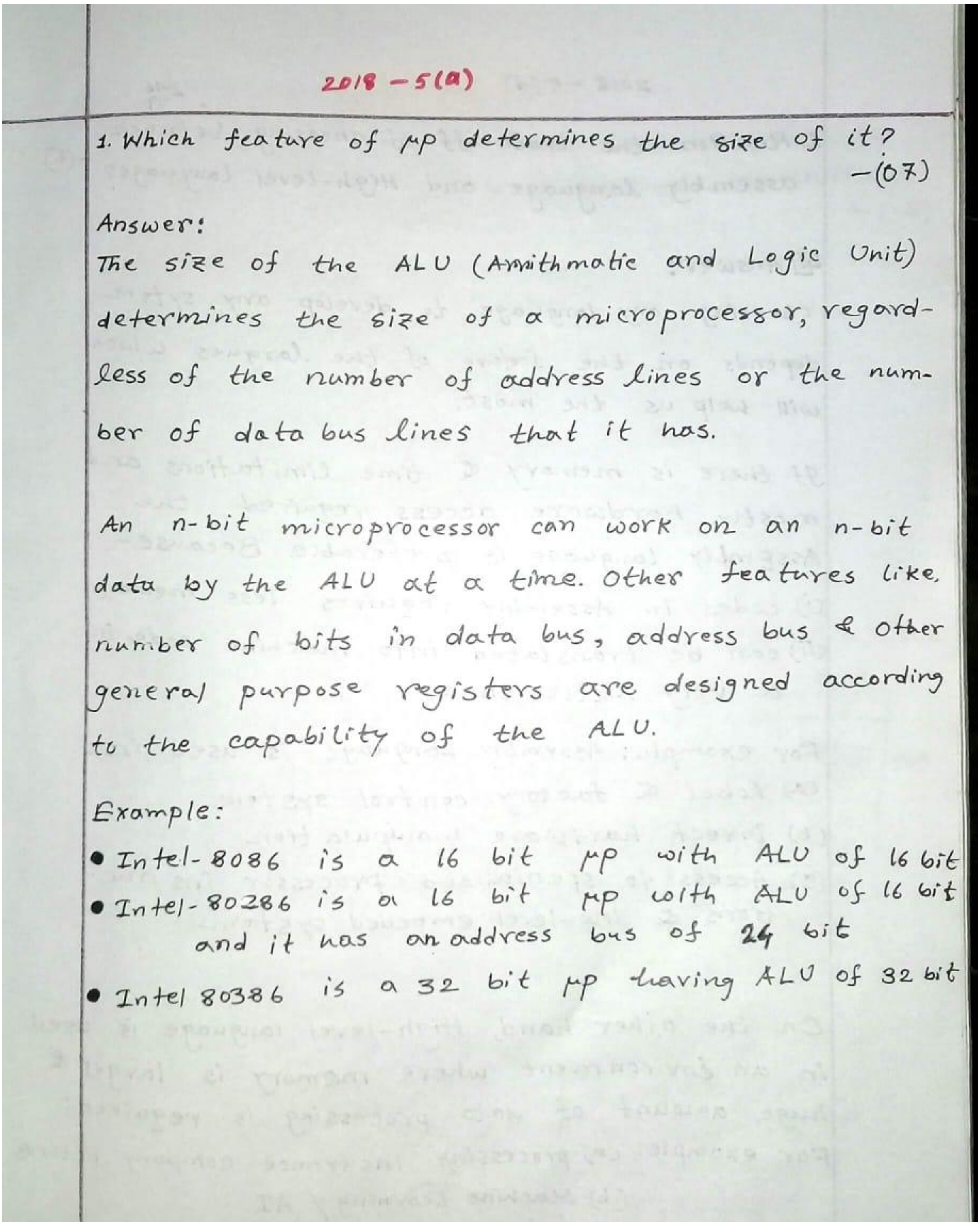
2018

2018 - 5(a):



2018 – 5(a)

1. Which feature of μP determines the size of it? –(07)

Answer:

The size of the ALU (Arrithmatic and Logic Unit) determines the size of a microprocessor, regardless of the number of address lines or the number of data bus lines that it has.

An n-bit microprocessor can work on an n-bit data by the ALU at a time. Other features like, number of bits in data bus, address bus & other general purpose registers are designed according to the capability of the ALU.

Example:

- Intel-8086 is a 16 bit μP with ALU of 16 bit
- Intel-80286 is a 16 bit μP with ALU of 16 bit and it has an address bus of 24 bit
- Intel 80386 is a 32 bit μP having ALU of 32 bit

2018 - 5(b):

Function of 8086 queue:

- While the EU is decoding an instruction or executing an instruction, which does not require use of the buses, the BIU fetches up to six instruction bytes for the following instructions.
- The BIU stores these pre-fetched bytes in a first-in-first-out register set called a queue.
- When the EU is ready for its next instruction from the queue in the BIU. This is much faster than sending out an address to the system memory and waiting for memory to send back the next instruction byte or bytes.
- Except in the case of JMP and CALL instructions, where the queue must be dumped and then reloaded starting from a new address, this pre-fetch and queue scheme greatly speeds up processing.
- Fetching the next instruction while the current instruction executes is called pipelining.

Queue speeds up processing :

The instruction prefetch queue speeds up the processing of microprocessors by attempting to have the next opcode bytes available to the execution unit before it actually needs them. This works because, statistically, there is time spent by the execution unit in executing a particular instruction; time that the bus interface unit can use to go ahead and prefetch the next opcode bytes. Sometimes, this results in a loss of time, because the execution unit may branch to some other location. Modern processors attempt to sidestep that by using branch prediction algorithms.

2018 - 5(c):

The flag register is one of the special purpose register. The flag bits are changed to 0 or 1 depending upon the value of result after arithmetic or logical operations.

8086 has 16-bit flag register, and there are 9 valid flag bits. The format of flag register is like below.

Bits	**D_{15}**	**D_{14}**	**D_{13}**	**D_{12}**	**D_{11}**	**D_{10}**	**D_9**	**D_8**	**D_7**	**D_6**	**D_5**	**D_4**	**D_3**	**D_2**	**D_1**	**D_0**
Flags					O	D	I	T	S	Z		AC		P		CY

We can divide the flag bits into two sections. The Condition Flags, and the Control Flags.

Condition Flags

In 8086 there are 6 different flags which are set or reset after 8-bit or 16-bit operations. These flags and their functions are listed below.

Flag Bit	**Function**
S	After any operation if the MSB is 1, then it indicates that the number is negative. And this flag is set to 1
Z	If the total register is zero, then only the Z flag is set
AC	When some arithmetic operations generates carry after the lower half and sends it to upper half, the AC will be 1
P	This is even parity flag. When result has even number of 1, it will be set to 1, otherwise 0 for odd number of 1s
CY	This is carry bit. If some operations are generating carry after the operation this flag is set to 1

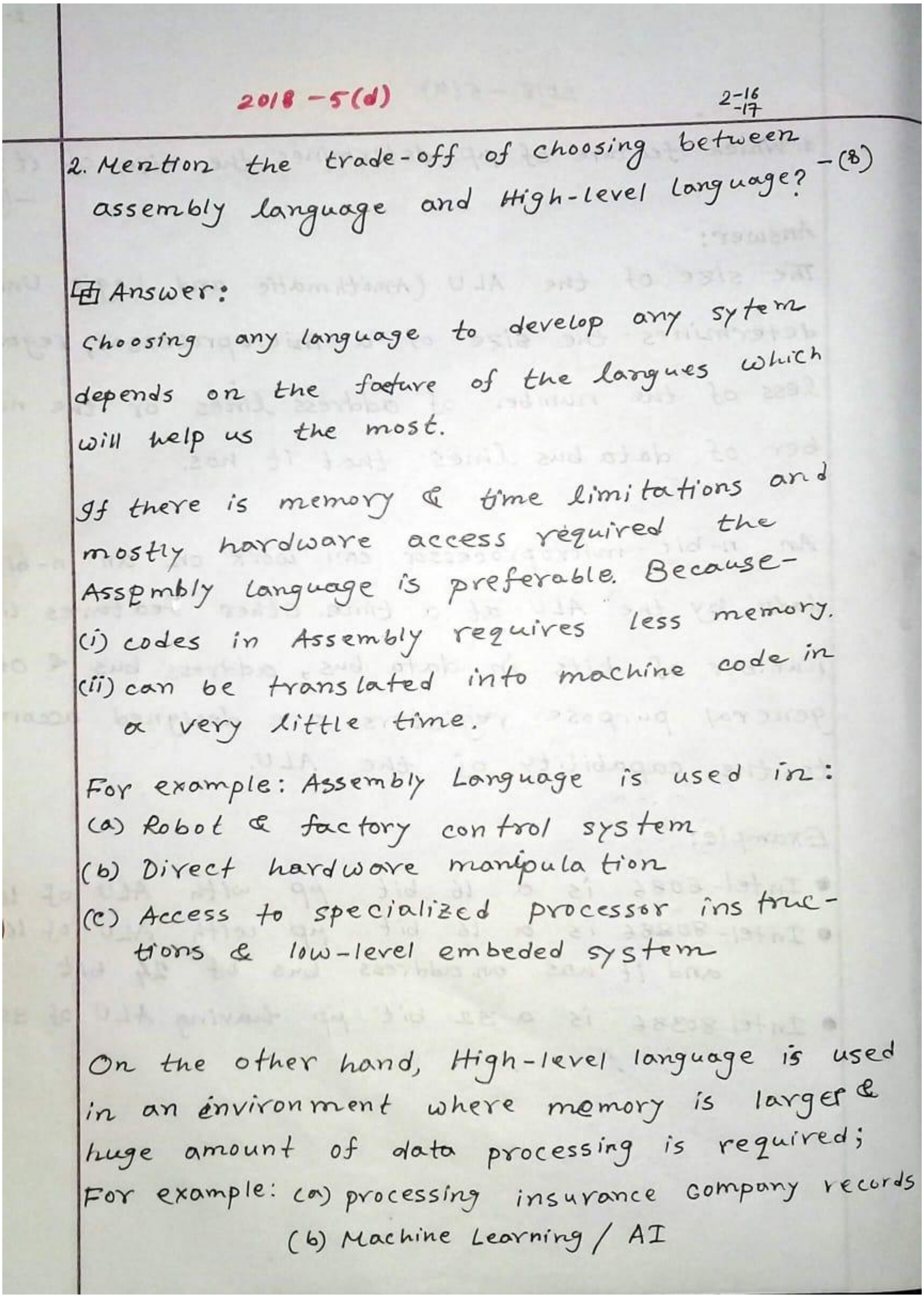
Flag Bit	Function
O	The overflow flag is set to 1 when the result of a signed operation is too large to fit.

Control Flags

In 8086 there are 3 different flags which are used to enable or disable some basic operations of the microprocessor. These flags and their functions are listed below.

Flag Bit	Function
D	This is directional flag. This is used in string related operations. D = 1, then the string will be accessed from higher memory address to lower memory address, and if D = 0, it will do the reverse.
I	This is interrupt flag. If I = 1, then MPU will recognize the interrupts from peripherals. For I = 0, the interrupts will be ignored
T	This trap flag is used for on-chip debugging. When T = 1, it will work in a single step mode. After each instruction, one internal interrupt is generated. It helps to execute some program instruction by instruction.

2018 5(d):



2018 – 5(d)

2-16
-17

2. Mention the trade-off of choosing between assembly language and High-level language? –(8)

Answer:

Choosing any language to develop any sytem depends on the facture of the langues which will help us the most.

If there is memory & time limitations and the mostly hardware access required the Assembly language is preferable. Because–

(i) codes in Assembly requires less memory.

(ii) can be translated into machine code in a very little time.

For example: Assembly Language is used in:

(a) Robot & factory control system

(b) Direct hardware manipulation

(c) Access to specialized processor instructions & low-level embeded system

On the other hand, High-level language is used in an environment where memory is larger & huge amount of data processing is required;

For example: (a) processing insurance company records

(b) Machine Learning / AI

2018 6(a):

Bit slicing is a method of combining processor modules to multiply the word length. Bit slicing was common with early processors, notably the AMD (Advanced Micro Devices) 2900 series that originated in 1975.

In a bit-sliced processor, each module contains an ALU (arithmetic-logic unit) usually capable of handling a 4-bit field. By combining two or more identical modules, it is possible to build a processor that can handle any multiple of this value, such as 8 bits, 12 bits, 16 bits, 20 bits, and so on. Each module is called a slice. The control lines for all the slices are connected effectively in parallel to share the processing "work" equally.

A **bit-slice microprocessor** (**BSM**) is a microprocessor designed as a module with the primary purpose of being able to assemble multiple identical such microprocessors to form a larger processor of some desired word size. Bit-slice microprocessors can be cascaded to produce any conventional (e.g. 4-bit, 8-bit, 16-bit) as well as unconventional word sizes (e.g. 10-bit, 12-bit, 18-bit). A notable advantage of a BSM over discrete logic components is the fact that most connections are internal to the chip with only few connections being external.

A departure from normal microprocessors is that fact that many bit-slice chips do not have an instruction set architecture. Such bit slicing systems allow designers to create their own architecture and other key characteristics such as I/O pins and address width. This flexibility of course came with overall more expensive system and larger amount of ICs.

2018 6(b):

The instructions to set the trap flag are:

```
PUSHF                             ; Push flags on stack
MOV BP,SP                         ; Copy SP to BP for use as index
OR WORD PTR[BP+0],0100H           ; Set TF flag
POPF                              ; Restore flag Register
```

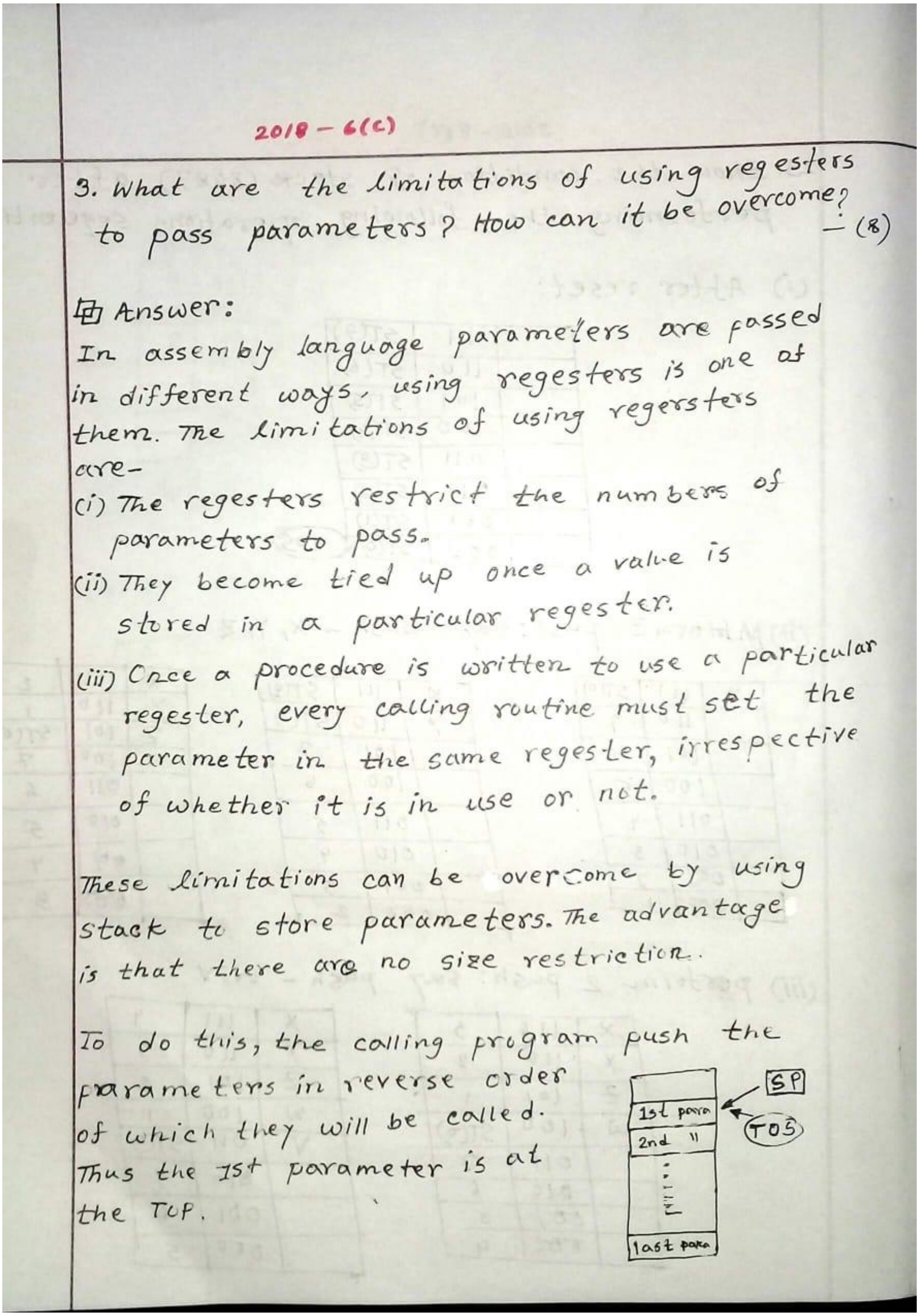
The instructions to set the trap flag are:

```
PUSHF                              ; Push flags on stack
MOV BP,SP                         ; Copy SP to BP for use as index
AND WORD PTR[BP+0],0FEFFH          ; Set TF flag
POPF                              ; Restore flag Register
```

The 8086 has no instruction to directly set or reset the trap flag. These operations are done by pushing the flag register on the stack, changing the trap flag bit to what the programmer wants it to be, and then popping the flag register back off the stack.

