UNIT-1

Introduction to 8086:

8086 Microprocessor is an enhanced version of 8085Microprocessor that was designed by Intel in 1976. It is a 16-bit Microprocessor having 20 address lines and16 data lines that provides up to 1MB storage. It consists of powerful instruction set, which provides operations like multiplication and division easily.

It supports two modes of operation, i.e. Maximum mode and Minimum mode. Maximum mode is suitable for system having multiple processors and Minimum mode is suitable for system having a single processor.

Features of 8086

- It is a 16-bit Microprocessor (µp).It's ALU, internal registers works with 16bit binary word.
- 8086 has a 20 bit address bus can access up to 2^{20}= 1 MB memory locations.
- 8086 has a 16bit data bus. It can read or write data to a memory/port either 16bits or 8 bit at a time.
- It can support up to 64K I/O ports.
- It provides 14, 16 -bit registers.
- Frequency range of 8086 is 6-10 MHz
- It has multiplexed address and data bus AD0- AD15 and A16 – A19.
- It requires single phase clock with 33% duty cycle to provide internal timing.
- It can prefetch up to 6 instruction bytes from memory and queues them in order to speed up instruction execution.
- It requires +5V power supply.
- A 40 pin dual in line package.
- 8086 is designed to operate in two modes, Minimum mode and Maximum mode.

Intel 8086 Microprocessor architecture:

The architecture of 8086 microprocessor is divided into two units:

1. BIU(Bus interface unit)
2. EU(execution unit)

1. BIU (Bus Interface Unit)

BIU takes care of all data and addresses transfers on the buses for the EU like sending addresses, fetching instructions from the memory, reading data from the ports and the memory as well as writing data to the ports and the memory. EU has no direction connection with System Buses so this is possible with the BIU. EU and BIU are connected with the Internal Bus.

It has the following functional parts –

- **Instruction queue** − BIU contains the instruction queue. BIU gets upto 6 bytes of next instructions and stores them in the instruction queue. When EU executes instructions and is ready for its next instruction, then it simply reads the instruction from this instruction queue resulting in increased execution speed.
- Fetching the next instruction while the current instruction executes is called **pipelining**.
- **Segment register** − BIU has 4 segment buses, i.e. CS, DS, SS& ES. It holds the addresses of instructions and data in memory, which are used by the processor to access memory locations. It also contains 1 pointer register IP, which holds the address of the next instruction to executed by the EU.
    - **CS** − It stands for Code Segment. It is used for addressing a memory location in the code segment of the memory, where the executable program is stored.
    - **DS** − It stands for Data Segment. It consists of data used by the program andis accessed in the data segment by an offset address or the content of other register that holds the offset address.
    - **SS** − It stands for Stack Segment. It handles memory to store data and addresses during execution.
    - **ES** − It stands for Extra Segment. ES is additional data segment, which is used by the string to hold the extra destination data.
- **Instruction pointer** − It is a 16-bit register used to hold the address of the next instruction to be executed.

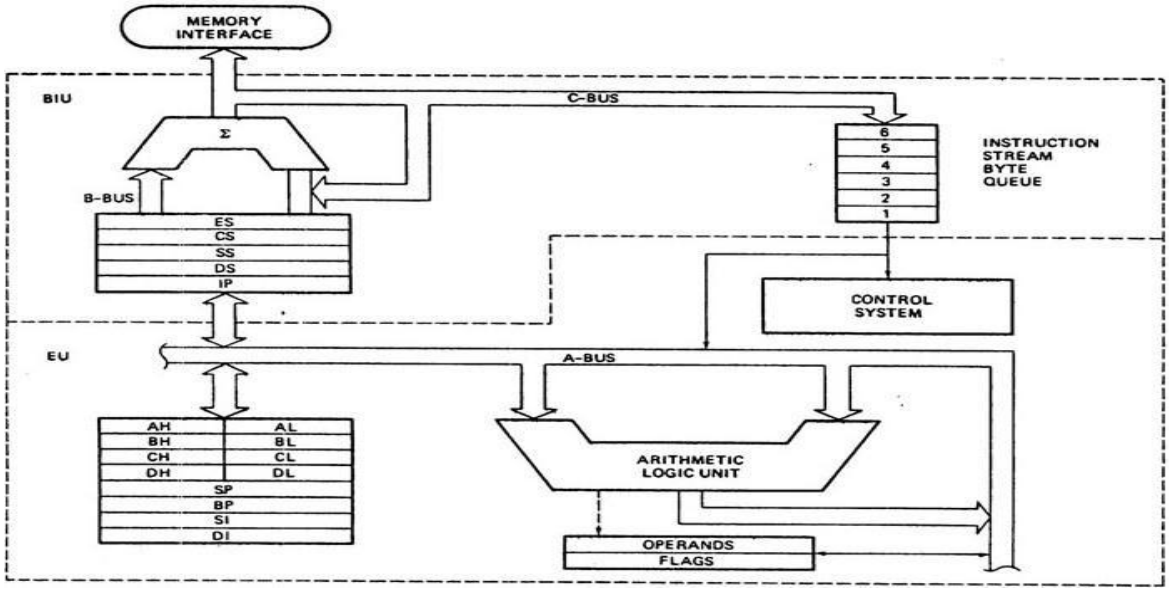


Fig: Internal Architecture of 8086

EU (Execution Unit)

Execution unit gives instructions to BIU stating from where to fetch the data and then decode and execute those instructions. Its function is to control operations on data using the instruction decoder & ALU. EU has no direct connection with system buses as shown in the above figure, it performs operations over data through BIU.

Let us now discuss the functional parts of 8086 microprocessors.

ALU

It handles all arithmetic and logical operations, like +, −, ×, /, OR, AND, NOT operations.

Flag Register

It is a 16-bit register that behaves like a flip-flop, i.e. it changes its status according to the result stored in the accumulator. It has 9 flags and they are divided into 2 groups − Conditional Flags and Control Flags.

Conditional Flags

It represents the result of the last arithmetic or logical instruction executed. Following is the list of conditional flags −

- **Carry flag** − This flag indicates an overflow condition for arithmetic operations.
- **Auxiliary flag** − When an operation is performed at ALU, it results in a carry/barrow from lower nibble (i.e. D0 – D3) to upper nibble (i.e. D4 – D7), then this flag is set, i.e. carry given by D3 bit to D4 is AF flag. The processor uses this flag to perform binary to BCD conversion.

- **Parity flag** − This flag is used to indicate the parity of the result, i.e. when the lower order 8-bits of the result contains even number of 1's, then the Parity Flag is set. For odd number of 1's, the Parity Flag is reset.
- **Zero flag** − This flag is set to 1 when the result of arithmetic or logical operation is zero else it is set to 0.
- **Sign flag** − This flag holds the sign of the result, i.e. when the result of the operation is negative, then the sign flag is set to 1 else set to 0.
- **Overflow flag** − This flag represents the result when the system capacity is exceeded.

Control Flags

Control flags controls the operations of the execution unit. Following is the list of control flags

- **Trap flag** − It is used for single step control and allows the user to execute one instruction at a time for debugging. If it is set, then the program can be run in a single step mode.
- **Interrupt flag** − It is an interrupt enable/disable flag, i.e. used to allow/prohibit the interruption of a program. It is set to 1 for interrupt enabled condition and set to 0 for interrupt disabled condition.
- **Direction flag** − It is used in string operation. As the name suggests when it is set then string bytes are accessed from the higher memory address to the lower memory address and vice-a-versa.

General purpose register

There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL. These registers can be used individually to store 8-bit data and can be used in pairs to store 16bit data. The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX, CX, and DX respectively.

- **AX register** − It is also known as accumulator register. It is used to store operands for arithmetic operations.
- **BX register** − It is used as a base register. It is used to store the starting base address of the memory area within the data segment.
- **CX register** − It is referred to as counter. It is used in loop instruction to store the loop counter.
- **DX register** − This register is used to hold I/O port address for I/O instruction.

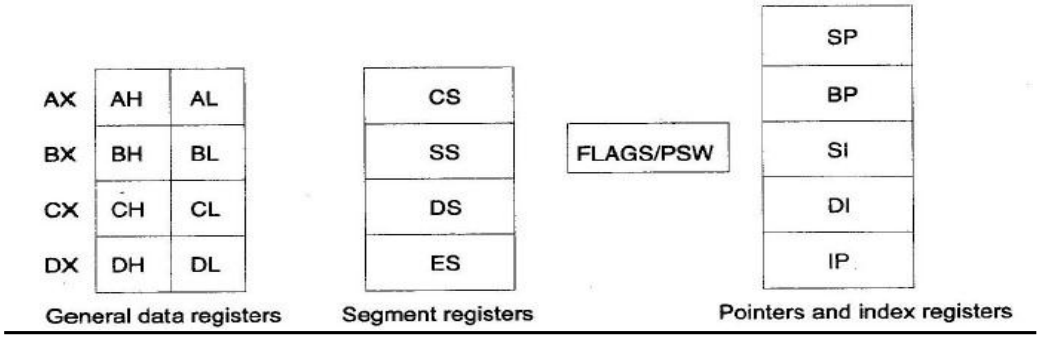
Stack pointer register

It is a 16-bit register, which holds the address from the start of the segment to the memory location, where a word was most recently stored on the stack.

