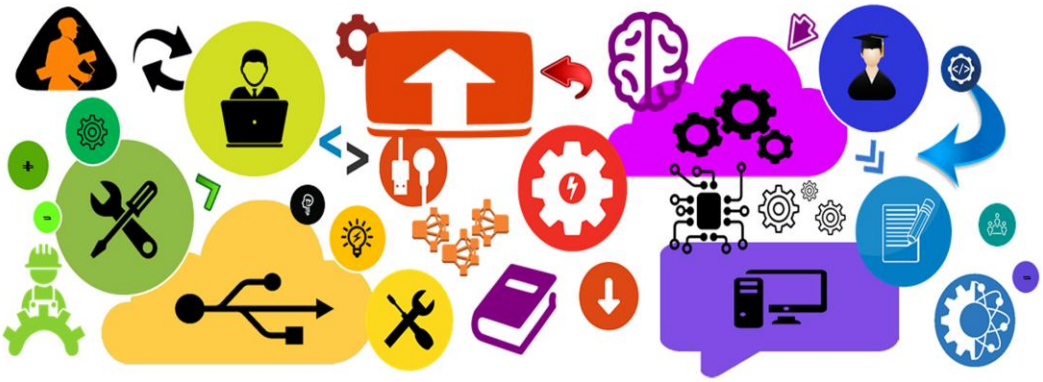
www.ToppersSolutions.com Support@ToppersSolutions.com

Topper's Solutions

....In Search of Another Topper



Microprocessor

Sem-5 (Computer)

(As Per Revised Syllabus 2012)

Topper's Solutions

.....In search of Another Topper

There are many existing paper solution available in market, but Topper's Solution is the one which students will always prefer if they refer... ;) Topper's Solution is not just paper solution, it includes many other important questions which are important from examination point of view. Topper's Solutions are the solution written by the Toppers for the students to be the upcoming Topper of the Semester.

It has been said that "**Action Speaks Louder than Words**" So Topper's Solution works on same principle. Diagrammatic representation of answer is considered to be easy & quicker to understand. So our major focus is on diagrams & representation how answers should be answered in examinations.

Why Topper's Solution:

- Point wise answers which are easy to understand & remember.
- Diagrammatic representation for better understanding.
- Additional important questions from university exams point of view.
- Covers almost every important question.
- In search of another topper.

****** Education is Free.... But its Technology used & Efforts utilized which charges ******

It takes lot of efforts for searching out each & every question and transforming it into Short & Simple Language. Entire Topper's Solutions Team is working out for betterment of students, do help us.

"Say No to Xerox...."

With Regards,

Topper's Solutions Team.

Microprocessor (MP)

Marks Distribution

CHAPTER NO.	CHAPTER NAME	DEC 2014	MAY 2015	DEC 2015	MAY 2016
1.	Intel 8086/8088 Architecture	25	35	25	35
2.	Instruction Set & Programming	15	05	10	-
3.	System Designing with 8086	25	30	20	22
4.	Intel 80386DX Processor	35	15	20	13
5.	Pentium Processor	20	25	35	28
6.	SuperSPARC Architecture	10	10	15	22
-	**Repeated Questions**	-	**45**	**55**	**83**

CHAPTER-1: INTEL 8086/8088 ARCHITECTURE

Q1) EXPLAIN PROGRAMMING MODEL OF 8086.

ANS: [5M – MAY16]

1. Programming Model of 8086 is similar to the picture of the processor as available to the programmer.
2. This registers are used to hold number & addresses.
3. It is also used to indicate status and acts as controls.
4. Figure 1.1 shows the 8086 Programming Model.

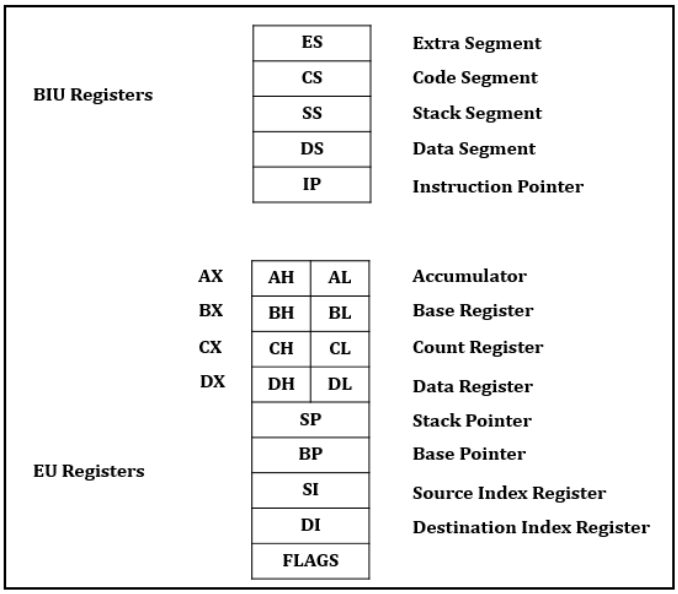


Figure 1.1: 8086 Programming Model.

SEGMENT REGISTERS:

I) Extra Segment (ES):

- It is a 16-bit register containing address of 64KB segment, usually with program data.
- ES register can be changed directly using POP and LES instructions.

II) Code Segment (CS):

- It is a 16-bit register containing address of 64 KB segment with processor instructions.
- The CS register is automatically updated during far jump, far call and far return instructions.

III) Stack Segment (SS):

- It is a 16-bit register containing address of 64KB segment with program stack.
- SS register can be changed directly using POP instruction.

IV) Data Segment (DS):

- It is a 16-bit register containing address of 64KB segment with program data.
- DS register can be changed directly using POP and LDS instructions.

INSTRUCTION POINTER (IP):

- It is a 16-bit register.
- It is used to point the instructions.

GENERAL DATA REGISTERS:

I) Accumulator Register:

- It consists of two 8-bit registers AL and AH, which can be combined together and used as a 16-bit register AX.
- Accumulator can be used for I/O operations and string manipulation.

II) Base register:

- It consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX.
- BX register usually contains a data pointer used for based, based indexed or register indirect addressing.

III) Count register:

- It consists of two 8-bit registers CL and CH, which can be combined together and used as a 16-bit register CX.
- Count register can be used in Loop, shift/rotate instructions and as a counter in string manipulation.

IV) Data register:

- It consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX.
- Data register can be used as a port number in I/O operations.

POINTER & INDEX REGISTERS:

I) Stack Pointer (SP):

- It is a 16-bit register pointing to program stack.

II) Base Pointer (BP):

- It is a 16-bit register pointing to data in stack segment.
- BP register is usually used for based, based indexed or register indirect addressing.

III) Source Index (SI):

- It is a 16-bit register.
- SI is used for indexed, based indexed and register indirect addressing, as well as a source data address in string manipulation instructions.

IV) Destination Index (DI):

- It is a 16-bit register.
- DI is used for indexed, based indexed and register indirect addressing, as well as a destination data address in string manipulation instructions.

FLAGS:

- Flag Register are 16 bits registers.
- Out of 16 bits only 9 bits are used as flags and the rest 7 bits are not used.
- CF, AF, PF, ZF, SF and OF Flags are status indicators.
- While TF, IF, and DF are Control Flags.
- Figure 1.2 shows Flag registers.

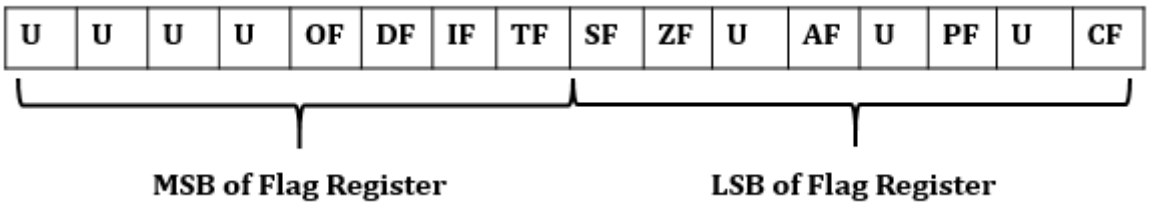


Figure 1.2: Flag Register.

Q2) WHAT IS SEGMENTATION? WHAT ARE THE ADVANTAGES OF SEGMENTATION?

Q3) ADVANTAGES OF MEMORY SEGMENTATION IN 8086.

Q4) EXPLAIN MEMORY SEGMENTATION WITH PROS & CONS.

ANS: [Q2 | 5M – MAY15] & [Q3 | 5M – DEC14 & DEC15] & [Q4 | 8M – MAY16]

1. Memory is considered as vast collection of bytes so it must be organized in efficient manner.
2. The total memory size is divided into segments of various sizes.
3. A segment is just an area in memory.
4. The process of dividing memory this way is called Segmentation.
5. In 8086, memory has four different types of segments.
6. These are: Code Segment (CS), Data Segment (DS), Stack Segment (SS) & Extra Segment (ES).

SEGMENT REGISTERS:

1. These registers are 16-bit in size.
2. The number of address lines in 8086 is 20.
3. Each of these segments is addressed by an address stored in corresponding segment register.
4. Each register stores the base address (starting address) of the corresponding segment.
5. This starting address will always be changing. They are not fix.
6. Because the segment registers cannot store 20 bits, they only store the upper 16 bits.
7. Figure 1.3 shows one of the possible ways to position the four 64 k bytes segments within the 1 M bytes memory space of 8086.

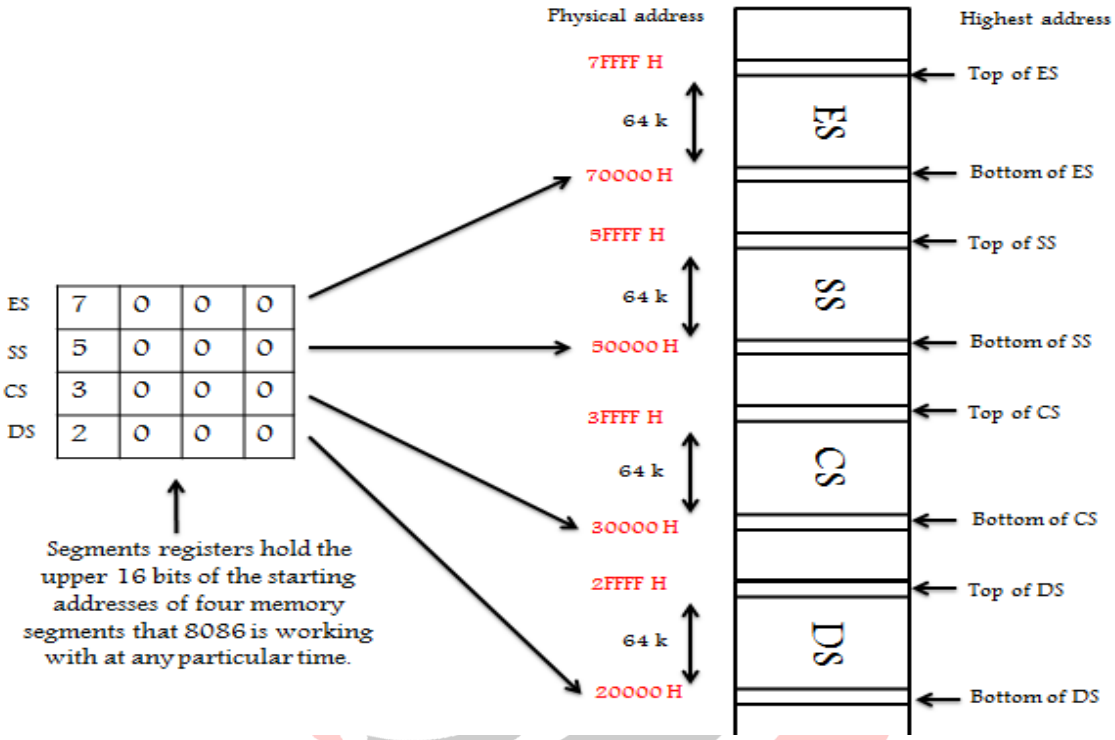


Figure 1.3: Segment Registers Segmentation.

ADVANTAGES OF MEMORY SEGMENTATION:

1. Segmentation provides a powerful memory management mechanism.
2. Segments provide a way to easily implement object oriented programs.
3. With the help of memory segmentation a user is able to work with registers having only 16-bits.
4. Using memory segmentation the data and code can be stored separately allowing for more flexibility.
5. Segmentation makes it possible to separate the memory areas for stack, code and data.
6. Also due to segmentation the logical address range is from 0000H to FFFFH the code can be loaded at any location in the memory.
7. Segmentation makes it possible to write programs which are position independent or dynamically re-locatable.
8. Segmented structure of 8086 memory space supports modular software design.

DISADVANTAGES:

It becomes complicated for the programmer as multiple register to access a memory location.

Q5) MAXIMUM MODE OF 8086

ANS: **[10M – DEC14]**

MAXIMUM MODE:

1. In the maximum mode, the 8086 is operated by strapping the MN/MX pin to ground.
2. In this mode, the processor derives the status signal S2, S1, S0.
3. In the maximum mode, there may be more than one microprocessor in the system configuration.

4. The components in the system are same as in the minimum mode system.
5. The basic function of the bus controller chip IC 8288 is to derive control signals like READ and WRITE (For memory and I/O devices), DEN, DT/R, ALE etc.
6. The bus controller chip has input lines S2, S1, S0 and CLK.
7. These inputs to 8288 are driven by CPU.
8. It derives the outputs ALE, DEN, DT/R, MRDC, MWTC, AMWC, IORC, IOWC and AIOWC.
9. The AEN, IOB and CEN pins are especially useful for multiprocessor systems.
10. AEN and IOB are generally grounded. CEN pin is usually tied to +5V.
11. The significance of the MCE/PDEN output depends upon the status of the IOB pin.
12. If IOB is grounded, it acts as master cascade enable to control cascaded 8259A; else it acts as peripheral data.
13. INTA pin used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device.
14. IORC, IOWC are I/O read command and I/O write command signals respectively.
15. These signals enable an IO interface to read or write the data from or to the address port.
16. The MRDC, MWTC are memory read command and memory write command signals respectively and may be used as memory read or write signals.
17. All these command signals instructs the memory to accept or send data from or to the bus.
18. Figure 1.4 shows the Maximum Mode Block Diagram of 8086.

BLOCK DIAGRAM:

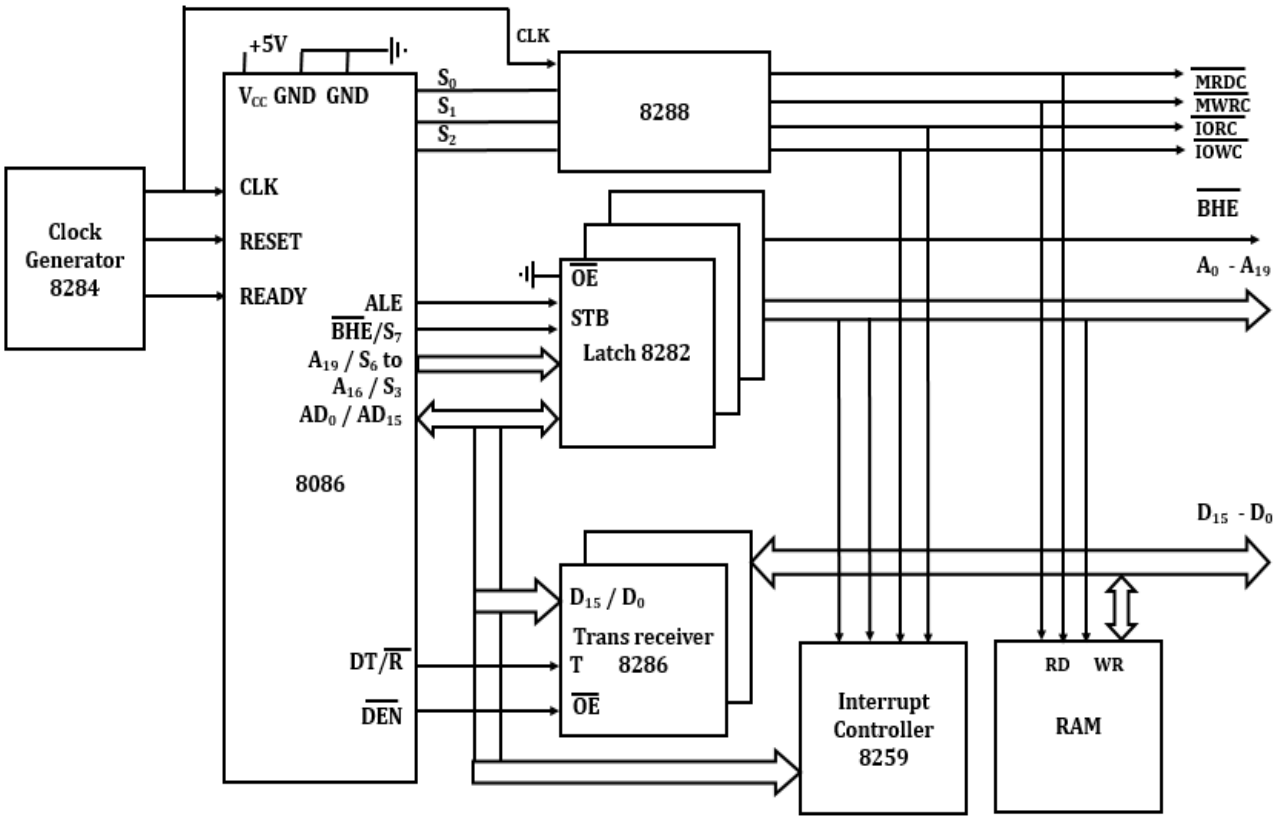


Figure 1.4: Maximum Mode of 8086.

Q6) DRAW AND EXPLAIN TIMING DIAGRAM FOR READ OPERATION IN MINIMUM MODE OF 8086

ANS: [10M - DEC14 & MAY16]

TIMING DIAGRAM:

1. In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX* pin to logic1.
2. In this mode, all the control signals are given out by the microprocessor chip itself.
3. Figure 1.5 shows the timing diagram for Read Machine Cycle of 8086.

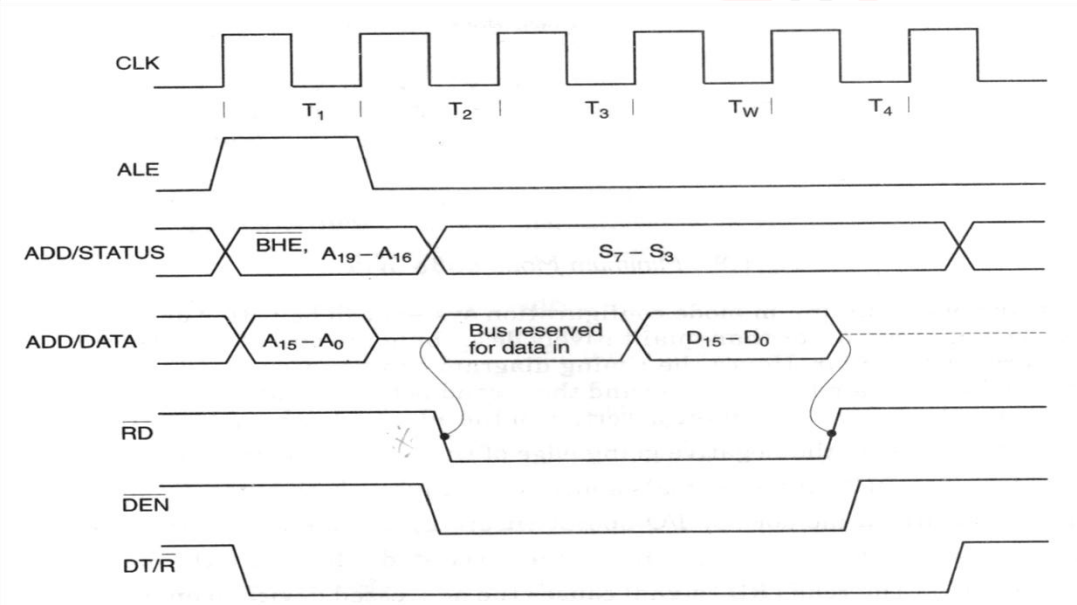


Figure 1.5: Timing Diagram for Read Machine Cycle of 8086.

SEQUENCES OF OPERATIONS DURING READ MACHINE CYCLE ARE AS FOLLOWS:

1. The 8086 will make M/$\overline{IO}$ = 1 if the read is from memory and M/$\overline{IO}$ = 0 if the read id from the I/O device.
2. Then ALE output is asserted to 1 i.e. it is made high.
3. Make $\overline{BHE}$ low/high and send out the desired address of memory location to be read on AD_0 to AD_{15} and A_{16} to A_{19} address lines.
4. After sometime pull down ALE (make it 0) i.e. it is made inactive. The address then gets latched into external latch.
5. 8086 removes the address from AD_0 to AD_{15} lines so that they can be used to read the data from memory and put them in the input mode (float them).
6. Assert the $\overline{RD}$ (read) signal low. This will put the data from the addressed memory location or I/O port on to the data bus.
7. Insert the "wait" T-states if the 8086 READY input is made low before or during the T_2 state of a machine cycle.
8. As soon as READY input goes high, 8086 comes out of the wait T-states and completes the machine cycle.
9. Complete the "Read" cycle by making the $\overline{RD}$ line high (inactive).
10. For larger systems we need to use the data buffers. Then the DT/$\overline{R}$ and $\overline{DEN}$ signals of 8086 are connected and enabled at the appropriate time.

Q7) DRAW AND EXPLAIN TIMING DIAGRAM FOR WRITE OPERATION IN MINIMUM MODE OF 8086

ANS: [10M - DEC15]

TIMING DIAGRAM:

1. In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX* pin to logic1.
2. In this mode, all the control signals are given out by the microprocessor chip itself.
3. Figure 1.6 shows the timing diagram for Write Machine Cycle of 8086.

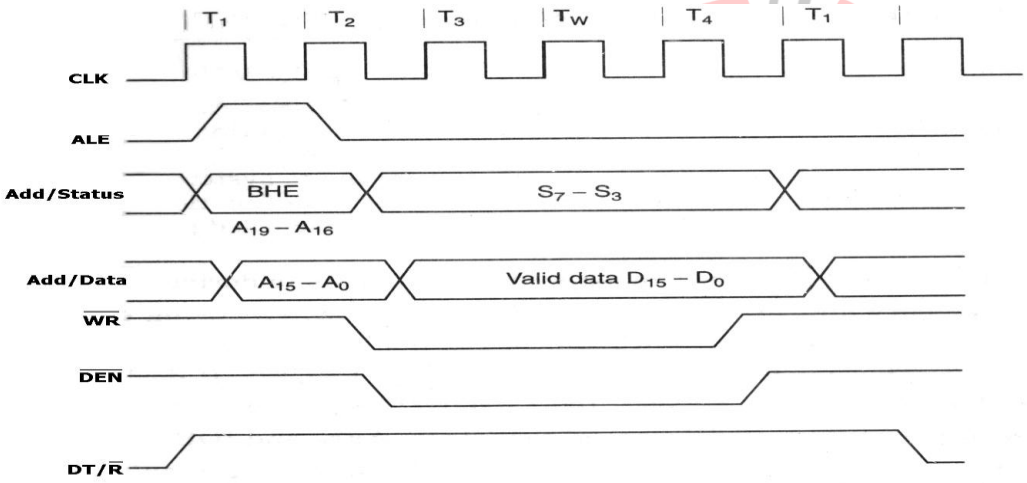


Figure 1.6: Timing Diagram for Write Machine Cycle of 8086.

SEQUENCES OF OPERATIONS DURING WRITE MACHINE CYCLE ARE AS FOLLOWS:

1. During T_1 state, the 8086 will make M/IO = 0 if it has to write to an I/O port and M/IO = 1 if the memory location is to be written.
2. Then ALE output is asserted to 1 i.e. it is made high. This will enable the address latches.
3. 8086 then outputs the BHE and the address of the desired port or memory location on the lines AD_0 to AD_{15} and A_{16} to A_{19}. Note that the address lines A_{16} to A_{19} = 0 if the port is to be written.
4. After sometime ALE output is pulled down. The address gets latched into external latches.
5. The 8086 removes the address information from AD_0 to AD_{15} and outputs the desired data on the data bus.
6. It then asserts the WR signal low. The low WR signal will turn on the memory or port where the data is to be written.
7. 8086 then gives sometime for the addressed I/O port or memory to accept the data from the data bus and then raise the WR output high and floats the data bus.
8. If the addressed memory or port devices cannot accept the data within the normal machine cycle, then the external hardware can be setup to produce a low READY input.
9. If the READY input is pulsed low before or during the T-state T_2 of the write machine cycle, then 8086 will introduce WAIT T-states after T_3 as long as the READY input is held low.
10. During the WAIT T-states, the signals on the data bus, address bus and control bus do not change.
11. If the READY input is made high before the end of a wait T-state, then 8086 will continue with the T_4 state as soon as it finishes the WAIT T-state.
12. If the system is large enough then it will need the external data buffers. Then the DT/R is used to decide the direction of data transfer.

13. During the write cycle, the 8086 pushes the DT/R high to configure the external data buffers in the "transmit" mode. Then the DEN signal is asserted low so as to enable the buffers.
14. The data output from 8086 is then passed through the data buffers to the addressed memory or I/O port.

Q8) WRITE SHORT NOTE ON 8288 BUS CONTROLLER.

ANS: **[5M – MAY15]**

8288 BUS CONTROLLER:

1. The Intel 8288 is a bus controller designed for Intel 8086/8087/8088/8089.
2. 8288 Bus Controller is a 20 Pin Bipolar IC.
3. In maximum mode, 8086 doesn't provide various control signals like ALE, DT/$\overline{R}$, $\overline{DEN}$ etc. & command signals like $\overline{WR}$, M/$\overline{IO}$ etc.
4. Hence 8288 Bus Controller is used to provide all these signals.
5. 8288 Bus Controller accepts the CLK signal along with $\overline{S_0}$, $\overline{S_1}$ & $\overline{S_2}$ outputs of 8086 and generates command, control & timing signals at its output.
6. 8288 Bus Controller is used to optimize the system performance.
7. It also provides the bipolar bus drive capability.
8. Figure 1.7 shows the internal block diagram of 8288 Bus Controller.

BLOCK DIAGRAM:

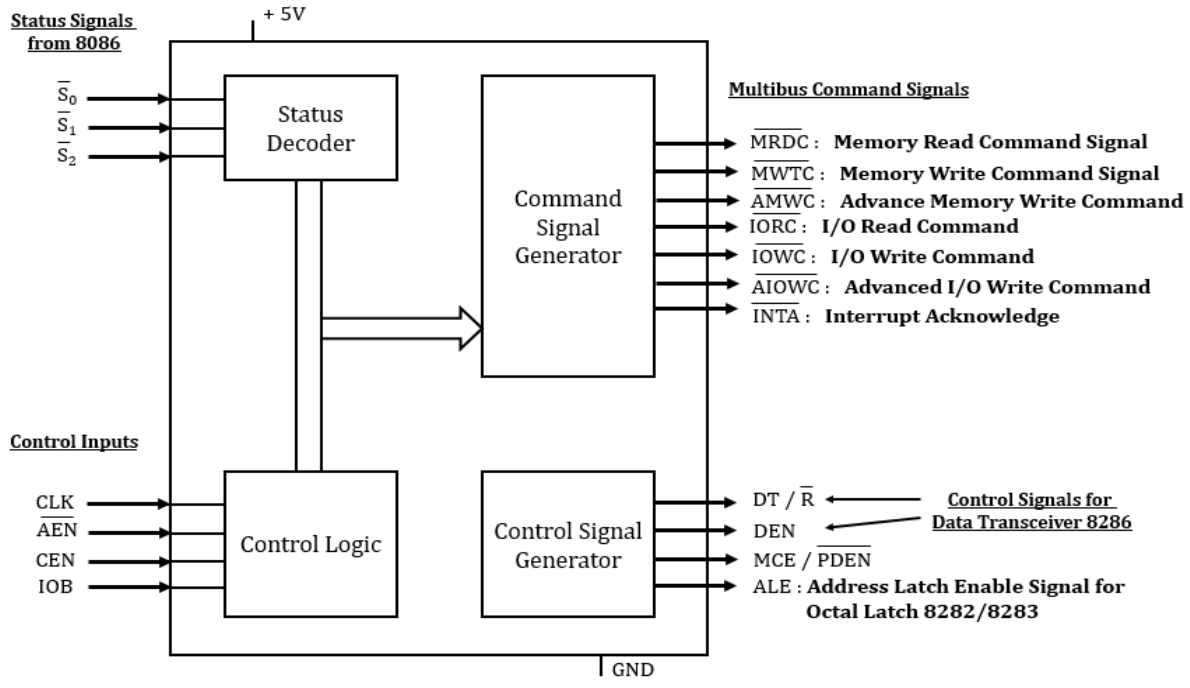


Figure 1.7 Internal Block Diagram of 8288 Bus Controller.

1. Block Diagram of 8288 Bus Controller includes Status Decoder, Control Logic, Command Signal Generator and Control Signal Generator.
2. 8288 Bus Controller accepts the CLK signal along with S_0, S_1 & S_2 outputs of 8086 and generates command, control & timing signals at its output.

Q9) STATE USE OF CONTROL FLAGS OF 8086.

ANS: [5M – DEC15]

1. Out of the 9 Active Flags, 6 are conditional (status) flags and the remaining 3 are called as control flags.
2. Control flags are used to control certain operations of the processors.
3. Figure 1.8 shows the 8086 flag register format.

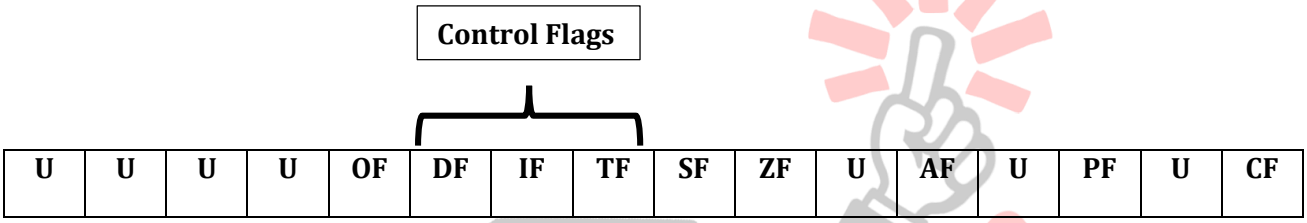


Figure 1.8: 8086 Flag Register Format.

I) TRAP FLAG (TF):

- Setting TF puts the processor into single step mode for debugging.
- In single stepping, microprocessor executes an instruction and enters into single step ISR.
- After that the user can check register.
- This utility is, to debug the program.
- If TP = 1, the CPU automatically generates an internal interrupt after each instruction, allowing a program to be inspected as it executes instruction by instruction.
- This flag is used by debuggers for single step operations.
- TF = 1 then Trap On, TF = 0 then Trap Off.

