Serial Communication	Parallel communication
i) Data is transmitted bit after the bit in a single line	i) through group of lines
ii) Data congestion take place.	ii) No.
iii) Low speed transmission	iii) High
iv) Implementation is not an easy task.	iv) easily implemented in hardware.
v) less expensive	v) more expensive
vi) No crosstalk problem	vi) having.
vii) bandwidth of serial wires is much higher	vii) is much lower.
viii) work effectively at high frequencies.	viii) hard to run at high frequencies.

** Why port 0 is connected to a 10 kΩ 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 10kΩ 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.

DULE 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 8051 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 $2^{16} = 64$ 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 ↴

P0 ⟶ data bus

= 0 ⟶ P0 ⟶ address bus

Alternate functions of Port 3

Pins 10-17 is known as port 3. This port serves some function like intercrupts, timer input, control signals, serial communication signals and RxD and TxD etc.

- RD — Read data control output } P3.6 and P3.7
- WR — Write " " " }
- T1 — Timer / counter1 external input } P3.4 and P3.5
- T0 — Timer / counter0 " "
- IIN1 → Intercrupt1
- INT0 → Intercrupt0
- TXD → Transmit data } P3.0 and P3.1
- RXD → Receive data

~~2019~~

2020 – 3(c)

SM0	SM1	SM2	REN	TB8	RB8	TI	RI

This serial control Registeres (SCON).

SM0, SM1 → Mode selection

SM0	SM1	
0	0	→ mode 0
0	1	→ mode 1
1	0	→ mode 2
1	1	→ mode 3

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

REN → Enable serial Reception

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)

TMOD Register of 8051 microcontroller ?

D7	D6	D5	D4	D3	D2	D1	D0
GATE	C/T	M1	M0	GATE	C/T	M1	M0

Here,

$D_0 - D_3$ → Timer 0

$D_4 - D_7$ → Timer 1

M0, M1 → Mode selection

M0	M1	
0	0	→ mode 0 → 13 bit
0	1	→ mode 1 → 16 bit
1	0	→ mode 2 → 8-bit auto reload
1	1	→ mode 3 → split

C/T → used to decide whether a timer or counter. If C/T = 1 then decide to use counter and C/T = 0 then decide to use timer.

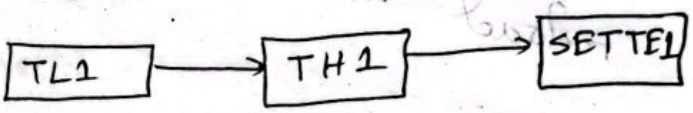
gate → means of starting and stopping. If Gate = 1 then it starts and gate = 0 then it ~~starts~~ stops.

2019 - 7(a)

Timer Mode Operation of 8051 Microcontroller :

Timer Mode 0 :

The Mode 0 operation is the 8-bit timer 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 or TH1.



Here the TLX counts up to 31 and then reset to 00 and increement THX by 1. suppose we load 0 in the timer 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 cycle. Timers can be started by →

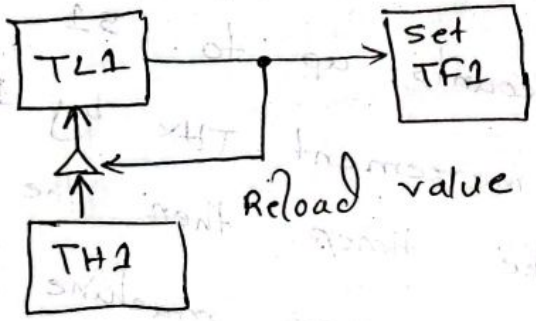
SETB TR0 → Timer 0

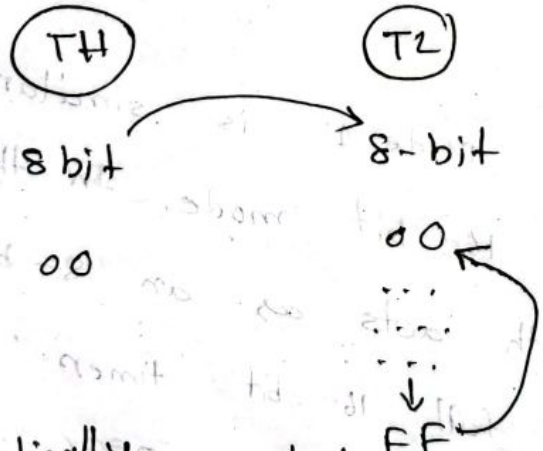
SETB TR1 → Timer 1

When flag bit is set, then going to stop the timers → CLR TR0

CLR TR1

Mode 2 : In this Mode TLx acts as the timer and THx contains the Reload value. It is 8-bit auto reload mode.





It automatically reloads the initial value from register TH into the TL register when the counter overflows.

Mode 3 :→

Mode 3 is different for Timer 0 and Timer 1. When the timer 0 is working in mode 3 the T2O will be used as an 8-bit timer/counter. And THO 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.

2019 - 8(c)

Register Banks of 8051 micro controller :—

The 8051 micro controller has four Register 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 PS.W.4 it also general purpose (R0 - R7).

Register Bank 2 : setting the bits PSW.2 and PSW

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 are :-

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
CY	AC	F0	RS1	RS0	OV	-	P

CY = carry flag

AC = Auxilliary carry flag

F0 = user flag

RS1, RS0 = selected Register bank

OV = overflow flag

(-) = reserved

P = parity flag.

* pins of 8051 are :-

PSEN : pin(29) : program store enable

This pin is used to enable external program memory.

This is an output pin.

/EA (pin 31) :-

i) external access

ii) /EA is connected to GND that means it connected to externally.

ALE (pin 30) :-

i) Address Latch Enable

ii) it is an output pin and is active high.

iii) 8051 port 0 provides both data and address.

I/O pins :-

(i) four ports P0, P1, P2, P3

(ii) Each port use 8 pins.

(iii) bi-directional

RST (pin 9) :-

(i) reset

(ii) input pin and is active high

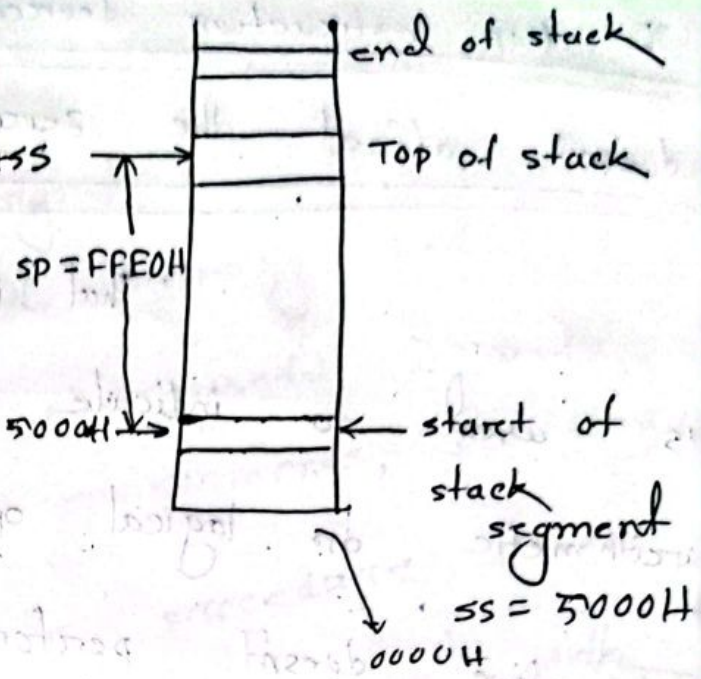
(iii) applying RST, we will reset and all values will be lost

Serial Port interrupt in 8051 uc :-

In an 8051 uc, a serial port interrupt occurs when the uc receives data via its serial port. Having two serial ports named SP0 and SP1. When a serial port interrupts occurs, the uc's CPU stop 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 main program.

2020 - 6(c)

* Given,



so, physical address is,

= ss + sp

$$= \begin{array}{r} 50000 \\ +\ \text{OFFE0} \\ \hline \text{5FFE0} \end{array}$$

physical address $\longrightarrow$ 5FFE0

(ii) lower range = 5000 x 10 H

= 50000 H

upper range = (50000 + offff) H

= 5FFFF H

(iii) Maximum size of the stack

= 5FFFF - 50000

= FFFF

$= (2^{16})_{10}$ = 64kB

6(b) - 2019

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.