Register Organization:

The 8086 microprocessor has a total of fourteen registers that are accessible to the programmer. It is divided into four groups. They are:

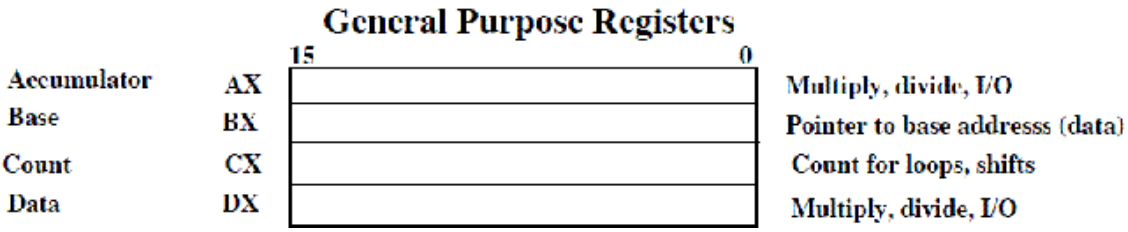
- Four General purpose registers
- Five Index/Pointer registers
- Four Segment registers
- Flag register



General purpose registers:

AX: 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 and string manipulation.

BX: Base register consists of two 8-bit registers BL and BH, which can be combined together and used as a 16-bit register BX. BL in this case contains the low-order byte of the word, and BH contains the high-order byte. BX register usually contains a data pointer used for based, based indexed or register indirect addressing.



CX: Count register consists of two 8-bit registers CL and CH, which can be

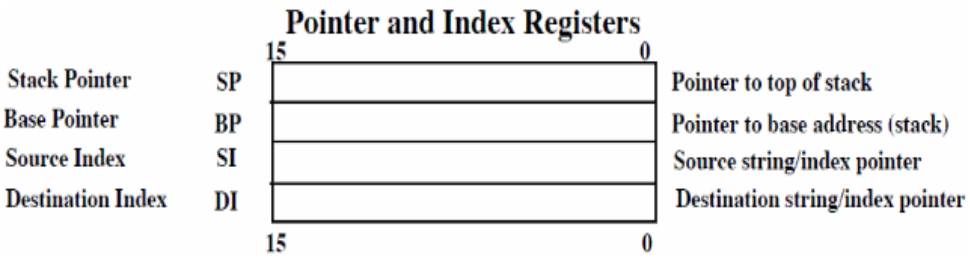
combined together and used as a 16-bit register CX. When combined, CL register contains the low order byte of the word, and CH contains the high-order byte. Count register can be used in Loop, shift/rotate instructions and as a counter in string manipulation

DX: Data register consists of two 8-bit registers DL and DH, which can be combined together and used as a 16-bit register DX. When combined, DL register contains the low order byte of the word, and DH contains the high-order byte. Data register can be used as a port number in I/O operations. In integer 32-bit multiply and divide instruction the DX register contains high-order word of the initial or resulting number.

Index or Pointer Registers

These registers can also be called as Special Purpose registers.

Stack Pointer (SP) is a 16-bit register pointing to program stack, i.e. it is used to hold the address of the top of stack. The stack is maintained as a LIFO with its bottom at the start of the stack segment (specified by the SS segment register).Unlike the SP register, the BP can be used to specify the offset of other program segments.



Base Pointer (BP) is a 16-bit register pointing to data in stack segment. It is usually used by subroutines to locate variables that were passed on the stack by a calling program. 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 address in string manipulation instructions. Used in conjunction with the DS register to point to data locations in the data segment.

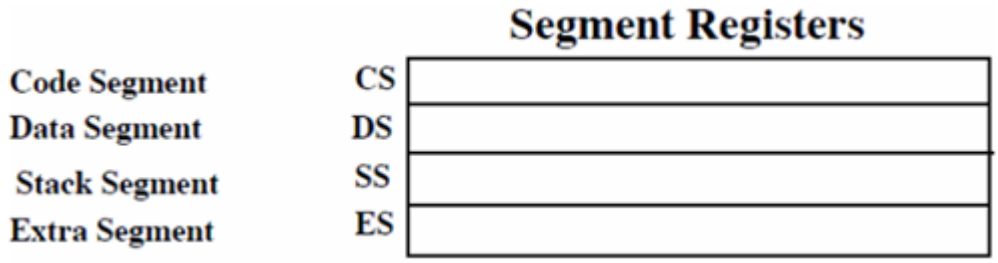
Destination Index (DI) is a 16-bit register. Used in conjunction with the ES register in string operations. DI is used for indexed, based indexed and register indirect addressing, as well as a destination data addresses in string manipulation

instructions. In short, Destination Index and SI Source Index registers are used to hold address.

Instruction Pointer (IP) is a 16-bit register. This is a crucially important register which is used to control which instruction the CPU executes. The IP, or program counter, is used to store the memory location of the next instruction to be executed. The CPU checks the program counter to ascertain which instruction to carry out next. It then updates the program counter to point to the next instruction. Thus the program counter will always point to the next instruction to be executed.

Segment Registers

Most of the registers contain data/instruction offsets within 64 KB memory segment. There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses 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.

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.

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.

Extra segment (ES) used to hold the starting address of Extra segment. Extra segment is provided for programs that need to access a second data segment. Segment

registers cannot be used in arithmetic operations.

Flag Register of 8086:

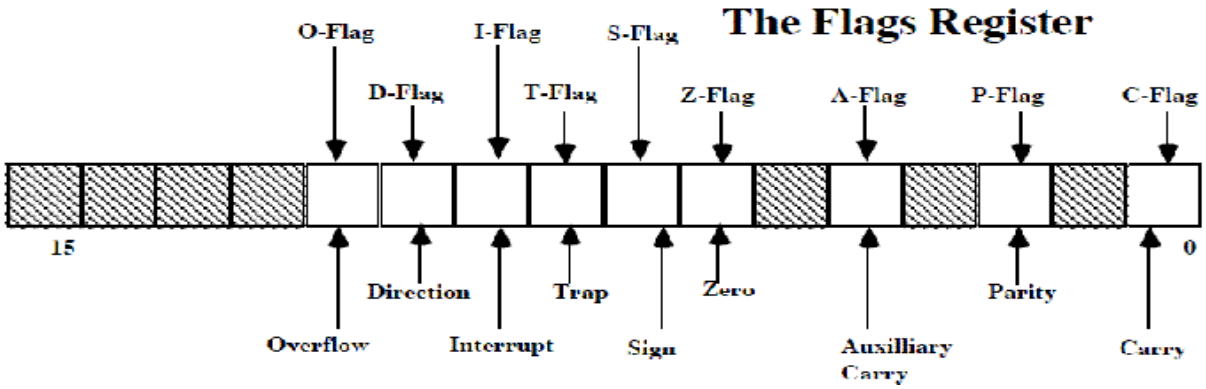
Flag Register contains a group of status bits called flags that indicate the status of the CPU or the result of arithmetic operations. There are two types of flags:

1. The **status flags** which reflect the result of executing an instruction. The programmer cannot set/reset these flags directly.

2. The control **flags enable** or disable certain CPU operations. The programmer can set/reset these bits to control the CPU's operation.

Nine individual bits of the status register are used as control flags (3 of them) and status flags (6 of them).The remaining 7 are not used.

A flag can only take on the values 0 and 1. We say a flag is set if it has the value 1.The status flags are used to record specific characteristics of arithmetic and of logical instructions.



Control Flags: There are three control flags

1. **The Direction Flag (D):** Affects the direction of moving data blocks by such instructions as MOVS, CMPS and SCAS. The flag values are 0 = up and 1 = down and can be set/reset by the STD (set D) and CLD (clear D) instructions.
2. **The Interrupt Flag (I):** Dictates whether or not system interrupts can occur. Interrupts are actions

Initiated by hardware block such as input devices that will interrupt the normal execution of programs. The flag values are 0 = disable interrupts or 1 = enable interrupts and can be manipulated by the CLI (clear I) and STI (set I) instructions.

2. **The Trap Flag (T):** Determines whether or not the CPU is halted after the execution of each instruction. When this flag is set (i.e. = 1), the programmer can single step through his program to debug any errors. When this flag = 0 this feature is off. This flag can be set by the INT 3 instruction.