II) INTERRUPT FLAG (IF):

- If user sets IF flag, the CPU will recognize external interrupt requests.
- Clearing IF disables these interrupts.
- IF Flag has no effect on either non-maskable external or internally generated interrupt.
- The IF Flag is used for allowing or prohibiting the interruption of a program.
- IF = 1 then Interrupt Enabled, IF = 0 then Interrupt Disabled.

III) DIRECTION FLAG (DF):

- DF Flag is used for string instructions.
- In string instructions we use SI (Source Index) & DI (Destination Index) registers as offset registers to point source & destination area respectively.
- DF Flags controls the direction of SI & DI Pointers.
- If DF = 1, the string instruction will automatically decrement the pointers.
- If DF = 0, the string instruction will automatically increment the pointers.
- DF = 1 then Up, DF = 0 then Down.

Q10) EXPLAIN THE FOLLOWING INSTRUCTIONS IN 8086: LAHF & STOSB

ANS:

LAHF: [5M – MAY15]

1. LAHF stands for Load AH register from flags.
2. LAHF is one of the Flag Instructions in 8086.
3. Flag Instructions are the instructions which are related to movement of flag register to/from a register and memory.
4. No flags are affected.
5. LAHF uses Implied Addressing Mode.
6. In LAHF the lower byte of 8086 flag register is copied to the AH register.

Before Execution:

			MSB of flag register = OC H								**LCB of flag register = C6 H**							
AH	**AL**		**U**	**U**	**U**	**U**	**OF**	**DF**	**IF**	**TF**	**SF**	**ZF**	**U**	**AF**	**U**	**PF**	**U**	**CF**
A_0	03		0	0	0	0	1	1	0	0	1	1	0	0	0	1	1	0

After Execution:

			MSB of flag register = OC H								**Contents of LSB of flag register = C6 H are copied to AH register**							
AH	**AL**		**U**	**U**	**U**	**U**	**OF**	**DF**	**IF**	**TF**	**SF**	**ZF**	**U**	**AF**	**U**	**PF**	**U**	**CF**
C_6	03		0	0	0	0	1	1	0	0	1	1	0	0	0	1	1	0

STOSB:

1. STOSB stands for String Byte in String.
2. STOSB is one of the String Instructions in 8086.
3. STOSB is used for Byte Operation whereas STOSW is used for Word Operation.
4. No flags are affected.
5. STOSB uses String Addressing Mode.
6. STOSB transfers a byte from register AL or AX to the string element addressed in ES by DI.
7. Depending on DF flag, DI is automatically incremented or decremented.
8. **Example:**

```
MOV DI, OFFSET STR 1;    DI points to top of string STR 1.
CLD;                     DF = 0
MOV AX, 00H
STOSB;                   Store string by AX value.
```

Algorithm:

ES: [DI] = AL

IF DF = 0 then DI = DI + 1

Else DI = DI – 1

Q11) DIFFERENTIATE BETWEEN MINIMUM MODE & MAXIMUM MODE IN 8086.

ANS: [5M – MAY15]

Table 1.1: Comparison between minimum mode & maximum mode in 8086.

Parameter	Minimum Mode	Maximum Mode
Operating Mode	The microprocessor system operates in the minimum mode by default on power on.	If MN/$\overline{MX}$ pin is grounded, the system operates in maximum mode.
CPU	When only one CPU is to be used in a microcomputer system it is used in minimum mode operation.	Maximum mode uses Multiple CPUs to operate.
Control Signals	CPU issues the control signals that are required by the memory and I/O devices.	In Maximum mode the control signals are issued by the Intel 8288 bus controller.
Pins Used	M/$\overline{IO}$, $\overline{INTA}$, ALE, HOLD, HLDA, DT/$\overline{R}$, $\overline{DEN}$, $\overline{WR}$	$\overline{S_2}$, $\overline{S_1}$, $\overline{S_0}$, $\overline{LOCK}$, QS_1, QS_0, $\overline{RQ_0}$ / $\overline{GT_0}$, $\overline{RQ_1}$ / $\overline{GT_1}$
Cost	Less Expensive.	More Expensive.
External Bus Controller	Not required.	Required like 8288
Memory & I/O Control Signals	Memory & I/O Control Signals are generated by the microprocessor itself.	Control Signals are generated by External Bus Controller like 8288.

Q12) DESIGN 8086 BASED MINIMUM MODE SYSTEM FOR FOLLOWING REQUIREMENT:

- **256 KB of RAM using 64 KB x 8 Bit device.**
- **128 KB of RAM using 64 KB x 8 Bit device.**
- **Three 8 Bit Parallel Ports using 8255.**
- **Support for 8 interrupts.**

ANS: [12M – MAY16]

STEP-1:

Total EPROM required = 256 KB

Chip size available = 64 KB

∴ Number of chips $= \frac{256\ KB}{64\ KB} = 4$

∴ Number of sets required $= \frac{No.of\ Chips}{No.of\ Banks} = \frac{4}{2} = 2$

Set-1:

Ending address = FFFFFH

Set size = chip size x 2

= 64 KB x 2 = 128 KB …….. (Since 128 KB = 1FFFFH)

Starting address = Ending Address – Set size

= FFFFF H – 1FFFF H

= E0000 H

	Even Bank	**Odd Bank**
Starting Address	E0000 H	E0001 H
Ending Address	FFFFE H	FFFFF H

Set-2:

Ending address = Previous Starting Address – 1

= E0000 H – 1

= DFFFF H

Set size = 1FFFF H

Starting address = DFFFF H – 1FFFF H

= C0000 H

	Even Bank	**Odd Bank**
Starting Address	C0000 H	C0001 H
Ending Address	DFFFE H	DFFFF H

STEP-2:

Total SRAM required = 128 KB

Chip size available = 64 KB

∴ Number of chips $= \frac{128\ KB}{64\ KB} = 2$

∴ Number of sets required $= \frac{No.of\ Chips}{No.of\ Banks} = \frac{2}{2} = 1$

Set-1:

Starting address = 00000 H

Set size = Chip size x 2

= 64 x 2 = 128 KB = 1FFFF H

Ending address = Starting address + Set size

= 00000 H + 1FFFF H

= 1FFFF H

	Even Bank	Odd Bank
Starting Address	00000 H	00001 H
Ending Address	1FFFE H	1FFFF H

STEP-3: MEMORY MAP

RAM:

			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
SET-1	**EB**	SA = 00000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = 1FFFE	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = 00001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = 1FFFF	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

ROM:

			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
SET-1	**EB**	SA = E0000	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = FFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = E0001	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = FFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SET-2	**EB**	SA = C0000	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = DFFFE	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = C0001	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = DFFFF	1	1	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

STEP-4: I/O MAP

- For three 8 Bit Parallel Ports we need two chips of 8255.

		A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
8255 **Even Bank** A_7	PA = 00 H	0	0	0	0	0	0	0	0	0
	PB = 02 H	0	0	0	0	0	0	0	1	0
	PC = 04 H	0	0	0	0	0	0	1	0	0
	CW = 06 H	0	0	0	0	0	0	1	1	0

		A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
8255	PA = 01 H	0	0	0	0	0	0	0	0	1
Odd Bank	PB = 03 H	0	0	0	0	0	0	0	1	1
A_7	PC = 05 H	0	0	0	0	0	0	1	0	1
	CW = 07 H	0	0	0	0	0	0	1	1	1

- For 8 Interrupts we need two 8259, one as a master & other as a slave.

		A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
8259	Address 1 = C0 H	1	1	0	0	0	0	0	0	0
Even Bank	Address 2 = C2 H	1	1	0	0	0	0	0	1	0
A_7	Address 1 = C1 H	1	1	0	0	0	0	0	0	1
Odd Bank	Address 2 = C3 H	1	1	0	0	0	0	0	1	1

Q13) Design 8086 microprocessor based system with following specifications:

- **Microprocessor 8086 working at 10 MHz in minimum mode.**
- **32 KB EPROM using 8 KB chips.**
- **16 KB SRAM using 4 KB chips.**

Ans: [10M – DEC14]

STEP-1:

Total EPROM required = 32 KB

Chip size available = 8 KB

∴ Number of chips $= \frac{32\ KB}{8\ KB} = 4$

∴ Number of sets required $= \frac{\text{No.of Chips}}{\text{No.of Banks}} = \frac{4}{2} = 2$

Set-1:

Ending address = FFFFFH

Set size = chip size x 2

= 8 KB x 2 = 16 KB …… (Since 16 KB = 03FFFH)

Starting address = Ending Address – Set size

= FFFFF H – 03FFF H

= FC000 H

	Even Bank	**Odd Bank**
Starting Address	FC000 H	FC001 H
Ending Address	FFFFE H	FFFFF H

Set-2:

Ending address = Previous Starting Address – 1
= FC000 H – 1
= FBFFF H

Set size = 03FFF H

Starting address = FBFFF H – 03FFF H
= F8000 H

	Even Bank	Odd Bank
Starting Address	F8000 H	F8001 H
Ending Address	FBFFE H	FBFFF H

STEP-2:

Total SRAM required = 16 KB

Chip size available = 4 KB

∴ Number of chips $= \frac{16\ KB}{4\ KB} = 4$

∴ Number of sets required $= \frac{No.of\ Chips}{No.of\ Banks} = \frac{4}{2} = 2$

Set-1:

Starting address = 00000 H

Set size = Chip size x 2
= 4 x 2 = 8 KB = 01FFF H

Ending address = Starting address + Set size
= 00000 H + 01FFF H
= 01FFF H

	Even Bank	Odd Bank
Starting Address	00000 H	00001 H
Ending Address	01FFE H	01FFF H

Set-2:

Starting address = Previous Ending Address + 1
= 01FFF H + 1 = 02000 H

Set size = Chip size x 2
= 4 x 2 = 8 KB = 01FFF H

Ending address = Starting address + Set size
= 02000 H + 01FFF H
= 03FFF H

	Even Bank	Odd Bank
Starting Address	02000 H	02001 H
Ending Address	03FFE H	03FFF H

STEP-3: MEMORY MAP

RAM:

			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
SET-1	**EB**	SA = 00000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = 01FFE	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = 00001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = 01FFF	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
SET-2	**EB**	SA = 02000	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = 03FFE	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = 02001	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = 03FFF	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

ROM:

			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
SET-1	**EB**	SA = FC000	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
		EA = FFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = FC001	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1
		EA = FFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
SET-2	**EB**	SA = F8000	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = FBFFE	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = F8001	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = FBFFF	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Q14) DESIGN 8086 BASED SYSTEM WITH FOLLOWING SPECIFICATIONS:

- **8086 in minimum mode working at 8 MHz.**
- **32 KB EPROM using 16 KB devices.**
- **64 KB SRAM using 32 KB devices.**

ANS: [10M – MAY15]

STEP-1:

Total EPROM required = 32 KB

Chip size available = 16 KB

∴ Number of chips $= \frac{32\ KB}{16\ KB} = 2$

∴ Number of sets required $= \frac{No.of\ Chips}{No.of\ Banks} = \frac{2}{2} = 1$

Set-1:

Ending address = FFFFFH

Set size = chip size x 2

= 16 KB x 2 = 32 KB ..... (Since 32 KB = 07FFFH)

Starting address = Ending Address – Set size

= FFFFF H – 07FFF H

= F8000 H

	Even Bank	**Odd Bank**
Starting Address	F8000 H	F8001 H
Ending Address	FFFFE H	FFFFF H

STEP-2:

Total SRAM required = 64 KB

Chip size available = 32 KB

∴ Number of chips $= \frac{64\ KB}{32\ KB} = 2$

∴ Number of sets required $= \frac{No.of\ Chips}{No.of\ Banks} = \frac{2}{2} = 1$

Set-1:

Starting address = 00000 H

Set size = Chip size x 2

= 32 x 2 = 64 KB = 0FFFF H

Ending address = Starting address + Set size

= 00000 H + 0FFFF H = 0FFFF H

	Even Bank	Odd Bank
Starting Address	00000 H	00001 H
Ending Address	0FFFE H	0FFFF H

STEP-3: MEMORY MAP

RAM:

			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
SET-1	**EB**	SA = 00000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = 0FFFE	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = 00001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = 0FFFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

ROM:

			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
SET-1	**EB**	SA = F8000	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = FFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = F8001	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = FFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Q15) DESIGN 8086 MICROPROCESSOR BASED SYSTEM WITH FOLLOWING SPECIFICATIONS.

- **Microprocessor 8086 working at 8 MHz in Maximum Mode.**
- **32 KB EPROM using 16 KB chips.**
- **16 KB SRAM using 8 KB chips.**

ANS: [10M – DEC15]

STEP-1:

Total EPROM required = 32 KB

Chip size available = 16 KB

∴ Number of chips $= \frac{32\ KB}{16\ KB} = 2$

∴ Number of sets required $= \frac{No.of\ Chips}{No.of\ Banks} = \frac{2}{2} = 1$

Set-1:

Ending address = FFFFFH

Set size = chip size x 2

= 16 KB x 2 = 32 KB ..... (Since 32 KB = 07FFFH)

Starting address = Ending Address – Set size

= FFFFF H – 07FFF H

= F8000 H

	Even Bank	**Odd Bank**
Starting Address	F8000 H	F8001 H
Ending Address	FFFFE H	FFFFF H

STEP-2:

Total SRAM required = 16 KB

Chip size available = 8 KB

∴ Number of chips $= \frac{16\ KB}{8\ KB} = 2$

∴ Number of sets required $= \frac{No.of\ Chips}{No.of\ Banks} = \frac{2}{2} = 1$

Set-1:

Starting address = 00000 H

Set size = Chip size x 2

= 8 x 2 = 16 KB = 03FFF H

Ending address = Starting address + Set size

= 00000 H + 03FFF H

= 03FFF H

	Even Bank	**Odd Bank**
Starting Address	00000 H	00001 H
Ending Address	03FFE H	03FFF H

STEP-3: MEMORY MAP

RAM:

			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
SET-1	**EB**	SA = 00000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = 03FFE	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = 00001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = 03FFF	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1

ROM:

			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
SET-1	**EB**	SA = F8000	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		EA = FFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	OB	SA = F8001	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		EA = FFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

*** EXTRA QUESTIONS ***

Q16) APPLICATIONS OF MICROPROCESSOR

ANS:

1. Microprocessor are used as CPU of Computers.
2. They are sued in industrial control Applications, Calculators, Commercial Appliances, Video Games, Toys Etc.
3. They are used in laboratory for training the students.
4. Microprocessor are used for word processing, database management, storing information and scientific & engineering calculations.
5. Microprocessors are also used in Ticket Reservation Systems, Smart Phones & Cameras.
6. They are also used to measure & control the temperature of a furnace.

Q17) EXPLAIN BASIC MICROPROCESSOR ARCHITECTURE

ANS:

1. Microprocessor is a small IC which can process data.
2. That is it can perform Arithmetic & Logical Operations.
3. But this can also be done from ALU.
4. Figure 1.9 shows the general architecture of microprocessor.

BLOCK DIAGRAM:

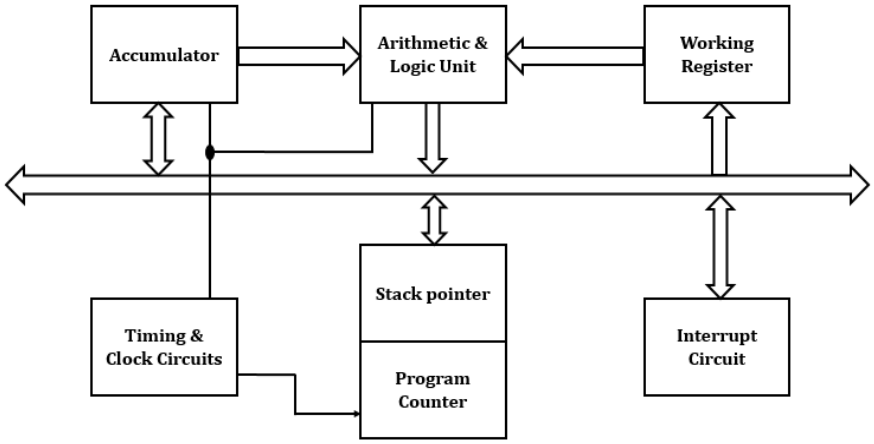


Figure 1.9: General Architecture of Microprocessor.

This architecture is divided into following groups:

I) REGISTERS:

- Microprocessor generally consists of PIPO (Parallel In Parallel Out) Registers.
- Registers are used to store the data & address of the memory.
- The architecture of microprocessor depends upon the number & type of the registers used in microprocessor.
- Microprocessor can consist 8-bit or 16-bit registers.
- Registers are classified as:
  - General Purpose Registers.
  - Temporary Registers.
  - Special Purpose Registers.

II) ARITHMETIC & LOGIC UNIT:

- ALU is used to perform the Arithmetic & Logical operations.
- It performs Arithmetic Operations like Addition, Subtraction & Logical Operations like AND, OR, EX-OR etc.
- ALU is controlled by Timing & Control Circuits.
- ALU is used to provide the Status of result to the flag register.
- ALU looks after the branching decisions.

III) INTERRUPT CONTROL:

- Interrupt Control accepts different interrupt request inputs.
- When a valid interrupt request is present it informs control logic to take action in response to each signal.

IV) TIMING & CONTROL UNIT:

- Timing & Control Unit controls all internal and external circuits.
- It operates with the reference to clock signal.
- It accepts information from instruction decoder & generates micro steps to perform it.
- This unit synchronizes all the data transfers.

Q18) EXPLAIN 8086 MICROPROCESSOR ARCHITECTURE

ANS:

FEATURES:

1. Intel 8086 was launched in 1978.
2. 8086 is 16-bit microprocessor.
3. This microprocessor had major improvement over the execution speed of 8085.
4. It is available as 40-pin Dual-Inline-Package (DIP).
5. It is available in three versions:
   a. 8086 (5 MHz)
   b. 8086-2 (8 MHz)
   c. 8086-1 (10 MHz)
6. It includes 16 bit address bus & 20 bit data bus.
7. It consists of 29,000 transistors.
8. 8086 can operate in 2 modes i.e. maximum and minimum modes.

BLOCK DIAGRAM:

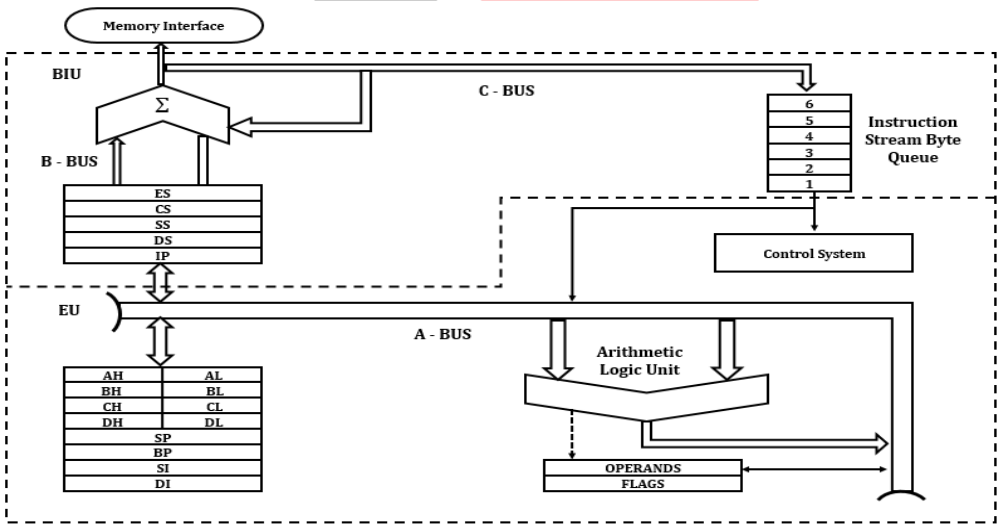


Figure 1.10: 8086 Microprocessor Block Diagram.

The 8086 CPU is divided into two independent functional units: Bus Interface Unit (BIU) & Execution Unit (EU)

I) EXECUTION UNIT (EU):

Execution unit performs the following functions:

a. The main function is decoding and execution of the instructions.
b. It performs the logic and arithmetic operation on memory or register.
c. It receives the instruction from Prefetch Queue and decodes it.
d. It stores the information in the register array.

Execution Unit consists of following parts:

ALU:

- EU has 16-bit ALU so it can perform 16-bit operations simultaneously.
- ALU is used to perform arithmetic and logical operations on 8-bit as well as 16-bit. (Addition, Subtraction, AND, OR, Increment, Decrement, Shift)

Flag Register:

- EU has 16-bit Flag register.
- Flag registers are used to control the certain operations of EU.
- Flag registers includes carry flag, parity flag, auxiliary flag, zero flag, sign flag, trap flag, interrupt flag, direction flag and overflow flag.

General Purpose Registers:

- EU has 8 general purpose registers named as AL, AH, BL, BH, CL, CH, DL & DH.
- These registers can be used as 8-bit registers individually or can be used as 16-bit in pair to have AX, BX, CX, and DX.

AX Register: AX register is also known as accumulator register that stores operands for arithmetic operation like divided, rotate.

BX Register: This register is mainly used as a base register.

CX Register: It is defined as a counter. It is primarily used in loop instruction to store loop counter.

DX Register: DX register is used to contain I/O port address for I/O instruction.

Control Circuit:

- It is used to control all the operations & flow of data in the microprocessor.
- The EU contains the control circuit to perform various internal operations.

Instruction decoder:

- It is used to decode the instructions which are fetched from memory.
- It is used to generate different internal or external control signals required to perform the operation.

Pointer & Index register:

- These registers are of 16 bit.
- Stack pointer (SP) and Base pointer (BP) are the two pointer registers.
- Whereas the Source index (SI) and Destination index (DI) are the index registers.
- Stack pointer (SP) and base pointer (BP) are used to access data in the stack segment.
- Source Index (SI) and Destination Index (DI) are used in indexed addressing.

II) BUS INTERFACE UNIT (BIU):

The function of BIU is to:

a. Fetch the instruction or data from primary memory.
b. Read / Write of data from / to primary memory.
c. I/O of data from / to peripheral ports.
d. Address generation for memory reference.

BIU consists of following parts:

Instruction Queue:

- It is 6 bit long instruction register.
- To increase the execution speed, BIU fetches six instruction bytes ahead to time from memory.
- All six bytes are then held in first in- first out (FIFO) queue.
- Then all bytes have to be given to EU one by one.

Segment Registers:

- **Code Segment (CS):** The CS register is used for addressing a memory location in the Code Segment of the memory, where the executable program is stored.
- **Data Segment (DS):** The DS contains most data used by program. Data are accessed in the Data Segment by an offset address or the content of other register that holds the offset address.
- **Stack Segment (SS):** SS defined the area of memory used for the stack.
- **Extra Segment (ES):** ES is additional data segment that is used by some of the string to hold the destination data.

Instruction Pointer:

- Instruction pointer is of one bit.
- It is used to point the particular instruction which is fetched by instruction register.

CHAPTER-2: INSTRUCTION SET & PROGRAMMING

Q1) EXPLAIN I/O RELATED ADDRESSING MODE OF 8086.

ANS: [5M - DEC14]

1. Instruction consists of Opcode followed by Operands.
2. The Opcode specifies the operation to be performed.
3. Operand specifies where data is.
4. The data may be instruction or may be in the register or in the memory or in the I/O Port etc.
5. The different ways by which microprocessor generates operand address are called addressing mode.

TYPES:

Figure 2.1 shows Addressing Mode of 8086.

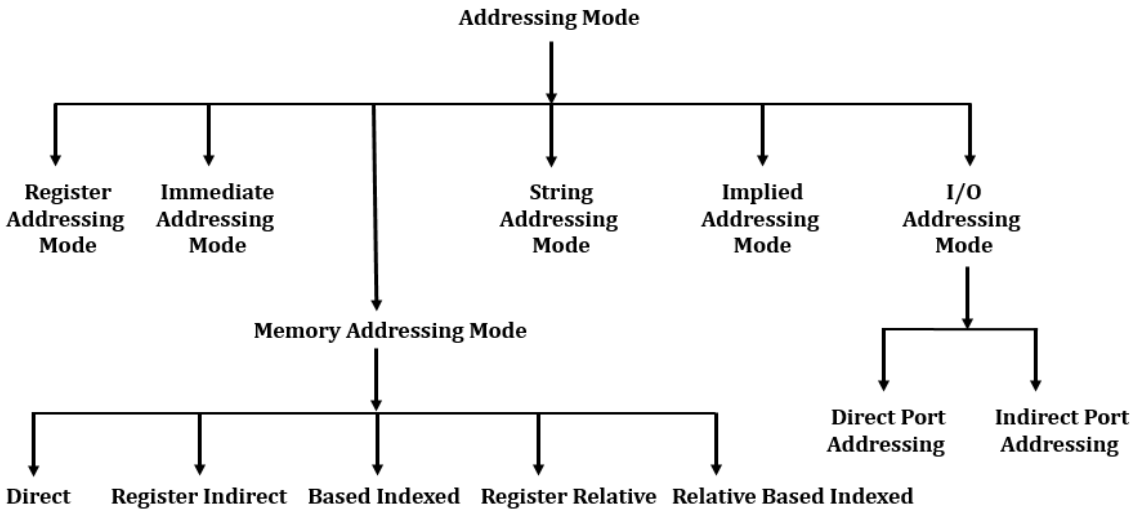


Figure 2.1: Addressing Modes of 8086.

I/O ADDRESSING MODE:

- I/O Addressing Mode is used for Inputs & Outputs.
- I/O Addressing Mode consists of Memory Mapped I/O & I/O Mapped I/O.

a. Memory Mapped I/O:

- In this mode, the memory is mapped to I/O port.
- In this any memory operand addressing modes are used to access the port.
- For example, a group of terminals can be accessed as an array.
- String instructions also can be used to transfer data to memory mapped I/O ports with in appropriate hardware interface.

b. I/O Mapped I/O:

- In this mode, the I/O is mapped to I/O port.
- There are two types: Direct Port Addressing & Indirect Port Addressing.

- **Direct Port Addressing:**
  - In this 8 bit port address is specified in the instruction itself.
  - Therefore it consists of 256 I/O Address i.e. ports numbered 0-255.
  - For **Example:** In AL, 04 H means move 8 bit data from Input Device having address 04 H into AL register.
  - In AX, 04 H means move 16 bit data from Input Device having address 04 H and 05 H into AL and AH register respectively.
- **Indirect Port addressing:**
  - It is similar to register indirect addressing of memory operands.
  - In this 16 bit I/O address is kept in DX register.
  - Therefore it consists of 65,536 I/O Address.
  - For **Example:** In AL, DX means move 8 bit data from the Input Device pointed by DX into AL register.
  - In AX, DX means move 16 bit data from the Input Device pointed by DX and DX + 1 into AX register.

Q2) Write assembly language program for 8086 to exchange contents of two memory blocks.

Ans: [10M - DEC14]

Assembly Language:

1. Machine Language is the only language understood directly by CPU in computers.
2. So the instructions in the text form are called as Mnemonic.
3. Assembly Language is a Mnemonic representation of machine code.
4. There is a one-to-one correlation between assembly language instructions and the machine code.

Program to exchange contents of two memory block:

1. Let consider the Memory Locations as: **01000 H (First Memory Block) & 02000 H (Second Memory Block)**
2. Let the block size which is to be exchange between two memory blocks be 1 KB i.e. 1024 byte.
3. The source block is at address 01000 H and destination block is at address 02000 H.
4. We will first copy 1 KB block starting at location 01000 H to another place at 03000 H onwards.
5. Then the block from 02000 H onwards is transferred to the first block i.e. starting block 01000 H.
6. Finally the block copied in location 03000 H onwards is transferred to the location 02000 H onwards.

Instruction	Comments
MOV AX, 0000 H	Move 0000 H to AX Register.
MOV DS, AX	
MOV ES, AX	
MOV SI, 1000 H	Move 1000 H to Source Index.
MOV DI, 3000 H	Move 3000 H to Destination Index.
MOV CX, 0400 H	
CLD	
REP MOVSB	
MOV SI, 2000 H	Move 2000 H to Source Index.
MOV DI, 1000 H	Move 1000 H to Destination Index.
MOV CX, 0400 H	
CLD	Clear Direction Flag.
REP MOVSB	
MOV SI, 3000 H	Move 3000 H to Source Index.
MOV DI, 2000 H	Move 2000 H to Destination Index.
MOV CX, 0400 H	
CLD	Clear Direction Flag.
REP MOVSB	

Therefore the content of blocks is exchange between 01000 H (First Memory Block) & 02000 H (Second Memory Block) using above 8086 Assembly Program.

Q3) WRITE ASSEMBLY LANGUAGE PROGRAM FOR 8086 TO REVERSE A STRING OF 10 CHARACTERS.

ANS: **[10M - DEC15]**

ASSEMBLY LANGUAGE:

1. Machine Language is the only language understood directly by CPU in computers.
2. So the instructions in the text form are called as Mnemonic.
3. Assembly Language is a Mnemonic representation of machine code.
4. There is a one-to-one correlation between assembly language instructions and the machine code.

FLOW CHART:

Figure 2.2 shows the flow chart to reverse a string using 8086 assembly language program.

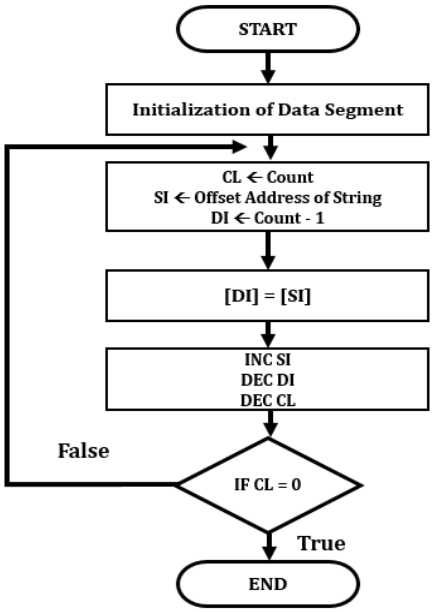


Figure 2.2: Flow chart to reverse a String.

PROGRAM TO REVERSE A STRING:

Let the string STR = "TECHNOLOGY"

Instruction	Comments
MOV AX, DATA	Move DATA to AX Register.
MOV DS, AX	Initialize DS
MOV CL, 10	Count is 10 Characters
MOV SI, OFFSET STR	
MOV DI, 9	Count – 1
REVERSE: MOV AL, [SI]	
XCHG [DI], AL	Exchange Contents of DI & AL
MOV [SI], AL	
INC SI	Increment Source Index.
DEC DI	Decrement Destination Index
DEC CL	
JNZ REVERSE	
HLT	

Q4) MIXED LANGUAGE PROGRAMMING.

ANS: [5M – MAY15]

1. C generates an Object Code that is Extremely Fast & Compact.
2. But it is not as fast as the Object Code generated by a good programmer using Assembly Language.
3. Since time needed to write a program in Assembly Language is much more than time taken in Higher Level Languages like C.
4. There are some cases where certain instructions cannot be executed in Higher Level Language like C.
5. For example: C does not have an instruction for performing bit-wise rotation operation.
6. Therefore combining multiple languages such as C & Assembly Language is known as Mixed Language Programming.
7. There are two ways of combining C & Assembly Language.

