 8-bit of the 16-bits address. It cannot be used for I/O operations.

ALE (Address Latch Enable) = 1  $\rightarrow$

PO  $\rightarrow$  data bus

= 0  $\rightarrow$  PO  $\rightarrow$  address bus

Alternate functions of Port 3 is best to store data  
state and address pins is known as port 3. This  
port serves some function like interrupt, timer  
input, control signals, serial communication signals  
and TxD etc.

TxD	etc.	
RD	Reads data control output	P3.6 and P3.7
WR	Write "	P3.4 and P3.5
T0	Timers / counter1	
TO	Timers / counter0	
IIN1	→ Interrupt 1	
INT0	→ Interrupt 0	
TxD	→ Transmit data → Receive data	P3.0 and P3.1
RxD		

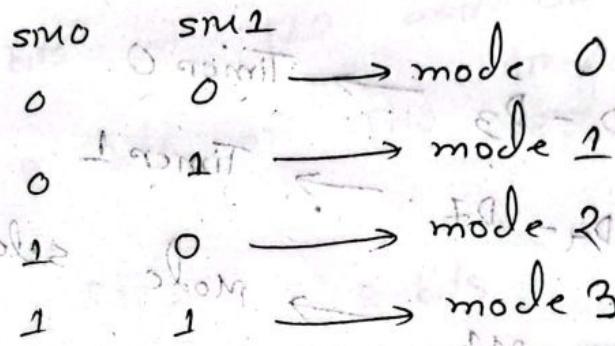
2019

2020 - 3(c)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI
-----	-----	-----	-----	-----	-----	----	----

This serial control Registers (scon).

SM0, SM1 → mode selection



SM2 → Multiprocessor communication bit in mode 2 and mode 3 if SM2 is set, will enable multiprocessor communication bit.

REN → Enable serial Reception transmitted in

TB8 → ninth bit is transmitted in mode 2 and 3.

RB8 → ninth bit is received in mode 2 and 3.

TI → Transmitted interrupt flags.

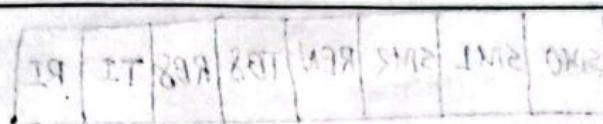
RI → Receiver Interrupt flags.

2019 - 6 (D)

ET

0110 → 0505

## TMOD Register of 8051 microcontroller



D7	D6	D5	D4	D3	D2	D1	D0
GATE	C/T	M1	MO	GATE	C/T	M1	MO

Hence,

D0 - D3 → Timer 0

D4 - D7 → Timer 1

MO, M1 → mode selection

MO      M1 → mode 0 → 13 bit

0      0 → mode 1 → 16 bit

0      1 → mode 2 → 8-bit auto reload

1      0 → mode 3 → split

C/T → used to decide whether a timer or counter. If C/T = 1 then decide to

use counters and if C/T = 0 then decide to use timers.

gate → means of starting and stopping.

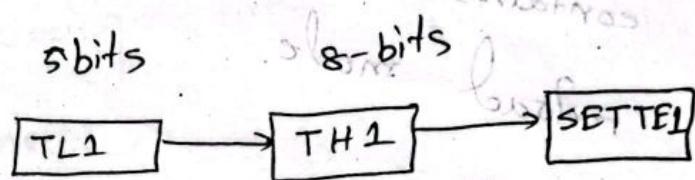
If Gate = 1 then it starts and gate = 0 then it stops.

(2019 - 7(a))

## Timer Mode Operation of 8051 Microcontrollers :

### Timer Mode 0 :

The Mode 0 operation is the 8-bit timers and it is a 13-bit timer/counter. It uses 5 bits of TL0 and or TL1 and all of the 8 bits of TH0 and TH1.



Hence the TLX counts up to 31 and then reset to 00 and increment THX by 1. suppose we load 0 in the timers then the timer will overflow in  $2^{13} \approx 8192$  machine cycles.

### Timer Mode 1 :

Mode 1 is similar to mode 0

except it is a 16-bit mode. In this mode,

THX and TLX both acts as an 8-bit timer.

Since this is a full 16-bit timer we can get a maximum of  $2^{16} \approx 65536$  machine cycles. Timers can be started by →

SET B TR0 → Timer 0

SET B TR1 → Timer 1

When flag bit is set, then going to stop the timers → no short reset

timers → no short reset

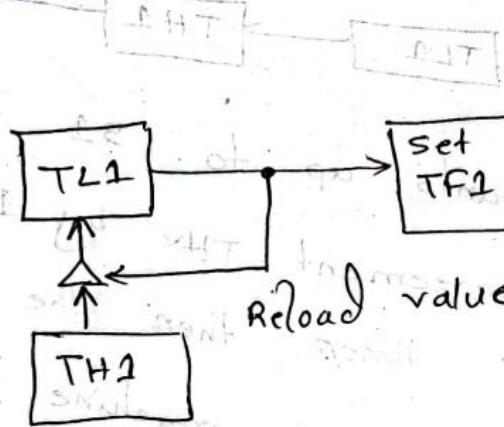
no short reset

## Mode 2:

In this Mode TLx acts as the timer and GT is 8-bit  
THx contains the Reload value.

TLx contains the Reload value.

auto



Observe that TH1 is 8-bit

TL1

8-bit

00

to

FF



It automatically reloads the initial value from registers TH into the TL registers when the counters overflow.

### Mode 3 $\Rightarrow$

Mode 3 is different for Timer0 and

Timer1. When the timer0 is working in mode 3

the T20 will be used as an 8-bit timer/counter.

And TH0 will be used as an 8-bit timer but not the counter. It is called split timer

mode. This mode uses two 8-bit timer/counter

registers to count events separately. It is also

used for measuring time intervals with two

different frequencies.

It has two 8-bit timer/counter

for measuring time intervals with two

different frequencies.

It has two 8-bit timer/counter

It has two 8-bit timer/counter

for measuring time intervals with two

different frequencies.

2019 - 8(c)

→ 8 bits

## Registers Banks of 8051 microcontroller :-

The 8051 microcontroller has four Registers banks, each consisting of eight registers (R0-R7). These registers banks are used to store data and instructions during the execution of a program.

Register Bank 0 : is selected by default, general purpose registers (R0-R7).

Register Bank 1 : This register bank can be selected by setting the bits psw.3 and psw.4 or also general purpose (R0-R7).

Register Bank 2 : setting the bits psw.2 and psw.3

Register Bank 3 : setting the bits psw.2, psw.3 and psw.4 in the psw register.

2018 - 1(c)

The functions of PSW register in 8051 etc.

B <sub>7</sub>	B <sub>6</sub>	B <sub>5</sub>	B <sub>4</sub>	B <sub>3</sub>	B <sub>2</sub>	B <sub>1</sub>	B <sub>0</sub>
CY	AC	FO	RS1	RS0	BV	-	P

$\text{Z} \leftarrow \text{carry}$  → ~~don't care~~ flag

AE = Auxiliary carry flag

$FOF$  = user flag  
 $PRO$  = selected register bank

ov = overflow flag

(-) = reserved

~~P~~ = parity flag

\* pins of 8051 ~~use~~

PSEN : pin(29) : program store enable  
This pin is used to

This pin is enable external program memory.

This is an output pin.

## /EA (pin 31) :

(D) 1 - 8051

- i) external access means it connected to GND that means it connected to externally.

## ALE (pin 30) :

- i) Address Latch Enable pin and is output pin = 30  
ii) if it is an output pin = 30  
active high.  
iii) 8051 port 0 provides both data and address.

## I/O pins :

- i) four ports P0, P1, P2, P3  
ii) Each port uses 8 pins.  
iii) bi-directional

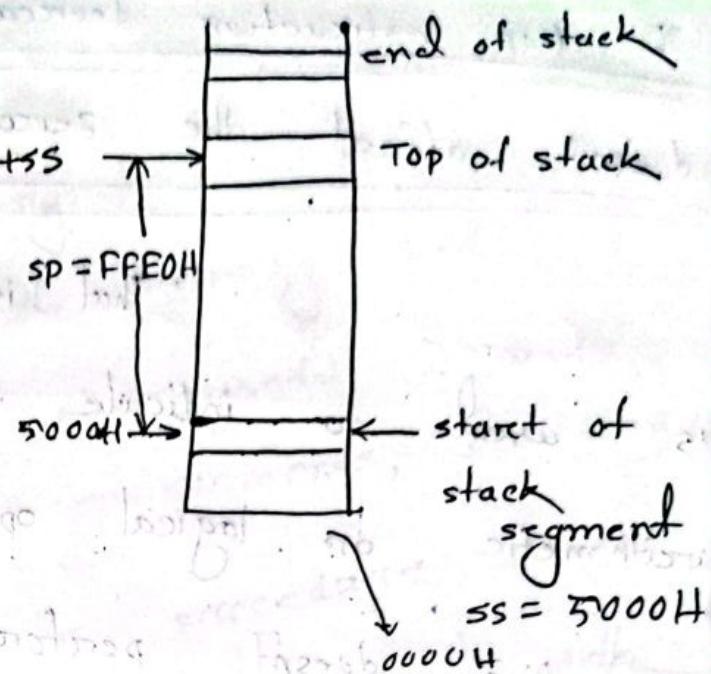
## RST (pin 9) :

- i) reset is active high  
ii) input pin and will reset  
iii) applying RST, all values will be

## Serial Port interrupt in 8051 microcontroller

In an 8051 microcontroller, a serial port interrupt occurs when the microcontroller receives data via its serial port. Having two serial ports named SP0 and SP1. When a serial port interrupt occurs, the microcontroller's CPU stops executing the current instruction and jumps to the interrupt service routine associated with the serial port. The ISR performs the necessary actions to process the received data and returns control to the main program.

\* Given,



physical address → SFFEO

$$(ii) \text{ lower range} = 5000 \times 10 \text{ H} \\ = 50000 \text{ H}$$

$$\text{upper range} = (50000 + \text{ffff}) \text{ H}$$

$$= 5FFFFH$$

size of the stack

$$\text{Maximum size} = 5FFF - 50000$$

= fFFFF

$$\binom{16}{2}_{10} = 64k\beta$$

The loop instruction decreases the cx register but  
doesn't affect the zero flag.

That is correct. The zero flag is used to indicate if the result of an arithmetic or logical operation is zero. The loop instruction doesn't perform any arithmetic or logical operation, it only decrements the cx register, so it doesn't affect the zero flag.

The 'Loop' instruction uses zero flag from the previous instruction to determine if the loop should continue or not. If the zero flag is set, the loop will not execute and the program will continue to the next instruction. If zero flag isn't set, the loop will execute and the program will jump to the specified label.