2019 - 6(a)

Significance of reentrant procedure :→

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.

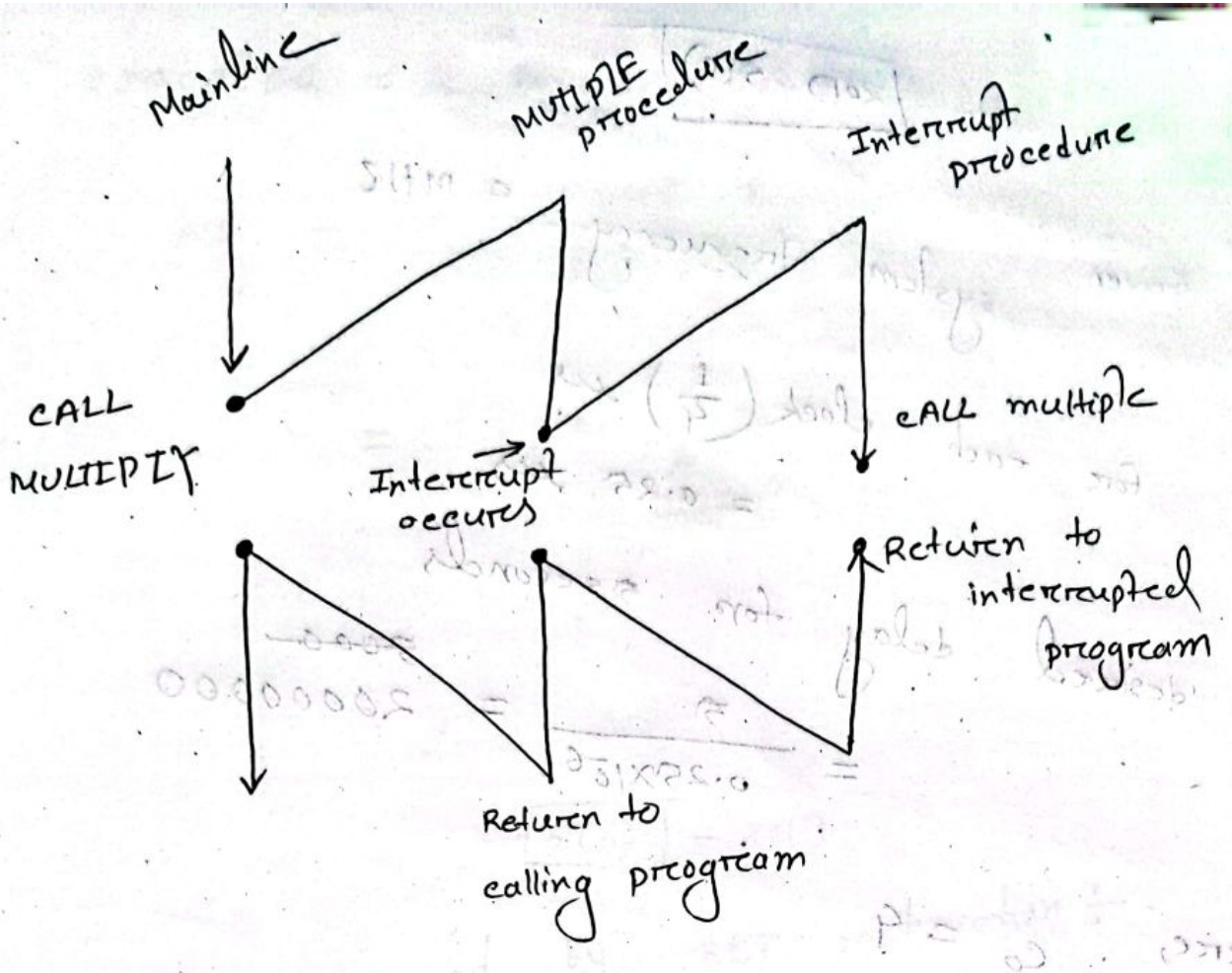
For example, consider a program that uses a non-reentrant procedure to calculate the factorial of a number. This procedure may use global variables and may also modify memory locations. If the procedure is called multiple times simultaneously, it causes conflicts and errors because global variable and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

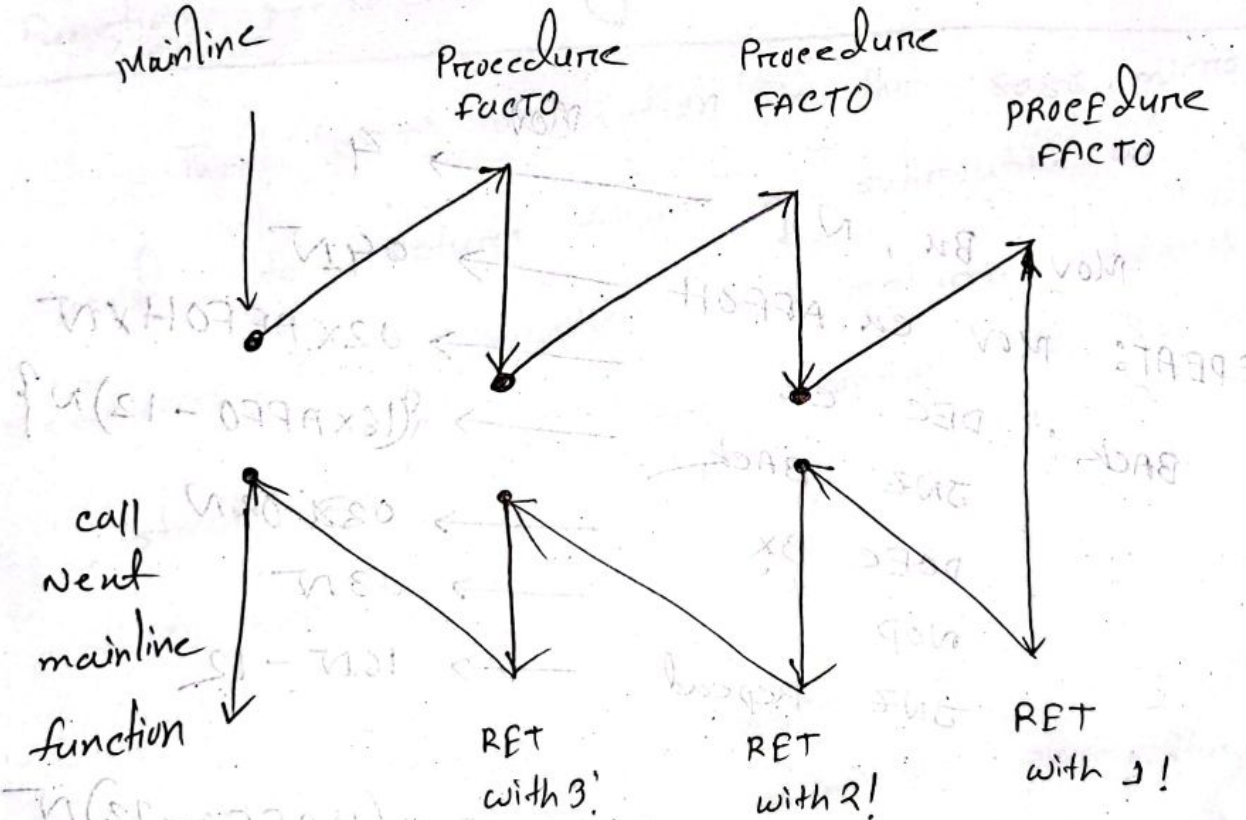
may be modified. On the other hand, a reentrant procedure in 8086 microprocessor uses only local variables and doesn't modify memory locations. This allow multiple calls.

Significance of recursive procedure :→

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.


Mainline
MULTIPLE procedure
Interrupt procedure
CALL MULTIPLY
Interrupt occurs
CALL multiple
Return to interrupted program
Return to calling program


Mainline
Procedure FACTO
Procedure FACTO
PROCEDURE FACTO
call next mainline function
RET with 3!
RET with 2!
RET with 1!

