UNIT III 8086 Microprocessor

Lecture 1

Features of 8086 Microprocessor (RGPV 2013)

• The 8086, announced in 1978, was the first 16-bit microprocessor introduced by Intel Corporation.

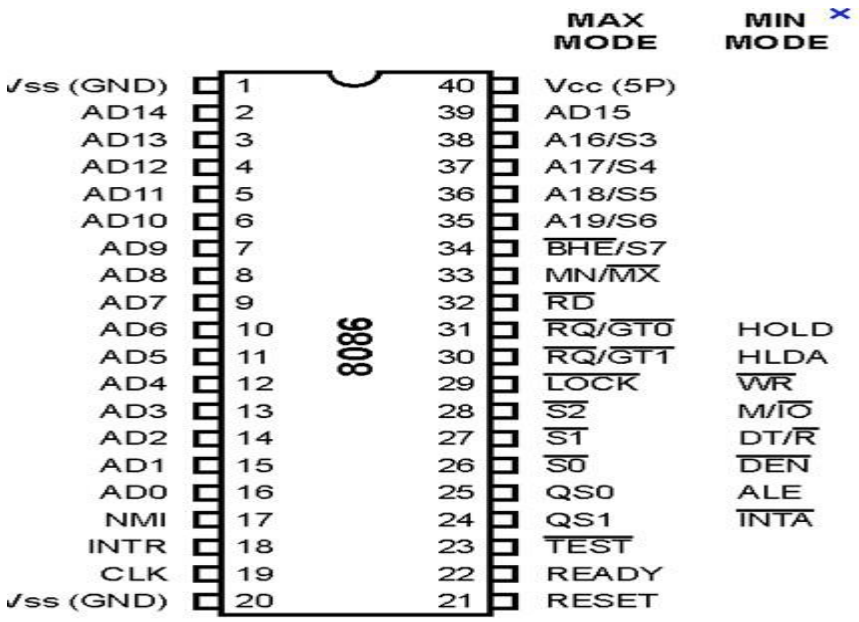
- 1**) 8086 has 16-bit ALU; this means 16-bit numbers are directly processed by 8086.**
- **2) It has 16-bit data bus, so it can read data or write data to memory or I/O ports either 16 bits or 8 bits at a time.**
- **3) It has 20 address lines, so it can address up to 2^{20} i.e. 1048576 = 1Mbytes of memory (words i.e. 16 bit numbers are stored in consecutive memory locations). Due to the 1Mbytes memory size multiprogramming is made feasible as well as several multiprogramming features have been incorporated in 8086 design.**
- **4) 8086 includes few features, which enhance multiprocessing capability (it can be used with math coprocessors like 8087, I/O processor 8089 etc.**
- **5) Operates on +5v supply and single phase (single line) clock frequency.(Clock is generated by separate peripheral chip 8284).**
- **6) 8086 comes with different versions. 8086 runs at 5 MHz, 8086-2 runs at 8 MHz, 8086-1 runs at 10 MHz**
- **7) It comes in 40-pin configuration with HMOS technology having around 20,000 transistors in its circuitry.**
- **8) It has multiplexed address and data bus like 8085 due to which the pin count is reduced considerably**
- **9) Higher Throughput (Speed)(This is achieved by a concept called pipelining)**
- **But the concept of 8086's principles and structures is very useful for understanding other advanced Intel microprocessors**

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Compare the basic features of 8086 & 80386?	JUNE 2013	10

UNIT III 8086 Microprocessor

Lecture 1,2

Pin Description



The Microprocessor 8086 is a 16-bit CPU available in different clock rates and packaged in a 40 pin DIP or plastic package. The 8086 operates in single processor or multiprocessor configuration to achieve high performance. The pins serve a particular function in minimum mode (single processor mode) and other function in maximum mode configuration (multiprocessor mode).
The 8086 signals can be categorized in three groups. The first are the signal having common functions in minimum as well as maximum mode.
 The second are the signals which have special functions for minimum mode
 The third are the signals having special functions for maximum mode.

1.AD15-AD0
The following signal descriptions are common for both modes.
 AD15-AD0:- These are the time multiplexed memory I/O address and data lines.
– Address remains on the lines during T1 state, while the data is available on the data bus during T2, T3, Tw and T4. These lines are active high and float to a tristate during interrupt acknowledge and local bus hold acknowledge cycles.

2.A19-A16

A19/S6,A18/S5,A17/S4,A16/S3 : These are the time multiplexed address and status lines. During T1 these are the most significant address lines for memory operations.

- During I/O operations, these lines are low.
- During memory or I/O operations, status information is available on those lines for T2,T3,Tw and T4.
- The status of the interrupt enable flag bit is updated at the beginning of each clock cycle.

3.Segment access

The S4 and S3 combinely indicate which segment register is presently being used for memory accesses as in below fig.

- These lines float to tri-state off during the

local bus hold acknowledge. The status line S6 is always low.

- The address bit are separated from the status bit using latches controlled by the ALE signal

S4	S3	Indication
0	0	Alternate Data
0	1	Stack
1	0	Code or None
1	1	Data
0	0	Whole word
0	1	Upper byte from or to even address
1	0	Lower byte from or to even address

Segment access

4.BHE/S7

BHE/S7:- The bus high enable is used to indicate the transfer of data over the higher order (D15-D8) data bus . It goes low for the data transfer over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals. BHE is low during T1 for read, write and interrupt acknowledge cycles, whenever a byte is to be transferred on higher byte of data bus. The status information is available during T2, T3 and T4. The signal is active low and tri-stated during hold. It is low during T1 for the first pulse of the interrupt acknowledges cycle.

5.Rd, READY

RD – Read: - This signal on low indicates the peripheral that the processor is performing memory or I/O read operation. RD is active low and shows the state for T2, T3, Tw of any read cycle. The signal remains tristated during the hold acknowledge.

READY: This is the acknowledgement from the slow device or memory that they have completed the data transfer. The signal made available by the devices is synchronized by the 8284A clock

generator to provide ready input to the 8086. the signal is active high.

6.INTR

INTR-Interrupt Request: This is a triggered input. This is sampled during the last clock cycles of each instruction to determine the availability of the request. If any interrupt request is pending, the processor enters the interrupt acknowledge cycle. This can be internally masked by resulting the interrupt enable flag. This signal is active high and internally synchronized.

7.TEST,CLK

TEST: This input is examined by a 'WAIT' instruction. If the TEST pin goes low, execution will Continue, else the processor remains in an idle state. The input is synchronized internally during each clock cycle on leading edge of clock.

 CLK- Clock Input: The clock input provides the basic timing for processor operation and bus control activity. It's an asymmetric square wave with 33% duty cycle.

Minimum Mode Pins

The following 8 pins function descriptions are for the 8086 in minimum mode; MN/MX= 1.
The corresponding 8 pins function descriptions for maximum mode is explained later.