Significance of reentrant procedure  $\Rightarrow$

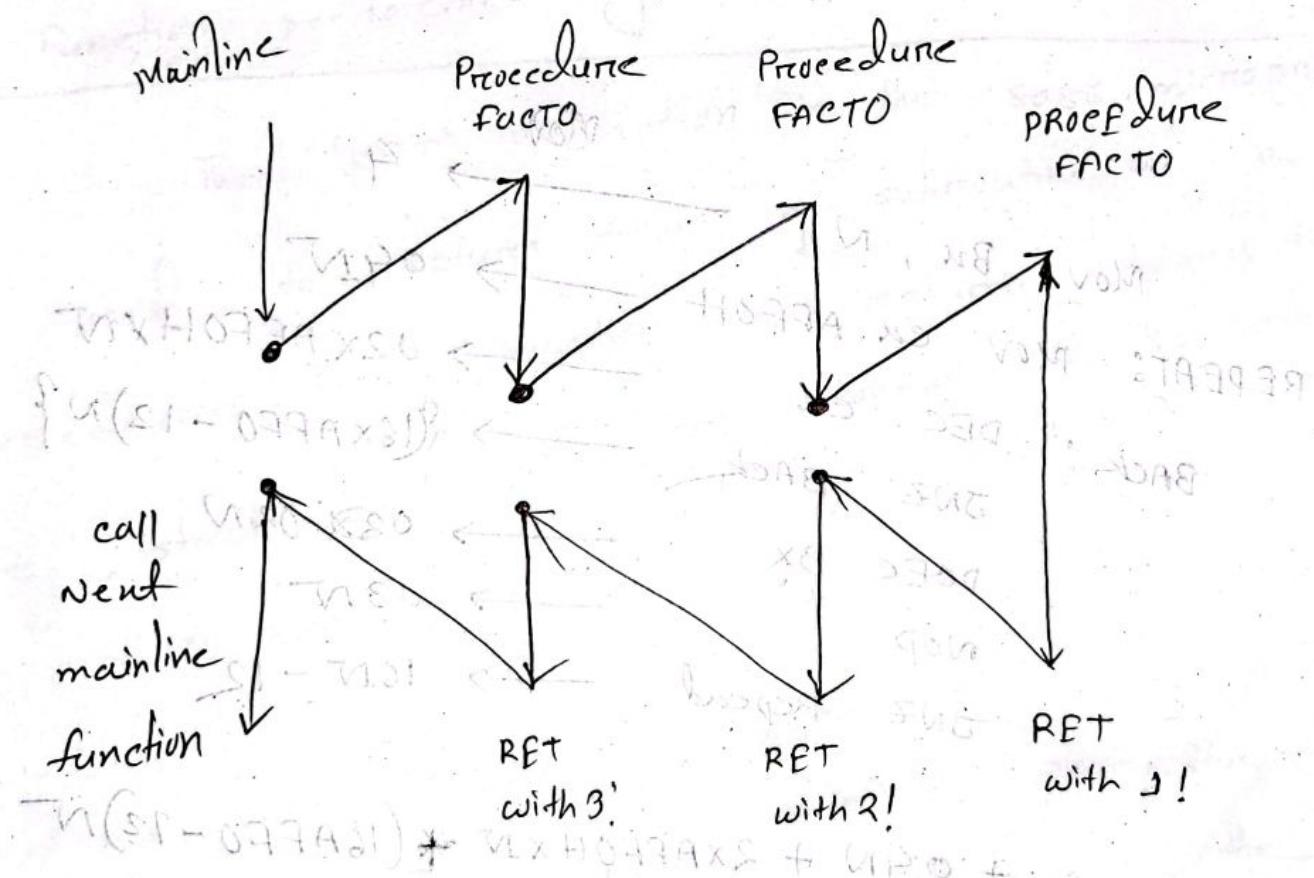
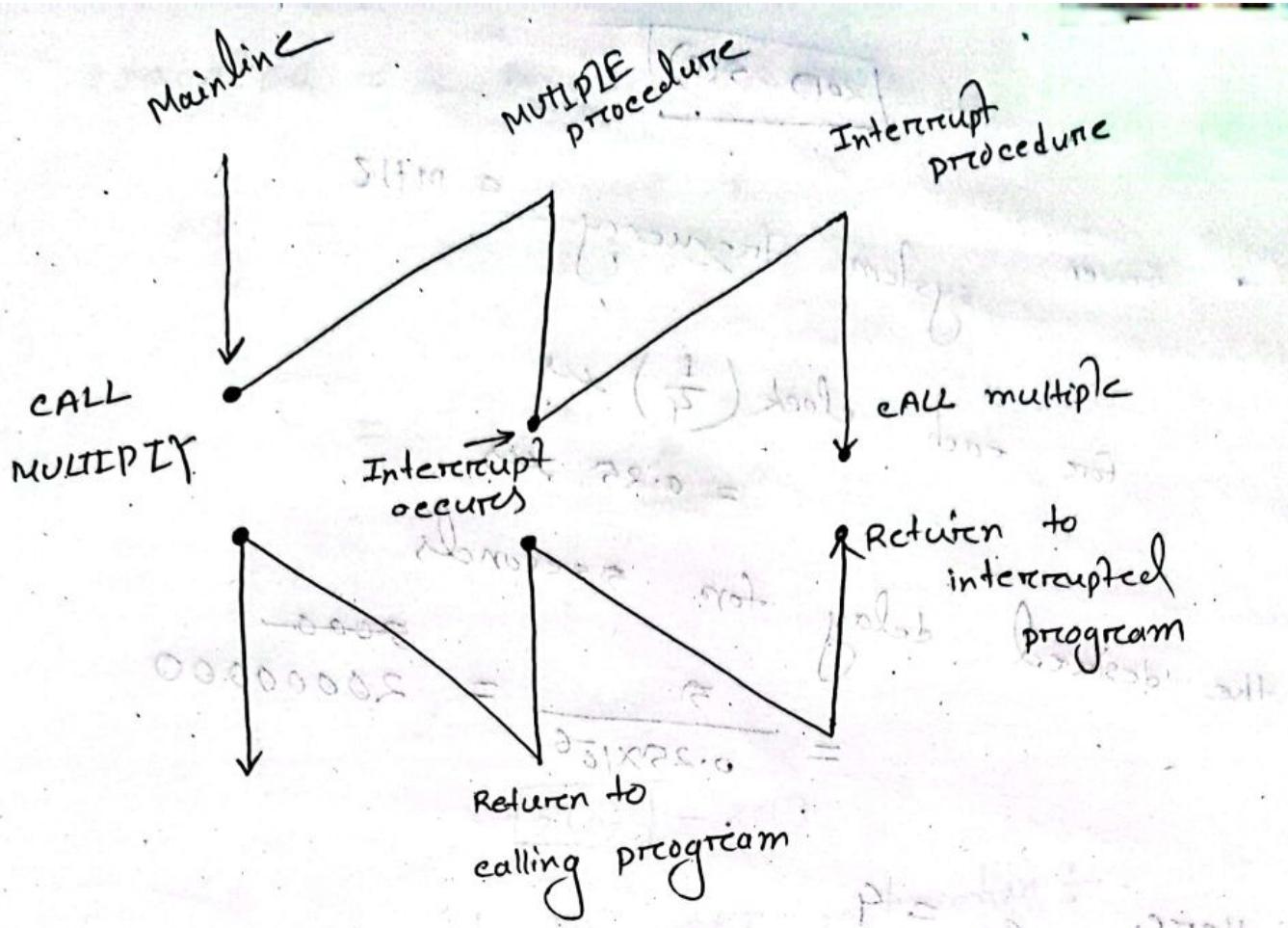
In the 8086 microprocessor, reentrant procedures are significant because they allow for efficient use of memory and processing resources. It is also useful in multi-tasking and multi-threaded environments, as they allow multiple instances of the procedure to be executed simultaneously without interfering with each other. This can improve the overall performance of the system.

For example, consider a program that uses a non-reentrant procedure to calculate the factorial of a number. This procedure may use global variables and may also modify memory locations. If the procedure is called multiple times simultaneously, it causes conflicts and errors because global variable and location

may be modified. On the other hand, a recent ~~concurrent~~  
procedure in 8086 reprocessor uses only local  
variables and doesn't modify memory locations.  
This allows multiple calls.

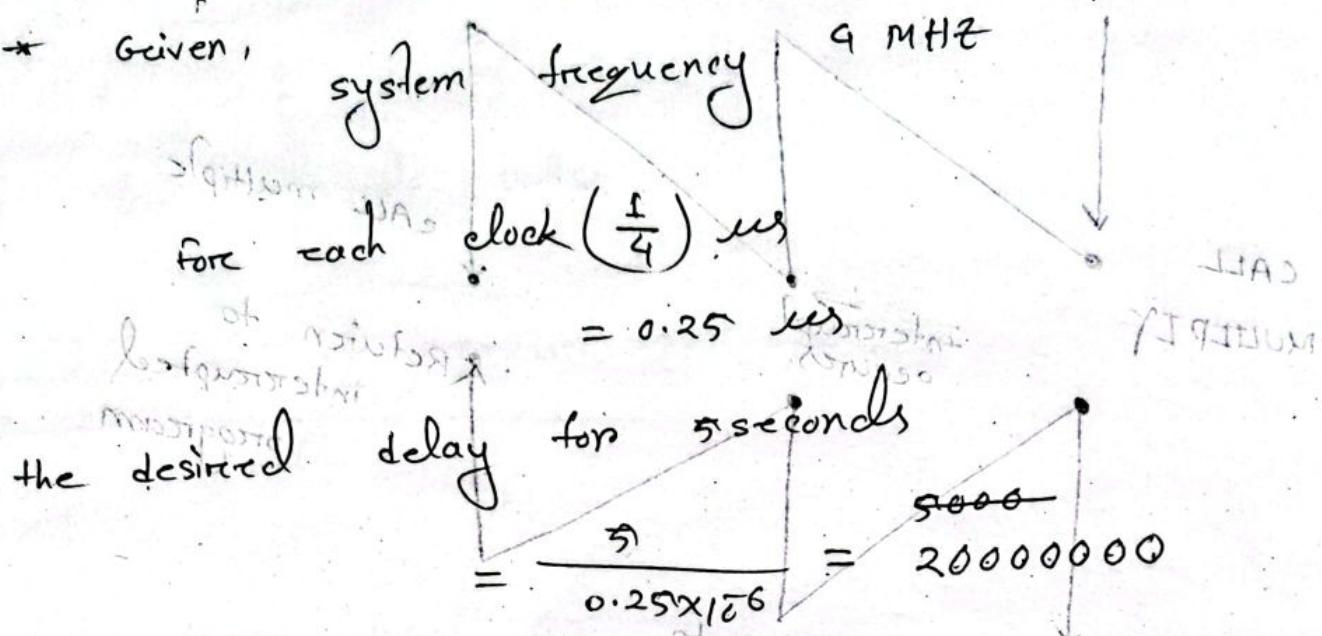
Significance of recursive procedures →  
They are significant because they allow for efficient use of memory and processing resources and provide a simple and elegant solution for solving certain types of problems. It is also useful in solving problems that have a recursive structure, such as tree traversal, backtracking and divide and conquer algorithms.

It is important to note that recursive procedures can consume a lot of stack space, so it is important to have a proper stack management mechanism in place to avoid stack overflow errors.

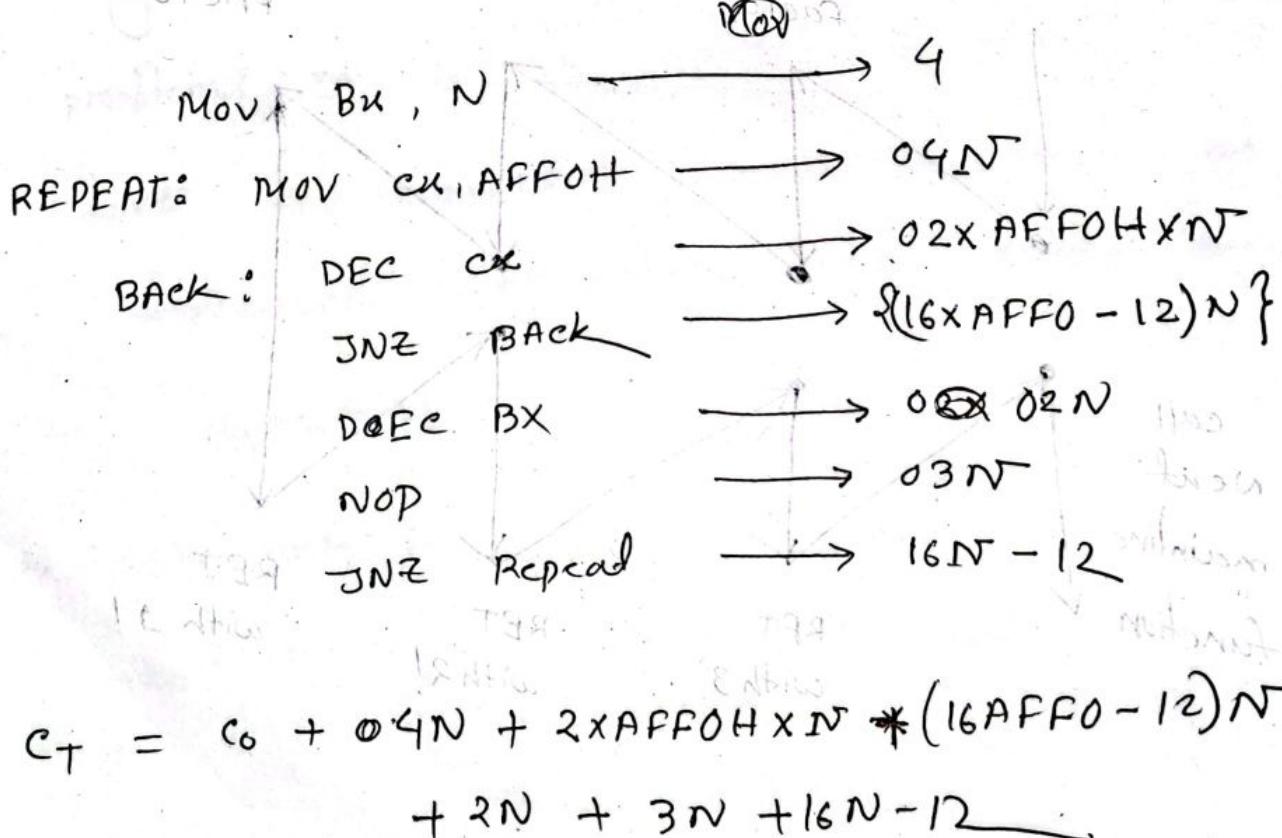


2019-5(c)

\* Given,



Hence,  $c_0 = 9$



$$= c_0 + N \{ 4 + 2 \times AFF0 + 16AFF0 - 12 + 2 + 3 + 16 \} - 12$$

$$= c_0 + N \{ 13 + 2AFF0 + 16AFF0 \}$$

$$20000000 = 4 + n \{ (13 + (18) \times 45040 \}$$

$$\frac{20000000 - 4}{810733} \\ \therefore n =$$

Ans = 24.67

Function performed by RET instruction :-

The 'RET' instruction in the 8086 microprocessor is used to return from a subroutine or a procedure. The instruction pops return address from the stack and transfers control at that address.

When a subroutine or a procedure is called, the return address is pushed onto the stack. This return address is the memory location of the instruction immediately following the call instruction.

When the subroutine is finished executing, the RET instruction is executed, it pops the return address from the stack.

'RET n' this means, the instruction pops the return address from the stack, then adds n to the stack pointer.

In summary, 'RET' is used to transfer control back to the instruction following call instruction.