2019 - 5(c)

* Given, system frequency 4 MHz

for each clock $\left(\frac{1}{4}\right)$ μs

$= 0.25$ μs

the desired delay for 5 seconds

$= \frac{5}{0.25 \times 10^{-6}} = 20000000$

Here, $C_0 = 4$

```
MOV BX, N          ------> 4
REPEAT: MOV CX, AFF0H  ------> 04N
BACK:   DEC CX         ------> 02 x AFF0H x N
        JNZ BACK       ------> {(16 x AFF0 - 12)N}
        DEC BX         ------> 02N
        NOP            ------> 03N
        JNZ Repeat     ------> 16N - 12
```

$C_T = C_0 + 04N + 2 \times AFF0H \times N + (16AFF0 - 12)N + 2N + 3N + 16N - 12$

$= C_0 + N\{4 + 2 \times AFF0 + 16AFF0 - 12 + 2 + 3 + 16\} - 12$

$= C_0 + N\{13 + 2AFF0 + 16AFF0\}$

$$20000000 = 4 + N\{(13 + (18) \times 450 40\}$$

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

$$= 24.67$$ Ans

5(b) – 2019

* Function performed by RET instruction :-

The 'RET' instruction in the 8086 microprocess. is used to return from a subroutine or a the procedure. The instruction pops return address from the stack and transfers control to the instructio 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 addresses from the stack.

'RET n' this means, the instruction pops the return addresses 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) JG
vii) JGE
viii) JL
ix) JLE
x) JNE

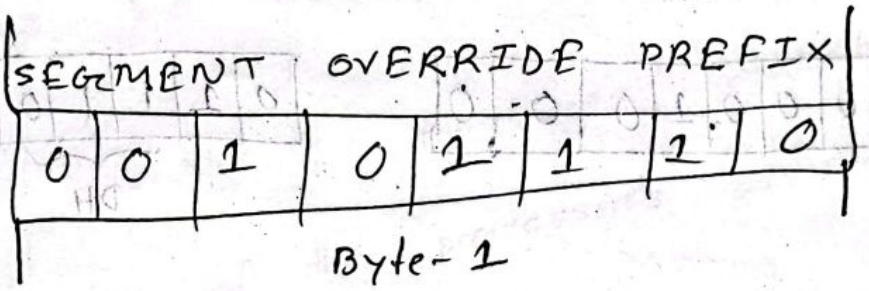
4 (b)

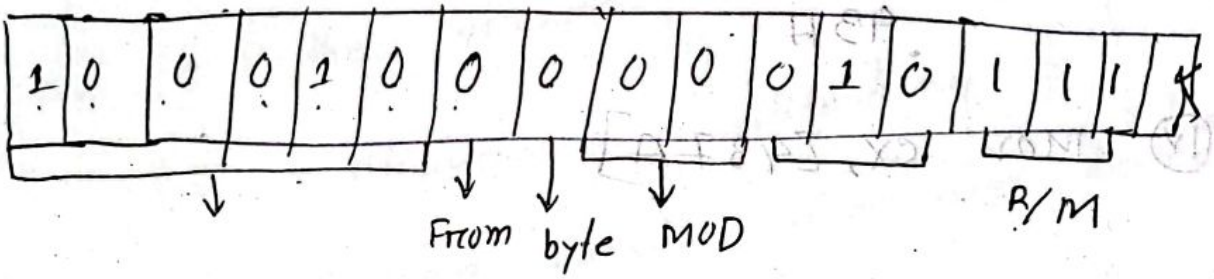
* Binary code :

(i) MOV cs : [BX], DL

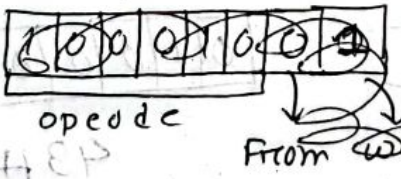
opcode for MOV = 100010

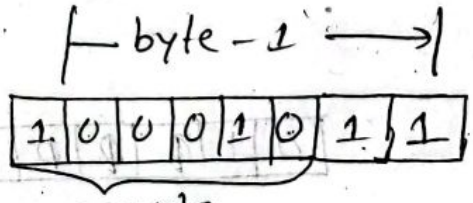
for cs = 01

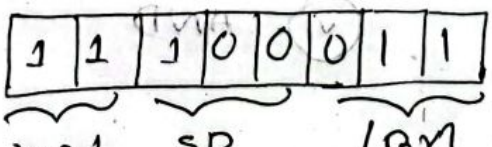




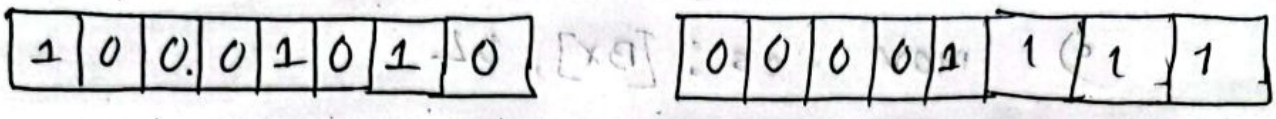
(ii) MOV SP, BX



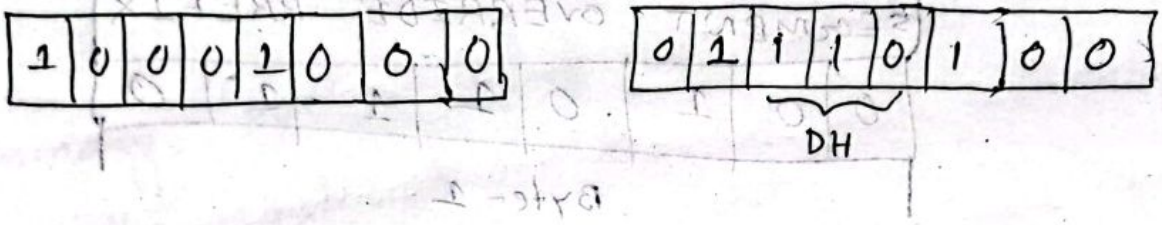


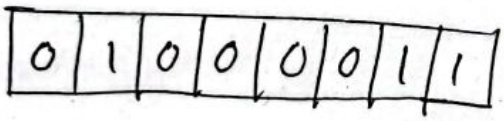


(ii) MOV CL, [BX]



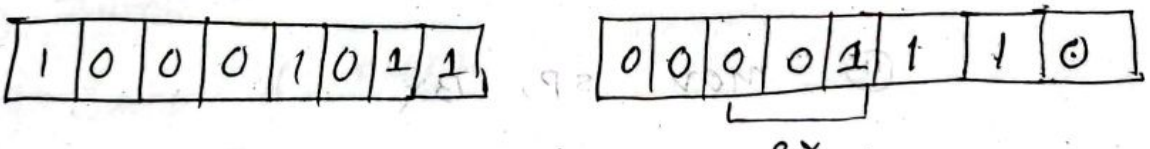
(iii) MOV 43H[SI], DH

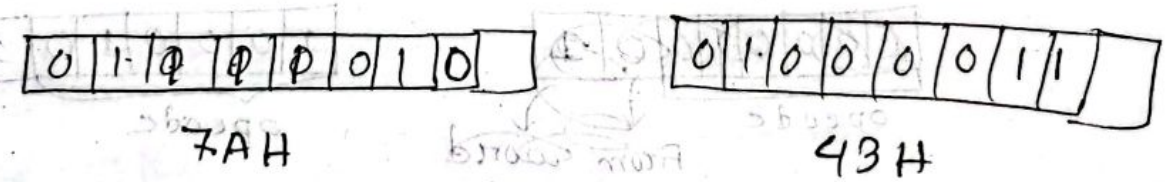




43H

(iv) MOV CX, [437A]





(v) IN AL, F0H

(vi) AND

OP CODE = 111001

2019 - 1(a)

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) Register based electronic device.

