Source: Apke Papa ☺

Date Page

MP - QB.

(1) Explain different types of interrupts? Explain Interrupt vector table for 8086.

Ans:- An interrupt is a condition that halts the microprocessor temporarily to work on a different task and then return to its previous task.

Different types of interrupt present in 8086 µp are:-

→ Hardware interrupts:

Hardware interrupts are those interrupts which are caused by any peripheral device by sending a signal through a specified pin to the microprocessor. There are 2 hardware interrupts in 8086 µp.

* Non-Maskable interrupt: It is a single pin non-maskable interrupt which cannot be disabled. It is the highest priority interrupt in 8086 µp. After its execution, this interrupt generates a Type 2 interrupt. IP is loaded from 00808H and CS is loaded from the word location 0000A H.

* Maskable interrupt (INTR): It provides a single interrupt request and is activated by I/O port. This interrupt can be masked or delayed. It is a level triggered interrupt. It can recieve any interrupt type, so the value of IP and CS will change on the interrupt type recieved.

→ Software Interrupt:- These are instructions that are inserted within the program to generate interrupts. There are 256 software interrupts in 8086 µp. The instructions are of the format

INT type where type ranges from 00_H to FF_H. The starting address ranges from 00000H to 003FFH. These are 2 byte instructions. IP is loaded from type * 04H and CS is loaded from the next address given by (type * 04) + 02H.

IVT table :-

Interrupt type	Content (16 bit)	Address	Comments
Type 0	ISR IP	0000:0000	Reserved for divide by zero interrupt
	ISR CS	0000:0002	
Type 1	ISR IP	0000:0004	Reserved for single step interrupt
	ISR CS	0000:0006	
Type 2	ISR IP	0000:0008	Reserved for NMI
	ISR CS	0000:000A	
Type 3	ISR IP	0000:000C	Reserved for INT single byte instruction.
	ISR CS	0000:000E	
Type 4	ISR IP	0000:0010	Reserved for INTO Instruction.
	ISR CS	0000:0012	
		0000:0014	
		0000:0016	
Type N	ISR IP	0000:004N	Reserved for two byte instruction INT Type
	ISR CS	0000:(004N+2)	
		0000:03FC	
Type FFH	ISR IP	0000:03FE	
	ISR CS	0000:03FF	

ISR: Interrupt Service Routine.

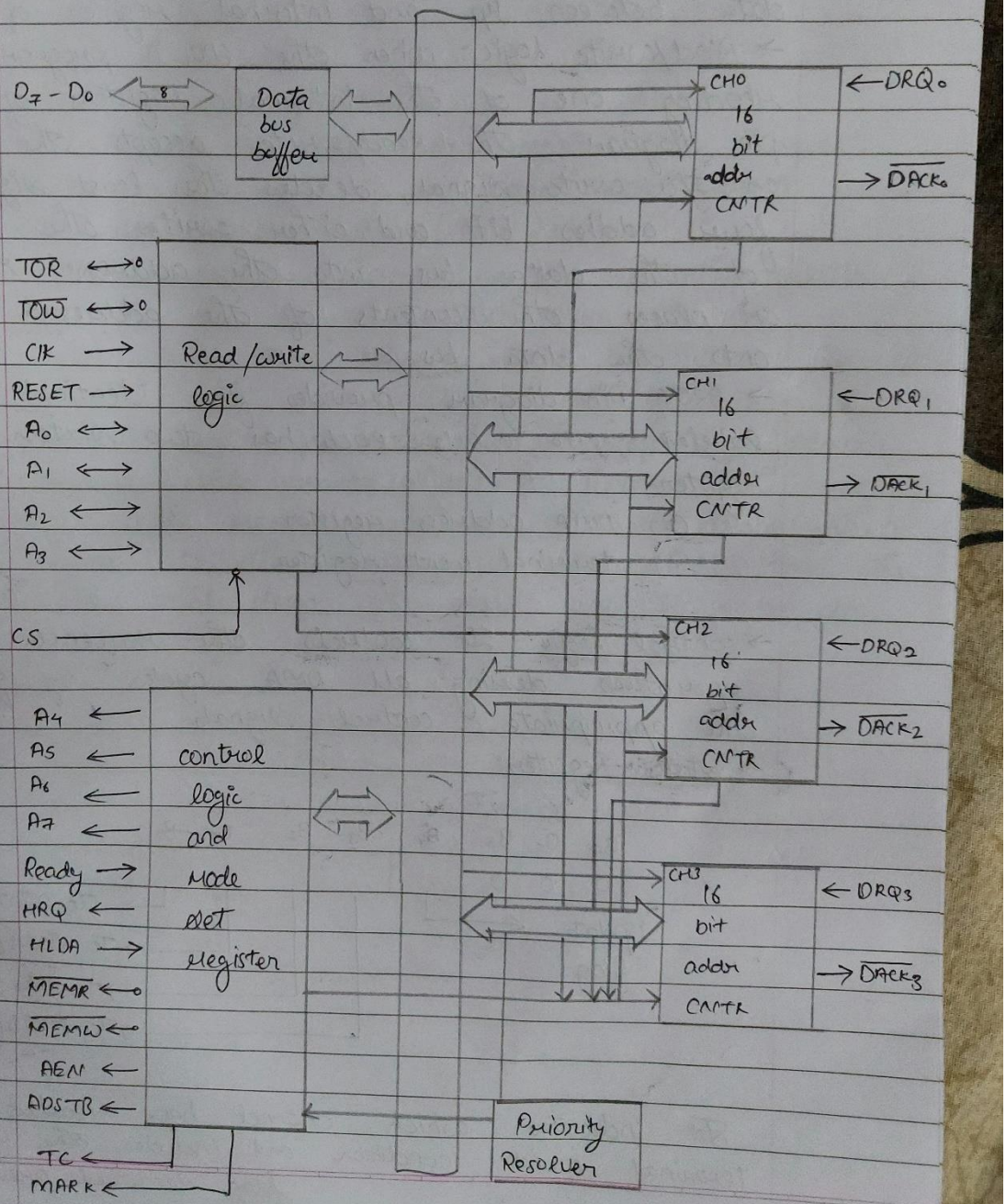
→ 8086 supports a total of 256 types of interrupt i.e. from 00H to FFH. Each interrupt requires 4 bytes, i.e two bytes each for IP and CS of its ISR. Thus a total of 1024 are required for 256 interrupt types. Hence, Interrupt vector table start at location 0000:0000 and ends at 0000:03FFH.

→ The interrupt type N is multiplied by 4 and the hexadecimal multiplication obtained gives the offset address in the zeroth code segment at which the IP and CS addresses of the interrupt service routine. (ISR) are stored.

→ The execution automatically from new CS: IP.

(Q) Draw and explain block diagram of 8257 ? How DMA operations are performed.?

Ans:



→ Data bus Buffer: It is a tri-state, bi-directional, eight bit buffer which interfaces the 8257 to the system data bus. It is used to transfer data between µp and internal register of 8257.

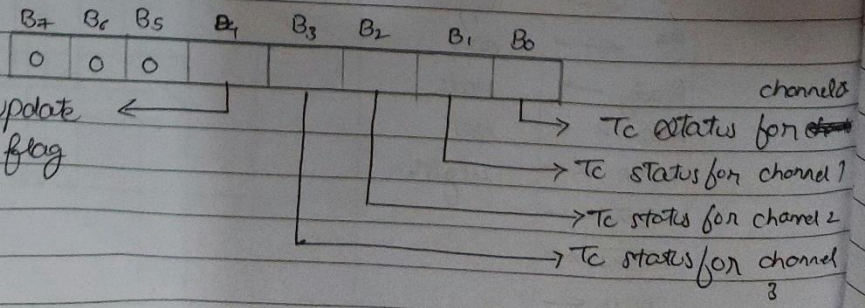
→ Read/write logic: when the CPU is programming or reading one of the internal registers of 8257 pin diagram, the Read/write accepts the I/O Read or I/O write signal, decodes the least significant four addies bit and either writes the contents of the data bus into the addressed register or places the contents of the addressed register onto the data bus.