### \* Jump instructions

that follow the comparison of signed numbers

- i) Jump if Above (JA)
- ii) Jump if Above or Equal (JAE)
- iii) Jump if Below (JB)
- iv) JBE
- v) JE
- vi) JGE
- vii) JGEB
- viii) JL
- ix) JLE
- x) JNE

9(b)

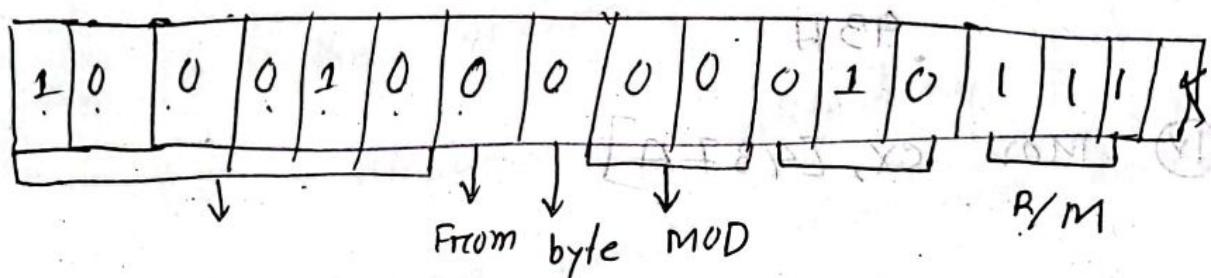
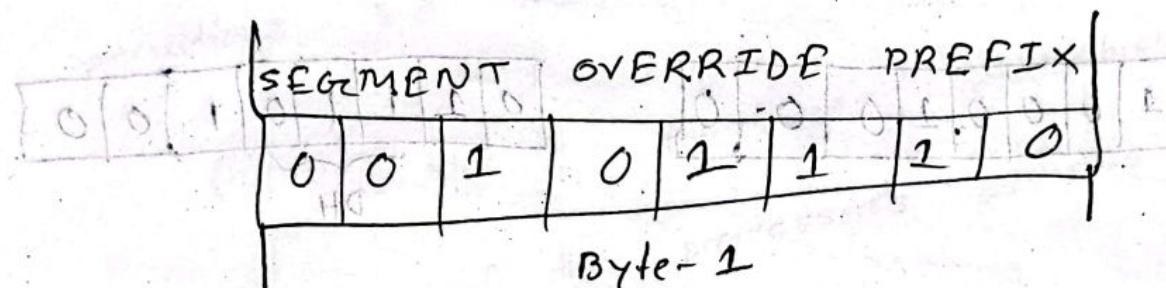
XOR 49 80H 00

Binary code :-

1)  $\text{MOV} \ B \ \text{CS} : [BX], DL$  100010 00000000

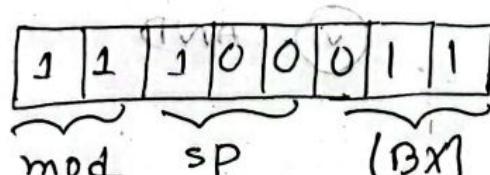
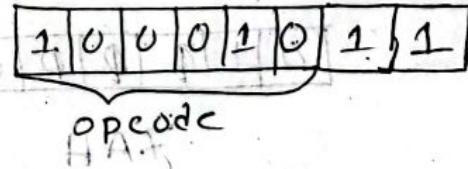
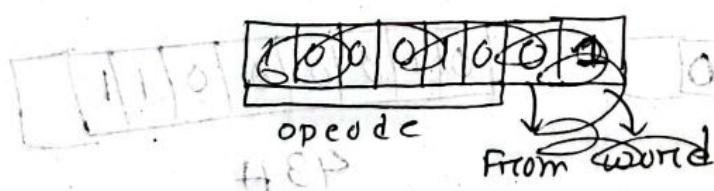
opcode for  $\text{MOV} = 100010$

for CS = 01



011100100010100011  
11 mov sp, BX

Byte - 1 →



HEP SA - 111 = 3000 90

⑪ MOV CL, [BX]

1	0	0	0	1	0	1	0
---	---	---	---	---	---	---	---

[BX]

0	0	0	0	1	1	1	1
---	---	---	---	---	---	---	---

⑫ MOV 43H[SI], DH

1	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

0	1	1	1	0	1	0	0
---	---	---	---	---	---	---	---

DH

0	1	0	0	0	0	1	1
---	---	---	---	---	---	---	---

43H

⑬ MOV CX, [437A]

MV

GOM. shd. mmti

1	0	0	0	1	0	1	1
---	---	---	---	---	---	---	---

0	0	0	0	1	1	1	0
---	---	---	---	---	---	---	---

CX

0	1	1	0	1	0	1	0
---	---	---	---	---	---	---	---

7AH

0	1	0	0	0	0	1	1
---	---	---	---	---	---	---	---

43H

⑭ IN AL, F0H

1	1	1	0	0	1	1	1
---	---	---	---	---	---	---	---

vi AND

OP CODE = 111001 (X8) qe base

## Important characteristic of a Microprocessor :-

### (i) 16-bit microprocessor

(i) instruction set : determines the types of operations that the processor can perform.

(ii) size of the processor's data bus

(iii) size of the processor's address bus

(iv) the number of cores

(v) the architecture

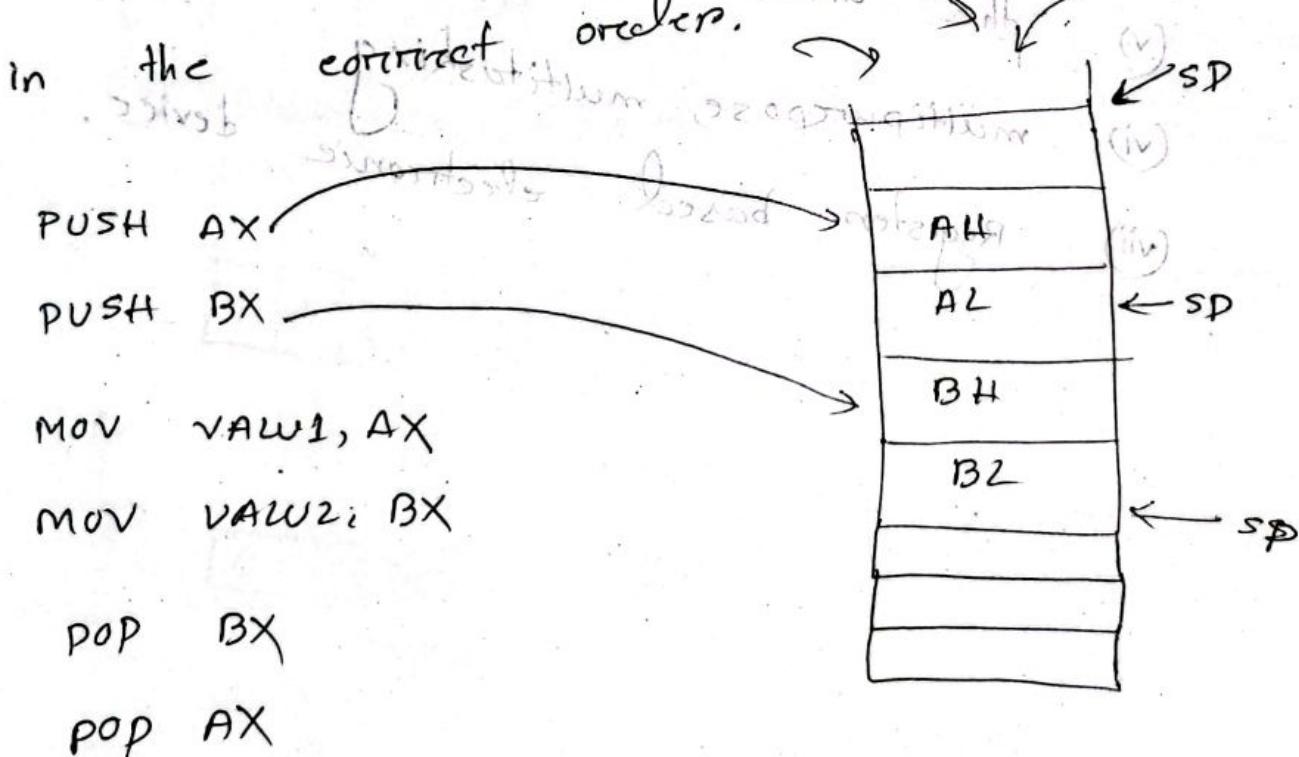
(vi) multipurpose, multitasking

(vii) Registers based electronic device.

Parameter passing to or from a procedure

using stack?

In the 8086 microprocessor, parameters are passed to a procedure using the stack. The stack is a last in first out data structure that is used for storing data temporarily. When a procedure is called, the parameters are pushed onto the stack in reverse order. The procedure then uses the stack pointer (SP) register to access the parameters.



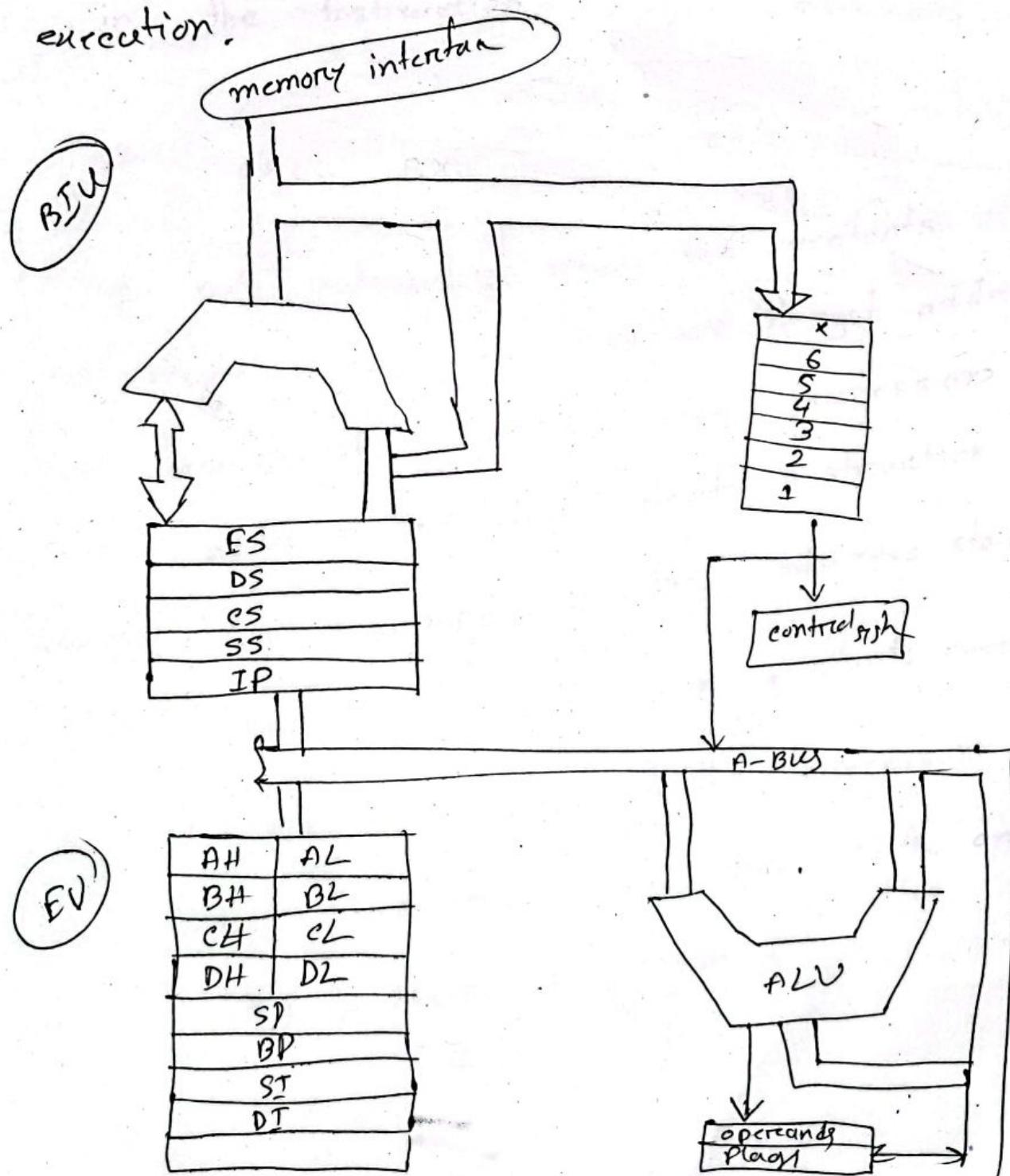
In the 8086 microprocessor, the Bus Interface Unit (BIU) and the Execution Unit (EU) interact with each other to execute instructions and access memory. The BIU is responsible for fetching instructions from memory and providing them to the EU. It also handles memory and I/O operations by communicating with the external bus.

The EU is responsible for decoding and executing the instructions provided by the BIU. It contains ALU which performs mathematical and logical operations as well as the Register Array Unit which holds the processor registers.

When the 8086 microprocessor is executing the instructions, the BIU first fetches an instruction from memory and sends it to the EU. The EU then decodes the instruction and performs

the necessary operations using ALU and RAL.

