The 8086 Microprocessor Architecture

Features of 8086

- 8086 Microprocessor is an enhanced version of 8085 Microprocessor that was designed by Intel in 1976.
- It is a **16-bit Microprocessor**, means that its arithmetic logic unit, internal registers and most of its instructions are designed to work with 16-bit binary word.
- **Data bus** 0f 8086 microprocessor is of **16-bit**, hence it can read or write 16-bit or 8-bit data at a time from memory or I/O devices.
- 8086 microprocessor has **20-bit address bus**, therefore, it can access 2^{20} = 10,48,576 memory location (max:1MB of memory)

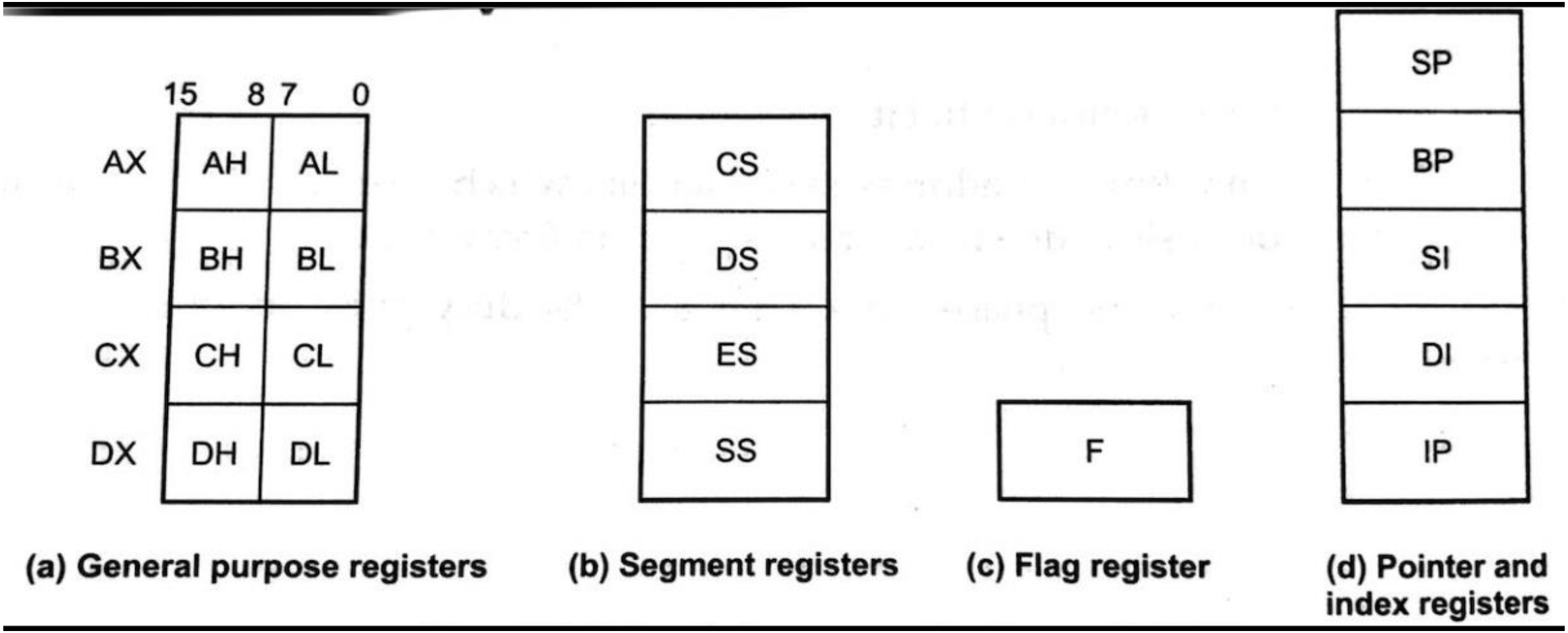
Features of 8086

- It supports two modes of operation:-
  - **Maximum mode**: suitable for system having multiple processors
  - **Minimum mode**: suitable for system having a single processor.
- It 8086 can generate 16-bit I/O address, hence it can access 2^{16} = 65536 I/O ports.
- It has an **instruction queue**, which is capable of *storing six instruction bytes from the memory* resulting in faster processing.
- It uses two stages of pipelining, i.e. Fetch Stage and Execute Stage, which improves performance.
- Fetch stage can prefetch up to 6 bytes of instructions and stores them in the queue. Execute stage executes these instructions.

Comparison between 8085 & 8086 Microprocessor

- **Size** – 8085 is 8-bit microprocessor, whereas 8086 is 16-bit microprocessor.
- **Address Bus** – 8085 has 16-bit address bus while 8086 has 20-bit address bus.
- **Memory** – 8085 can access up to 64Kb, whereas 8086 can access up to 1 Mb of memory.
- **Instruction** – 8085 doesn’t have an instruction queue, whereas 8086 has an instruction queue.
- **Pipelining** – 8085 doesn’t support a pipelined architecture while 8086 supports a pipelined architecture.
- **I/O** – 8085 can address 2^8 = 256 I/O's, whereas 8086 can access 2^16 = 65,536 I/O's.

Register Organization of 8086 Microprocessor



Register Organization of 8086 Microprocessor

a) GENERAL PURPOSE REGISTER

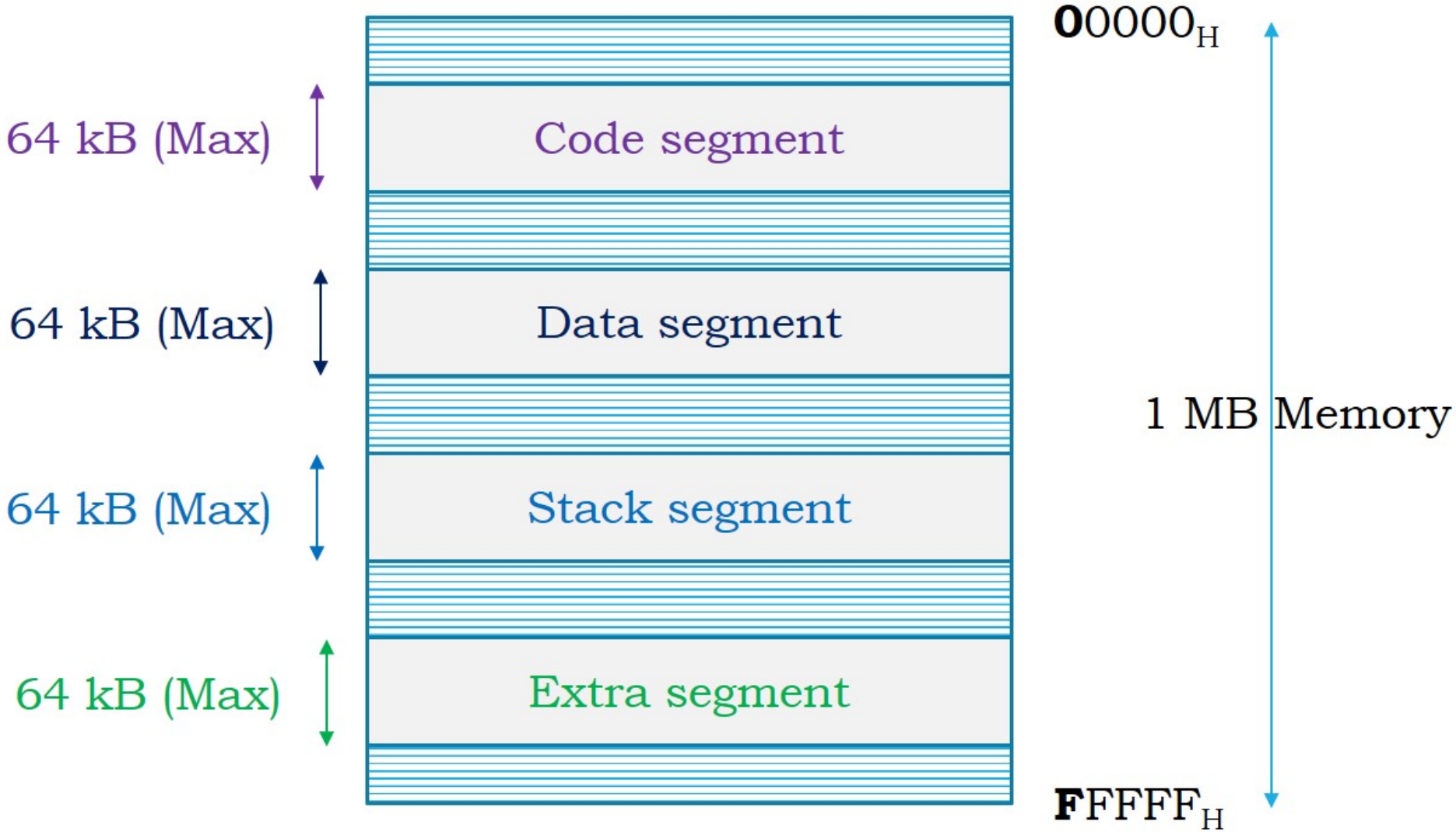
- There are 8 general purpose registers, i.e., AH, AL, BH, BL, CH, CL, DH, and DL.
- These registers can be used individually to store 8-bit data and can be used in pairs to store 16-bit data.
- The valid register pairs are AH and AL, BH and BL, CH and CL, and DH and DL. It is referred to the AX, BX,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