To reset the trap flag, simply replace the OR instruction in the preceding sequence with the instruction AND.

2018 6(c):



2018 – 6(c)

3. What are the limitations of using regesters to pass parameters? How can it be overcome? – (8)

⊞ Answer:

In assembly language parameters are passed in different ways, using regesters is one of them. The limitations of using regersters are–

(i) The regesters restrict the numbers of parameters to pass.

(ii) They become tied up once a value is stored in a particular regester.

(iii) Once a procedure is written to use a particular regester, every calling routine must set the parameter in the same regester, irrespective of whether it is in use or not.

These limitations can be overcome by using stack to store parameters. The advantage is that there are no size restriction.

To do this, the calling program push the parameters in reverse order of which they will be called. Thus the 1st parameter is at the TOP.

2018 6(d):

Near procedure	**Far Procedure**
1. A near procedure refers to a procedure which is in the same code segment from that of the call instruction	1. A far procedure refers to a procedure which is in the different code segment from that of the call instruction.
2. It is also called intra-segment procedure	2. It is also called inter-segment procedure call
3 A near procedure call replaces the old IP with new IP.	3. A far procedure call replaces the old CS:IP pairs with new CS:IP pairs
4. The value of old IP is pushed on to the stack. SP=SP-2	4. The value of the old CS:IP pairs are pushed on to the stack SP=SP-2
5. Less stack locations are required	5. More stack locations are required
6. Procedure is inside Code Segment	6. Procedure is outside Code Segment
7.Uses keyword near	7. Uses keyword FAR
8.Syntax:- CALL near_procedure_name	8. Example :- Call FAR PTR Delay
9. Example :- Call Delay	9. Syntax:- CALL far_procedure_name

2018 6(e):

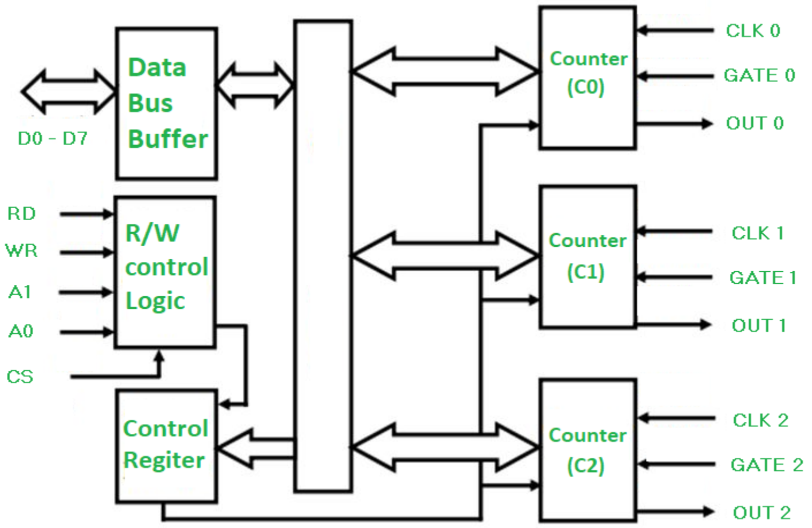
Below are the advantages of assembly language:

1. It allows complex jobs to run in a simpler way.
2. It is memory efficient, as it requires less memory.
3. It is faster in speed, as its execution time is less.
4. It is mainly hardware oriented.
5. It requires less instruction to get the result.
6. It is used for critical jobs.
7. It is not required to keep track of memory locations.
8. It is a low-level embedded system.

2018 7(a):

8254 is a device designed to solve the timing control problems in a microprocessor. It has 3 independent counters, each capable of handling clock inputs up to 10 MHz and size of each counter is 16 bit. It operates in +5V regulated power supply and has 24 pin signals. All modes are software programmable. The 8254 is an advanced version of 8253 which did not offered the feature of read back command.

The basic block diagram of 8254 is:



It has 3 counters each with two inputs (Clock and Gate) and one output. Gate is used to enable or disable counting. When any value of count is loaded and value of gate is set(1), after every step value of count is decremented by 1 until it becomes zero.
Depending upon the value of CS, A1 and A0 we can determine addresses of selected counter.

CS	A1	A0	SELECETION
0	0	0	C0
0	0	1	C1
0	1	0	C2
0	1	1	Control Register

Applications –

1. To generate accurate time delay
2. As an event counter
3. Square wave generator
4. Rate generator
5. Digital one shot

2018 7(b):

(i) Mode - 1 : Hardware retrigger able one-shot

- A trigger signal in GATE activates the counting.
- OUT will be initially high. OUT will go low on the CLK pulse following a trigger and will remain low until the Counter reaches zero.
- If GATE is triggered again in between counting, then the counter reloaded and starts counting from the beginning. Thus called retrigger able.
- If a new count is written while output is low, then the current one-shot has no affect unless the counter is retriggered.

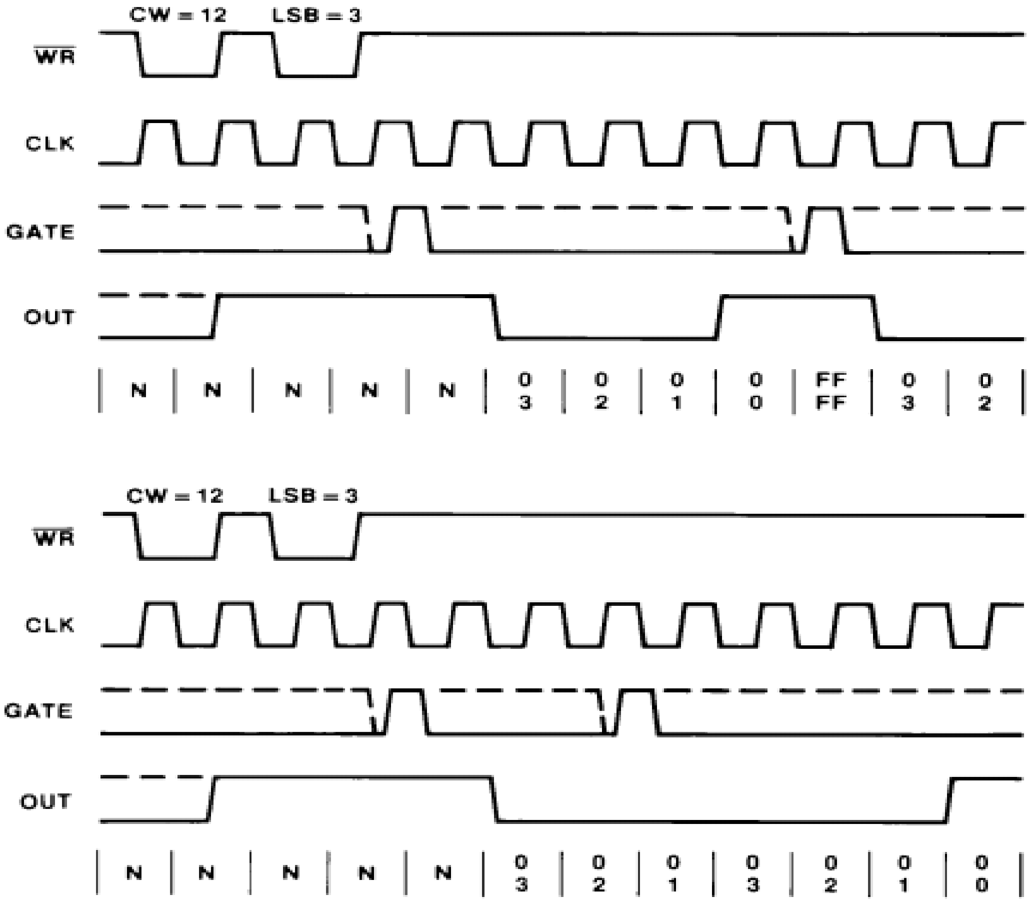


Figure : Hardware retrigger able one-shot

Mode : 5 – Hardware Triggered strobe Mode

- This mode generates a strobe in response to an externally generated signal.
- This mode is similar to mode 4 except that the counting is initiated by a signal at the gate input, which means it is hardware triggered instead of software triggered.
- After it is initialized, the output goes high.
- When the terminal count is reached, the output goes low for one clock cycle.

Mode 5 : Hardware triggered strobe

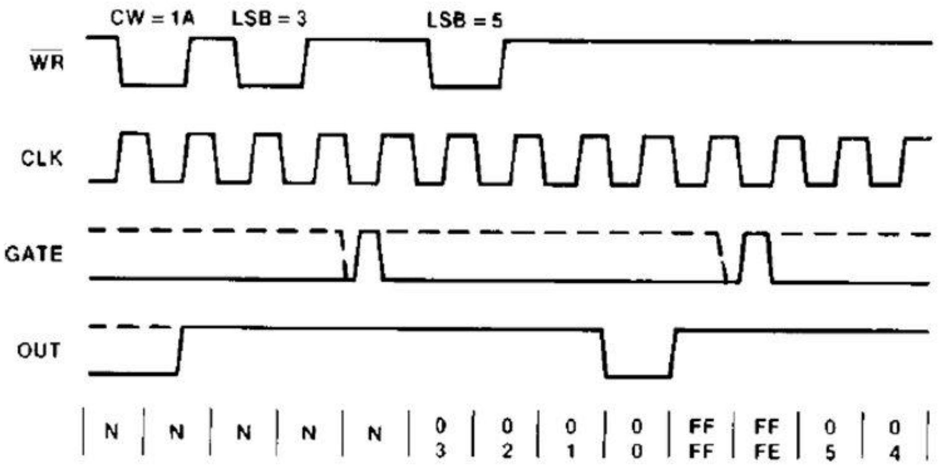


Figure : Hardware Triggered strobe Mode

2018 7(c):

Real Mode:

Following a system reset the 386 is initialized in Real Mode. In this mode the chip looks virtually identical to an 8086. That is, it has the following features:

The address space is limited to 1 MB using address lines AO-A19 (the high address lines A20-A31 are inactive).

The segment memory addressing mechanism of the 8086 is retained with each segment limited to 64 KB.

Two new features are available to the programmer in Real Mode—access to the 32-bit register set of the 386 and the addition of two new segments called F and G. This will be explained in more detail when the programming model is introduced later in this chapter.

Protected Mode:

The primary difference between Real Mode and Protected Mode is the latter's new addressing mechanism and protection levels.

Although memory segments are still retained, each segment may range from a single byte to 4 GB (the full physical address space of the 386).

The addresses stored in the segment registers are now interpreted as pointers into a descriptor table.

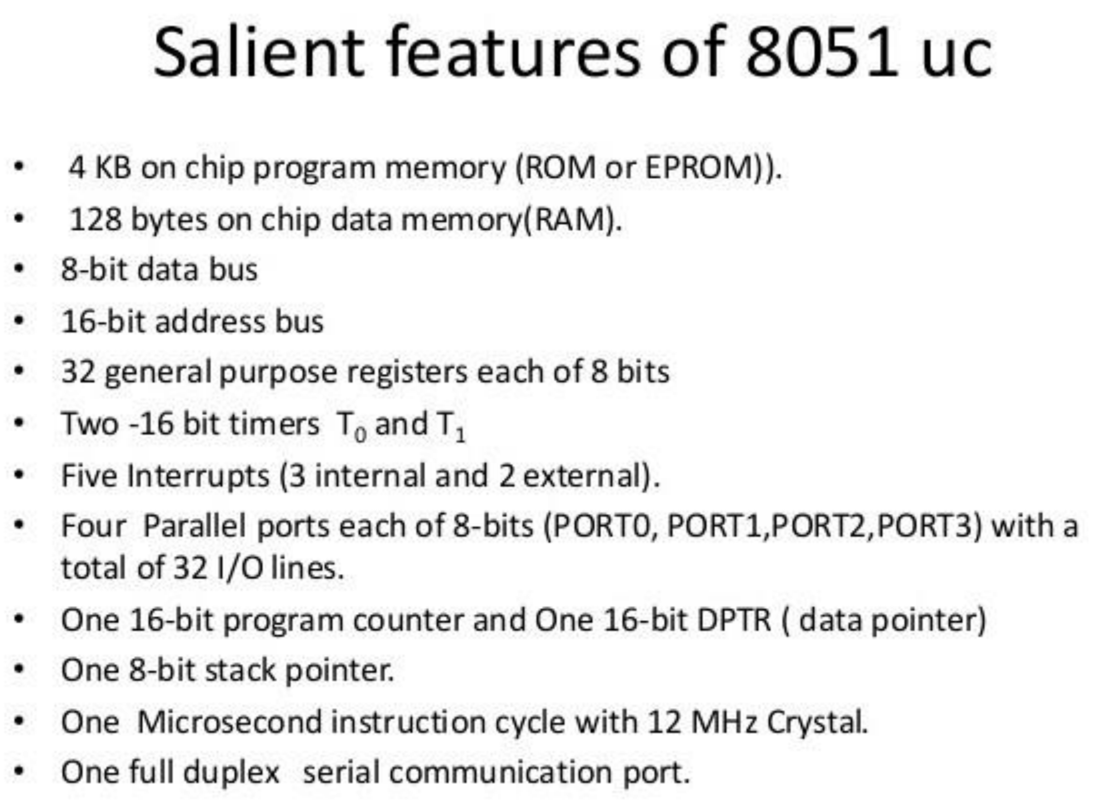
Each segment's entry in this table is eight bytes long and identifies the 32-bit base address of the segment, the segment size, and the access rights. Memory addresses are computed by adding the offset specified by the instruction to the segment base address.

2018 - 7(d):

Limitations of 80286:

- 16-bit ALU
- 64K segment size
- cannot be easily switched back and forth between real and protected mode–to come back to the real mode from protected mode, you have to switched off the 80286

2018 - 8(a):



Salient features of 8051 uc

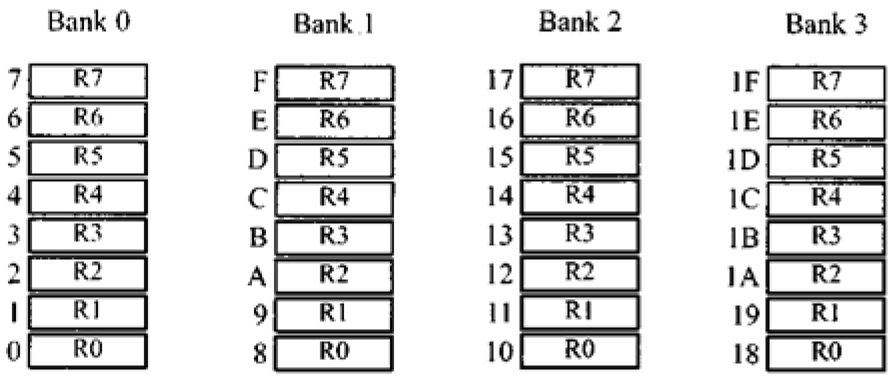
- 4 KB on chip program memory (ROM or EPROM)).
- 128 bytes on chip data memory(RAM).
- 8-bit data bus
- 16-bit address bus
- 32 general purpose registers each of 8 bits
- Two -16 bit timers T_0 and T_1
- Five Interrupts (3 internal and 2 external).
- Four Parallel ports each of 8-bits (PORT0, PORT1,PORT2,PORT3) with a total of 32 I/O lines.
- One 16-bit program counter and One 16-bit DPTR (data pointer)
- One 8-bit stack pointer.
- One Microsecond instruction cycle with 12 MHz Crystal.
- One full duplex serial communication port.