In summary, the BIU and EU work together to access memory and execute instructions with BIU handling the memory and I/O operations and EU performing the instruction decoding and execution.



Indirect near call using Registers

An indirect near call using a register is a way to call a procedure by specifying the address of the procedure in a register rather than providing the address directly in the instruction.

CALL BX

Firstly the call instruction reads the contents of the BX register as the target address for the procedure call. Then the processor saves the address of the next instruction in the stack and jumps to the address stored in BX register. Then the function starts executing. When the procedure is finished, the processor pops the return address from the stack and returns to the main program.

## Using Memory location $\Rightarrow$

An indirect near call

The more flexible is using a memory location is a way to call a procedure by specifying the address of the procedure in a memory location, rather than providing the instruction.

CALL word ptr [BX]

[BX+1]  $\leftarrow$  [BX]

[BX+1][BX]

CALL [5678][5679]



CALL 1234

PROC	1234
12	5679
34	5678

## limitations of 80286

- (i) it has only a 16-bit ALU
- (ii) maximum segment size is 64kbytes
- (iii) can't easily be switched back and forth between real and protected mode.

LMSW: stands for Load Machine status word.

It is a machine instruction before the 80286 microprocessor that allows the contents of the memory location, register to be loaded from a memory location. LMSW can be used to change the state of the microprocessor such as switching between protected mode and real mode.

is typically used by the operating system to change the state of the CPU when switching between different tasks.

2019 - 8(a)

8086 to 8088

for 80286 switching real mode to protected

mode is

first step is to set the protection enable bit in the MSW register in the 8086.

2nd step: To change bits in the MSW,

execute load machine status word instruction

3rd step: execute an intersegment jump to the start of the main program.

\* \* The jump is necessary to flush the instruction mode the

byte of queue, because in protected

function differently from the way it

does in real mode.

① prepare the descriptor tables

② set the PE

③ paging enable

④ load MSW

⑤ jump

## \* Features of 80286

(i) advanced microprocessors

(ii) 24-bit address bus

(iii) 16 MB. of physical Memory

(iv) Memory Management, virtual memory

management

(v) swapping IN : secondary to physical

(vi) swapping out : physical to secondary

(vii) two operating modes

(viii) Interrupt handling

(ix) Co-processor support

(x) lower power consumption.

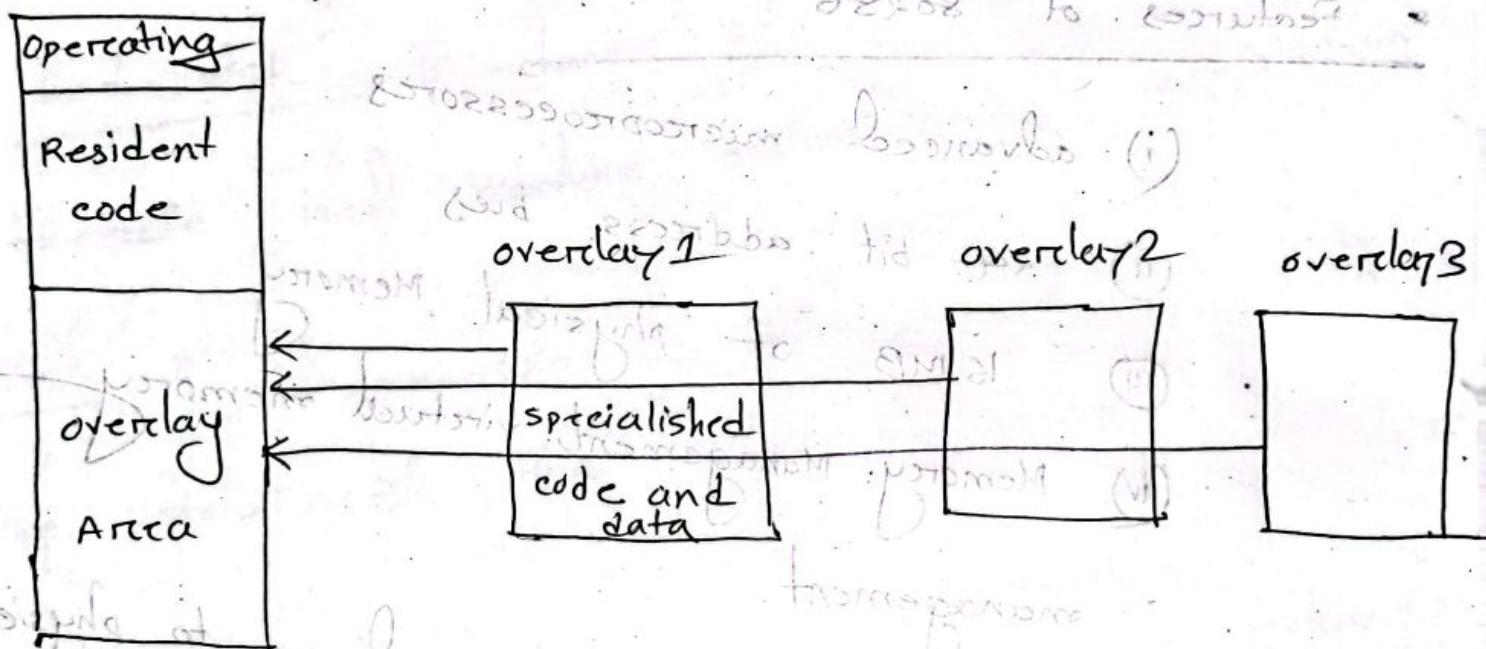
2019 - 8(d)

QUESTION

## Overlay :

Overlay is a method by which large size process can be put into the main memory means if the size of my process is more than the size of memory.

### System Memory

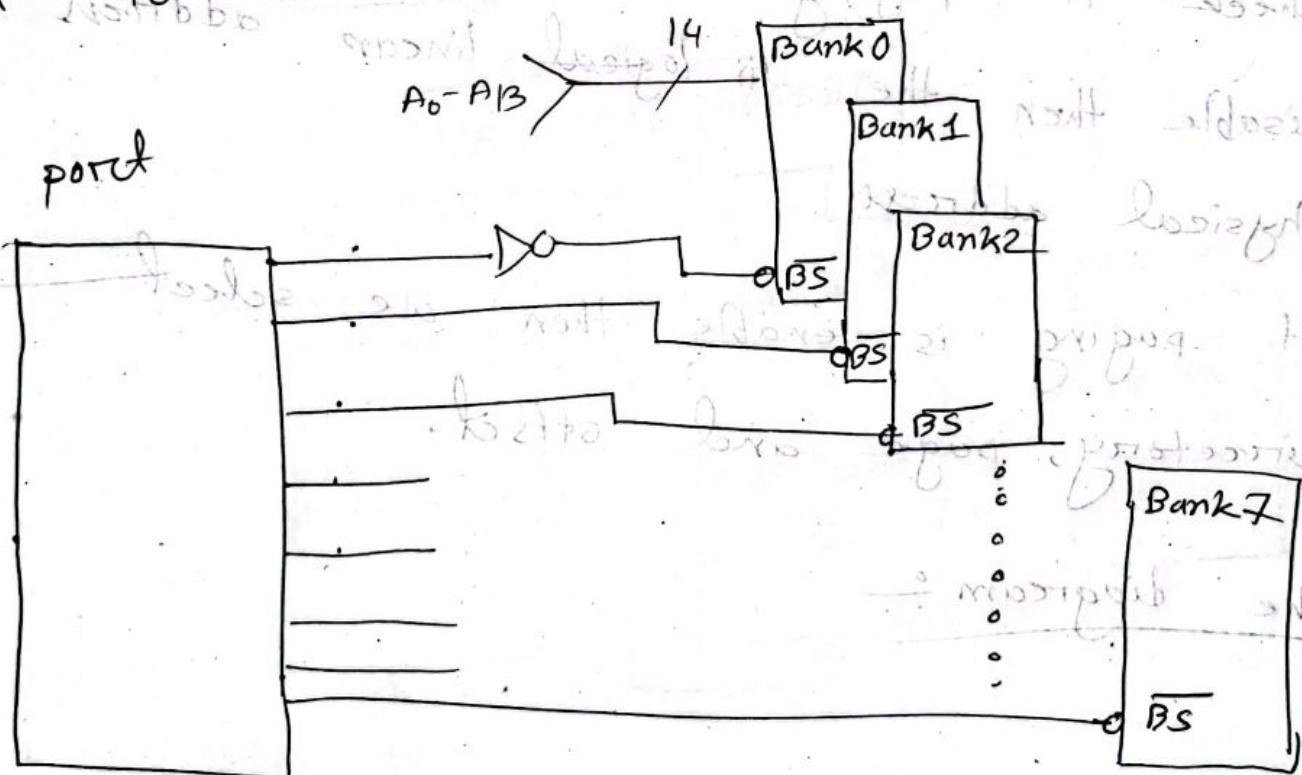


When the assembler reaches a point where it needs the next module then it references as an overlay then overlay n reserved in the memory.

When needs another overlay, it reads that overlay from disk and loads it onto the same overlay area in memory.

## Bank switching =

g. We have 16 address line. We can store 64 kbytes of memory - But using bank switching, we have 14 bits address line and the address space is  $2^{14}$ . There are 8 bank and every bank is 16 bytes. So they hold 128 bytes data together. To switch to bank 1, turns off bank 0 and turns on bank 1 is output to the selection port.



2019 - 7(c)

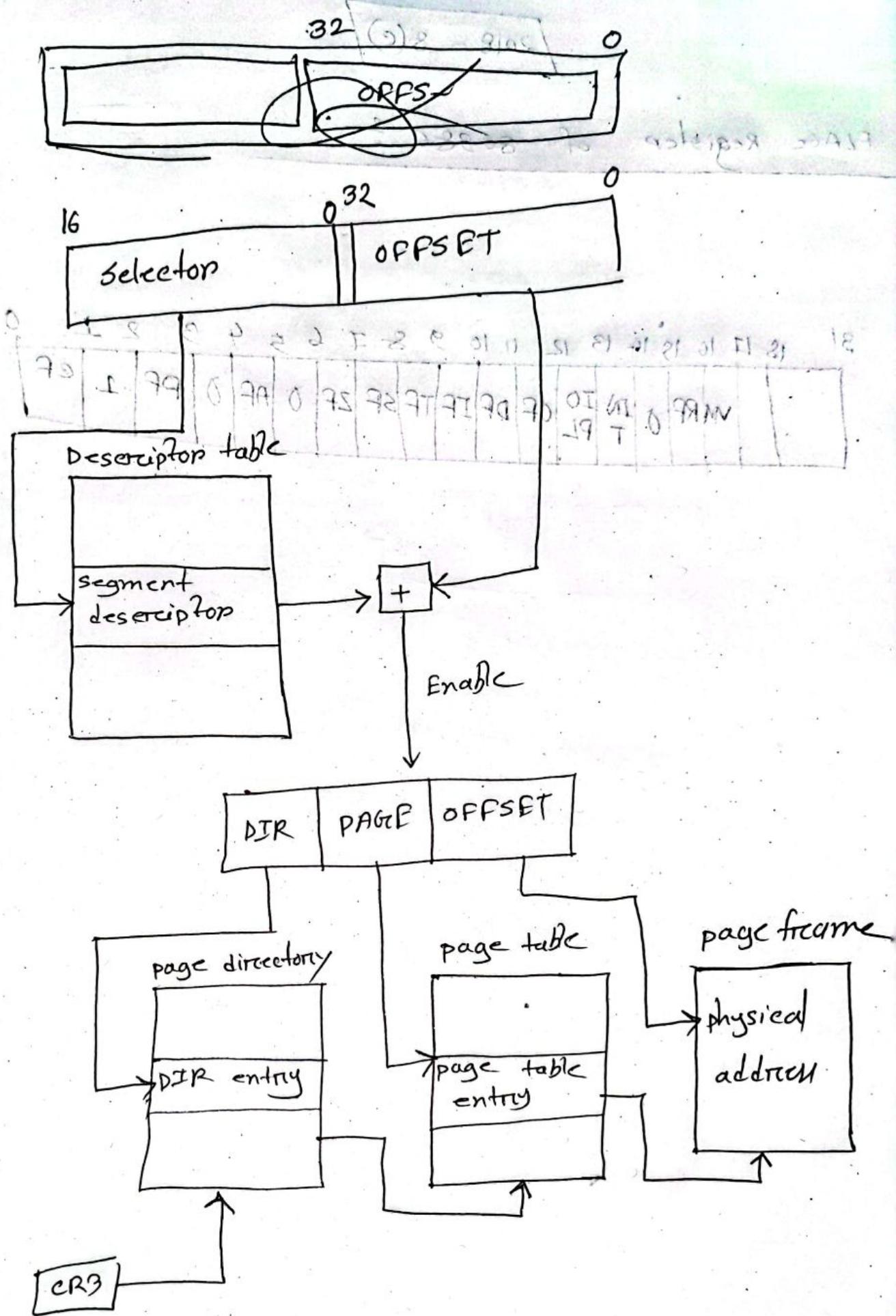
## Paging Memory Management mode of 80386

Here, we have 16-bit selector and 14-bit offset. We choose from descriptor table and the others 2-bit are privileged bit. And the offset are 32-bits.