1(d) → 2019

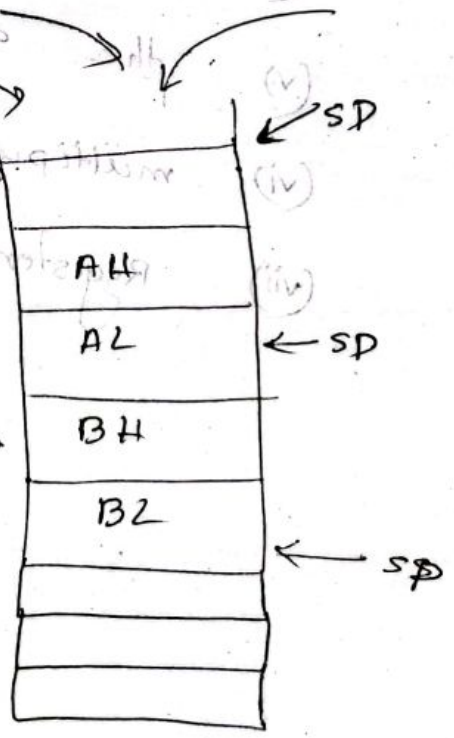
Parameter passing to or from a procedure using stack:-

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 correct order.

```
PUSH  AX
PUSH  BX

MOV   VALU1, AX
MOV   VALU2; BX

 POP   BX
 POP   AX
```



2019 - 1(b)

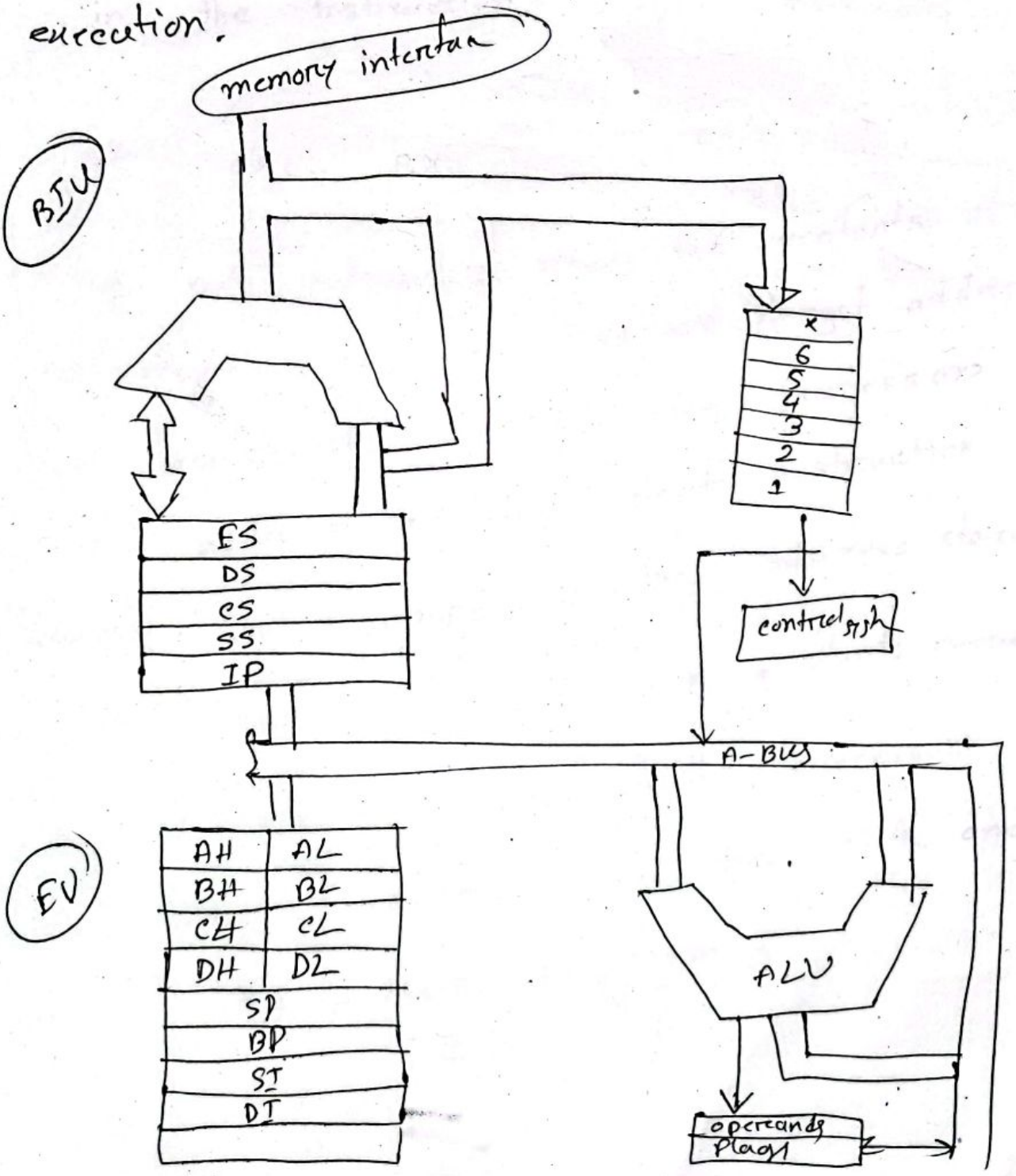
In the 8086 microprocessor, the Bus interface Unit (BIU) and the execution unit (EU) interact with each other to execute instructions and access memory. The BIU is responsible for fetching instructions from memory and providing them to the EU. It also handles memory and I/O operations by communicating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

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.

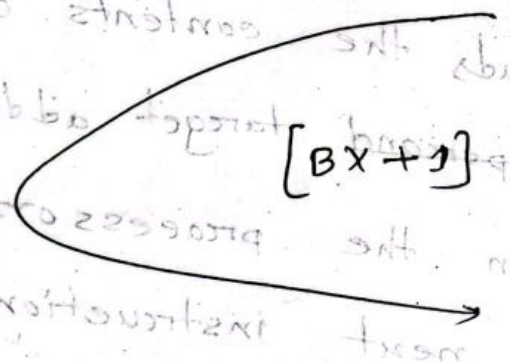
```
CALL BX
```

Firstly the call instruction reads the contents of the BX register as the ~~operand~~ 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 :→

An indirect near call 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 address directly in the instruction.

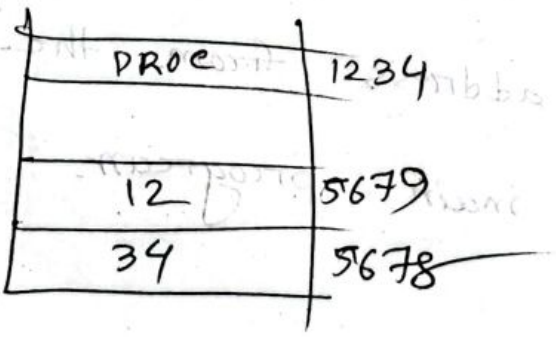
CALL word ptr [BX]



CALL [5678][5679]

↓

CALL 1234



limitations of 80286

(i) it has only a 16- bit ALU

(ii) maximum segment size is 64kbytes

(iii) cann't easily be switched back and forth between real and protected mode.

LMSW:

stands for Load Machine status word.

It is a machine instruction for the 80286 microprocessor that allows the contents of the MSW 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 up when switching between different tasks.

2019 - 8(a)

For 80286 switching real mode to protected mode :

First step : is to set the protection enable bit in the MSW register in the 80826.

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 byte queue, because in protected mode the queue function differently from the way it does in real mode.