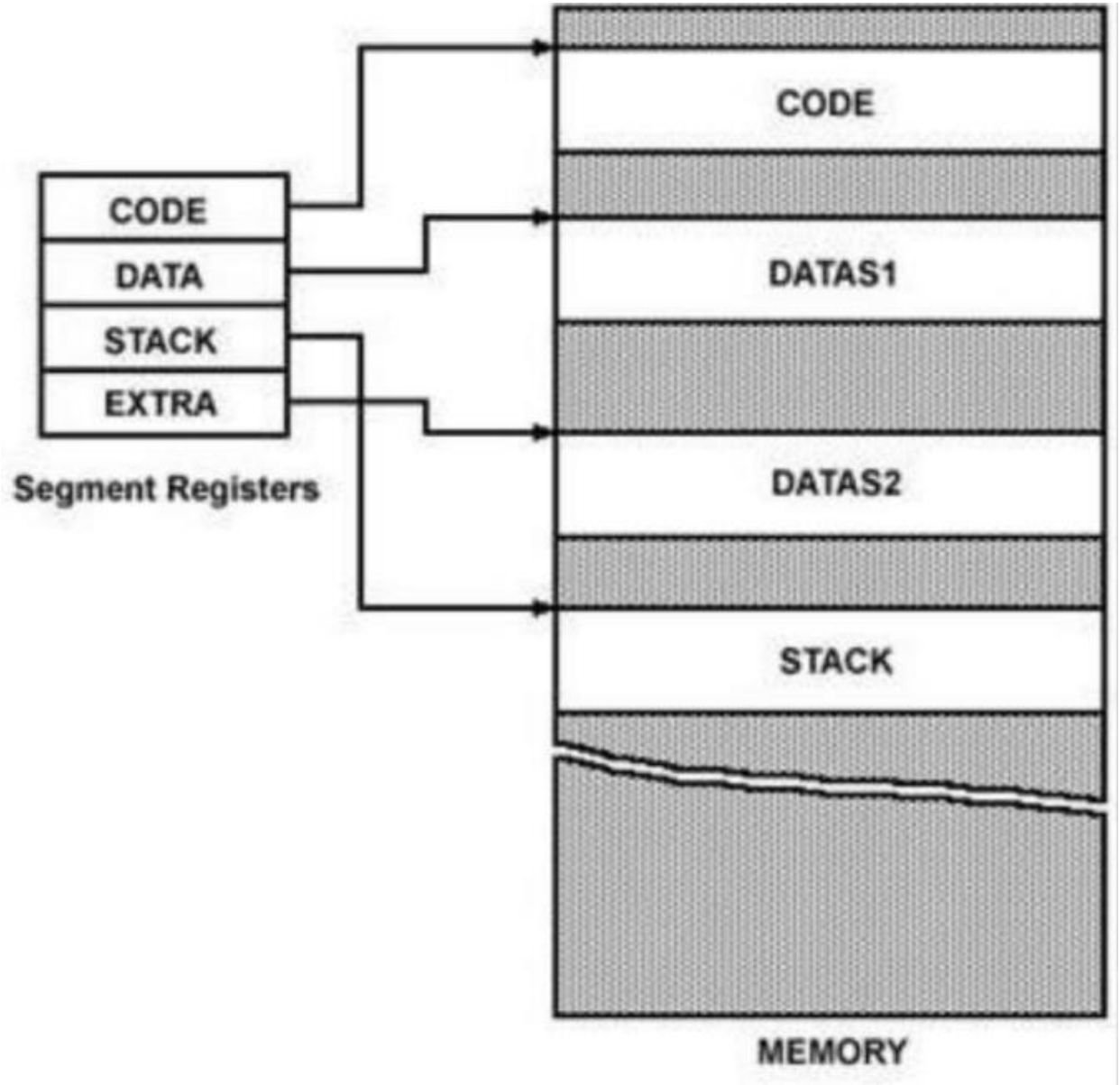
Register Organization of 8086 Microprocessor

b) SEGMENT REGISTERS

- The **Physical address** of 8086 is *20-bits*, but its registers and memory location uses **Logical addresses** which are *16-bit* wide. Therefore, 8086 uses **Memory Segmentation**.
- Total 1MB memory is divided into Segments, each segment having maximum size of 64KB.
- Four Segments:-
  1. Code Segment (CS): It contains the instructions of a program.
  2. Data Segment (DS): It contains data for the program.
  3. Stack Segment (SS): It holds the stack of the program.
  4. Extra Segment (ES): It is an additional data segment that is used by some string instructions.

Memory Segmentation of Intel 8086





Each of these segments are addressed by an address stored in corresponding segment register.

These registers are 16-bit in size.

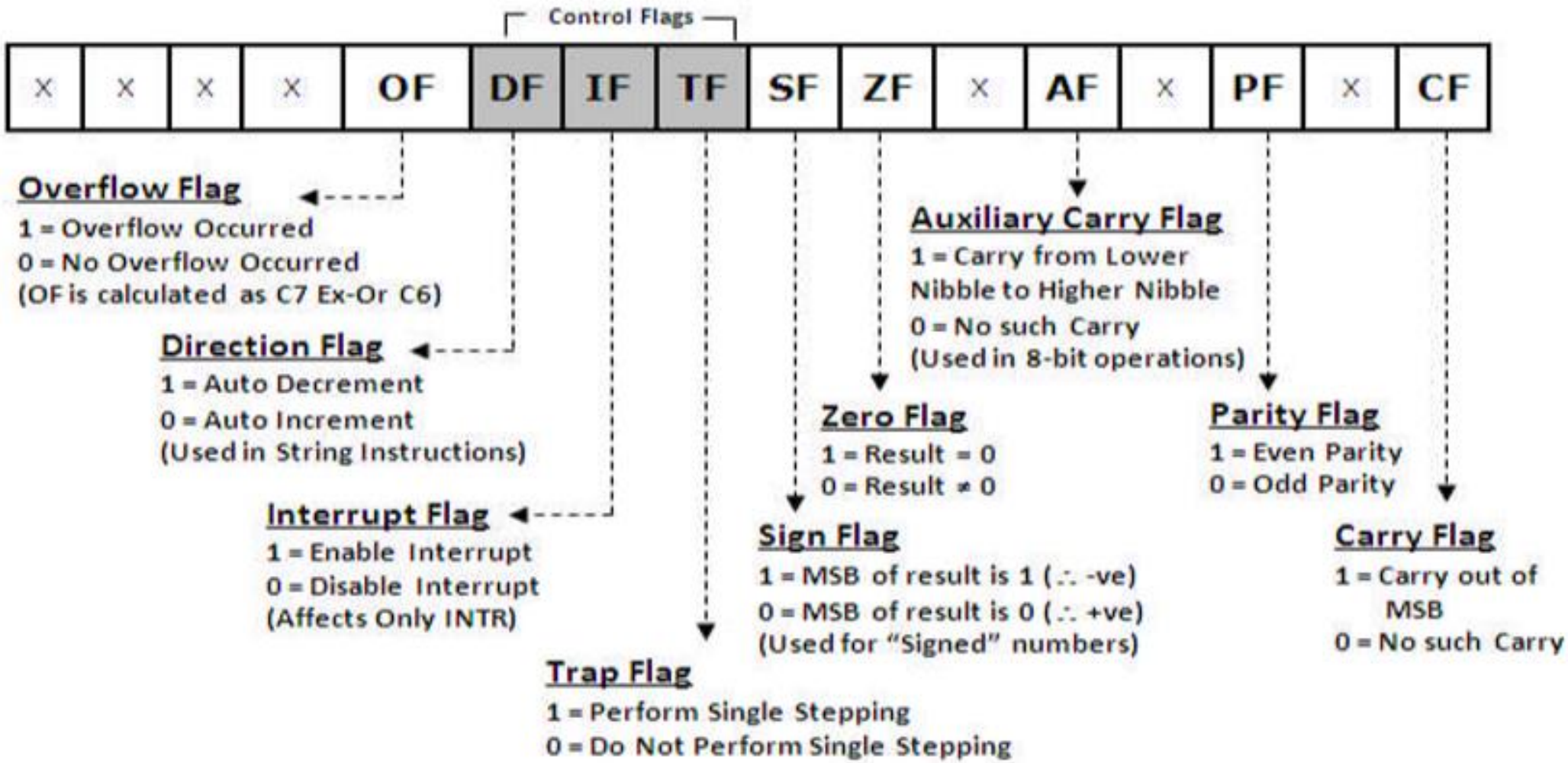
Each register stores the base address (starting address) of the corresponding segment.

Register Organization of 8086 Microprocessor

c) POINTERS AND INDEX REGISTERS

- All segment registers are 16-bit wide. But it is necessary to generate 20-bit address (physical address) on the address bus. To get 20-bit physical address one or more pointer or index register are associated with each segment register.
- **Pointer Registers:-**
  - **Instruction Pointer (IP)** register is associated with CS (code segment)
  - **Base Pointer (BP)** register is associated with DS (data segment)
  - **Stack Pointer (SP)** register is associated with SS (stack segment)
- **Index Register:** **SI (Source Index)** and **DI (Destination Index)** registers are used as offset in indexed, base indexed and relative base indexed addressing modes. Also, for some string manipulation instructions.

Flag Register Of 8086 Microprocessor



Carry Flag (CF) The carry flag is set to 1 if after arithmetic operation a carry is generated or a borrow is generated in subtraction. When there is no carry out, the carry flag is reset or zero. This flag can also be used in some shift and rotate instructions.

Parity Flag (PF) If the result of 8-bit operation or lower byte of the word operation contains an even number of 1s, parity flag is set.

Auxiliary Carry Flag (AF) This flag is set to 1 if there is a carry out of the lower nibble to the higher nibble of an 8-bit operation. It is used for BCD operations.

Zero Flag (ZF) The zero flag is set to 1 if the result of any arithmetic or logical operation is zero. While the result is zero, it is reset.

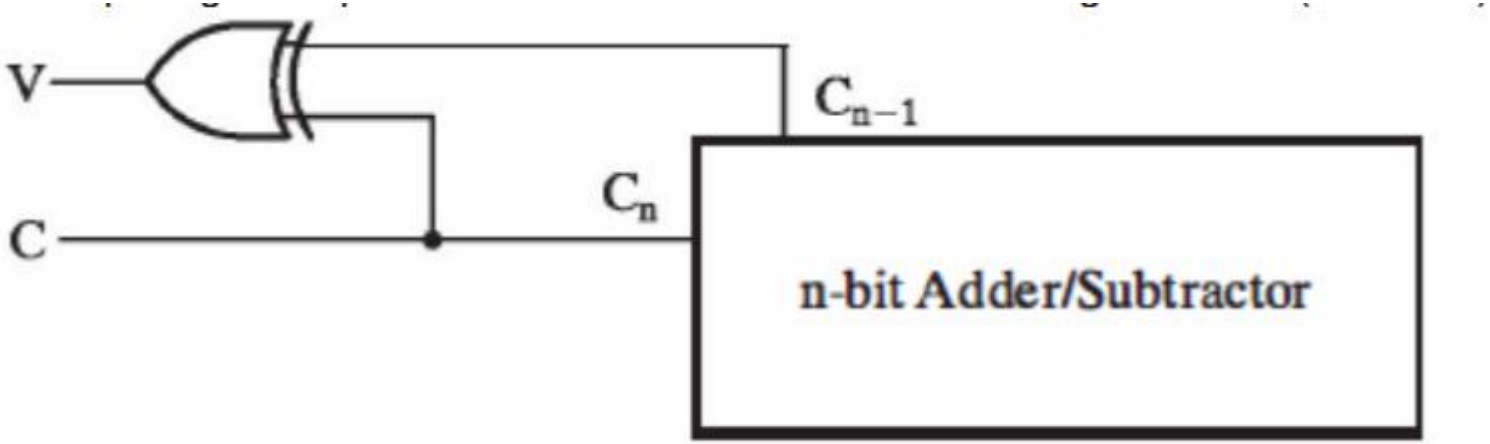
Sign Flag (SF) The sign flag is set to 1, if the MSB of the result is 1 after the arithmetic or logic operations. This flag represents a sign number. Logic 0 indicates a positive number and logic 1 is used to represent negative number.