→ 8257 Pin diagram provides four identical channels, labeled CH_0 to CH_3, each has two sixteen bit register.

(1) DMA address register
(2) terminal count register.

→ control logic: It controls the sequence of operations during all DMA cycles by generating the appropriate control signals.

→ Status Register:



It indicates which channel have reached terminal count condition and includes the update flag described previously.

→ Priority Resolver: It resolves the peripherals request. It can be programmed to work in two modes either in fixed mode or rotating priority mode

DMA operation performed:

→ The 8086 microprocessor receives bus requests through its HOLD pin and issues grants from the hold acknowledge (HLDA) pin.

→ A request is made when a potential master sends a 1 to the HOLD pin.

→ Normally, after the current bus cycle is complete the 8086 will respond by putting a 1 on the HLDA pin. When the requesting device receives this grant signal it becomes the master.

→ It will remain master until it drops the signal to the HOLD pin, at which time the 8086 will drop the grant on the HLDA pin.

(3) Explain what is branch prediction logic in Pentium? Explain working of Branch prediction with suitable example?

Ans:- → why do we need branch prediction?

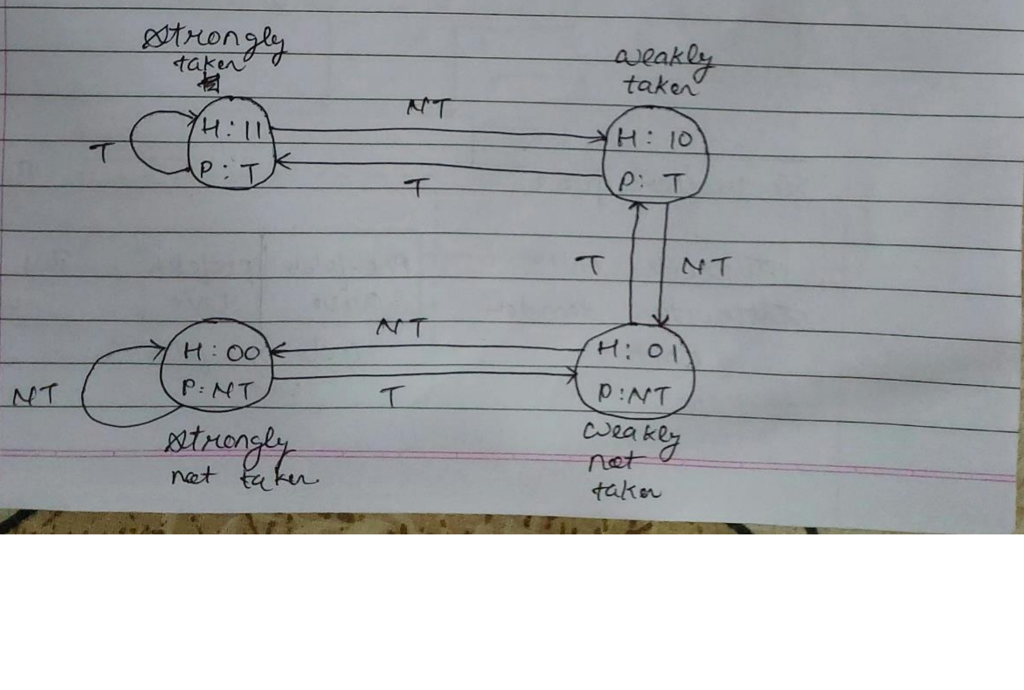
1. The gain produced by Pipelining can be reduced by the presence of program transfer instruction.
2. They change the sequence causing all the instruction that entered the pipeline after program transfer instructions invalid.
3. Thus no work is done as the pipeline stages are reloaded.

To avoid this problem, Pentium uses a scheme called Dynamic branch prediction. In this scheme, a prediction is made for branch instructions currently in the pipeline. The prediction will either be taken or not taken. If the prediction is true then the pipeline will not be flushed and no clock cycles will be lost. If the prediction is false then the pipeline will be flushed and starts over with the current instruction.

working of Branch prediction:

→ BTB is a lookaside cache that sits to the side of Decode instruction (DI) stage of two pipelines and monitors for branch instruction.

→ The first time the branch instruction enters the pipeline, the BTB uses its source memory to perform a lookup in the cache.
→ Since the instruction was never seen before, it is BTB miss. It predicts that the branch will not be taken even though it is unconditional jump instruction.
→ when the instruction reaches the EU, the branch will be either taken or not taken. If taken, the next instruction to be executed will be fetched from the branch target address. If not taken, there will be ~~sequet~~ sequential fetch of instruction.
→ when the branch is taken for the first time, the execution unit provides feedback to the branch prediction. The branch target address ~~target address~~ is sent back which is recorded in BTB.
→ A directory entry is made containing the source memory address and history bit is set as strongly taken.

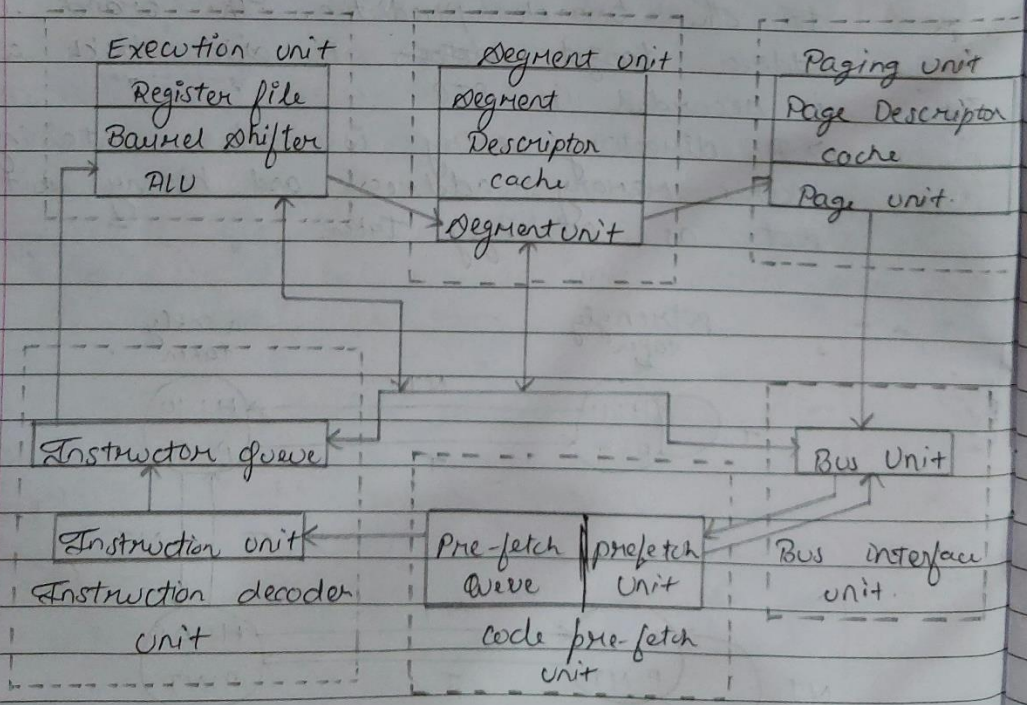


Q4) Compare the 8086, 80386, Pentium Processor.

Product	8086	80286	80386	80486	Pentium
Year introduced	1978	1982	1985	1989	1992
Technology	NMOS	NMOS	CMOS	CMOS	BICMOS
Clock rate (MHz)	3 - 10	10 - 16	16 - 33	25 - 33	60, 66
Number of pins	40	68	132	168	273
Number of transistors	29,000	130,000	275,000	1.2 million	3.1 million
Physical memory	1M	16M	4G	4G	4G
Virtual memory	None	1G	64T	64T	64T
Internal data bus	16	16	32	32	32
External data bus	16	16	32	32	64
Address bus	20	24	32	32	32
Data type (bits)	8, 16	8, 16	8, 16, 32	8, 16, 32	8, 16, 32

(5) Draw and explain the internal architecture of 80386 Microprocessor?