Then find the page from the descriptors table and offset of this mechanism then add to the offset of this mechanism if paging enable or not. If paging disable then the logical linear address is physical address.

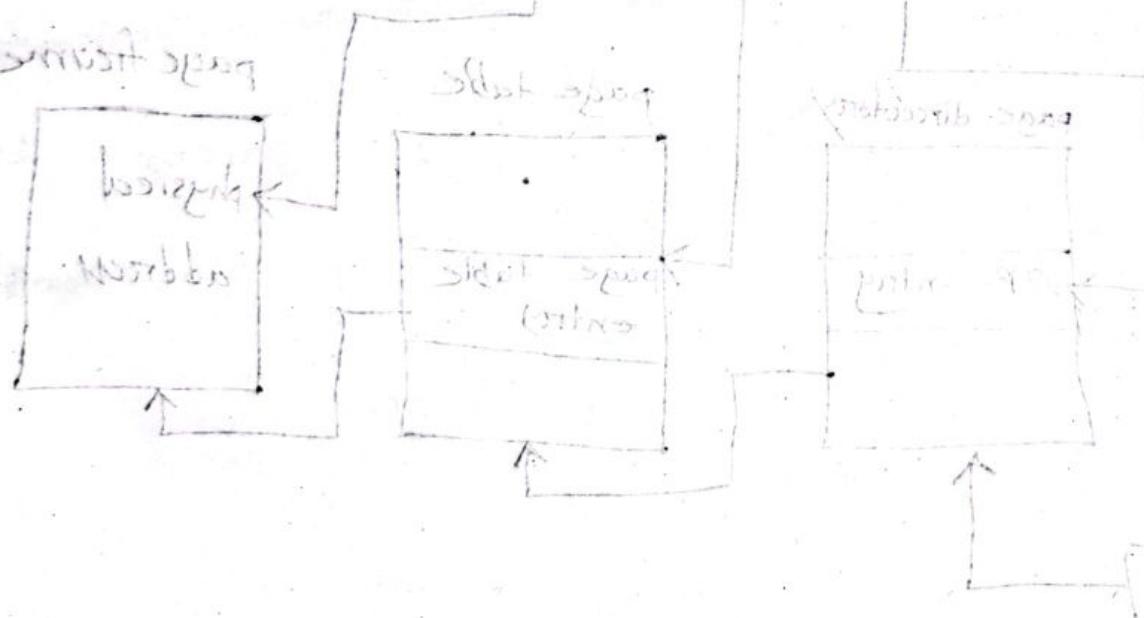
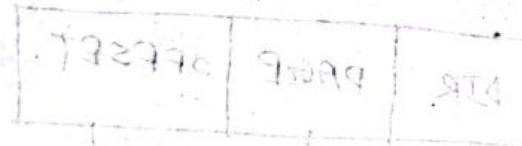
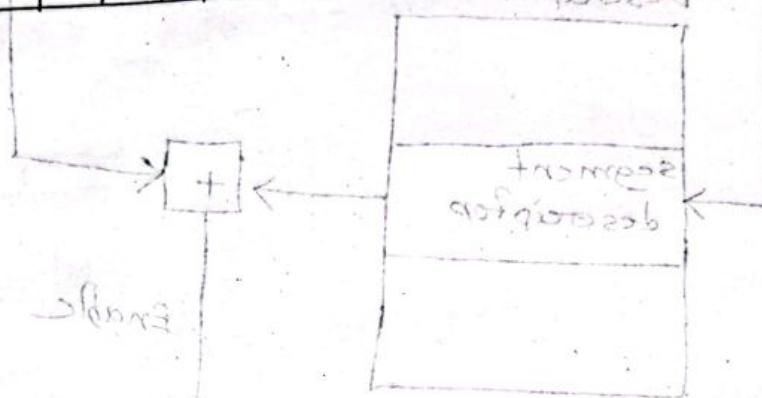
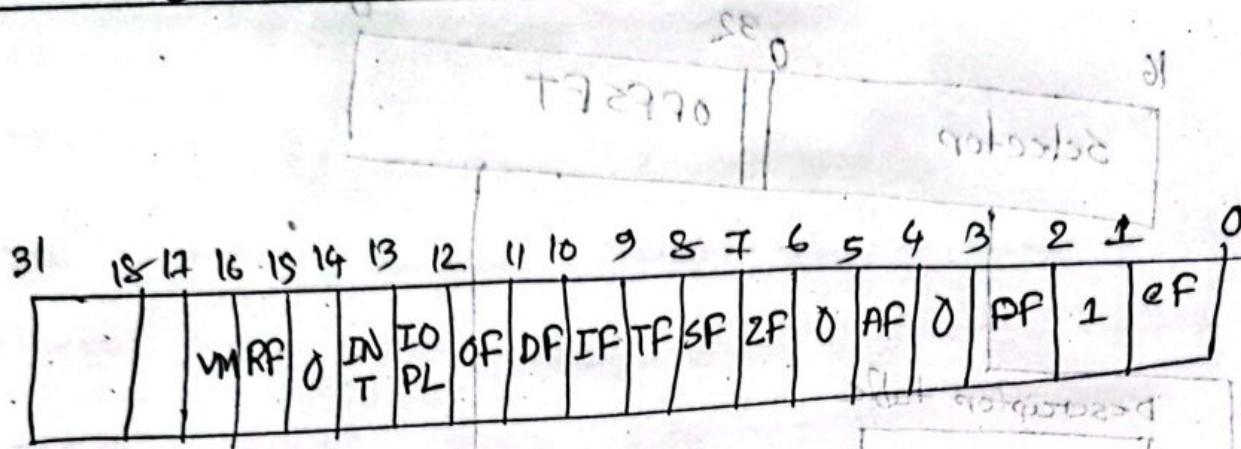
If paging is enable then we select directory, page and offset.

The diagram?



2018 - 8(c)

## FLAGS Register of 80386 :-



## Features of 80386:

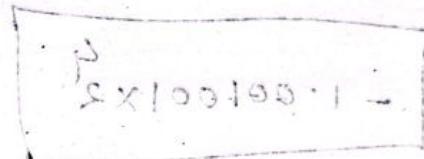
- (i) 32-bit ALU
- (ii) operate directly 32-bit data words
- (iii) 4 Gbytes
- (iv) 1634 segments
- (v) virtual 8086 mode

$SI = \text{Mid address}$

$DI = \text{Mid addresses}$

$FS16+181 \rightarrow \text{8-bit address}$

$P =$



$00010011 =$

~~00010011~~

(A32 - P16)

V803 and cache architecture

Cache memory control



$0112 \rightarrow 000$

$0112$

2017 - 5(b)

11000001110010010000000000000000

1.100000111001001  $\times 10^{16}$

sign bit = 1

exponent bit = 16

Exponent field = 131 - 127  
= 4

$-1.001001 \times 2^4$

= -0.0010010010.00

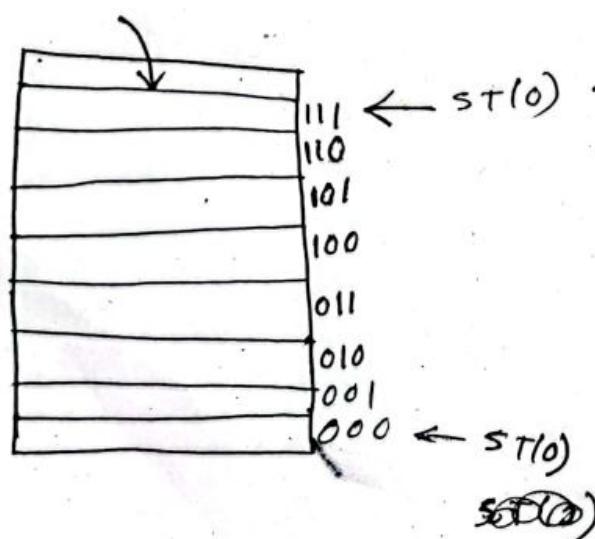
= ~~-18.00~~ Ans. = -18.25 Ans.

$$0x2^1 + 1 \times 2^2 \\ = 1/9$$

$$\frac{25}{0x5d} \times 2 \\ \cancel{\text{---}} \\ 1.0$$

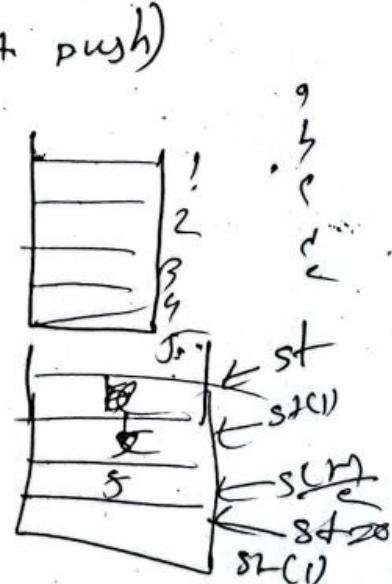
2017 - 6(a)

stack operations for 8087 :



st(0) & first push

st(0)



9  
3  
2  
1

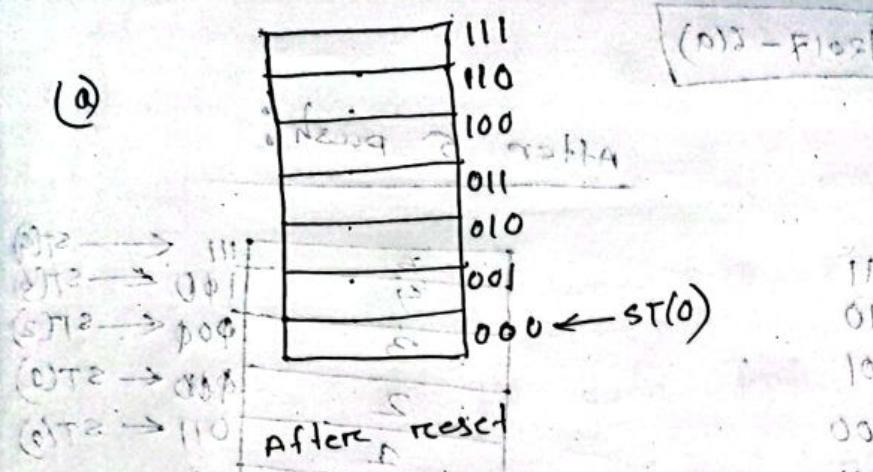
st  
st(1)

st  
st(2)

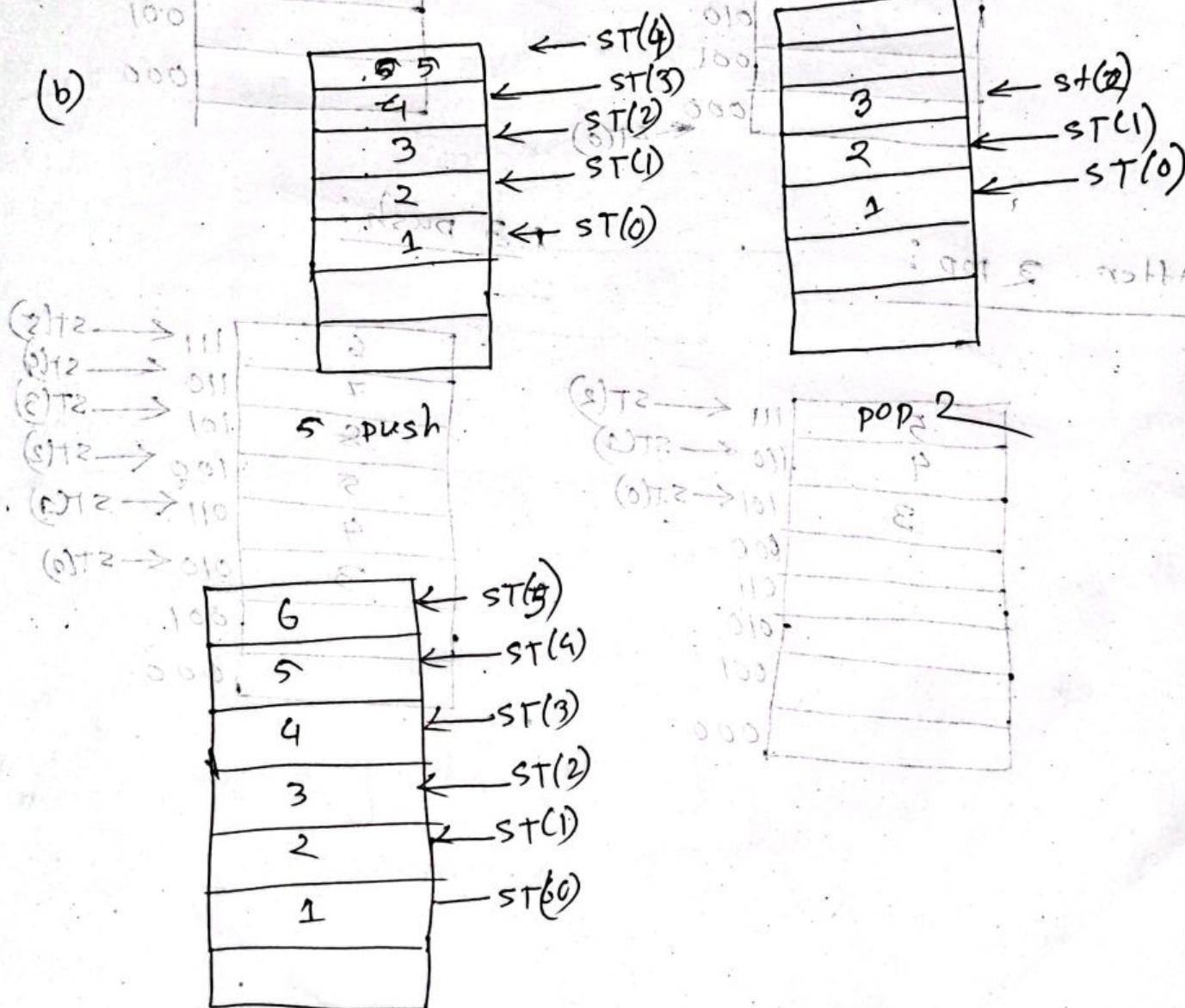
st  
st(3)

st  
st(4)

(a)



(b)



2017 - C(a)

(a) After reset:

111
110
101
100
011
010
001
000

After 2 pop:

5
4
3
2
1
0
1
0
1
0
0
0

After 5 push:

111
110
101
000
011
010
001
000

After 3 push:

6
7
8
5
4
3
2
1
0
1
0
0
0

2017-6(c)

80386 overcomes this:

- i) 80286 having 16 bit data bus but
- ii) 80386 having 32-bit data bus.
- iii) supports virtual memory, protected memory, multitasking.
- iv) improved instruction set.