2018 - 8(b):

Register banks

A total of 32 bytes of RAM are set aside for the register banks and stack. These 32 bytes are divided into 4 banks of registers in which

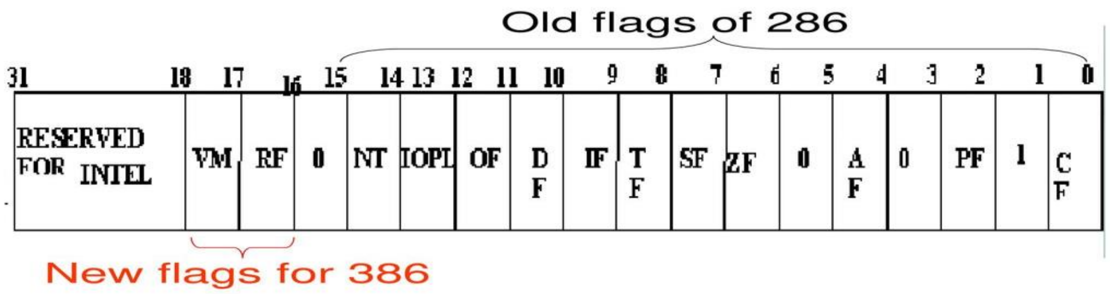
each bank has 8 registers, RO – R7. RAM locations from 0 to 7 are set aside for bank 0 of RO – R7 where RO is RAM location 0, Rl is RAM location 1, R2 is location 2, and so on, until memory location 7, which belongs to R7 of bank 0. The second bank of registers RO – R7 starts at RAM location 08 and goes to location OFH. The third bank of RO – R7 starts at memory location 10H and goes to location 17H. Finally, RAM locations 18H to 1FH are set aside for the fourth bank of RO – R7. The following shows how the 32 bytes are allocated into 4 banks:



Bank 0		Bank 1		Bank 2		Bank 3	
7	R7	F	R7	17	R7	1F	R7
6	R6	E	R6	16	R6	1E	R6
5	R5	D	R5	15	R5	1D	R5
4	R4	C	R4	14	R4	1C	R4
3	R3	B	R3	13	R3	1B	R3
2	R2	A	R2	12	R2	1A	R2
1	R1	9	R1	11	R1	19	R1
0	R0	8	R0	10	R0	18	R0

2018 - 8(c):

Flag Register...



20

VM - *Virtual Mode Flag*

- If this flag is set to 1, the 80386 enters the virtual 8086 mode within the protection mode.
- When VM bit is 0, 386 operates in protected mode
- This is to be set only when the 80386 is in protected mode.
- This bit can be set using IRET instruction or any task switch operation only in the protected mode.

RF- Resume Flag

- If RF=1, 386 ignores debug faults
  - does not take another exception so that an instruction can be restarted after a normal debug exception.
- If RF=0, 386 takes another debug exception to service debug faults

- This flag is used with the debug register breakpoints.
- It is checked at the starting of every instruction cycle and if it is set, any debug fault is ignored during the instruction cycle.
- The RF is automatically reset after successful execution of every instruction, except for IRET and POPF instructions.

- Also, it is not automatically cleared after the successful execution of JMP, CALL and INT instruction causing a task switch.

- These instruction are used to set the RF to the value specified by the memory data available at the stack.

2018 - 8(d):

Logical address: the address you work with in a program are called logical address.

- Logical address indicates the logical positions of code and data.
- An example of this is the 8086 instruction JNZ NEXT.
- The level NEXT represents a logical address that execution will go to if the zero flag is not set.
- When an 8086 program is assembled, each logical address is represented with a 16-bit offset and a 16-bit segment base.
- The 8086 BIU then produces the actual physical memory address by simply adding these two parts together.

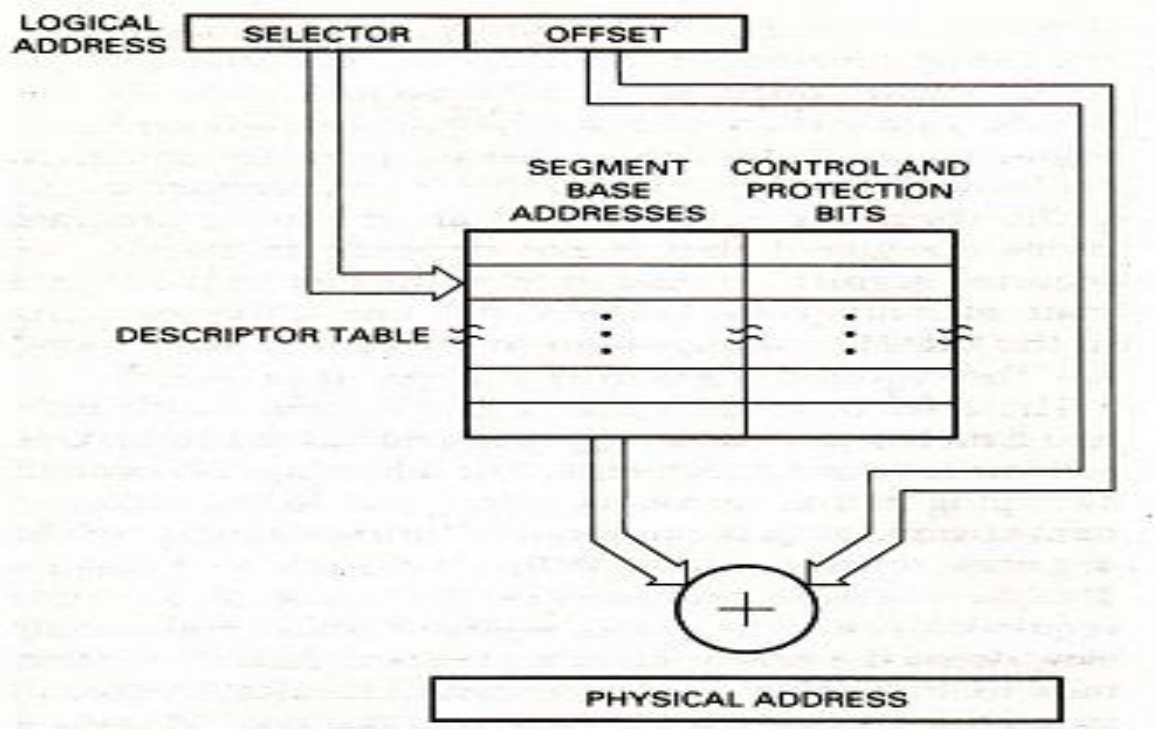


FIGURE 15-8 Block diagram showing how segment-based virtual memory is implemented in 80286, 80386, and 80486 processors.

- When a program is assembled or compiled to run on a system with an MMU, each logical or virtual address is also represented by two components, but the components function differently.
- In a segment-oriented system such as an 80286, the upper 16-bit component is referred to as a segment selector and the lower component is referred to as the offset.
- MMU uses the segment selector to access a *descriptor* for the desired segment in a table of descriptors in memory.
- A descriptor is a series of memory locations that contain the physical base address for a segment, the privilege level of the segment and some control bits.
- The selector for the 80286,80386 and 80486 have 14 address bits and 2 privilege-level bits.
- The 14 address bits in the selector can select any one of 16384 descriptors in the descriptor table.
- Since each descriptor represents a segment, this means that a program can access up to 16384 segments.
- For an 80286 the offset part of the virtual address is 16 bits, so each segment can contain up to 64 Kbytes.
- The logical or virtual address space accessible by an 80286 then is 16384 segments X 65536 bytes/segment, or 1 Gbyte.

2018 - 8(e):

80186 Basic Features

□ The 80186 contains 16 – bit data bus

□ The internal register structure of 80186 is virtually identical to the 8086

□ About the only difference is that the 80186 contain additional reserved interrupt vectors and some very powerful built-in I/O features 18-Nov-2009 ROSHAN FERNANDES, DEPT OF CSE 3

Clock Generator:

 The internal clock generator replaces the external 8284A clock generator used with the 8086 microprocessors. This reduces the component count in a system

Programmable Interrupt Controller:

 The PIC arbitrates all internal and external interrupts and controls up to two external 8259A PICs. When an external 8259 is attached, the 80186 microprocessors function as the master and the 8259 functions as the slave 18-Nov-2009 ROSHAN FERNANDES, DEPT OF CSE 4

Timers:

 The timer section contains three fully programmable 16-bit timers

 The timers 0 and 1 generate wave-forms for external use and driven by either the master clock of the 80186 or by an external clock

 The third timer, timer 2 is internal and clocked by the master clock

Programmable DMA Unit:

 The programmable DMA unit contains two DMA channels, or four DMA channels in some models

 Each channel can transfer data between memory locations, between memory and IO, or between IO devices

Programmable chip selection unit:

 The chip selection is a built-in programmable memory and I/O decoder

 It has 6 output lines to select memory, 7 lines to select I/O

Power save/Power Down Feature:

 The power save feature allows the system clock to be divided by 4, 8, or 16 to reduce power consumption

 The power saving feature is started by software and exited by a hardware event such as an interrupt

Refresh Control Unit:

 The refresh control unit generates the refresh row address at the interval programmed

2017

2017 - 5(a):

S.No	Microprocessor	Microcontroller
1	Microprocessor acts as a heart of computer system.	Microcontroller acts as a heart of embedded system.
2	It is a processor in which memory and I/O output component is connected externally.	It is a controlling device in which memory and I/O output component is present internally.
3	Since memory and I/O output is to be connected externally. Therefore the circuit is more complex.	Since on chip memory and I/O output component is available. Therefore the circuit is less complex.
4	It cannot be used in compact system. Therefore microprocessor is inefficient.	It can be used in compact system. Therefore microcontroller is more efficient.
5	Microprocessor has less number of registers. Therefore most of the operations are memory based.	Microcontroller has more number of registers. Therefore a program is easier to write.
6	A microprocessor having a zero status flag.	A microcontroller has no zero flag.
7	It is mainly used in personal computers.	It is mainly used in washing machines, air conditioners etc.

2017 - 5(b):

Step	1 10000011 10010010000000000000000

1	Sign = 1
	Exponent = 10000011
	Significand = 10010010000000000000000

2	100 = 10000011 - 01111111
3	1.1001001 2^4
4	11001.001
5	-25.125

2017 - 5(c):

- Logical address: the address you work with in a program are called logical address.
- Logical address indicates the logical positions of code and data.
- An example of this is the 8086 instruction JNZ NEXT.
- The level NEXT represents a logical address that execution will go to if the zero flag is not set.
- When an 8086 program is assembled, each logical address is represented with a 16-bit offset and a 16-bit segment base.
- The 8086 BIU then produces the actual physical memory address by simply adding these two parts together.

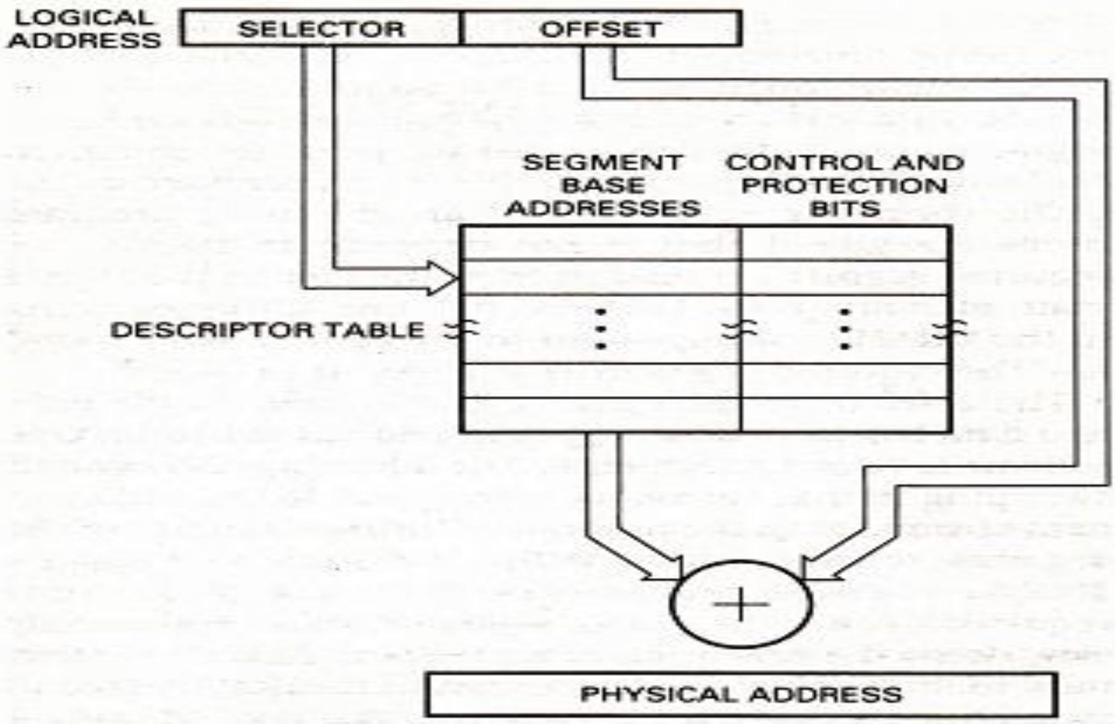


FIGURE 15-8 Block diagram showing how segment-based virtual memory is implemented in 80286, 80386, and 80486 processors.

- When a program is assembled or compiled to run on a system with an MMU, each logical or virtual address is also represented by two components, but the components function differently.

- In a segment-oriented system such as an 80286, the upper 16-bit component is referred to as a segment selector and the lower component is referred to as the offset.

- MMU uses the segment selector to access a *descriptor* for the desired segment in a table of descriptors in memory.

- A descriptor is a series of memory locations that contain the physical base address for a segment, the privilege level of the segment and some control bits.

- The selector for the 80286,80386 and 80486 have 14 address bits and 2 privilege-level bits.

- The 14 address bits in the selector can select any one of 16384 descriptors in the descriptor table.

- Since each descriptor represents a segment, this means that a program can access up to 16384 segments.

- For an 80286 the offset part of the virtual address is 16 bits, so each segment can contain up to 64 Kbytes.

- The logical or virtual address space accessible by an 80286 then is 16384 segments X 65536 bytes/segment, or 1 Gbyte.

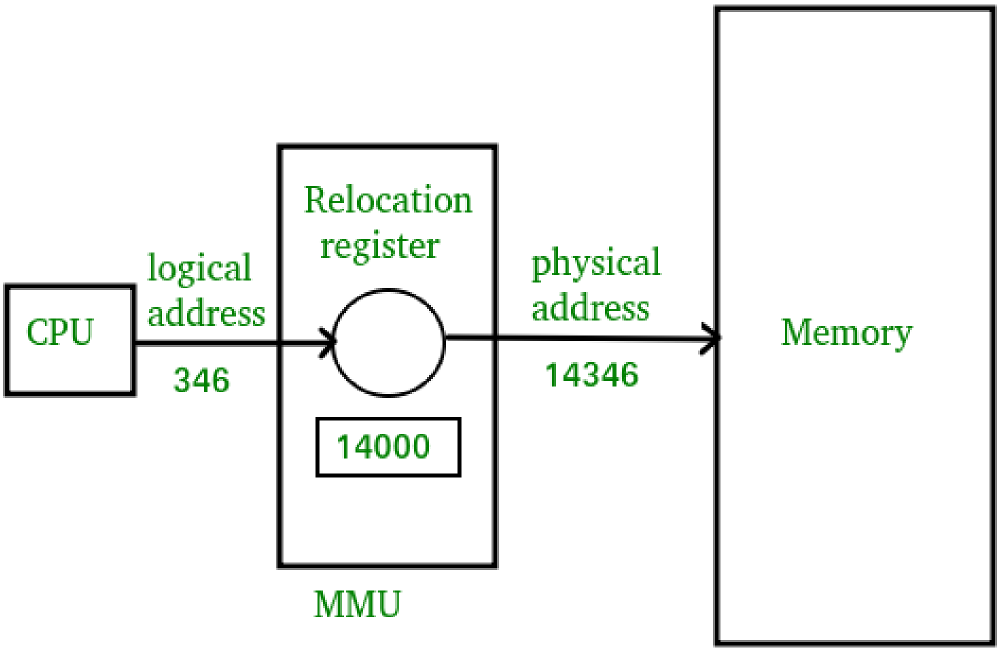
2017 - 5(d):

A computer’s memory management unit (MMU) is the physical hardware that handles its virtual memory and caching operations. The MMU is usually located within the computer’s central processing unit (CPU), but sometimes operates in a separate integrated chip (IC). All data request inputs are sent to the MMU, which in turn determines whether the data needs to be retrieved from RAM or ROM storage.

A memory management unit is also known as a paged memory management unit.