M/ IO **(O): Status line**

This pin is used to distinguish a memory access or an I/O accesses. When this pin is Low, it accesses I/O and when high it access memory. M / IO become valid in the T4 state preceding a bus cycle and remain valid until the final T4 of the cycle. M/ IO floats to 3 - state OFF during local bus "hold acknowledge".

WR **(O): Write**

Indicates that the processor is performing a write memory or write IO cycle, depending on the state of the M / IOsignal. WR is active for T2, T3 and Tw of any write cycle. It is active LOW, and floats to 3-state OFF during local bus "hold acknowledge ".

INTA **(O): Interrupt Acknowledge**

It is used as a read strobe for interrupt acknowledge cycles. It is active LOW during T2, T3, and T4 of each interrupt acknowledge cycle.

ALE (O): Address Latch Enable

ALE is provided by the processor to latch the address into the 8282/8283 address latch. It is an active high pulse during T1 of any bus cycle. ALE signal is never floated.

DT/ *R* **(O): DATA Transmit/Receive**

In minimum mode, 8286/8287 transceiver is used for the data bus. DT/ *R* is used to control the direction of data flow through the transceiver. This signal floats to tri-state off during local bus "hold acknowledge".

DEN **(O): Data Enable**

It is provided as an output enable for the 8286/8287 in a minimum system which uses the transceiver. DEN is active LOW during each memory and IO access. It will be low beginning with T2 until the middle of T4, while for a write cycle, it is active from the beginning of T2 until the middle of T4. It floats to tri-state off during local bus "hold acknowledge".

HOLD & HLDA (I/O): Hold and Hold Acknowledge

Hold indicates that another master is requesting a local bus "HOLD". To be acknowledged, HOLD must be active HIGH. The processor receiving the "HOLD" request will issue HLDA (HIGH) as an acknowledgement in the middle of the T1-clock cycle. Simultaneous with the issue of HLDA, the processor will float the local bus and control lines. After "HOLD" is detected as being Low, the processor will lower the HLDA and when the processor needs to run another cycle, it will again drive the local bus and control lines.

Maximum Mode

The following pins function descriptions are for the 8086/8088 systems in maximum mode (*i.e.* MN/MX= 0). Only the pins which are unique to maximum mode are described below.

S2, S1, S0 (O): Status Pins

These pins are active during T4, T1 and T2 states and is returned to passive state (1,1,1 during T3 or Tw (when ready is inactive). These are used by the 8288 bus controller to generate all memory and I/O operation) access control signals. Any change by S2, S1, S0 during T4 is used to indicate the beginning of a bus cycle.

S2	S1	S0	Characteristics
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive State

Status Lines

QS0, QS1 (O): Queue – Status

Queue Status is valid during the clock cycle after which the queue operation is performed. QS0, QS1 provide status to allow external tracking of the internal 8086 instruction queue. The

condition of queue status is shown in table 4. Queue status allows external devices like In-circuit Emulators or special instruction set extension co-processors to track the CPU instruction execution. Since instructions are executed from the 8086 internal queue, the queue status is presented each CPU clock cycle and is not related to the bus cycle activity.

This mechanism allows

(1) A processor to detect execution of a ESCAPE instruction which directs the coprocessor to perform a specific task and

(2) An in-circuit Emulator to trap execution of a specific memory location.

QS1	QS1	Characteristics
0	0	No operation
0	1	First byte of opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

LOCK (O)

It indicates to another system bus master, not to gain control of the system bus while LOCK is active Low. The LOCK signal is activated by the "LOCK" prefix instruction and remains active until the completion of the instruction. This signal is active Low and floats to tri-state OFF during 'hold acknowledge".

Example:

LOCK XCHG reg., Memory ; Register is any register and memory 0 *GT* ; is the address of the semaphore.

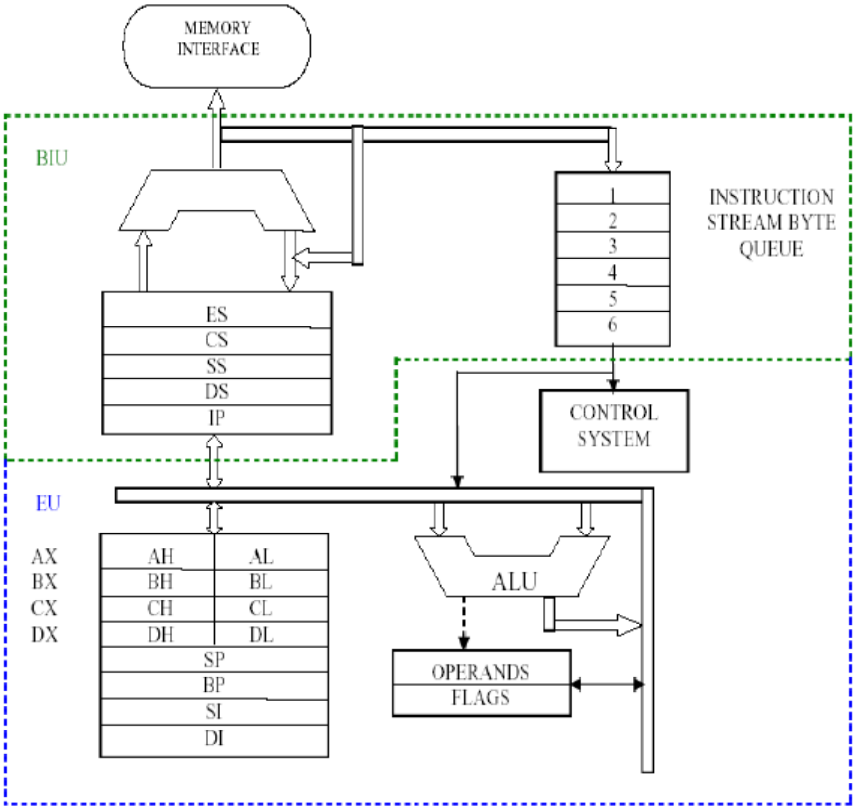
RQ / 0 GT and RQ / 1 GT (I/O): Request/Grant

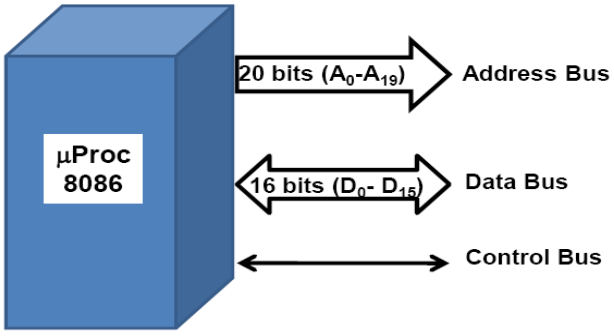
These pins are used by other processors in a multi processor organization. Local bus masters of other processors force the processor to release the local bus at the end of the processors current bus cycle. Each pin is bi-directional and has an internal pull up resistors. Hence they may be left un-connected.