Status Flags: There are six status flags

1. **The Carry Flag (C):** This flag is set when the result of an unsigned arithmetic operation is too large to fit in the destination register. This happens when there is an end carry in an addition operation or there an end borrows in a subtraction operation. A value of 1 = carry and 0 = no carry.
2. **The Overflow Flag (O):** This flag is set when the result of a signed arithmetic operation is too large to fit in the destination register (i.e. when an overflow occurs). Overflow can occur when adding two numbers with the same sign (i.e. both positive or both negative). A value of 1 = overflow and 0 = no overflow.
3. **The Sign Flag (S):** This flag is set when the result of an arithmetic or logic operation is negative. This flag is a copy of the MSB of the result (i.e. the sign bit). A value of 1 means negative and 0 = positive.
4. **The Zero Flag (Z):** This flag is set when the result of an arithmetic or logic operation is equal to zero. A value of 1 means the result is zero and a value of 0 means the result is not zero.
5. **The Auxiliary Carry Flag (A):** This flag is set when an operation causes a carry from bit 3 to bit 4 (or a borrow from bit 4 to bit 3) of an operand. A value of 1 = carry and 0 = no carry.
6. **The Parity Flag (P):** This flags reflects the number of 1s in the result of an operation. If the number of 1s is even its value = 1 and if the number of 1s is odd then its value = 0.

Pin Diagram and Pin description of 8086:

The Microprocessor 8086 is a 16-bit CPU available in different clock rates and packaged in a 40 pin

CERDIP 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.

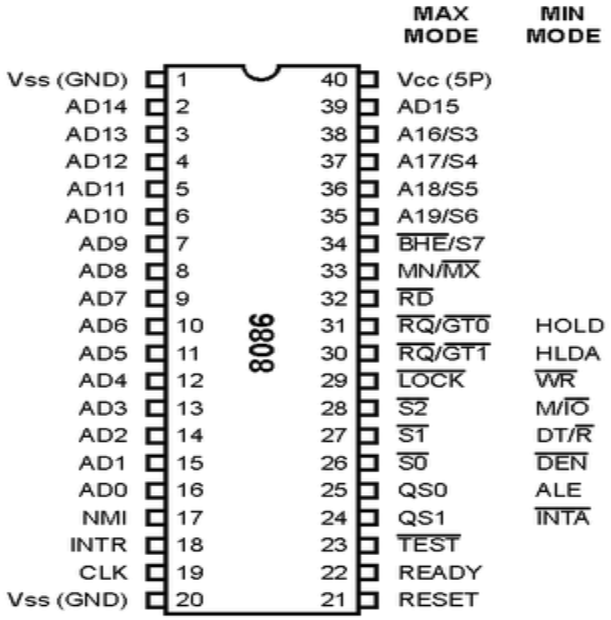


Fig: Pin diagram of 8086 microprocessor

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

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.

•The S4 and S3 combinedly 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.

BHE /S7 : The bus high enable is used to indicate the transfer of data over the higherorder (D15-D8) data bus as shown in table. 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 tristated during hold. It is low during T1 for the first pulse of the interrupt acknowledge cycle.

BHE	A_0	Indication
0	0	Whole word
0	1	Upper byte from or to even address
1	0	Lower byte from or to even address
1	1	None

RD – Read : This signal on low indicates the peripheral that the processor is performing s 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.

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.

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. Its an asymmetric square wave with 33% duty cycle

MN/ MX: The logic level at this pin decides whether the processor is to operate in either minimum or maximum mode.

The following pin functions are for the minimum mode operation of 8086.

M/ IO – Memory/IO: This is a status line logically equivalent to S2 in maximum mode. When it is low, it indicates the CPU is having an I/O operation, and when it is high, it indicates that the CPU is having a memory operation. This line Becomes active high in the previous T4 and remains active till final T4 of the current cycle. It is tri stated during local bus "hold acknowledge ".

INTA – Interrupt Acknowledge: This signal is used as a read strobe for interrupt acknowledge cycles. i.e. when it goes low, the processor has accepted the interrupt.

ALE – Address Latch Enable :This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches. This signal is active high and is never tri stated.

DT/ R – Data Transmit/Receive: This output is used to decide the direction of data flow through the transceivers (bidirectional buffers). When the processor sends out data, this signal is high and when the processor is receiving data, this signal is low.

DEN – Data Enable: This signal indicates the availability of valid data over the address/data lines. It is used to enable the transceivers (bi directional buffers) to separate the data from the multiplexed address/data signal. It is active from the middle of T2 until the middle of T4. This is tri stated during hold acknowledge' cycle.

HOLD, HLDA- Acknowledge: When the HOLD line goes high, it indicates to the processor that another master is requesting the bus access. The processor, after receiving the HOLD request, issues the hold acknowledge signal on HLDA pin, in the middle of the next clock cycle after completing the current bus cycle.

•At the same time, the processor floats the local bus and control lines. When the processor detects the HOLD line low, it lowers the HLDA signal. HOLD is an asynchronous input, and is should be externally synchronized.

If the DMA request is made while the CPU is performing a memory or I/O cycle, it will release the local bus during T4 provided :

1.The request occurs on or before T2 state of the current cycle.

2.The current cycle is not operating over the lower byte of a word.

3.The current cycle is not the first acknowledge of an interrupt acknowledge sequence.

4. A Lock instruction is not being executed

The following pin functions are applicable for maximum mode operation of 8086.

S2, S1, S0 – Status Lines: These are the status lines which reflect the type of operation, being carried out by the processor. These become activity during T4 of the previous cycle and active during T1 and T2 of the current bus cycles.

S_2	S_1	S_0	Indication
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

LOCK: This output pin indicates that other system bus master will be prevented from gaining the system bus, while the LOCK signal is low

•The LOCK signal is activated by the 'LOCK' prefix instruction and remains active until the completion of the next instruction. When the CPU is executing a critical instruction which requires the system bus, the LOCK prefix instruction ensures that other processors connected in the system will not gain the control of the bus.

• The 8086, while executing the prefixed instruction, asserts the bus lock signal output, which may be connected to an external bus controller.

QS1, QS0 – Queue Status: These lines give information about the status of the code-prefetch queue. These are active during the CLK cycle after while the queue operation is performed.

•This modification in a simple fetch and execute architecture of a conventional microprocessor offers an added advantage of pipelined processing of the instructions.

•The 8086 architecture has 6-byte instruction prefetch queue. Thus even the largest (6-bytes) instruction can be prefetched from the memory and stored in the prefetch. This results in a faster execution of the instructions.

•In 8085 an instruction is fetched, decoded and executed and only after the execution of this instruction, the next one is fetched.

•By prefetching the instruction, there is a considerable speeding up in instruction execution in 8086. This is known as instruction pipelining.

•At the starting the CS:IP is loaded with the required address from which the execution is to be started. Initially, the queue will be empty an the microprocessor starts a fetch operation to bring one byte (the first byte) of instruction code, if the CS:IP address is odd or two bytes at a time, if the CS:IP address is even.

•The first byte is a complete opcode in case of some instruction (one byte opcode instruction) and is a part of opcode, in case of some instructions (two byte opcode instructions), the remaining part of code lie in second byte.

•The second byte is then decoded in continuation with the first byte to decide the instruction length and the number of subsequent bytes to be treated as instruction data.

•The queue is updated after every byte is read from the queue but the fetch cycle is initiated by BIU only if at least two bytes of the queue are empty and the EU may be concurrently executing the fetched instructions

RQ / GT0 , RQ / GT1 – Request/Grant : These pins are used by the other local bus master in maximum mode, to force the processor to release the local bus at the end of the processor current bus cycle.

•Each of the pin is bidirectional with RQ/GT0 having higher priority than RQ/GT1.

•RQ/GT pins have internal pull-up resistors and may be left unconnected.

Request/Grant sequence is as follows:

1. A pulse of one clock wide from another bus master requests the bus access to 8086. 2.During T4(current) or T1(next) clock cycle, a pulse one clock wide from 8086 to the requesting master, indicates that the 8086 has allowed the local bus to float and that it will enter the 'hold acknowledge' state at next cycle. The CPU bus interface unit is likely to be disconnected from the local bus of the system.

2. A one clock wide pulse from the another master indicates to the 8086 that the hold request is about to end and the 8086 may regain control of the local bus at the next clock cycle. Thus each master to master exchange of the local bus is a sequence of 3 pulses. There must be at least one dead clock cycle after each bus exchange.

•The request and grant pulses are active low.

•For the bus request those are received while 8086 is performing memory or I/O cycle,

The granting of the bus is governed by the rules as in case of HOLD and HLDA in minimum mode.