Ans: → The internal architecture of the 80386 included 6 functional unit that operate in parallel. The parallel operation is called pipelining processing.
~~Also~~, the six functional unit of the 80386 Architecture are:
→ Bus interface unit
→ code pre-fetch unit
→ Instruction decoder unit
→ Execution unit
→ Segmentation unit
→ Paging unit.



- The Bus Interface Unit connects the 80386 with memory and I/O. Based on internal requests for fetching instructions and transferring data from the code pre-fetch unit, the 80386 Architecture generates the address, data and control signals for the current bus cycles.
- Also, The code pre-fetch unit pre-fetches instructions when-the bus interface unit is not executing the bus cycles. It then stores them in a 16-byte instruction queue for decoding by the instruction decode unit.
- Moreover, The instruction decode unit translates instructions from the pre-fetch queue into microcode. The decoded instructions then stored in an instruction queue (FIFO) for processing by the execution unit.
- The execution unit processes the instructions from die instruction queue. It contains a control unit a data unit and a protection test unit.
- The control unit contains microcode and parallel hardware for fast multiply, divide, and effective address calculation. The unit includes a 32-bit ALU, 8 general purpose registers and a 64-bit barrel shifter for performing multiple bit shifts in one clock. The data unit carries out data operations requested by the control unit.
- Moreover, The protection test unit checks for segmentation violations under the control of microcode.
- Also, The segmentation unit calculates and translates the logical address into linear addresses at the request of the execution unit.
- The translated linear address sent to the paging unit. Upon enabling the paging mechanism, the 80386 translates these linear addresses into Physical addresses.

- Also if paging not enabled, the physical address is identical to the linear address and no translation is necessary.

2.5) Explain the operating modes of 80386?

=> The operating mode of the 80386 has three processing modes.

1) Real-Address Mode

→ i) Power on default mode

ii) In real address mode, 80386 acts as a version OF 8086 which is faster than original 8086 as 80386 operates on higher clock frequency.

iii) It offers memory addressability OF 1 MB OF physical memory

iv) segmented addressing: Total 6 segments

2) Protected Mode

i) It is the natural 32-bit environment of the 80386 Processor.

ii) In this mode all instructions and features are available.

iii) Segment size = 1 to 4GB

iv) IF 80386 is working in protected Mode it can not switch to real mode unless it is reset.

3) Virtual 8086 Mode

i) ~~V86 mode~~ is Virtual 8086 Mode also called as V86 Mode. It is a dynamic in the sense that the processor can switch repeatedly and rapidly b/w V86 mode and protected mode

Date

ii) The CPU enters V86 mode from protected mode to execute an 8086 program, then leaves V86 mode and enters protected mode to continue executing a native 80386 program.

iii) The features that are available to applications programs in protected mode and to all programs in V86 mode are the same.

(7) Explain internal architecture of 8086 µp ? Differentiate the funtioning of minimum mode and maximum mode?

Ans:- → The architecture of 8086 provides a number of improvements over 8085 µp architecture.
→ It supports 16-bit ALU, a set of 16-bit registers, a rich instruction set, powerful interrupt structure, fetched instruction queue.
→ The 8086 architecture is divided into two parts (a) Bus Interface Unit (BIU) and (b) Execution unit.

(a) Bus Interface unit (BIU):
It provides the interface of 8086 to external memory and I/O devices via the system Bus. It performs various machine cycles such as memory read, I/O read, etc. to transfer data between memory and I/O devices.

BIU mainly contains 4 segment registers, instruction pointer, prefetch queue and an address generation circuit.
The four segment registers are,
code segment (CS)
Data segment (DS)
stack segment (SS)
Extra segment (ES)

(b) Execution Unit (EU): The main components of EU are General purpose register, the ALU, special purpose register, Instruction register, and decoder, and flag registers.

There are 16 general purpose registers:

→ AX register.

It holds operands and result during multiplication and division operation. Also an accumulator during string operation.

→ BX register.

Holds memory address in indirect addressing mode.

→ CX register

It holds count for instruction like loop, rotate, shift and string operation.

→ DX register.

It is used with AX to hold 32 bit value during multiplication and division.

Special purpose registers.

→ Stack pointer

→ Base pointer.

→ Source index

→ Destination index.

9 flags registers. Flag register consist of 9 flags.

6 status flags:

carry flag (CY)
Parity flag (PF)
Auxiliary carry flag (AF)
Zero flag (ZF)
Sign flag (S)
Overflow flag (O)

3 control flags.

Trap flag (TF)
interrupt flag (IF)
Direction flag (DF).

(8) write an assembly language program to find the largest number from an ~~array~~ unordered array of 8-bit numbers?

Sol: example:

	no. of element in array	elements of array			
Input data →	04	20	1A	55	2B
Memory address →	2050	2051	2052	2053	2054

output data →	55	← largest element
Memory address →	3050	

Program:

Memory address	Mnemonics	comment
2000	LXI H, 2050	H ← 20, L ← 50
2003	MOV C, M	C ← M
2004	DCR C	C ← C-01
2005	INX H	HL ← HL+0001
2006	MOV A, M	A ← M
2007	INX H	HL ← HL+0001
2008	CMP M	A-M
2009	JNC 2000	~~C ← C-1~~ If carry flag = 0, go to 2000
200C	MOV A, M	A ← M
200D	DCR C	C ← C-1
200E	JNZ 2007	If zero flag = 0, goto 2007
2011	STA 3050	A → 3050
2014	HLT	

Q9)

Q. 2: Interface 32 K word of memory to the 8086 microprocessor system . Available memory chips are 16 K x 8 RAM. Use suitable decoder for generating chip select logic.

Step_1: Total memory = 32 K word = 32*2 K = 64 K

IC available = 16 K

hence,

number of RAM IC required = 64 K x 8/ 16 Kx8 = 4 ICs

so,

EVEV Bank = 2 ICs of 16 Kx8 RAM

ODD Bank = 2 ICs of 16 Kx8 RAM

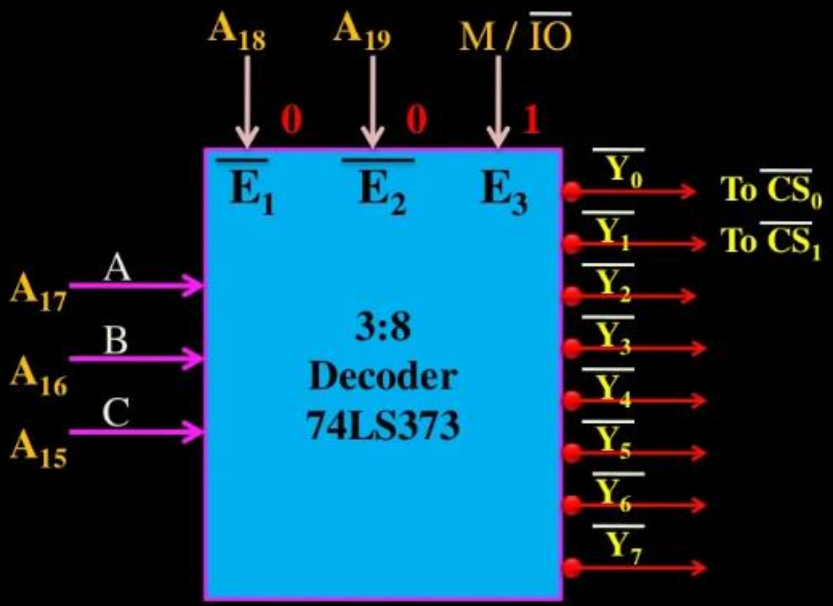
Even bank	Odd bank
RAM _1 (16K)	RAM _2 (16K)
RAM_ 3 (16K)	RAM _4 (16K)