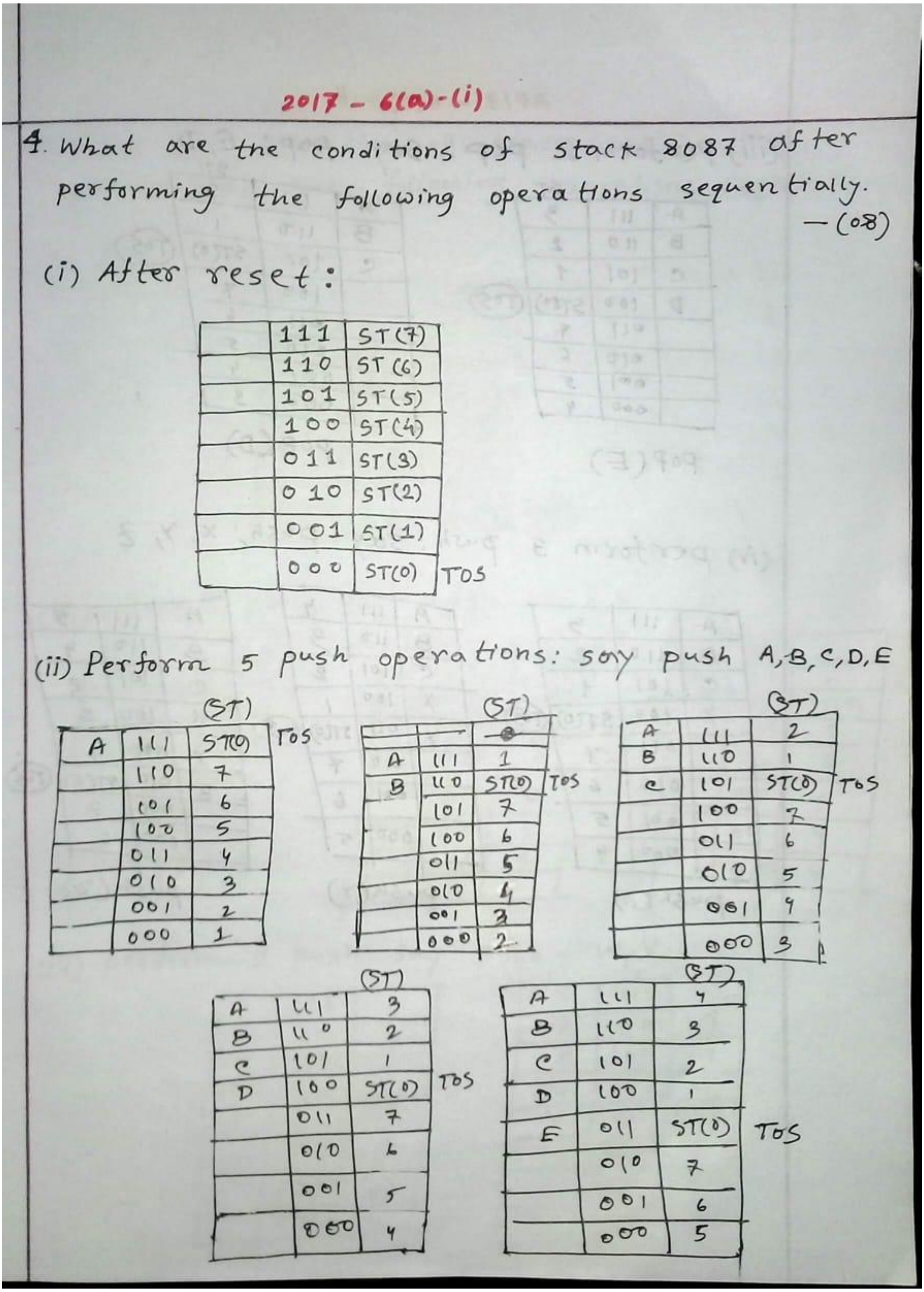
MMU manages segment based virtual memory using the following procedure given below:

- When the MMU receives a logical address from the CPU, it checks to see if that segment is currently in the physical memory.
- If the segment is present in physical memory, the MMU adds the offset component of the address to the segment base component of the address from the segment descriptor to form the physical address.
- It then outputs the physical address to memory on the memory address bus.
- If the MMU finds that the segment specified by the selector part of the logical address is not in memory, it sends an interrupt signal to the CPU.
- In response to the interrupt, the operating system executes an interrupt procedure which reads the desired code or data segment from disk and loads in into the physical memory.
- The MMU then computes and outputs the physical address.



1. CPU will generate logical address for eg: 346
2. MMU will generate relocation register(base register) for eg:14000
3. In Memory physical address is located eg:(346+14000= 14346)

2017 - 6(a):



2017 – 6(a)-(i)

4. What are the conditions of stack 8087 after performing the following operations sequentially. — (08)

(i) After reset:

	111	ST(7)	
	110	ST (6)	
	101	ST(5)	
	100	ST(4)	
	011	ST(3)	
	0 10	ST(2)	
	001	ST(1)	
	000	ST(0)	TOS

(ii) Perform 5 push operations: say push A,B,C,D,E

		(ST)	
A	111	ST(0)	TOS
	110	7	
	101	6	
	100	5	
	011	4	
	010	3	
	001	2	
	000	1	

		(ST)	
A	111	1	
B	110	ST(0)	TOS
	101	7	
	100	6	
	011	5	
	010	4	
	001	3	
	000	2	

		(ST)	
A	111	2	
B	110	1	
C	101	ST(0)	TOS
	100	7	
	011	6	
	010	5	
	001	9	
	000	3	

		(ST)	
A	111	3	
B	110	2	
C	101	1	
D	100	ST(0)	TOS
	011	7	
	010	6	
	001	5	
	000	4	

		(ST)	
A	111	4	
B	110	3	
C	101	2	
D	100	1	
E	011	ST(0)	TOS
	010	7	
	001	6	
	000	5	

2017 — 6(a) — ii

(iii) perform 2 pop: say pop; E, D

		ST	
A	111	3	
B	110	2	
C	101	1	
D	100	ST(0)	TOS
	011	7	
	010	6	
	001	5	
	000	4	

POP(E)

		ST	
A	111	2	
B	110	1	
C	101	ST(0)	TOS
	100	7	
	011	6	
	010	5	
	001	4	
	000	3	

POP(D)

(iv) perform 3 push: say push; x, y, z

A	111	3	
B	110	2	
C	101	1	
X	100	ST(0)	TOS
	011	7	
	010	6	
	001	5	
	000	4	

push(x)

A	111	4	
B	110	3	
C	101	2	
X	100	1	
Y	011	ST(0)	TOS
	010	7	
	001	6	
	000	5	

push(Y)

A	111	5	
B	110	4	
C	101	3	
X	100	2	
Y	011	1	
Z	010	ST(0)	TOS
	001	7	
	000	6	

push(Z)

2017 - 6(b):

Execution of 8086:

- The **EU** receives opcode of an instruction from the queue, decodes it and then executes it. While Execution, unit decodes or executes an instruction, then the BIU fetches instruction codes from the memory and stores them in the queue.
- The BIU and EU operate in parallel independently. This makes processing faster.
- General purpose registers, stack pointer, base pointer and index registers, ALU, flag registers (FLAGS), instruction decoder and timing and control unit constitute execution unit (EU). Let's discuss them:
-
- **General Purpose Registers:** There are four 16-bit general purpose registers: AX (Accumulator Register), BX (Base Register), CX (Counter) and DX. Each of these 16-bit registers are further subdivided into 8-bit registers as shown below:

16-bit registers	8-bit high-order registers	8-bit low-order registers
AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

- **Index Register:** The following four registers are in the group of pointer and index registers:
    - Stack Pointer (SP)
    - Base Pointer (BP)
    - Source Index (SI)
    - Destination Index (DI)
- **ALU:** It handles all arithmetic and logical operations. Such as addition, subtraction, multiplication, division, AND, OR, NOT operations.

- **Flag Register:** It is a 16?bit register which exactly behaves like a flip-flop, means it changes states according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups i.e. conditional and control flags.
    - **Conditional Flags:** This flag represents the result of the last arithmetic or logical instruction executed. Conditional flags are:
        - Carry Flag
        - Auxiliary Flag
        - Parity Flag
        - Zero Flag
        - Sign Flag
        - Overflow Flag
    - **Control Flags:** It controls the operations of the execution unit. Control flags are:
        - Trap Flag
        - Interrupt Flag
        - Direction Flag

Numeric Execution of 8087:

- This performs all operations that access and manipulate the numeric data in the coprocessor‟s registers.
- Numeric registers in NUE are 80 bits wide.
- NUE is able to perform arithmetic, logical and transcendental operations as well as supply a small number of mathematical constants from its on-chip ROM.
- Numeric data is routed into two parts ways a 64 bit mantissa bus and a 16 bit sign/exponent bus.

2017 - 6(c):

Steps to switch the 80286 to protected address mode from real address mode operation:

- The first step in switching an 80286 to protected mode is to set the protection enable bit in the machine status word register in the 80286.

- Bits 1,2 and 3 of the MSW are for the most part used to indicate whether a processor extension is present in the system or not.
- Bit 0 of the MSW is used to switch the 80286 into protected mode.
- To change bits in the MSW we have to load the desired word in a register or memory location and execute the load machine status word instruction.
- The final step to get the 80286 operating in protected mode is to execute an intersegment jump to the start of the main system program. This jump is necessary to flush the instruction byte queue because in protected mode the queue functions differently from the way it does in real mode.

2017 - 6(d):

Limitations of 80286:

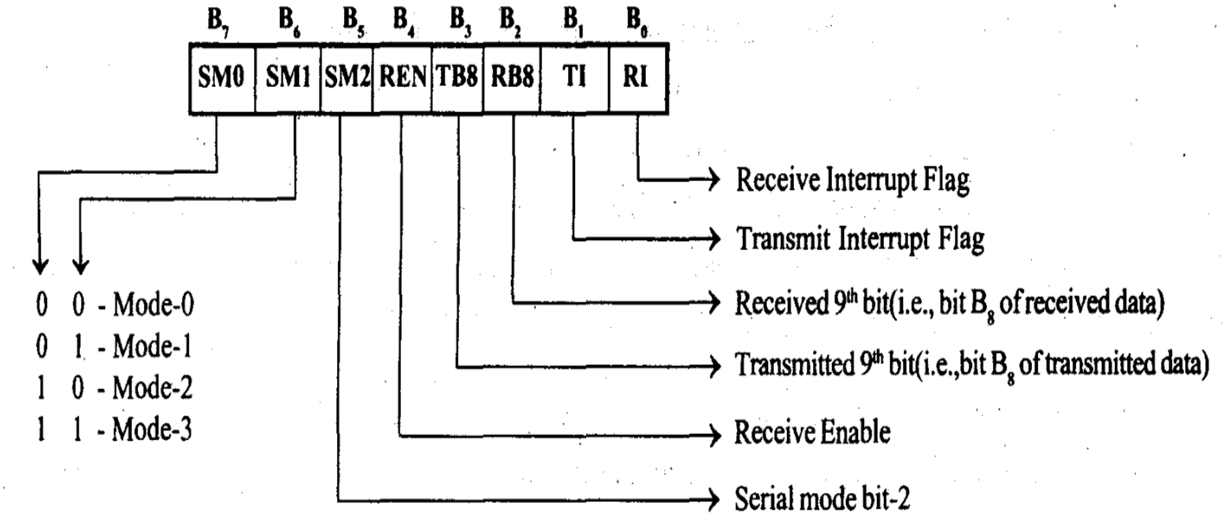
- 16-bit ALU
- 64K segment size
- cannot be easily switched back and forth between real and protected mode–to come back to the real mode from protected mode, you have to switched off the 80286

80386 was designed to overcome these limitations:

- 32 bit ALU
- segment size can be as large as 4G–a program can have as many as 16K segments. –So, a program has access to 4Gx16K=64TB of virtual memory
- 80386 has a virtual 86mode which allows easy switching between real and protected modes.

2017 – 7(a):

SERIAL PORT CONTROL REGISTER (SCON) OF 8051/8031 MICROCONTROLLER • The format of SCON register is shown.



- Mode 0:
  - ➢ In this mode the serial port function as half duplex serial port with fixed baud rate.
  - ➢ The 8- bit serial data is received and transmitted through RxD pin and the controller output the shift clock through TxD pin during reception and transmission.
  - ➢ The baud rate is fixed at 1 / 12 of the oscillator frequency.
- Mode 1:
  - ➢ In this mode the serial port function as full duplex serial port with variable baud rate.
  - ➢ In this mode one data consists of 10 bits, which includes one start bit, eight data bit and one stop bit. During reception the stop bit is stored as RB8 in SCON register.
  - ➢ Baud rate in mode-1 depends on the value of SMOD bit in PCON register and the timer1overflow rate.
- Mode 2:

- In this mode the serial port function as full duplex serial port with a baud rate of either 1/32 or 1/64 of the oscillator frequency.
- In this mode one data consists of 11 bits which includes one start bit, eight data bit, a programmable 9th data bit and one stop bit.
- During transmission the TB8 of SCON register is added as 9th data bit and during reception the 9th data bit is stored as RB8 in SCON register.
- The baud rate depends on the value of SMOD bit in PCON register.

- Mode 3:

  - The mode-3 is same as mode-2, except the baud rate.
  - In mode-3, the baud rate is variable. The baud rate depends on the value of SMOD bit in PCON register and the timer- 1 overflow rate.

2017 – 7(b):

Not found

2017 – 7(c):

The reasons why memory must be specially managed in a multitasking operating systems are given below:

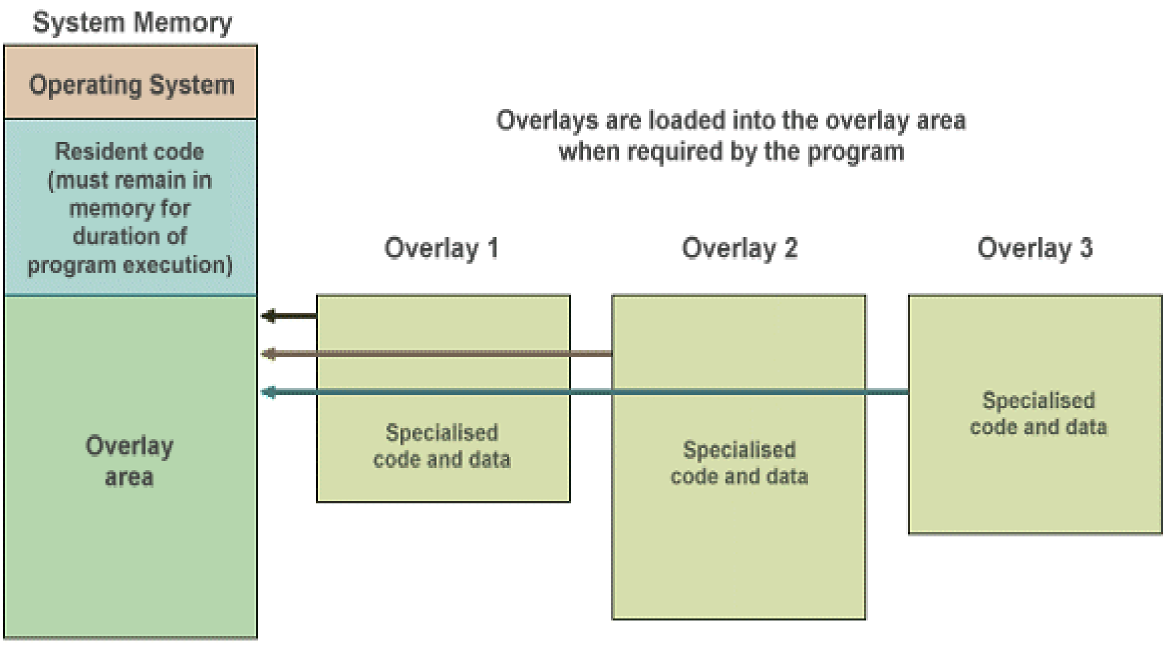
- There are two major reasons why memory must be specially managed in a multitasking operating systems.

- First reason is that the physical memory is usually not large enough to hold the operating system and all of the application programs that are being executed by the different users.

- The second reason is to make sure that executing tasks do not access protected areas of memory

Overlays

- A common problem in single-user systems is that the physical memory is not large enough to hold, for example an assembler and the program being assembled.
- The traditional solution is to write assembler in modules and use an overlay scheme.
- When the assembler is invoked, the executive module of the assembler is loaded into memory, and the additional block of memory space called the overlay area is reserved for the assembler.
- The first module of the assembler is loaded into the overlay area.


System Memory
Operating System
Resident code (must remain in memory for duration of program execution)
Overlay area
Overlays are loaded into the overlay area when required by the program
Overlay 1
Overlay 2
Overlay 3
Specialised code and data
Specialised code and data
Specialised code and data

- When the assembler reaches a point where it needs the next module, it reads that module, referred to as an overlay, from the disk into the overlay area reserved in memory.

- When the assembler reaches a point where it needs another overlay, it reads that overlay from disk and loads it into the same overlay area in memory.
- The overlay approach is commonly used in with assemblers, compilers, word processors and spreadsheet programs.