MINIMUM MODE 8086 SYSTEM AND TIMINGS

In a minimum mode 8086 system, the microprocessor 8086 is operated in minimum mode by strapping its MN/MX* pin to logic1. In this mode, all the control signals are given out by the microprocessor chip itself. There is a single microprocessor in the minimum mode system. The remaining components in the system are latches, transreceivers, clock generator, memory and I/O devices. Some type of chip selection logic may be required for selecting memory or I/O devices, depending upon the address map of the system.

The latches are generally buffered output D-type flip-flops, like, 74LS373 or 8282. They are used for separating the valid address from the multiplexed address/data signals and are controlled by the ALE signal generated by 8086.

Transreceivers are the bidirectional buffers and some times they are called as data amplifiers. They are required to separate the valid data from the time multiplexed address/data signal. They are controlled by two signals, namely, DEN* and DT/R*. The

DEN* signal indicates that the valid data is available on the data bus, while DT/R indicates the direction of data, i.e. from or to the processor.

The system contains memory for the monitor and users program storage. Usually, EPROMS are used for monitor storage, while RAMs for users program storage. A system may contain I/O devices for communication with the processor as well as some special purpose I/O devices.

The clock generator generates the clock from the crystal oscillator and then shapes it and divides to make it more precise so that it can be used as an accurate timing reference for the system. The clock generator also synchronizes some external signals with the system clock. The general system organization is shown in Fig. Since it has 20 address lines and 16 data lines, the 8086 CPU requires three octal address latches and two octal data buffers for the complete address and data separation. The working of the minimum mode configuration system can be better described in terms of the timing diagrams rather than qualitatively describing the operations.

The opcode fetch and read cycles are similar. Hence the timing diagram can be categorized in two parts, the first is the timing diagram for read cycle and the second is the timing diagram for write cycle. The read cycle begins in T1 with the assertion of the address latch enable (ALE) signal and also M/IO* signal.

During the negative going edge of this signal, the valid address is latched on the local bus. The BHE* and A0 signals address low, high or both bytes. From Tl to T4, the M/IO* signal indicates a memory or I/O operation. At T2 the address is removed from the local bus and is sent to the output. The bus is then tristated. The read (RD*) control signal is also activated in T2 .The read (RD) signal causes the addressed device to enable its data bus drivers. After RD* goes low, the valid data is available on the data bus. The addressed device will drive the READY line high, when the processor returns the read signal to high level, the addressed device will again tristate its bus drivers.

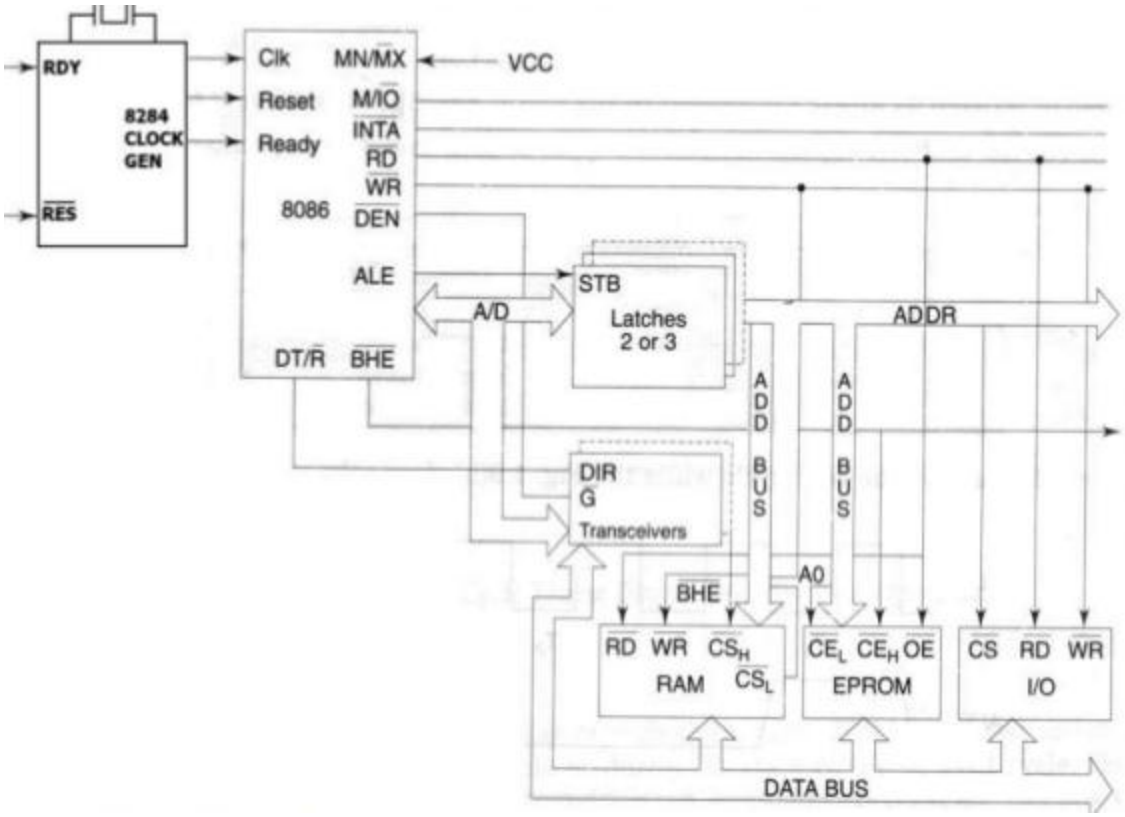


Fig shows the write cycle timing diagram. A write cycle also begins with the assertion of ALE and the emission of the address. The M/IO* signal is again asserted to indicate a memory or I/O operation.

In T2 after sending the address in Tl the processor sends the data to be written to the addressed location. The data remains on the bus until middle of T4 state. The WR* becomes active at the beginning ofT2 (unlike RD* is somewhat delayed in T2 to provide time for floating).

The BHE* and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or written. The M/IO*, RD* and WR* signals indicate the types of data transfer as specified in Table

M/IO	RD	WR	Transfer Type
0	0	1	I/O read
0	1	0	I/O write
1	0	1	Memory read
1	1	0	Memory write