2017-4(c)

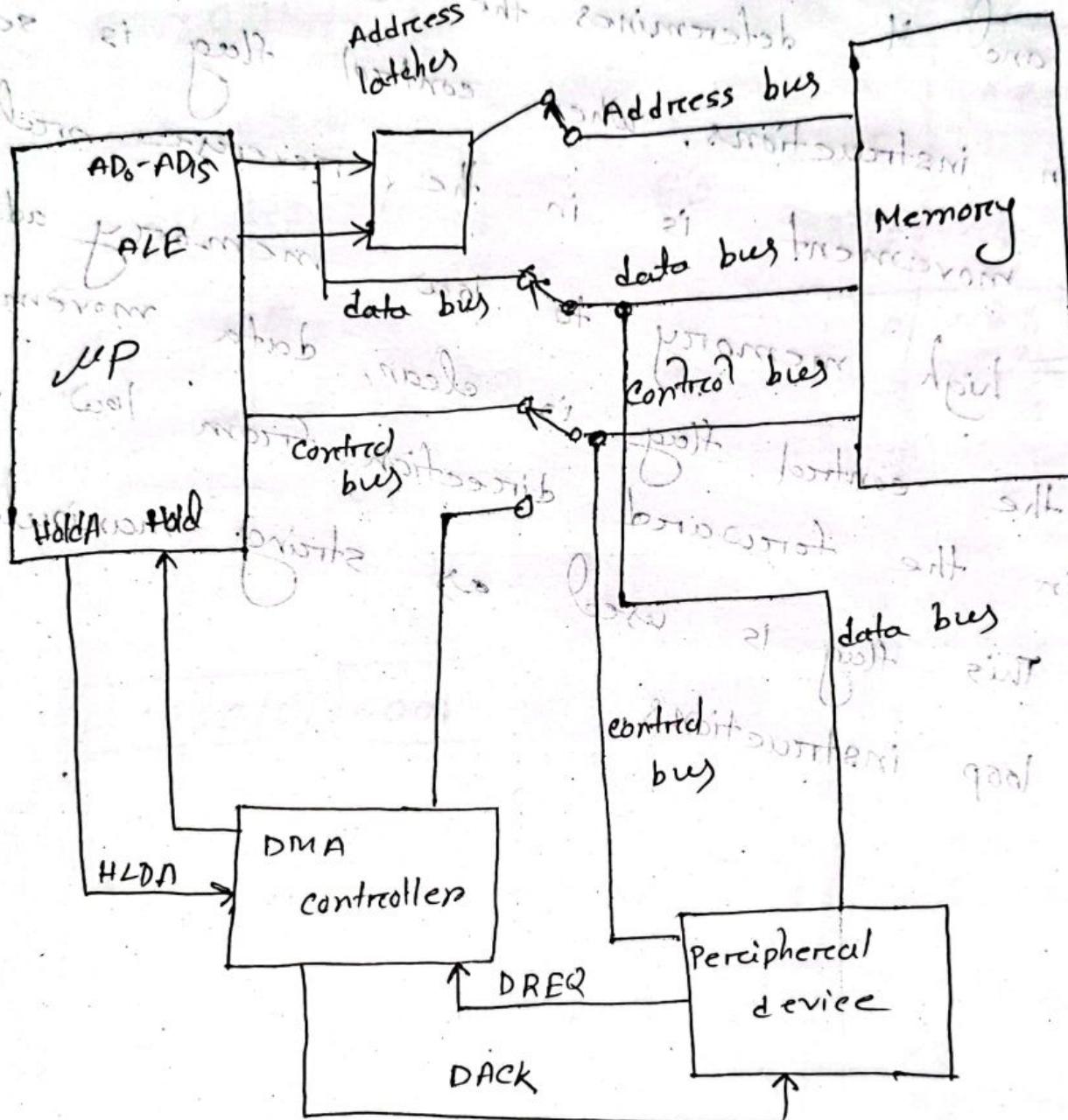
DMA: Direct Memory Access (DMA) is a method that allows an input / output device to send data directly to or from the main memory by passing operation.

At first, if we want to send data to main memory, firstly the CPU would read the first byte of the program and that writes that byte to memory. same for the second byte. This process proves to be inefficient slows down transfer and modify the data.

If we pass the data directly to the main memory from I/O then the process would be much quicker. This is done by DMA.

DMA connects directly to the I/O device at one end and the system bus at other end. Firstly I/O device sends a signal to the DMA-controller that is send request (Request) then DMA controller is send the Hold signal to the CPU. if CPU will not proceed any work then it sends back the signal HLDA then DMA sends ACK signal to the I/O device. Then this I/O

device directly pass the data to the main memory. An address is to be at port B.



control flag in 8086

It is known as the direction flag and it determines the data movement from certain instructions. When control flag is set, data movement is in the reverse order from high memory to low memory address. From low to high, data movement is clear. When the control flag is in the forward direction from low to high. This flag is used as string manipulation loop instructions.



2018-2(2)

8.67

Millions and billions (ii)

= ~~0008011010101~~

=  $1000 \cdot 1010101100001$

=  $1.00010101011100001 \dots E3$

sign bit = 0

the exponent form = 0130. (vi)

①

10000010

0001010101100001...

0.07

$\times 2$

0.34

$\times 2$

0.68

$\times 2$

1.36

$\times 2$

2.72

$\times 2$

5.44

$\times 2$

10.88

$\times 2$

21.76

$\times 2$

43.52

$\times 2$

87.04

$\times 2$

174.08

$\times 2$

348.16

$\times 2$

696.32

$\times 2$

1392.64

$\times 2$

2785.28

$\times 2$

5570.56

$\times 2$

11141.12

$\times 2$

22282.24

$\times 2$

44564.48

$\times 2$

89128.96

$\times 2$

178257.92

$\times 2$

356515.84

$\times 2$

713031.68

$\times 2$

1426063.36

$\times 2$

2852126.72

$\times 2$

5704253.44

$\times 2$

11408506.88

$\times 2$

22817013.76

$\times 2$

45634027.52

$\times 2$

91268055.04

$\times 2$

18253610.08

$\times 2$

36507220.16

$\times 2$

73014440.32

$\times 2$

14602880.64

$\times 2$

29205761.28

$\times 2$

58411522.56

$\times 2$

116823045.12

$\times 2$

233646090.24

$\times 2$

467292180.48

$\times 2$

934584360.96

$\times 2$

1869168721.92

$\times 2$

3738337443.84

$\times 2$

7476674887.68

$\times 2$

14953349755.36

$\times 2$

29906699510.72

$\times 2$

59813399021.44

$\times 2$

119626798042.88

$\times 2$

239253596085.76

$\times 2$

478507192171.52

$\times 2$

957014384343.04

$\times 2$

1914028768686.08

$\times 2$

3828057537372.16

$\times 2$

7656115074744.32

$\times 2$

15312230149488.64

$\times 2$

30624460298977.28

$\times 2$

61248920597954.56

$\times 2$

122497841195909.12

$\times 2$

244995682391818.24

$\times 2$

489991364783636.48

$\times 2$

979982729567272.96

$\times 2$

1959965459134555.92

$\times 2$

3919930918269111.84

$\times 2$

7839861836538223.68

$\times 2$

1567972367307646.32

$\times 2$

3135944734615292.64

$\times 2$

6271889469230585.28

$\times 2$

12543778938461170.56

$\times 2$

25087557876922341.12

$\times 2$

50175115753844682.24

$\times 2$

10035023506768364.48

$\times 2$

20070047013536728.96

$\times 2$

40140094027073457.92

$\times 2$

80280188054146915.84

$\times 2$

16056037610829831.68

$\times 2$

32112075221659663.36

$\times 2$

64224150443319326.72

$\times 2$

128448300886638653.44

$\times 2$

256896601773277306.88

$\times 2$

513793203546554613.76

$\times 2$

102758640709310922.52

$\times 2$

205517281418621845.04

$\times 2$

411034562837243690.08

$\times 2$

822069125674487380.16

$\times 2$

1644138251348946760.32

$\times 2$

3288276502697893520.64

$\times 2$

6576553005395787041.28

$\times 2$

13153106010791574082.56

$\times 2$

26306212021583148165.12

$\times 2$

52612424043166296320.24

$\times 2$

105224848086334592640.48

$\times 2$

210449696172669185280.96

$\times 2$

420899392345338370561.92

$\times 2$

841798784690676741123.84

$\times 2$

1683597569381353482247.68

$\times 2$

3367195138762706964495.36

$\times 2$

6734390277525413928990.72

$\times 2$

1346878055505082785781.44

$\times 2$

2693756111010165571562.88

$\times 2$

5387512222020331143125.76

$\times 2$

1077502444040066228650.52

$\times 2$

2155004888080132457301.04

$\times 2$

4310009776160264914602.08

$\times 2$

8620019552320529829204.16

$\times 2$

1724003910464105965808.32

$\times 2$

3448007820928211931616.64

$\times 2$

6896015641856423863232.12

$\times 2$

13792031283712847726544.24

$\times 2$

27584062567425695453088.48

$\times 2$

55168125134851390906176.96

$\times 2$

## Minimum Mode

- i) single processor
- ii) 8086 is responsible for generating all control signals for memory and I/O ports.
- iii) used for simple applications.
- iv) Pin A<sub>30</sub> is high.
- v) can't add extra co-processor.