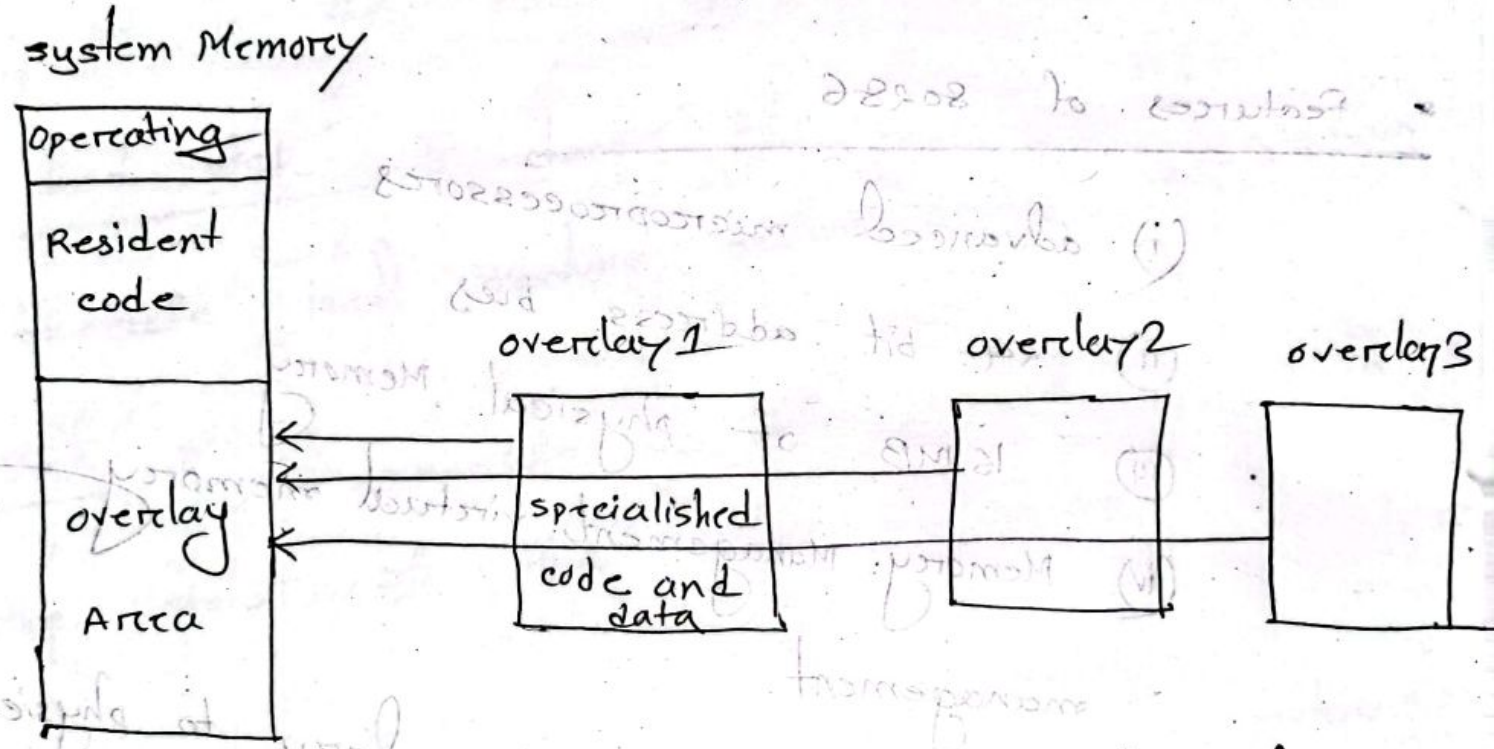
(i) prepare the descriptor tables
(ii) set the PE
(iii) paging enable
(iv) Load MSW
(v) Jump

* 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)

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

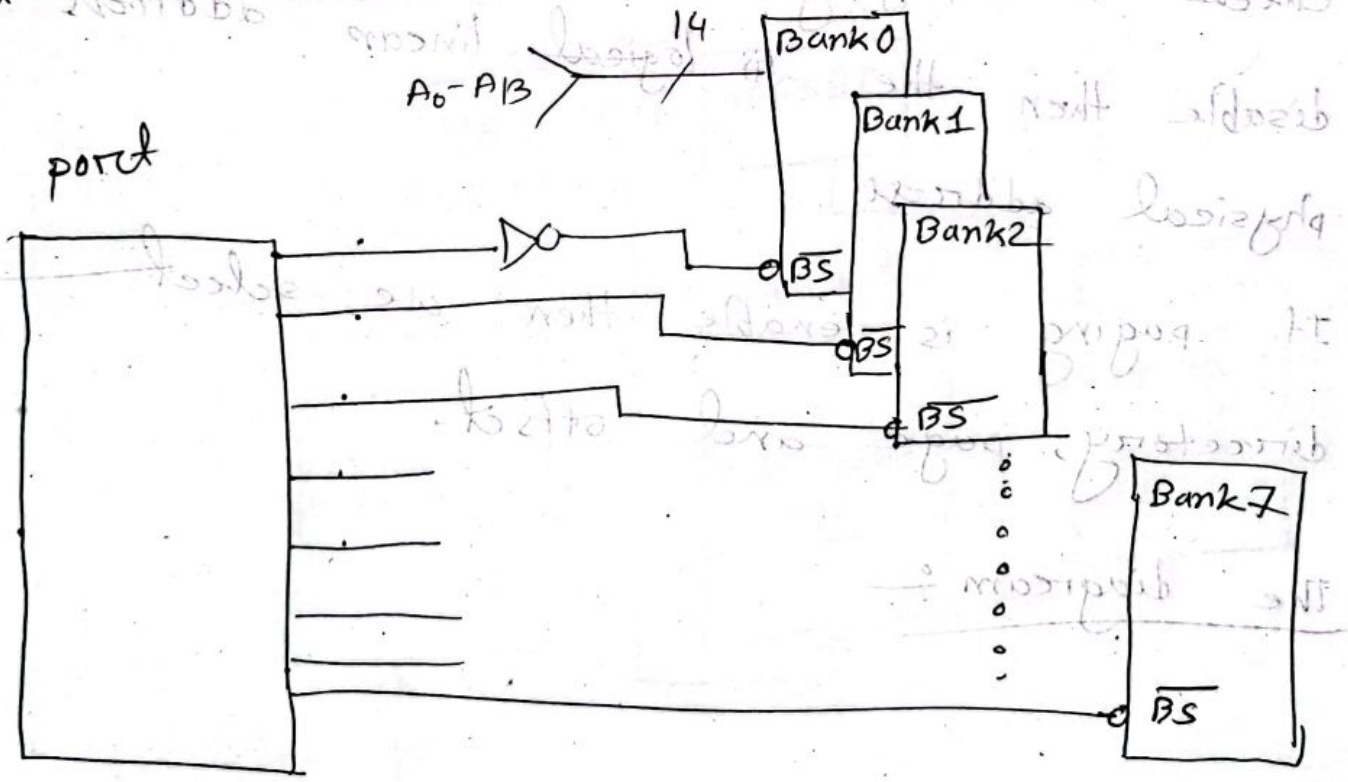


When the assembler reaches a point where it needs the next module then it references as an overlay then overlay area reserved in the memory.

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

Bank switching :-

8 bit 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 8 bank and every bank 8 16 bytes. So they hold 128 bytes data together. To switch to bank 1, a byte which turns off bank 0 and turns on bank 1 is output to the selection port.



2019 - 7(c)