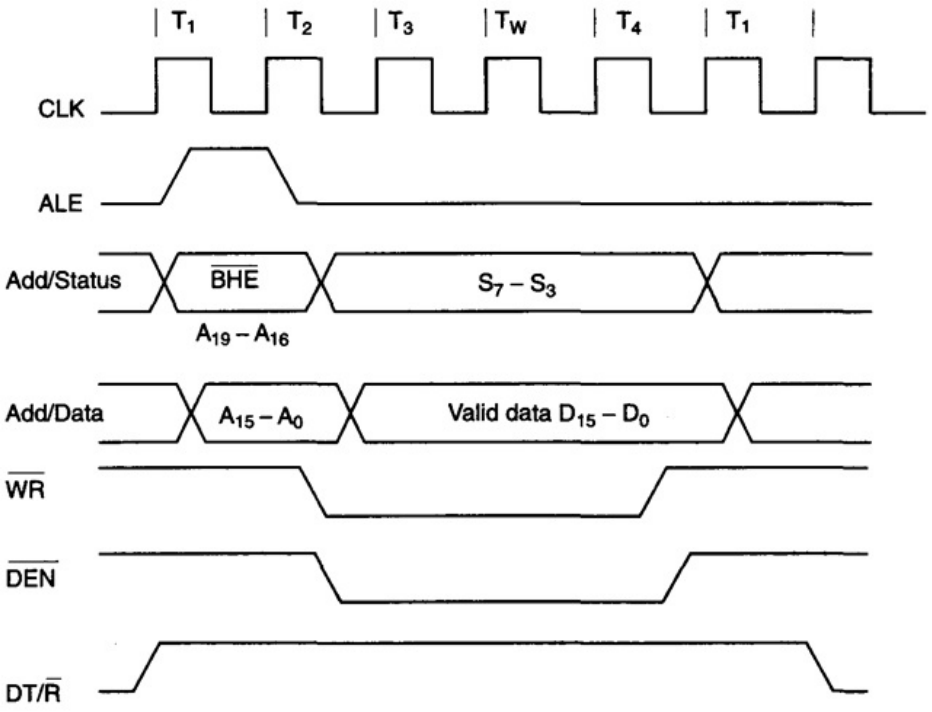


Fig: Memory Write Timings in Minimum Mode

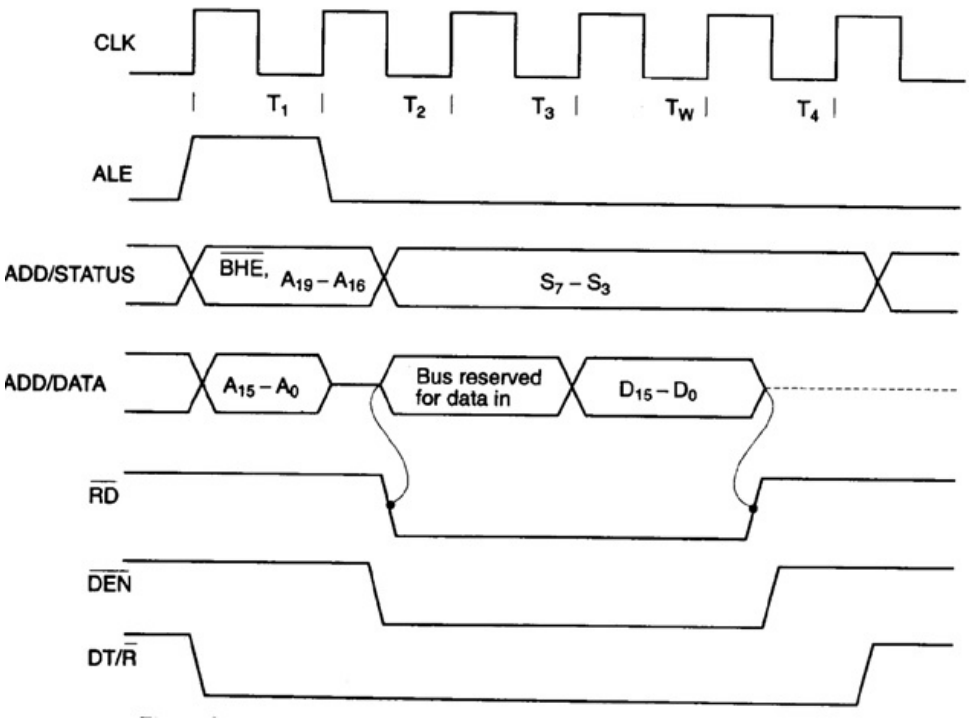
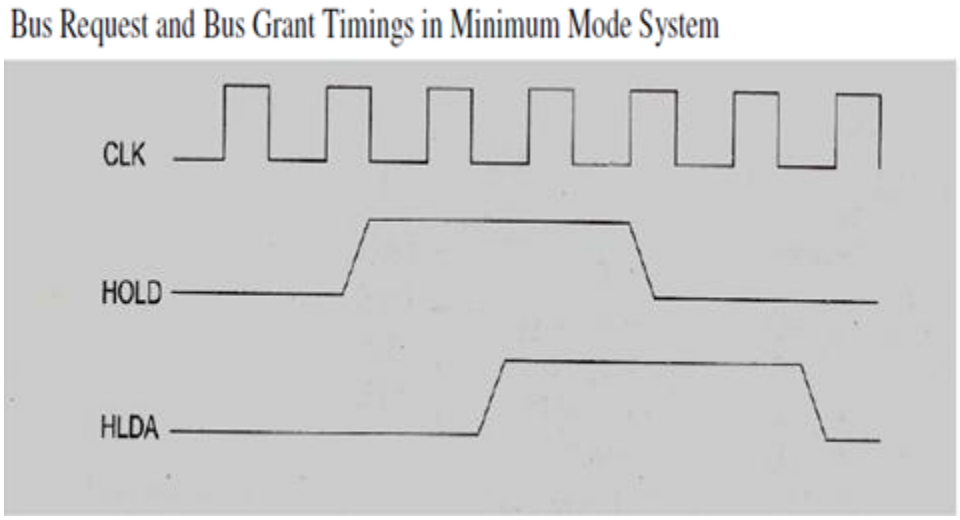


Fig: Memory read Timings in Minimum Mode

HOLD Response Sequence



The HOLD pin is checked at the end of the each bus cycle. If it is received active by the processor before T4 of the previous cycle or during T1 state of the current cycle, the CPU activities HLDA in the next clock cycle and for the succeeding bus cycles, the bus

will be given to another requesting master The control control of the bus is not regained by the processor until the requesting master does not drop the HOLD pin low. When the request is dropped by the requesting master, the HLDA is dropped by the processor at the trailing edge of the next clock as shown in fig 1.4.

MAXIMUM MODE 8086 SYSTEM AND TIMINGS

In the maximum mode, the 8086 is operated by strapping the MN/MX* pin to ground. In this mode, the processor derives the status signals S2*, S1* and S0*. Another chip called bus controller derives the control signals using this status information. In the maximum mode, there may be more than one microprocessor in the system configuration. The other components in the system are the same as in the minimum mode system. The general system organization is as shown in the fig1.1

The basic functions of the bus controller chip IC8288, is to derive control signals like RD* and WR* (for memory and I/O devices), DEN*, DT/R*, ALE, etc. using the information made available by the processor on the status lines. The bus controller chip has input lines S2*, S1* and S0* and CLK. These inputs to 8288 are driven by the CPU. It derives the outputs ALE, DEN*, DT/R*, MWTC*, AMWC*, IORC*, IOWC* and AIOWC*. The AEN*, IOB and CEN pins are specially useful for multiprocessor systems. AEN* and IOB are generally grounded. CEN pin is usually tied to +5V.