UNIT III 8086 Microprocessor

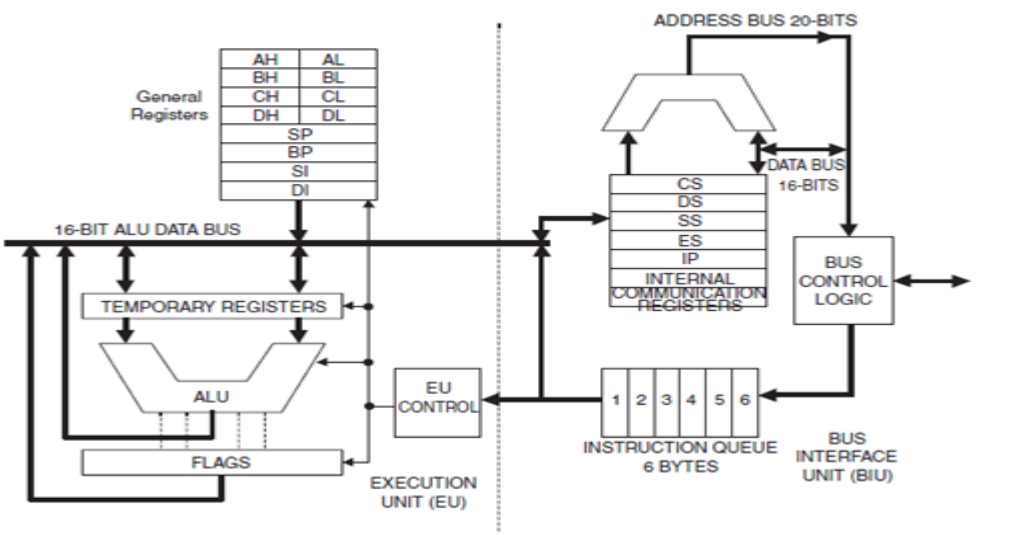
Lecture 3

8086 Architecture (RGPV 2013)





In 8086-based System ,we can access220=1M memory locations at most



Bus Interface Unit (BIU)

The function of BIU is to:

1. Fetch the instruction or data from memory.
2. Write the data to memory.
3. Write the data to the port.
4. Read data from the port.

Instruction Queue

1. To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory.
2. All six bytes are then held in first in first out 6 byte register called instruction queue.
3. Then all bytes have to be given to EU one by one.
4. This pre fetching operation of BIU may be in parallel with execution operation of EU, which improves the speed execution of the instruction.

Execution Unit (EU)

The functions of execution unit are:

1. To tell BIU where to fetch the instructions or data from.
2. To decode the instructions.
3. To execute the instructions.

The EU contains the control circuitry to perform various internal operations. A decoder in EU decodes the instruction fetched memory to generate different internal or external control signals required to perform the operation. EU has 16-bit ALU, which can perform arithmetic and logical operations on 8-bit as well as 16-bit.

General Purpose Registers of 8086

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.

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

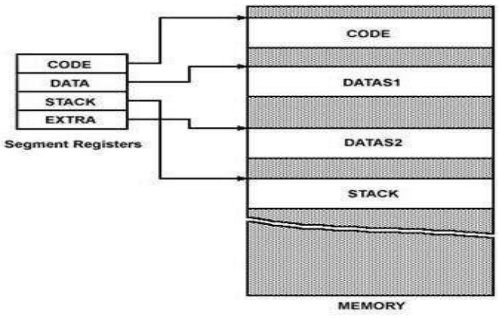
2. BX Register: This register is mainly used as a base register. It holds the starting base location of a memory region within a data segment.

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

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

Segment Registers

Additional registers called segment registers generate memory address when combined with other in the microprocessor. In 8086 microprocessor, memory is divided into 4 segments as follow:



Memory Segments of 8086

1. 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.

2. 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.

3. Stack Segment (SS): SS defined the area of memory used for the stack.

4. Extra Segment (ES): ES is additional data segment that is used by some of the string to hold the destination data.

Flag Registers of 8086

Flag Register of 8086

Flags Register determines the current state of the processor. They are modified automatically by CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program. 8086 has 9 flags and they are divided into two categories:

1. Conditional Flags
2. Control Flags

Conditional Flags

Conditional flags represent result of last arithmetic or logical instruction executed. Conditional flags are as follows:

1. **Carry Flag (CF):** This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.
2. **Auxiliary Flag (AF):** If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AF flag is set i.e. carry given by D3 bit to D4 is AF flag. This is not a general-purpose flag, it is used internally by the processor to perform Binary to BCD conversion.
3. **Parity Flag (PF):** This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity Flag is reset.
4. **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.
5. **Sign Flag (SF):** In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.
6. **Overflow Flag (OF):** It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine.

Control Flags

Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

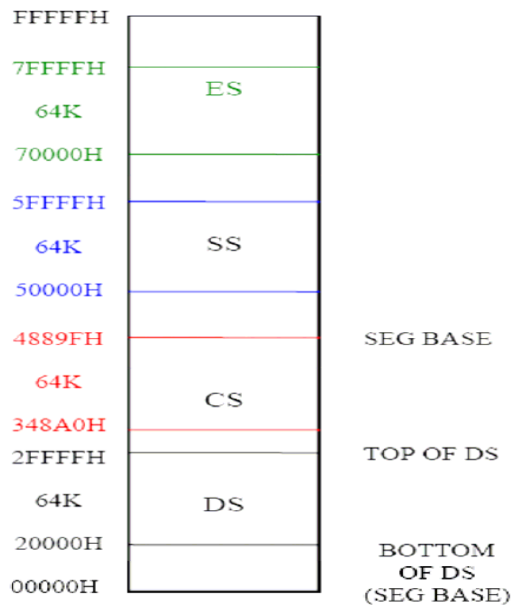
1. Trap Flag (TP):
a. It is used for single step control.
b. It allows user to execute one instruction of a program at a time for debugging.
c. When trap flag is set, program can be run in single step mode.
2. Interrupt Flag (IF):
a. It is an interrupt enable/disable flag.
b. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled.
c. It can be set by executing instruction sit and can be cleared by executing CLI instruction.
3. Direction Flag (DF):
a. It is used in string operation.
b. If it is set, string bytes are accessed from higher memory address to lower memory address.
c. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Draw & Discuss the internal block diagram of 8086	JUNE 2013	10

UNIT III 8086 Microprocessor

Lecture 4

Memory Segmentation



Memory Segmentation

Advantages of memory segmentation

- **Allow the memory capacity to be 1Mb even though the addresses associated with the individual instructions are only 16 bits wide.**
- **Facilitate the use of separate memory areas for the program, its data and the stack.**
- **Permit a program and/or its data to be put into different areas of memory each time the program is executed.**
- **Multitasking becomes easy.**

Generation of 20 bit physical address

The 20-bit Physical address is often represented as:

Segment Base : Offset OR CS : IP

CS 3 4 8 0 0 →Implied Zero (from shift Left)

+IP 1 2 3 4

----

3 5 A3 4 H

UNIT III 8086 Microprocessor

Lecture 5

Memory Read and Write Bus Cycle of 8086

BUS CYCLE AND TIME STATES:-

A bus cycle defines the sequence of events when the MPU communicates with an external device, which starts with an address being output on the system bus followed by a read or write data transfer.