I) Method 1:

- In method 1 there is a Built-In-Inline assembler used to include assembly language routines in the C-program, without any need for a specific assembler.
- Such assembly language routines are called as In-Line Assembly.
- They are compiled along with C-routine & linked together using linked modules provided by the C Compiler.
- **Example:** Turbo C has inline Assembly.
- Table 2.1 shows the Mixed Language Programming Code Example for Method – 1.

Table 2.1: Mixed Language Programing Code.

```
#include<iostream.h>
void main()
{
      int a, b, c;
      cout <<  "Enter Two Numbers";
      cin >> a >> b;
      asm mov ax, a;
      asm mov bx, b;
      asm add ax, bx;
      asm mov c, ax;
      cout << "The Sum is" << c;

}
```

II) Method 2:

- There are times when programs written in one language have to call modules written in other language.
- Instead of compiling all source programs using the same compiler, different compilers or assemblers are used as per the language used in the program.

- Microsoft C support Mixed Language Programming.
- Therefore, it can combine assembly language routines in C as a separate language.
- C program calls assembly language routines that are separately assembled by MASM (MASM Assembler) or TASM (Turbo Assembler).
- These assembled modules are linked with the complied C modules to get the combine executable file.
- Figure 2.3 shows Compile, Assemble & Link processes using C Compiler, MASM Assembler & TLINK.

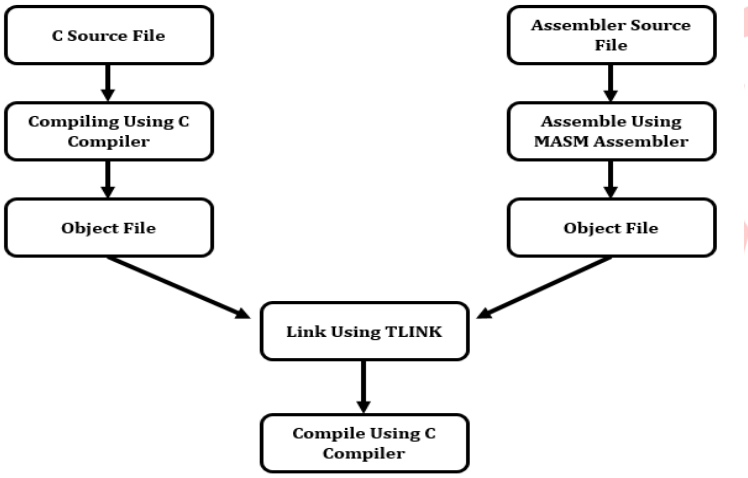


Figure 2.3: Compile, Assemble & Link processes in Mixed Language Programming.

Q5) EXPLAIN ADDRESSING MODES OF 8086

ANS: [10M - DEC14]

1. Instruction consists of Opcode followed by Operands.
2. The Opcode specifies the operation to be performed.
3. Operand specifies where data is.
4. The data may be instruction or may be in the register or in the memory or in the I/O Port etc.
5. The different ways by which microprocessor generates operand address are called addressing mode.

TYPES:

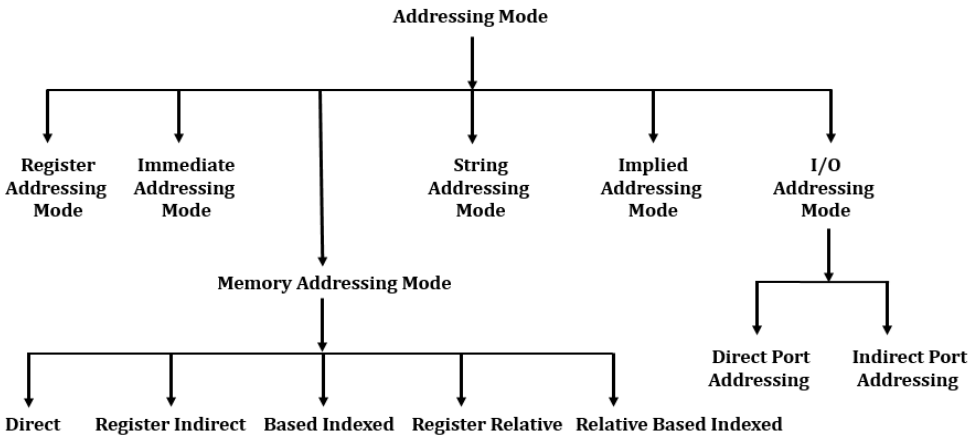


Figure 2.4: Addressing Modes of 8086.

I) REGISTER ADDRESSING MODE:

- In this mode the operands are specified using registers.
- The data is in the register, and the instruction specifies the particular register as shown in figure 2.5.
- In Register Addressing Mode, the instructions are compact & comparatively faster for execution.
- Register may be used as Source operands, Destination operands or both.
- The register may be 8/16 bit.
- Example: **MOV AX, BX**
- This instruction copies the contents of BX register to AX register.

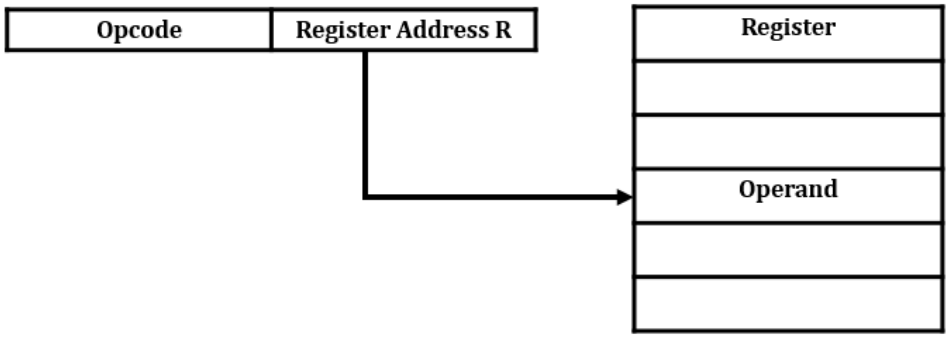


Figure 2.5: Register Addressing Mode.

II) IMMEDIATE ADDRESSING MODE:

- In this mode the operand is specified by instruction itself.
- Immediate operand is nothing but the constant data contained in an instruction.
- Thus if the source operand is part of instruction instead of register or memory, it is referred as Immediate Addressing Mode as shown in figure 2.6.
- Immediate Data may be 8/16 bit.
- Immediate operands are used as Source operands and they are constant values.
- Immediate operands are accessed quickly like Register Addressing Mode.
- Example: **MOV CL, 02H** **MOV CL, 2**
- This instruction copies the immediate number 2H in the CL Register.

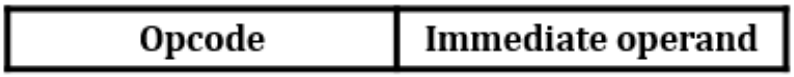


Figure 2.6: Instruction encoded with an Immediate Operand.

III) MEMORY ADDRESSING MODE:

- In Register & Immediate Addressing Mode, Execution Unit (EU) has direct access to Register & Immediate data.
- In Memory Addressing Mode, memory operands must be transferred to/from the CPU over the BUS.
- Whenever EU needs to read or write a memory operand, it must pass an offset value to the BIU.

- The BIU adds offset to the shifted contents of segment register, which produces 20 bit physical address.
- After that it executes the bus cycle needed to access the operand.
- The offset for a memory operand is called the operand's Effective Address.

Types:

a. Direct Memory Addressing Mode:

- In this mode, the 16 bit Effective Address is taken from the displacement field of the instruction.
- The physical address is generated by adding this to segment register * 10 H as shown in figure 2.7.

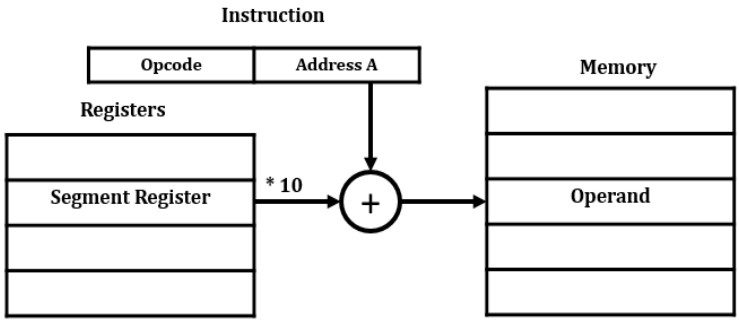


Figure 2.7: Direct Memory Addressing Mode.

b. Register Indirect Addressing Mode:

- In this mode, the Effective Address of the memory is taken directly from one of the base register or index register, specified by the instruction.
- The address is added with the segment register * 10 H to generate the physical address as shown in figure 2.8.

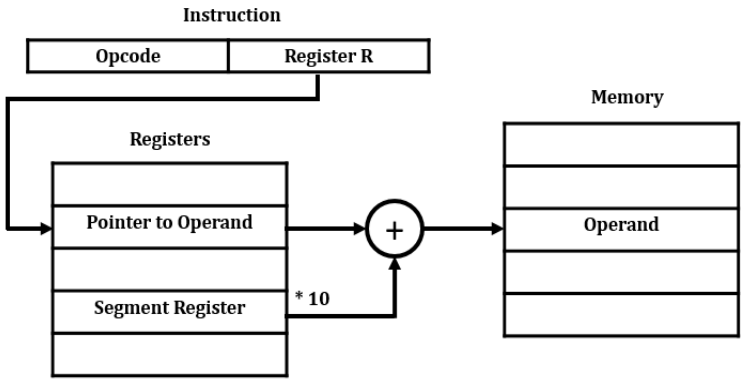


Figure 2.8: Register Indirect Addressing Mode.

c. Register Relative Addressing Mode:

- In this mode, the Effective Address is sum of an 8 or 16 bit displacement and the contents of base register or an index register are added.
- This sum is added with the segment register * 10 H to generate Effective Address as shown in figure 2.9.

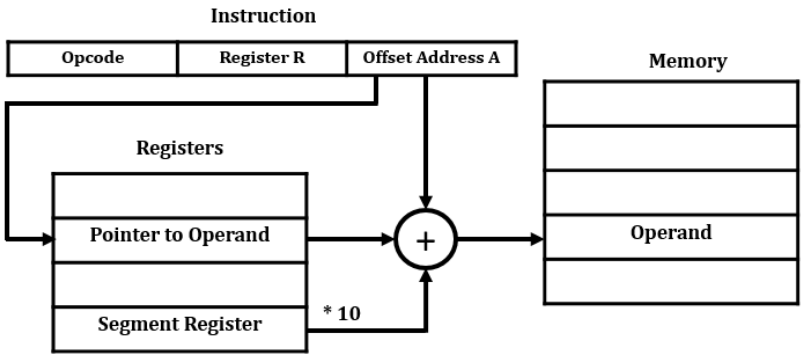


Figure 2.9: Register Relative Addressing Mode.

d. Based Indexed Addressing Mode:

- In this mode, the Effective Address is sum of a base register and an index register, both of which are specified by the instruction.
- The sum is added with the segment register * 10 H to generate Effective Address as shown in figure 2.10.

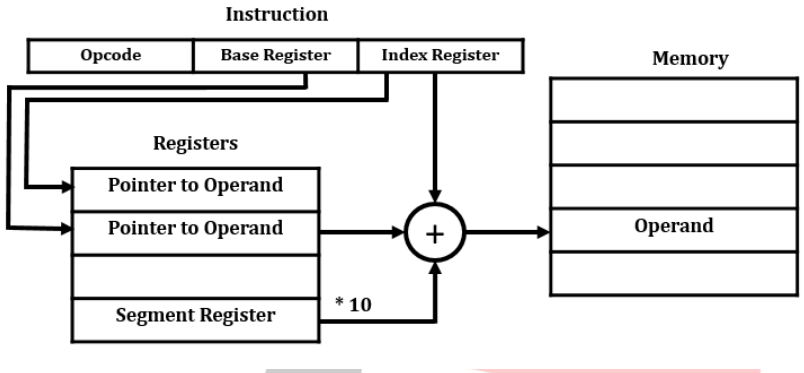


Figure 2.10: Based Indexed Addressing Mode.

e. Relative Based Indexed Addressing Mode:

- This Addressing Mode generates an Effective Address that is the sum of a base register, an index register and a displacement.
- This sum is added with the segment register * 10 H to generate Effective Address as shown in figure 2.11.

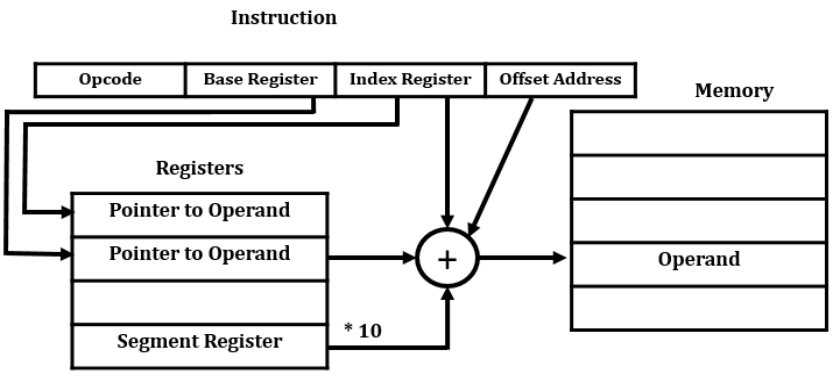


Figure 2.11: Relative Based Indexed Addressing Mode.

IV) STRING ADDRESSING MODE:

- String Addressing Mode does not use normal Memory Addressing Mode to access their operands.
- When a string instruction is executed, SI is assumed to point to the first byte or word of the source string and by default DS is assumed as Segment Register.
- DI will point to the first byte or word of the destination string and by default ES is assumed as Segment Register.
- In repeated string operation, the CPU automatically adjust the SI & DI to obtain subsequent bytes or word.
- This automatic adjustment is done with the help of Direct Flag (DF) in flag register.

V) I/O ADDRESSING MODE:

- I/O Addressing Mode is used for Inputs & Outputs.
- I/O Addressing Mode consists of Memory Mapped I/O & I/O Mapped I/O.
- **Memory Mapped I/O:**
  - In this mode, the memory is mapped to I/O port.
  - In this any memory operand addressing modes are used to access the port.
  - For example, a group of terminals can be accessed as an array.
  - String instructions also can be used to transfer data to memory mapped I/O ports with in appropriate hardware interface.
- **I/O Mapped I/O:**
  - In this mode, the I/O is mapped to I/O port.
  - There are two types:
    - **Direct Port Addressing:**
      - In this 8 bit port address is specified in the instruction itself.
      - Therefore it consists of 256 I/O Address i.e. ports numbered 0-255.
      - For **Example:** In AL, 04 H means move 8 bit data from Input Device having address 04 H into AL register.
      - In AX, 04 H means move 16 bit data from Input Device having address 04 H and 05 H into AL and AH register respectively.
    - **Indirect Port addressing:**
      - It is similar to register indirect addressing of memory operands.
      - In this 16 bit I/O address is kept in DX register.
      - Therefore it consists of 65,536 I/O Address.
      - For **Example:** In AL, DX means move 8 bit data from the Input Device pointed by DX into AL register.
      - In AX, DX means move 16 bit data from the Input Device pointed by DX and DX + 1 into AX register.

VI) IMPLIED ADDRESSING MODE:

- The instructions which do not have operands are considered as Implied Addressing Mode.
- For **Example:** STC (Sets the Carry Flag), CLD (Clears the Direction Flag), STD (sets the Direction Flag) etc.

CHAPTER-3: SYSTEM DESIGNING WITH 8086

Q1) DRAW & EXPLAIN BLOCK DIAGRAM OF 8259 PIC

Q2) 8259 – PIC.

ANS: [Q1 | 10M - DEC14] & [Q2 | 5M – MAY16]

1. For the application where we require multiple interrupt sources, we need to use an external device called as Programmable Interrupt Control (PIC).
2. By connecting a PIC to the microprocessor we can increase the interrupt handling capacity of the microprocessor.
3. 8259 is commonly used PIC.

FEATURES OF 8259 PIC:

1. PIC 8259 is a Programmable Interrupt Controller that can work with 8085, 8086, etc.
2. 8259 has flexible priority structure.
3. 8259 PIC is used to implement 8 level interrupt system.
4. While cascaded configuration of 1 master 8259 & 8 slave 8259s can handle up to 64 interrupt systems.
5. 8259 can handle edge as well as level triggered interrupts.
6. In 8259, interrupts can masked individually.
7. The vector address of interrupt is programmable.
8. Status of interrupts (pending, in-service, and masked) can be easily read by microprocessor.
9. 8259 does not requires clock signal.
10. It can also be used in buffered mode.

BLOCK DIAGRAM:

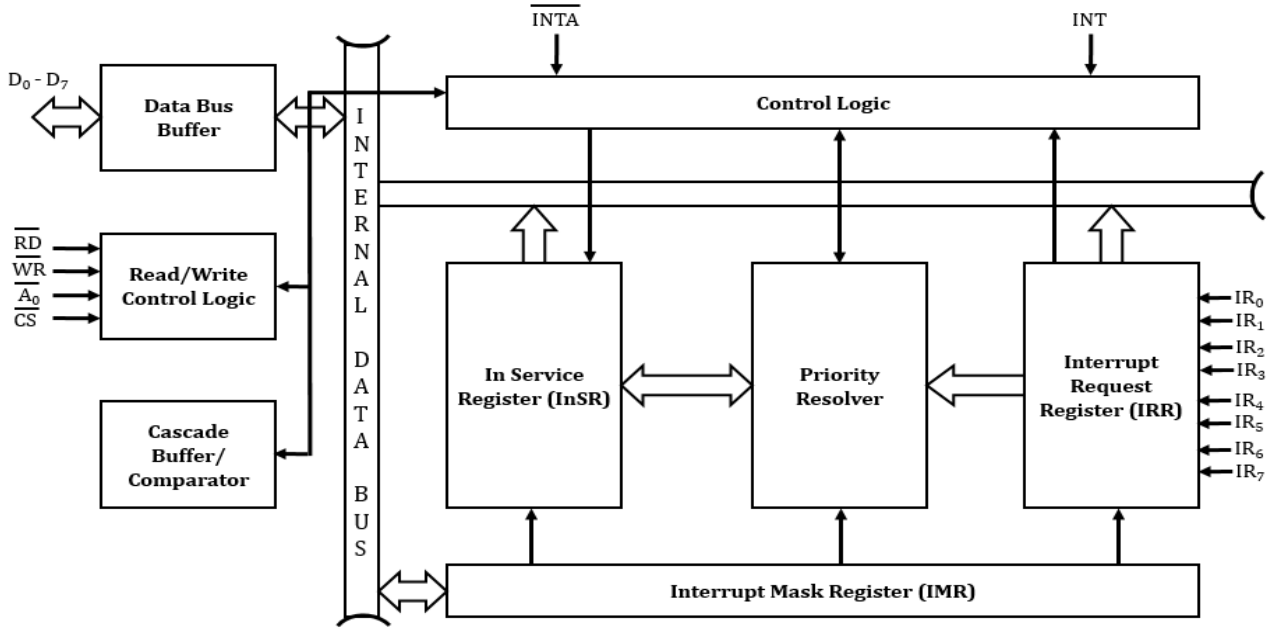


Figure 3.1: Block Diagram of 8259 PIC.

Block Diagram of 8259 PIC is shown in figure 3.1 & it contains following blocks:

I) Data Bus Buffer:

- It is 8 bit Bidirectional Buffer.
- It is used to transfer data between microprocessor & internal bus.

II) Read/Write Logic:

- It sets the direction of data bus buffer.
- It controls all internal read/write operations.
- It consists of Initialization Command Word Register (ICW) & Operation Command Word Register (OCW).

III) Cascade Buffer & Comparator:

- In master mode, it functions as a cascaded buffer.
- In slave mode, it functions as a comparator.
- In Buffered mode it generates an $\overline{EN}$ Signal.

IV) Control Logic:

- It has two signals: INT & $\overline{INTA}$
- **INT:** It is an Output Signal. It is connected to INTR of microprocessor. Whenever this line goes high microprocessor is interrupted.
- **$\overline{INTA}$:** It is an output signal to 8259. Whenever this line goes high microprocessor acknowledge the arrival of interrupt request to 8259.

V) Interrupt Request Register (IRR):

- It has 8 Input Lines. (IR_7 – IR_0)
- Peripheral Devices are connected to these lines.
- IRR is used to store all the pending interrupt requests.

VI) In-Service Register (InSR):

- It is used to store all the Levels which are being serviced.
- Microprocessor can read contents of this register by issuing appropriate command word.

VII) Interrupt Mask Register (IMR):

- It is a programmable register.
- It is used to mark some interrupt lines by selecting proper bits.
- Microprocessor can read contents of this register without issuing any command word.

VIII) Priority Resolver:

- Priority Register is used to examine IRR, InSR & IMR.
- It determines which INT should be sent out of IRR (IR_7 to IR_0) to microprocessor.

Q3) EXPLAIN INTERFACING OF 8259 WITH 8086 IN MINIMUM MODE.

ANS: [5M - DEC15]

*** NOTE: WRITE ANY ONE OUT OF THE FOLLOWING FOR 5 MARKS ***

INTERFACING 8259 WITH 8286 IN MINIMUM MODE: (8259 IN SINGLE MODE)

1. First 8259 is initialized by writing proper ICW's & OCW's.
2. INTR of 8086 is enabled by using the instruction STI.
3. When request appears on any one of IR inputs, then after resolving priority, 8259 makes INT high.
4. Now microprocessor is interrupted.
5. Microprocessor gives 2 $\overline{INTA}$ pulses to 8259.
6. During 1st $\overline{INTA}$ pulse 8259 calculates type number.
7. At the same time the corresponding bit in ISR is set.
8. During 2nd $\overline{INTA}$ pulse 8259 gives type number.
9. Now 8086 execute particular type ISR.
10. At the end of ISR, EOI command is executed.
11. Due to this, corresponding bit in ISR is reset & so 8259 can response to other low priority interrupts.
12. Figure 3.2 shows interfacing of 8259 with 8086 in minimum mode (8259 in single mode).

DIAGRAM:

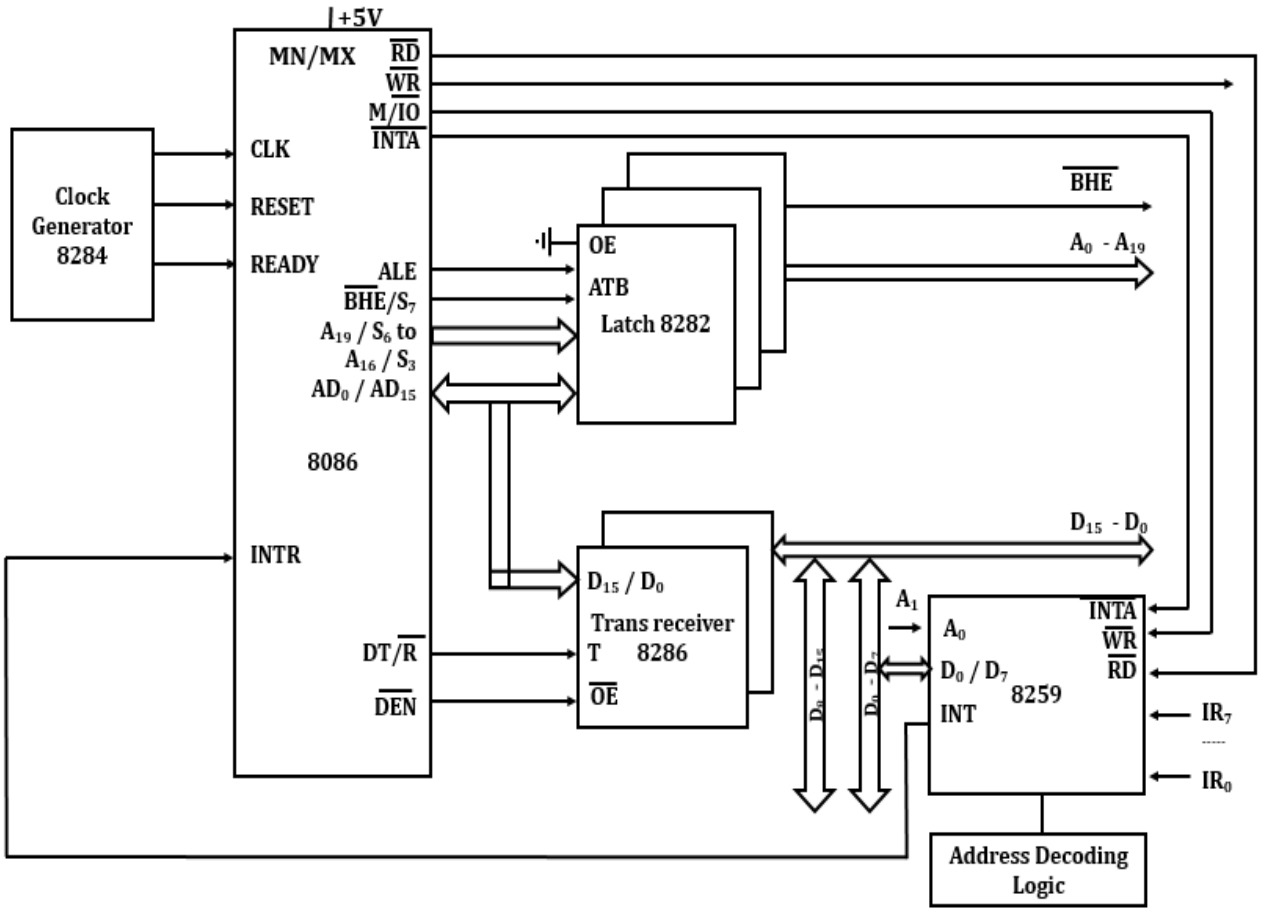


Figure 3.2: 8086 in Minimum Mode, 8259 in Single Mode.

INTERFACING 8259 WITH 8286 IN MINIMUM MODE: (8259 IN CASCADE MODE)

1. 8259 in cascade mode is preferred to increase number of interrupts more than 8 up to 64.
2. In this, three 8259 are used. One 8259 master and other two 8259 are known as slave-0, slave-1.
3. For Master, $\overline{SP}/\overline{EN}$ → +5V, For Slave, $\overline{SP}/\overline{EN}$ → +0V.
4. First all 8259's are initialized by writing proper ICW's & OCW's.
5. INTR of 8086 is enabled by using the instruction STI (now IF is set).
6. When request appears on any one of IR inputs of slave, then after resolving priority, slave makes INT high which is given at IR inputs of master.
7. After resolving priority, master makes INT high which is given to microprocessor.
8. Now microprocessor is interrupted.
9. Microprocessor gives 2 INTA pulses to 8259.
10. First INTA pulse is accepted only by master.
11. Now master gives slave identification number on CAS_2, CAS_1 & CAS_0.
12. The second INTA pulse is recognized only by that master for which its identification number matches with CAS_2, CAS_1 & CAS_0.
13. Now slave gives type number to 8086.
14. Now 8086 execute ISR.
15. At the end of ISR, 2 EOI command is executed.
16. One EOI command for master & another for slave to clear corresponding bit in ISR of master & slave.
17. Due to this, master & slave can respond to other low priority interrupts.
18. Figure 3.3 shows interfacing of 8259 with 8086 in minimum mode (8259 in cascade mode).

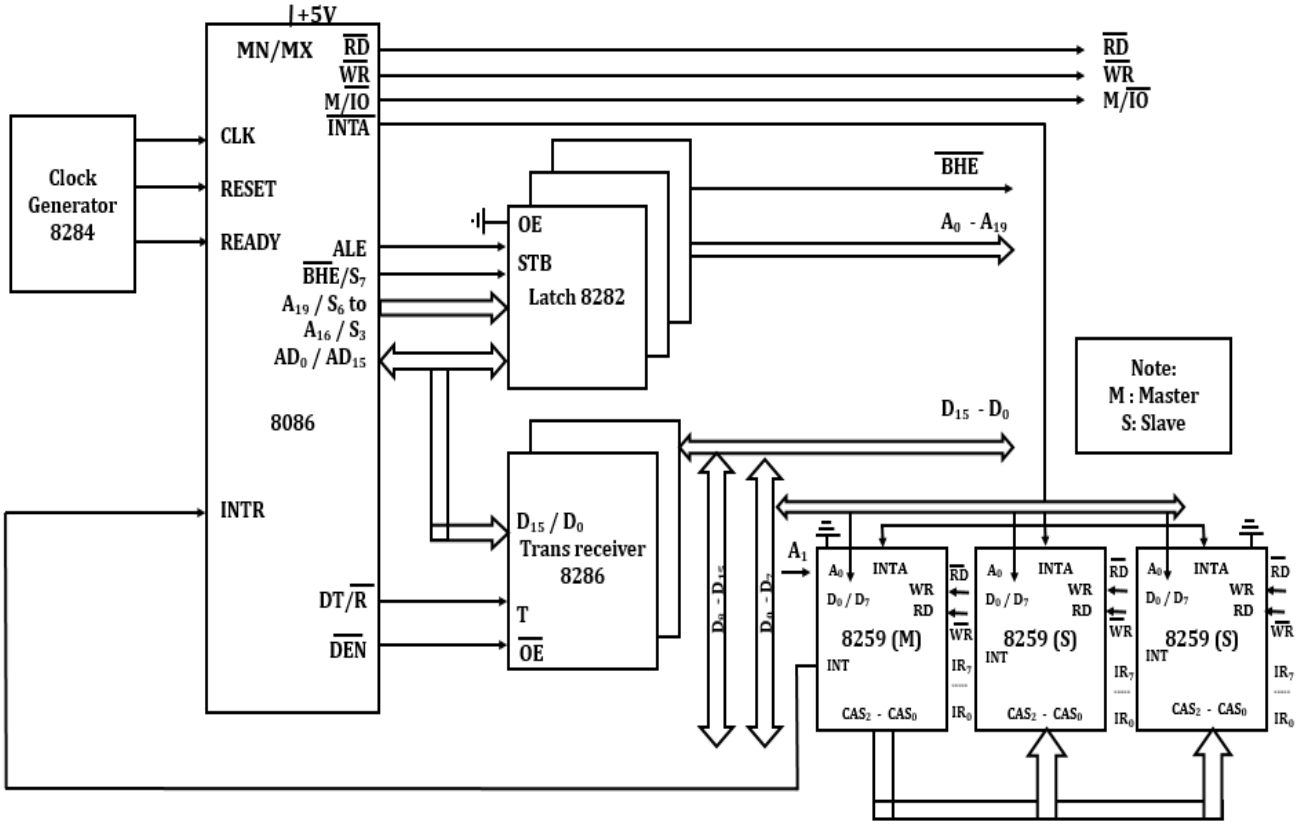


Figure 3.3: 8086 in Minimum Mode, 8259 in Cascade Mode.

Q4) EXPLAIN WITH BLOCK DIAGRAM WORKING OF 8255 PPI.