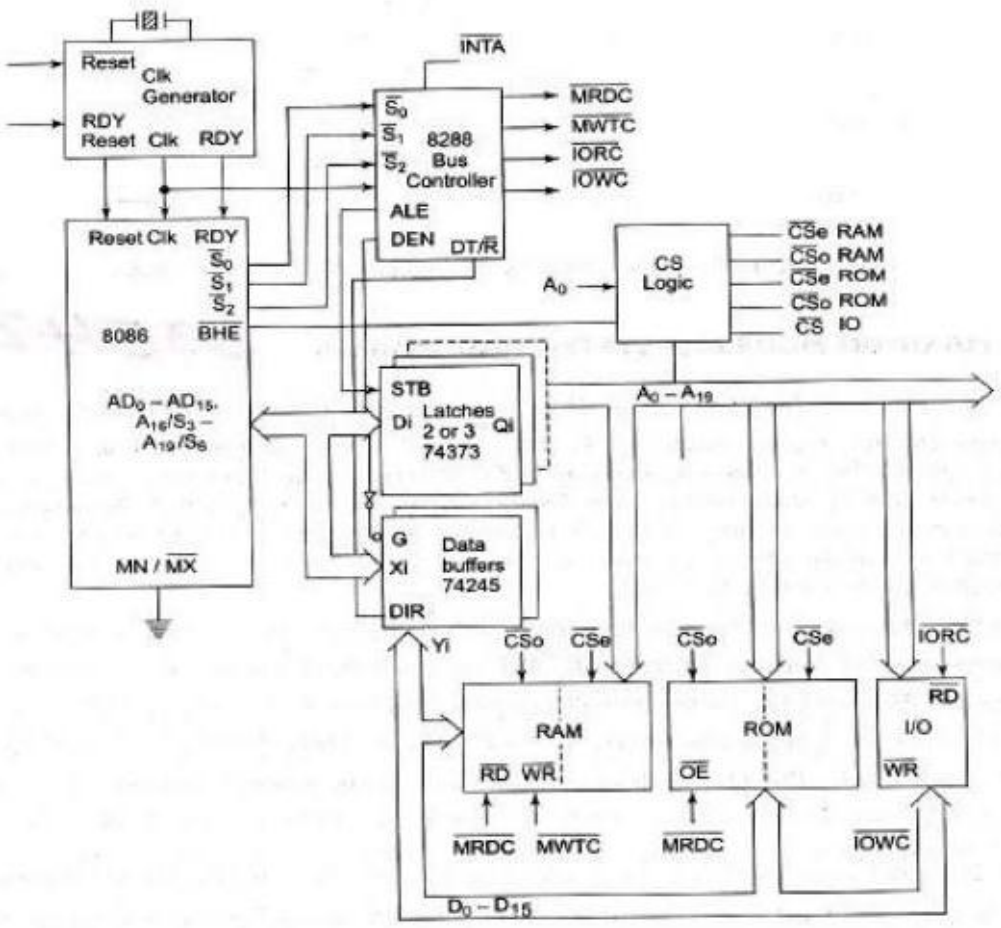


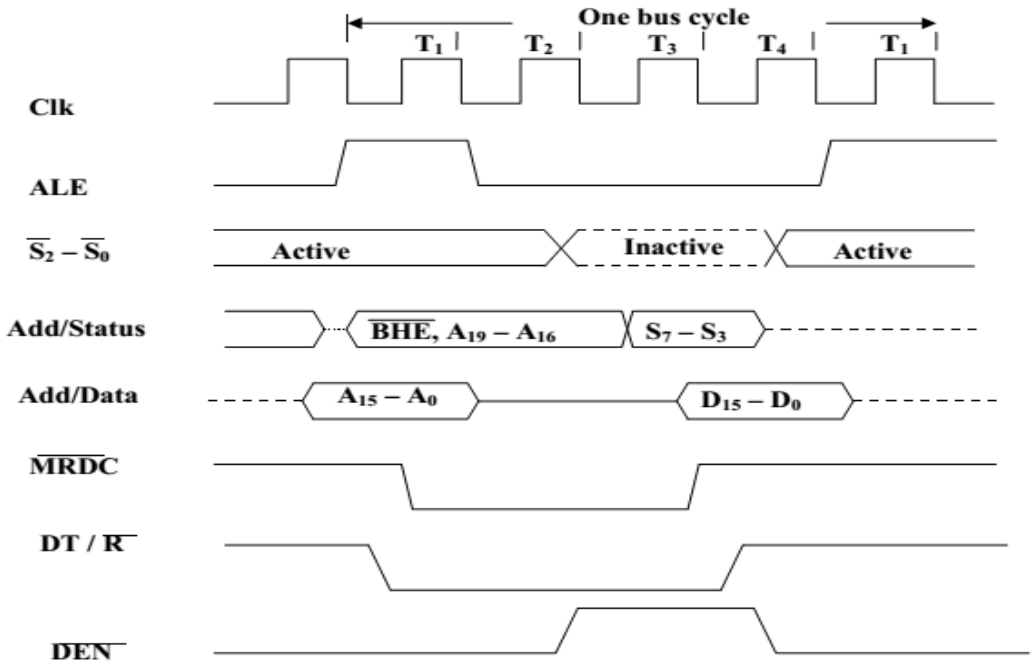
Fig: Maximum Mode 8086 System

The significance of the MCE/PDEN* output depends upon the status of the IOB pin. If IOB is grounded, it acts as master cascade enable to control cascaded 8259A; else it acts as peripheral data enable used in the multiple bus configurations. INTA* pin is used to issue two interrupt acknowledge pulses to the interrupt controller or to an interrupting device. IORC*, IOWC* are I/O read command and I/O write command signals respectively. These signals enable an IO interface to read or write the data from or to the addressed port.

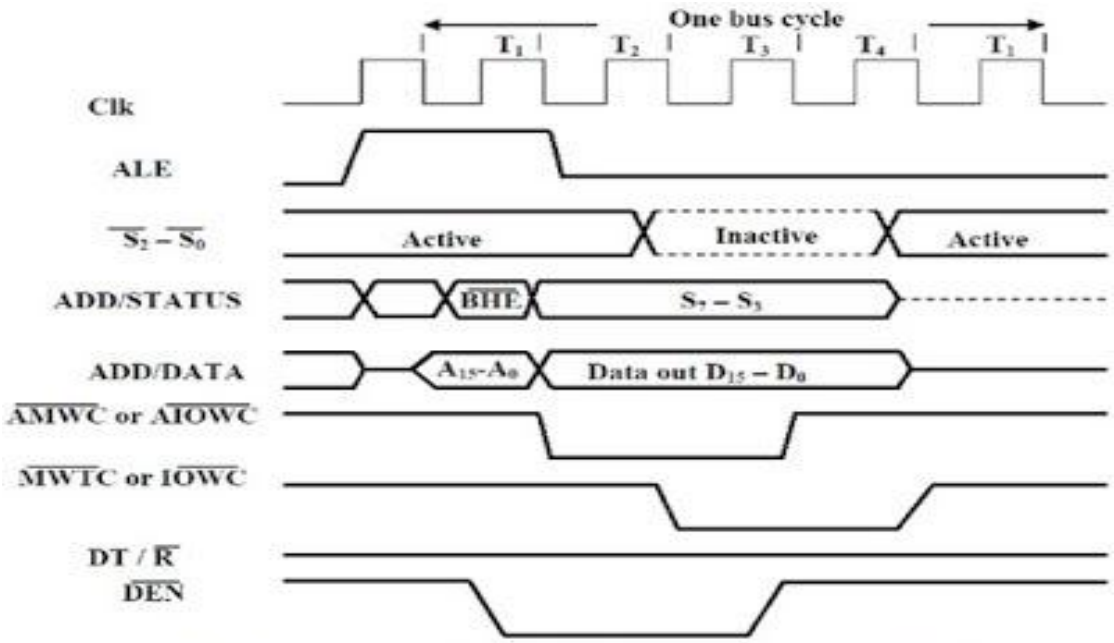
The MRDC*, MWTC* are memory read command and memory write command signals respectively and may be used as memory read and write signals. All these command signals instruct the memory to accept or send data from or to the bus. For both of these write command signals, the advanced signals namely AIOWC* and AMWTC* are available. They also serve the same purpose, but are activated one clock

cycle earlier than the IOWC* and MWTC* signals, respectively. The maximum mode system is shown in fig. 1.1.

The maximum mode system timing diagrams are also divided in two portions as read (input) and write (output) timing diagrams. The address/data and address/status timings are similar to the minimum mode. ALE is asserted in T1, just like minimum mode. The only difference lies in the status signals used and the available control and advanced command signals. The fig. 1.2 shows the maximum mode timings for the read operation while the fig. 1.3 shows the same for the write operation.



Memory Read Timing in Maximum Mode



Memory Write Timing in Maximum mode.

Timings for RQ*/GT*

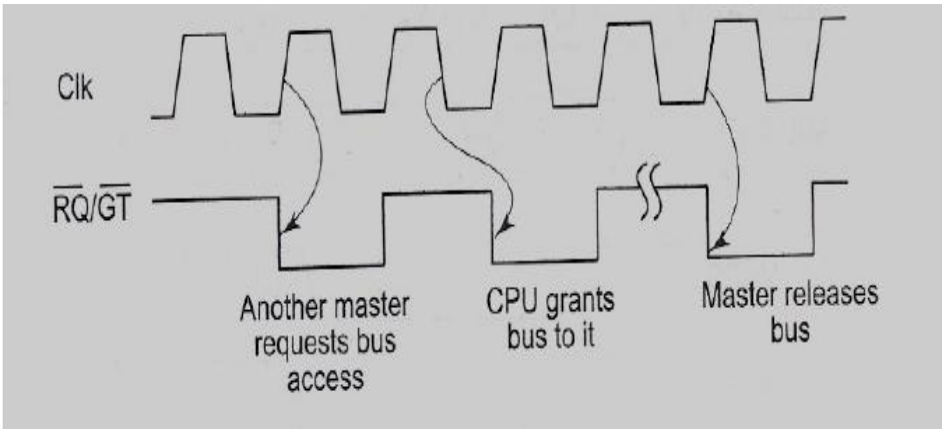


Fig: RQ*/GT* Timings in Maximum Mode

Memory segmentation:

- Since address registers and address operands are only 16 bits they can only address 64k bytes. In order to address the 20-bit address range of the 8086, physical addresses (those that are put on the address bus)are always formed by adding the values of one of the instruction is executed? The use of segment registers reduces the size of pointers to 16 bits.
- This reduces the code size but also restricts the addressing range of a pointer to 64k bytes. Performing address arithmetic within data structures larger than 64k is awkward.

This is the biggest drawback of the 8086 architecture. We will restrict ourselves to short programs where all of the code, data and stack are placed into the same 64k segment (i.e. CS=DS=SS).

- Most of the registers contain data/instruction offsets within 64 KB memory segment. There are four different 64 KB segments for instructions, stack, data and extra data. To specify where in 1 MB of processor memory these 4 segments are located the processor uses four segment registers:

Memory

- Program, data and stack memories occupy the same memory space. As the most of the processor instructions use 16-bit pointers the processor can effectively address only 64 KB of memory.
- To access memory outside of 64 KB the CPU uses special segment registers to specify where the code, stack and data 64 KB segments are positioned within 1 MB of memory (see the "Registers" section below).
- 16-bit pointers and data are stored as: address: low-order byte
  Address +1: high-order byte
- Program memory - program can be located anywhere in memory. Jump and call instructions can be used for short jumps within currently selected 64 KB code segment, as well as for far jumps anywhere within 1 MB of memory.
- All conditional jump instructions can be used to jump within approximately +127 to -127 bytes from current instruction.
- Data memory - the processor can access data in any one out of 4 available segments, which limits the size of accessible memory to 256 KB (if all four segments point to different 64 KB blocks).
- Accessing data from the Data, Code, Stack or Extra segments can be usually done by prefixing instructions with the DS:, CS:, SS: or ES: (some registers and instructions by default may use the ES or SS segments instead of DS segment).
- Word data can be located at odd or even byte boundaries. The processor uses two memory accesses to read 16-bit word located at odd byte boundaries. Reading word data from even byte boundaries requires only one memory access.
- Stack memory can be placed anywhere in memory. The stack can be located at odd memory addresses, but it is not recommended for performance reasons (see "Data Memory" above).