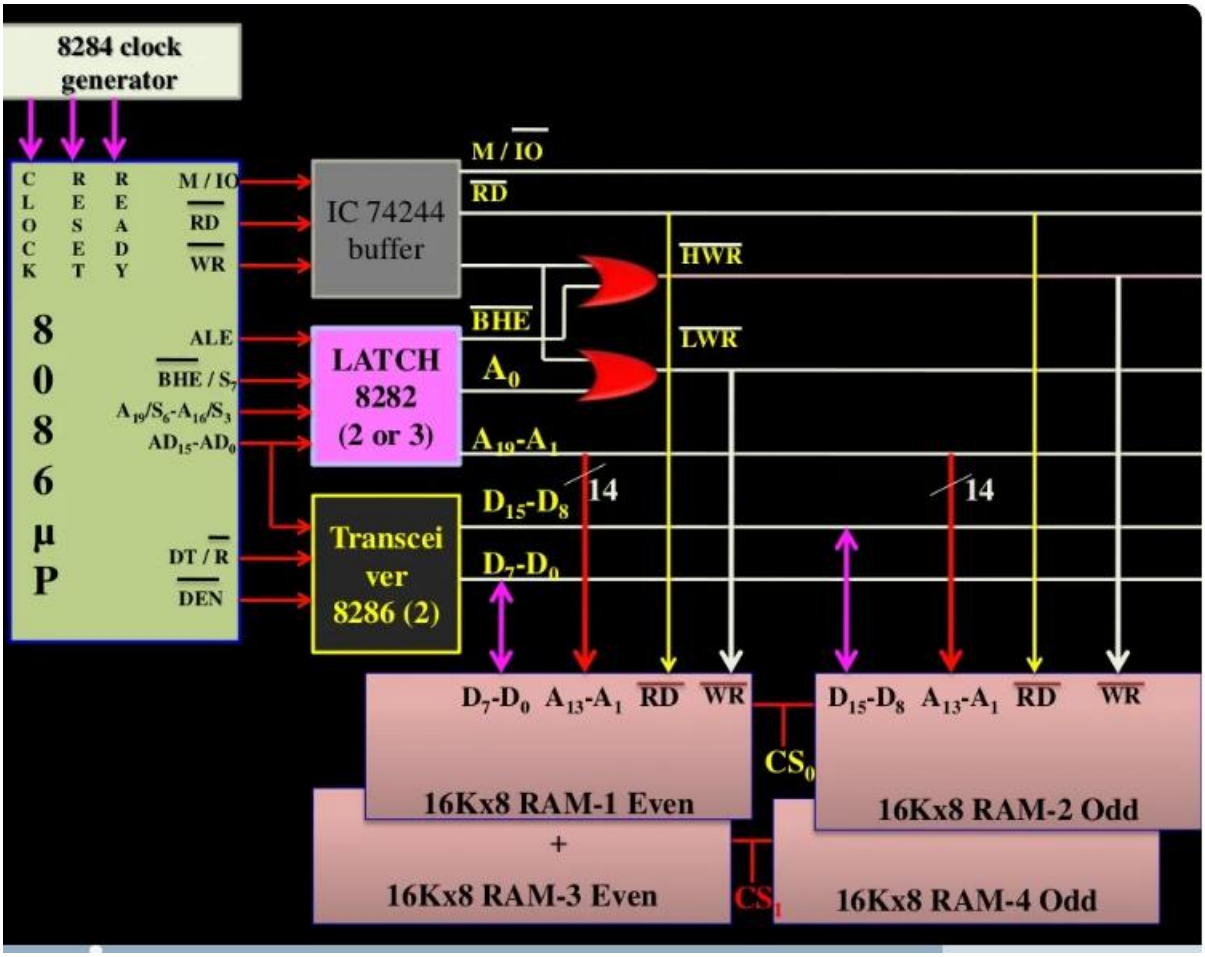
Step_2: Number of address lines required = 15 address lines

Step_3: Address decoding table

MEMORY IC	HEX ADDRESS	BINARY ADDRESS																			
		A_{19}	A_{18}	A_{17}	A_{16}	A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
16 K x 8 RAM-(1)	OOOOO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	07FFE	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
16 K x 8 RAM-(3)	8000	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0FFFE	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

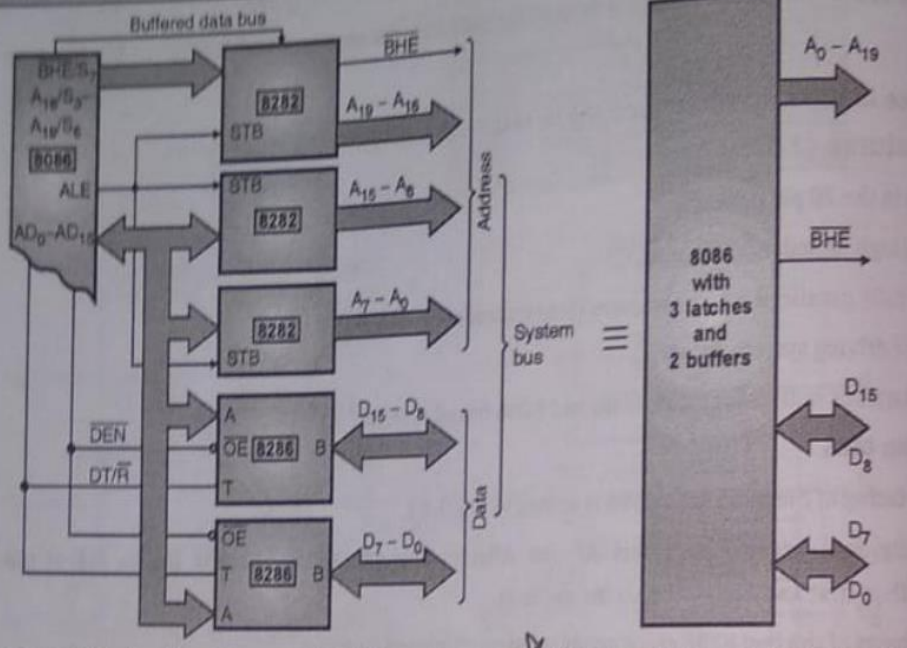
Step_3: Generation of chip select logic




8284 clock generator
CLOCK
RESET
READY
M / IO
RD
WR
ALE
BHE / S7
A19/S6-A16/S3
AD15-AD0
DT / R
DEN
8086 µP
IC 74244 buffer
LATCH 8282 (2 or 3)
Transceiver 8286 (2)
M / IO
RD
HWR
BHE
LWR
A0
A19-A1
D15-D8
14
D7-D0
D7-D0 A13-A1 RD WR
D15-D8 A13-A1 RD WR
CS0
CS1
16Kx8 RAM-1 Even
+
16Kx8 RAM-3 Even
16Kx8 RAM-2 Odd
16Kx8 RAM-4 Odd

Q10)

Q. Draw and explain demultiplexing of address bus in 8086.



(a) Demultiplexing the address and data buses (b) Simplified equivalent circuit

Fig. 3.4.3

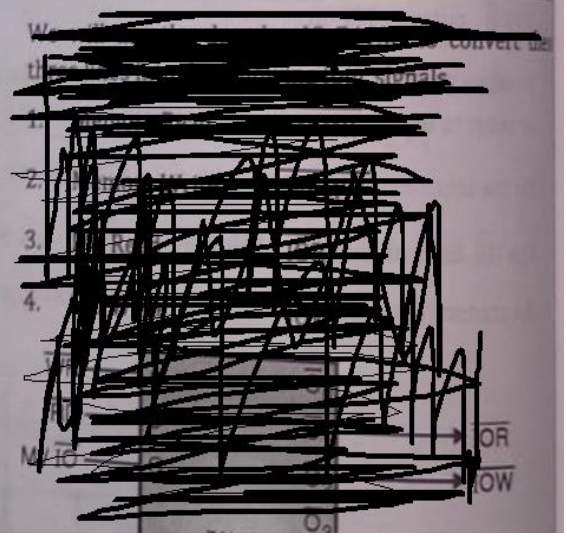
We have discussed the use of 8282 latches as address latching and 8286 as data bus buffer.

- If we use them simultaneously then we can demultiplex the address data bus as shown in Fig. 3.4.3(a).

Thus we require three octal latch chips (8282) and two octal transceiver chips (8286) in order to completely demultiplex the address and data bus.

The basic diagram of Fig. 3.4.3 is often required in the memory interfacing and I/O interfacing.

The simplified equivalent of Fig. 3.4.3(a) is shown in Fig. 3.4.3(b).