Types of bus cycles:

- **Memory Read Bus Cycle**
- **Memory Write Bus Cycle**
- **Input/output Read Bus Cycle**
- **Input/output Write Bus Cycle**

The bus cycle of the 8086 microprocessor consists of at least four clock periods. These four time states are called T1, T2, T3 and T4.

MEMORY READ AND WRITE BUS CYCLES

During period T1

1. The 8086 outputs the 20-bit address of the memory location to be accessed on its multiplexed address/data bus. BHE is also output along with the address during T1.
2. At the same time a pulse is also produced at ALE. The trailing edge or the high level of this pulse is used to latch the address in external circuitry.
3. Signal M/IO is set to logic 1 and signal DT/R is set to the 0 logic level and both are maintained throughout all four periods of the bus cycle.

Beginning with period T2

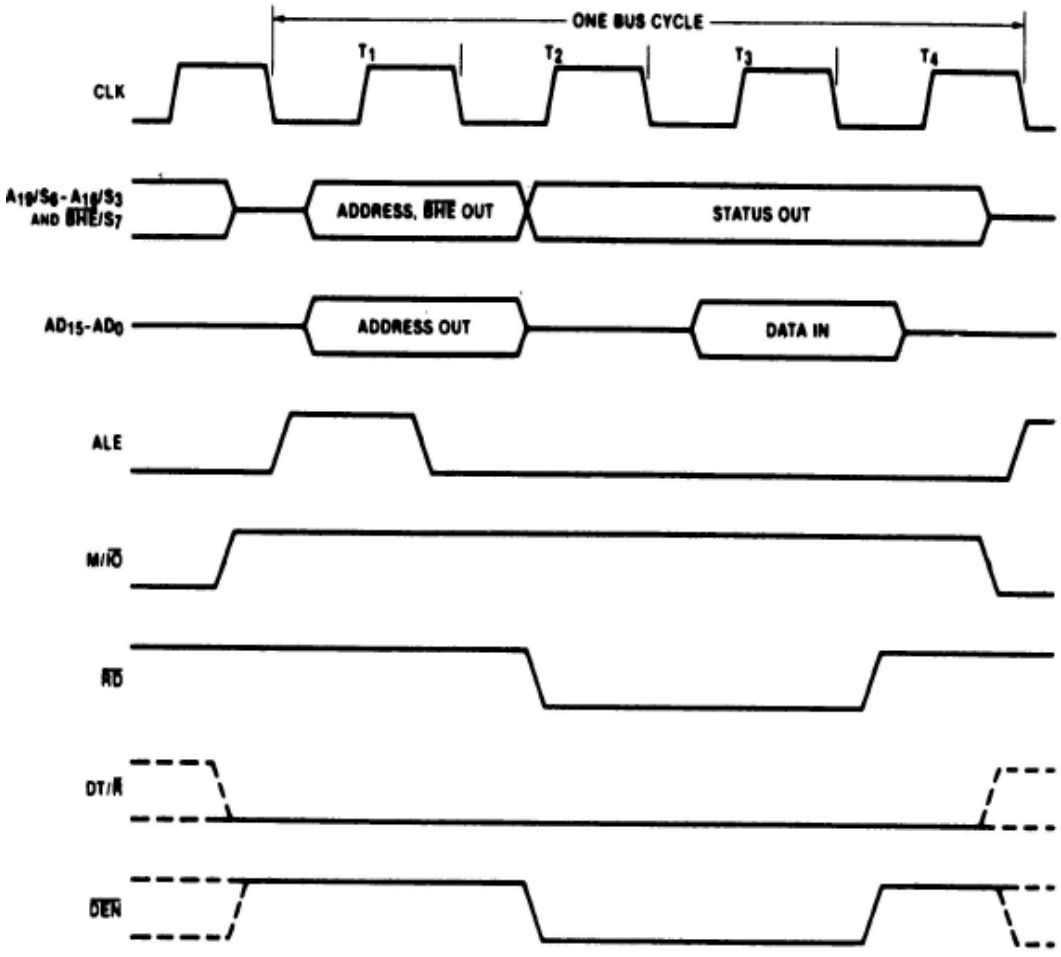
1. Status bits S3 through S6 are output on the upper four address bus lines. This status information is maintained through periods T3 and T4.
2. On the other hand, address/data bus lines AD0 through AD7 are put in the high-Z state during T2.
3. Late in period T2, RD is switched to logic 0. This indicates to the memory subsystem that a read cycle is in progress. DEN is switched to logic 0 to enable external circuitry to allow the data to move from memory onto the microprocessor's data bus.

During period T3

1. The memory must provide **valid data** during **T3** and maintain it until after the processor terminates the read operation. The data read by the 8086 microprocessor can be carried over all **16 data bus** lines

During T4

1. The 8086 switches RD to the inactive 1 logic level to terminate the read operation. DEN returns to its inactive logic level late during T4 to disable the external circuitry.