Bank switching

- Another approach traditionally used to expand the available memory in a microcomputer is bank switching.
- Intel 8085/8086 has only 16 address lines, so they can directly address only 64 Kbytes of memory.
- The figure on the next slide shows how the amount of memory can be expanded beyond the address limit.
- The hardware is configured so that when the power is first turned on, the 16-Kbytes bank labeled bank 0 is enabled**.**

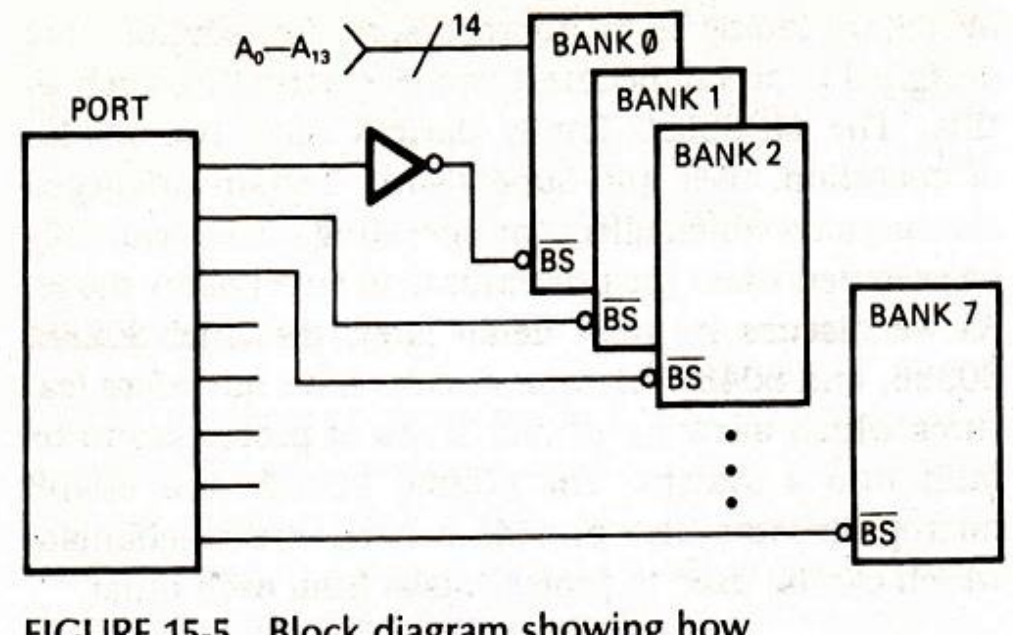


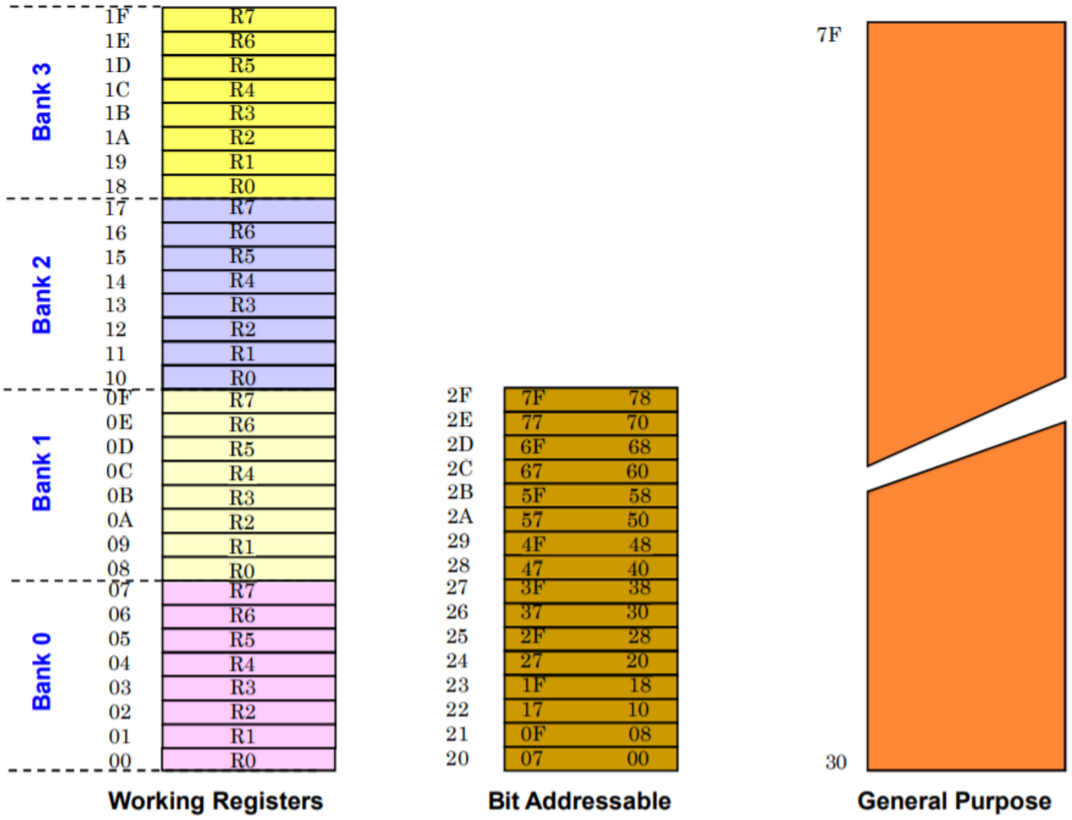
FIGURE 15-5 Block diagram showing how microcomputer memory can be expanded with bank switching.

- Let’s assume that this bank occupies system address space 4000H-7FFFH and that system address lines A0-A13 are used to address the bytes in this bank.
- To switch to bank 1, a byte which turns off bank 0 and turns on bank 1 is output to the selection port.
- The bank 1 devices now occupy the address space 4000H-7FFFH and system address lines A0-A13 are used to address the bytes in this bank.
- Any of the other banks can be switched into the 4000H-7FFFH memory window by simply sending the appropriate word to the control port.

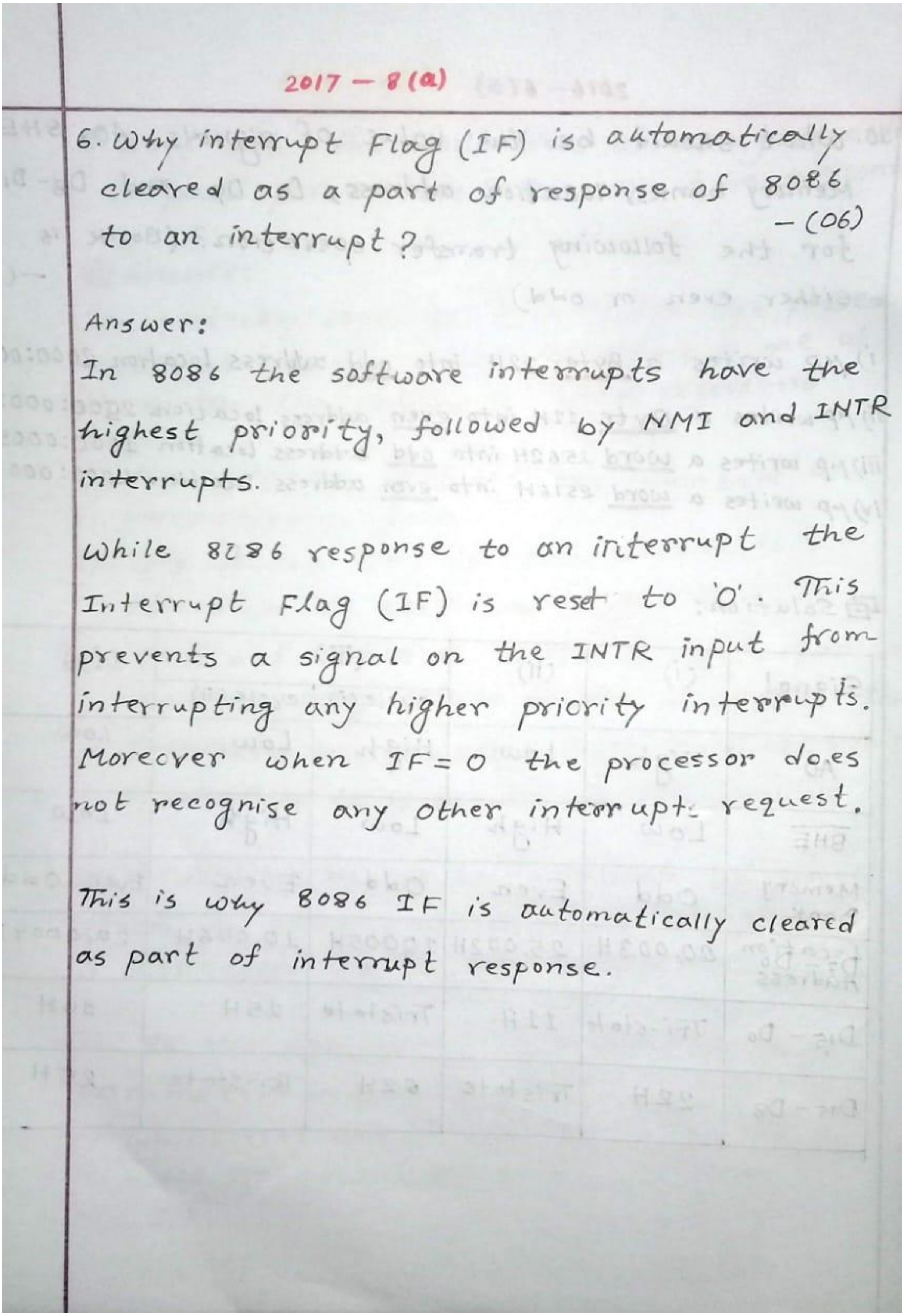
2017 – 7(d):

INTERNAL MEMORY

- A functioning computer must have memory for program code bytes, commonly in ROM, and
- RAM memory for variable data that can be altered as the program runs
- 8051 has internal RAM (128 bytes) and ROM (4Kbytes)
- 8051 uses the same address but in different memories for code and data
- Internal circuitry access the correct memory based on the nature of the operation in progress
- Can add memory externally if needed


Bank 3
1F R7
1E R6
1D R5
1C R4
1B R3
1A R2
19 R1
18 R0
Bank 2
17 R7
16 R6
15 R5
14 R4
13 R3
12 R2
11 R1
10 R0
Bank 1
0F R7
0E R6
0D R5
0C R4
0B R3
0A R2
09 R1
08 R0
Bank 0
07 R7
06 R6
05 R5
04 R4
03 R3
02 R2
01 R1
00 R0
Working Registers
2F 7F 78
2E 77 70
2D 6F 68
2C 67 60
2B 5F 58
2A 57 50
29 4F 48
28 47 40
27 3F 38
26 37 30
25 2F 28
24 27 20
23 1F 18
22 17 10
21 0F 08
20 07 00
Bit Addressable
7F
30
General Purpose

2017 – 8(a):



2017 — 8 (a)

6. Why interrupt Flag (IF) is automatically cleared as a part of response of 8086 to an interrupt? — (06)

Answer:

In 8086 the software interrupts have the highest priority, followed by NMI and INTR interrupts.

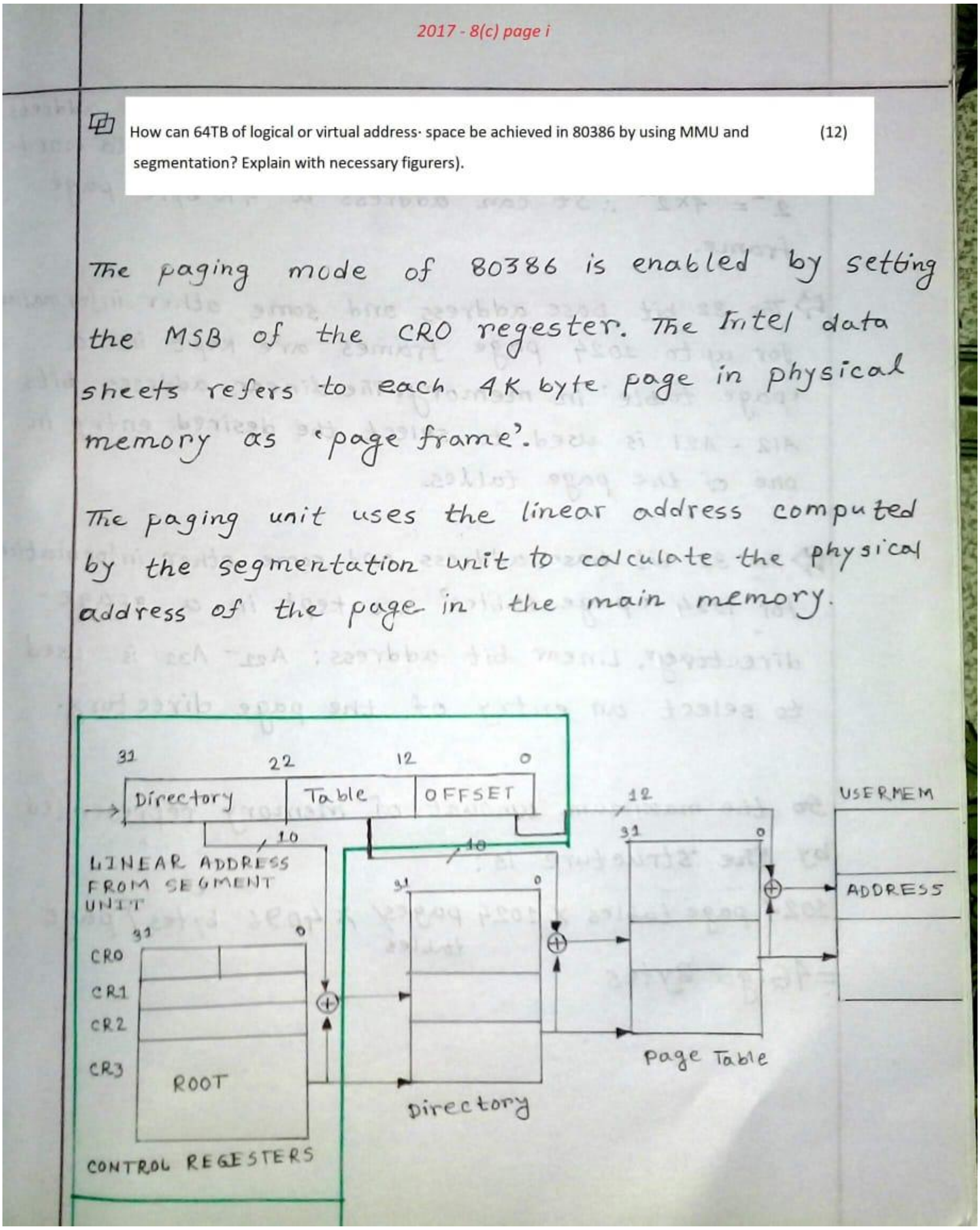
While 8286 response to an interrupt the Interrupt Flag (IF) is reset to '0'. This prevents a signal on the INTR input from interrupting any higher priority interrupts. Morecver when IF = 0 the processor does not recognise any other interrupt request.

This is why 8086 IF is automatically cleared as part of interrupt response.

2017 – 8(b):

Not found

2017 – 8(c):

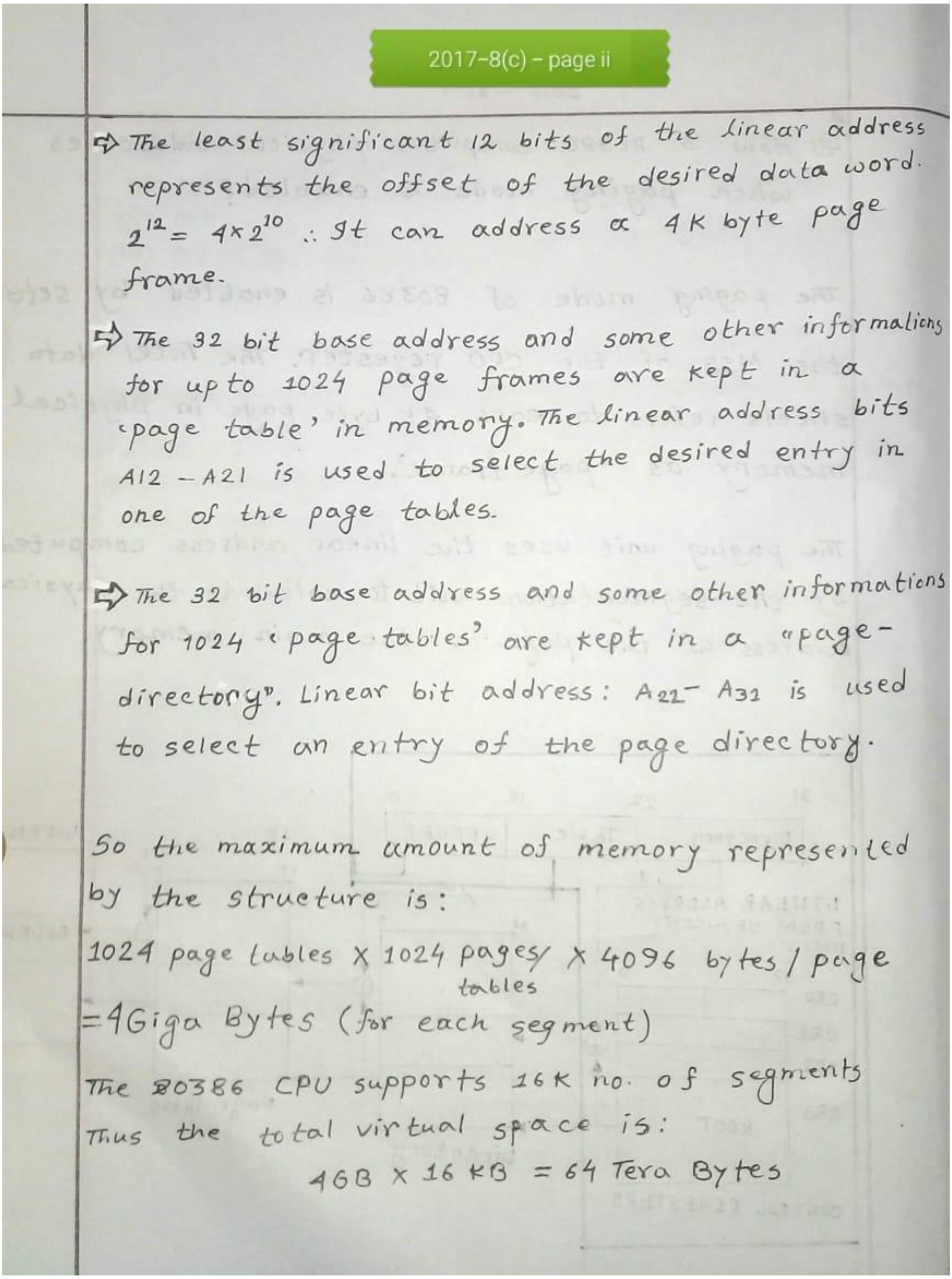


2017 - 8(c) page i

How can 64TB of logical or virtual address· space be achieved in 80386 by using MMU and segmentation? Explain with necessary figurers). (12)

The paging mode of 80386 is enabled by setting the MSB of the CRO regester. The Intel data sheets refers to each 4K byte page in physical memory as 'page frame'.