(11) Discuss need for memory banking in 8086.

Ans: → The 8086 processor provides a 16-bit data bus. So it is capable of transferring 16 bit in one cycle but each memory location is only of a byte, therefore we need two cycles to acess 16 bits from two different memory locations. The solution to this problem is memory banking.

→ The memory chip is equally divided into two parts (banks). One of the banks contains even addresses called Even bank and other contains odd addresses called odd bank. Even bank always gives lower byte so even bank is called lower bank and ODD bank is called Higher bank.

→ This banking scheme allows to access two aligned memory location from both banks simultaneously and process 16-bit data transfer. memory banking doesn't make it compulsory to transfer 16-bit, it facilitates the 16-bit data transfer.

→ The choice between 8-bit and 16-bit transfer depends on the instructions given by the programmers.

(12) Explain Mode 0 and Mode 2 of 8255.

Sol:

Mode-0:

→ This mode is also known as basic I/O mode.
→ This mode provides simple input and output capability using each of the three port
→ Data can be simply read from and written to input and output ports respectively, after appropriate initialisation.

Salient features of this mode.

(i) Two 8-bit ports (port A and port B) and two 4-bit ports (port c upper and lower) are available
(ii) Any port can be used as an input or output port.
(iii) Output ports are latched. Input ports are not latched
(iv) A maximum of four ports are available so that overall 16 I/O configurations are possible

Mode-2: → This mode of operation of 8255 is also known as strobed bidirectional I/O.
→ This mode of operation provides 8255 with an additional feature for communicating with a peripheral device on an 8-bit data bus.
→ Handshaking signals are provided to maintain proper data flow and synchronization between data transmitter and reciever.

→ The interrupt generation and other functions are similar to mode 1.
→ Thus, in this mode, 8255 is a bidirectional 8-bit port with handshake signals.
→ The $\overline{RD}$ and $\overline{WR}$ signals decide whether the 8255 is going to operate as an input port or output port.

Salient features:
→ The single 8-bit port in group A is available.
→ The 8-bit port is bidirectional and additionally a 5-bit control port is available.
→ Three I/O lines are available at port C, viz $PC_2 - PC_0$.
→ Inputs and outputs are both latched.

(13) Explain interrupt procedure of 8086.

Ans:- An interrupt is a condition that halts the microprocessor temporarily to work on a different task and then return to its previous task

In 8086 following tasks are performed when up encounters an interrupt.

→ The value of flag register is pushed into the stack. It means that first the value of SP (stack pointer) is decremented by 2 then the value of the flag register is pushed to the memory address of stack segment.

→ The value of starting memory address of CS (code segment) is pushed into the stack.
→ The value of IP (Instruction pointer) is pushed into the stack.
→ IP is loaded from word location (interrupt type) * 04.
→ CS is loaded from next word location.
→ Interrupt and trap flag are reset to 0.

There are two types of interrupt
(1) Hardware interrupt
   (a) Non-Maskable interrupt
   (b) Maskable interrupt
(2) Software interrupt.

(4) Explain integer pipeline of Pentium.

Ans: Pentium processor uses a five stages pipeline. The stages are as follows.

(i) Pre-fetch stage: The instructions are of variable length and stored in a prefetch buffer. A 256 bit path exist from the instruction cache to the prefetch buffer.

(ii) Decode 1 stage: The processor decodes the instruction. It finds the opcode and addressing information. It checks which information can be paired for simultaneous execution. It participates in branch address prediction.

(iii) Decode 2 stage: In this stage it finds the addresses for memory refrence.

(iv) Execute stage: Data cache fetch or ALU or FPU operation is carried out. Two operations can be carried out at this stage.

(v) write back stage: In this stage the registers and flags are updated on the basis of the results of execution.

(15) Write a note on Hyperthreading.

Ans:- → Hyperthreading used the concept of simultaneous multithreading and shows an improvement in the Intel microarchitecture development.

→ The major elegance of this architecture lies in devising appropriate resource sharing policy for each shared resource. Several resource sharing strategies have been investigated by the developers. Some of these are (a) partitioned resources (b) threshold sharing, and (c) full sharing. The choice of sharing strategy to be adopted depends on several factors, such as, the traffic pattern, size of the resource, potential deadlock probabilities and other considerations.

→ To do this, there is one copy of the architecture state for each logical processor, and the logical processor share a single set of physical execution resources. From a software or architecture perspective, this means operating systems and user program can schedule processes or threads to logical processors as they would on conventional physical processor in a multi-processor system. From a microarchitecture perspective, this means that instructions from logical processors will persist and execute simultaneously on shared execution resources.

(Q.16)

Program 3.5

A program to find out the number of even and odd numbers from a given series of 16-bit hexadecimal numbers.

Solution The simplest logic to decide whether a binary number is even or odd, is to check the least significant bit of the number. If the bit is zero, the number is even, otherwise it is odd. Check the LSB by rotating the number through carry flag, and increment even or odd number counter.

```
ASSUME CS:CODE, DS:DATA
DATA SEGMENT
LIST DW 2357H, 0A579H, 0C322H, 0C91EH, 0C000H, 0957H
COUNT EQU 006H
DATA ENDS
CODE SEGMENT
START:      XOR BX, BX
            XOR DX, DX
            MOV AX, DATA
            MOV DS, AX
            MOV CL, COUNT
            MOV SI, OFFSET LIST
AGAIN:      MOV AX, [SI]
            ROR AX, 01
            JC ODD
            INC BX
            JMP NEXT
ODD:        INC DX
NEXT:       ADD SI, 02
            DEC CL
            JNZ AGAIN
            MOV AH, 4CH
            INT 21H
            CODE ENDS
            END START
```

Program 3.5 *Listings*

Q18) Explain protection mechanism of 80386 with diagram.

2. To access any memory location 2 registers (Base and Offset) need to be initialized.

12.6 Protection Mode

This mode is mainly meant for multi tasking operations. Multiple tasks running simultaneously using separate code, data and stack segment.

Besides it also takes care of proper authentication of a task to access a particular segment.

12.6.1 Protection Mechanism

University Questions

Q. Explain the protection mechanism in x86 DX via call gates, priviledge levels. **MU - Dec. 12, 10 Marks**

Q. Explain the protection mechanism of x86 intel family microprocessor. **MU - Dec. 13, 10 Marks**

Q. Write short notes on : CALL gate mechanism **MU - Dec. 13, 10 Marks**

Q. Explain protection mechanism used in 80386. **MU - May 15, 10 Marks**

Q. Explain in brief protection mechanism in 80386DX processor. **MU - Dec. 15, May 16, 5 Marks**

1. Most processors have only two protection levels (user and supervisor), but x86 architecture features four levels of protection, called **Privilege Levels (PL).**
2. They are designed to support the needs of multitasking OS to isolate and protect user programs from each other and the OS from unauthorized access.

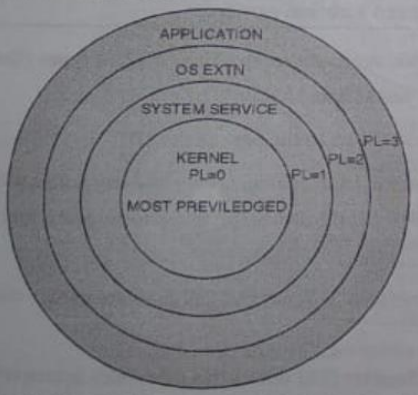


Fig. 12.6.1 : Privilege levels of the tasks in x86 architecture

3. The privilege levels control the use of privileged instructions, I/O instructions, and access to segments and segment descriptors. The x86 architecture offers an additional type of protection on a page basis, when paging is enabled.