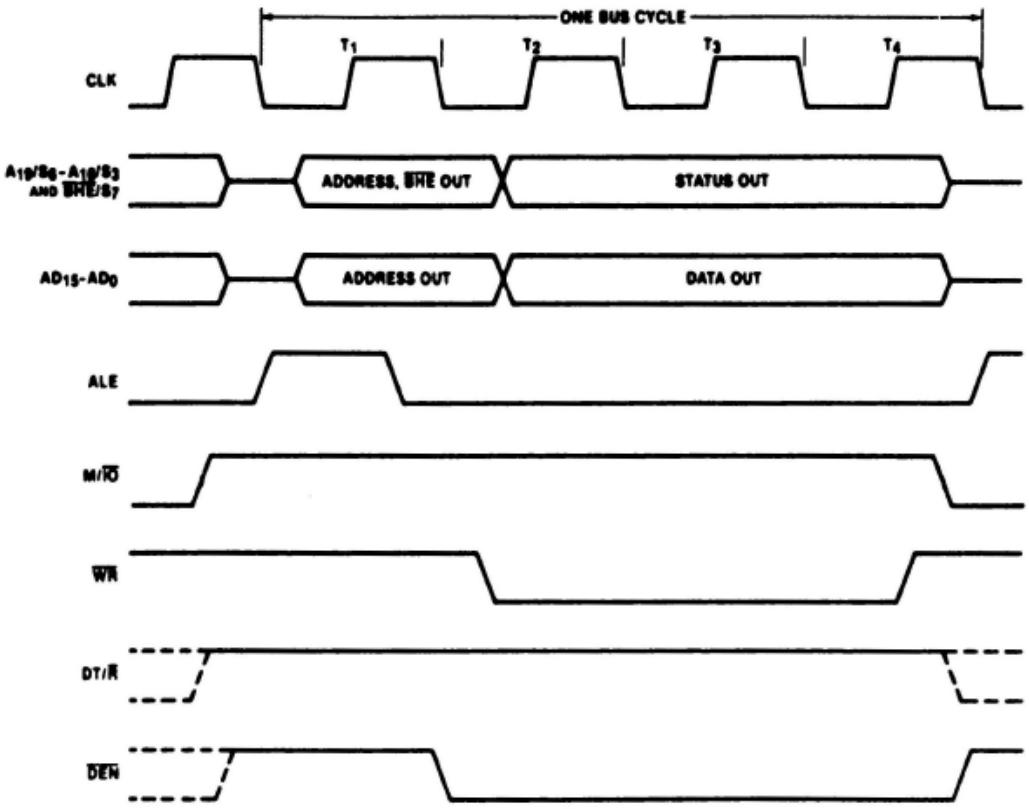
Minimum-mode memory read bus cycle of the 8086.

During period T1

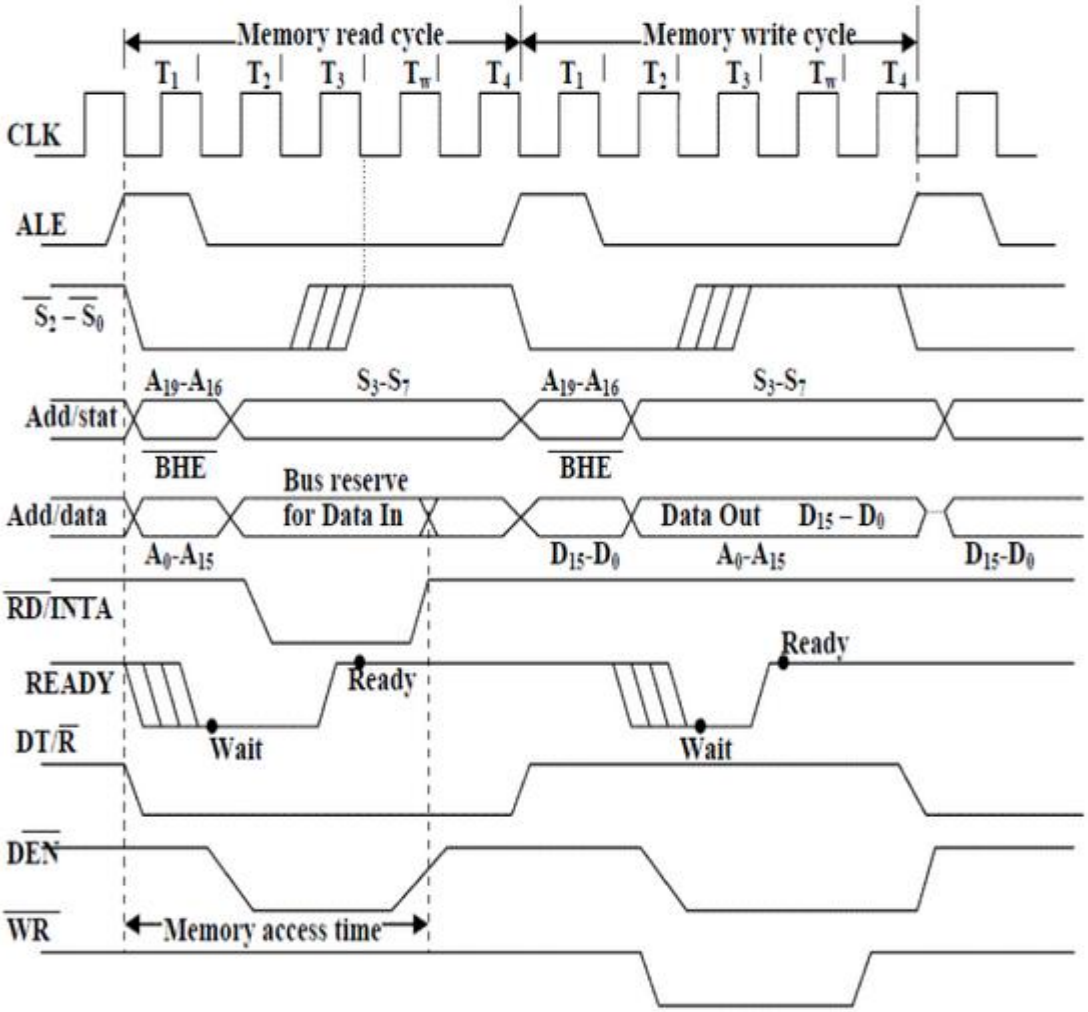
1. The address along with BHE is output and latched with the ALE pulse.
2. M/IO is set to logic 1 to indicate a memory cycle.
3. However, this time DT/R is switched to logic 1. This signals external circuits that the 8086 is going to transmit data over the bus.

Beginning with period T2

1. WR is switched to logic 0 telling the memory subsystem that a write operation is to follow.
2. The 8086 puts the data on the bus late in T2 and maintains the data valid through T4. Data will be carried over all 16 data bus lines.
3. DEN enables the external circuitry to provide a path for data from the processor to the memory.



Minimum-mode memory write bus cycle of the 8086.



Memory Read Write Cycle of 8086

UNIT III 8086 Microprocessor

Lecture 6,7

INSTRUCTION SET OF 8086 (RGPV 2014)

- Classified into 7 categories:
- 1] Data Transfer
- 2] Arithmetic
- 3] Logical
- 4] Control
- 5]Processor Control Instructions
- 6] String Manipulation
- 7] Interrupt Control

Data Transfer Instructions

(Note : Data Transfer Instructions do not affect any flags)

- 1] MOV dest, src

Note that source and destination cannot be memory location. Also source and destination must be same type.

- 2] PUSH Src: Copies word on stack.
- 3] POP dest: Copies word from stack into dest. Reg.
- 4] IN acc, port : Copies 8 or 16 bit data from port to accumulator.
- a) Fixed Port
- b) Variable Port
- 5] OUT port, acc
- 6] LES Reg, Mem: Load register and extra segment register with words from memory.
- 7] LDS Reg,Mem: Load register and data segment register with words from memory.
- 8] LEA Reg,Src: load Effective address.(Offset is loaded in specified register)
- 9] LAHF: Copy lower byte of flag register into AH register.
- 10] SAHF: Copy AH register to lower byte of flag
- 11] XCHG dest, src: Exchange contents of source and destination.
- 12] XLAT: Translate a byte in AL.