Reserved locations:

- 0000h - 03FFh are reserved for interrupt vectors. Each interrupt vector is a 32-bit pointer in format segment: offset.
- FFFF0h – FFFFF h - after RESET the processor always starts program execution at the FFFF0h address.
- Segment registers to the 16-bit address to form a 20-bit address.

- The segment registers themselves only contain the most-significant 16 bits of the 20-bit

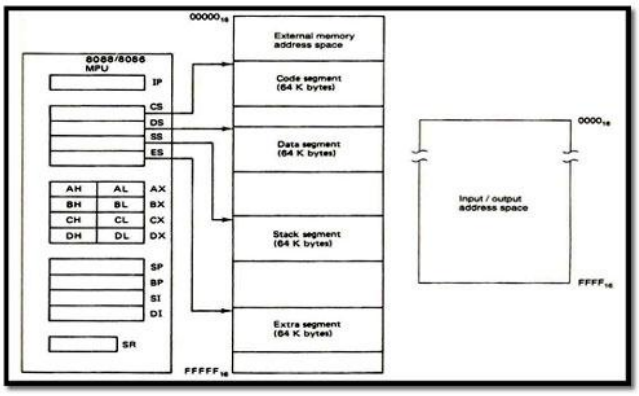
value that is contributed by the segment registers. The least significant four bits of the segment address are always zero.

- By default, the DS (data segment) is used for data transfer instructions (e.g. MOV), CS (code segment) is used with control transfer instructions (e.g. JMP or CALL), and SS is used with the stack pointer (e.g. PUSH or to save/restore addresses during CALL/RET or INT instructions).

  Exercise: If DS contains 0100H, what address will be written by the instruction MOV [2000H],AL? If CX contains 1122H, SP contains 1234H, and SS contains 2000H, what memory values will change and what will be their values when the PUSH CX

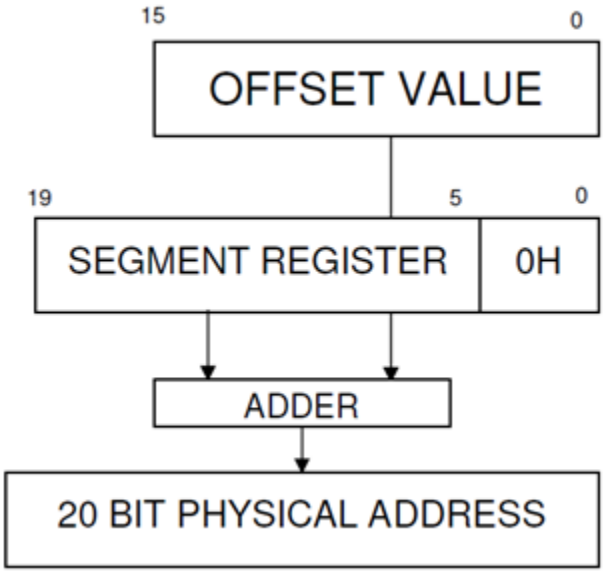
- Code segment (CS) is a 16-bit register containing address of 64 KB segment with processor instructions. The processor uses CS segment for all accesses to instructions referenced by instruction pointer (IP) register. CS register cannot be changed directly.
- The CS register is automatically updated during far jump, far call and far return instructions.
- 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.

- 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.
- 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 is possible to change default segments used by general and index registers by prefixing instructions with a CS, SS, DS or ES prefix.



Physical address formation

- The 8086 addresses a segmented memory. The complete physical address which is 20-bits long is generated using segment and offset registers each of the size 16-bit.The content of a segment register also called as segment address, and content of an offset register also called as offset address. To get total physical address, put the lower nibble 0H to segment address and add offset address. The fig 1.3 shows formation of 20-bit physical address.



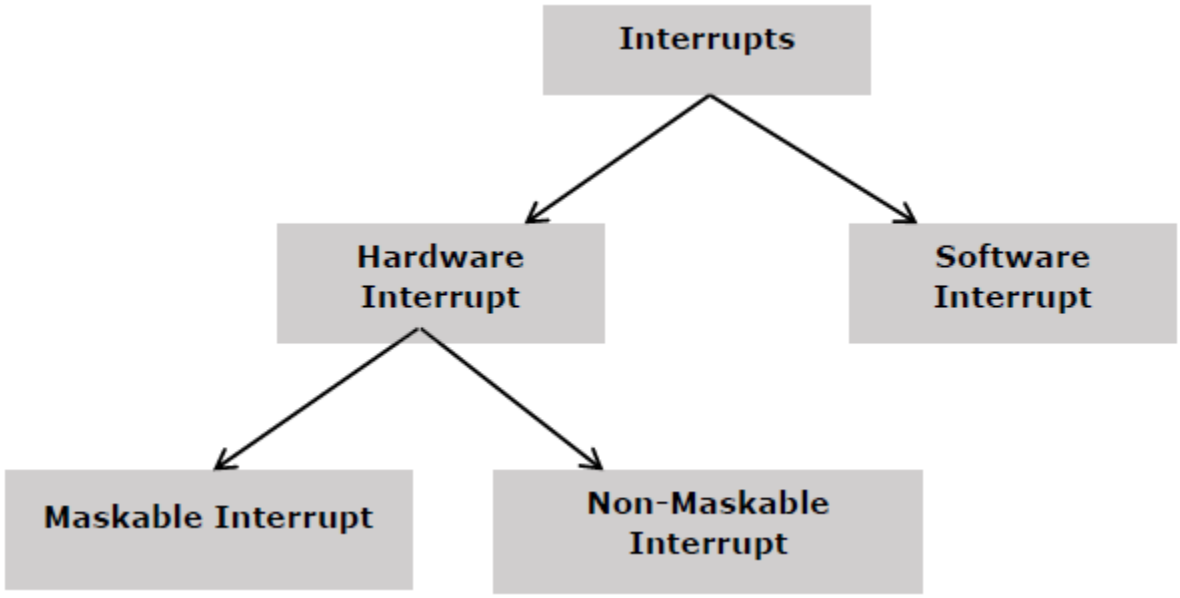
Physical address formation

Interrupts:

Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor.

The microprocessor responds to that interrupt with an **ISR** Interrupt Service Routine Interrupt Service Routine, which is a short program to instruct the microprocessor on how to handle the interrupt.

The following image shows the types of interrupts we have in a 8086 microprocessor −



Hardware Interrupts

Hardware interrupt is caused by any peripheral device by sending a signal through a specified pin to the microprocessor.

The 8086 has two hardware interrupt pins, i.e. NMI and INTR. NMI is a non-maskable interrupt and INTR is a maskable interrupt having lower priority. One more interrupt pin associated is INTA called interrupt acknowledge.

NMI

It is a single non-maskable interrupt pin NMINMI having higher priority than the maskable interrupt request pin INTRINTRand it is of type 2 interrupt.

When this interrupt is activated, these actions take place −

- Completes the current instruction that is in progress.
- Pushes the Flag register values on to the stack.
- Pushes the CS code segment code segment value and IP instruction pointer instruction pointer value of the return address on to the stack.
- IP is loaded from the contents of the word location 00008H.

- CS is loaded from the contents of the next word location 0000AH.
- Interrupt flag and trap flag are reset to 0.

INTR

The INTR is a maskable interrupt because the microprocessor will be interrupted only if interrupts are enabled using set interrupt flag instruction. It should not be enabled using clear interrupt Flag instruction.

The INTR interrupt is activated by an I/O port. If the interrupt is enabled and NMI is disabled, then the microprocessor first completes the current execution and sends '0' on INTA pin twice. The first '0' means INTA informs the external device to get ready and during the second '0' the microprocessor receives the 8 bit, say X, from the programmable interrupt controller.

These actions are taken by the microprocessor −

- First completes the current instruction.
- Activates INTA output and receives the interrupt type, say X.
- Flag register value, CS value of the return address and IP value of the return address are pushed on to the stack.
- IP value is loaded from the contents of word location X × 4
- CS is loaded from the contents of the next word location.
- Interrupt flag and trap flag is reset to 0

Software Interrupts

Some instructions are inserted at the desired position into the program to create interrupts. These interrupt instructions can be used to test the working of various interrupt handlers. It includes −

INT- Interrupt instruction with type number

It is 2-byte instruction. First byte provides the op-code and the second byte provides the interrupt type number. There are 256 interrupt types under this group.

Its execution includes the following steps −

- Flag register value is pushed on to the stack.

- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 'type number' × 4
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

The starting address for type0 interrupt is 000000H, for type1 interrupt is 00004H similarly for type2 is 00008H and ……so on. The first five pointers are dedicated interrupt pointers. i.e. −

- **TYPE 0** interrupt represents division by zero situation.
- **TYPE 1** interrupt represents single-step execution during the debugging of a program.
- **TYPE 2** interrupt represents non-maskable NMI interrupt.
- **TYPE 3** interrupt represents break-point interrupt.
- **TYPE 4** interrupt represents overflow interrupt.