Q5) DRAW AND EXPLAIN THE BLOCK DIAGRAM OF 8255. ALSO EXPLAIN DIFFERENT OPERATING MODES OF 8255.

ANS: [Q4 | 10M - MAY15] & [Q5 | 12M – MAY16]

***** Note: For Operating Modes of 8255 Refer Q6 *****

1. 8255 is a Programmable Peripheral Interface i.e. PPI 8255.
2. It is general purpose programmable parallel I/O Device.
3. It has 3 8-bit bidirectional I/O Ports: Port A, Port B & Port C.
4. These I/O ports can be programmed in different modes.

FEATURES:

1. 8255 PPI contains 24 programmable I/O pins arranged as 2 8-bit ports & 2 4-bit ports.
2. 8255 PPI contains 3 ports and they are arranged in two groups of 12 pins.
3. It is fully compatible with Intel Microprocessor families.
4. It is also TTL compatible.
5. It has improved DC driving capability.
6. 8255 can operates in 3 modes:
   a. **Mode 0:** Simple I/O.
   b. **Mode 1:** Strobed I/O.
   c. **Mode 2:** Strobed bidirectional I/O.

BLOCK DIAGRAM:

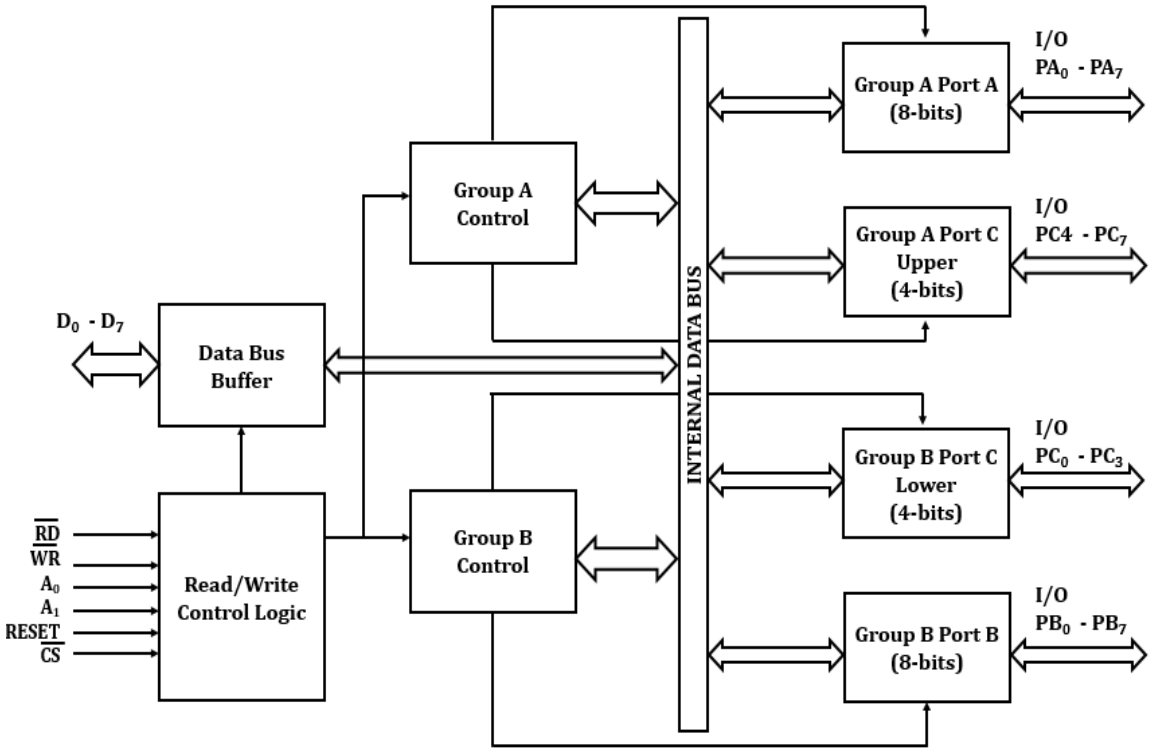


Figure 3.4: 8255 PPI Block Diagram.

It contains the following blocks:

I) Data Bus Buffer:

- This is 8-bit bidirectional buffer.
- It is used to interface the internal data bus of 8255 with the external data bus.
- When read is activated, it transmits data to the system data bus.
- When write is activated, it receives data from the system data bus.

II) Read/Write Control Logic:

- It accepts address and control signals from the microprocessor.
- The control signals are read & write while address signals used are A_0 & A_1 and $\overline{CS}$.
- The address bits (A_1, A_0) are used to select the ports or the control word register as shown:

For 8255		For 8086		Selection	Sample Address
A_1	A_0	A_2	A_1		
0	0	0	0	Port A	80 H (i.e. 1000 0000)
0	1	0	1	Port B	82 H (i.e. 1000 0010)
1	0	1	0	Port C	84 H (i.e. 1000 0100)
1	1	1	1	Control Word	86 H (i.e. 1000 1000)

- $\overline{CS}$ is connected to address chip select decoder.
- The 8255 operation/selection is enabled/disabled by CS signal.

III) Group A & Group B Control:

- 8255 I/O ports are divided into 2 sections: Group A & Group B
- Group A control block controls Port A & Port C upper i.e. $PC_7 – PC_4$.
- Group B control block controls Port B & Port C lower i.e. $PC_3 – PC_0$.
- Each group is programmed through software.
- Group A & Group B controls accepts the control signals from the control word and forwards them to the respective ports.

IV) Port A, Port B & Port C:

- This are 8-bit bidirectional ports.
- They can be programmed to work in the various modes as follows:

Port	Mode 0	Mode 1	Mode 2
Port A	Yes	Yes	Yes
Port B	Yes	Yes	No (Mode 0 or Mode 1)
Port C	Yes	No (Handshake signals)	No (Handshake signals)

- Group A control block controls Port A & Port C upper i.e. $PC_7 – PC_4$.
- Only Port C can also be programmed to work in Bit Set/Reset Mode to manipulate its individual bits.
- Port C function is dependent on mode of operation.
- It can be used as Simple I/O, Handshake Signals & Status Signal Input.

Q6) CONTROL WORD REGISTER OF 8255.

ANS: [5M - DEC14]

Control Word Register defines the function of each I/O Port and in which mode they should operate.

CONTROL WORD OF 8255: (I/O MODE)

1. To perform 8-bit data transfer using Ports A, B & C, 8255 needs to be in the I/O Mode.
2. The bit pattern for the control word in the I/O Mode is as follows:
3. Figure 3.5 shows the bit pattern of control word in 8255 I/O Mode.

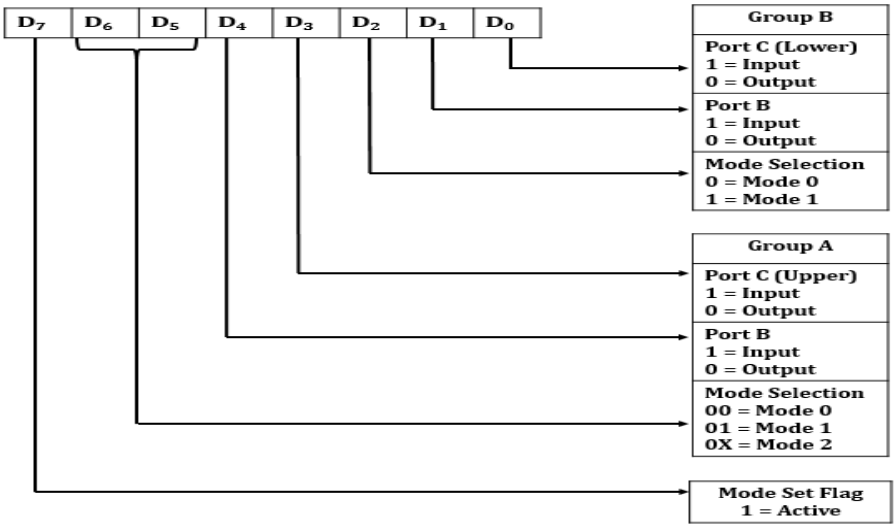


Figure 3.5: Bit pattern of control word in 8255 I/O Mode.

CONTROL WORD OF 8255: (BSR MODE)

1. The BSR Mode is used only for Port C.
2. In this mode the individual bits of Port C can be set or reset.
3. This is very useful as it provides 8 individually controllable lines which can be used while interfacing with devices like an A to D Converter or a 7-segment display etc.
4. The individual bit is selected and set/reset through the control word.
5. Since the D7 bit of the control word is 0, the BSR operation will not affect the I/O operations of 8255.
6. Figure 3.6 shows bit pattern for the control word in the BSR Mode.

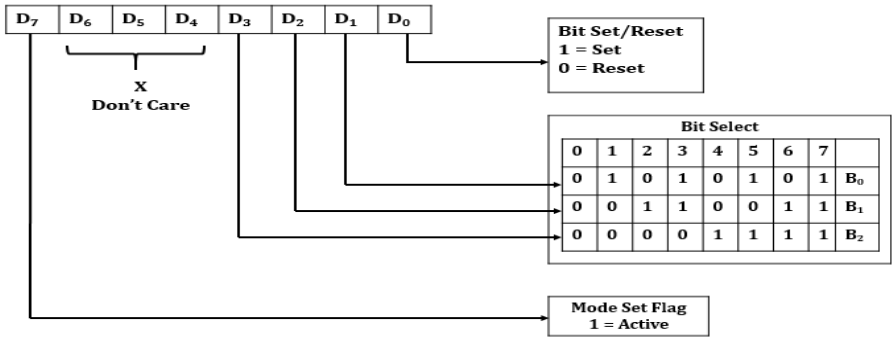


Figure 3.6: Bit pattern of control word in 8255 BSR Mode.

Q7) MODE 1 OF 8255 FOR INPUT OPERATION.

ANS: [5M - DEC15]

8255 Operates in Three I/O Modes: Mode 0, Mode 1 & Mode 2.

MODE 1:

1. In this Port A & Port B are provided 3 signals i.e. 2 Handshake Signal & 1 Interrupt Signal.
2. These signal are provided by Port C.
3. The remaining 2 lines of Port C can be used for Simple I/O function.
4. There are two case: Input Mode & Output Mode.

I) Input Mode:

- In Mode 1 Input Mode Port A uses PC_3, PC_4, PC_5 and Port B uses PC_0, PC_1, PC_2 signals as handshake signals.
- Port A, Port B and handshake signals are interfaced with peripheral.
- The aim of interfacing is to transfer data from peripheral to CPU through 8255.
- The different handshake signals used are $\overline{STB}$, IBF & INTR.
- Figure 3.7 shows P_A, P_B & P_C in Mode 1 output mode.

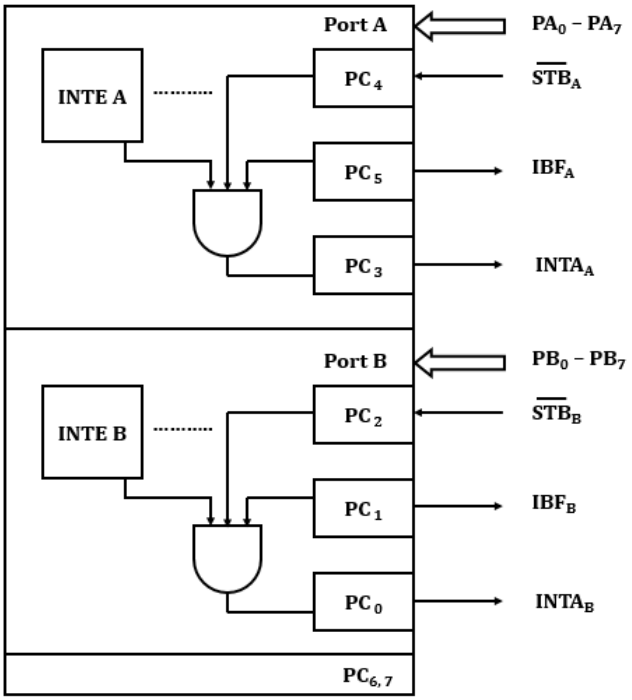


Figure 3.7: P_A, P_B & P_c in Mode 1 Input Mode.

Timing Diagram of Mode 1 Input Mode:

- Figure 3.8 shows timing diagram of mode 1 input mode.
- $\overline{STB}$ & IBF are handshake signals. Now the following action takes place.
- Peripheral devices places data in the port bus and inform 8255 making $\overline{STB}$ low.
- Now input port accepts the data.

- After that IBF goes high and then STB also goes high.
- Now INTR goes high if internal INTE F/F is enabled.
- Now microprocessor is interrupted to read data or microprocessor can read data by checking Port C.
- Once microprocessor reads data IBF goes low.
- Now Peripheral devices places next data.

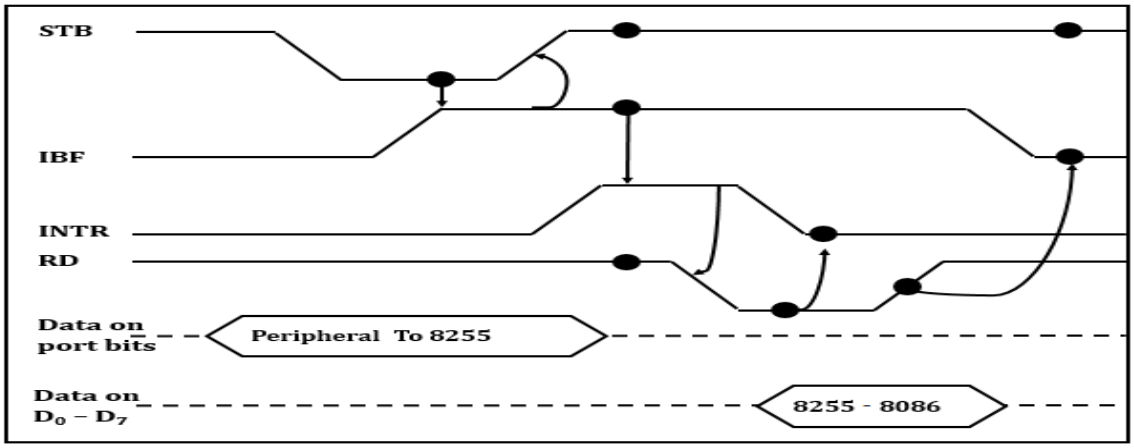


Figure 3.8: Timing Diagram of Mode 1 Input Mode.

II) Output Mode:

- In Mode 1 Output Mode Port A uses PC_3, PC_6, PC_7 and Port B uses PC_0, PC_1, PC_2 signals as handshake signals.
- Port A, Port B and handshake signals are interfaced with peripheral.
- The aim of interfacing is to transfer data from peripheral to CPU through 8255.
- The different handshake signals used are $\overline{OBF}$, $\overline{ACK}$ & INTR.
- Figure 3.9 shows P_A, P_B & P_C in Mode 1 output mode.

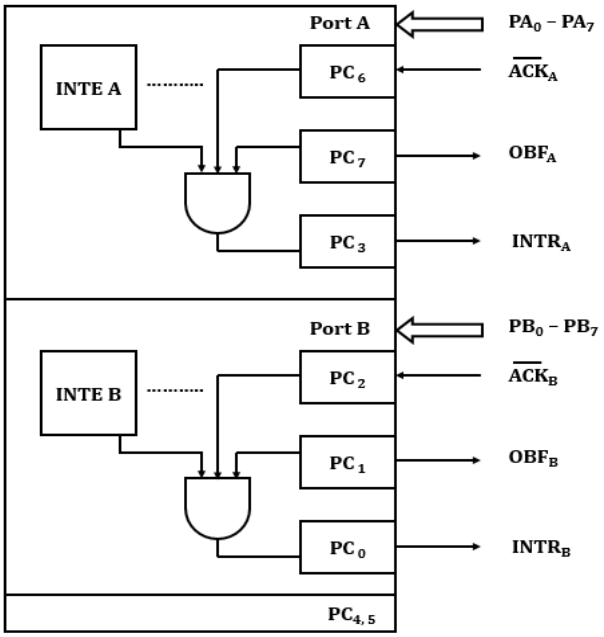


Figure 3.9: P_A, P_B & P_c in Mode 1 Output Mode.

Timing Diagram of Mode 1 Output Mode:

- Figure 3.10 shows timing diagram of mode 1 output mode.
- ACK & OBF are handshake signals. Now the following action takes place.
- When Output Port is empty, microprocessor writes data in the port.
- Now OBF goes low to indicate peripheral that data is available.
- Peripheral devices accepts the data by making ACK low.
- Now OBF goes high and then ACK goes high.
- Now INTR goes high, if internal INTE F/F is set.
- Now microprocessor is interrupted to write next data or microprocessor can write next data by checking Port C.
- Now microprocessor writes next data.

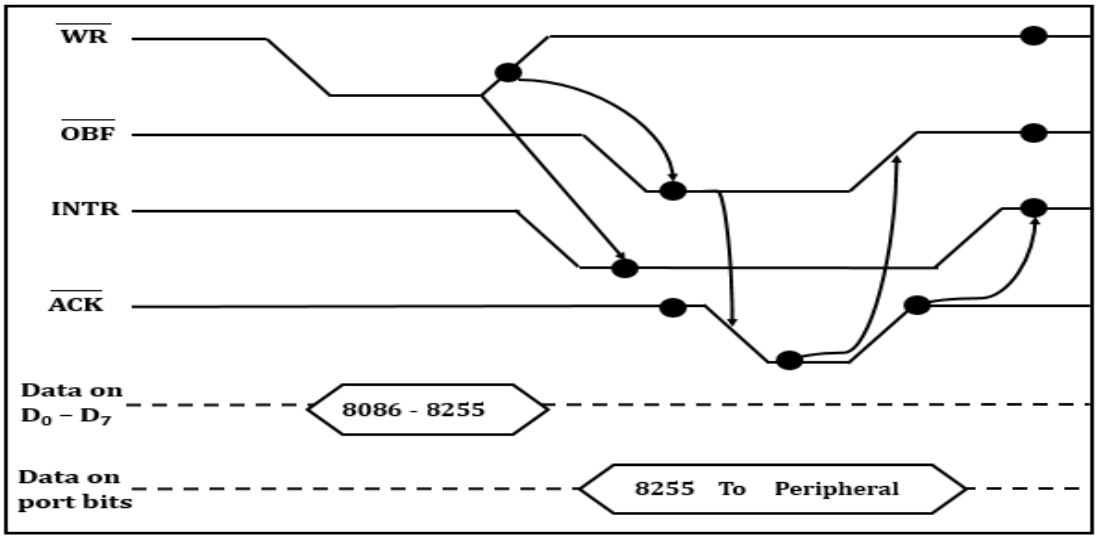


Figure 3.10: Timing Diagram of Mode 1 Output Mode.

Q8) LIST OPERATING MODES OF 8253.

ANS: [5M - DEC15]

*** NOTE: ANSWER BELOW IS WRITTEN FOR 10 MARKS. FOR 5 MARKS MAKE IT SHORT. ***

OPERATING MODES OF 8253/8254:

I) Mode 0: Interrupt on Terminal Count.

- When this mode is selected OUT Pin is initially low.
- The count value is loaded.
- GATE Pin is made high, so counting is enabled.
- During Counting, OUT Pin remains low.
- On Terminal Count (TC) the OUT Pin goes high, and remains high.
- During Counting if GATE is made low, it disables counting.
- When the GATE is made high, the counting resumes.
- Figure 3.11 shows the Mode 0 Timing Diagram.

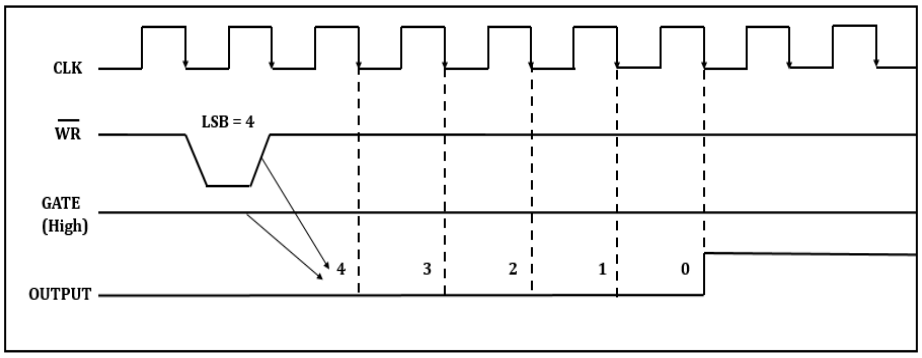


Figure 3.11: Mode 0 Timing Diagram.

II) Mode 1: Monostable Multivibrator.

- When this mode is selected OUT Pin is initially high.
- The count value is loaded.
- Counting begins only when a rising edge is applied to the GATE.
- OUT Pin goes low and remains low during counting.
- On Terminal Count (TC) the OUT Pin goes high, and remains high.
- During Counting if GATE is made low, it has no effect on counting.
- The GATE Pin can be used as Trigger.
- The counter can be re-triggered by applying a rising edge on the GATE.
- This would restart the counting, and hence re-trigger it.
- Figure 3.12 shows the Mode 1 Timing Diagram.

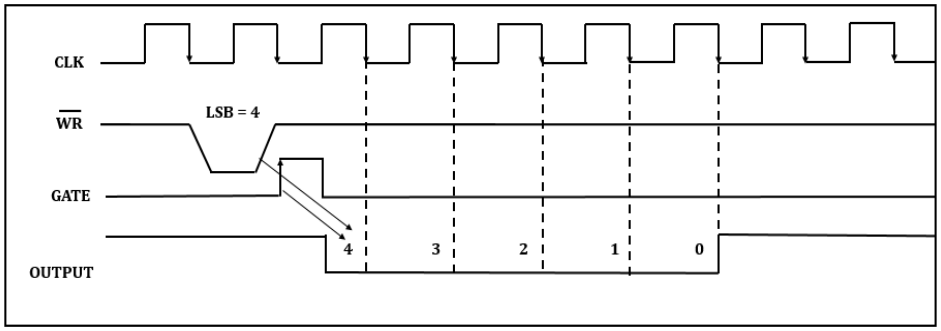


Figure 3.12: Mode 1 Timing Diagram.

III) Mode 2: Rate Generator.

- When this mode is selected OUT Pin is initially high.
- The count value is loaded.
- GATE Pin is made high, so counting is enabled.
- During Counting, OUT Pin remains high.
- The OUT Pin goes low for one clock cycle just before the TC.
- The initial count is reloaded and above process repeats.
- Thus this mode produces a Continuous pulse.
- During Counting if GATE is made low, it disables counting.
- When the GATE is made high, the counting restarts.
- Figure 3.13 shows the Mode 2 Timing Diagram.

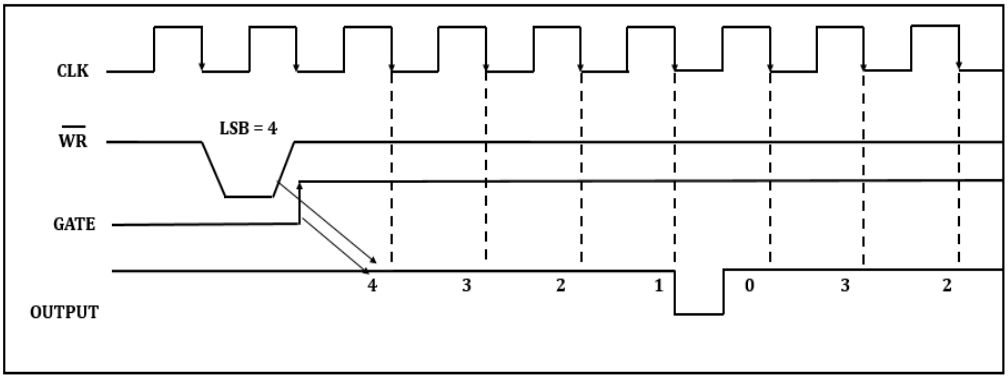


Figure 3.13: Mode 2 Timing Diagram.

IV) Mode 3: Square Wave Generator.

- When this mode is selected OUT Pin is initially high.
- The count value is loaded.
- GATE Pin is made high, so counting is enabled.
- The OUT Pin remains high for half of the count (n/2) and remains low for the remaining half.
- On TC the count is reloaded and the process repeat itself producing a continuous square wave.
- During Counting if GATE is made low, it disables counting.
- When the GATE is made high, the counting restarts.
- If the count is odd, the OUT Pin remains high for (n+1)/2 and low for (n-1)/2.
- Figure 3.14 shows the Mode 3 Timing Diagram.

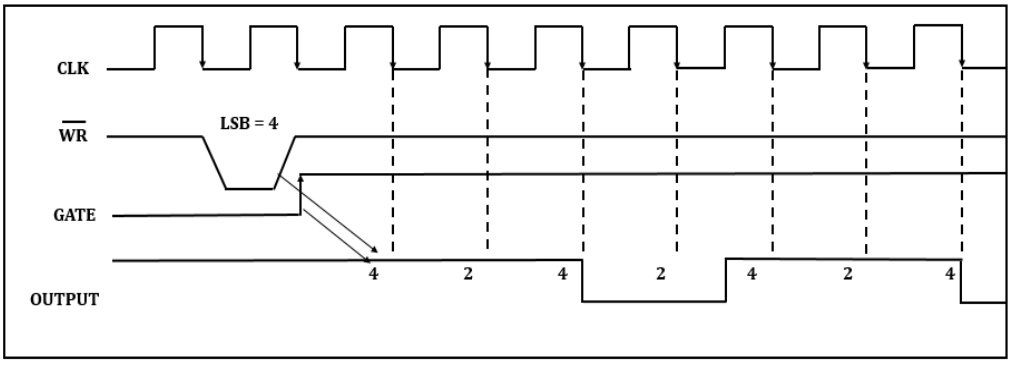


Figure 3.14: Mode 3 Timing Diagram.

V) Mode 4: Software Triggered Strobe.

- When this mode is selected OUT Pin is initially high.
- The count value is loaded.
- GATE Pin is made high, so counting is enabled.
- During Counting, OUT Pin remains high.
- The OUT Pin goes low for one clock cycle just before the TC.
- After that OUT Pin goes high & remains high.
- During Counting if GATE is made low, it disables counting.
- When the GATE is made high, the counting restarts.
- Figure 3.15 shows the Mode 4 Timing Diagram.

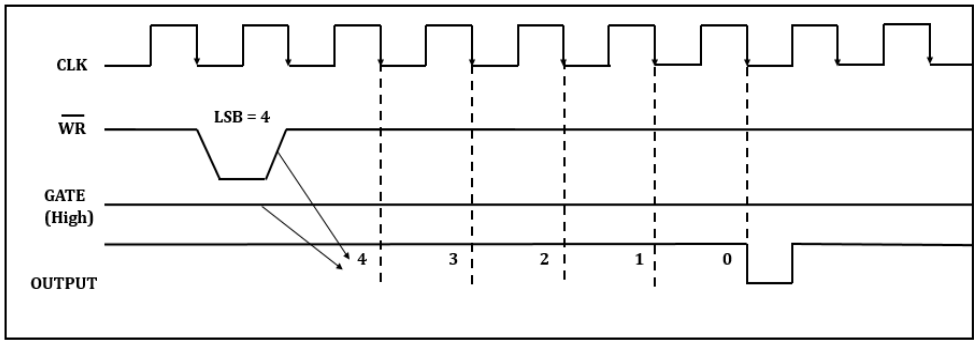


Figure 3.15: Mode 4 Timing Diagram.

VI) Mode 5: Hardware Triggered Strobe.

- When this mode is selected OUT Pin is initially high.
- The count value is loaded.
- Counting starts only after a trigger is applied to the GATE Pin.
- Also, the GATE Pin need not remain high for the counting to continue.
- During Counting, OUT Pin remains high.
- The OUT Pin goes low for one clock cycle just before the TC.
- After that OUT Pin goes high & remains high.
- Thus GATE is used as a Trigger i.e. it has to be triggered to start counting.
- When the GATE is made high, the counting restarts.
- Figure 3.16 shows the Mode 5 Timing Diagram.

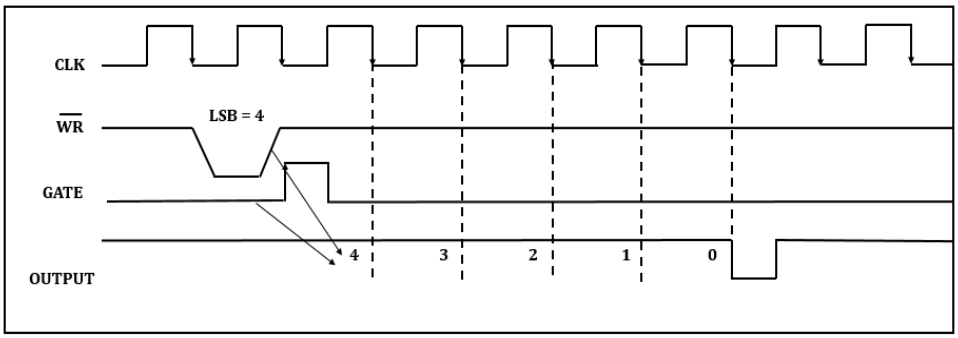


Figure 3.16: Mode 5 Timing Diagram.

Q9) DESIGN INTERFACING OF 8282 LATCHES TO 8086 SYSTEM.

ANS: [5M - MAY15]

1. A latch is a register with a specific purpose.
2. 8282/8283 are 8 bit bipolar latches with tristate output buffer.
3. 8282 is a non-inverting latch, while 8283 is an inverting octal latch.
4. 8282/8283 operates on single polarity supply.
5. It has high output drive capability.

INTERFACING OF 8282 LATCHES TO 8086 SYSTEM:

1. Interfacing of the latch 8282 with the 8086 processor is shown in figure 3.19.
2. The ALE signal is being used as the strobe (STB) input for the latches, which enables the latches and loads the address into the latches.
3. Since each latch is an octal, we have to use 3 such latches in order to latch a 20 bit address.
4. On receiving the HIGH (1) ALE signal from the processor, the latches will be enabled and the A_0 to A_{19} address is latched.
5. Before the address disappears from the multiplexed bus, the ALE reduces to 0 and the latches are disabled.