The paging unit uses the linear address computed by the segmentation unit to calculate the physical address of the page in the main memory.



2017-8(c) - page ii

⇨ The least significant 12 bits of the linear address represents the offset of the desired data word. $2^{12} = 4 \times 2^{10}$ ∴ It can address a 4K byte page frame.

⇨ The 32 bit base address and some other informations for up to 1024 page frames are kept in a 'page table' in memory. The linear address bits A12 – A21 is used to select the desired entry in one of the page tables.

⇨ The 32 bit base address and some other informations for 1024 'page tables' are kept in a "page-directory". Linear bit address: A_{22} – A_{31} is used to select an entry of the page directory.

So the maximum umount of memory represented by the structure is:

1024 page tables × 1024 pages/tables × 4096 bytes/page

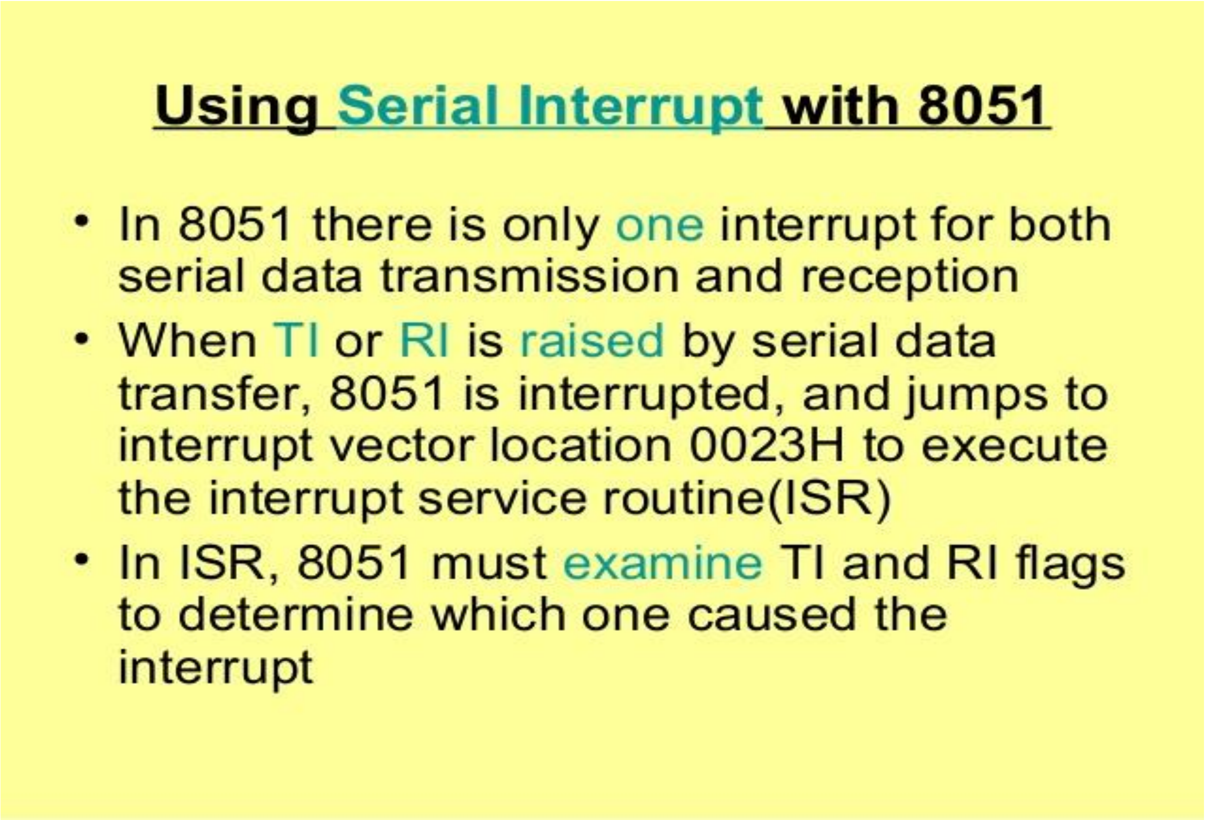
=4Giga Bytes (for each segment)

The 80386 CPU supports 16K no. of segments

Thus the total virtual space is:

4GB × 16 KB = 64 Tera Bytes

2017 – 8(d):



Using Serial Interrupt with 8051

- In 8051 there is only one interrupt for both serial data transmission and reception
- When TI or RI is raised by serial data transfer, 8051 is interrupted, and jumps to interrupt vector location 0023H to execute the interrupt service routine(ISR)
- In ISR, 8051 must examine TI and RI flags to determine which one caused the interrupt

Dual Role of Port 0 and Port 2:

- **Dual role of Port 0** − Port 0 is also designated as AD0–AD7, as it can be used for both data and address handling. While connecting an 8051 to external memory, Port 0 can provide both address and data. The 8051 microcontroller then multiplexes the input as address or data in order to save pins.
- **Dual 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 designated 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 up to 64KB and this is possible by 16 bit address bus because we know 216 = 64KB. Port2 is used for the upper 8-bit of the 16 bits address, and it cannot be used for I/O and this is the way any Program code of external ROM is addressed.

2016

2016 – 5(a):

1. Meeting the computing needs of the task at hand efficiently and cost effectively.

- Speed
- Power consumption
- The amount of RAM and ROM on chip
- The number of I/O pins and the timer on chip
- How easy to upgrade to higher performance or lower power-consumption versions.
- Cost per unit

2. Availability of software development tools, such as compilers, assemblers, and debuggers.

3. Wide availability and reliable sources of the microcontroller.

- The 8051 family has the largest number of diversified (multiple source) suppliers
  - Intel (original)
  - Atmel
  - Philips/Signetics
  - AMD
  - Infineon (formerly Siemens)
  - Dallas Semiconductor/Maxim

Von-Neumann Architecture

- only one bus.
- used for both data transfer and instruction fetches.
- cannot be performed at same time.

Harvard Architecture

- Separate data and instruction buses.
- Transfers to be performed simultaneously on both buses.

2016 – 5(b):

[Repeat – 2017 - 5(a)]

2016 – 5(c):

Not found

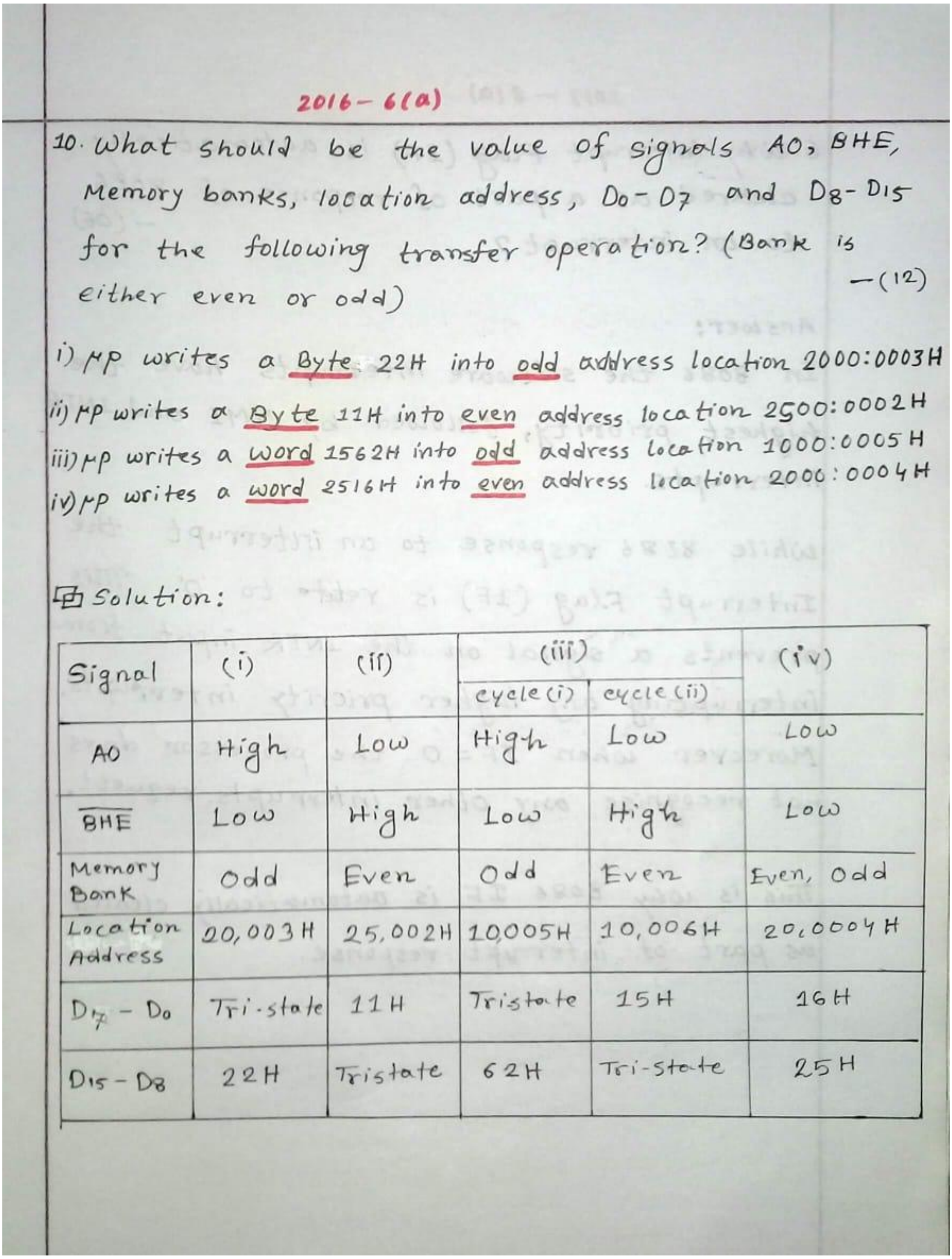
2016 – 5(d):

Ports	**Function**	**Pin number**
P3.0	RxD(serial input port)	10
P3.1	TxD(serial output port)	11
P3.2	INT0 (external Interrupt)	12
P3.3	INT1(external Interrput)	13
P3.4	T0(Timer/ Counter 0 external Input)	14
P3.5	T1(Timer/ Counter 1 external Input)	15
P3.6	WR (External data memory write strobe)	16
P3.7	RD(External data memory read strobe)	17

2016 – 5(e):

[Repeat – 2018 – 8(a)]

2016 – 6(a):



2016 – 6(a)

10. What should be the value of signals A0, BHE, Memory banks, location address, $D_0 - D_7$ and $D_8 - D_{15}$ for the following transfer operation? (Bank is either even or odd) —(12)

i) μP writes a Byte 22H into odd address location 2000:0003H

ii) μP writes a Byte 11H into even address location 2500:0002H

iii) μP writes a word 1562H into odd address location 1000:0005H

iv) μP writes a word 2516H into even address location 2000:0004H

Solution:

Signal	(i)	(ii)	(iii)		(iv)
			cycle (i)	cycle (ii)	
A0	High	Low	High	Low	Low
$\overline{BHE}$	Low	High	Low	High	Low
Memory Bank	Odd	Even	Odd	Even	Even, Odd
Location Address	20,003H	25,002H	10,005H	10,006H	20,0004H
$D_7 - D_0$	Tri-state	11H	Tristate	15H	16H
$D_{15} - D_8$	22H	Tristate	62H	Tri-state	25H

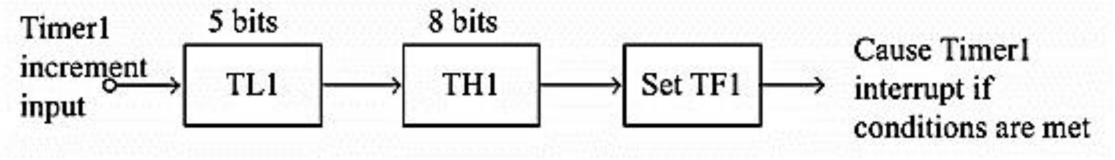
2016 – 6(b):

Not found

2016 – 6(c):

Mode 0 of Timer/Counter

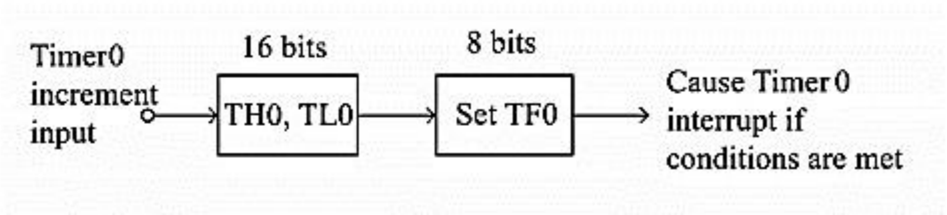
The Mode 0 operation is the 8-bit timer or counter with a 5-bit pre-scaler. So it is a 13-bit timer/counter. It uses 5 bits of TL0 or TL1 and all of the 8-bits of TH0 or TH1.



In this example the Timer1is selected, in this case, every 32 (25)event for counter operations or 32 machine cycles for timer operation, the TH1 register will be incremented by 1. When the TH1overflows from FFH to 00H, then the TF1 of TCON register will be high, and it stops the timer/counter. So for an example, we can say that if the TH1 is holding F0H, and it is in timer mode, then TF1will be high after 10H * 32 = 512 machine cycles.

Mode 1 of Timer/Counter

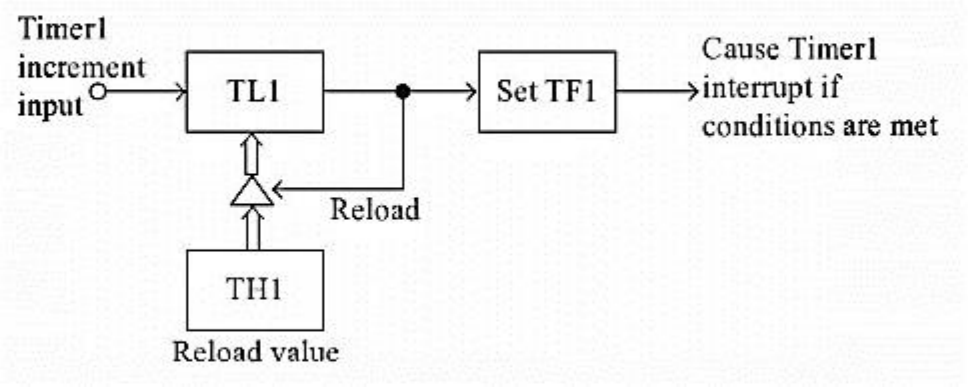
The Mode 1 operation is the 16-bit timer or counter. In the following diagram, we are using Mode 1 for Timer0.



In this case every event for counter operations or machine cycles for timer operation, the TH0– TL0 register-pair will be incremented by 1. When the register pair overflows from FFFFH to 0000H, then the TF0 of TCON register will be high, and it stops the timer/counter. So for an example, we can say that if the TH0 – TL0 register pair is holding FFF0H, and it is in timer mode, then TF0 will be high after 10H = 16 machine cycles. When the clock frequency is 12MHz, then the following instructions generate an interrupt 16 µs after Timer0 starts running.

Mode 2 ofTimer/Counter

The Mode 2 operation is the 8-bit auto reload timer or counter. In the following diagram, we are using Mode 2 for Timer1.