This instruction replaces the byte in AL with byte pointed by BX.To point desired byte in look up table instruction adds contains of BX with AL (BX+ AL). Goes to this location and loads into AL.

Arithmetic Instructions

- 1]ADD dest,src
- 2] ADC dest,src: Add with carry
- 3] AAA : ASCII adjust after addition.

We can add two ASCII numbers directly and use AAA after addition so as to get result directly in BCD. (Works with AL only)

- 4] DAA : Decimal adjust accumulator. (Works with AL only)
- 5] SUB dest, src
- 6] SBB dest, src: Subtract with borrow.
- 7] AAS: ASCII adjust for subtraction .(same as AAA and works with AL only)
- 8] DAS : Decimal adjust after Subtraction.(works with AL only)
- 9] MUL src
- 10] IMUL src: Multiplication of signed byte.
- 11] AAM: BCD adjust after multiply. (works with AL only)
- 12]DIV src

If any one attempts to divide by 0 , then ?

- 13] IDIV: Division of signed numbers
- 14]AAD: BCD to Binary convert before Division.
- 15] DEC dest
- 16] INC dest
- 17] CWD: Convert signed word to signed double word.
- 18] CBW : Convert signed byte to signed word.
  (CBW and CWD works only with AL, AX and DX)
- 19] NEG dest: Forms 2's complement.

Logical Instructions

- 1] AND dest, src
- 2] NOT dest: Invert each bit in destination
- 3] OR dest, src
- 4] XOR dest, src
- 5] RCL dest, count : Rotate left through Carry

Rotate as many times as directly specified in the instruction. For more no.of rotations, count can be specified in CL register.

- 6] RCR dest, count : Rotate right through carry
- 7] ROL dest, count : Rotate left (into carry as well as into LSB)
- 8] ROR dest, Count : Rotate left (into carry as well as into MSB)
- 9] SAL/ SHL dest, count : Shift left and append 0s on right.
- 10] SAR dest, count : Shift right retain a copy of the S-bit and shift all bits to right.
- 11]SHR dest, count : Shift right append 0s on left
- 12] TEST dest, src: AND logically, updates flags but source and dest are unchanged.
- 13] CMP dest, src
- CF, ZF and SF are used

Ex. CMP CX,BX

	CF	ZF	SF
CX = BX	**0**	**1**	**0**
CX> BX	**0**	**0**	**0**
CX<BX	**1**	**0**	**1**

CONTROL TRANSFER INSTRUCTIONS

- 1]CALL : Call a procedure
  Two types of calls:
    i) Near Call (Intrasegment)
    ii) Far Call (Intersegment)
- 2] RET : Return execution from procedure
- 3] JMP : Unconditional Jump to specified destination. Two types near and Far
- 4] JA / JNBE: Jump if above / Jump if not below

The terms above and below are used when we refer to the magnitude of Unsigned number .Used normally after CMP.

- 5] JAE / JNB / JNC
- 6] JB / JC / JNAE
- 7] JBE / JNA
- 8] JE/ JZ
- 9] JCXZ: Jump if CX is Zero.
- 10] JG / JNLE: Jump if Greater /Jump if NOT less than or equal.

The term greater than or less than is used in connection with two signed numbers.

- 11] JGE / JNL:
- 12] JL / JNGE :
- 13] JLE / JNG :
- 14]JNE / JNZ :
- 15] JNO : Jump if no overflow
- 16] JNS : Jump if no sign
- 17] JS
- 18] JO
- 19] JNP / JPO
- 20] JP / JPE

In all above conditional instructions the destination of jump is in the range of -128 to + 127 bytes from the address after jump.

- 21] LOOP: Loop to the specified label if CX is not equal to Zero.

The count is loaded in CX reg. Every time LOOP is executed, CX is automatically decremented - used in delay programs

- 22] LOOPE/ LOOPZ: Loop while CX is not equal to zero and ZF = 1.
- 23] LOOPNE / LOOPNZ: Loop while CX not equal to zero and ZF = 0.

In all above LOOP instructions the destination of jump is in the range of -128 to + 127 bytes from the address after LOOP.

PROCESSOR CONTROL

- 1] CLC: Clear Carry flag.
- 2] STC :Set carry Flag
- 3] CMC :Complement Carry Flag
- 4] CLD: Clear Direction Flag.

- 5] STD: Set Direction Flag
- 6] CLI :Clear Interrupt Flag.
- 7] STI : Set Interrupt Flag.
- 8] HLT: Halt Processing.
- 9] NOP : No Operation
- 10] ESC: Escape

Executed by Co-processors and actions are performed according to 6 bit coding in the instruction.

- 11] LOCK : Assert bus lock Signal

This is a prefix instruction.

- 12] WAIT: Wait for test or Interrupt Signal.

Assert wait states.

STRING CONTROL

- 1] MOVS/ MOVSB/ MOVSW

This instn moves data byte or word from location in DS to location in ES.

- 2] REP / REPE / REPZ / REPNE / REPNZ

Repeat string instructions until specified conditions exist.This is prefix a instruction

- 3] CMPS / CMPSB / CMPSW

Compare string bytes or string words.

- 4] SCAS / SCASB / SCASW

Scan a string byte or string word. Compares byte in AL or word in AX. String address is to be loaded in DI.

- 5] STOS / STOSB / STOSW

Store byte or word in a string.Copies a byte or word in AL or AX to memory location pointed by DI.

- 6] LODS / LODSB /LODSW

Load a byte or word in AL or AX .Copies byte or word from memory location pointed by SI into AL or AX register.

Interrupt Control

- 1]INT type
- 2] INTO Interrupt on overflow
- 3] IRET Interrupt return

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Describe the instruction format of 8086.Also describe MOV instruction of 8086	DEC 2014	7

UNIT III 8086 Microprocessor

Lecture 8

ADDRESSING MODES OF 8086 (RGPV 2013)

The set of mechanisms by which an instruction can specify how to obtain its operands is known as Addressing modes. The CPU can access the operands (data) in a number of different modes.
The addressing modes available in Intel 8086 are:
1. Register Addressing
2. Immediate Addressing
3. Direct Addressing
4. Register Indirect Addressing
5. Based Relative Addressing
6. Indexed Relative Addressing
7. Based Indexed Relative Addressing

1.Register Addressing Mode:
With the Register Addressing mode the operand to be accessed is specified as residing in an internal register of the 8086.

Example: MOV AX , BX

This stands for move the contents of BX (the source operand) to AX (the destination operand).Both the source and the destination operands have been specified as the contents of internal registers of the 8086.

2.Immediate Addressing Mode:
If a source operand is part of the instruction instead of the contents of a register or memory location, it represents what is called the immediate operand and is accessed using immediate addressing mode. Typically immediate operand represents constant data. Immediate operands can be either a byte or word of data.