The interrupts from Type 5 to Type 31 are reserved for other advanced microprocessors, and interrupts from 32 to Type 255 are available for hardware and software interrupts.

INT 3-Break Point Interrupt Instruction

It is a 1-byte instruction having op-code is CCH. These instructions are inserted into the program so that when the processor reaches there, then it stops the normal execution of program and follows the break-point procedure.

Its execution includes the following steps −

- Flag register value is pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of the word location 3×4 = 0000CH
- CS is loaded from the contents of the next word location.
- Interrupt Flag and Trap Flag are reset to 0

INTO - Interrupt on overflow instruction

It is a 1-byte instruction and their mnemonic **INTO**. The op-code for this instruction is CEH. As the name suggests it is a conditional interrupt instruction, i.e. it is active only when the overflow flag is set to 1 and branches to the interrupt handler whose interrupt

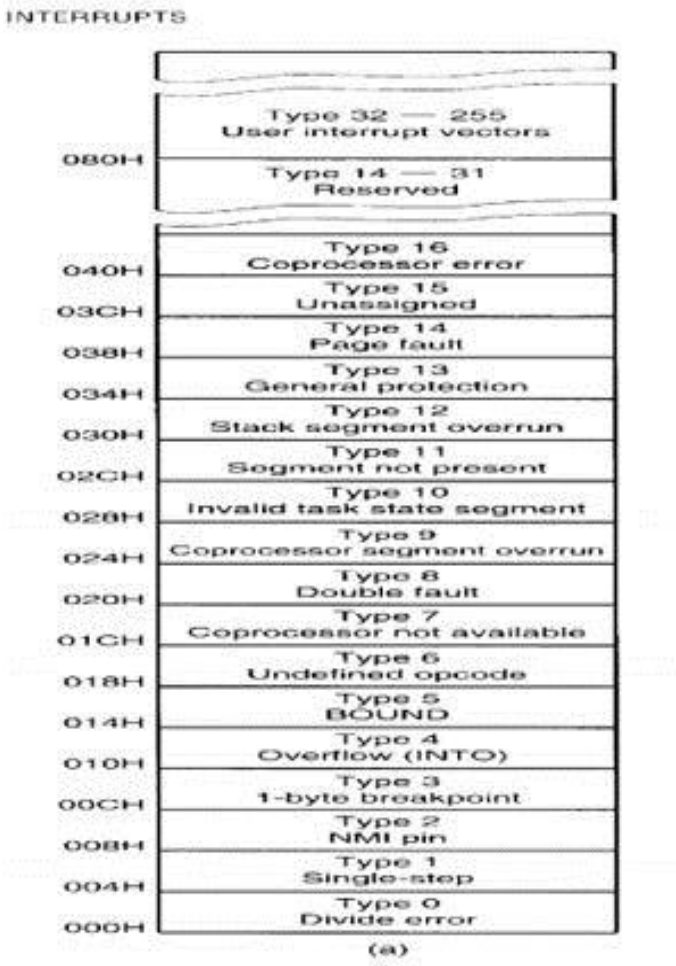
type number is 4. If the overflow flag is reset then, the execution continues to the next instruction.

Its execution includes the following steps −

- Flag register values are pushed on to the stack.
- CS value of the return address and IP value of the return address are pushed on to the stack.
- IP is loaded from the contents of word location 4×4 = 00010H
- CS is loaded from the contents of the next word location.
- Interrupt flag and Trap flag are reset to 0

Interrupt Vector Table (IVT) on 8086

Interrupt vector table on 8086 is a vector that consists of 256 total interrupts placed at first 1 kb of memory from 0000h to 03ffh, where each vector consists of segment and offset as a lookup or jump table to memory address of bios interrupt service routine (f000h to ffffh) or dos interrupt service routine address, the call to interrupt service routine is similar to far procedure call.
The size for each interrupt vector is 4 bytes (2 word in 16 bit), where 2 bytes (1 word) for segment and 2 bytes for offset of interrupt service routine address.
So it takes 1024 bytes (1 kb) memory for interrupt vector table. On 8086 with dos operating system, interrupt vector table at 00h-1fh (int num 0-31) consists of lookup / jump table address to hardware or bios interrupt handler routine, meanwhile 20h-ffh (int num 32-255) consist of jump table address to dos interrupt handler routine.
For example int 13h that located on ivt at 0000:004c contains address of Bios ROM Interrupt Service Routine, what it records is segment F000h, and offset 1140h, each bytes of that address will be placed little endian. The lower the interrupt number on interrupt vector table means the more priority needed for an interrupt.



Physical Memory Organization:

In an 8086 based system, the 1Mbyte memory is physically organized as odd bank and even bank, each of 512kbytes, addressed in parallel by the processor. Byte data with even address is transferred on $D_7 - D_0$ and byte data with odd address is transferred on $D_{15} - D_8$. The processor provides two enable signals, $\overline{BHE}$ and A_0 for selecting of either even or odd or both the banks.

$D_{15} - D_0 / A_{15} - A_0$ are the common signal line in 8086 as $AD_{15} - AD_0$

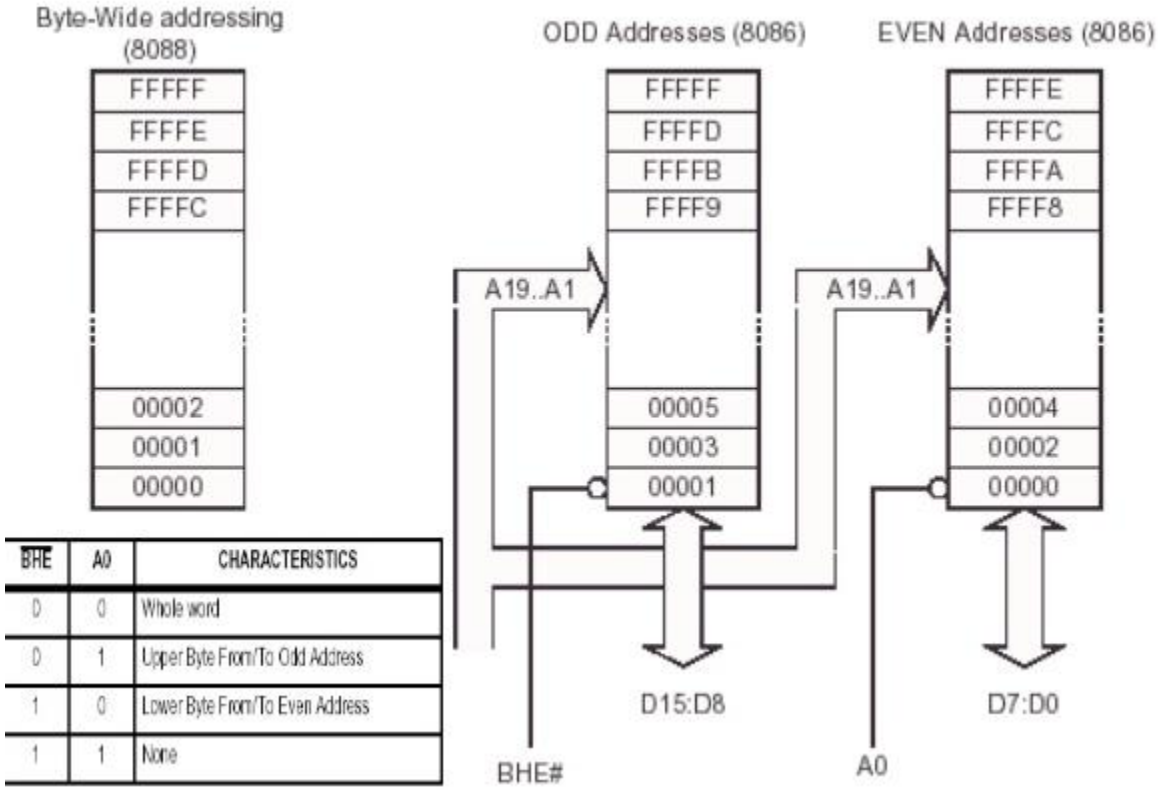
$\overline{BHE}$	A_0	Function
0	0	Whole word

0	1	Upper byte/ odd address
1	0	Lower byte/even address
1	1	none

If the address bus contains $FFFFO_H$

If $\overline{BHE} = 0$, then it reads the 16 bits data from memory location $FFFFO_H$ and $FFFF1_H$.

If $\overline{BHE} = 1$, it reads the 8 bits data from memory location $FFFFO_H$



$\overline{BHE}$	A0	CHARACTERISTICS
0	0	Whole word
0	1	Upper Byte From/To Odd Address
1	0	Lower Byte From/To Even Address
1	1	None