Overflow Flag (OF) This flag is set to 1 if the signed result cannot be expressed within the number of bits in the destination operand. This flag is used to detect magnitude overflow in signed arithmetic operations. During addition operation, the flag is set when there is a carry into the MSB and the flag is reset if there is no carry out of the MSB. For subtraction operation, the flag is set when the MSB desires a borrow, and the flag is reset if there is no borrow from MSB.

Direction Flag (DF) The direction flag is used in string operations. When it is set to 1, string bytes can be accessed from a memory address in decrement order, i.e., high memory address to low memory address. If it is zero, string bytes can be accessed from memory address in increasing order, i.e., low memory address to high memory address. For example, in MOVS instruction if DF is set to 1, the contents of the index registers SI and DI are automatically decremented to access the string bytes. If DF = 0, index registers SI and DI are automatically incremented to access the string bytes.

Interrupt Enable Flag (IF) This flag can be used as an interrupt enable or disable flag. When this flag is set, the maskable interrupt is enabled and 8086 recognizes the external interrupt requests, and the CPU transfer control to an interrupt vector specified location. When IF is 0, all maskable interrupts are disabled and there will be no effect on nonmaskable interrupts as well as internally generated interrupts. If 8086 is reset, IF is automatically cleared.

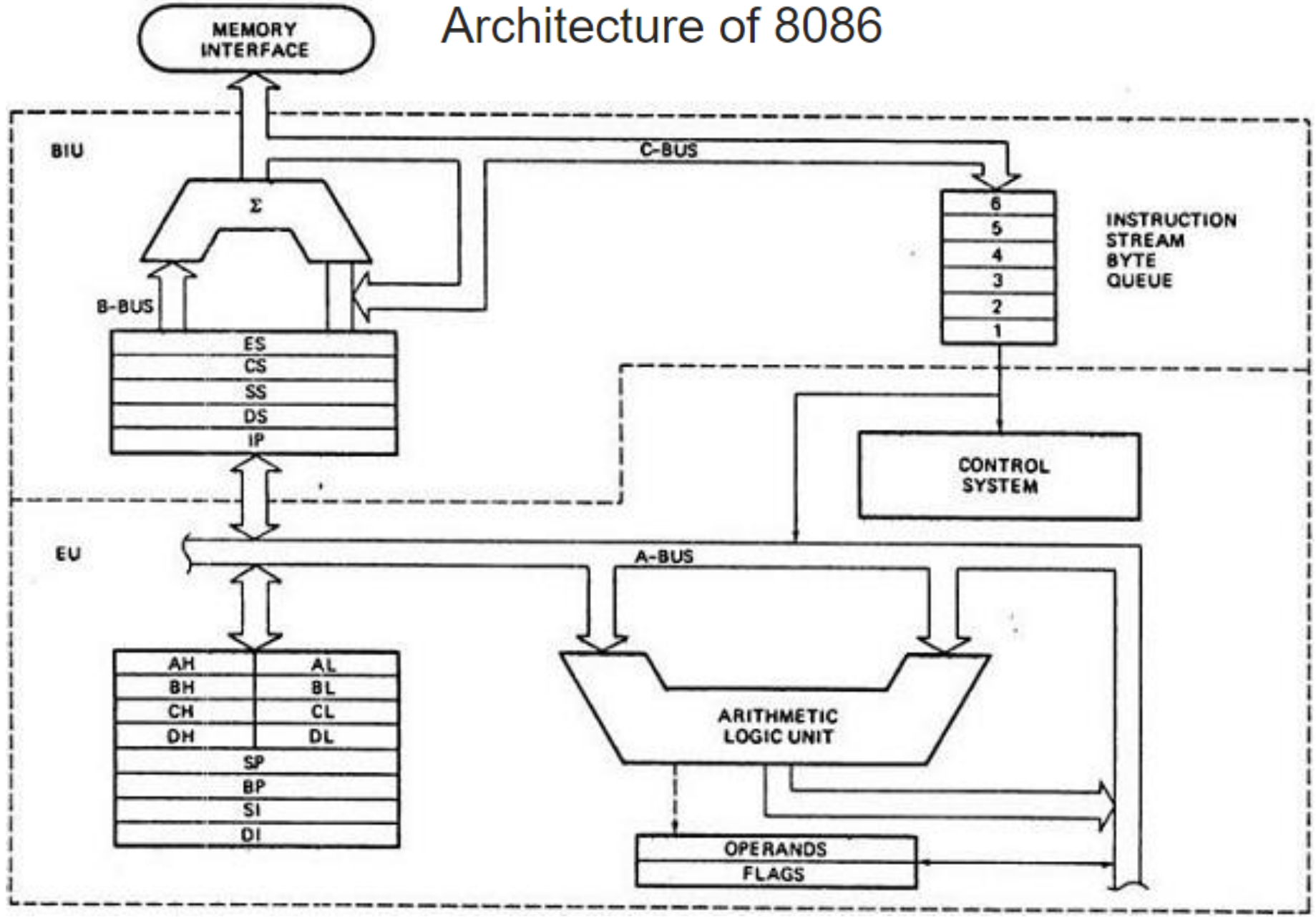
Trap Flag (TF) TF is a single-step flag. When TF is set to 1, a single step interrupt occurs after the next instruction executes and the program can be executed in single-step mode. The TF will be cleared by the single-step interrupt.



Overflow Detection Logic for Addition and Subtraction

An example, suppose we add 127 and 127 using 8-bit registers. 127+127 is 254, but using 8-bit arithmetic the result would be 1111 1110 binary, which is the two's complement encoding of −2, a negative number. A negative sum of positive operands (or vice versa) is an overflow.

Architecture of 8086



Bus Interface Unit [BIU]

The bus interface unit is the 8086's interface to the outside world. It provides a full 16-bit bi-directional data bus and 20-bit address bus. The bus interface unit is responsible for performing all external bus operations, as listed below.

Functions of Bus Interface Unit

1. It sends address of the memory or I/O.
2. It fetches instruction from memory.
3. It reads data from port/memory.
4. It writes data into port/memory.
5. It supports instruction queuing.
6. It provides the address relocation facility.

To implement these functions the BIU contains the instruction queue, segment registers instruction pointer, address summer and bus control logic.

Execution Unit [EU]

The execution unit of 8086 tells the BIU from where to fetch instructions or data, decodes instructions and executes instructions. It contains

- Control Circuitry
- Instruction Decoder
- Arithmetic Logic Unit (ALU)
- Flag Register
- General Purpose Registers
- Pointers and Index Registers

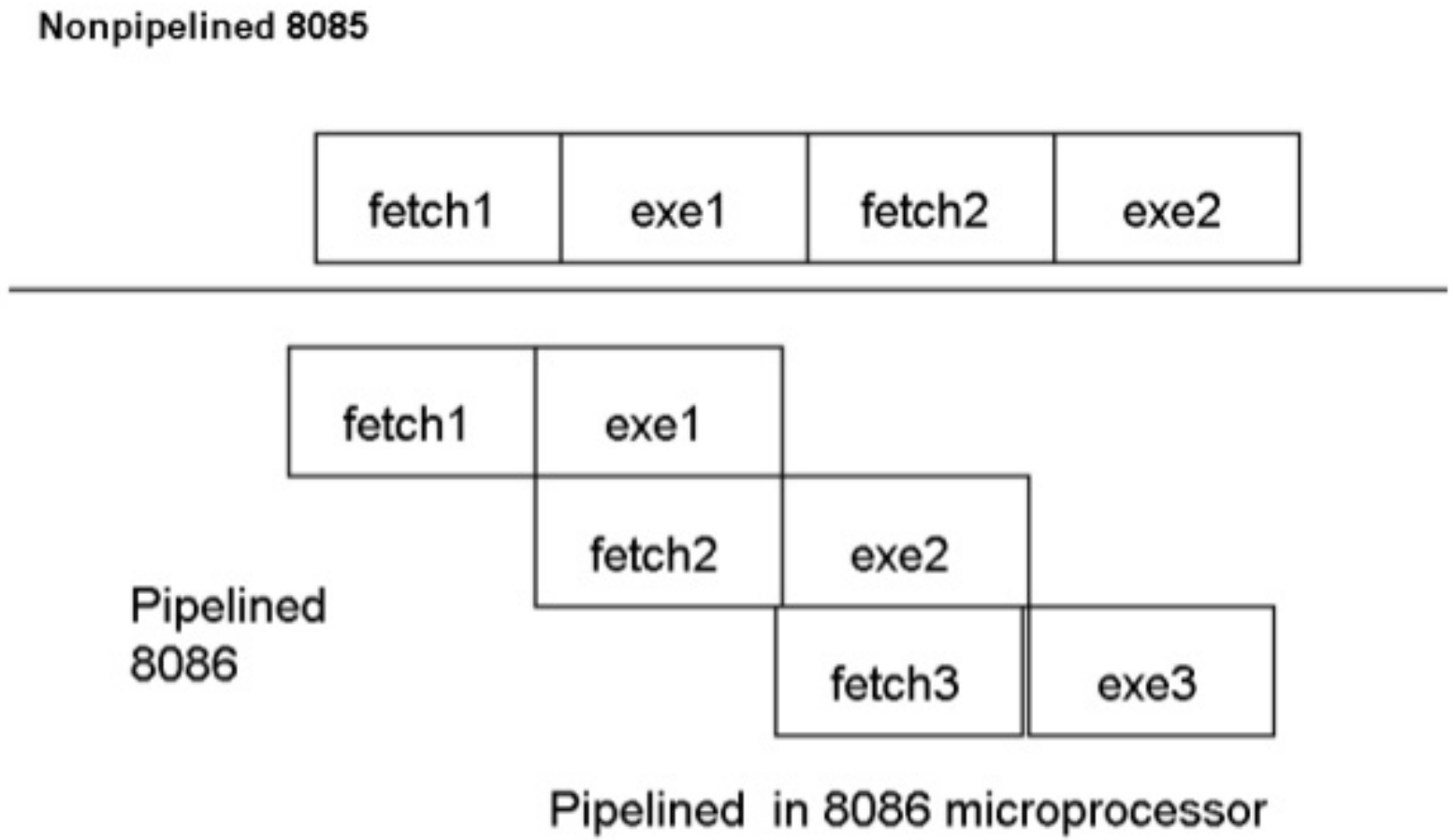
Control Circuitry, Instruction Decoder, ALU