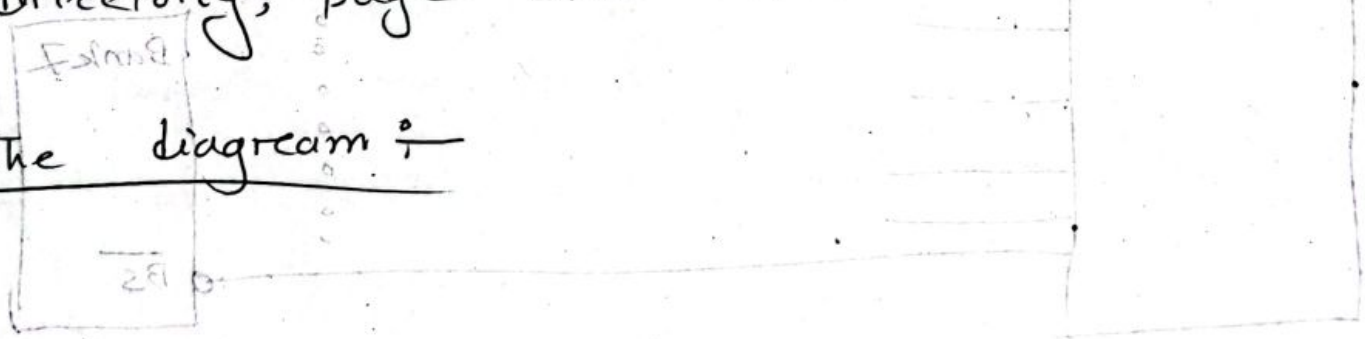
Paging Memory Managemend mode of 80386

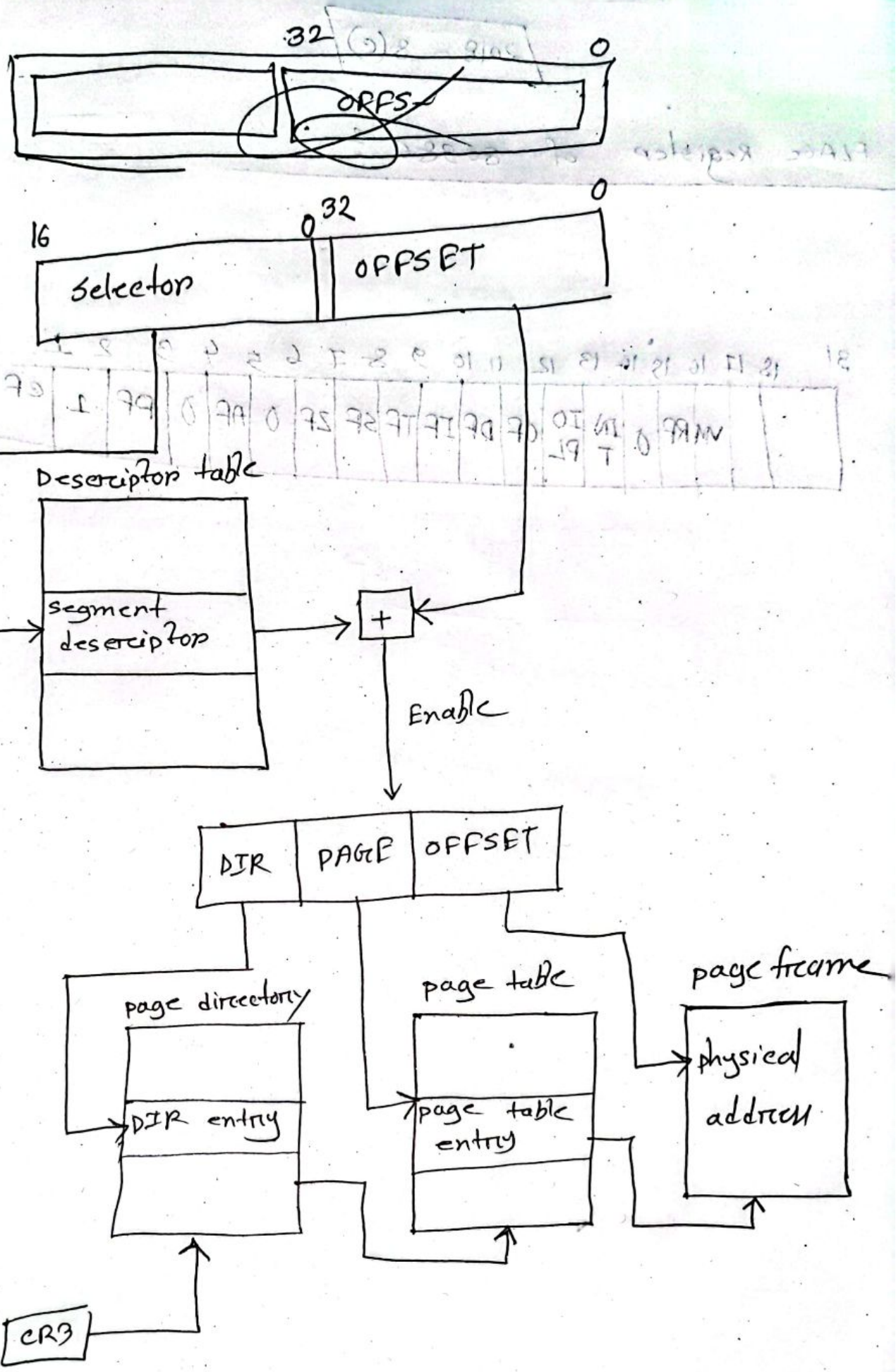
Here, we have 16 bit selector and 14 bit used for choose from descriptore table and othere 2 bit are privilged bit. And the OFFSET are 32 bits.

Then find the page from the descriptor table then add to the offset of this Mechanism check 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.

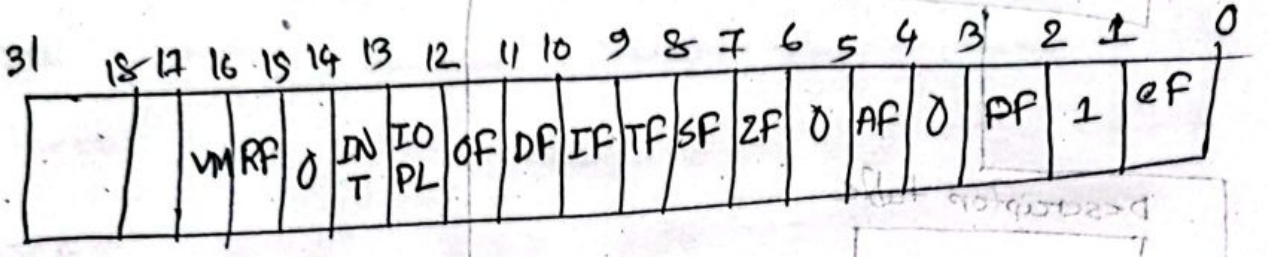
The diagream :-




32
0
OFFS
16
0
32
0
Selector
OFFSET
Descriptor table
segment descriptor
+
Enable
DIR
PAGE
OFFSET
page directory
DIR entry
page table
page table entry
page frame
physical address
CR3

2018 – 8(c)

FLAG Register of 80386 :–



Features of 80386:

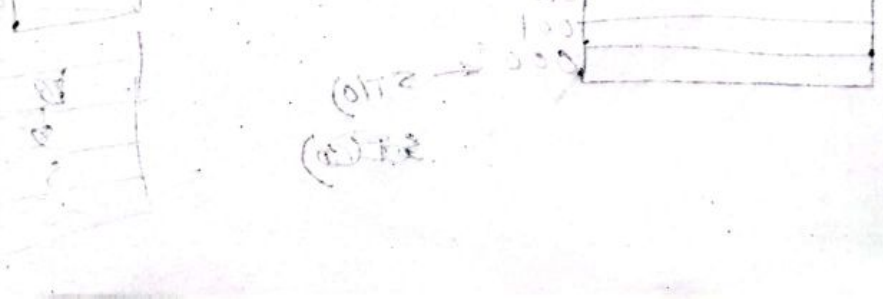
(i) 32-bit ALU

(ii) operate directly 32-bit data words

(iii) 4 gbytes

(iv) 1634 segments

(v) virtual 8086 mode



2017 - 5(b)