Example: MOV AL , 015H

In this instruction the operand 015H is an example of a byte wide immediate source operand. The destination operand, which consists of the contents of AL, uses register addressing. Thus this instruction employs both immediate and registers addressing modes.

3.Direct Addressing Mode:
Direct addressing differs from immediate addressing, that the locations following the instruction op-code hold an effective memory address (EA). This effective address is a 16-bit offset of the storage location of the operand from the current value in the data segment (DS) register .EA is

combined with the contents of DS in the BIU to produce the physical address of the operand.

Example: MOV CX , BETA

This stands for move the contents of the memory location, which is offset by BETA from the current value in DS into internal register CX.

4.Register Indirect Addressing Mode:

Register indirect addressing is similar to direct addressing, that an effective address is combined with the contents of DS to obtain a physical address. However it differs in a way that the offset is specified. Here EA resides in either a pointer register or an index register within the 8086.The pointer register can be either a base register BX or a base pointer register BP and the index register can be source index register SI or the destination index register DI.

Example MOV AX , [SI]

This instruction moves the contents of the memory location offset by the value of EA in SI from the current value in DS to the AX register.

5.Based Addressing Mode:
In the based addressing mode, the physical address of the operand is obtained by adding a direct or indirect displacement of the contents of either base register BX or base pointer register BP and the current value in DS and SS respectively.

Example MOV [BX] . BETA , AL

This instruction uses base register BX and direct displacement BETA to derive the EA of the destination operand. The based addressing mode is implemented by specifying the base register in the brackets followed by a period and direct displacement .The source operand is located in the byte accumulator AL

6.Indexed Addressing Mode:
Indexed addressing mode works identically to the based addressing but it uses the contents of the index registers instead of BX or BP, in the generation of the physical address.

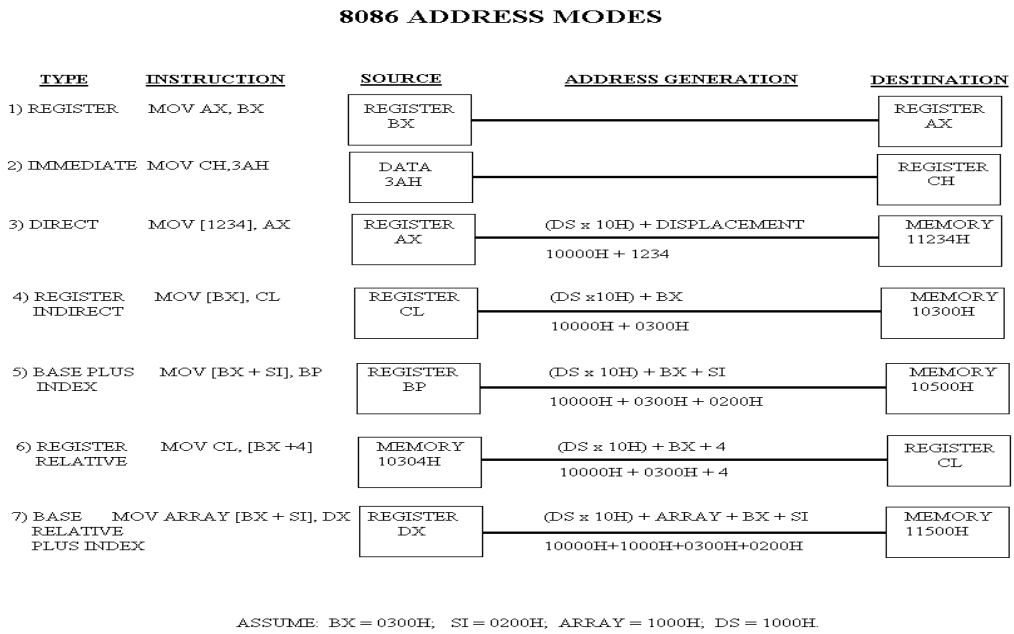
Example MOV AL , ARRAY [SI]

The source operand has been specified using direct index addressing. The notation this time is such ARRAY, which is a direct displacement, prefixes the selected index register, SI.

7.Based Indexed Addressing Mode:
Combining the based addressing mode and the indexed addressing mode together results in a new, more powerful mode known as based indexed addressing.

Example: MOV AH , [BX] . BETA [SI]

Here the source operand is accessed using the based indexed addressing mode. The effective address of the source operand is obtained as EA=(BX)+BETA+(SI)



Addressing Modes	*Examples*
❑ Immediate addressing	MOV AL, 12H
❑ Register addressing	MOV AL, BL
❑ Direct addressing	MOV [500H], AL
❑ Register Indirect addressing	MOV DL, [SI]
❑ Based addressing	MOV AX, [BX+4]
❑ Indexed addressing	MOV [DI-8], BL
❑ Based indexed addressing	MOV [BP+SI], AH
❑ Based indexed with displacement addressing	MOV CL, [BX+DI+2]

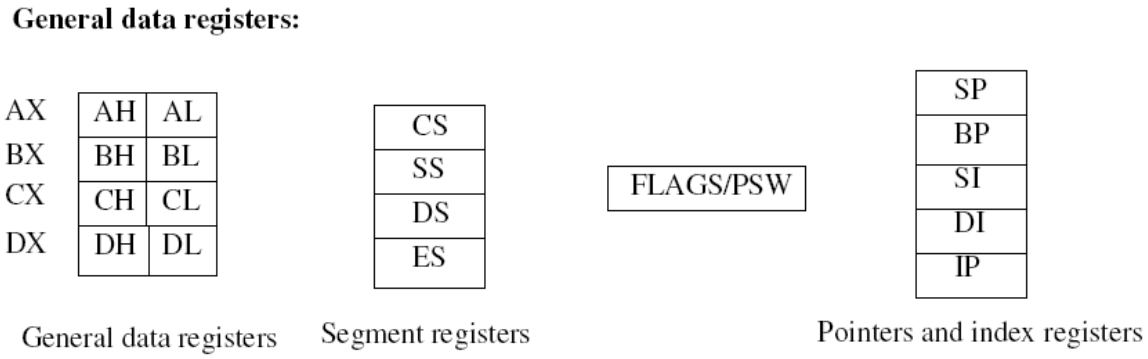
S.NO	RGPV QUESTIONS	Year	Marks
Q.1	What do you mean by addressing modes? What are the different addressing modes supported by 8086.	JUNE 2013	10

UNIT III 8086 Microprocessor

Lecture 9

Register organization of 8086: (RGPV 2013)

8086 has a powerful set of registers containing general purpose and special purpose registers. All the registers of 8086 are 16-bit registers. The general purpose registers, can be used either 8-bit registers or 16-bit registers. The general purpose registers are either used for holding the data, variables and intermediate results temporarily or for other purpose like counter or for storing offset address for some particular addressing modes etc. The special purpose registers are used as segment registers, pointers, index registers or as offset storage registers for particular addressing modes.