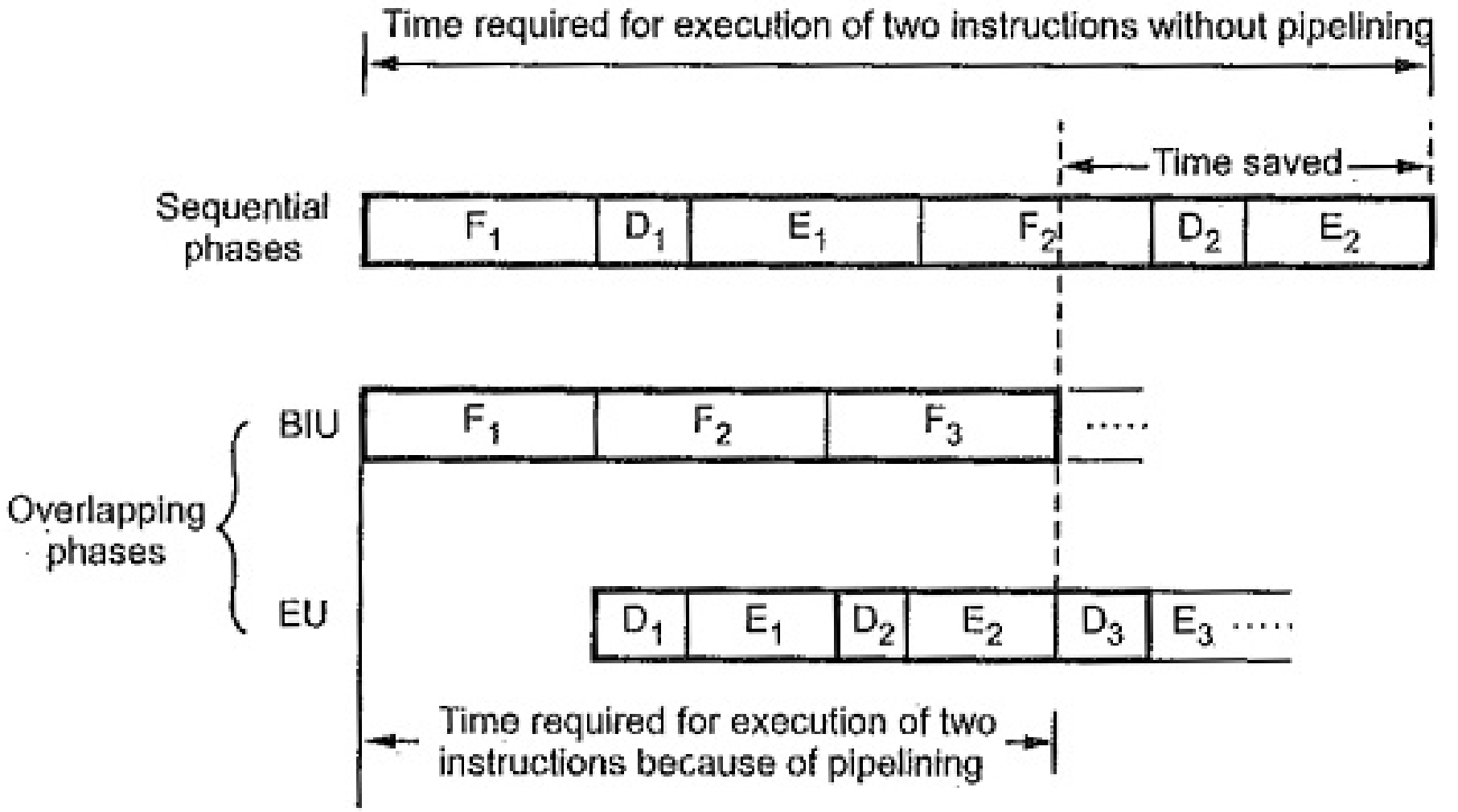
The control circuitry in the EU directs the internal operations. A decoder in the EU translates the instructions fetched from memory into a series of actions which the EU performs. ALU is 16-bit. It can add, subtract, AND, OR, XOR, increment, decrements, complement and shift binary numbers.

Instruction queue in 8086

- 8086 provides queuing facility where BIU fetches the six instruction byte from memory while current instruction is being executed. It speed up the program execution.
- It works on first in first out (FIFO) fashion.
- The size of queue for 8086 is 6 bytes.
- BIU fetches the instruction code from memory and stores in queue, EU fetches the instruction from queue for execution.

Pipelining in 8086




Time required for execution of two instructions without pipelining
Time saved
Sequential phases
F1
D1
E1
F2
D2
E2
Overlapping phases
BIU
F1
F2
F3
.....
EU
D1
E1
D2
E2
D3
E3 .....
Time required for execution of two instructions because of pipelining

Pipelining

What is pipelining?

- The greater performance of the cpu is achieved by instruction pipelining.
- 8086 microprocesor has two blocks

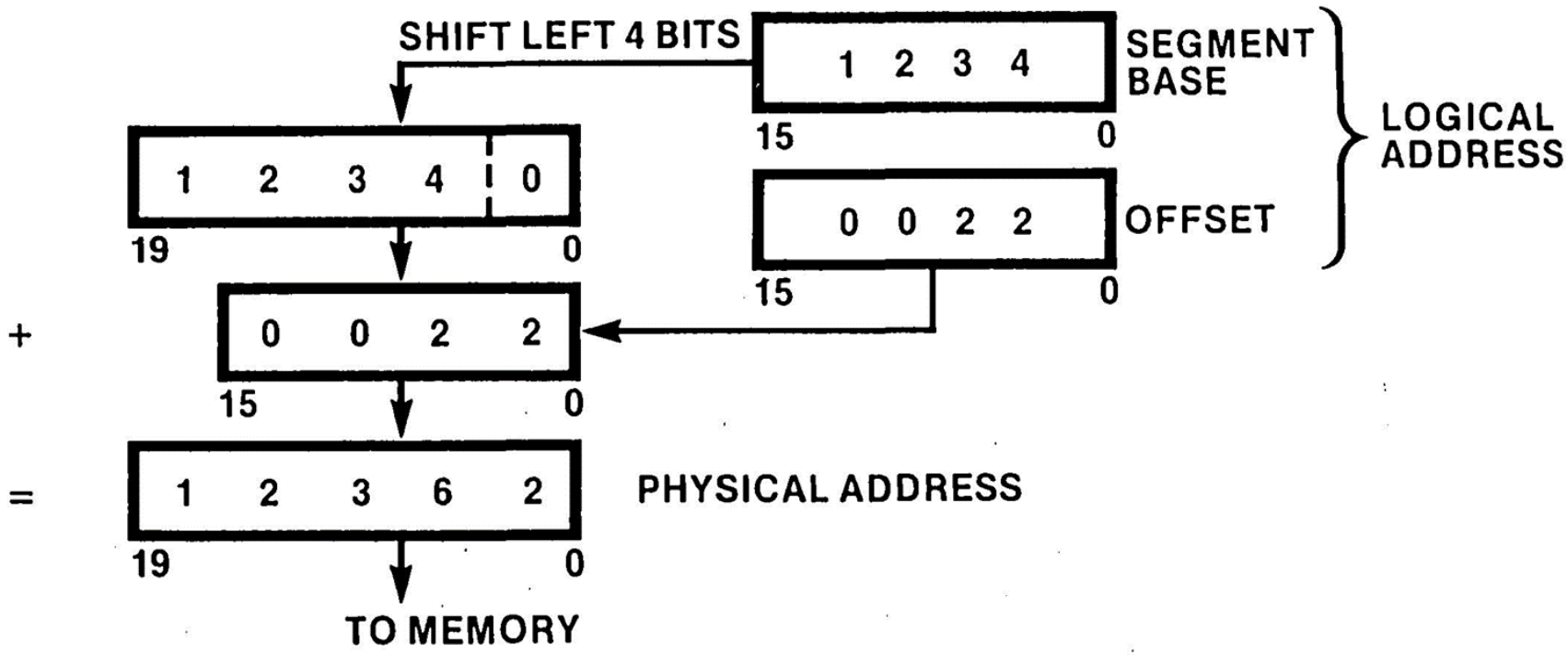
  - BIU(BUS INTERFACE UNIT)
  - EU(EXECUTION UNIT)

- The BIU performs all bus operations such as instruction fetching,reading and writing operands for memory and calculating the addresses of the memory operands. The instruction bytes are transferred to the instruction queue.
- EU executes instructions from the instruction system byte queue.
- Both units operate asynchronously to give the 8086 an overlapping instruction fetch and execution mechanism which is called as Pipelining.

- Pipelining is the process of fetching the next instruction when the current instruction is being executed.

➢Generation of 20-bit Physical Address

- To access a specific memory location from any segment we need 20-bit Physical address.
- 8086 generates this address using the contents of **SEGMENT REGISTER** and the **OFFSET REGISTER** associated with it.



- The physical address is calculated from the segment address and offset address.
- The segment register contains the higher-order 16 bits of the starting address of a memory segment.
- **The CPU shifts the content of the segment register left by four bits or inserts four zeros for the lowest four bits of the 20-bit memory address.**
- The adder circuit in the BIU section of the 8086 Architecture adds contents of segment base and the offset to obtain the 20-bit physical address

Determine the physical address when CS = 5300H and IP = 0200H. Write the starting and ending address of the code segment.

Solution

The content of the code segment is left shifted by 4 bits and the base address becomes 53000H. To determine the physical address, the content of IP will be added with base address. Hence physical address = 53000 + 0200 = 53200H.

The starting of code segment memory = 53200H.

As each segment memory consists of 64K memory locations, the end address will be computed after addition of 64K with the starting of code segment memory.

The ending address of code segment = 53200 + FFFF = 631FFH.

Determine the physical address when ES is 6500H and offset address is 4767H

Solution

The content of the segment register ES is 6500H. When it is left shifted by 4 bits or multiplied by $(16)_D$ or $(10)_H$, the base address is equal to 6500H × (10)H = 65000H.

Physical address = Content of segment register $\times(10)_H$ + Offset address

= 6500H$\times(10)_H$ + 4567H = 65000H + 4567H = 69567H

What is the content of data segment DS to locate the physical address 43657H? Assume the content of IP = 2057H.