11000001110010010000000000000000

$1.10000011,1001001 \times 10^{16}$

sign bit = 1

exponent bit = 16

Exponent field = 131 - 127

= 4

-1.001001×2^{4}

= 10010.00

= ~~18.01 Ans~~ = − 18.25 Ans

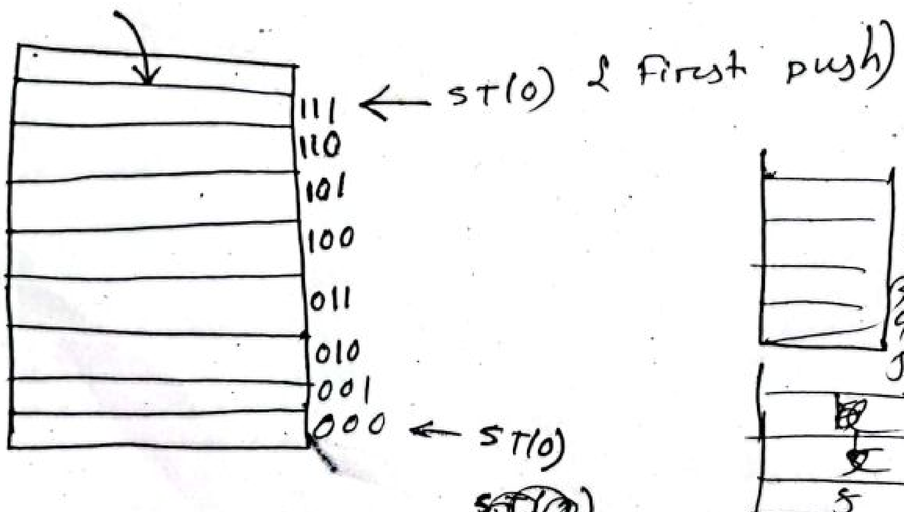
$0 \times 2^{-1} + 1 \times 2^{-2}$

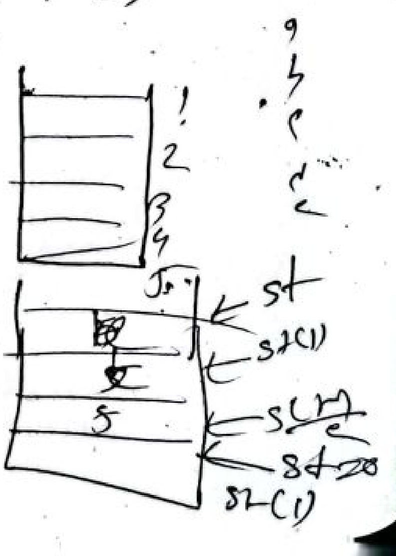
= 1/4

.25 × 2 = 0.50

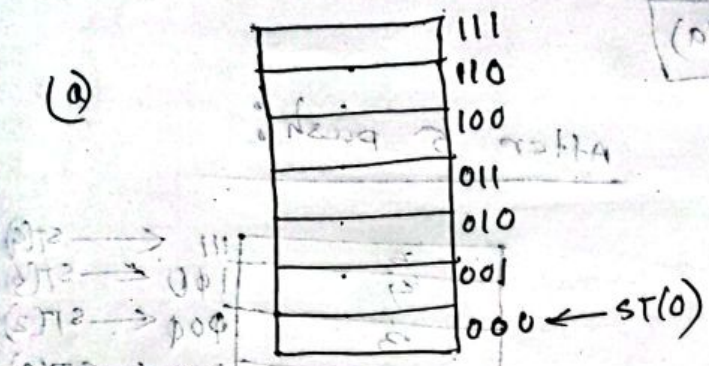
2017 - 6(a)

stack operations for 8087 :



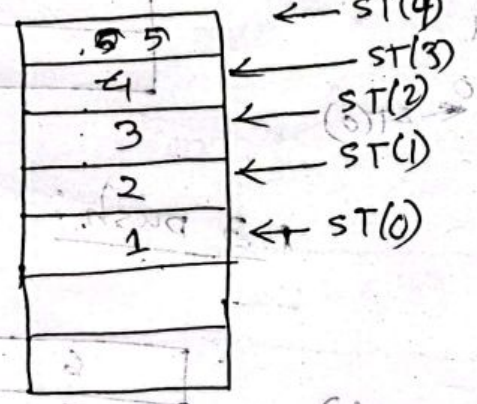


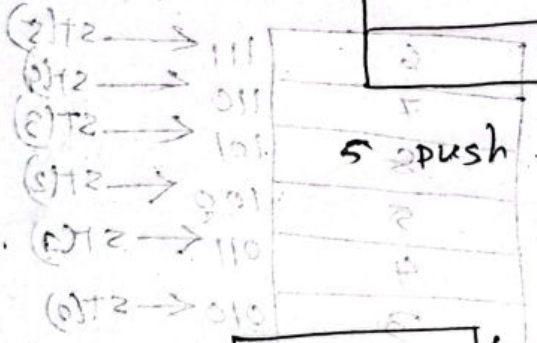
(a)

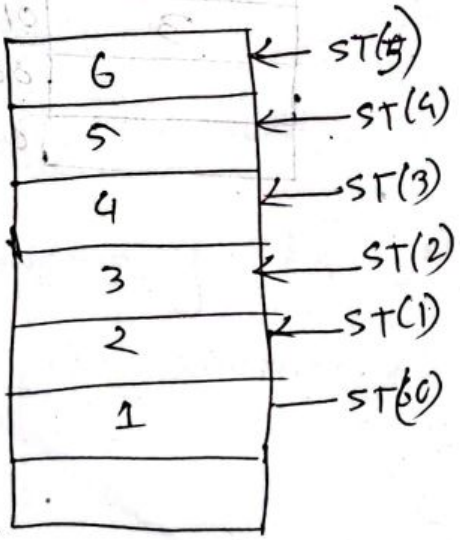


After reset

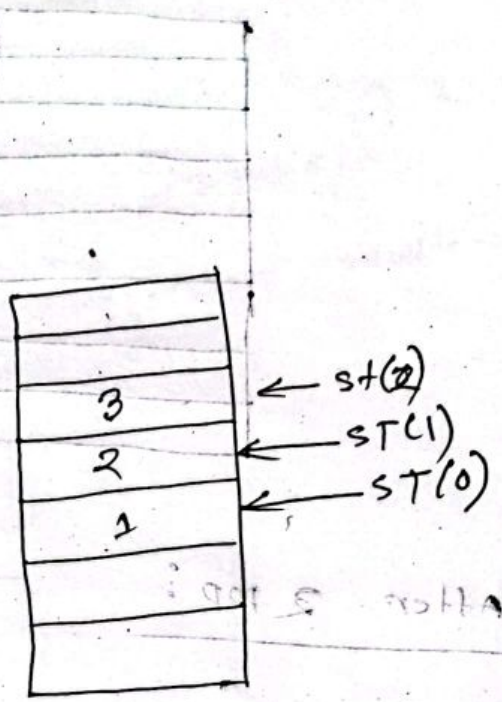
(b)

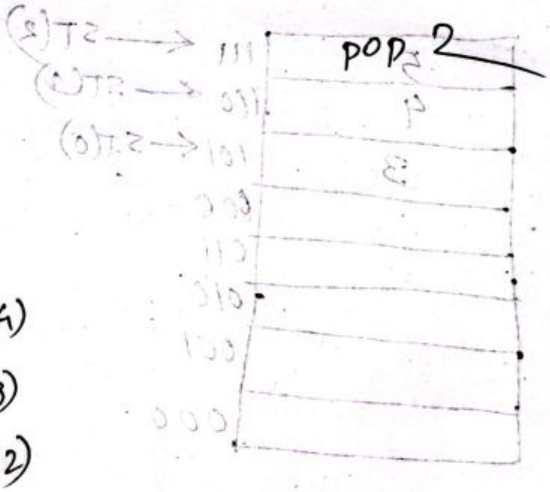






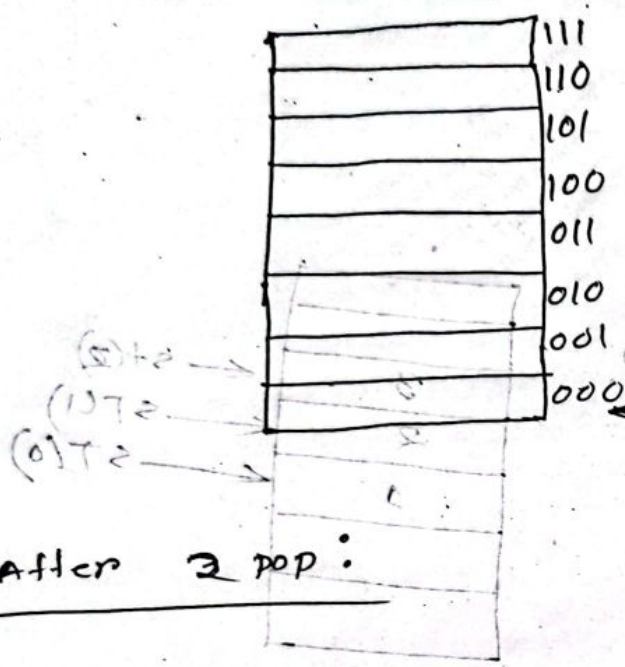
3 PUSH



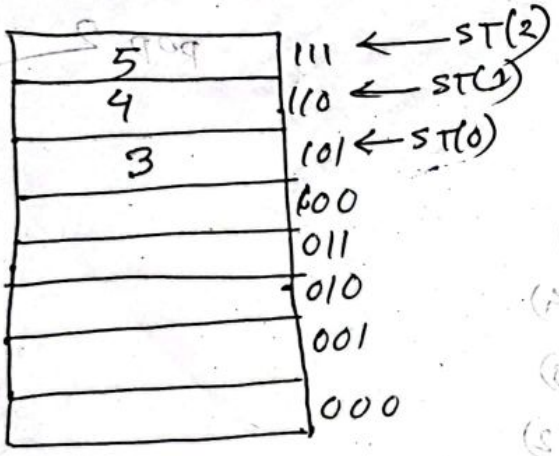


2017 - 6(a)