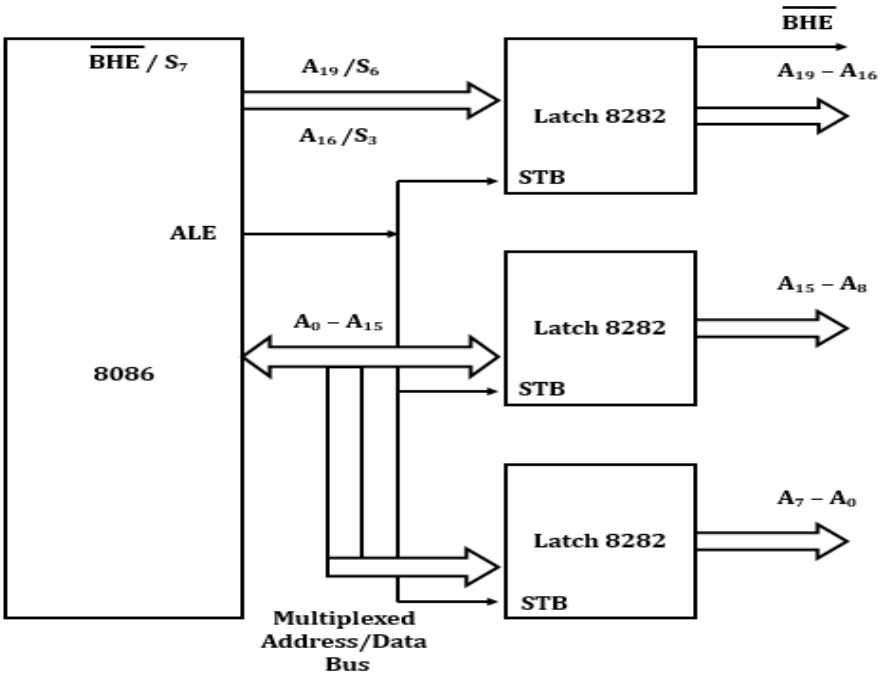


Figure 3.17: Interfacing of the latch 8282 with the 8086 processor.

Q10) INTERFACE DMA CONTROLLER 8237 WITH 8086 MICROPROCESSOR. EXPLAIN DIFFERENT DATA TRANSFER MODES OF 8237 DMA CONTROLLER.

Q11) EXPLAIN DIFFERENT DATA TRANSFER MODES OF 8237 DMA CONTROLLER.

ANS: [Q10 | 10M – DEC14] & [Q11 | 10M – MAY15 & 5M - DEC15]

1. Usually Data Transfer takes place between Microprocessor & Peripheral Device by using a program stored in a memory.
2. It is known as Microprocessor Controlled Data Transfer.
3. This method is used when the speed of Peripheral is less than or equal to the speed of Microprocessor.
4. If the speed of Peripheral is greater than the speed of microprocessor then microprocessor is disconnected from the system and DMA Controller is used to transfer the data between peripheral devices.
5. Now it is known as Device Controlled Data Transfer.

6. This method does not requires software and so data transfer takes place at high speed.
7. For example: Data transfer between system memory and floppy disk.

INTERFACING DMA CONTROLLER 8237 WITH 8086:

1. Address generated by 8237 is only 16 bit.
2. $A_0 – A_3$ & $A_4 – A_7$ is directly connected.
3. While $A_8 – A_{15}$ is de-multiplexed from $DB_0 – DB_7$.
4. The upper 4 bit address i.e. $A_{16} – A_{19}$ is provided through a latch.
5. This latch is to be written on by the programmers.
6. When 8237 issues the address, corresponding latches are enabled and similarly for 8086 issuing the address.
7. This is controlled by AEN Pin of 8237.
8. Figure 3.18 shows Interfacing of 8237 with 8086.

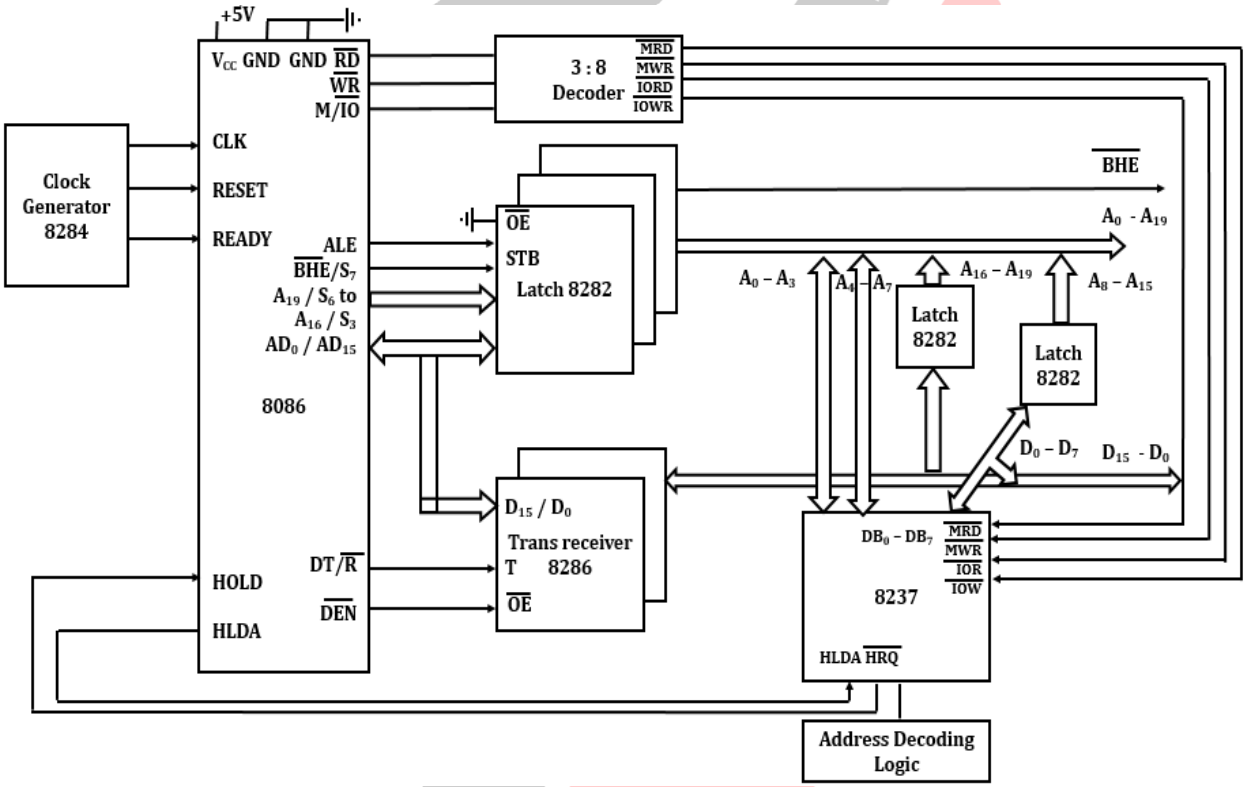


Figure 3.18: Interfacing of 8237 with 8086.

DATA TRANSFER MODES OF 8237:

I) Single Transfer Mode:

- In this mode, 8237 is programmed to make only one data transfer.
- After one data transfer, it decrements the word count register & increments or decrements the address register.
- After transferring one byte the 8237 disables HRQ and enters into idle state or slave mode.
- It is also called as **Cycle Steal Mode**.
- Then buses are returned to microprocessor.
- If we want other data transfer then DREQ has to go high again.

II) Block Transfer Mode:

- In this mode, all the bytes are transferred continuously.
- After each transfer, it decrements the word count register & increments or decrements the address register.
- It maintains HRQ high during all DMA Cycles.
- The action continuous until TC is reached or EOP is activated.
- In this DREQ has to be kept high only at the beginning.

III) Demand Transfer Mode:

- It is very similar to block transfer, except that the DREQ must active throughout the DMA operation.
- In this mode, the number of bytes to be transferred is controlled by I/O device.
- If during the operation DREQ gets low, the DMA operation is stopped and the busses are returned to the microprocessor.
- In meantime, the microprocessor can continue with its own operations.
- Once DREQ goes high again, the DMA operation continues from where it had stopped.

IV) Cascade Mode:

- This mode more than one DMACs are cascaded together.
- It is used to increase the number of devices interfaced to the microprocessor.
- Here we have one master DMAC, to which one or more slave DMACs are connected.
- The slave gives HRQ to the master on the DREQ of the master, and the master gives HRQ to the microprocessor on the HOLD of the microprocessor.
- Figure 3.19 shows the cascaded 8237's.

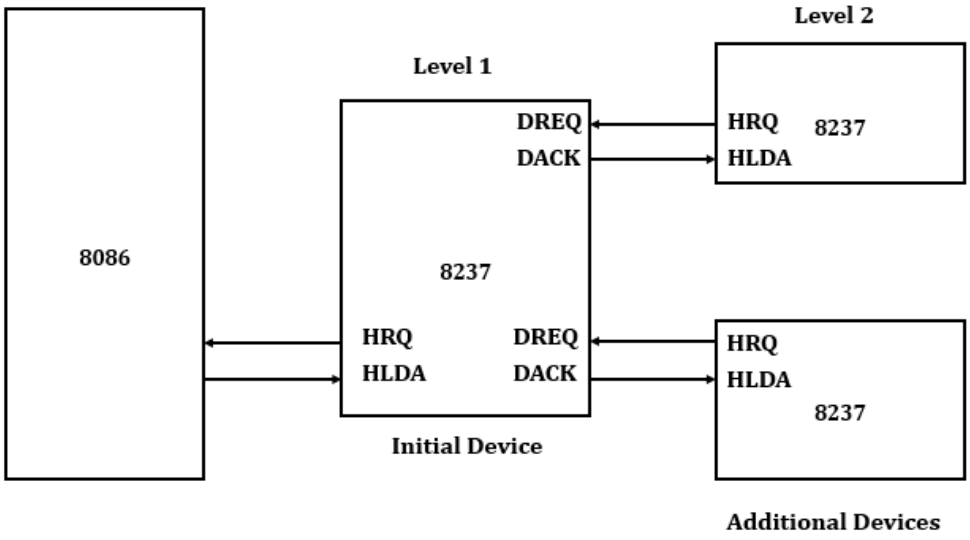


Figure 3.19: Cascaded 8237's.

Q12) 8087 MATH CO-PROCESSOR.

ANS: [5M - MAY15]

FEATURES:

1. It is a high performance numeric co-processor.
2. It can work on integer, decimal and real type numbers.
3. It has an instruction set capable of doing complex arithmetic and trigonometric calculations.
4. It follows IEEE Floating Point Standard.

BLOCK DIAGRAM:

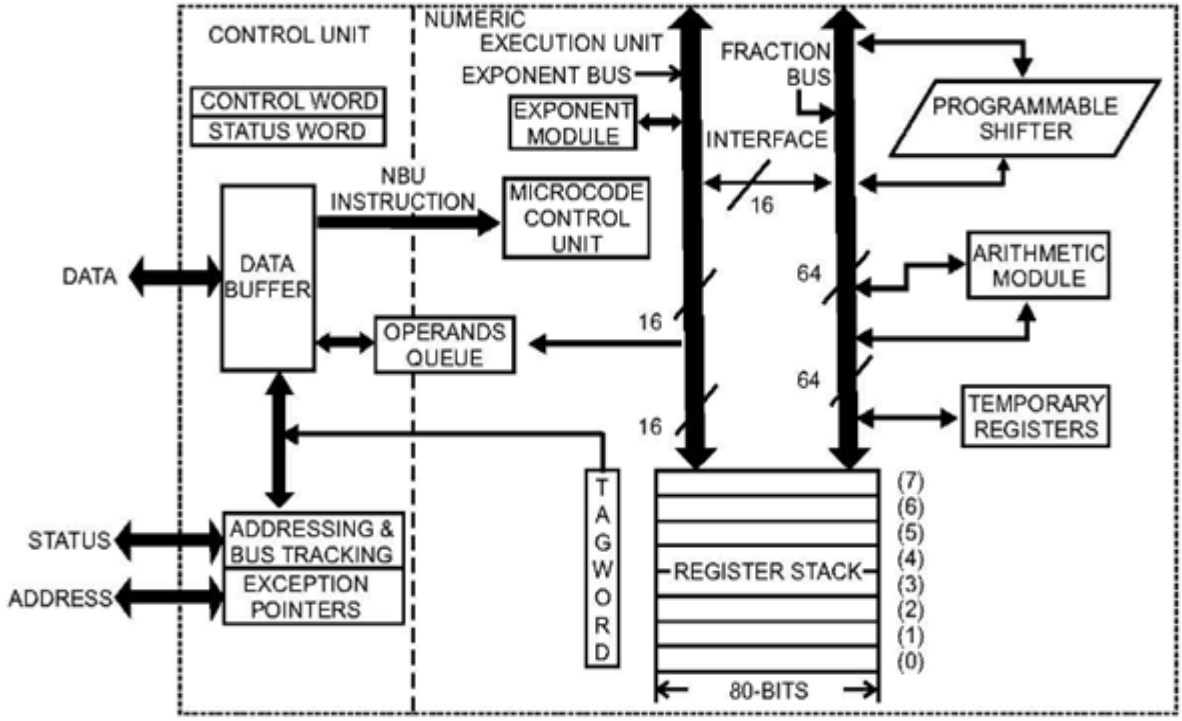


Figure 3.20: Block Diagram of 8087 Math Coprocessor.

It consists of following blocks:

I) Control Unit:

- It receives all instructions for the NDP.
- Instructions involving the register stack are given to the NEU, while the control unit executes the remaining instructions.
- Its main components are:
  a. Instruction Queue.
  b. Control Word.
  c. Status Word.
  d. Instruction Pointer.
  e. Data Pointer.

II) Numeric Execution Unit:

- Instructions involving the register stack are executed by the NEU.
- It uses operands from register stack.
- During arithmetic operations, 8087 handles exponents and mantissas separately.
- **Example:** For addition, it will first compare and equalize the exponents by shifting, and only then add the mantissa.
- Its main components are:
  a. Register Stack.
  b. Tag Word.

Q13) 8089 I/O PROCESSOR

ANS: [5M – MAY16]

1. 8089 is an I/O processor.
2. It was available for use with the 8086/8088 central processor.
3. It uses the same programming technique as 8087 for I/O Operations, such as transfer of data from memory to a peripheral device.

FEATURES:

1. 8089 has very high speed DMA capability.
2. It has 1 MB address capability.
3. It is compatible with iAPX 86, 88.
4. It supports local mode and remote mode I/O processing.
5. 8089 allows mixed interface of 8-and 16-bit peripherals, to 8-and 16-bit processor buses.
6. It supports two I/O channels.
7. Multibus compatible system interface.
8. Memory based communications with CPU.

BLOCK DIAGRAM:

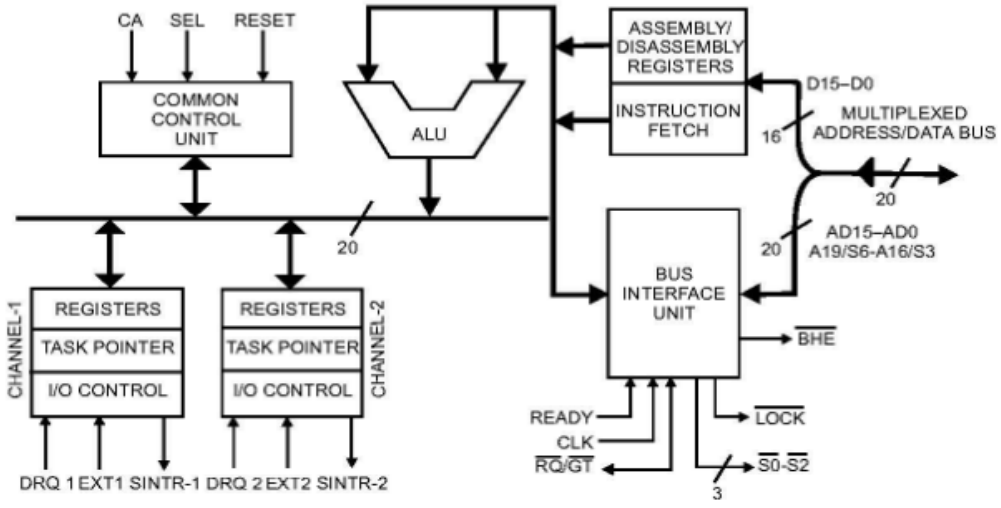


Figure 3.21: 8089 I/O Processor Block Diagram.

I) Common Control Unit (CCU):

- 8089 I/O Processor has two channels.
- The activities of these two channels are controlled by CCU.
- CCU determines which channel—1 or 2 will execute the next cycle.
- In a particular case where both the channels have equal priority, an interleave procedure is adopted in which each alternate cycle is assigned to channels 1 and 2.

II) Arithmetic & Logic Unit (ALU):

- ALU is used to perform the Arithmetic & Logical operations.
- It performs Arithmetic Operations like Addition, Subtraction & Logical Operations like AND, OR, EX-OR etc.
- ALU looks after the branching decisions.

III) Assembly/Disassembly registers:

- This registers permits 8089 to deal with 8-or 16-bit data width devices or a mix of both.
- In a particular case of an 8–bit width I/O device inputting data to a 16-bit memory interface, 8089 capture two bytes from the device and then write it into the assigned memory locations with the help of assembly/disassembly register.

IV) Bus Interface Unit (BIU):

- Fetch the instruction or data from primary memory.
- Read / Write of data from / to primary memory.
- I/O of data from / to peripheral ports.
- Address generation for memory reference.

V) Instruction Fetch:

- It is used to fetches the instructions from the external memory and stores them in the Queue to be executed further.

*** EXTRA QUESTIONS ***

Q14) DRAW & EXPLAIN BLOCK DIAGRAM OF 8253/8254 PIT.

ANS:

1. 8253/8254 is used as Programmable Interval Timer i.e. 8253/8254 PIT.
2. It is used as a device to produce Hardware delays.

FEATURES:

1. 8253/8254 PIT has 3 independent 16 bit down counters.
2. Counters can be programmed in 6 different programmable counter modes.

3. This counters can take a count in BCD or Binary.
4. It is compatible with Intel & other Microprocessor.
5. It consists of single +5V supply.
6. It consists of 24 dual in-line package.
7. It is completely TTL Compatible.
8. It can be used to generate a real time clock, or a square wave generator etc.

BLOCK DIAGRAM:

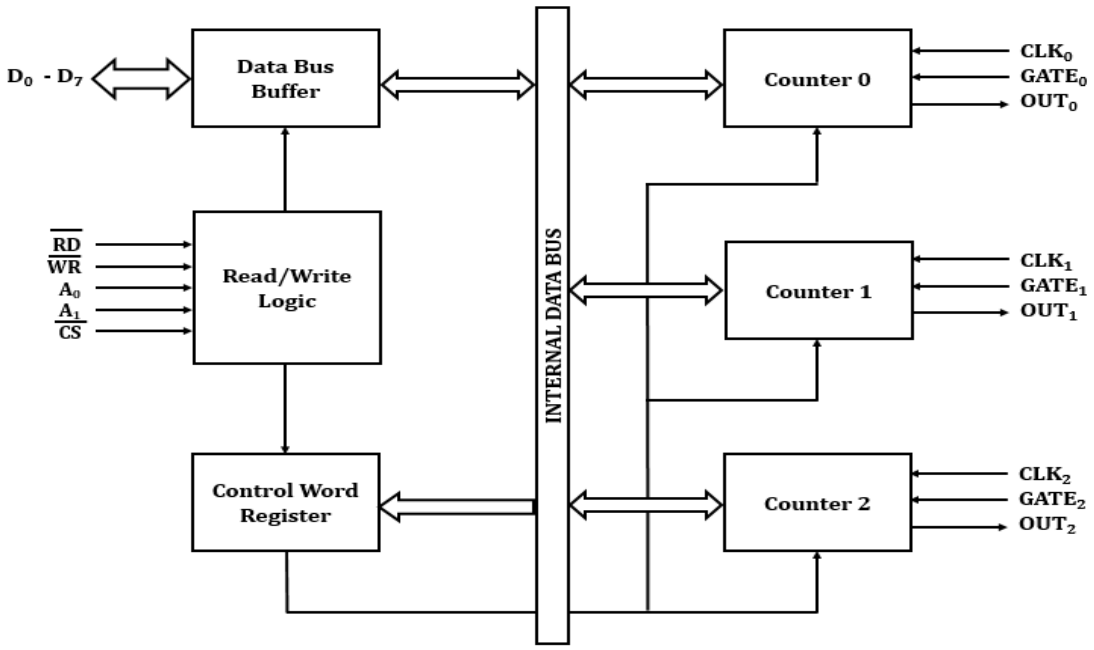


Figure 3.22: Block Diagram of 8253/8254 PIT.

It contains the following blocks:

I) Data Bus Buffer:

- It is tristate, 8-bit bidirectional data bus buffer.
- It is used to interface the internal data bus of 8253/8254 with the system data bus.
- It is internally connected to Internal Data Bus & its outer pins $D_0 – D_7$ are connected to system data bus.
- The direction of data buffer is decided by read and write control signals.

II) Read/Write Logic:

- It accepts the read & write signals, which are used to control the flow of data through data bus.
- It also accepts the $A_1 – A_0$ Address lines which are used to select one of the counters or the control word as shown below:

A_0	A_1	Selection
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word

- It also accepts the CS signal to select the 8254 chip.

III) Control Word Register:

- Control Word Register is 8 bit register that holds the Control Word.
- It is selected when $A_1 - A_0$ contain 11.
- It is used to specify the BCD or Binary Counter to be used, its mode of operation and the data transfer to be used.
- The data can only be written into control word register, no read operation is allowed.

IV) Counters:

- There are 3 independent, 16 bit down counters.
- Each counter can operate have a binary or BCD count.
- Each counter can be in one of six possible modes.
- Each counter can have a max count of $2^{16} = 65535$.
- Each counter has a following signals:
  - **CLK (Clock Input).**
  - **GATE (Gate Input).**
  - **OUT (Clock Output).**

CHAPTER-4: INTEL 80386DX PROCESSOR

Q1) EXPLAIN MEMORY MANAGEMENT IN DETAILS IN 80386DX PROCESSOR.

Q2) EXPLAIN ADDRESS TRANSLATION MECHANISM USED IN PROTOCOL MODE OF 80386.

ANS: [Q1 | 10M – MAY15] & [Q2 | 10M – DEC14]

1. Memory management is the act of managing computer memory at the system level.
2. Memory Management dynamically allocate portions of memory to programs at their request, and free it for reuse when no longer needed.
3. In 80386 DX, the logical (virtual) address is converted to the linear address.
4. This is done with the help of segmentation mechanism.
5. While the linear address is converted to physical address with the help of paging mechanism.
6. Thus, memory management in 80386 DX is done by converting logical address to physical address with the help of segmentation & paging.
7. Figure 4.1 shows logical to physical address translation in 80386.

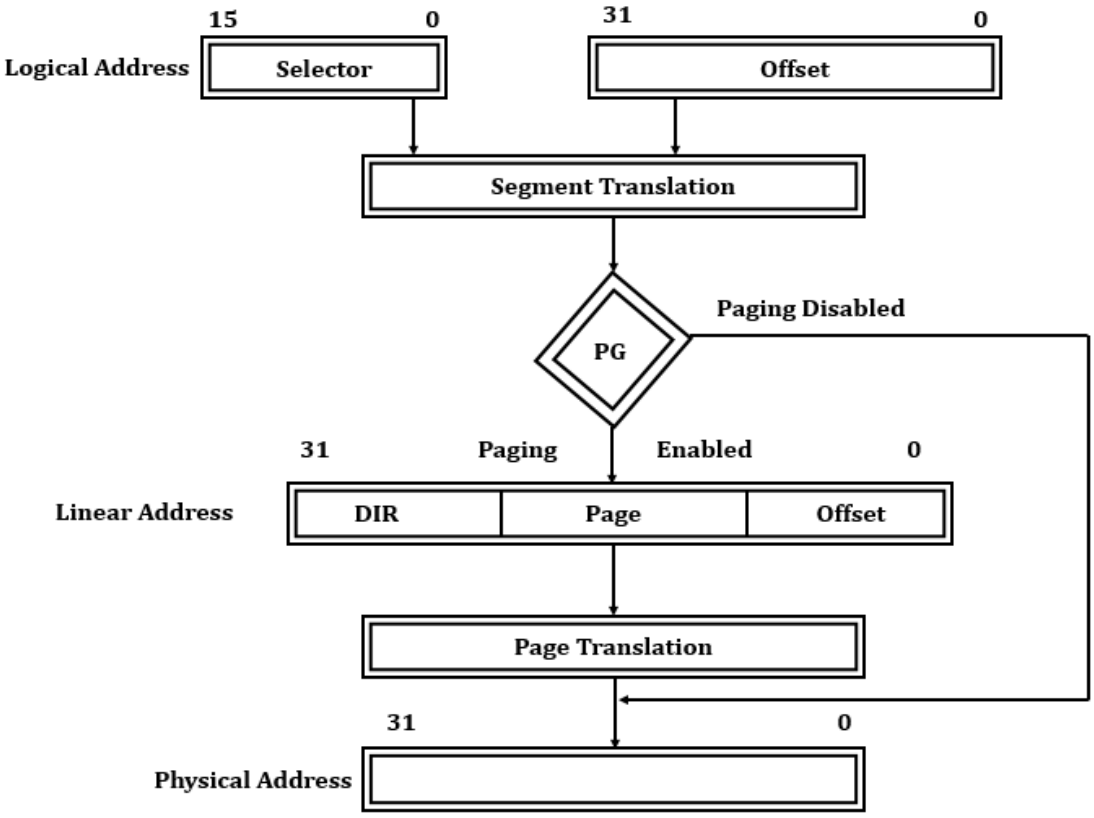


Figure 4.1: Logical to Physical Address Translation in 80386.

SEGMENT TRANSLATION:

1. Segment Translation is used to convert Logical Address to Linear Address.
2. Figure 4.2 shows the segment translation.

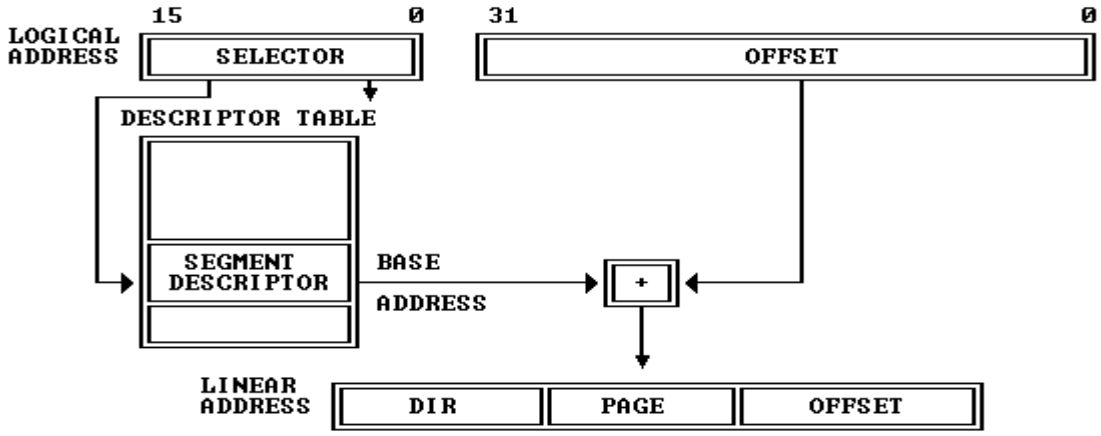


Figure 4.2: Segment Translation.

3. To perform this translation, the processor uses the following data structures:

Descriptors:

- The segment descriptor is used to provide the data which is required to map a logical address into a linear address.
- Descriptors are created by compilers, linkers, loaders, or the operating system, not by applications programmers.

Byte No.							Byte No.
7	Base (B_{31} – B_{24})	G	D	0	AV	Limit (L_{19} – L_{16})	6
5	Access Right Byte	Base Address (B_{23} – B_{16})					4
3	Base Address (B_{15} – B_0)						2
1	Limit (L_{15} – L_0)						0

Descriptor Table:

- A descriptor table is simply a memory array of 8-byte entries that contain descriptors.
- Segment descriptors are stored in either of two kinds of descriptor table:
  a. The global descriptor table (GDT): GDT is common table accessible to all the tasks.
  b. A local descriptor table (LDT): LDT is separate for each task.

Selector:

- Selector is used to select one of the 8192 descriptors from one of the tables Viz. GDT & LDT.
- Structure of selector:

16	3	2	1	0
Selector		TI		RPL

- When TI = 0 then GTD is used, when TI = 1 then LDT is used.

Segment Register:

- 80386 stores information from descriptors in segment registers.
- Every segment register has a "visible" portion and an "invisible" portion.
- 0The visible portions of segment address registers are manipulated by programs as if they were simply 16-bit registers.
- The invisible portions are manipulated by the processor.

PAGE TRANSLATION:

1. Page Translation is used to convert Linear Address to Physical Address.
2. The page-translation step is optional.
3. Page translation is in effect only when the PG bit of CR0 is set.
4. This bit is typically set by the operating system during software initialization.
5. The PG bit must be set if the operating system is to implement multiple virtual 8086 tasks, page-oriented protection, or page-oriented virtual memory.
6. Figure 4.3 shows page translation.

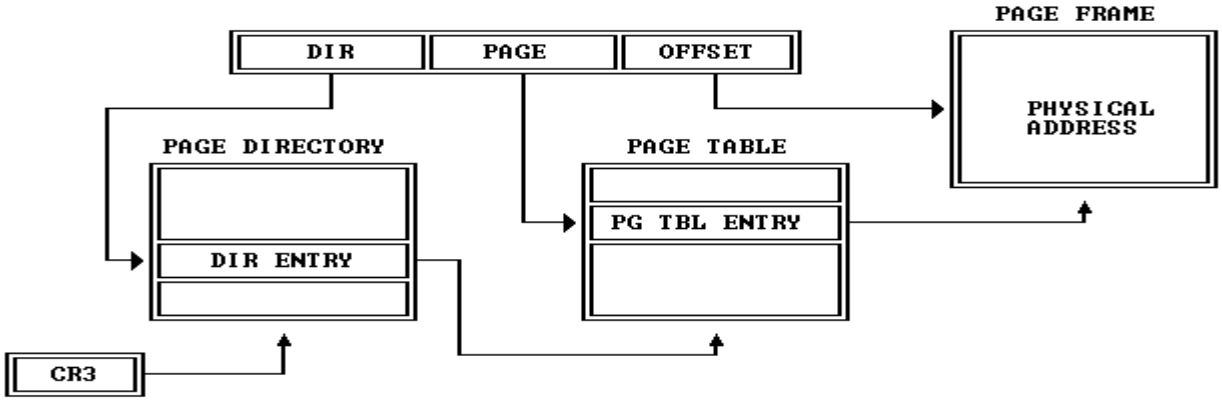


Figure 4.3: Page Translation.

7. A page table is simply an array of 32-bit page specifiers.
8. A page table is itself a page, and therefore contains 4 KB of memory.
9. Two levels of tables are used to address a page of memory.
10. At the higher level is a page directory.
11. The physical address of the current page directory is stored in the CPU register CR3, also called the page directory base register (PDBR).
12. Page Table/Directory Entry is shown as follows:

31 12	11 10 9	8	7	6	5	4	3	2	1	0
Page Frame Table Address	**OS Reserved**	**0**	**0**	**D**	**A**	**PCT**	**PWT**	**U/S**	**R/W**	**P**

13. D stands for Dirty, A stands for Access, P for Present & R/W for Read & write.
14. PCD stands for Page Cache Disabled & PWT Stands for Page Write Through.
15. U/S bit differentiates between User & Supervisor Privileges.

Q3) EXPLAIN IN BRIEF PROTECTION MECHANISM IN 80386DX PROCESSOR.