Solution

The physical address is 43657H when the content of IP is 2057H

Physical address = Content of data segment register $\times(10)_H$ + IP address

Therefore,

43657H = Content of data segment register $\times(10)_H$ + 2057H

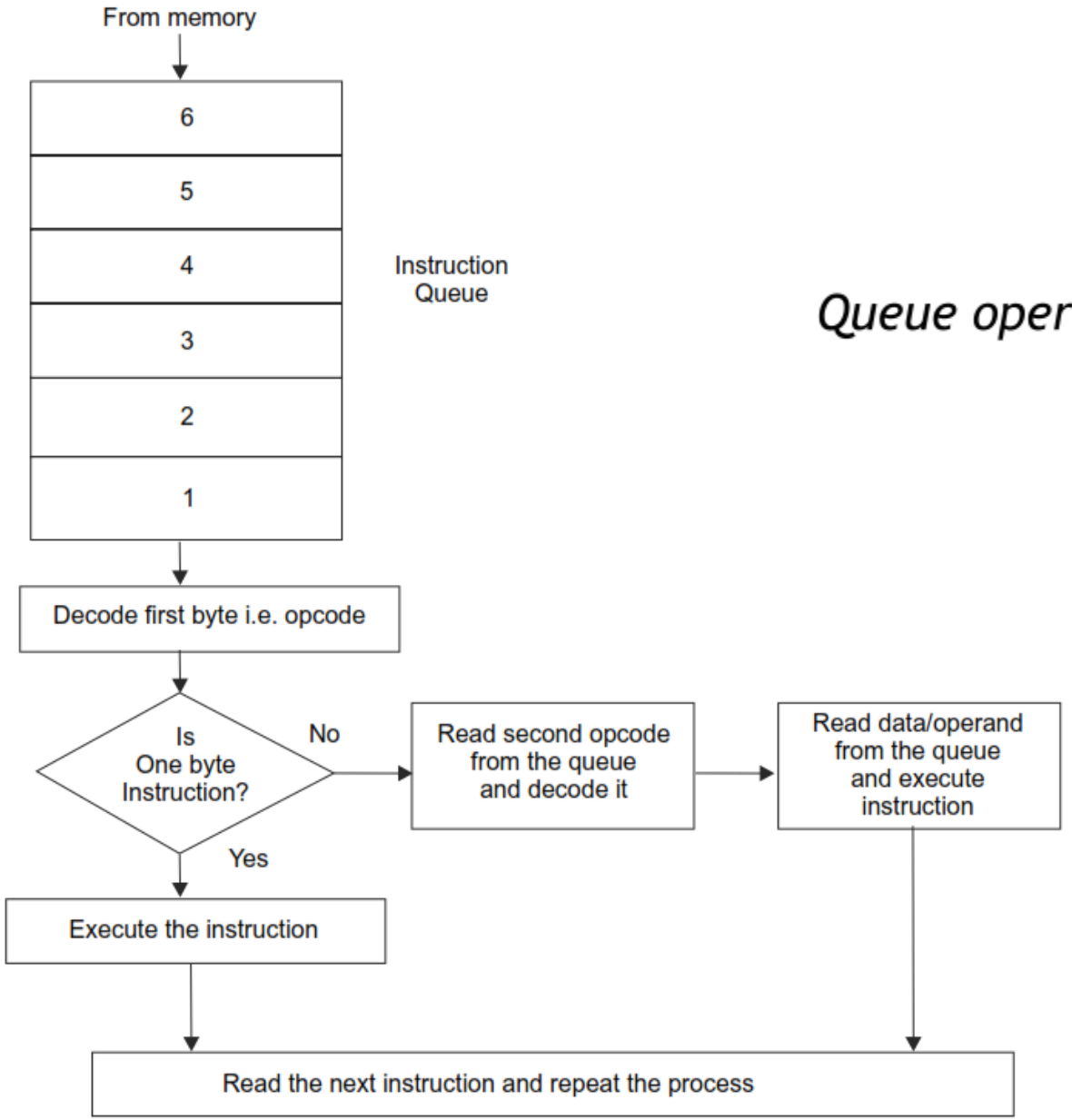
Then the content of data segment register × $(10)_H$ = 43657H – 2057H = 4 1600H

The content of data segment register (DS) is 4160H.

Process of Fetching and Decoding of Instructions

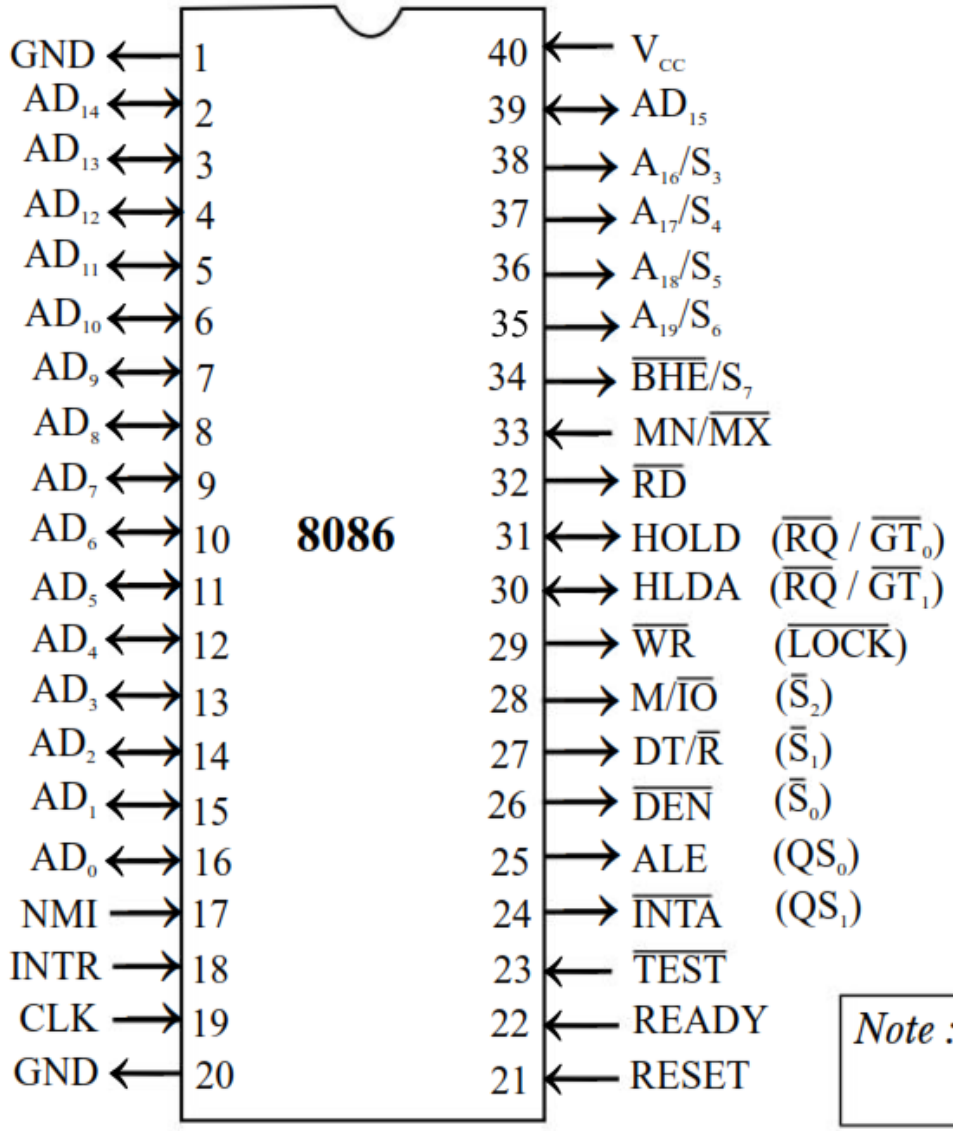
Initially, CS:IP must be loaded with the addresses from which the program will be executed. At first, the queue is empty and the microprocessor starts a fetch operation and the first byte, i.e., opcode of the instruction is loaded into the queue. Subsequently, data will also be fetched in the queue. When the first byte of the queue, i.e., opcode of instruction, is loaded into decoder for decoding, one byte becomes empty from the queue. Therefore, the queue must be updated with the next instruction opcode and data.

After decoding the opcode, the decoder takes a decision whether the instruction is of one byte or multi-byte. When the instruction is of one byte, the opcode can perform operations during executing the instruction. If the instruction is multi-byte, the first byte is opcode and other remaining bytes are data, or first two bytes are opcodes and other bytes are data. Therefore, the microprocessor should read the operand from queue and decode it. Subsequently, it executes the instruction by accepting data from the queue. After completion of fetching and decoding of an instruction, the next instruction will be fetched and decoded and executed. Figure