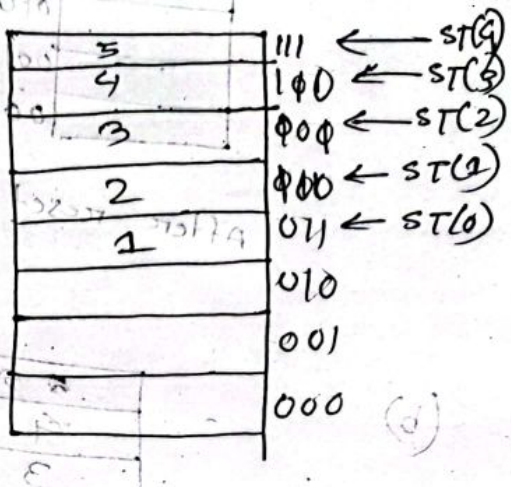
(a) After reset:



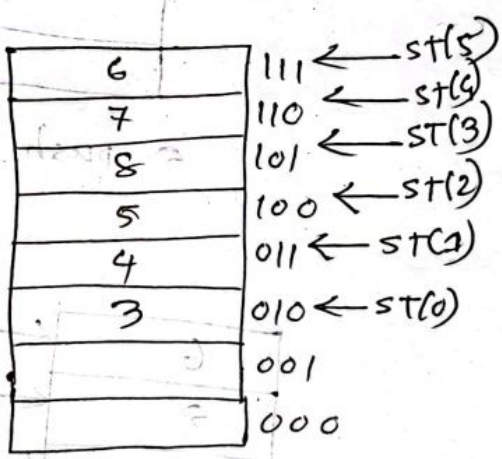
After 2 pop:



After 5 push:



3 push:



2017-6(c)

80386 overcome this:

ii) 80286 having 16 bit data bus but 80386 have 32-bit data bus.

ii) supports virtual memory, protected mode, allow to run multiple programs, multitasking improved instruction set.

2017-4(c)

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

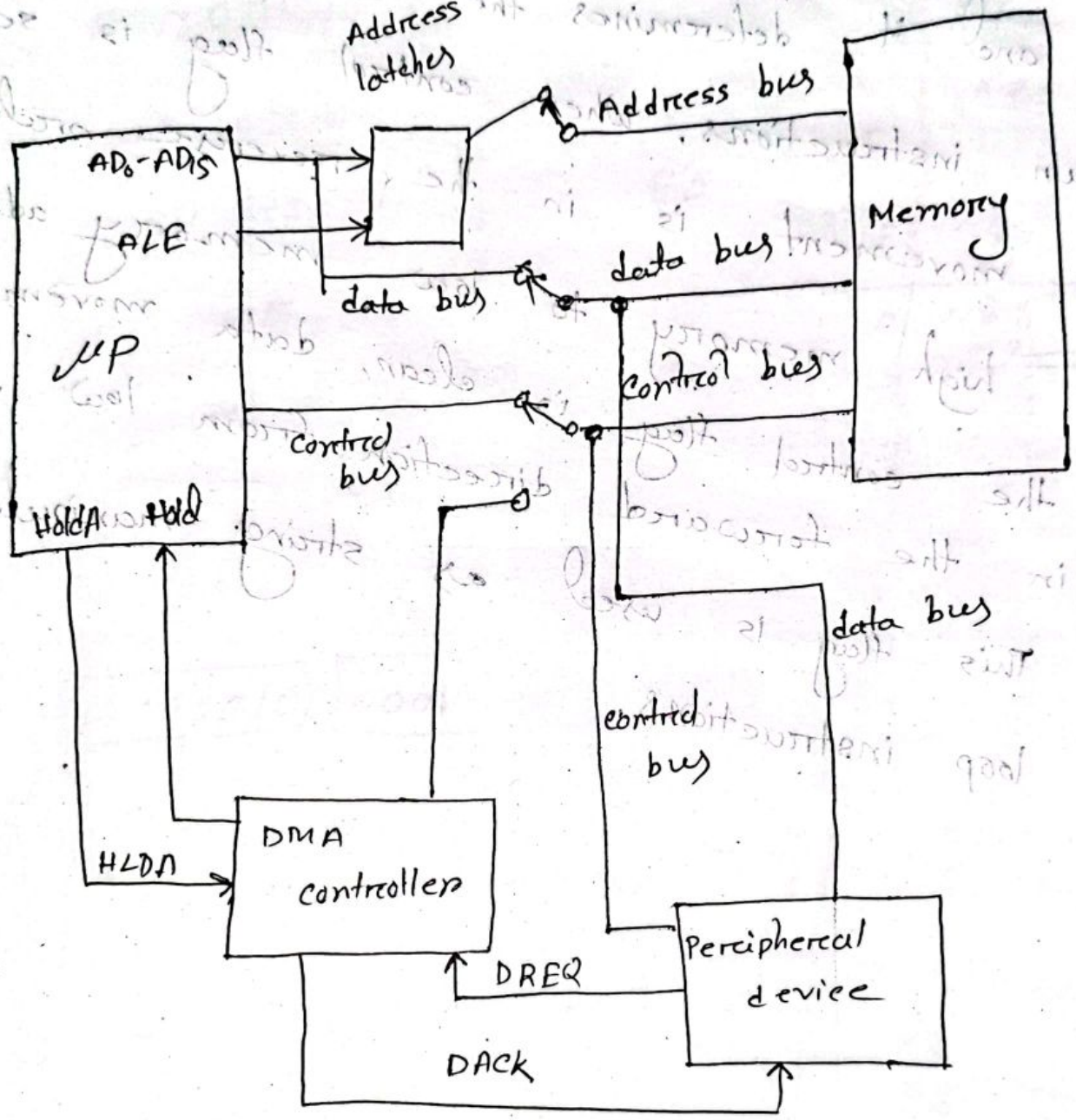
At first, we if want to send data to main memory, firstly the CPU would read the first byte of the program and that write that byte to memory. same fore the second byte. This process proves to be inefficient slows down transfere and modify the data.

If we pass the data directly 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 buses at other end.

firstly I/O device send a singnal the DMA-controller that is send request (DRequest) then DMA controller is send the Hold signal to the CPU. if CPU will not process any work then it sends back the signal HLDA then DMA sends DACK signal to the I/O device. Then this I/O

device directly pass the data to the main memory.



Control flag in 8086 :

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

2018 - 2 (e)

8.67

= ~~1000.0111010101~~

= 1000.10101011100001

= 1.0001010101111000 01 .... E3

sign bit = 1

the exponent form = 130

1

(1) 10000010 00010101011100001...

.67
×2
1 .34
×2
0 .68
×2
1 .36
×2
0 .72
×2
1 .44
×2
0 .88
×2
1 .76
×2
1 .52
×2
1 .04
×2
0 .08
×2
0 .16
×2
0 .32
×2
0 .64
×2
1 .28

Minimum Mode	Maximum Mode
i) single processor	ii) multiprocessor
ii) 8086 is responsible for generating all control signals for memory and I/O ports	ii) External Bus controller
iii) Used for simple applications	iii) used for complex
iv) Pin #30 is high	iv) #30 is low
v) can't add extra co-processor	v) can add extra co-processor

* 8086 and 8087 execute their respective instructions

The 8087 co-processor is designed to perform with the 8086 μp to perform floating point arithmetic operations.

The 8086 execute instructions using their complex instruction set, (Data transfer, Arithmetic, logic, string manipulation) while the 8087 co-processor is a specialized device that performs floating point operations on behalf of the main microprocessor.

MMU manage 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 a 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 needs.

5(c)

-39.65

$= -100111.10100110011001$

$=$

sign bit $= 1$

$= 1.00111.10100110011000 \times E3$

Exponent field $= 5 + 127$

$= 132$

1	100 0010	0 0011	10100110001000 ...

1100 0010 0001 1101 0011 0001 0000

C21E98

$(C21E98)_{16}$

	.65
	x2
1	.3
	x2
0	.6
	x2
1	.2
	x2
0	.4
	x2
0	.8
	x2
1	.6
	x2
1	.2
	x2
0	.4
	x2
0	.8
	x2
1	.6
	x2
1	.2
	x2
0	.4
	x2
0	.8
	x2
1	.6

REP	loop
means repeat string instruction untill specified conditions exist.	means jump to a specified label if $cx \neq 0$ after auto decrement

stc	stD
set the carry flag '1'	set the direction flag to '1'

JMP	JB
an unconditional jump to a specified destination	jump if below.

ROL	RCL
Rotate operand around to the left through	Rotate all bits of operand left.