   Requested PL ⇒ RPL
   Descriptor PL ⇒ DPL
   Current PL ⇒ CPL
   Effective PL ⇒ EPL
   EPL = max (RPL, CPL)

4. The PLs are numbered 0, 1, 2, and 3. Level 0 is the most privileged level. Level 3 is the least privileged. As shown in Fig. 12.6.1, level 3 is used for user application, level 2 is used for OS extensions, level 1 for system services, and the most privileged level 0 is used for kernel.
5. The x86 architecture controls access to both data and code between levels of task, according to the following rules of privilege:
   (a) Data stored in a segment with PL = p can be accessed only by code executing at a PL, numerically, at least as privileged as p.
   (b) A code segment (a procedure) with PL = p can be called only by a task executing at, numerically, the same or lower PL than p.
   (c) A stack segment with PL = p can be used only by a task executing at the same PL.
6. The following PLs are used to maintain privilege level check :
   (a) Requestor PL, RPL, the PL of the original task that supplies the selector. RPL is determined by the two LSBs of the selector.
   (b) Descriptor PL, DPL, the PL (according to the above rules) at which a task may access that descriptor and the segment associated with that descriptor. Bits 6 and 5 of the access rights byte (ARB) of a descriptor determine the DPL.
   (c) Current PL, CPL, the PL at which a task is currently executing, i.e. at which the code segment is being executed. CPL is stored in the processor, but not accessible to the programmer.
   (d) Effective PL, EPL, the least privileged of the RPL and the CPL. Since smaller PL values indicate greater privilege, EPL is the numerical maximum of RPL and CPL, i.e. EPL = max (RPL, CPL). EPL is not stored anywhere, but is *immediately copied* into CPL.

11. Accessing different types of programs using gates is shown in Fig. 12.6.2.
12. Gate descriptors follow all the access rules of privilege; i.e., gates can be accessed by a task if its EPL is equal to, or more privileged than the gate descriptor's DPL.
13. A task can access the code of higher PL using a gate (task gate, call gate, trap gate or interrupt gate)
12. A task can access data of lower or equal PL.
15. A task can access stack of same PL only.
16. The PL of the task (CPL) is compared with the PL of code /data /stack (DPL) for the above decisions.

(19) Explain memory segmentation in 8086 with neat diagram.

Ans: Segmentation is the process in which the main memory of the computer is logically divided into different segments and each segment has its own base address. It is basically used to enhance the speed of execution of the computer system, so that the processor is able to fetch and execute the data from the memory easily and fast.

The Bus interface unit (BIU) contains four 16-bit special purpose registers called as segment Registers.

→ Code segment Register (CS): is used for addressing memory location in code segment of the memory, where the executable program is stored.

→ Data segment Register (DS): points to the data segment of the memory where the data is stored.

→ Extra - segment Register (ES): also refers to the segment in the memory which is another data segment in the memory.

→ Stack segment Register (SS): it is used for addressing stack segment of the memory.

diagram:

Q20) Draw timing diagram of memory read operation in minimum mode.

k, as shown in Fig. 1.14 (c). The other conditions
on for the HOLD and HLDA signals.