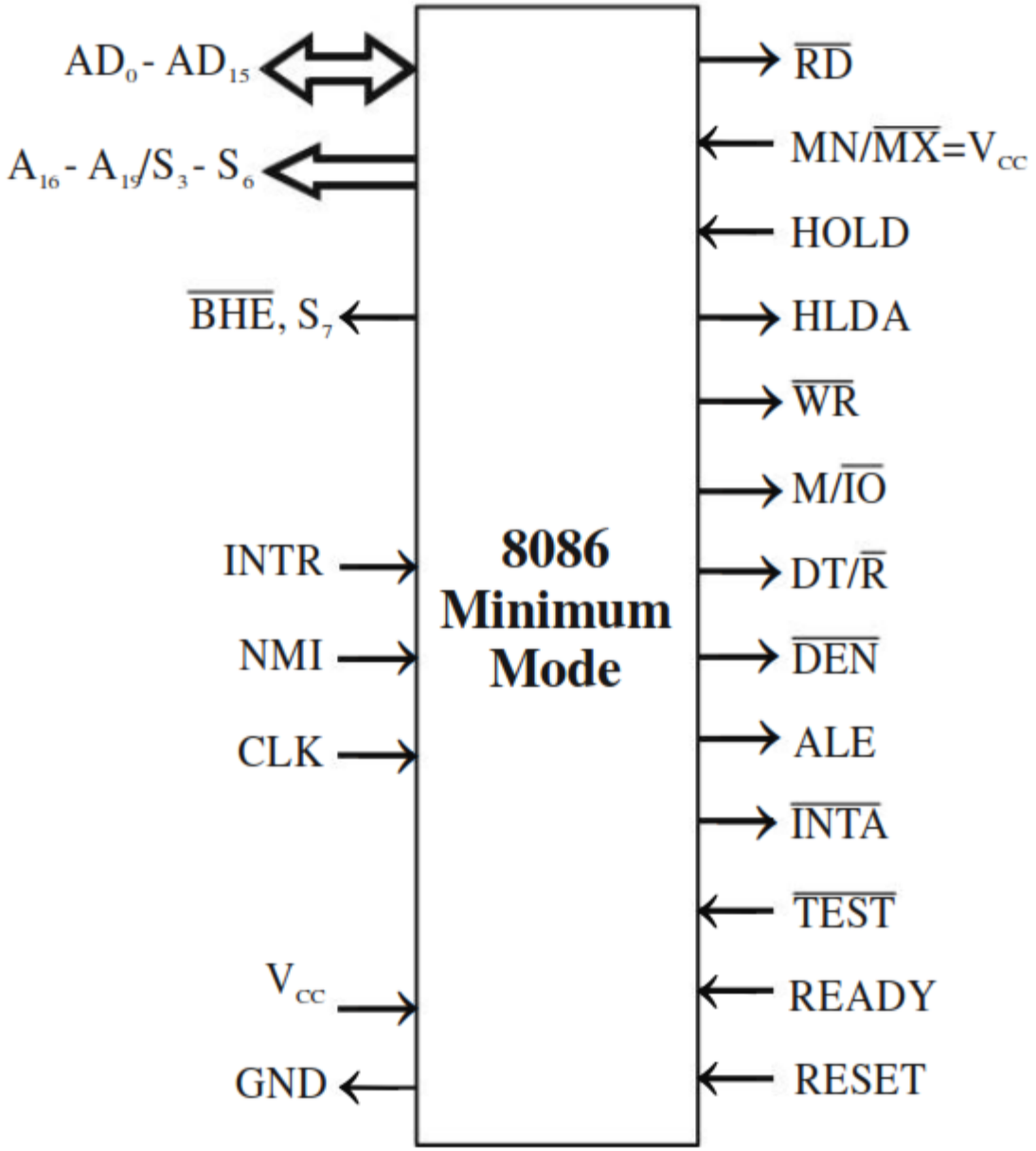
Queue operation of 8086 microprocessor

PINS AND SIGNALS OF INTEL 8086

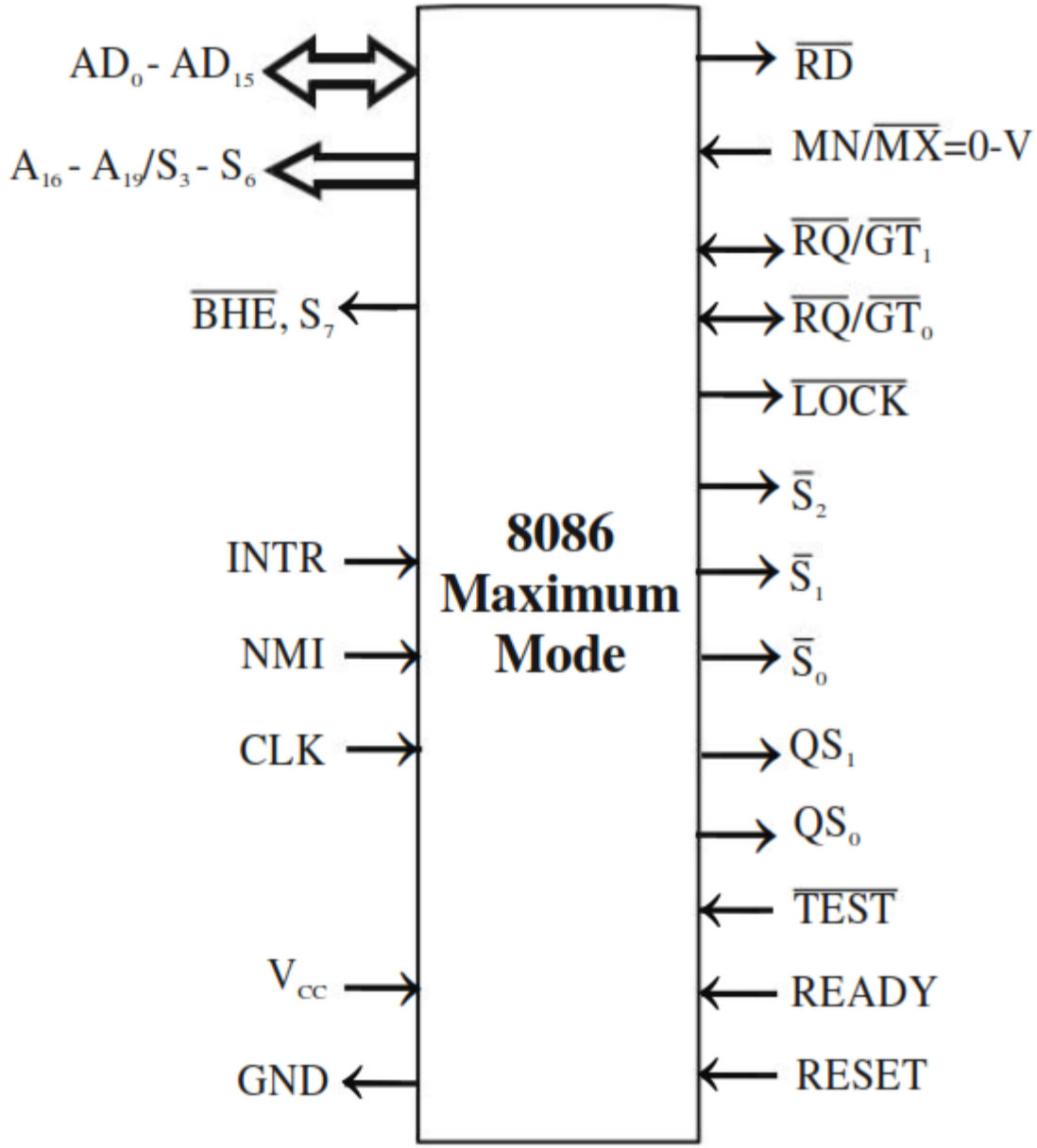


8086 pin assignments.

Note : Signals shown in parenthesis are maximum mode signals.



8086-Minimum mode.



8086-Maximum mode.

COMMON SIGNALS

Name	Description/Function	Type
AD_{15} - AD_0	Address/Data	Bidirectional, Tristate
A_{19}/S_6 - A_{16}/S_3	Address/Status	Output, Tristate
$\overline{BHE}/S_7$	Bus high enable/Status	Output, Tristate
MN/$\overline{MX}$	Minimum/Maximum mode control	Input
$\overline{RD}$	Read control	Output, Tristate
$\overline{TEST}$	Wait on test control	Input
READY	Wait state control	Input
RESET	System reset	Input
NMI	Non-maskable interrupt request	Input
INTR	Interrupt request	Input
CLK	System clock	Input
V_{cc}	+5-V	Power supply input
GND	Ground	Power supply ground

MINIMUM MODE SIGNALS [MN/$\overline{MX}$ = V_{CC} (Logic high)]

Name	Description/Function	Type
HOLD	Hold request	Input
HLDA	Hold acknowledge	Output
$\overline{WR}$	Write control	Output, Tristate
M/$\overline{IO}$	Memory / IO control	Output, Tristate
DT/$\overline{R}$	Data transmit / Receive	Output, Tristate
$\overline{DEN}$	Data enable	Output, Tristate
ALE	Address latch enable	Output
$\overline{INTA}$	Interrupt acknowledge	Output

The 8086 can operate in two modes: minimum mode and maximum mode. The mode is decided by a signal at MN/$\overline{MX}$ pin. When the MN/$\overline{MX}$ is tied **high**, it works in minimum mode and the system is called a uniprocessor system. When MN/$\overline{MX}$ is tied **low**, it works in maximum mode and the system is called a multiprocessor system. Usually the pin MN/$\overline{MX}$ is permanently tied to **low** or **high** so that the 8086 system can work in any one of the two modes. The 8086 can work with the 8087 coprocessor in maximum mode. In this mode an external bus controller 8288 is required to generate bus control signals.

For accessing IO-mapped devices, the 8086 uses a separate 16-bit address, and so the 8086 can generate 64 k(2^{16}) IO addresses. The signal M/$\overline{IO}$ is used to differentiate the memory and IO addresses. For memory address, the signal M/$\overline{IO}$ is asserted **high** and for IO address the signal is M/$\overline{IO}$ asserted **low** by the processor.

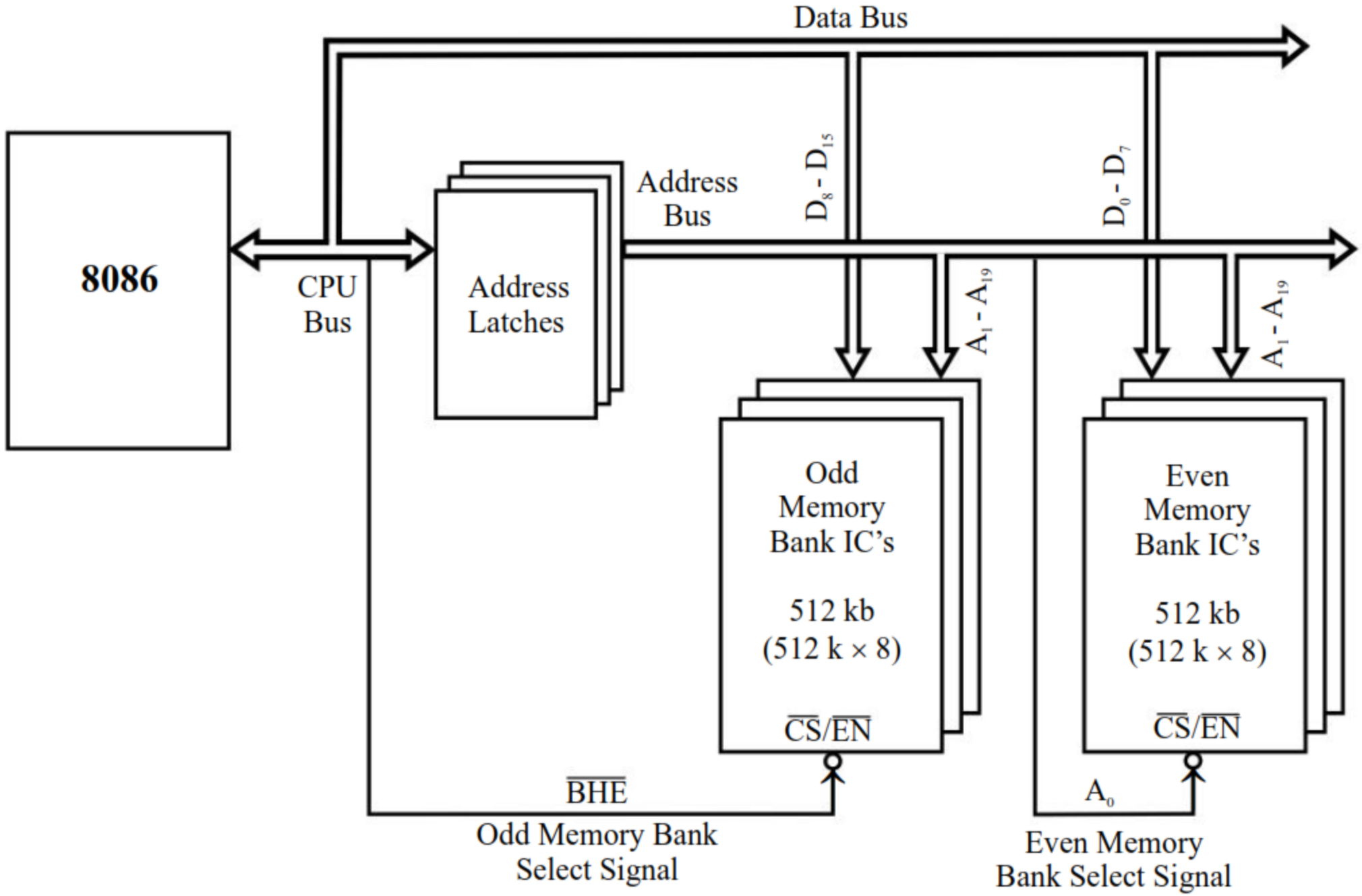
The 8086 has two families of processors. They are 8086 and 8088. The 8088 uses 8-bit data bus externally but the 8086 uses 16-bit data bus externally. The 8086 accesses memory in words but the 8088 accesses memory in bytes. IBM designed its first **P**ersonal **C**omputer (PC) using an INTEL 8088 microprocessor as the CPU.