ANS: [5M – MAY15] & [10M – DEC15] & [8M – MAY16]

1. Protection Mechanism in 80386 is used to detect & identify the bugs.
2. Protection Mechanism operates in Protection Mode.
3. It is meant for Multi-tasking operations.
4. Most processor have only 2 protection levels i.e. User & Supervisor.
5. But x86 architecture features 4 levels of protection, called Privilege Levels (PL).
6. They are designed to support the needs of multi-masking OS to isolate and protect user programs from each other.
7. Figure 4.4 shows Privilege levels of the tasks in x86 Architecture.

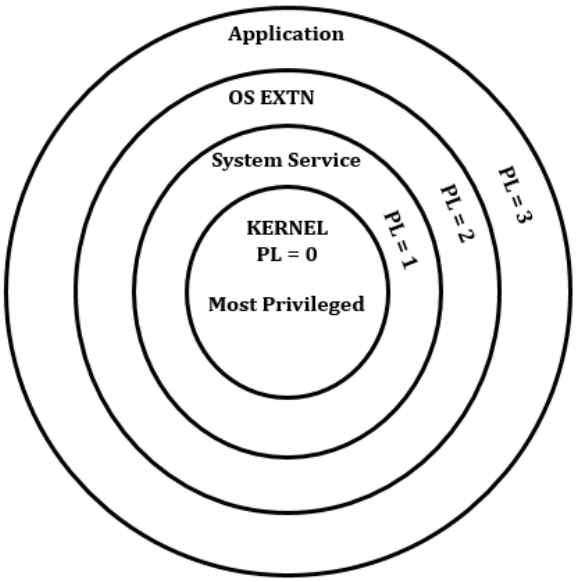


Figure 4.4: Privilege levels of the tasks in x86 Architecture.

8. The privilege levels (PL) are numbered 0, 1, 2, and 3.
9. Level 0 is the most privileged level.
10. Level 3 is the least privileged.
11. The privilege levels control the use of privileged instructions, I/O instructions, access to segments & segment descriptors.
12. The x86 architecture offers an additional type of protection on a page basis, when paging is enabled.
    a. RPL ➔ Requested PL.
    b. DPL ➔ Descriptor PL.
    c. CPL ➔ Current PL.
    d. EPL ➔ Effective PL.
13. The x86 architecture control access to both data & code between levels of task, according to the following rules of privilege:
    a. Data stored in a segment with PL = a can be accessed only by code executing at PL, numerically, at least as privilege as p.
    b. A code segment 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.

Q4) DIFFERENTIATE BETWEEN REAL MODE & PROTECTED MODE.

ANS: [10M – DEC14 & DEC15]

Parameter	Real Mode	Protected Mode
General	Real Mode is also called Real Address Mode.	Protected Mode is also called Protected Address Mode.
Default Mode	It is the default operating mode on Reset.	It is not default operating mode on reset.
Entering the Mode	The 80386 begins its execution in real mode on power up or reset.	For protected mode operation the Protection Enable (PE) bit of Control Register 0 must be set.
Leaving the Mode	To leave the Real Mode & Enter Protected Mode the PE bit of Control Register 0 must be set.	To leave the Protected Mode the User can clear the PE bit in Control Register 0.
Use	It is use to initialize 80386 for Protected Mode operation.	It is use for Segmentation, Paging, Multi-tasking, Virtual Memory and Protection.
Access	In the Real Mode 80386 can access all the registers.	Protected Mode can access all general purpose registers, control registers, debug registers, test registers, segment selectors and segment descriptors.
Memory Addressing	It can address up to 1 MB of memory.	It can address up to 4 GB of memory with 32 bit addressing.
Segment Base Address	20 Bit.	32 bit, from descriptor for 32 bit Protected Mode.
Segment Protection	No	Yes

REAL MODE ADDRESSING:

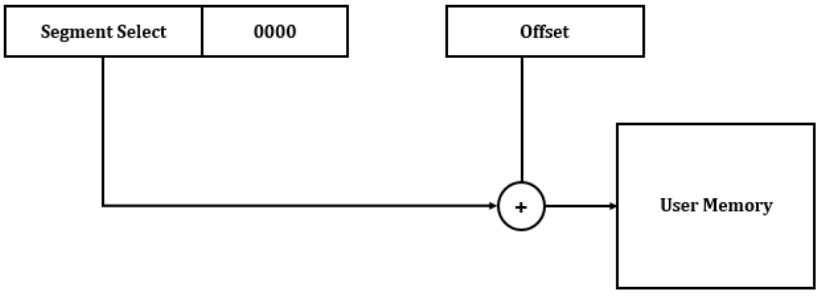


Figure 4.5: Real Mode Addressing.

PROTECTED MODE ADDRESSING:

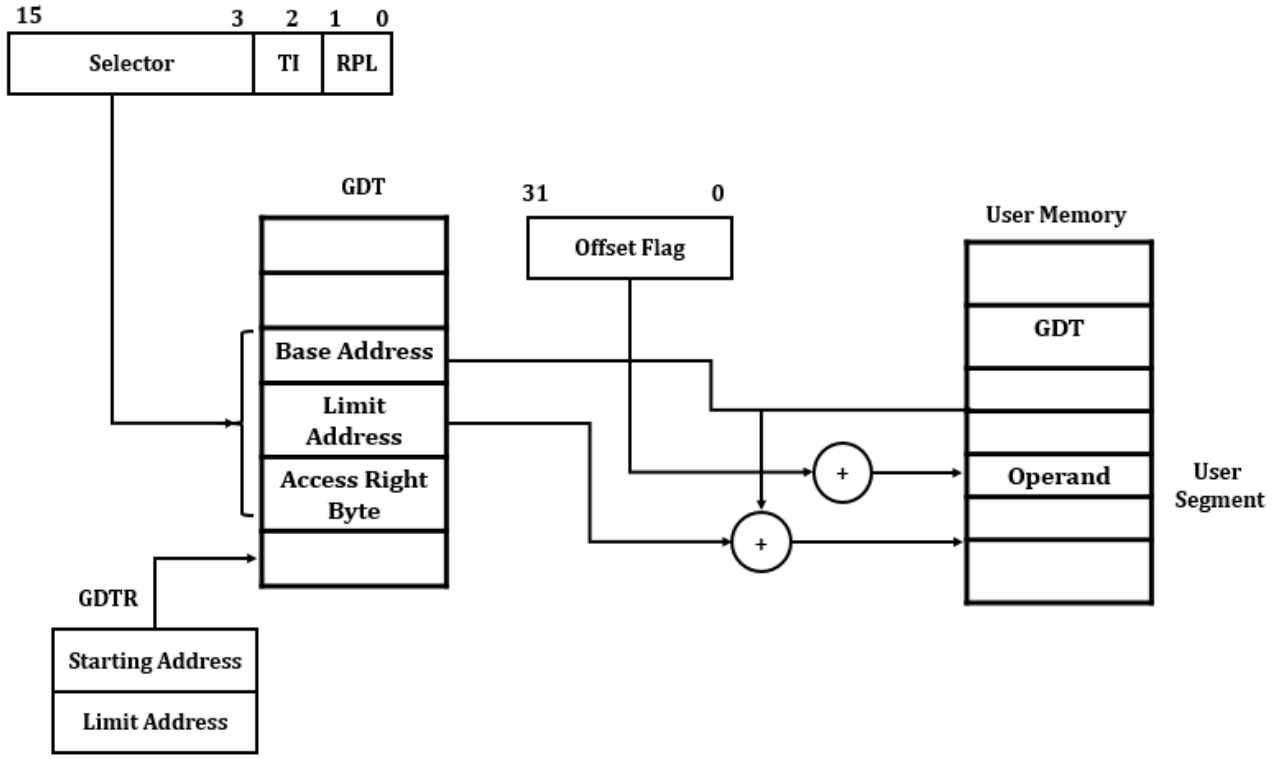


Figure 4.6: Protected Mode Addressing.

GDT: Global Descriptor Table.
GDTR: Global Descriptor Table Register.
TI: Table Identity.
RPL: Requested Privilege Level.

Q5) DRAW A SEGMENT DESCRIPTOR FORMAT & EXPLAIN DIFFERENT FIELDS.

ANS: [10M – DEC14]

1. The segment descriptor is used to provide the data which is required to map a logical address into a linear address.
2. Descriptors are created by compilers, linkers, loaders, or the operating system, not by applications programmers.
3. Descriptor is used to give the different details of the segment.
4. The structure of Segment Descriptors is shown below:

Byte No.							Byte No.
7	Base ($B_{31} - B_{24}$)	G	D	0	AV	Limit ($L_{19} - L_{16}$)	6
5	Access Right Byte	Base Address ($B_{23} - B_{16}$)					4
3	Base Address ($B_{15} - B_0$)						2
1	Limit ($L_{15} - L_0$)						0

Limit Address ($L_0 – L_9$):

- Limit Address defines the size of the segment.
- When the processor concatenates the two parts of the limit field, a 20-bit value results.
- Limit Address when added with the base address gives the last address of the segment.
- In case of 286 there is a 24 bit base address and 16 bit limit.
- While in 386 we have 32 bits base address and 20 bit limit address.

Base Address ($B_0 – B_{31}$):

- Base Address indicates the starting address of the memory segment.
- It defines the location of the segment within the 4 gigabyte linear address space.
- The processor concatenates the three fragments of the base address to form a single 32-bit value.

Granularity Bit (G):

- Granularity Bit specifies the units with which the LIMIT field is interpreted.
- When G = 0, 20 bit Limit specifies the segment size from 1 byte to 1 MB.
- When G = 1, 20 bit Limit is to be multiplied by 4K hence segment size varies from 4 KB to 4 GB.

Availability (AV):

- It indicates the availability of segment.
- When AV = 1, it indicates the corresponding segment is not used by any other task and hence is available.
- When AV = 0, it indicates the corresponding segment is used by some other task and hence is not available.

Data Size (D):

- It indicates the Data Size.
- When D = 0, it indicates 16 bit OS instructions.
- When D = 1, it indicates 32 bit OS instructions.

Access Right Byte (ARB):

7	6 5	4	3	2	1	0
P	DPL	S	E	ED/C	RW	A

Present (P):

- When P = 1, the entry of the descriptor is initialized.
- When P = 0, the entry of the descriptor is not initialized.

Descriptor Privilege Level (DPL):

- Segment Privilege Level, used in Privilege tests.

Segment Descriptor (S):

- When S = 1, Code or Data Segment Descriptor.
- When S = 0, System Segment Descriptor or Gate Descriptor.

Executable (E):

- When E = 0, Descriptor type is data (or stack) segment.
- When E = 1, Descriptor type is code segment.

Expansion Direction (ED):

- When ED = 0, Expand up segment.
- When ED = 1, Expand down Segment.

Writeable (W):

- When W = 0, Data Segment cannot be written into, i.e. Read only.
- When W = 1, Data Segment may be written into.

Readable (R):

- When R = 0, Code Segment cannot be read.
- When R = 1, Code Segment may be read.

Accessed (A):

- When A = 0, Segment has not been accessed.
- When A = 1, Segment has been accessed earlier.

Q6) STATE THE USE OF RF, TF, VM, NT, IOPL FLAG BITS.

ANS: **[5M – DEC14]**

FLAG REGISTER:

1. The flags register is a 32-bit register named EFLAGS.
2. The flags control certain operations and indicate the status of the microprocessor.
3. Figure 4.7 shows the Flag Register for 80386.

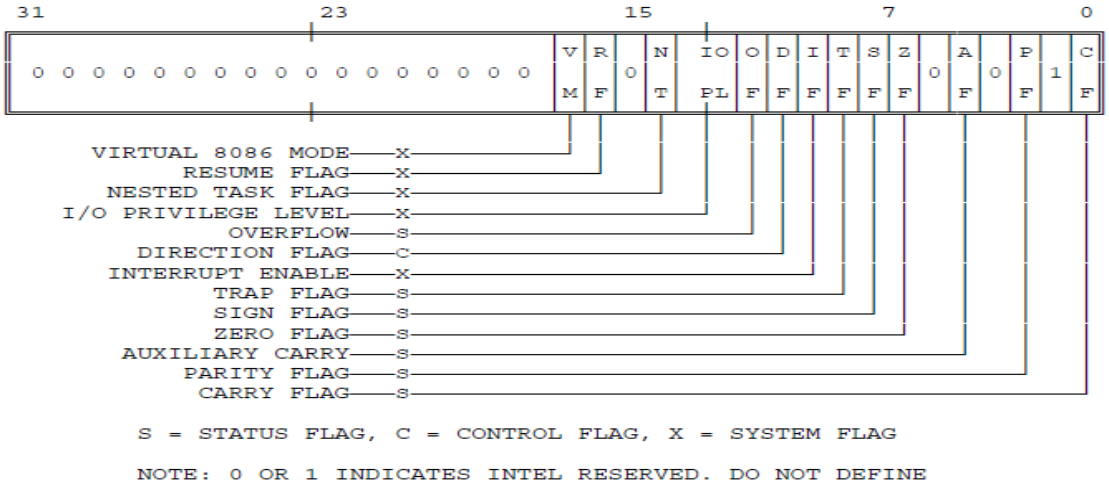


Figure 4.7: Flag Register of 80386.

RF:

1. RF stands for Resume Flag.
2. This flag is used with the debug register breakpoints.
3. It is checked at the starting of every instruction cycle and if it is set, any debug fault is ignored during the instruction cycle.
4. The RF is automatically reset after successful execution of every instruction.

TF:

1. TF stands for Trap Flag.
2. Setting TF puts the processor into single-step mode for debugging.
3. In this mode, the CPU automatically generates an exception after each instruction, allowing a program to be inspected as it executes each instruction.
4. Single-stepping is just one of several debugging features of the 80386.

VM:

1. VM stands for Virtual 8086 Mode.
2. If this flag is set, the 80386 enters the virtual 8086 mode within the protection mode.
3. This is to be set only when the 80386 is in protected mode.
4. This bit can be set using IRET instruction or any task switch operation only in the protected mode.

NT:

1. NT stands for Nested Task.
2. When NT = 1, it indicates that the currently executing task is nested within another task and it has valid link to caller task.
3. The processor uses the nested task flag to control chaining of interrupted and called tasks.

IOPL:

1. IOPL stands for I/O Privilege Level.
2. The IOPL field in the FLAGS register defines the right to use I/O-related instructions.

Q7) EXPLAIN V86 MODE OF 80386DX

ANS: **[5M – MAY16]**

1. In protected mode of operation, 80386DX provides a virtual 8086 operating environment to execute the 8086 programs.
2. The real mode can also use to execute the 8086 programs along with the capabilities of 80386, like protection and a few additional instructions.
3. Once the 80386 enters the protected mode from the real mode, it cannot return back to the real mode without a reset operation.
4. Thus, the virtual 8086 mode of operation of 80386, offers an advantage of executing 8086 programs while in protected mode.
5. V86 Mode is also known as Virtual Mode of 80386.
6. V86 Mode is a Dynamic Mode.

7. It can switch repeatedly & rapidly between V86 Mode & Protected Mode.
8. To execute an 8086 program, the CPU enters in V86 Mode from Protected Mode.
9. CPU Leaves V86 Mode and enters protected mode to continue executing a native 80386 program.

ADDRESSING IN VIRTUAL MODE:

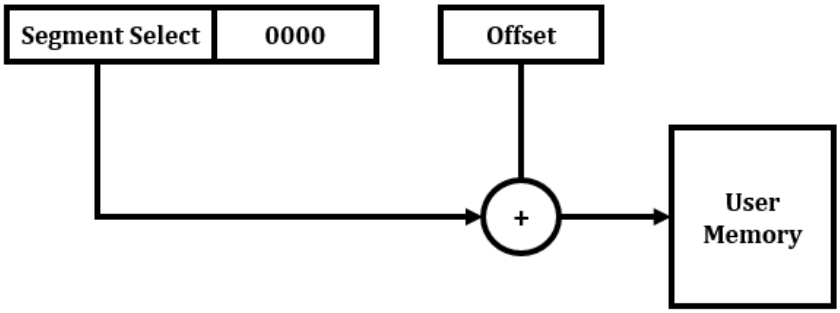


Figure 4.8: Virtual Mode Addressing.

1. The address forming mechanism in virtual 8086 mode is exactly identical with that of 8086 real mode.
2. In virtual mode, 8086 can address 1MB of physical memory that may be anywhere in the 4GB address space of the protected mode of 80386.
3. Like 80386 real mode, the addresses in virtual 8086 mode lie within 1MB of memory.
4. In virtual mode, the paging mechanism and protection capabilities are available at the service of the programmers.
5. The virtual mode allows the multiprogramming of 8086 applications.
6. The virtual 8086 mode executes all the programs at privilege level 3.

*** EXTRA QUESTIONS ***

Q8) DRAW & EXPLAIN BLOCK DIAGRAM OF 80386 DX

ANS: [10M - DEC14]

FEATURES:

1. Microprocessor 80386 DX is a 32-bit processor.
2. It has 32-bit address bus and a 32-bit data bus.
3. It has 8 General Purpose 32 bit registers supports 8, 16, 32 bit data types.
4. 80386 DX can access up to 4 GB physical memory or 64 TB of virtual memory.
5. It runs with speed up to 20 MHz instructions per second.
6. It has 4 levels of protection. i.e. PL_0 – PL_3.
7. The 80386DX can operate in real mode, protected mode & virtual 8086 mode.
8. It has built-in virtual memory management circuitry and protection circuitry.
9. 80386 Microprocessor supports hardware debugging.
10. The 80386DX microprocessor is compatible with their earlier 8086, 8088, 80186, 80188, 80286 chips.

BLOCK DIAGRAM:

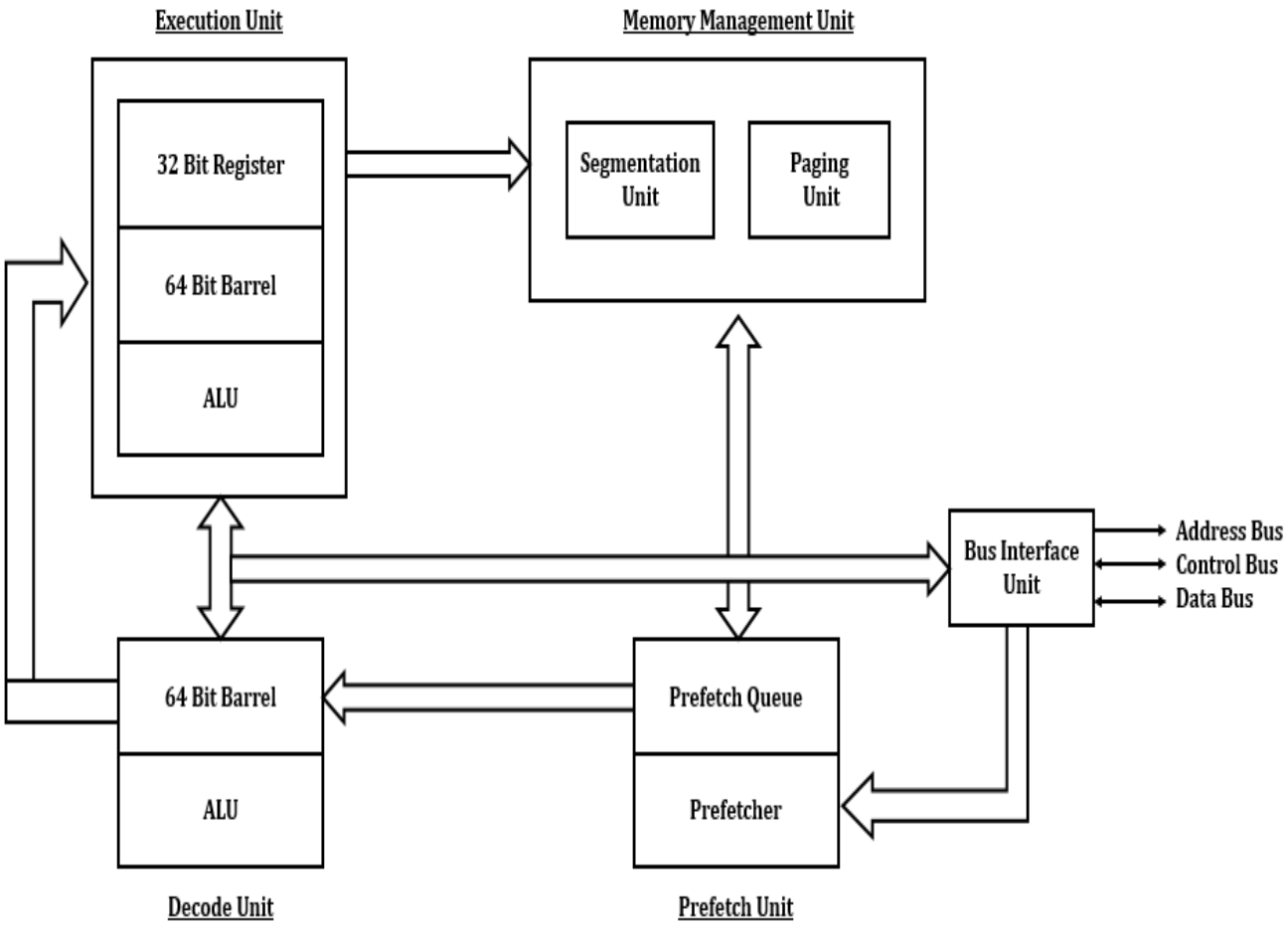


Figure 4.9: 80386DX Block Diagram.

The block diagram of 80386 DX includes the following units:

I) Bus Interface Unit:

- This unit includes the address drivers, transceivers for data bus & bus control signals.
- It handles the communication with devices external to the microprocessor chip.

II) Prefetcher & Prefetch Queue:

- The Prefetcher fetches the instructions from the external memory and stores them in the Prefetch queue to be executed further.
- Prefetch queue is 16 byte in size.

III) Instruction Decoder and Decoded Instruction Queue:

- The instruction decoder takes the instruction from the Prefetch queue and after decoding it, stores them in the decoded instruction queue.
- The decoded instruction queue can store up to three decoded instructions.

IV) Execution Unit:

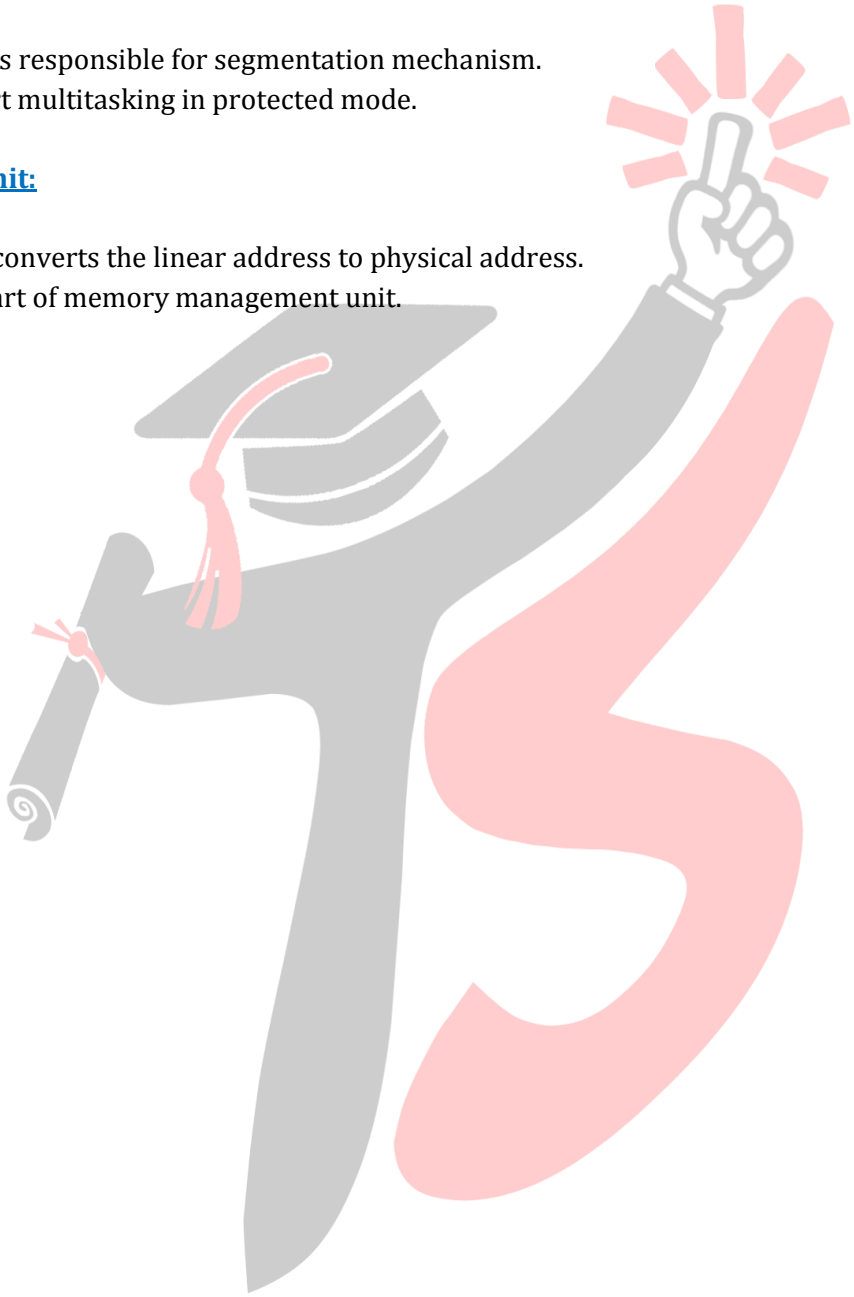
- It consists of 8 general purpose 32 bits register, 64 bit barrel shifter & ALU.
- The execution unit executes each instruction received from the decoded instruction queue.

V) Segmentation Unit:

- This unit is responsible for segmentation mechanism.
- Its support multitasking in protected mode.

VI) Paging Unit:

- This unit converts the linear address to physical address.
- It is the part of memory management unit.



CHAPTER-5: PENTIUM PROCESSOR

Q1) WRITE DOWN FEATURES OF PENTIUM PROCESSOR.

Q2) DRAW AND EXPLAIN ARCHITECTURE OF PENTIUM PROCESSOR.

ANS: [Q1 | 10M – DEC15] & [Q2 | 10M – MAY16]

1. Pentium Processor is produced by Intel since 1993.
2. It is the brand used for a series of P5 x86 compatible microprocessor.

FEATURES:

1. Pentium Processor has a superscalar architecture.
2. Pentium Processor is 32 bit microprocessor.
3. Physical memory is of 64 GB.
4. Virtual memory is of 64 TB.
5. It has 36 bit addressing lines.
6. It has 64 bit data path.
7. It has Dedicated Instruction Cache & Dedicated Data Cache.
8. Pentium Processor supports Parallel Integer Execution.

BLOCK DIAGRAM:

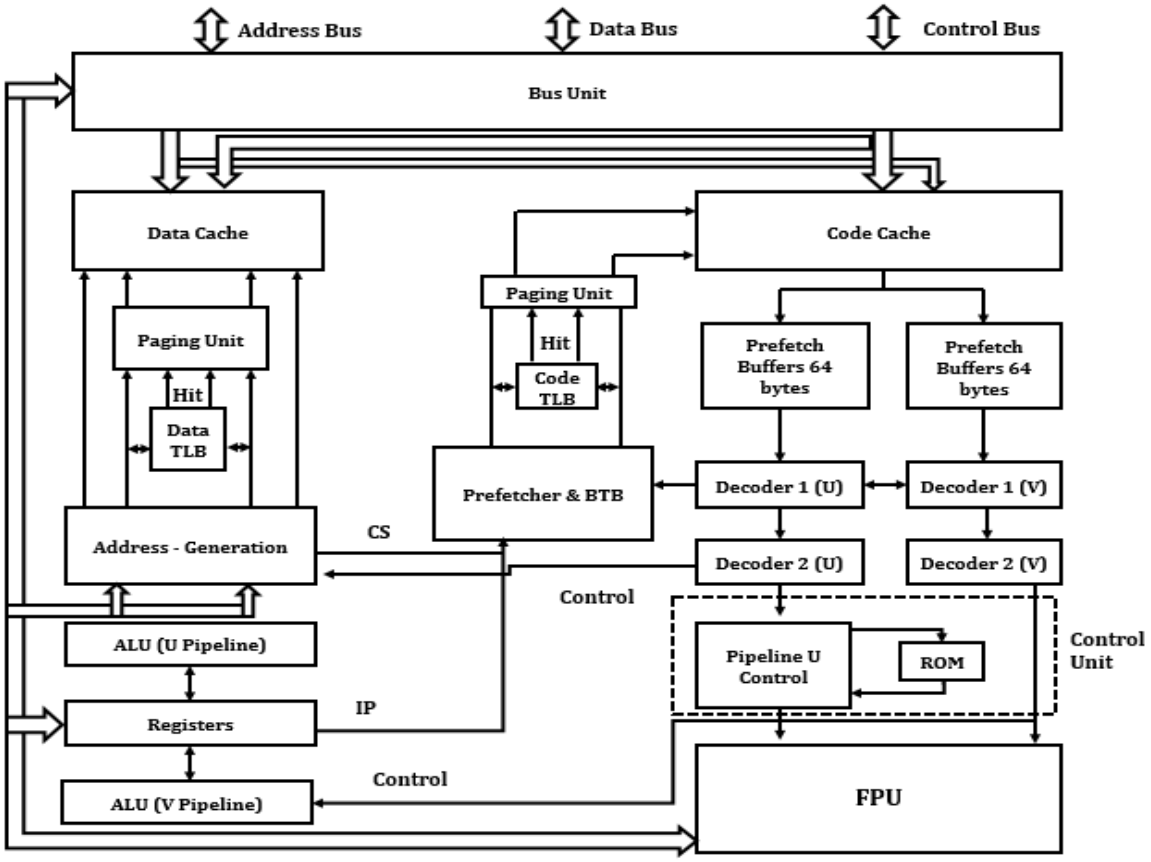


Figure 5.1: Pentium Architecture.

Pentium Architecture includes following blocks:

Bus Unit:

7. It provides the physical interface between the Pentium processor & rest of the system.
8. It consists of Address Drivers & Receivers, Data Bus Transceiver & Bus Control Logic.

Data Cache:

- Data Cache in Pentium is 8 KB write back cache.
- It is triple ported to allow simultaneous access from each of the pipelines and snooping.
- Data cache keeps copy of most frequently used data by 2 integer pipelines and FPU.

Code Cache:

- Data Cache in Pentium is 8 KB write back cache.
- It is triple ported to allow simultaneous access from each of the pipelines and snooping.
- Data cache keeps copy of most frequently used data by 2 integer pipelines and FPU.

Prefetcher:

- Instructions are requested from code cache by the Prefetcher.
- If the requested line is not in cache, a burst cycle is run to external memory to perform a cache line fill.

Prefetch Buffers:

- Pentium processor consists of four Prefetch buffers as 2 independent pairs.
- When instruction is prefetched it is placed into one set of Prefetch buffers, while the other pair is idle.

Instruction Decode Unit:

- It occurs in two stages – Decode1 (D1) and Decode2 (D2).
- D1 checks whether instructions can be paired.
- D2 calculates the address of memory resident operands.