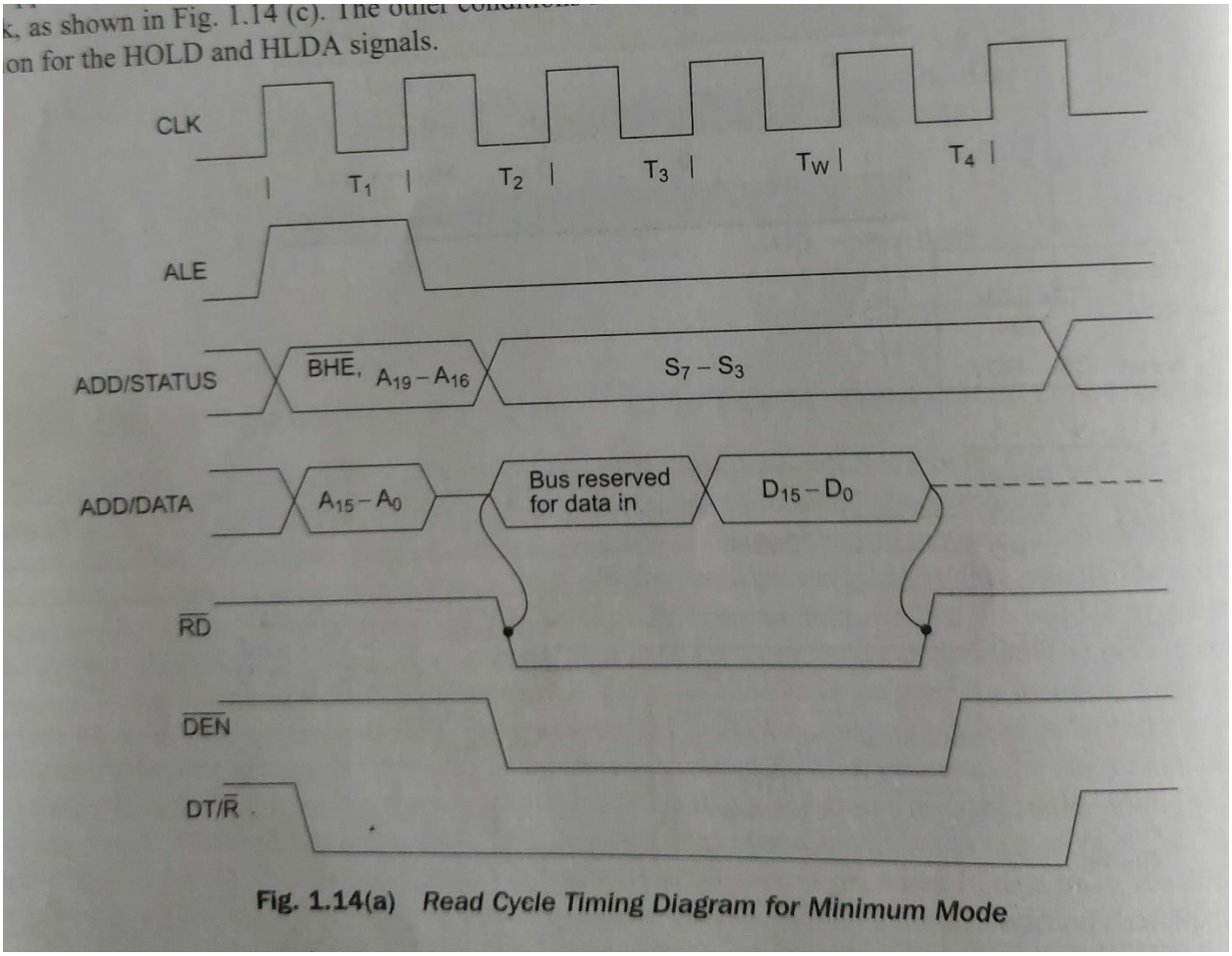


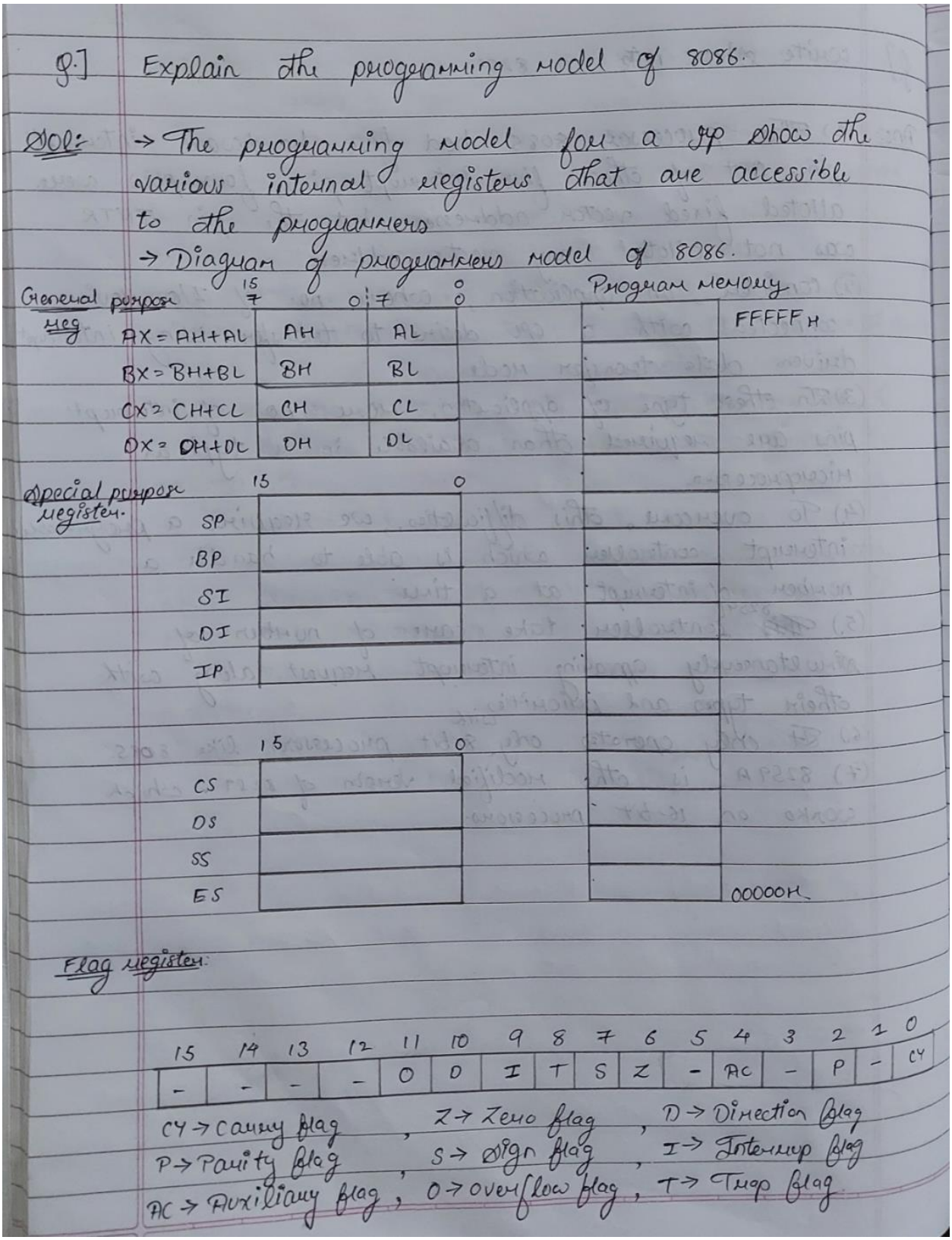
Fig. 1.14(a) *Read Cycle Timing Diagram for Minimum Mode*

Q21) Explain programmer's model of 8086 microprocessor.

Q.] Explain the programming model of 8086.

Sol: → The programming model for a µp show the various internal registers that are accessible to the programmers

→ Diagram of programmers model of 8086.



Flag register:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	O	D	I	T	S	Z	-	AC	-	P	-	CY

CY → Carry flag, Z → Zero flag, D → Direction flag
P → Parity flag, S → Sign flag, I → Interrupt flag
AC → Auxiliary flag, O → Overflow flag, T → Trap flag

In the programming model there are:
- 4 general purpose registers
- 4 segment registers
- 2 pointer registers
- 2 index registers
- 1 instruction pointer register
- 1 flag register

→ General purpose registers
(1) AX register: This is accumulator. It is used in arithmetic, logic and data transfer instructions.
(2) BX register: It is a base register. It usually contains data pointers used for based, based indexed or register indirect addressing.
(3) CX register: This is count register. Program loop constructions are facilitated by it.
(4) DX register: This is data register. Can be used as port number in I/O operations.

→ Segment registers:
(1) code segment: It is used for addressing a memory location in the code segment of the memory.
(2) Data segment: It points to the data segment of the memory
(3) Stack segment: It is used for addressing stack segment of the memory
(4) Extra segment: It is another refers to it the segment which essentially is another data segment of the memory.

→ Pointer registers:

(1) SP Register: This is stack pointer register pointing to program stack

(2) BP Register: This is Base pointer register pointing to data in stack segment.

→ Index Register:

(1) SI Register (Source Index): This is used to point to memory locations in the data segment addressed by DS.

(2) DI Register. (Destination Index): This register performs the same function as SI. There is a class of instruction called string instruction that use DI to access the memory locations addressed by ES.

→ Instruction pointer: The instruction pointer points to the address of the next instruction to be executed.

→ Flag register: The 8086 flag register contents include the result of computation in the ALU.

(23) Explain Branch prediction logic with neat diagram.

Ans: same as Q.3.

(22) Explain BSR Mode of 8255.

Ans: → In this mode, any of the 8-bit of port c can be set or reset depending on B_0 of the control word.

→ The bit to be set or reset is selected by bit select flag B_3, B_2, B_1. of the CWR. as given in the below table

B_3	B_2	B_1	Selected Bits of port c
0	0	0	B_0
0	0	1	B_1
0	1	0	B_2
0	1	1	B_3
1	0	0	B_4
1	0	1	B_5
1	1	0	B_6
1	1	1	B_7

→ CWR Format:

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
1	X	X	X				

B_7: 0 - for BSR mode.

B_3, B_2, B_1: Bit select flag

B_0: 0 - Reset, 1 - set

B_3, B_2, B_1 are from 000 to 111 for bits PC0 to PC7.

Q24) With neat diagram explain Net burst micro architecture of Pentium 4

14.6.2 Pentium 4 Architecture

Q. Explain Intel's Net Burst Micro architecture with neat schematic. Also highlight on hyper-pipeline concept and rapid execution engine. **(5 Marks)**

Q. Write short note on Intel's Net burst micro-architecture **(5 Marks)**

The basic blocks of the Pentium 4 processor are shown in Fig. 14.6.1.

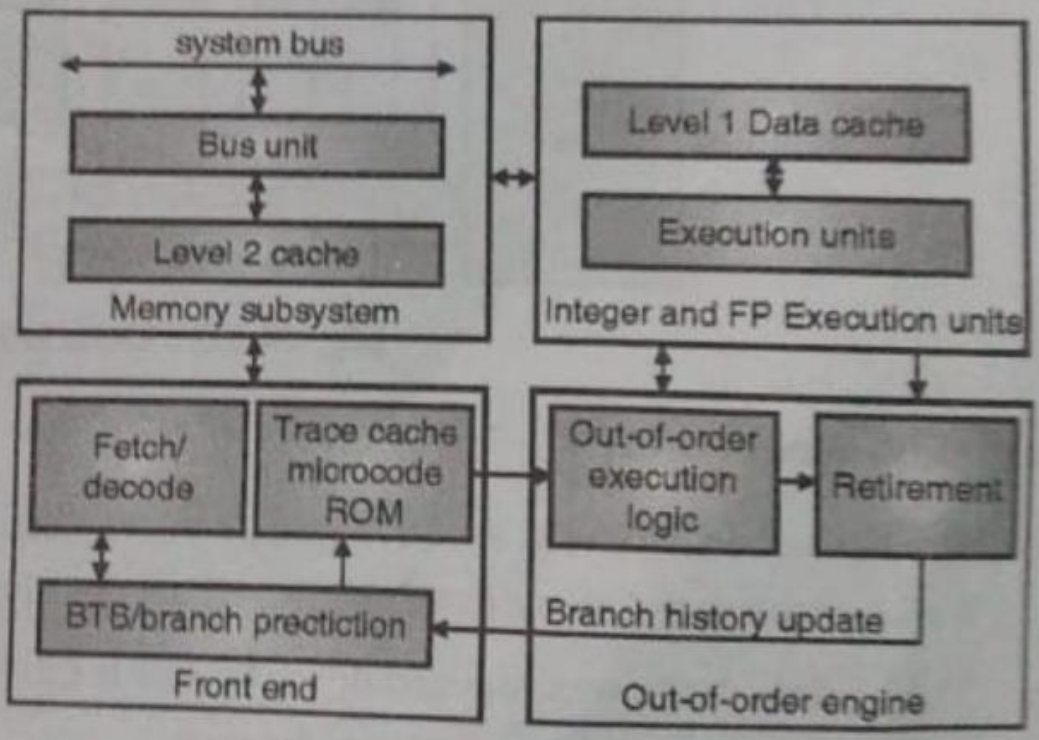


Fig. 14.6.1 : Basic Blocks of Pentium 4 processor

The basic blocks of Pentium 4, as shown in Fig. 14.6.1, consists of Memory subsystem, Front end, Out-of-order Engine and the Integer and Floating point execution units.