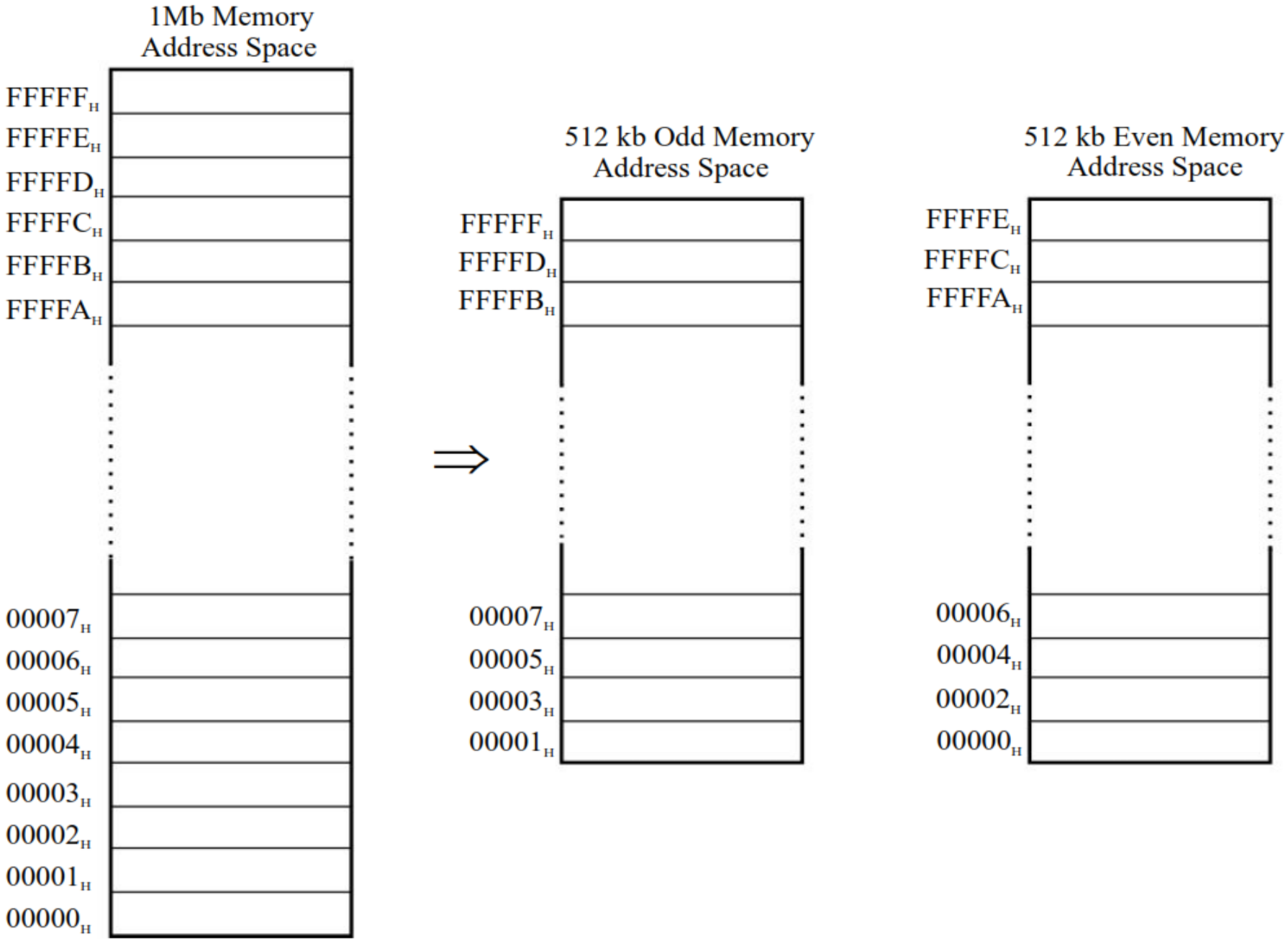
The 8086 uses a 20-bit address to access memory and hence it can directly address up to one mega-byte (2^{20} = 1 Mega) of memory space. One mega-byte (1 Mb) of addressable memory space of 8086 are organized as two memory banks of 512 kilo bytes each (512 kb + 512 kb = 1 Mb). The memory banks are called even (or lower) bank and odd (or upper) bank. The address line A_0 is used to select the even bank and the control signal $\overline{BHE}$ is used to select the odd bank.

EVEN AND ODD MEMORY BANKS

The 8086 microprocessor uses a 20-bit address to access memory. With 20-bit address the processor can generate 2^{20} = 1 Mega address. The basic memory word size of the memories used in the 8086 system is 8-bit or 1-byte (i.e., in one memory location an 8-bit binary information can be stored). Hence, the physical memory space of the 8086 is 1Mb (1 Mega-byte).

For the programmer, the 8086 memory address space is a sequence of one mega-byte in which one location stores an 8-bit binary code/data and two consecutive locations store 16-bit binary code/data. But physically (i.e., in the hardware), the 1Mb memory space is divided into two banks of 512kb (512kb + 512kb = 1Mb). The two memory banks are called Even (or Lower) bank and Odd (or Upper) bank. The organization of even and odd memory banks in the 8086-based system is shown in Fig.


Data Bus
8086
CPU Bus
Address Latches
Address Bus
D8 - D15
D0 - D7
A1 - A19
A1 - A19
Odd Memory Bank IC's
512 kb
(512 k × 8)
CS/EN
Even Memory Bank IC's
512 kb
(512 k × 8)
CS/EN
BHE
Odd Memory Bank Select Signal
A0
Even Memory Bank Select Signal


1Mb Memory
Address Space
FFFFF$_H$
FFFFE$_H$
FFFFD$_H$
FFFFC$_H$
FFFFB$_H$
FFFFA$_H$
00007$_H$
00006$_H$
00005$_H$
00004$_H$
00003$_H$
00002$_H$
00001$_H$
00000$_H$
512 kb Odd Memory
Address Space
FFFFF$_H$
FFFFD$_H$
FFFFB$_H$
00007$_H$
00005$_H$
00003$_H$
00001$_H$
512 kb Even Memory
Address Space
FFFFE$_H$
FFFFC$_H$
FFFFA$_H$
00006$_H$
00004$_H$
00002$_H$
00000$_H$

The 8086-based system will have two sets of memory IC's. One set for even bank and another set for odd bank. The data lines D_0-D_7 are connected to even bank and the data lines D_8-D_{15} are connected to odd bank. The even memory bank is selected by the address line A_0 and the odd memory bank is selected by the control signal $\overline{BHE}$. The memory banks are selected when these signals are **low**(active low). Any memory location in the memory bank is selected by the address line A_1 to A_{19}.

The organization of memory into two banks and providing bank select signals allows the programmer to read/write the byte (8-bit) operand in any memory address through 16-bit data bus. It also allows the programmer to read/write the word (16-bit) operand starting from even address or odd address.

The memory access for byte and word operand from the even and odd bank by the 8086 processor will be as follows:

Case i **: Byte access from even bank**

For read/write operation of a byte in even memory address, A_0 is asserted **low** and $\overline{BHE}$ is asserted **high** (i.e., $A_0 = 0$ and $\overline{BHE} = 1$). Now the even bank alone is enabled and the data transfer take place through D_0-D_7 data lines.

Case ii **: Byte access from odd bank**

For read/write operation of a byte in odd memory address, A_0 is asserted **high** and $\overline{BHE}$ is asserted **low** (i.e., $A_0 = 1$ and $\overline{BHE} = 0$). Now odd bank alone is enabled and the data transfer take place through D_8-D_{15} data lines.

Case iii **: Word access**

For read/write operation of a word (16-bit) in even boundary (i.e., low byte in even address and high byte in next address (odd address)), both A_0 and $\overline{BHE}$ are asserted **low** (i.e., $A_0 = 0$ and $\overline{BHE} = 0$). Now both the memory banks are enabled simultaneously and the processor read/write the 16-bit operand in one bus cycle through D_0-D_{15} data lines.

AD_{15}–AD_0 (Bi-directional) Address/Data Bus

These lines constitute the time-multiplexed address/data bus. These lines are low-order address bus. They act as an address bus during the first clock cycle multiplexed. When AD lines are used to transmit memory/IO address, the symbol A is used of AD. For example, A represents A_{15}–A_0. When data are transmitted through AD lines, the symbol D is used in place of AD. For example, D represents D_7_D_0, D_{15}–D_8 or D_{15}–D_0.

A_{19}–A_{16} (Output)

These are high-order address lines and they are time-multiplexed lines. During T_1, these lines can be used as higher order 4 bits of memory address. But in I/O operation, these lines are low. During T_2, T_3, and T_4, they carry status signals.