Control Unit: It Consists of Microcode Sequencer and Microcode Control ROM.

ALU:

- The ALU for 'U' Pipeline can complete an instruction prior to the ALU in 'V' Pipeline.
- But the ALU for 'V' Pipeline cannot complete an instruction prior to the ALU in 'U' Pipeline.

Paging Unit:

- If Paging is enabled, the Paging Unit translates the linear address from address generator to a physical address.
- Two translation look aside Buffers (TLB) are implemented, one for each code & data cache.

Floating Point Unit:

- The FPU uses an 8 stage pipeline.
- Three types of floating point operations can operate simultaneously within FPU: Addition, Division and Multiplication.

Q3) EXPLAIN HOW THE FLUSHING OF PIPELINE PROBLEM IS MINIMIZED IN PENTIUM ARCH.

Q4) EXPLAIN BRANCH PREDICTION LOGIC USED IN PENTIUM.

ANS: [Q3 | 10M – DEC14] & [Q4 | 10M – MAY15 & DEC15]

1. Program Transfer Instructions such as JMP, CALL, RET and Conditional Jumps reduces the performance gain through pipelining.
2. This is because they change the sequence of all the instructions that entered the pipeline after Program Transfer Instruction, thus assuming the previous instructions invalid.
3. Suppose instruction I_3 is a conditional jump to I_{50} at some other address (target address), then the instructions that entered after I_3 is invalid and new sequence beginning with I_{50} need to be loaded in.
4. This causes bubbles in pipeline, where no work is done as the pipeline stages are reloaded.
5. To avoid this problem, the Pentium uses a scheme called Dynamic Branch Prediction.
6. In this scheme, a prediction is made concerning the branch instruction currently in pipeline.
7. Prediction will be either taken or not taken.
8. If the prediction turns out to be true, the pipeline will not be flushed and no clock cycles will be lost.
9. If the prediction turns out to be false, the pipeline is flushed and started over with the correct instruction.
10. It results in a 3 cycle penalty if the branch is executed in the u-pipeline and 4 cycle penalty in v-pipeline.
11. It is implemented using a 4-way set associative cache with 256 entries.
12. This is referred to as the Branch Target Buffer (BTB).

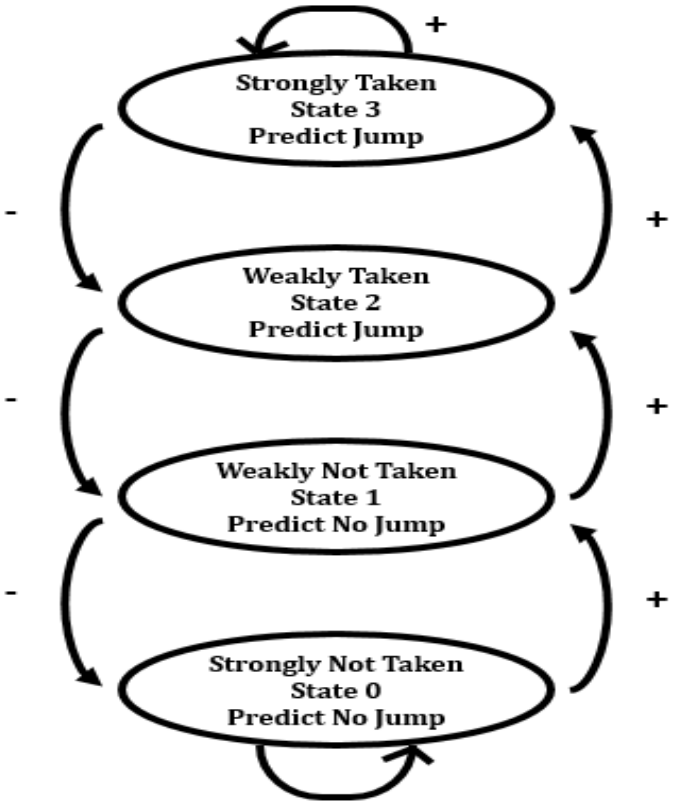


Figure 5.2: State Transition Diagram of BTB Entry.

13. The directory entry for each line contains the following information:
    a. **Valid Bit:** Indicates whether or not the entry is in use.
    b. **History Bits:** Track how often the branch has been taken.
    c. Source memory address that the branch instruction was fetched from (Address of I_3).
14. The history bits indicates one of four possible states.

History Bits	Resulting Description	Prediction Made	If branch is taken	If branch is not taken
11	**Strongly Taken**	Branch Taken	Remains Strongly Taken	Downgrades to Weakly Taken
10	**Weakly Taken**	Branch Taken	Upgrades to Strongly Taken	Downgrades to Weakly Not Taken
01	**Weakly Not Taken**	Branch Not Taken	Upgrades to Weakly Taken	Downgrades to Strongly Not Taken
00	**Strongly Not Taken**	Branch Not Taken	Upgrades to Weakly Not Taken	Remains Strongly Not Taken

15. Thus, if the branch was correctly predicted to be taken, the history bits are upgraded and no further action necessary i.e. correct instructions are already in the pipeline.
16. While, if branch was incorrectly predicted to be taken, the history bits are downgraded and pipeline needs to be flushed and switching of pre-fetcher queue takes place.
17. And if the branch was correctly predicted not to be taken, history bits are downgraded and no action required.
18. While, if incorrectly predicted not to be taken then history bits are upgraded and the queue is flushed and instructions fetched from previous Prefetch queue that contains sequential instructions.
19. Hence time is saved in this case because there are two Prefetch queues.

Q5) Enlist the instruction pairing rules for U & V pipeline in Pentium.

Ans: **[5M – DEC14]**

1. Pentium processor can execute 2 integer instructions simultaneously.
2. The first instruction enters into U pipeline and the next into V pipeline.
3. U pipeline has barrel shifter and V pipeline has no barrel shifter.
4. Due to this some instructions can be executed only in U pipeline.
5. The instruction are pairable only if following conditions are satisfied.
   - Instruction should be simple such as:
     - MOV register, immediate.
     - MOV register, register.
     - MOV register, memory.

- MOV memory, register.
- MOV memory, immediate.
- INC register.
- INC memory.
- DEC register.
- DEC memory.

➢ Instruction should not have register contention:

- Register contention occurs if two instruction access the same register at a time.

  MOV AX, 5238H

  MOV [SI], AX

- Old value of AX is stored in the memory instead of new value 5238. So they cannot be paired.

  ADD AX, BX

  ADD AX, CX

- Old value to AX is added with CX instead of new value in AX. So they cannot be paired.

  DEC CL

  JNZ DOWN

- Both access flag register at a time. So they cannot be paired.

➢ Some instruction are only pairable only if they are first instruction in the pair. i.e.: these instruction needs compulsorily U pipeline. The reason is V pipeline has no barrel shifter.

➢ Some instruction are only pairable if they are second instruction pair.

For example: Unconditional branch instruction.

Q6) Write the instruction issue algorithm used in Pentium.

Ans: [5M – DEC15]

Algorithm:

1. Decode the two consecutive instructions I_1 and I_2.
2. If all the following conditions are true, the two instructions are pairable and issue I_1 to 'U' Pipeline and I_2 to 'V' Pipeline.
3. Else they are not pairable, and only the instruction I_1 is to be given to 'U' Pipeline.
4. The Conditions are:
   i. I_1 and I_2 are simple instructions.
   ii. I_1 is not a jump instruction.
   iii. Destination of I_1 is not a source of I_2.
   iv. Destination of I_1 is not a destination of I_2.

Q7) COMPARATIVE STUDY OF MULTICORE I3, I5 & I7 PROCESSORS.

Q8) COMPARISON BETWEEN I5 & I7.

ANS: [Q7 | 5M – MAY15] & [Q8 | 5M – MAY16]

Table 5.1 shows the Comparative Study of multicore i3, i5 & i7 processors.

Table 5.1: Comparative Study of multicore i3, i5 & i7 processors.

Features	i3	i5	i7	i7 Extreme
Number of Cores	2 for Desktop as well as for Laptop.	4 for Desktop. 2 for Laptop.	4 or 6 for Desktop. 2 or 4 for Laptop.	6 for Desktop. 4 for Mobile.
Processing Threads	4 for Desktop as well as Laptop.	8 threads for Desktop. 4 threads for Laptop.	8 or 12 threads for Desktop. 4 or 8 threads for Laptop.	12 threads for Desktop. 8 threads for Laptop.
Maximum Base Clock Frequency	3.4 GHz	3.4 GHz	3.2 GHz	3.3 GHz
Maximum Turbo Boost Frequency	Not Applicable.	3.8 GHz	3.8 GHz	3.9 GHz
Maximum Smart Cache Size	3 MB	6 MB	12 MB	15 MB
Intel Turbo Boost 2.0	Not present.	Present.	Present.	Present.
Intel Hyper Threading	Present.	Present only in Laptop Processor.	Present.	Present.
K Model	Not present.	Present.	Present.	Present.
Best Desktop Processor	Intel Core i3-2130 (3.4 GHz, 3 MB)	Intel Core i5-2550 K (3.4 GHz, 6 MB)	Intel Core i7-3930 (3.2 GHz, 12 MB)	Intel Core i7-3960 (3.3 GHz, 15 MB)
Best Mobile (Laptop) Processor	Intel Core i3-2370 (2.4 GHz, 3 MB)	Intel Core i5-2540 M (2.6 GHz, 3 MB)	Intel Core i7-2860 (2.5 GHz, 8 MB)	Intel Core i7-2960 XM (2.7 GHz, 8 MB)

Q9) COMPARE PENTIUM 2, PENTIUM 3 & PENTIUM 4 PROCESSOR.

ANS: [10M – MAY15]

Table 5.2 shows the Comparative Study of Pentium 2, 3 & 4 processors.

Table 5.2: Comparative Study of Pentium 2, 3 & 4 Processor.

Features	Pentium 2	Pentium 3	Pentium 4
Processor Size.	32 Bits.	32 Bits.	32 Bits.
Speed.	233 to 300 MHz.	450 to 500 MHz.	400 MHz.
Address Bus Size.	32 bit.	32 bit.	32 bit.
Data Bus Size.	64 bit.	64 bit.	64 bit.
No. of Transistors.	7.5 Million.	9.5 To 28 Million.	77 Million.
Addressable Memory.	4 GB.	4 GB.	4GB.
Virtual Memory.	64 TB.	64 TB.	64 TB.
Superscalar.	Yes.	Yes.	Yes.
MMX Instruction Set.	SSE1.	SSE2.	SSE2.
Hyper Threading Support.	No.	No.	Yes.
Generation.	P6.	P6.	Net Burst.
Multiprocessor Support.	No.	No.	Yes.
Integer Pipeline Stages.	14.	14.	20.
No. of Integer Pipelines.	2.	2.	4.
Floating Point Pipeline Stages.	8.	11.	20.
No. of Floating Point Pipelines.	1.	1.	2.
On Chip FPU.	Yes.	Yes.	Yes.
No. of Cores.	1.	1.	1.
Overclocking Feature.	No.	No.	Yes.
Fabrication Processor.	0.35 µm.	0.18 µm.	0.13 µm.

Q10) COMPARE 8086, 80386 & PENTIUM.

ANS: [10M – DEC15]

Table 5.3 shows the Comparative Study of 8086, 80386 & Pentium processors.

Table 5.3: Comparative Study of Pentium 8086, 80386 & Pentium Processor.

Features	8086	80386	Pentium
Processor Size.	16 Bits.	32 Bits.	32 Bits.
Speed.	5 to 10 MHz.	16 to 33 MHz.	60 To 66 MHz.
Address Bus Size.	20 bit.	32 bit.	32 bit.
Data Bus Size.	16 bit.	32 bit.	64 bit.
No. of Transistors.	29000.	275000.	3.1 Million.
Addressable Memory.	1 MB.	4 GB.	4GB.
Virtual Memory.	-	64 TB.	64 TB.
Superscalar.	No	No	Yes.
MIPS	0.33 to 0.75.	5 to 11.4.	100 to 112.
L1 Cache.	Not Present.	Not Present.	16 KB Split.
Generation.	P1.	P3.	P5.
Multiprocessor Support.	No.	No.	No.
Integer Pipeline Stages.	2.	3.	5.
No. of Integer Pipelines.	1	1.	2.
Floating Point Pipeline Stages.	Not Present.	Not Present.	8.
No. of Floating Point Pipelines.	Not Present.	Not Present.	1.
On Chip FPU.	No.	No.	Yes.
No. of Cores.	1.	1.	1.
Overclocking Feature.	No.	No.	No.
Fabrication Processor.	03 µm.	1 µm.	0.6 µm.

Q11) CODE CACHE ORGANIZATION OF PENTIUM.

ANS: [5M – DEC14 & 8M – MAY16]

1. Code Cache is designed to permit two simultaneous Prefetch accesses.
2. This helps in accessing an instruction that resides in two adjacent cache lines in a single cycle.
3. Figure 5.3 shows Pentium Code Cache Architecture.

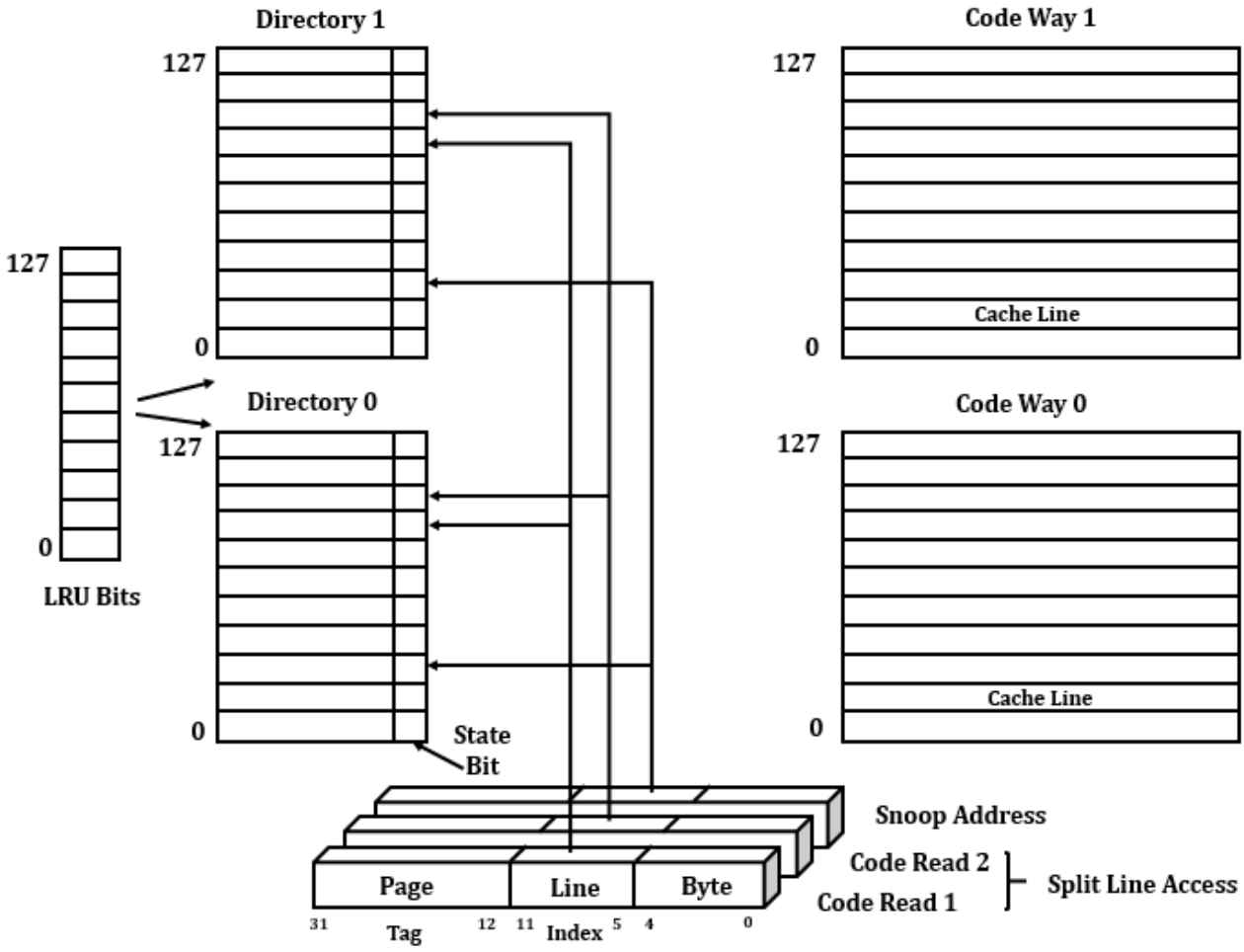


Figure 5.3: Pentium Code Cache Architecture.

4. Code Cache in Pentium is 8 KB in Size.
5. It is organized as 2 way set-associative mapping configuration.
6. There are 2 cache ways i.e. Cache Way 0 & Cache Way 1.
7. Each cache line is (256 bits) 32 bytes wide.
8. The bus connected from the cache to the Prefetcher is also (256 bits) 32 bytes, allowing 32 bytes to be delivered to Prefetcher Queue during a single Prefetch.
9. Each Cache Way contains 128 cache lines with an associated 128 entry directory with each of the cache ways.
10. The cache directories are tripled ported, to support split line access and snooping.
11. The directory entry consists Tag Field, State Bit & Parity Bit.
    a. **Tag Field:** It is of 20 bit, used to identify the page in the memory.
    b. **State Bit:** It indicates whether the line in cache contains valid or invalid information.
    c. **Parity Bit:** It is used to detect error when reading each entry.

12. The directories are accessed by the address issued by the Prefetcher.
13. When the Prefetcher initiates a split-line access, the two line address are submitted to the code cache.
14. Address bits $A_{11} – A_{15}$ from the Prefetcher identify the set where the target line may reside in cache, and are used as index into the cache directories.
15. The lower portion of the Prefetcher address $A_4 – A_0$ identifies a byte within the line.

Q12) DATA CACHE ORGANIZATION OF PENTIUM.

ANS: [5M – DEC15]

1. All accesses by the Execution Units for data are routed through the data cache.
2. Data Cache is used to stores operand information.
3. Figure 5.4 shows Pentium Data Cache Architecture.

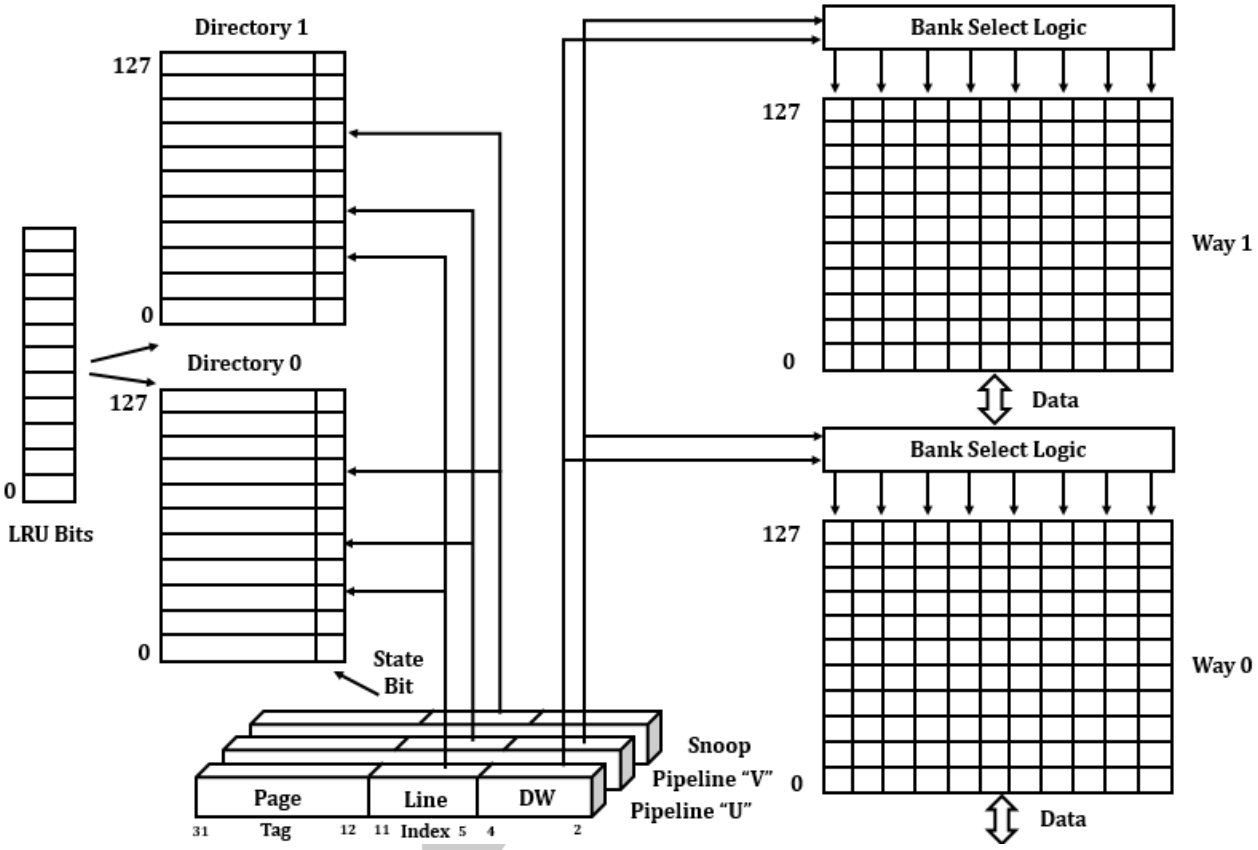


Figure 5.4: Pentium Data Cache Architecture.

4. Data Cache in Pentium is 8 KB in Size.
5. It is organized into two 4 KB ways referred to as Way 0 & Way 1.
6. It contains triple-ported high speed SRAM.
7. Each way consists of 128 lines numbered 0 through 127 with a line size of 32 bytes.
8. Each data cache line consists of 8 double words and the cache ways are banked on double word boundaries.
9. A parity is generated for each byte within a line that is placed in the internal cache.
10. When a byte of information is read from cache, parity is checked.

11. And in case of parity error being detected, an internal parity error is signaled to external logic through the Internal Error Output.
12. Also the processor generates a special shutdown bus cycle and stops execution.
13. Each Directory entry has a tag field used to record the page number of the memory page where the line of information came from.
14. Figure 5.5 shows the Data Cache Directory Entry Structure.

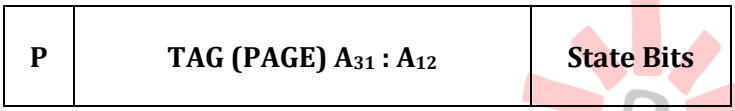


Figure 5.5: Data Cache Directory Entry Structure.

15. **State Bits:** 00 → Invalid.
    11 → Exclusive.
    22 → Modified.
    33 → Shared.

Q13) EXPLAIN IN BRIEF PIPELINE STAGES ON PENTIUM PROCESSOR

Q14) INTEGER PIPELINE STAGES

ANS: [Q13 | 5M – MAY16]

1. To execute instruction at high speed, pipelining is used.
2. In Pentium, there are two pipeline known as U pipeline and V pipeline.
3. There are 5 stages in Integer Pipelining.
4. They are:

A. PREFETCH STAGE:

- The 2 Prefetch queue A and B are available in integer pipelining stage.
- Each has 64 bytes.
- At a time only one queue is active.
- Each queue have 32 bytes buffer.
- It is used to fetch instructions from code cache and align the code to the initial byte of next instruction.
- Then place it in the active Prefetch Buffer.
- Two instructions from active Prefetch buffer reaches decode one stage of pipeline.

B. DECODE ONE:

- In this stage partial decoding is performed.
- It is responsible to identify whether 2 instructions or functions can be perform in parallel.
- The basic function of Decode 1 stage is:
  - Check pair ability.
  - Barrier shifter.
  - Branch prediction.
- The 2 instructions are checked whether they can form a pair.

- If they can form a pair, then both the instruction move in together.
- If they cannot, then instruction in the V pipeline of decode one stage is transferred to decode one stage of U pipeline.
- At the same time, first instruction in decode one stage of U pipeline is moved to decode two stage of U pipeline.

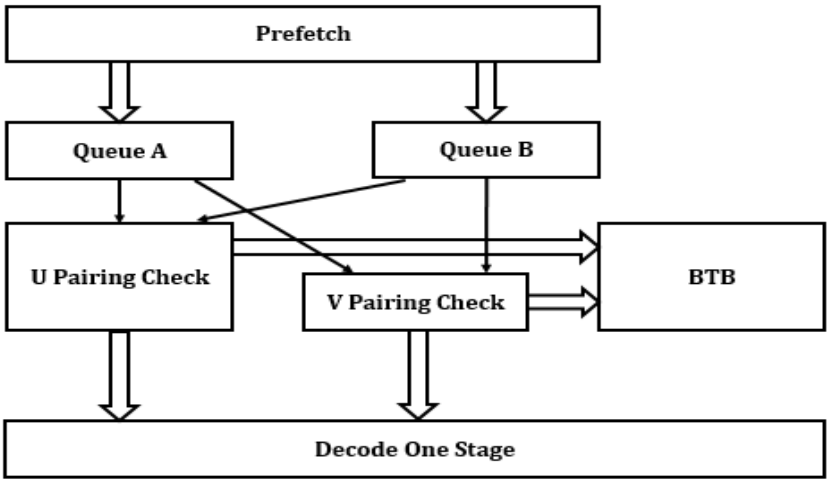


Figure 5.6: Decode One Stage.

C. DECODE TWO:

- In this stage, complete decoding is performed.
- It is responsible for address calculation.
- If the instruction try to access the address which is not allowed to be access by the instruction till the interrupt is generated.
- The responsibility of decode two is to serve interrupt.
- In this stage protection change is performed.

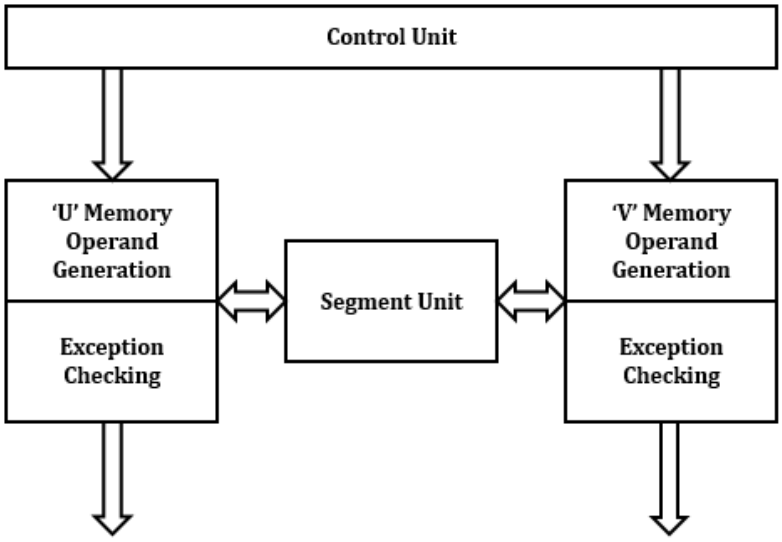


Figure 5.7: Decode Two Stage.

D. EXECUTION UNIT:

- It is used to access all the data from main memory and perform all the operation on that data.
- In this U pipeline has ALU and barrel shifter.
- V pipeline only have ALU.
- Due to this V pipeline cannot handle all instructions.

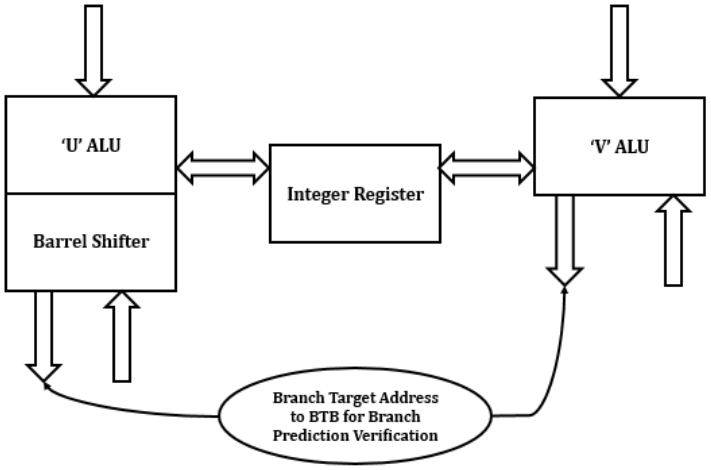


Figure 5.8: Execution Unit.

E. WRITE BACK STAGE:

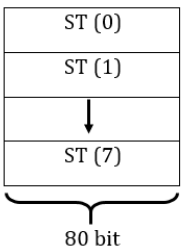
- Memory location are updated during this stage.
- In this stage, final result is written on integer register.

*** EXTRA QUESTIONS ***

Q15) FLOATING POINT PIPELINE STAGES.

ANS:

1. Pentium has on chip floating point unit.
2. FPU pipeline has 8 stages.
3. Most of the floating point instruction have to be in the U pipeline only and cannot be paired with integer instructions.
4. The first fourth stages are shared with integer pipeline and the next four are placed in FPU itself.
5. FPU has 8 register files.
6. Width of each register is of 80 bits.
7. They are known as ST (0), ST (1)…ST(7)



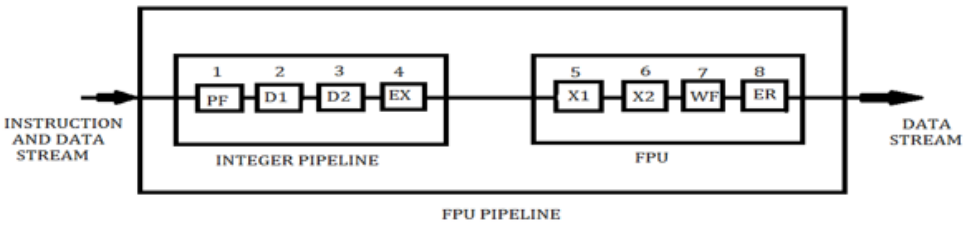


Figure 5.9: Floating Point Pipeline Stages.

STAGES:

I) PREFETCH STAGE:

- The 2 Prefetch queue A and B are available in integer pipelining stage.
- Each has 64 bytes.
- At a time only one queue is active.
- Each queue have 32 bytes buffer.
- It is used to fetch instructions from code cache and align the code to the initial byte of next instruction.
- Then place it in the active Prefetch Buffer.
- Two instructions from active Prefetch buffer reaches decode one stage of pipeline.

II) DECODE ONE:

- In this stage partial decoding is performed.
- It is responsible to identify whether 2 instructions or functions can be perform in parallel.
- The basic function of Decode 1 stage is:
  - Check pair ability.
  - Barrier shifter.
  - Branch prediction.
- The 2 instructions are checked whether they can form a pair.
- If they can form a pair, then both the instruction move in together.
- If they cannot, then instruction in the V pipeline of decode one stage is transferred to decode one stage of U pipeline.
- At the same time, first instruction in decode one stage of U pipeline is moved to decode two stage of U pipeline.