Register organization of 8086 Microprocessor

The registers AX, BX, CX, and DX are the general 16-bit registers.

AX Register: Accumulator register consists of two 8-bit registers AL and AH, which can be combined together and used as a 16- bit register AX. AL in this case contains the low-order byte of the word, and AH contains the high-order byte. Accumulator can be used for I/O operations, rotate and string manipulation.

BX Register: This register is mainly used as a **base register**. It holds the starting base location of a memory region within a data segment. It is used as offset storage for forming physical address in case of certain addressing mode.

CX Register: It is used as default counter or **count register** in case of string and loop instructions.

DX Register: **Data register** can be used as a port number in I/O operations and implicit operand or destination in case of few instructions. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number

Segment registers:
To complete 1Mbyte memory is divided into 16 logical segments. The complete 1Mbyte memory segmentation is as shown in fig 1.5. Each segment contains 64Kbyte of memory. There are four segment registers.

Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly. The CS register is automatically updated during far jump, far call and far return instructions. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.

Stack segment (SS) is a 16-bit register containing address of 64KB segment with program stack. By default, the processor assumes that all data referenced by the stack pointer (SP) and base pointer (BP) registers is located in the stack segment. SS register can be changed directly using POP instruction. It is used for addressing stack segment of memory. The stack segment is that segment of memory, which is used to store stack data.

Data segment (DS) is a 16-bit register containing address of 64KB segment with program data. By default, the processor assumes that all data referenced by general registers (AX, BX, CX, DX) and index register (SI, DI) is located in the data segment. DS register can be changed directly using POP and LDS instructions. It points to the data segment memory where the data is resided.

Extra segment (ES) is a 16-bit register containing address of 64KB segment, usually with program data. By default, the processor assumes that the DI register references the ES segment in string manipulation instructions. ES register can be changed directly using POP and LES instructions. It also refers to segment which essentially is another data segment of the memory. It also contains data.

Pointers and index registers.
The pointers contain within the particular segments. The pointers IP, BP, SP usually contain offsets within the code, data and stack segments respectively

Stack Pointer (SP) is a 16-bit register pointing to program stack in stack segment.

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

Source Index (SI) is a 16-bit register. SI is used for indexed, based indexed and register indirect addressing, as well as a source data addresses in string manipulation instructions.

Destination Index (DI) 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 Register determines the current state of the processor. They are modified automatically by

CPU after mathematical operations, this allows to determine the type of the result, and to determine conditions to transfer control to other parts of the program.

The 8086 flag register as shown in the fig 1.6. 8086 has 9 active flags and they are divided into two categories:

1. Conditional Flags
2. Control Flags

Conditional Flags

Conditional flags are as follows:

Carry Flag (CY): This flag indicates an overflow condition for unsigned integer arithmetic. It is also used in multiple-precision arithmetic.

Auxiliary Flag (AC): If an operation performed in ALU generates a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), the AC flag is set i.e. carry given by D3 bit to D4 is AC flag. This is not a general-purpose flag, it is used internally by the Processor to perform Binary to BCD conversion

Parity Flag (PF): This flag is used to indicate the parity of result. If lower order 8-bits of the result contains even number of 1's, the Parity Flag is set and for odd number of 1's, the Parity flag is reset.

Zero Flag (ZF): It is set; if the result of arithmetic or logical operation is zero else it is reset.

Sign Flag (SF): In sign magnitude format the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.

Control Flags

Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:

Trap Flag (TF): It is used for single step control. It allows user to execute one instruction of a program at a time for debugging. When trap flag is set, program can be run in single step mode.

Interrupt Flag (IF): It is an interrupt enable/disable flag. If it is set, the maskable interrupt of 8086 is enabled and if it is reset, the interrupt is disabled. It can be set by executing instruction sit and can be cleared by executing CLI instruction.

Direction Flag (DF): It is used in string operation. If it is set, string bytes are accessed from higher memory address to lower memory address. When it is reset, the string bytes are accessed from lower memory address to higher memory address.

UNIT III 8086 Microprocessor

Lecture 10

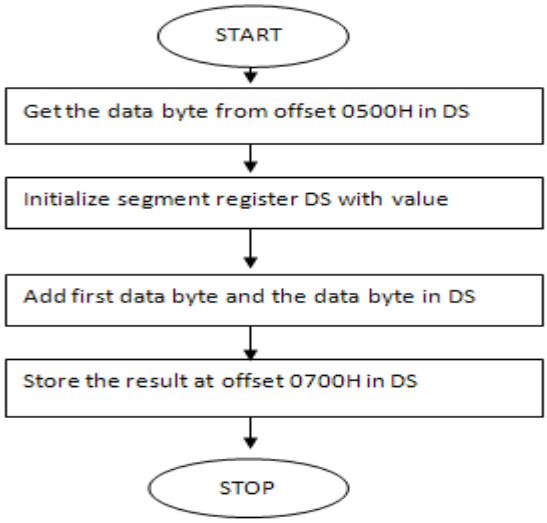
Assembly language programs of 8086 microprocessor

Add a data byte located at offset 0500 in 2500H segment to another data byte available at 0600H in same segment and store the result at 0700H in the same segment.

In this program, we have to initialize data segment DS with a value 2000H.We know that, we can not directly load immediate data in segment register. Thus we have to first load the value 2000H in one of the general purpose register say AX and the register AX contents are copied to the segment register Ds.

PROGRAM –

```
Label    opcode  oprand          ;  comment
0400     Mov AX, 2000H           ;  [Initialize Ds
0402      Mov DS, AX             ;  with value 2000H]
0404      Mov AX,[0500H]         ;   Get the data byte located at offset 0500H in DS
0406     ADD AX,[6000H]          ;   Add first data byte and second data byte
                                   located at offset 6000H
0407     Mov [7000H],AX          ;   Store the result of addition i.e the contents of AX
                                   offset 0700H in DS
0408     INT_3                   ; Stop the processing
```



FLOW CHART

S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Draw the register organization of 8086 & explain typical applications of each register.	JUNE 2013	10
S.NO	RGPV QUESTIONS	Year	Marks
Q.1	What are interrupt available in 8086?Give the Interrupt vector table of 8086.	JUNE 2013	10
S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Write a Program in assembly language for 8086 to get the sum of two 8 bit no's. Draw the flowchart for program also.	JUNE 2013	10
S.NO	RGPV QUESTIONS	Year	Marks
Q.1	Draw & discuss the read & write cycle timing diagram of 8086 in maximum mode.	JUNE 2012	10
S.NO	RGPV QUESTIONS	Year	Marks
Q.1	What are interrupt available in 8086?Give the Interrupt vector table of 8086.	JUNE 2012	10