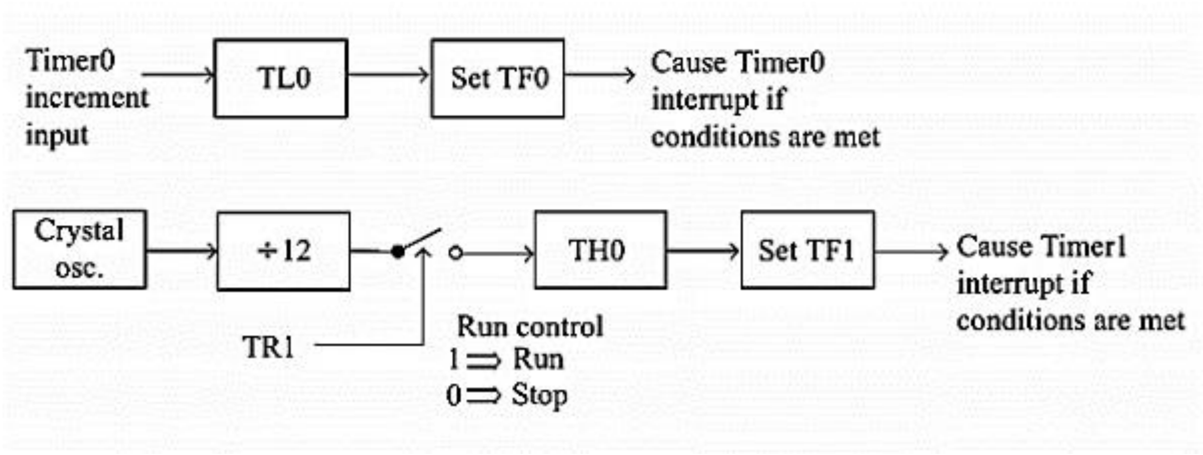


In this case every event for counter operations or machine cycles for timer operation, the TL1register will be incremented by 1. When the register pair overflows from FFH to 00H, then the TF1 of TCON register will be high, also theTL1 will be reloaded with the content of TH1 and starts the operation again.

So for an example, we can say that if the TH1 and TL1 register both are holding F0H and it is in timer mode, then TF1 will be high after 10H= 16 machine cycles. When the clock frequency is 12MHz this happens after 16 µs, then the following instructions generate an interrupt once every 16 µs after Timer1 starts running.

Mode 3 of Timer/Counter

Mode 3 is different for Timer0 and Timer1. When the Timer0 is working in mode 3, the TL0 will be used as an 8-bit timer/counter. It will be controlled by the standard Timer0 control bits, T0 and INT0 inputs. The TH0 is used as an 8-bit timer but not the counter. This is controlled by Timer1 Control bit TR1. When the TH0 overflows from FFH to 00H, then TF1 is set to 1. In the following diagram, we can Timer0 in Mode 3.



When the Timer1 is working in Mode 3, it simply holds the count but does not run. When Timer0 is in mode 3, the Timer1 is configured in one of the mode 0, 1 and 2. In this case, the Timer1 cannot interrupt the microcontroller. When the TF1 is used by TH0 timer, the Timer1 is used as Baud Rate Generator.

The meaning of gate bit in Timer0 and Timer1 for mode 3 is as follows

It controls the running of 8-bit timer/counter TL0 as like Mode 0, 1, or 2. The running of TH0 is controlled by TR1 bit only. So the gate bit in this mode for Timer0 has no specific role.

2016 – 6(d):

Not found

2016 – 7(a):

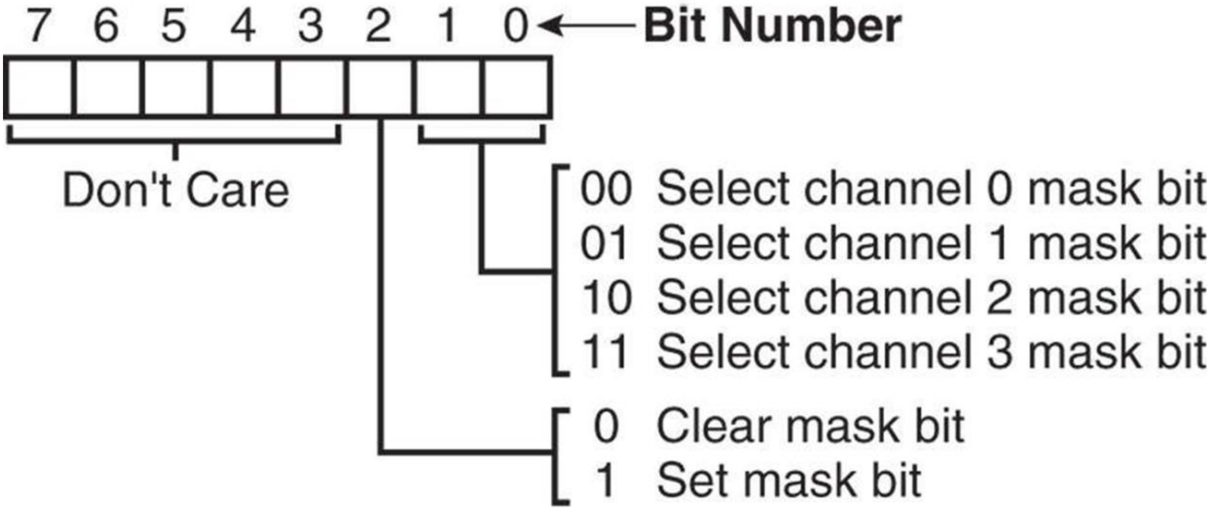
CWCR

- ❖ The current word count register programs a channel for the number of bytes to transfer during a DMA operation. The number loaded into this register is one less than the number of bytes transferred. For example, if 10 is loaded into the CWCR, then 11 bytes are transferred during DMA action.

RR

MSR

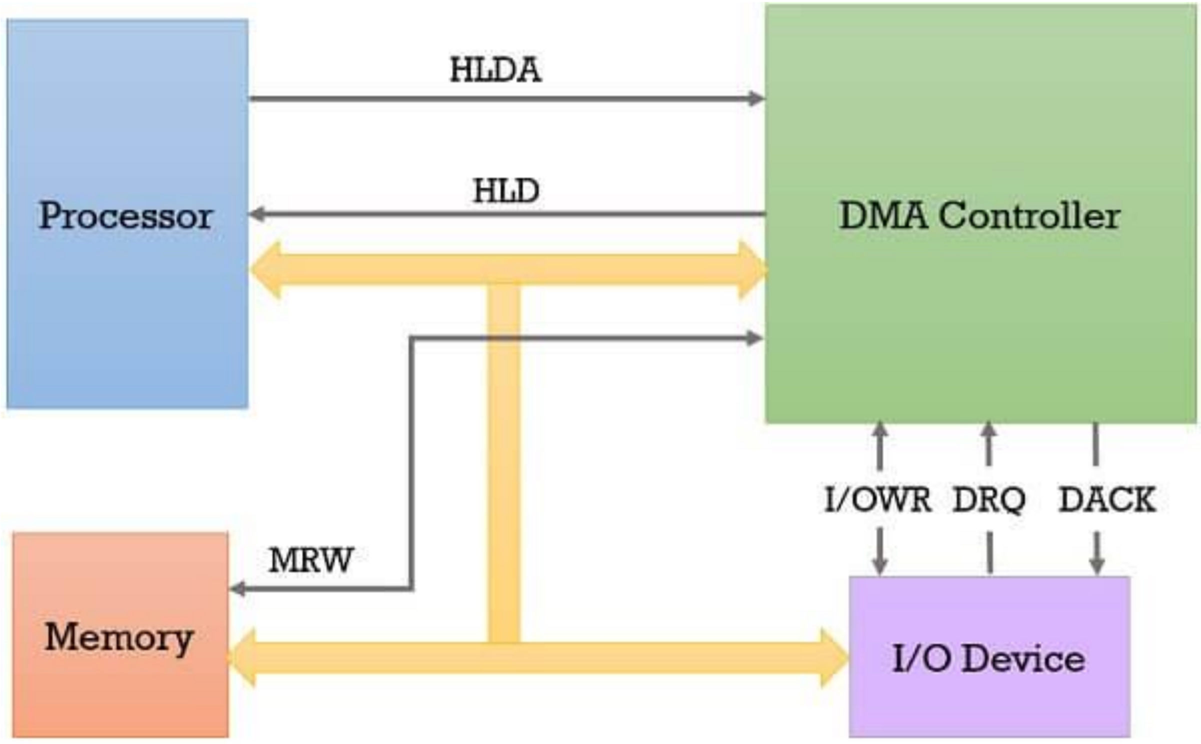
- **The mask register set/reset sets or clears the channel mask.**
  - **if the mask is set, the channel is disabled**
  - **the RESET signal sets all channel mask to disable them**



7 6 5 4 3 2 1 0
Bit Number
Don't Care
00 Select channel 0 mask bit
01 Select channel 1 mask bit
10 Select channel 2 mask bit
11 Select channel 3 mask bit
0 Clear mask bit
1 Set mask bit

DMA data transfer:

DMA controller is a **hardware unit** that allows I/O devices to access memory directly without the participation of the processor. Here, we will discuss the working of the DMA controller. Below we have the diagram of DMA controller that explains its working:



DMA Controller Data Transfer

1. Whenever an I/O device wants to transfer the data to or from memory, it sends the DMA request (**DRQ**) to the DMA controller. DMA controller accepts this DRQ and asks the CPU to hold for a few clock cycles by sending it the Hold request (**HLD**).
2. CPU receives the Hold request (HLD) from DMA controller and relinquishes the bus and sends the Hold acknowledgement (**HLDA**) to DMA controller.
3. After receiving the Hold acknowledgement (HLDA), DMA controller acknowledges I/O device **(DACK)** that the data transfer can be performed and DMA controller takes the charge of the system bus and transfers the data to or from memory.
4. When the data transfer is accomplished, the DMA raise an **interrupt** to let know the processor that the task of data transfer is finished and the processor can take control over the bus again and start processing where it has left.

2016 – 7(b):

[Repeat from 2017 – 7(c)]

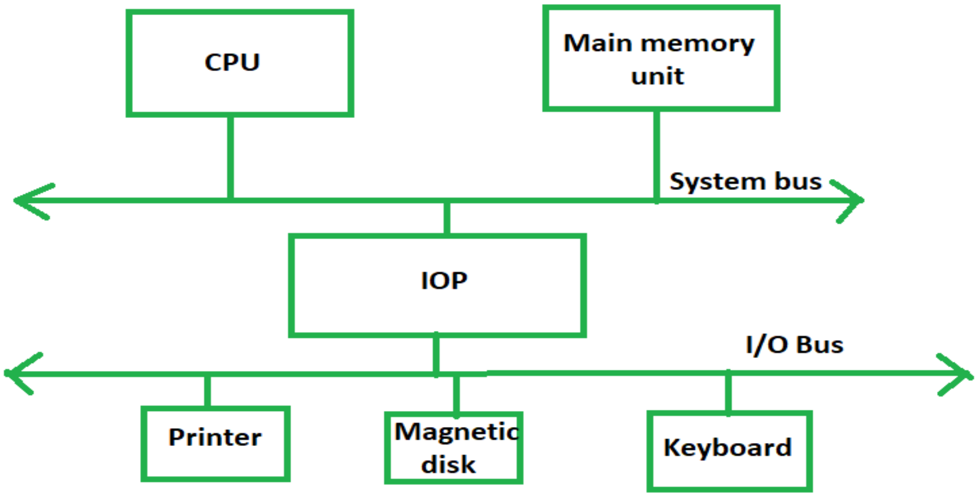
2016 – 7(c):

[Repeat from 2018 – 8(d)]

2016 – 7(d):

[Repeat from 2017 – 6(c)]

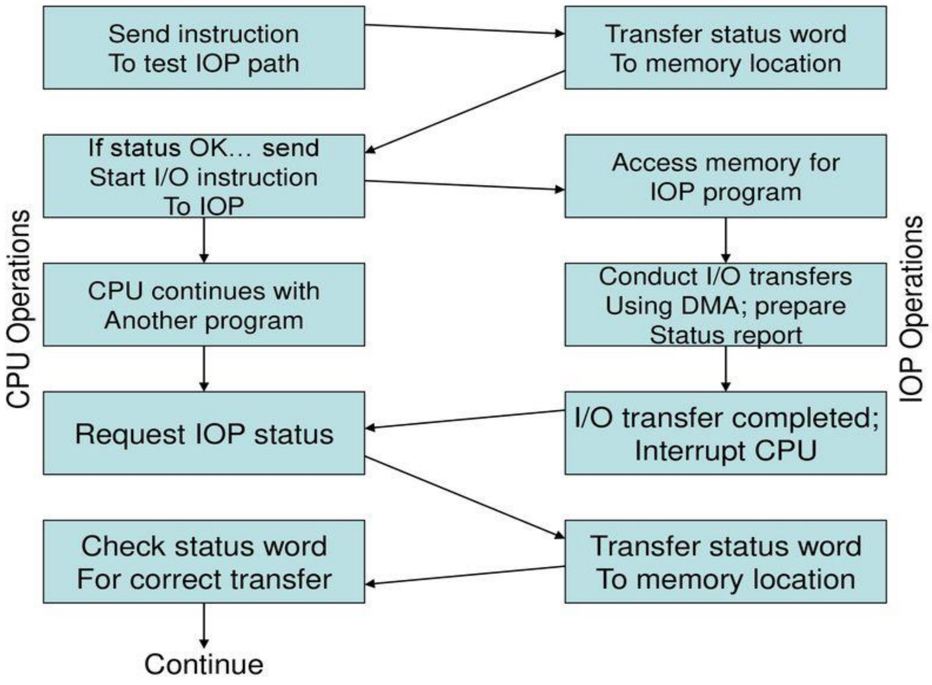
2016 – 8(a):



Input-Output Processor

The Input Output Processor is a specialized processor which loads and stores data into memory along with the execution of I/O instructions. It acts as an interface between system and devices. It involves a sequence of events to executing I/O operations and then store the results into the memory.

CPU-IOP Communication



CPU Operations

Send instruction
To test IOP path

If status OK… send
Start I/O instruction
To IOP

CPU continues with
Another program

Request IOP status

Check status word
For correct transfer

Continue

IOP Operations

Transfer status word
To memory location

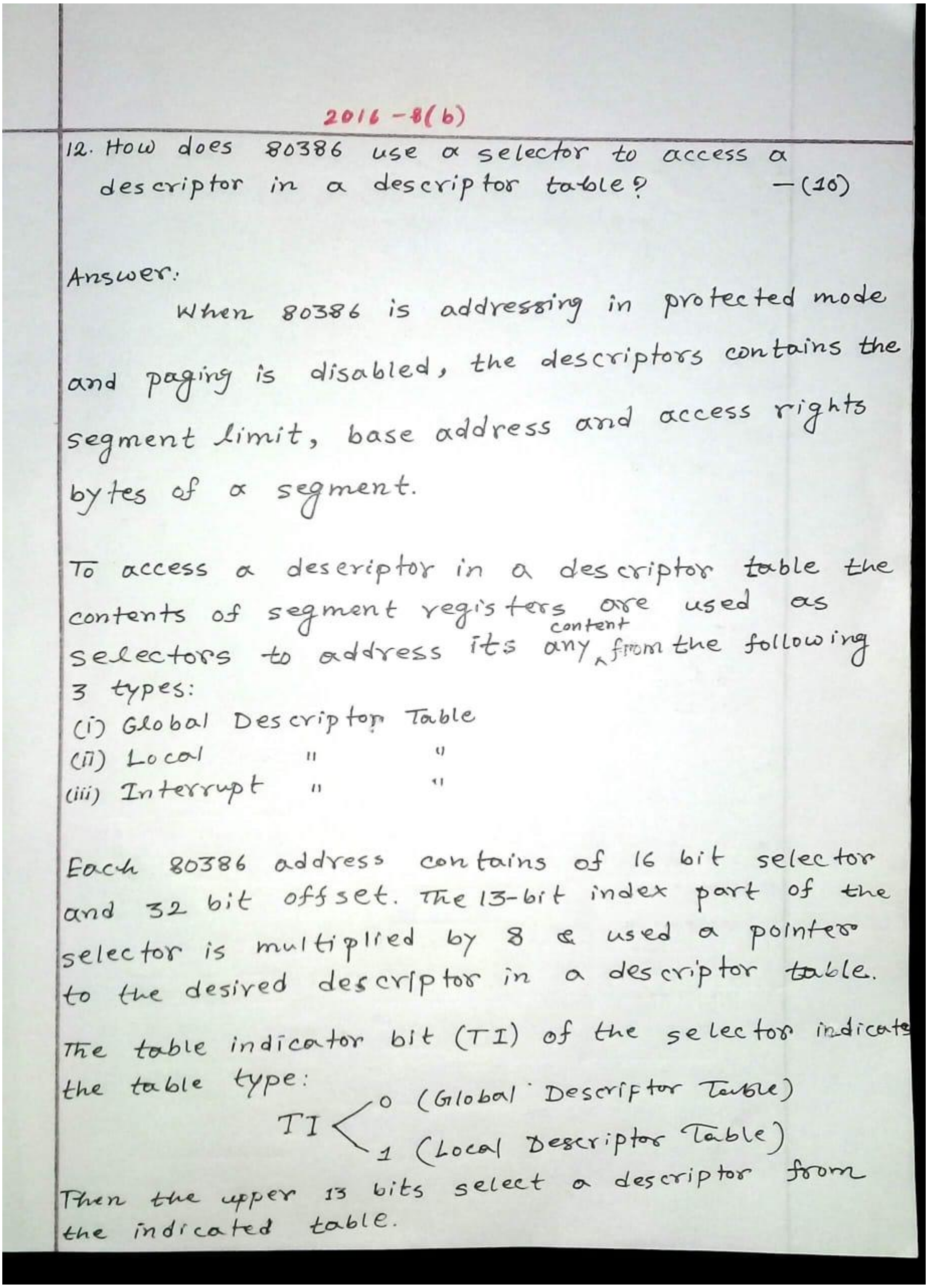
Access memory for
IOP program

Conduct I/O transfers
Using DMA; prepare
Status report

I/O transfer completed;
Interrupt CPU

Transfer status word
To memory location

2016 – 8(b):



2016 – 8(b)

12. How does 80386 use a selector to access a descriptor in a descriptor table? — (10)

Answer:

When 80386 is addressing in protected mode and paging is disabled, the descriptors contains the segment limit, base address and access rights bytes of a segment.