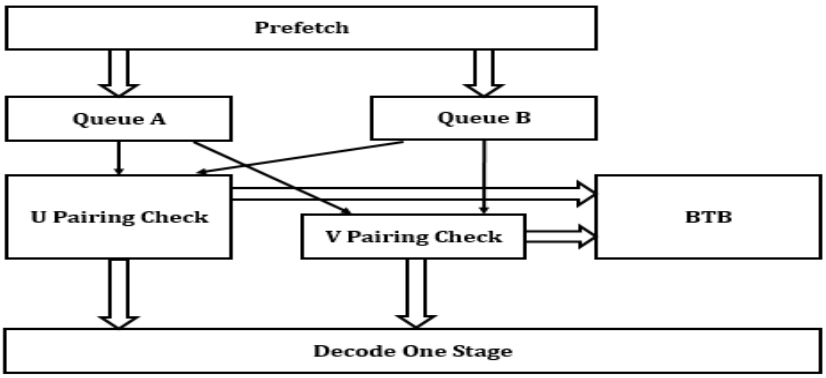


Figure 5.10: Decode One Stage.

III) DECODE TWO:

- In this stage, complete decoding is performed.
- It is responsible for address calculation.
- If the instruction try to access the address which is not allowed to be access by the instruction till the interrupt is generated.
- The responsibility of decode two is to serve interrupt.
- In this stage protection change is performed.

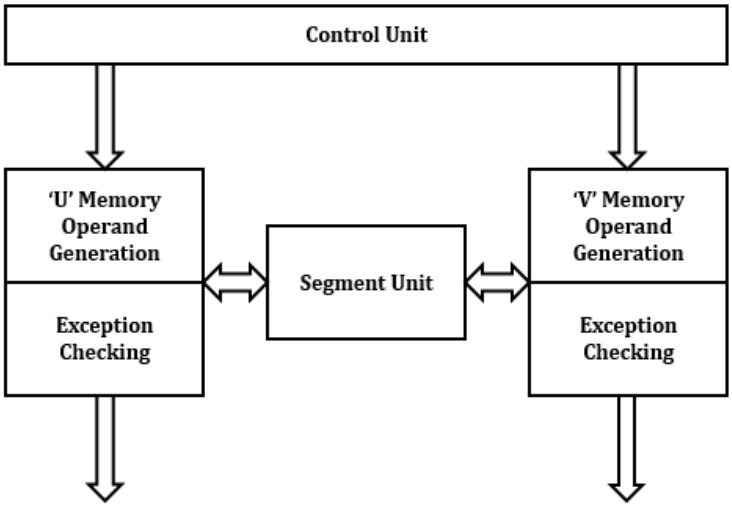


Figure 5.11: Decode Two Stage.

IV) EXECUTION UNIT:

- It is used to access all the data from main memory and perform all the operation on that data.
- In this U pipeline has ALU and barrel shifter.
- V pipeline only have ALU.
- Due to this V pipeline cannot handle all instructions.

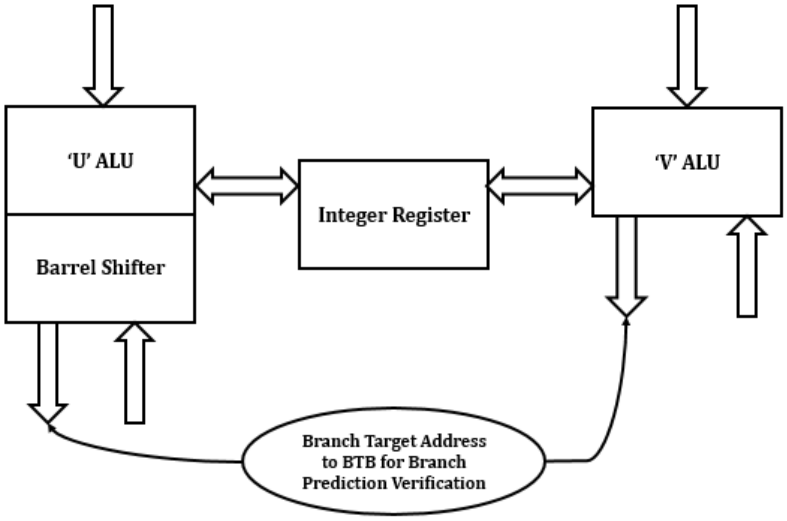


Figure 5.12: Execution Unit.

V) **Floating point execution one:**

- This stage is used to read the information from the memory or register and move them into floating point register.

VI) **Floating point execution two:**

- In this stage, all the floating point operation is performed.

VII) **Write floating point result:**

- In this stage the result is rounded of and written in the target of floating point register.
- If results are to be stored in memory, then transfer to EX stage is done.

VIII) **Error reporting:**

- If error is detected than it is reported.
- At the same time, FPU status word is updated.

Q16) Floating point instruction pairing.

Ans:

1. Majority of floating point instructions are executed sequentially in the U pipeline.
2. Only limited pairing is permitted.
3. To have pairing following conditions should be satisfied.
   - Both should be floating point instruction.
   - First instruction should be simple.
   - **Example:**
     - FLD single precision.
     - FLD double precision.
     - FLD ST (1).
     - FADD.
     - FSUB.
     - FMUL.
     - FDIV.
     - FCOM.
     - FTST.
     - FABS.
     - FCHS
   - Second instruction must be FXCH except case I and case II.
     - **CASE-I:** FMUL can be combined with FADD instruction.
     - **CASE-II:** FADD can be combined with FADD instruction.

CHAPTER-6: SUPERSPARC ARCHITECTURE

Q1) WRITE DOWN FEATURES OF SUPER SPARC PROCESSOR.

Q2) EXPLAIN THE ARCHITECTURE OF SUPER SPARC PROCESSOR WITH A NEAT DIAGRAM.

Q3) DRAW ARCHITECTURE OF SUPER SPARC PROCESSOR & EXPLAIN IN SHORT.

Q4) DRAW AND EXPLAIN ARCHITECTURE OF SUPERSPARC PROCESSOR.

ANS: [Q1 | 5M – DEC14], [Q2 | 10M – MAY15], [Q3 | 10M – DEC15] & [Q4 | 12M - MAY16]

FEATURES:

1. It is 32 bit Microprocessor.
2. It was developed by Sun Microsystems & Texas Instruments.
3. It is RISC Processor. (Reduced Instruction Set Computer)
4. It has 32 bit address bus.
5. It has 64 bit data path.
6. Physical memory is of 64 GB.
7. It includes Level 1 Code Cache of 20 KB.
8. It includes Level 1 Data Cache of 20 KB.
9. SuperSPARC Processor has Prefetch Queue of 16 bytes.
10. It has 3 issues Superscalar.
11. Super SPARC Architecture consists of Integer Unit, Floating Point Unit & Memory Management Unit.
12. It has 4 stage pipelining of 8 phases.
13. It uses technique of delayed branch.
14. It has 3.1 Million transistors.

BLOCK DIAGRAM:

Architecture of SuperSPARC consist of following blocks:

I) CACHE COHERENT BUS INTERFACE UNIT:

- The Sun SuperSPARC has 36 Address Lines & 64 Data Lines.
- The maximum physical memory accessed is 64 GB.

II) SPLIT CACHE:

- It consist of 20 KB Instruction Cache (Code Cache) with 5 way set associate mapping.
- It also consist of 16 KB Data Cache with 4 way set associate.
- Figure 6.1 shows the SuperSPARC Architecture.

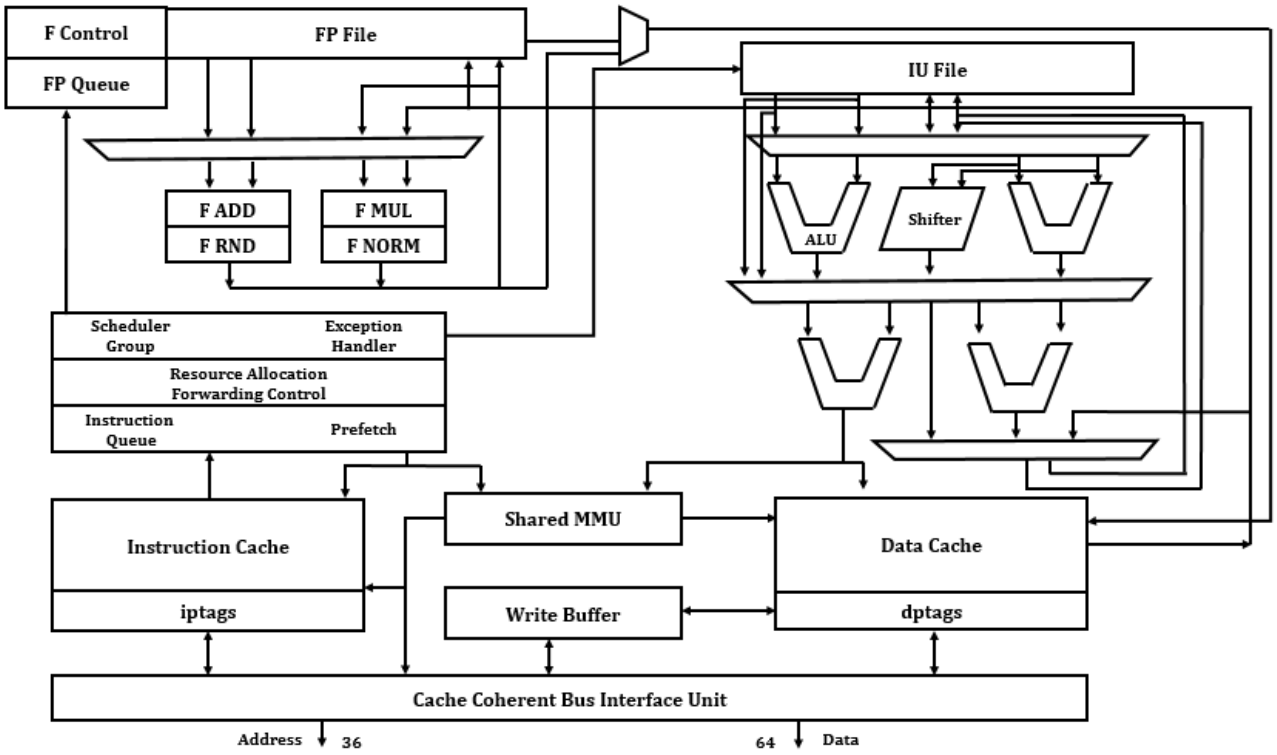


Figure 6.1: SuperSPARC Architecture.

III) PREFETCHER & PREFETCH QUEUE:

- The Prefetcher fetches the instructions from the instruction cache & place them in the instruction queue.
- Up to 3 instructions are issued to be executed simultaneously.

IV) IU FILE & FP FILE:

- This are the register file for Integer Execution Unit & Floating Point Execution Unit.
- There are 32 registers per window.

V) ALU: It is used to perform all the Arithmetic & Logical operations.

VI) CACHE & MMU OPERATION:

- SuperSPARC uses fully physically addressed caches.
- The cache is organized as associative cache where each set of cache is required to be equal to the minimum page size. (4 KB)
- The cache are accessed in Parallel with MMU.

VII) INTEGER CONTROL UNIT:

- The Integer unit dynamically selects "a group" of up to three instructions in each cycle.
- It allows combination of independent as well as dependent integer operation to be completed.

- The memory reference address is calculated here using register file ports & virtual address adder.
- The virtual address is then used by the cache and MMU to access load/store data.

VIII) FP UNIT:

- It is capable of executing single & double precision floating point operation.
- It provide 4 entry floating point instruction queue.

Q5) DATA TYPES SUPPORTED BY SPARC PROCESSOR.

Q6) EXPLAIN IN BRIEF DATA FORMAT SUPPORTED BY SUPERSPARC PROCESSOR.

ANS: [Q5 | 5M – DEC14 & DEC15] & [Q6 | 5M – MAY16]

Sun SPARC implements two types of data formats viz. Integer & Floating Point Data Formats.

I) INTEGER DATA FORMATS:

1. The Integer Data Formats are 8 bit (byte), 16 bit (half word), 32 bit (word) & 64 bit (double word).
2. These Data Formats are available as signed or unsigned.
3. The signed data formats have MSB as sign bit and remaining bits as magnitude.
4. While the unsigned data formats have all the bits as magnitude.

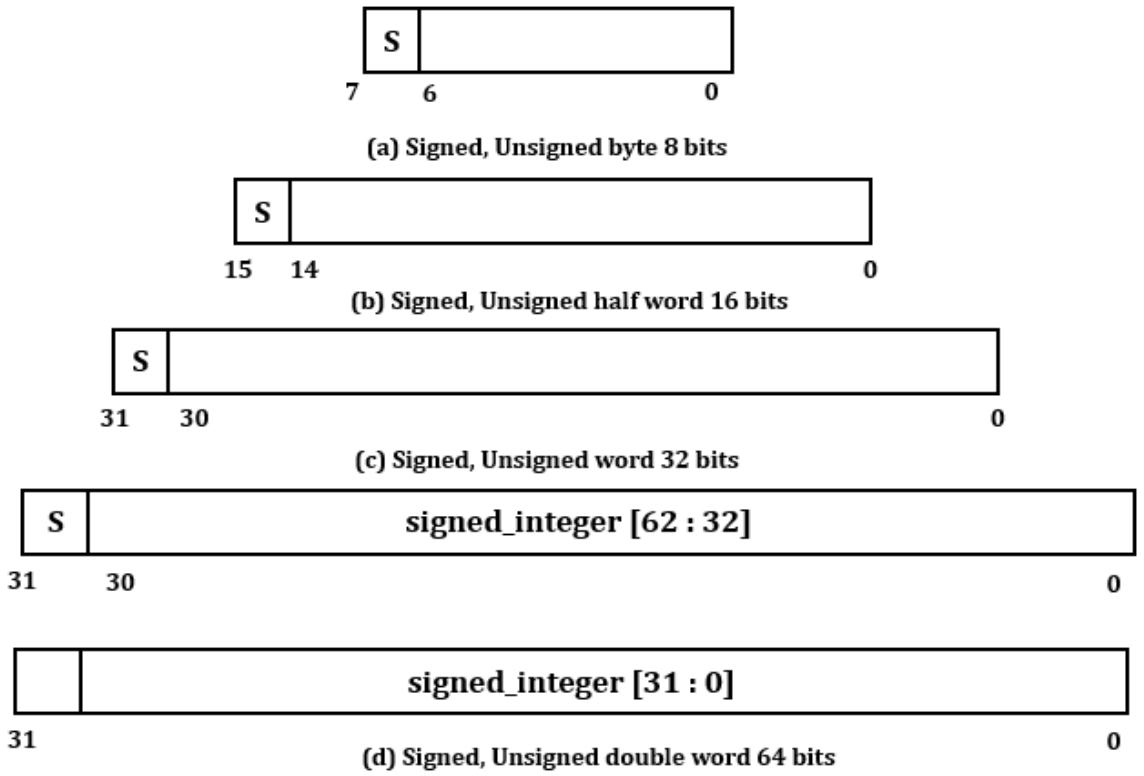


Figure 6.2: Integer Data Formats.

II) FLOATING POINT DATA FORMATS:

1. Beside the IEEE 754 standards of floating point representation, Sun SPARC also supports a higher precision called as Quad-Precision Format.
2. Sun SPARC supports Single Precision Format, Double Precision Format & Quad Precision Format.
3. Single Precision Format is a 32 bit IEEE 754 Floating Point Standard.
4. Double Precision Format is a 64 bit IEEE 754 Floating Point Standard.
5. Quad Precision Format is a 128 bit IEEE 754 Floating Point Standard.
6. The MSB of each of these formats is the sign bit.
7. The remaining bits are meant for fraction (mantissa) part.

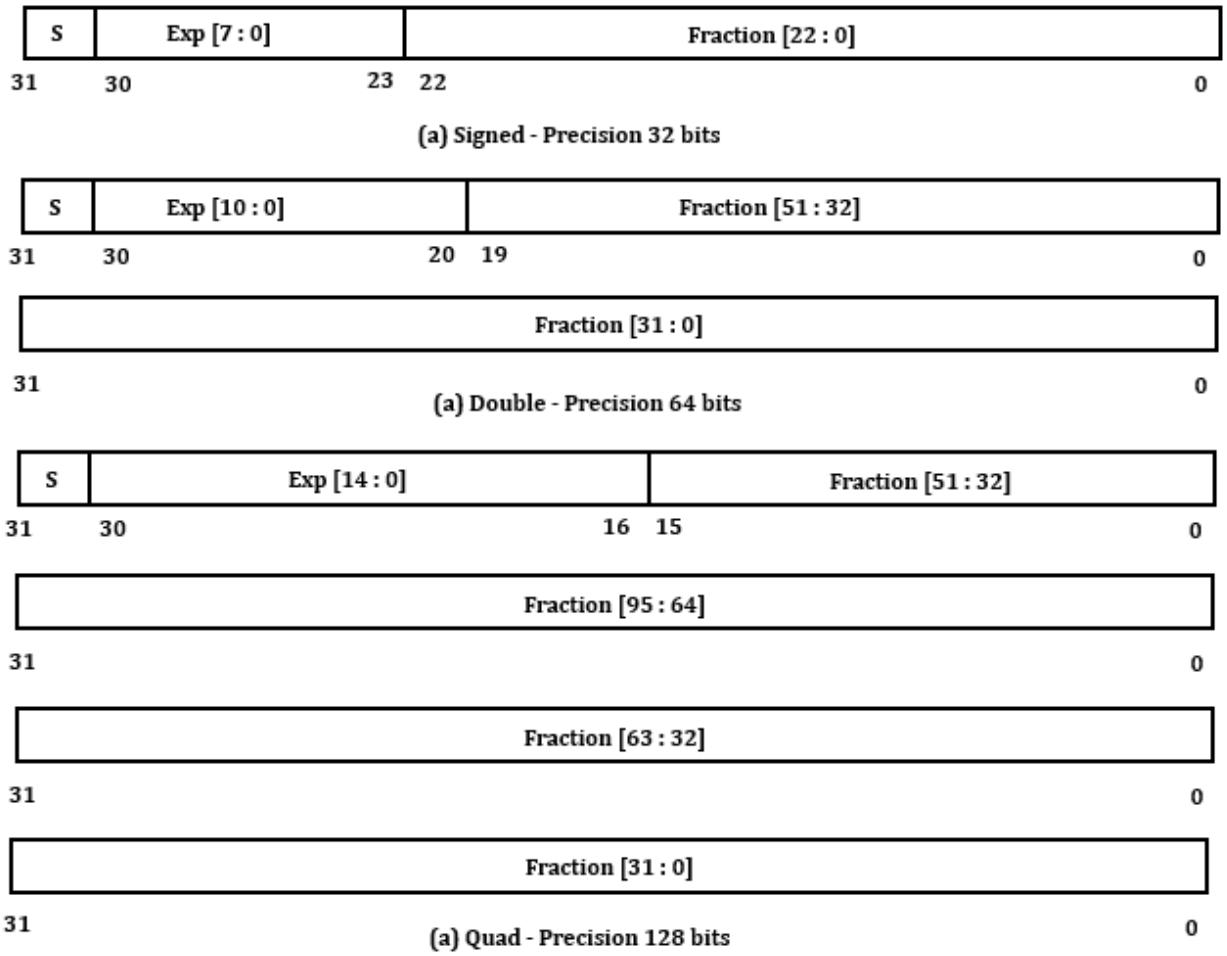


Figure 6.3: Floating Point Data Formats.

Q7) SUPERSPARC REGISTERS.

ANS: [5M –MAY16]

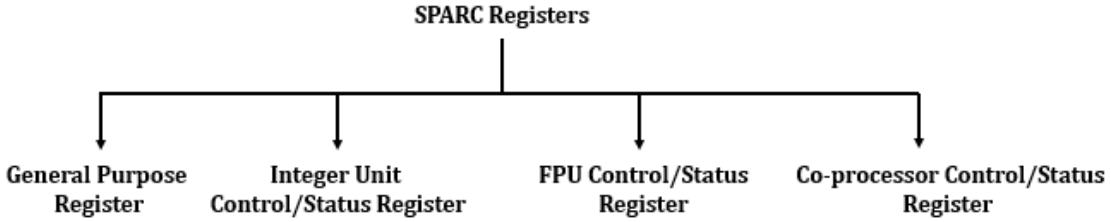


Figure 6.4: SuperSPARC Registers.

GENERAL PURPOSE REGISTER:

1. General Purpose Registers are also called as Working Data Registers.
2. It includes Integer Unit's 'r' registers & Floating Point Unit's 'f' registers.

INTEGER UNIT CONTROL/STATUS REGISTER:

It includes:

I) Processor State Register (PSR):

- It is 32 it register.
- It contains various fields that control and hold status information.
- The privileged instructions RDPSR & WRPSR are used to read & write the PSR respectively.

II) Window Invalid Mask (WIM):

- WIM register works with the register windowing mechanisms of the Sun SPARC Processor.
- It is used to determine a window overflow or underflow has taken place.

III) Trap Base Register (TBR):

- This register contains 3 fields that determines the address to which the control is to be transferred in case of occurrence of trap.
- The TBR can be written by the instruction WRTBR.

IV) Multiple/Divide Register (Y):

- It is 32 bit register.
- It contains most significant word of the double precision product of an integer multiplication & double precision dividend for an integer divide instruction.

V) Program Counter (PC, nPC):

- It is 32 bit register.
- PC contains the address of the instruction currently being executed by the Integer Unit.
- nPC holds the address of the next instruction to be executed.

VI) Ancillary State Register (ASR):

- SPARC provides 31 Ancillary Registers.
- ASR's numbered from 1 to 15 are reserved for future use by the architecture and should not be referenced by software.
- ASR's numbered from 16 to 31 are available for implementation-dependent uses such as timers, counters, diagnostic registers, self-test registers and trap-control register.

VII) IU Deferred-Trap Queue:

- The contents and operation of an IU Deferred-Trap Queue are implementation dependent and are not visible to user application programs.

FLOATING POINT UNIT CONTROL/STATUS REGISTER:

I) FPU 'f' Register:

- It is 32 bit register.
- A single 'f' register can hold one single-precision operand.

II) Floating Point State Register (FSR):

- FSR register field contain FPU Mode and Status Information.
- The FSR is read and written by the STFSR & LDFSR.

III) Floating Point Deferred Trap Queue (FQ):

- Floating Point Deferred Trap Queue, if present in implementation, contains sufficient state information to implement resumable, deferred floating point traps.
- If floating point instructions execute synchronously with integer instructions, provision of a floating point queue is optional.

CO-PROCESSOR CONTROL/STATUS REGISTER:

1. All of the co-processor data and control/status registers are optional.
2. Co-processor Control/Status Registers are implementation-dependent & there structures are not fixed.
3. It includes:
    a. Implementation-Dependent Coprocessor State Register (CSR).
    b. Implementation-Dependent Coprocessor Deferred Trap Queue (CQ).
4. These registers are accessed via load/store coprocessor and CPop1/CPop2 format instructions.

*** DEC – 2014 ***

Q1] (a) Draw and explain timing diagram for read operation in minimum mode of 8086. **[5]**
Ans: [Chapter - 1]
(b) Explain I/O related addressing mode of 8086. **[5]**
Ans: [Chapter - 2]
(c) Write down features of Super SPARC Processor. **[5]**
Ans: [Chapter - 6]
(d) Enlist the instruction pairing rules for U & V Pipeline in Pentium. **[5]**
Ans: [Chapter - 5]

Q2] (a) Explain address translation mechanism used in protocol mode of 80386. **[10]**
Ans: [Chapter - 4]
(b) Write assembly language program for 8086 to exchange contents of two memory blocks. **[10]**
Ans: [Chapter - 2]

Q3] (a) Design 8086 microprocessor based system with following specifications **[10]**
(i) Microprocessor 8086 working at 10 MHz in minimum mode.
(ii) 32 KB EPROM using 8 KB chips.
(iii) 16 KB SRAM using 4 KB chips.
Ans: [Chapter - 1]
(b) Explain how the flushing of pipeline problem is minimized in Pentium Arch. **[10]**
Ans: [Chapter - 5]

Q4] (a) Interface DMA controller 8237 with 8086 Microprocessor. Explain Different data transfer modes of 8237 DMA Controller. **[10]**
Ans: [Chapter - 3]
(b) Differentiate between real mode & protected mode. **[10]**
Ans: [Chapter - 4]

Q5] (a) Draw & Explain block diagram of 8259 PIC. **[10]**
Ans: [Chapter - 3]
(b) Draw a segment descriptor format & explain different fields. **[10]**
Ans: [Chapter - 4]

Q6] Write Short Notes on any four: **[20]**
(a) Code Cache Organization of Pentium.
Ans: [Chapter - 5]
(b) State the use of RF, TF, VM, NT, IOPL flag Bits.
Ans: [Chapter - 4]
(c) Data types supported by SPARC Processor.
Ans: [Chapter - 6]
(d) Advantages of memory segmentation in 8086.
Ans: [Chapter - 1]
(e) Maximum mode of 8086.
Ans: [Chapter - 1]
(f) Control Word Register of 8255.
Ans: [Chapter - 3]

*** MAY – 2015 ***

Q1] (a) Draw and Explain Timing Diagram for Write operation in minimum mode of 8086 **[5]**
Ans: [Chapter - 1]
(b) List operating modes of 8253. **[5]**
Ans: [Chapter - 3]
(c) Write down features of Pentium Processor. **[5]**
Ans: [Chapter - 5]
(d) Write the instruction issue algorithm used in Pentium. **[5]**
Ans: [Chapter - 5]

Q2] (a) Explain protection mechanism used in 80386. **[10]**
Ans: [Chapter - 4]
(b) Write assembly language program for 8086 to reverse a string of 10 characters. **[10]**
Ans: [Chapter - 4]

Q3] (a) Design 8086 microprocessor based system with following specifications. **[10]**
(i) Microprocessor 8086 working at 8 MHz in Maximum Mode.
(ii) 32 KB EPROM using 16 KB chips.
(iii) 16 KB SRAM using 8 KB chips.
Ans: [Chapter - 1]
(b) Explain branch prediction logic used in Pentium. **[10]**
Ans: [Chapter - 5]

Q4] (a) Explain Different data transfer modes of 8237 DMA Controller. **[5]**
Ans: [Chapter - 3]
(b) Explain Interfacing of 8259 with 8086 in minimum mode. **[5]**
Ans: [Chapter - 3]
(c) Differentiate between real mode & protected mode. **[10]**
Ans: [Chapter - 4]

Q5] (a) Compare 8086, 80386 & Pentium. **[10]**
Ans: [Chapter - 5]
(b) Draw Architecture of Super SPARC Processor & Explain in short. **[10]**
Ans: [Chapter - 6]

Q6] Write Short Notes on (Any 4): **[20]**
(a) Data cache organization of Pentium.
Ans: [Chapter - 5]
(b) State use of control flags of 8086.
Ans: [Chapter - 1]
(c) Data Types supported by SPARC Processor.
Ans: [Chapter - 6]
(d) Advantages of memory segmentation in 8086.
Ans: [Chapter - 1]
(e) Mode 1 of 8255 for input operation.
Ans: [Chapter - 3]

*** DEC – 2015 ***

Q1] (a) Write Short Note on 8288 Bus Controller. **[5]**
Ans: [Chapter - 1]
(b) Explain the following instructions in 8086: LAHF & STOSB **[5]**
Ans: [Chapter - 1]
(c) Design interfacing of 8282 latches to 8086 system. **[5]**
Ans: [Chapter - 3]
(d) Explain in brief Protection Mechanism in 80386DX Processor. **[5]**
Ans: [Chapter - 4]

Q2] (a) Explain Memory Management in details in 80386DX Processor. **[10]**
Ans: [Chapter - 4]
(b) Design 8086 based system with following specifications **[10]**
(i) 8086 in minimum mode working at 8 MHz.
(ii) 32 KB EPROM using 16 KB devices.
(iii) 64 KB SRAM using 32 KB devices.
Ans: [Chapter - 1]

Q3] (a) Explain with block diagram working of 8255 PPI. **[10]**
Ans: [Chapter - 3]
(b) What is segmentation? What are the advantages of segmentation? **[5]**
Ans: [Chapter - 1]
(c) Differentiate between minimum mode & maximum mode in 8086. **[5]**
Ans: [Chapter - 1]

Q4] (a) Explain branch prediction logic used in Pentium. **[10]**
Ans: [Chapter - 5]
(b) Compare Pentium 2, Pentium 3 & Pentium 4 Processor. **[10]**
Ans: [Chapter - 5]

Q5] (a) Explain different data transfer modes of 8237 DMA Controller. **[10]**
Ans: [Chapter - 3]
(b) Explain the architecture of Super SPARC Processor with a neat diagram. **[10]**
Ans: [Chapter - 6]

Q6] Write Short Notes on:
(a) 8087 Math Co-processor. **[5]**
Ans: [Chapter - 3]
(b) Generation of Reset signals in 8086 based system. **[5]**
Ans: [Chapter - 1]
(c) Comparative Study of multicore i3, i5 & i7 processors. **[5]**
Ans: [Chapter - 5]
(d) Mixed Language Programming. **[5]**
Ans: [Chapter - 2]

*** MAY – 2016 ***

Q1] Answer following questions in brief.

(a) Explain programming model of 8086. **[5]**

Ans: [Chapter - 1]

(b) Explain V86 mode of 80386DX. **[5]**

Ans: [Chapter - 4]

(c) Explain in brief pipeline stages on Pentium Processor. **[5]**

Ans: [Chapter - 5]

(d) Explain in brief data format supported by SuperSPARC Processor. **[5]**

Ans: [Chapter - 6]

Q2] (a) Explain memory segmentation with pros & cons. **[8]**

Ans: [Chapter - 1]

(b) Draw and explain the block diagram of 8255. Also Explain Different Operating Modes of 8255. **[12]**

Ans: [Chapter - 3]

Q3] (a) Design 8086 based minimum mode system for following requirement: **[12]**

(i) 256 KB of RAM using 64 KB x 8 Bit device.

(ii) 128 KB of RAM using 64 KB x 8 Bit device.

(iii) Three 8 Bit Parallel Ports using 8255.

(iv) Support for 8 interrupts.

Ans: [Chapter - 1]

(b) Explain in brief, cache organization of Pentium processor. **[8]**

Ans: [Chapter - 5]

Q4] (a) Draw and explain architecture of SuperSPARC Processor. **[12]**

Ans: [Chapter - 6]

(b) Discuss in brief protection mechanism of 80386DX. **[08]**

Ans: [Chapter - 4]

Q5] (a) Draw and explain architecture of Pentium Processor. **[10]**

Ans: [Chapter - 5]

(b) Draw timing diagram of read operation on 8086 based system. **[10]**

Ans: [Chapter - 1]

Q6] Write Short Notes on:

(a) 8089 I/O Processor. **[5]**

Ans: [Chapter - 3]

(b) Comparison between i5 & i7. **[5]**

Ans: [Chapter - 5]

(c) SuperSPARC Registers. **[5]**

Ans: [Chapter - 6]

(d) 8259 – PIC. **[5]**

Ans: [Chapter - 3]