1. Memory Subsystem

- This unit handles the memory accesses and also consists of the L2 cache.
- The L2 cache is an Advanced Trace Cache as discussed in the features of the processor.

2. Front End

- This block consists of a Fetch / Decode unit, Trace cache along with microcode ROM and BTB along with branch prediction logic.
- The fetch decode unit fetches the instructions and decodes them.
- The trace cache, as discussed already, stores the decoded instructions and hence reduces the time for decoding in loops.
- The branch prediction logic predicts the sequence of instructions during a branch instruction with the help of branch history stored in Branch Target Buffer (BTB).

3. Out-of-order Engine

- The out-of-order engine executes the instructions out of order in case of their dependencies and hence keeps the execution units continuously busy.
- But the retirement of the instructions are in order.

4. Execution units

- There are integer execution units to execute integer instructions, floating point unit to floating point instructions and MMX unit to execute SIMD vector instructions. The detailed block diagram of Pentium 4 is shown in Fig. 14.6.2.

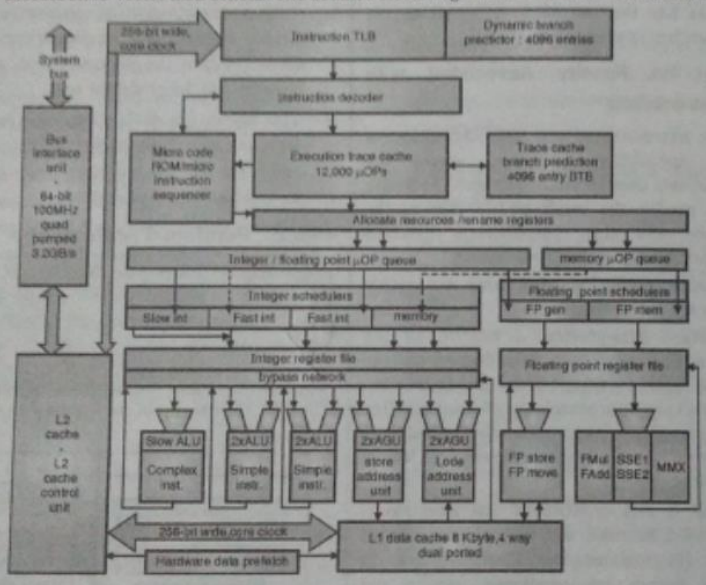
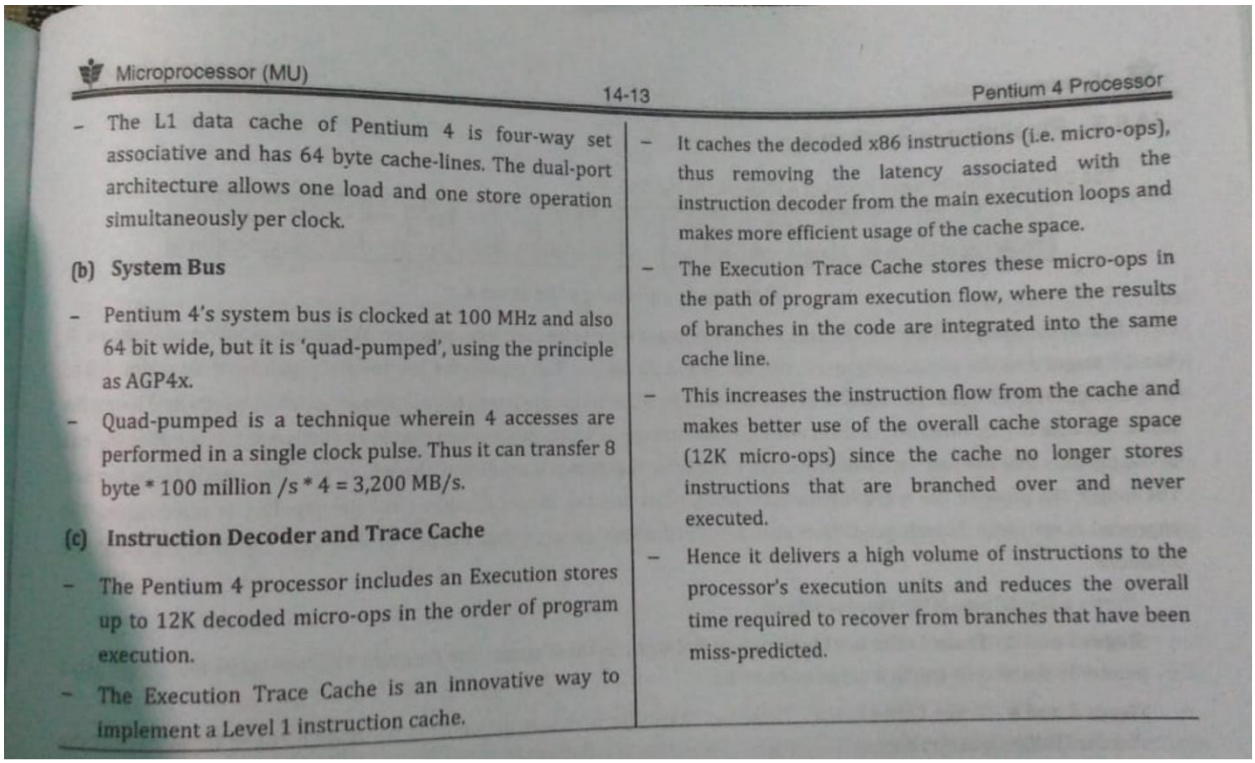


Fig. 14.6.2 : Detailed Block Diagram of Pentium 4

(a) AdvancedL2Cache

- The L2-cache also called as 'Advanced Transfer Cache' is 256 KB (or 512KB or 1 MB) and is 8-way associative. Pentium 4's L2-cache uses 128 byte cache lines, which are divided in two 64-byte pieces.
- The L1 data cache was reduced down to only 8 KB, to enable its extremely low latency of only 2 clock cycles and hence results in an overall improved read latency.



- The L1 data cache of Pentium 4 is four-way set associative and has 64 byte cache-lines. The dual-port architecture allows one load and one store operation simultaneously per clock.

(b) System Bus

- Pentium 4's system bus is clocked at 100 MHz and also 64 bit wide, but it is 'quad-pumped', using the principle as AGP4x.
- Quad-pumped is a technique wherein 4 accesses are performed in a single clock pulse. Thus it can transfer 8 byte * 100 million /s * 4 = 3,200 MB/s.

(c) Instruction Decoder and Trace Cache

- The Pentium 4 processor includes an Execution stores up to 12K decoded micro-ops in the order of program execution.
- The Execution Trace Cache is an innovative way to implement a Level 1 instruction cache.
- It caches the decoded x86 instructions (i.e. micro-ops), thus removing the latency associated with the instruction decoder from the main execution loops and makes more efficient usage of the cache space.
- The Execution Trace Cache stores these micro-ops in the path of program execution flow, where the results of branches in the code are integrated into the same cache line.
- This increases the instruction flow from the cache and makes better use of the overall cache storage space (12K micro-ops) since the cache no longer stores instructions that are branched over and never executed.
- Hence it delivers a high volume of instructions to the processor's execution units and reduces the overall time required to recover from branches that have been miss-predicted.

Q25) Same as Q3.

Q27) Write an 8086 assembly language program to print content of flag register.

```
start:
 mov ax,Data
 mov DS,ax

 mov dx,offset msg
 mov ah,09h
 int 21h

 mov dx,offset newl
 mov ah,09h
 int 21h

 cli
 stc
 std

 pushf

 pop bx

 mov flag,bx

 mov cx,16
 mov bx,8000h

loops:
 mov ax,flag
 and ax,bx
 jz zero
 mov dl,31h
 mov ah,02h
 int 21h
 jmp space

zero: mov dl,30h
 mov ah,02h
 int 21h

space: mov dl,' '
 mov ah,02h
 int 21h

 mov ah,02h
 int 21h
 ror bx,1

 loop loops

 mov ah,4ch
 int 21h
Code ends
end start
```