## Maximum Mode

- i) no multiprocessors.
- ii) External Bus controllers.  
A<sub>30</sub> = P<sub>0</sub> + P<sub>1</sub>  
B<sub>0</sub> = P<sub>2</sub>  
B<sub>1</sub> = P<sub>3</sub>  
B<sub>2</sub> = P<sub>4</sub>  
B<sub>3</sub> = P<sub>5</sub>  
B<sub>4</sub> = P<sub>6</sub>  
B<sub>5</sub> = P<sub>7</sub>  
B<sub>6</sub> = P<sub>8</sub>  
B<sub>7</sub> = P<sub>9</sub>  
B<sub>8</sub> = P<sub>10</sub>  
B<sub>9</sub> = P<sub>11</sub>  
B<sub>10</sub> = P<sub>12</sub>  
B<sub>11</sub> = P<sub>13</sub>  
B<sub>12</sub> = P<sub>14</sub>  
B<sub>13</sub> = P<sub>15</sub>  
B<sub>14</sub> = P<sub>16</sub>  
B<sub>15</sub> = P<sub>17</sub>  
B<sub>16</sub> = P<sub>18</sub>  
B<sub>17</sub> = P<sub>19</sub>  
B<sub>18</sub> = P<sub>20</sub>  
B<sub>19</sub> = P<sub>21</sub>  
B<sub>20</sub> = P<sub>22</sub>  
B<sub>21</sub> = P<sub>23</sub>  
B<sub>22</sub> = P<sub>24</sub>  
B<sub>23</sub> = P<sub>25</sub>  
B<sub>24</sub> = P<sub>26</sub>  
B<sub>25</sub> = P<sub>27</sub>  
B<sub>26</sub> = P<sub>28</sub>  
B<sub>27</sub> = P<sub>29</sub>  
B<sub>28</sub> = P<sub>30</sub>  
B<sub>29</sub> = P<sub>31</sub>  
B<sub>30</sub> = P<sub>32</sub>  
B<sub>31</sub> = P<sub>33</sub>  
B<sub>32</sub> = P<sub>34</sub>  
B<sub>33</sub> = P<sub>35</sub>  
B<sub>34</sub> = P<sub>36</sub>  
B<sub>35</sub> = P<sub>37</sub>  
B<sub>36</sub> = P<sub>38</sub>  
B<sub>37</sub> = P<sub>39</sub>  
B<sub>38</sub> = P<sub>40</sub>  
B<sub>39</sub> = P<sub>41</sub>  
B<sub>40</sub> = P<sub>42</sub>  
B<sub>41</sub> = P<sub>43</sub>  
B<sub>42</sub> = P<sub>44</sub>  
B<sub>43</sub> = P<sub>45</sub>  
B<sub>44</sub> = P<sub>46</sub>  
B<sub>45</sub> = P<sub>47</sub>  
B<sub>46</sub> = P<sub>48</sub>  
B<sub>47</sub> = P<sub>49</sub>  
B<sub>48</sub> = P<sub>50</sub>  
B<sub>49</sub> = P<sub>51</sub>  
B<sub>50</sub> = P<sub>52</sub>  
B<sub>51</sub> = P<sub>53</sub>  
B<sub>52</sub> = P<sub>54</sub>  
B<sub>53</sub> = P<sub>55</sub>  
B<sub>54</sub> = P<sub>56</sub>  
B<sub>55</sub> = P<sub>57</sub>  
B<sub>56</sub> = P<sub>58</sub>  
B<sub>57</sub> = P<sub>59</sub>  
B<sub>58</sub> = P<sub>60</sub>  
B<sub>59</sub> = P<sub>61</sub>  
B<sub>60</sub> = P<sub>62</sub>  
B<sub>61</sub> = P<sub>63</sub>  
B<sub>62</sub> = P<sub>64</sub>  
B<sub>63</sub> = P<sub>65</sub>  
B<sub>64</sub> = P<sub>66</sub>  
B<sub>65</sub> = P<sub>67</sub>  
B<sub>66</sub> = P<sub>68</sub>  
B<sub>67</sub> = P<sub>69</sub>  
B<sub>68</sub> = P<sub>70</sub>  
B<sub>69</sub> = P<sub>71</sub>  
B<sub>70</sub> = P<sub>72</sub>  
B<sub>71</sub> = P<sub>73</sub>  
B<sub>72</sub> = P<sub>74</sub>  
B<sub>73</sub> = P<sub>75</sub>  
B<sub>74</sub> = P<sub>76</sub>  
B<sub>75</sub> = P<sub>77</sub>  
B<sub>76</sub> = P<sub>78</sub>  
B<sub>77</sub> = P<sub>79</sub>  
B<sub>78</sub> = P<sub>80</sub>  
B<sub>79</sub> = P<sub>81</sub>  
B<sub>80</sub> = P<sub>82</sub>  
B<sub>81</sub> = P<sub>83</sub>  
B<sub>82</sub> = P<sub>84</sub>  
B<sub>83</sub> = P<sub>85</sub>  
B<sub>84</sub> = P<sub>86</sub>  
B<sub>85</sub> = P<sub>87</sub>  
B<sub>86</sub> = P<sub>88</sub>  
B<sub>87</sub> = P<sub>89</sub>  
B<sub>88</sub> = P<sub>90</sub>  
B<sub>89</sub> = P<sub>91</sub>  
B<sub>90</sub> = P<sub>92</sub>  
B<sub>91</sub> = P<sub>93</sub>  
B<sub>92</sub> = P<sub>94</sub>  
B<sub>93</sub> = P<sub>95</sub>  
B<sub>94</sub> = P<sub>96</sub>  
B<sub>95</sub> = P<sub>97</sub>  
B<sub>96</sub> = P<sub>98</sub>  
B<sub>97</sub> = P<sub>99</sub>  
B<sub>98</sub> = P<sub>100</sub>  
B<sub>99</sub> = P<sub>101</sub>  
B<sub>100</sub> = P<sub>102</sub>  
B<sub>101</sub> = P<sub>103</sub>  
B<sub>102</sub> = P<sub>104</sub>  
B<sub>103</sub> = P<sub>105</sub>  
B<sub>104</sub> = P<sub>106</sub>  
B<sub>105</sub> = P<sub>107</sub>  
B<sub>106</sub> = P<sub>108</sub>  
B<sub>107</sub> = P<sub>109</sub>  
B<sub>108</sub> = P<sub>110</sub>  
B<sub>109</sub> = P<sub>111</sub>  
B<sub>110</sub> = P<sub>112</sub>  
B<sub>111</sub> = P<sub>113</sub>  
B<sub>112</sub> = P<sub>114</sub>  
B<sub>113</sub> = P<sub>115</sub>  
B<sub>114</sub> = P<sub>116</sub>  
B<sub>115</sub> = P<sub>117</sub>  
B<sub>116</sub> = P<sub>118</sub>  
B<sub>117</sub> = P<sub>119</sub>  
B<sub>118</sub> = P<sub>120</sub>  
B<sub>119</sub> = P<sub>121</sub>  
B<sub>120</sub> = P<sub>122</sub>  
B<sub>121</sub> = P<sub>123</sub>  
B<sub>122</sub> = P<sub>124</sub>  
B<sub>123</sub> = P<sub>125</sub>  
B<sub>124</sub> = P<sub>126</sub>  
B<sub>125</sub> = P<sub>127</sub>  
B<sub>126</sub> = P<sub>128</sub>  
B<sub>127</sub> = P<sub>129</sub>  
B<sub>128</sub> = P<sub>130</sub>  
B<sub>129</sub> = P<sub>131</sub>  
B<sub>130</sub> = P<sub>132</sub>  
B<sub>131</sub> = P<sub>133</sub>  
B<sub>132</sub> = P<sub>134</sub>  
B<sub>133</sub> = P<sub>135</sub>  
B<sub>134</sub> = P<sub>136</sub>  
B<sub>135</sub> = P<sub>137</sub>  
B<sub>136</sub> = P<sub>138</sub>  
B<sub>137</sub> = P<sub>139</sub>  
B<sub>138</sub> = P<sub>140</sub>  
B<sub>139</sub> = P<sub>141</sub>  
B<sub>140</sub> = P<sub>142</sub>  
B<sub>141</sub> = P<sub>143</sub>  
B<sub>142</sub> = P<sub>144</sub>  
B<sub>143</sub> = P<sub>145</sub>  
B<sub>144</sub> = P<sub>146</sub>  
B<sub>145</sub> = P<sub>147</sub>  
B<sub>146</sub> = P<sub>148</sub>  
B<sub>147</sub> = P<sub>149</sub>  
B<sub>148</sub> = P<sub>150</sub>  
B<sub>149</sub> = P<sub>151</sub>  
B<sub>150</sub> = P<sub>152</sub>  
B<sub>151</sub> = P<sub>153</sub>  
B<sub>152</sub> = P<sub>154</sub>  
B<sub>153</sub> = P<sub>155</sub>  
B<sub>154</sub> = P<sub>156</sub>  
B<sub>155</sub> = P<sub>157</sub>  
B<sub>156</sub> = P<sub>158</sub>  
B<sub>157</sub> = P<sub>159</sub>  
B<sub>158</sub> = P<sub>160</sub>  
B<sub>159</sub> = P<sub>161</sub>  
B<sub>160</sub> = P<sub>162</sub>  
B<sub>161</sub> = P<sub>163</sub>  
B<sub>162</sub> = P<sub>164</sub>  
B<sub>163</sub> = P<sub>165</sub>  
B<sub>164</sub> = P<sub>166</sub>  
B<sub>165</sub> = P<sub>167</sub>  
B<sub>166</sub> = P<sub>168</sub>  
B<sub>167</sub> = P<sub>169</sub>  
B<sub>168</sub> = P<sub>170</sub>  
B<sub>169</sub> = P<sub>171</sub>  
B<sub>170</sub> = P<sub>172</sub>  
B<sub>171</sub> = P<sub>173</sub>  
B<sub>172</sub> = P<sub>174</sub>  
B<sub>173</sub> = P<sub>175</sub>  
B<sub>174</sub> = P<sub>176</sub>  
B<sub>175</sub> = P<sub>177</sub>  
B<sub>176</sub> = P<sub>178</sub>  
B<sub>177</sub> = P<sub>179</sub>  
B<sub>178</sub> = P<sub>180</sub>  
B<sub>179</sub> = P<sub>181</sub>  
B<sub>180</sub> = P<sub>182</sub>  
B<sub>181</sub> = P<sub>183</sub>  
B<sub>182</sub> = P<sub>184</sub>  
B<sub>183</sub> = P<sub>185</sub>  
B<sub>184</sub> = P<sub>186</sub>  
B<sub>185</sub> = P<sub>187</sub>  
B<sub>186</sub> = P<sub>188</sub>  
B<sub>187</sub> = P<sub>189</sub>  
B<sub>188</sub> = P<sub>190</sub>  
B<sub>189</sub> = P<sub>191</sub>  
B<sub>190</sub> = P<sub>192</sub>  
B<sub>191</sub> = P<sub>193</sub>  
B<sub>192</sub> = P<sub>194</sub>  
B<sub>193</sub> = P<sub>195</sub>  
B<sub>194</sub> = P<sub>196</sub>  
B<sub>195</sub> = P<sub>197</sub>  
B<sub>196</sub> = P<sub>198</sub>  
B<sub>197</sub> = P<sub>199</sub>  
B<sub>198</sub> = P<sub>200</sub>  
B<sub>199</sub> = P<sub>201</sub>  
B<sub>200</sub> = P<sub>202</sub>  
B<sub>201</sub> = P<sub>203</sub>  
B<sub>202</sub> = P<sub>204</sub>  
B<sub>203</sub> = P<sub>205</sub>  
B<sub>204</sub> = P<sub>206</sub>  
B<sub>205</sub> = P<sub>207</sub>  
B<sub>206</sub> = P<sub>208</sub>  
B<sub>207</sub> = P<sub>209</sub>  
B<sub>208</sub> = P<sub>210</sub>  
B<sub>209</sub> = P<sub>211</sub>  
B<sub>210</sub> = P<sub>212</sub>  
B<sub>211</sub> = P<sub>213</sub>  
B<sub>212</sub> = P<sub>214</sub>  
B<sub>213</sub> = P<sub>215</sub>  
B<sub>214</sub> = P<sub>216</sub>  
B<sub>215</sub> = P<sub>217</sub>  
B<sub>216</sub> = P<sub>218</sub>  
B<sub>217</sub> = P<sub>219</sub>  
B<sub>218</sub> = P<sub>220</sub>  
B<sub>219</sub> = P<sub>221</sub>  
B<sub>220</sub> = P<sub>222</sub>  
B<sub>221</sub> = P<sub>223</sub>  
B<sub>222</sub> = P<sub>224</sub>  
B<sub>223</sub> = P<sub>225</sub>  
B<sub>224</sub> = P<sub>226</sub>  
B<sub>225</sub> = P<sub>227</sub>  
B<sub>226</sub> = P<sub>228</sub>  
B<sub>227</sub> = P<sub>229</sub>  
B<sub>228</sub> = P<sub>230</sub>  
B<sub>229</sub> = P<sub>231</sub>  
B<sub>230</sub> = P<sub>232</sub>  
B<sub>231</sub> = P<sub>233</sub>  
B<sub>232</sub> = P<sub>234</sub>  
B<sub>233</sub> = P<sub>235</sub>  
B<sub>234</sub> = P<sub>236</sub>  
B<sub>235</sub> = P<sub>237</sub>  
B<sub>236</sub> = P<sub>238</sub>  
B<sub>237</sub> = P<sub>239</sub>  
B<sub>238</sub> = P<sub>240</sub>  
B<sub>239</sub> = P<sub>241</sub>  
B<sub>240</sub> = P<sub>242</sub>  
B<sub>241</sub> = P<sub>243</sub>  
B<sub>242</sub> = P<sub>244</sub>  
B<sub>243</sub> = P<sub>245</sub>  
B<sub>244</sub> = P<sub>246</sub>  
B<sub>245</sub> = P<sub>247</sub>  
B<sub>246</sub> = P<sub>248</sub>  
B<sub>247</sub> = P<sub>249</sub>  
B<sub>248</sub> = P<sub>250</sub>  
B<sub>249</sub> = P<sub>251</sub>  
B<sub>250</sub> = P<sub>252</sub>  
B<sub>251</sub> = P<sub>253</sub>  
B<sub>252</sub> = P<sub>254</sub>  
B<sub>253</sub> = P<sub>255</sub>  
B<sub>254</sub> = P<sub>256</sub>  
B<sub>255</sub> = P<sub>257</sub>  
B<sub>256</sub> = P<sub>258</sub>  
B<sub>257</sub> = P<sub>259</sub>  
B<sub>258</sub> = P<sub>260</sub>  
B<sub>259</sub> = P<sub>261</sub>  
B<sub>260</sub> = P<sub>262</sub>  
B<sub>261</sub> = P<sub>263</sub>  
B<sub>262</sub> = P<sub>264</sub>  
B<sub>263</sub> = P<sub>265</sub>  
B<sub>264</sub> = P<sub>266</sub>  
B<sub>265</sub> = P<sub>267</sub>  
B<sub>266</sub> = P<sub>268</sub>  
B<sub>267</sub> = P<sub>269</sub>  
B<sub>268</sub> = P<sub>270</sub>  
B<sub>269</sub> = P<sub>271</sub>  
B<sub>270</sub> = P<sub>272</sub>  
B<sub>271</sub> = P<sub>273</sub>  
B<sub>272</sub> = P<sub>274</sub>  
B<sub>273</sub> = P<sub>275</sub>  
B<sub>274</sub> = P<sub>276</sub>  
B<sub>275</sub> = P<sub>277</sub>  
B<sub>276</sub> = P<sub>278</sub>  
B<sub>277</sub> = P<sub>279</sub>  
B<sub>278</sub> = P<sub>280</sub>  
B<sub>279</sub> = P<sub>281</sub>  
B<sub>280</sub> = P<sub>282</sub>  
B<sub>281</sub> = P<sub>283</sub>  
B<sub>282</sub> = P<sub>284</sub>  
B<sub>283</sub> = P<sub>285</sub>  
B<sub>284</sub> = P<sub>286</sub>  
B<sub>285</sub> = P<sub>287</sub>  
B<sub>286</sub> = P<sub>288</sub>  
B<sub>287</sub> = P<sub>289</sub>  
B<sub>288</sub> = P<sub>290</sub>  
B<sub>289</sub> = P<sub>291</sub>  
B<sub>290</sub> = P<sub>292</sub>  
B<sub>291</sub> = P<sub>293</sub>  
B<sub>292</sub> = P<sub>294</sub>  
B<sub>293</sub> = P<sub>295</sub>  
B<sub>294</sub> = P<sub>296</sub>  
B<sub>295</sub> = P<sub>297</sub>  
B<sub>296</sub> = P<sub>298</sub>  
B<sub>297</sub> = P<sub>299</sub>  
B<sub>298</sub> = P<sub>300</sub>  
B<sub>299</sub> = P<sub>301</sub>  
B<sub>300</sub> = P<sub>302</sub>  
B<sub>301</sub> = P<sub>303</sub>  
B<sub>302</sub> = P<sub>304</sub>  
B<sub>303</sub> = P<sub>305</sub>  
B<sub>304</sub> = P<sub>306</sub>  
B<sub>305</sub> = P<sub>307</sub>  
B<sub>306</sub> = P<sub>308</sub>  
B<sub>307</sub> = P<sub>309</sub>  
B<sub>308</sub> = P<sub>310</sub>  
B<sub>309</sub> = P<sub>311</sub>  
B<sub>310</sub> = P<sub>312</sub>  
B<sub>311</sub> = P<sub>313</sub>  
B<sub>312</sub> = P<sub>314</sub>  
B<sub>313</sub> = P<sub>315</sub>  
B<sub>314</sub> = P<sub>316</sub>  
B<sub>315</sub> = P<sub>317</sub>  
B<sub>316</sub> = P<sub>318</sub>  
B<sub>317</sub> = P<sub>319</sub>  
B<sub>318</sub> = P<sub>320</sub>  
B<sub>319</sub> = P<sub>321</sub>  
B<sub>320</sub> = P<sub>322</sub>  
B<sub>321</sub> = P<sub>323</sub>  
B<sub>322</sub> = P<sub>324</sub>  
B<sub>323</sub> = P<sub>325</sub>  
B<sub>324</sub> = P<sub>326</sub>  
B<sub>325</sub> = P<sub>327</sub>  
B<sub>326</sub> = P<sub>328</sub>  
B<sub>327</sub> = P<sub>329</sub>  
B<sub>328</sub> = P<sub>330</sub>  
B<sub>329</sub> = P<sub>331</sub>  
B<sub>330</sub> = P<sub>332</sub>  
B<sub>331</sub> = P<sub>333</sub>  
B<sub>332</sub> = P<sub>334</sub>  
B<sub>333</sub> = P<sub>335</sub>  
B<sub>334</sub> = P<sub>336</sub>  
B<sub>335</sub> = P<sub>337</sub>  
B<sub>336</sub> = P<sub>338</sub>  
B<sub>337</sub> = P<sub>339</sub>  
B<sub>338</sub> = P<sub>340</sub>  
B<sub>339</sub> = P<sub>341</sub>  
B<sub>340</sub> = P<sub>342</sub>  
B<sub>341</sub> = P<sub>343</sub>  
B<sub>342</sub> = P<sub>344</sub>  
B<sub>343</sub> = P<sub>345</sub>  
B<sub>344</sub> = P<sub>346</sub>  
B<sub>345</sub> = P<sub>347</sub>  
B<sub>346</sub> = P<sub>348</sub>  
B<sub>347</sub> = P<sub>349</sub>  
B<sub>348</sub> = P<sub>350</sub>  
B<sub>349</sub> = P<sub>351</sub>  
B<sub>350</sub> = P<sub>352</sub>  
B<sub>351</sub> = P<sub>353</sub>  
B<sub>352</sub> = P<sub>354</sub>  
B<sub>353</sub> = P<sub>355</sub>  
B<sub>354</sub> = P<sub>356</sub>  
B<sub>355</sub> = P<sub>357</sub>  
B<sub>356</sub> = P<sub>358</sub>  
B<sub>357</sub> = P<sub>359</sub>  
B<sub>358</sub> = P<sub>360</sub>  
B<sub>359</sub> = P<sub>361</sub>  
B<sub>360</sub> = P<sub>362</sub>  
B<sub>361</sub> = P<sub>363</sub>  
B<sub>362</sub> = P<sub>364</sub>  
B<sub>363</sub> = P<sub>365</sub>  
B<sub>364</sub> = P<sub>366</sub>  
B<sub>365</sub> = P<sub>367</sub>  
B<sub>366</sub> = P<sub>368</sub>  
B<sub>367</sub> = P<sub>369</sub>  
B<sub>368</sub> = P<sub>370</sub>  
B<sub>369</sub> = P<sub>371</sub>  
B<sub>370</sub> = P<sub>372</sub>  
B<sub>371</sub> = P<sub>373</sub>  
B<sub>372</sub> = P<sub>374</sub>  
B<sub>373</sub> = P<sub>375</sub>  
B<sub>374</sub> = P<sub>376</sub>  
B<sub>375</sub> = P<sub>377</sub>  
B<sub>376</sub> = P<sub>378</sub>  
B<sub>377</sub> = P<sub>379</sub>  
B<sub>378</sub> = P<sub>380</sub>  
B<sub>379</sub> = P<sub>381</sub>  
B<sub>380</sub> = P<sub>382</sub>  
B<sub>381</sub> = P<sub>383</sub>  
B<sub>382</sub> = P<sub>384</sub>  
B<sub>383</sub> = P<sub>385</sub>  
B<sub>384</sub> = P<sub>386</sub>  
B<sub>385</sub> = P<sub>387</sub>  
B<sub>386</sub> = P<sub>388</sub>  
B<sub>387</sub> = P<sub>389</sub>  
B<sub>388</sub> = P<sub>390</sub>  
B<sub>389</sub> = P<sub>391</sub>  
B<sub>390</sub> = P<sub>392</sub>  
B<sub>391</sub> = P<sub>393</sub>  
B<sub>392</sub> = P<sub>394</sub>  
B<sub>393</sub> = P<sub>395</sub>  
B<sub>394</sub> = P<sub>396</sub>  
B<sub>395</sub> = P<sub>397</sub>  
B<sub>396</sub> = P<sub>398</sub>  
B<sub>397</sub> = P<sub>399</sub>  
B<sub>398</sub> = P<sub>400</sub>  
B<sub>399</sub> = P<sub>401</sub>  
B<sub>400</sub> = P<sub>402</sub>  
B<sub>401</sub> = P<sub>403</sub>  
B<sub>402</sub> = P<sub>404</sub>  
B<sub>403</sub> = P<sub>405</sub>  
B<sub>404</sub> = P<sub>406</sub>  
B<sub>405</sub> = P<sub>407</sub>  
B<sub>406</sub> = P<sub>408</sub>  
B<sub>407</sub> = P<sub>409</sub>  
B<sub>408</sub> = P<sub>410</sub>  
B<sub>409</sub> = P<sub>411</sub>  
B<sub>410</sub> = P<sub>412</sub>  
B<sub>411</sub> = P<sub>413</sub>  
B<sub>412</sub> = P<sub>414</sub>  
B<sub>413</sub> = P<sub>415</sub>  
B<sub>414</sub> = P<sub>416</sub>  
B<sub>415</sub> = P<sub>417</sub>  
B<sub>416</sub> = P<sub>418</sub>  
B<sub>417</sub> = P<sub>419</sub>  
B<sub>418</sub> = P<sub>420</sub>  
B<sub>419</sub> = P<sub>421</sub>  
B<sub>420</sub> = P<sub>422</sub>  
B<sub>421</sub> = P<sub>423</sub>  
B<sub>422</sub> = P<sub>424</sub>  
B<sub>423</sub> = P<sub>425</sub>  
B<sub>424</sub> = P<sub>426</sub>  
B<sub>425</sub> = P<sub>427</sub>  
B<sub>426</sub> = P<sub>428</sub>  
B<sub>427</sub> = P<sub>429</sub>  
B<sub>428</sub> = P<sub>430</sub>  
B<sub>429</sub> = P<sub>431</sub>  
B<sub>430</sub> = P<sub>432</sub>  
B<sub>431</sub> = P<sub>433</sub>  
B<sub>432</sub> = P<sub>434</sub>  
B<sub>433</sub> = P<sub>435</sub>  
B<sub>434</sub> = P<sub>436</sub>  
B<sub>435</sub> = P<sub>437</sub>  
B<sub>436</sub> = P<sub>438</sub>  
B<sub>437</sub> = P<sub>439</sub>  
B<sub>438</sub> = P<sub>440</sub>  
B<sub>439</sub> = P<sub>441</sub>  
B<sub>440</sub> = P<sub>442</sub>  
B<sub>441</sub> = P<sub>443</sub>  
B<sub>442</sub> = P<sub>444</sub>  
B<sub>443</sub> = P<sub>445</sub>  
B<sub>444</sub> = P<sub>446</sub>  
B<sub>445</sub> = P<sub>447</sub>  
B<sub>446</sub> = P<sub>448</sub>  
B<sub>447</sub> = P<sub>449</sub>  
B<sub>448</sub> = P<sub>450</sub>  
B<sub>449</sub> = P<sub>451</sub>  
B<sub>450</sub> = P<sub>452</sub>  
B<sub>451</sub> = P<sub>453</sub>  
B<sub>452</sub> = P<sub>454</sub>  
B<sub>453</sub> = P<sub>455</sub>  
B<sub>454</sub> = P<sub>456</sub>  
B<sub>455</sub> = P<sub>457</sub>  
B<sub>456</sub> = P<sub>458</sub>  
B<sub>457</sub> = P<sub>459</sub>  
B<sub>458</sub> = P<sub>460</sub>  
B<sub>459</sub> = P<sub>461</sub>  
B<sub>460</sub> = P<sub>462</sub>  
B<sub>461</sub> = P<sub>463</sub>  
B<sub>462</sub> = P<sub>464</sub>  
B<sub>463</sub> = P<sub>465</sub>  
B<sub>464</sub> = P<sub>466</sub>  
B<sub>465</sub> = P<sub>467</sub>  
B<sub>466</sub> = P<sub>468</sub>  
B<sub>467</sub> = P<sub>469</sub>  
B<sub>468</sub> = P<sub>470</sub>  
B<sub>469</sub> = P<sub>471</sub>  
B<sub>470</sub> = P<sub>472</sub>  
B<sub>471</sub> = P<sub>473</sub>  
B<sub>472</sub> = P<sub>474</sub>  
B<sub>473</sub> = P<sub>475</sub>  
B<sub>474</sub> = P<sub>476</sub>  
B<sub>475</sub> = P<sub>477</sub>  
B<sub>476</sub> = P<sub>478</sub>  
B<sub>477</sub> = P<sub>479</sub>  
B<sub>478</sub> = P<sub>480</sub>  
B<sub>479</sub> = P<sub>481</sub>  
B<sub>480</sub> = P<sub>482</sub>  
B<sub>481</sub> = P<sub>483</sub>  
B<sub>482</sub> = P<sub>484</sub>  
B<sub>483</sub> = P<sub>48</sub>