To access a descriptor in a descriptor table the contents of segment registers are used as selectors to address its any content from the following 3 types:

(i) Global Descriptor Table
(ii) Local " "
(iii) Interrupt " "

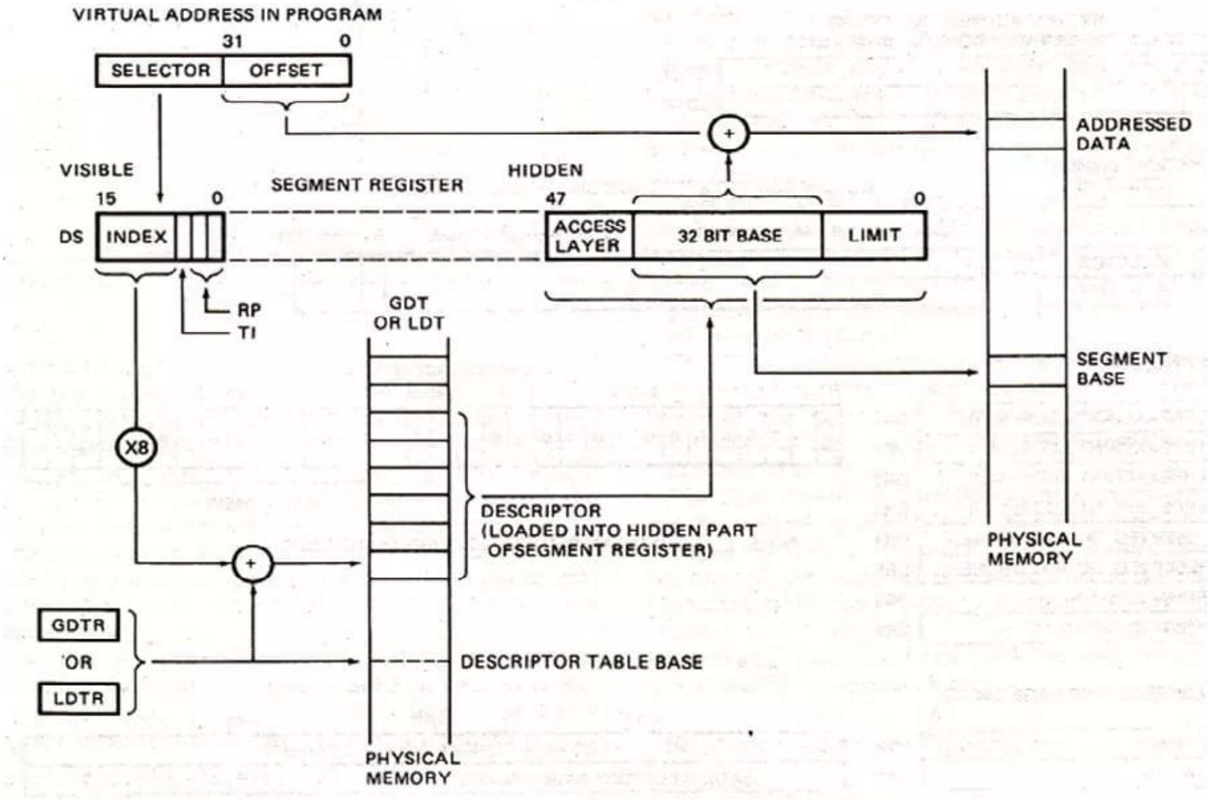
Each 80386 address contains of 16 bit selector and 32 bit offset. The 13-bit index part of the selector is multiplied by 8 & used a pointer to the desired descriptor in a descriptor table.

The table indicator bit (TI) of the selector indicate the table type:

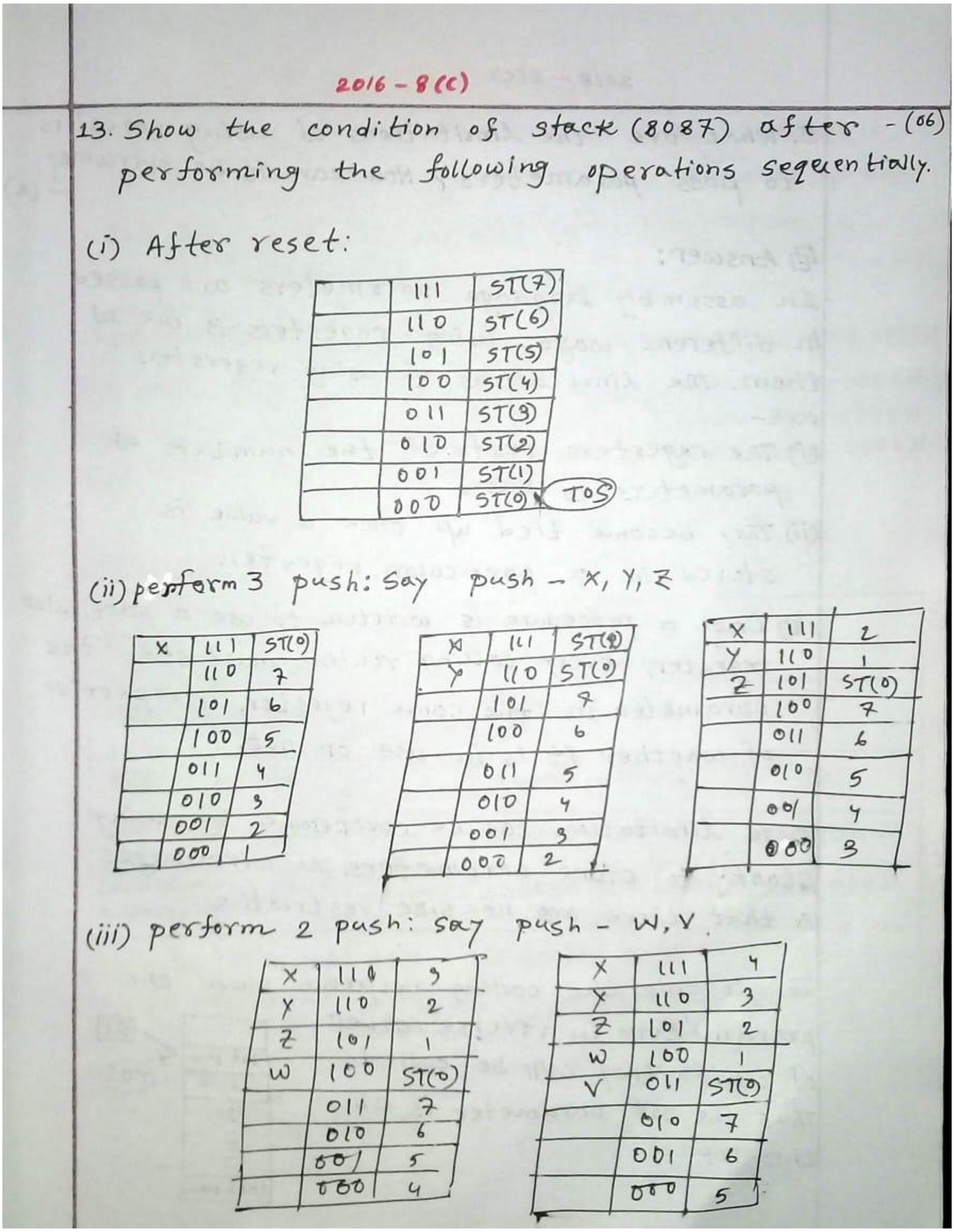
TI < 0 (Global Descriptor Table)
1 (Local Descriptor Table)

Then the upper 13 bits select a descriptor from the indicated table.

2016 - 8(b) – page ii



2016 – 8(c):



2016 – 8(c)

13. Show the condition of stack (8087) after performing the following operations sequentially. - (06)

(i) After reset:

	111	ST(7)
	110	ST(6)
	101	ST(5)
	100	ST(4)
	011	ST(3)
	010	ST(2)
	001	ST(1)
	000	ST(0) ← TOS

(ii) perform 3 push: say push → X, Y, Z

X	111	ST(0)
	110	7
	101	6
	100	5
	011	4
	010	3
	001	2
	000	1

X	111	ST(1)
Y	110	ST(0)
	101	7
	100	6
	011	5
	010	4
	001	3
	000	2

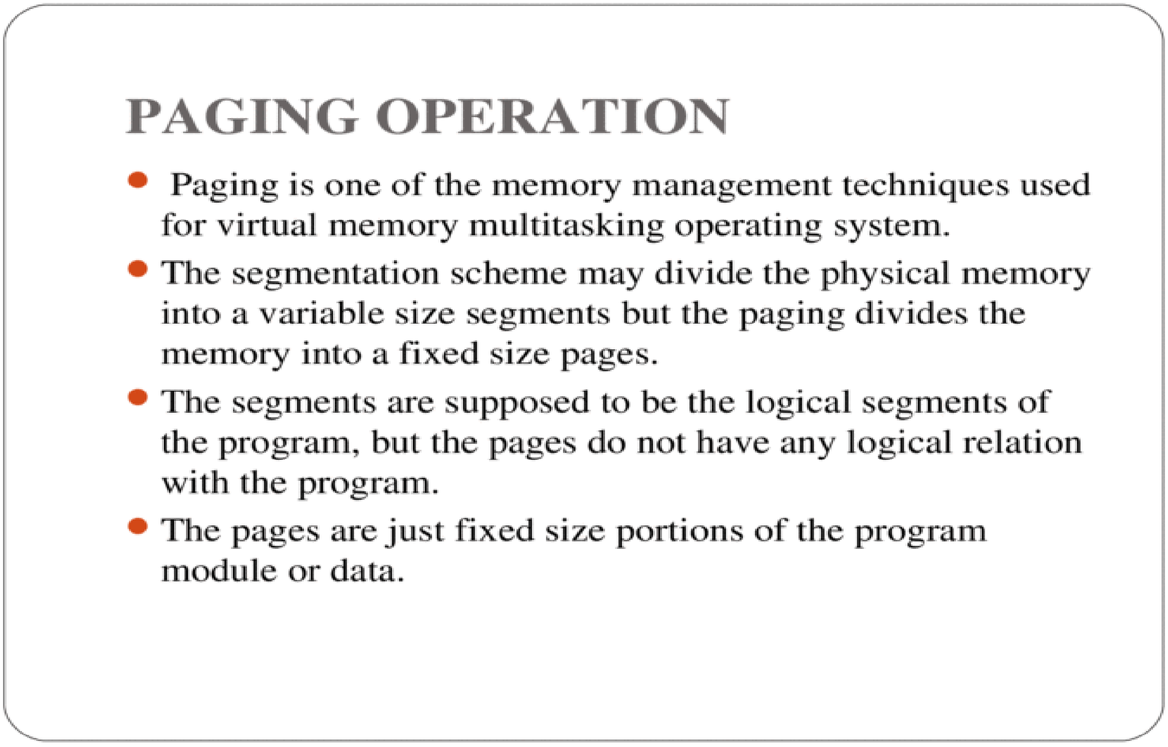
X	111	2
Y	110	1
Z	101	ST(0)
	100	7
	011	6
	010	5
	001	4
	000	3

(iii) perform 2 push: say push - W, V.

X	110	3
Y	110	2
Z	101	1
W	100	ST(0)
	011	7
	010	6
	001	5
	000	4

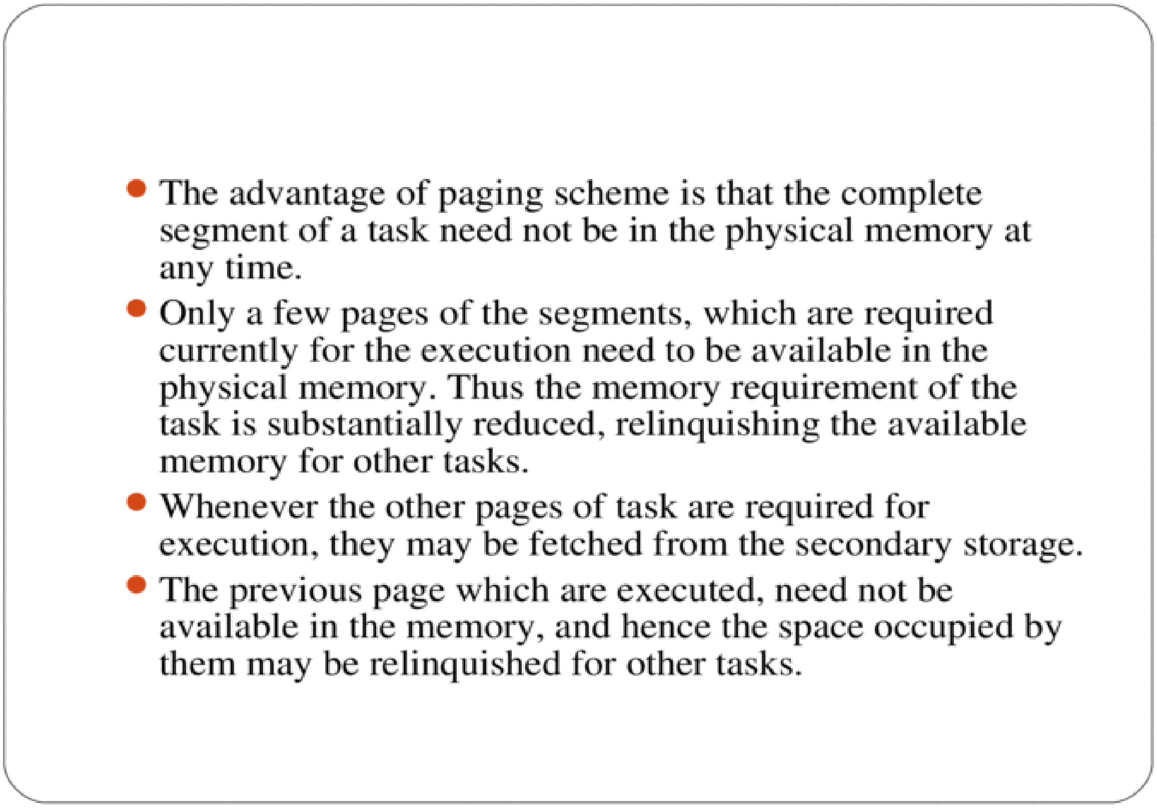
X	111	4
Y	110	3
Z	101	2
W	100	1
V	011	ST(0)
	010	7
	001	6
	000	5

2016 – 8(d):



PAGING OPERATION

- Paging is one of the memory management techniques used for virtual memory multitasking operating system.
- The segmentation scheme may divide the physical memory into a variable size segments but the paging divides the memory into a fixed size pages.
- The segments are supposed to be the logical segments of the program, but the pages do not have any logical relation with the program.
- The pages are just fixed size portions of the program module or data.



- The advantage of paging scheme is that the complete segment of a task need not be in the physical memory at any time.
- Only a few pages of the segments, which are required currently for the execution need to be available in the physical memory. Thus the memory requirement of the task is substantially reduced, relinquishing the available memory for other tasks.
- Whenever the other pages of task are required for execution, they may be fetched from the secondary storage.
- The previous page which are executed, need not be available in the memory, and hence the space occupied by them may be relinquished for other tasks.

- Thus paging mechanism provides an effective technique to manage the physical memory for multitasking systems.
- ***Paging Unit***: The paging unit of 80386 uses a two level table mechanism to convert a linear address provided by segmentation unit into physical addresses.
- The paging unit converts the complete map of a task into pages, each of size 4K. The task is further handled in terms of its page, rather than segments.
- The paging unit handles every task in terms of three components namely page directory, page tables and page itself.