MMU manages segment based virtual memory

The memory management unit manages a virtual memory in a segment based system.

In segment based virtual memory system, the memory is divided into segments, each of which can be assigned a unique base address and limit. The base address is the starting location of the segment in the physical memory.

When a program tries to access memory, the MMU translates the virtual address into a physical address.

The MMU does this using a page table. The page table contains entries for each segment.

The MMU looks up the corresponding segment in the page table and then translates the virtual address into a physical address by adding the segment's base address.

Overall, the MMU plays a crucial role in managing virtual memory and ensuring that programs can access the memory when they need.

-39.65

$$= -100111.10100110011001$$

sign bit = 1

$$= 1.00111101001100110001 \dots E3$$

Exponent field =  $8 + 127$   
= 132

1 1000010000111010010000

$$11000010000111010010000$$

RC 2 1 E 98

(C21E98)<sub>16</sub>

$$\begin{array}{r} .65 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .3 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .6 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .2 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .4 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .8 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .6 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .2 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .4 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .8 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .6 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .2 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .4 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .8 \\ \times 2 \\ \hline \end{array}$$

$$\begin{array}{r} .6 \\ \times 2 \\ \hline \end{array}$$

REP

means repeat string instruction until specified conditions exist.

loop

means jump to a specified label if cx ≠ 0 after auto decrement

stc

set the carry flag  
'1'

std

set the direction flag to '1'

JMP

JB

an unconditional jump to a specified destination

jump if below.

ROL

Rotate operand around to the left through

RCL

Rotate all bits of operand left.