A_{16}/S_3, A_{17}/S_4 (Output)

A_{16} and $A_{17}$are time multiplexed with segment identifier signals S_3 and S_4. During T_1 clock cycle, A_{16} and A_{17} are used as address bits. In T_2 to T_4 clock cycles, these lines carry status signals. Table 5.5 shows memory segment identification.

Memory segment identification

S_4	S_3	Function
0	0	Extra segment memory access
0	1	Stack segment memory access
1	0	Code segment memory access
1	1	Data segment memory access

A_{18}/S_5 (Output)

A_{18} is time multiplexed with interrupt status S_5. During T_1 clock cycle, A_{18} is transmitted to the address bus. During other clock cycles (T_2, T_3 and T_4), the status signal S_5 is transmitted through this line. S_5 is an interrupt-enable status signal. At the beginning of each clock cycle, the status of the interrupt enable flag S_5 is updated.

A_{19}/S_6 (Output)

A_{19} is multiplexed with the status signal S_6. During T_1 clock cycle, A_{19} is transmitted to an address bus. During T_2 to T_4, the status signal S_6 is available on this line. It is low during T_2 to T_4.

$\overline{BHE}/S_7$ (Output) Bus High Enable/Status

During T_1, the bus high enable signal $\overline{BHE}$ can be used to enable data onto the most significant half of the data bus D_{15}–D_8. An 8-bit device connected to the upper half of the data bus uses a $\overline{BHE}$ signal to condition chip select functions. $\overline{BHE}$ is low during T_1 for read, write and interrupt acknowledge cycles when a byte is to be transferred on the high portion of the bus. This pin is multiplexed with the status signal S_7. The S_7 status signal is available during T_2 to T_4. The signal is active low, and floats to 3-state OFF in hold. It is low during T_1 for the first interrupt acknowledge cycle. Table 5.6 shows the function of $\overline{BHE}$ and A_0

Selection of proper byte from even-addressed and odd-addressed memory banks for processing

$\overline{BHE}$	A_0	*Processing*
0	0	Both banks active. 16-bit data transfer, 16-bit word transfer on AD_{15}–AD_0
0	1	Only high bank active, one byte transfer on AD_{15}–AD_8
1	0	Only low bank active, one byte transfer on AD_7–AD_0
1	1	No bank active

A **low** on the line S_6 indicates that the 8086 is on the bus (i.e., it indicates that 8086 is the bus master) and during hold acknowledge, this pin is driven to **high impedance** state.

$\overline{RD}$ (Output) (Read) This control signal is used for read operation. It is an output signal. It is active when LOW. The Read signal indicates that the processor is performing a memory or I/O read cycle,

READY (Input) The addressed I/O or memory devices send acknowledgment through this pin and it indicates that the data transfer is completed.

INTR (Interrupt Request) It is a level-triggered input which is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt vector-look up table located in the system memory. It can be internally masked by software, resetting the interrupt enable bit. INTR is internally synchronized. This signal is active HIGH.

$\overline{TEST}$ (Input) This is used in conjunction with the WAIT instruction. If the $\overline{TEST}$ input is LOW, execution continues. Otherwise the processor waits in an idle state. This input is synchronized internally during each clock cycle on the leading edge of CLK. When it is low, the microprocessor continues execution otherwise it waits.

NMI (Input) Nonmaskable Interrupt This is an edge-triggered input which causes a type 2 interrupt. A subroutine is vectored to via an interrupt vector look-up table located in system memory. NMI is not maskable internally by software. A transition from LOW TO HIGH initiates the interrupt at the end of the current instruction. This input is internally synchronized.

MN/$\overline{MX}$ (Input) The minimum/maximum signal indicates the operating mode of 8086. When it is high, the 8086 processor operates in minimum mode. If this pin is low, the processor operates in maximum mode.

RESET (Input) The reset signal is active HIGH. The processor immediately terminates its present activity and system is reset. The signal must be active HIGH for at least four clock cycles. It restarts execution, as described in the instruction set, when RESET returns low. RESET is internally synchronised.

V_{CC} Power supply, + 5 V dc

GND Ground

MAXIMUM MODE SIGNALS [MN/$\overline{MX}$ = Ground(Logic low)]

Name	Description/Function	Type
$\overline{RQ}/\overline{GT}_1$, $\overline{RQ}/\overline{GT}_0$	Request/Grant bus access control	Bidirectional
$\overline{LOCK}$	Bus priority lock control	Output, Tristate
$\overline{S}_2$, $\overline{S}_1$, $\overline{S}_0$	Bus cycle status	Output, Tristate
QS_1, QS_0	Instruction queue status	Output

$\bar{S}_0, \bar{S}_1, \bar{S}_2$ - These are status signals and they are used by the 8288 bus controller to generate the bus timing and control signals.

STATUS SIGNALS DURING VARIOUS MACHINE CYCLES

Status signal			Machine cycle
$\bar{S}_2$	$\bar{S}_1$	$\bar{S}_0$	
0	0	0	Interrupt acknowledge
0	0	1	Read IO port
0	1	0	Write IO port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive/Inactive

$\overline{RQ}/\overline{GT}_0$, $\overline{RQ}/\overline{GT}_1$ - (Bus Request/Bus Grant) These requests are used by the other local bus masters to force the processor to release the local bus at the end of the processor's current bus cycle. These pins are bidirectional. The request on GT_0 will have higher priority than GT_1.

The bus request to 8086 work as follows:

1. When a local bus master requires system bus control, it sends a low pulse to the 8086.
2. At the end of the current bus cycle, the processor (8086) drives its pins to high impedance state and sends an acknowledge as a low pulse on the same pin to the device which requested the bus control.
3. On receiving the acknowledge the local master will take control of the system bus. After completing its work, at the end, the local bus master sends a low signal on the same pin to the 8086 to inform the end of control. Now 8086 regains the control of the bus.

$\overline{\text{LOCK}}$ - It is an output signal, activated by the LOCK prefix instruction and remains active until the completion of the instruction prefixed by LOCK. The 8086 outputs **low** on the $\overline{\text{LOCK}}$ pin while executing an instruction prefixed by LOCK to prevent other bus masters from gaining control of the system bus.

QS_1, QS_0 - (**Q**ueue **S**tatus) - The processor provides the status of queue on these lines. The queue status can be used by the external device to track the internal status of the queue in the 8086. The QS_0 and QS_1 are valid during the clock period following any queue operation. The output on QS_0 and QS_1 can be interpreted as shown in Table-2.6.

QUEUE STATUS

Queue status		Queue operation
QS_1	QS_0	
0	0	No operation
0	1	First byte of an opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

BUS CYCLES AND TIMING DIAGRAM

The 8086 processor has two functional units called **B**us **I**nterface **U**nit (BIU) and **E**xecution **U**nit (EU). Most of the time each unit works independently. The BIU takes care of fetching instruction codes from memory, and data from memory and IO devices. The EU takes care of executing the instructions prefetched by BIU.

The BIU initiates all external operations which are also called bus activity. The external bus activities are repetitions of certain basic operations. The basic operations performed by the CPU bus are called bus cycles. Depending on the activities of 8086, the bus cycles can be classified as follows:

1) Memory read cycle (Four T states)
2) Memory write cycle (Four T states)
3) IO read cycle (Four T states)
4) IO write cycle (Four T states)
5) Interrupt acknowledge cycle (Eight T states)

The processor takes a definite time to perform a bus cycle. The time taken to perform a bus cycle are specified in terms of T states. In an 8086 processor the time duration of one T-state is equal to one time period of the internal clock of the processor. The T-state starts in the middle of falling edge of the clock signal as shown in Fig. 2.5.

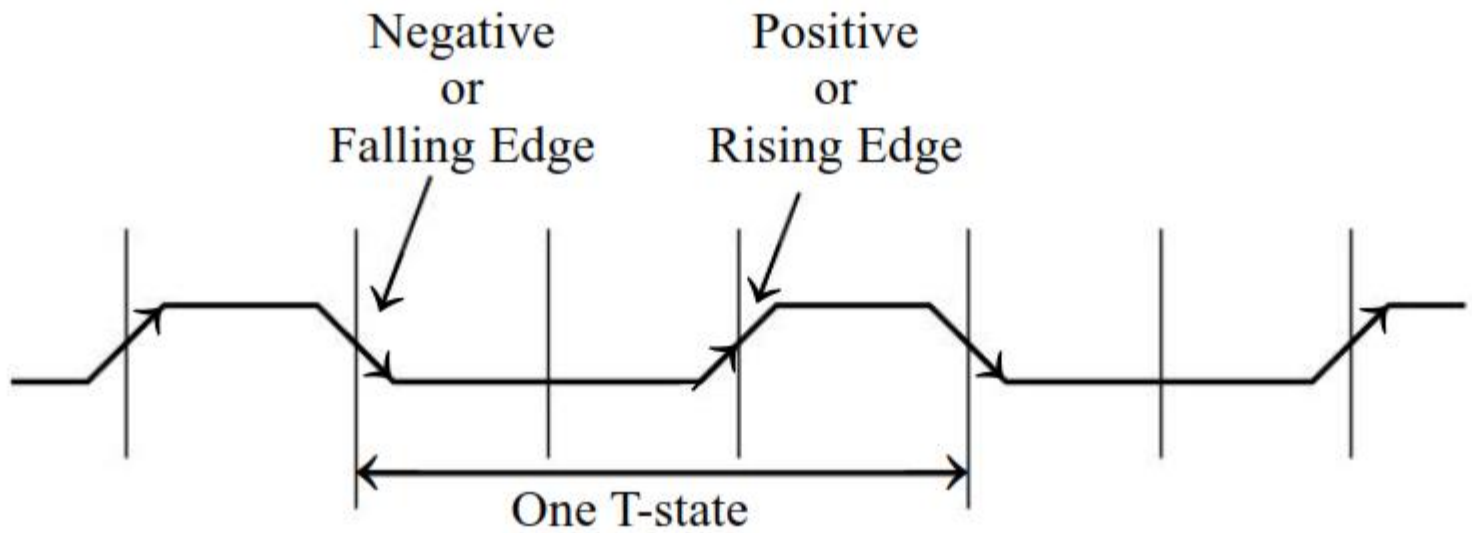


Fig. 2.5 : Clock signal and one T-state of 8086.

The normal time taken by 8086 to perform read/write cycle is four T states. The processor also has facility to extend the timing of bus cycles by introducing extra T states called wait states using READY control signal. If READY is permanently tied **high** then the bus cycles are executed in normal timing.

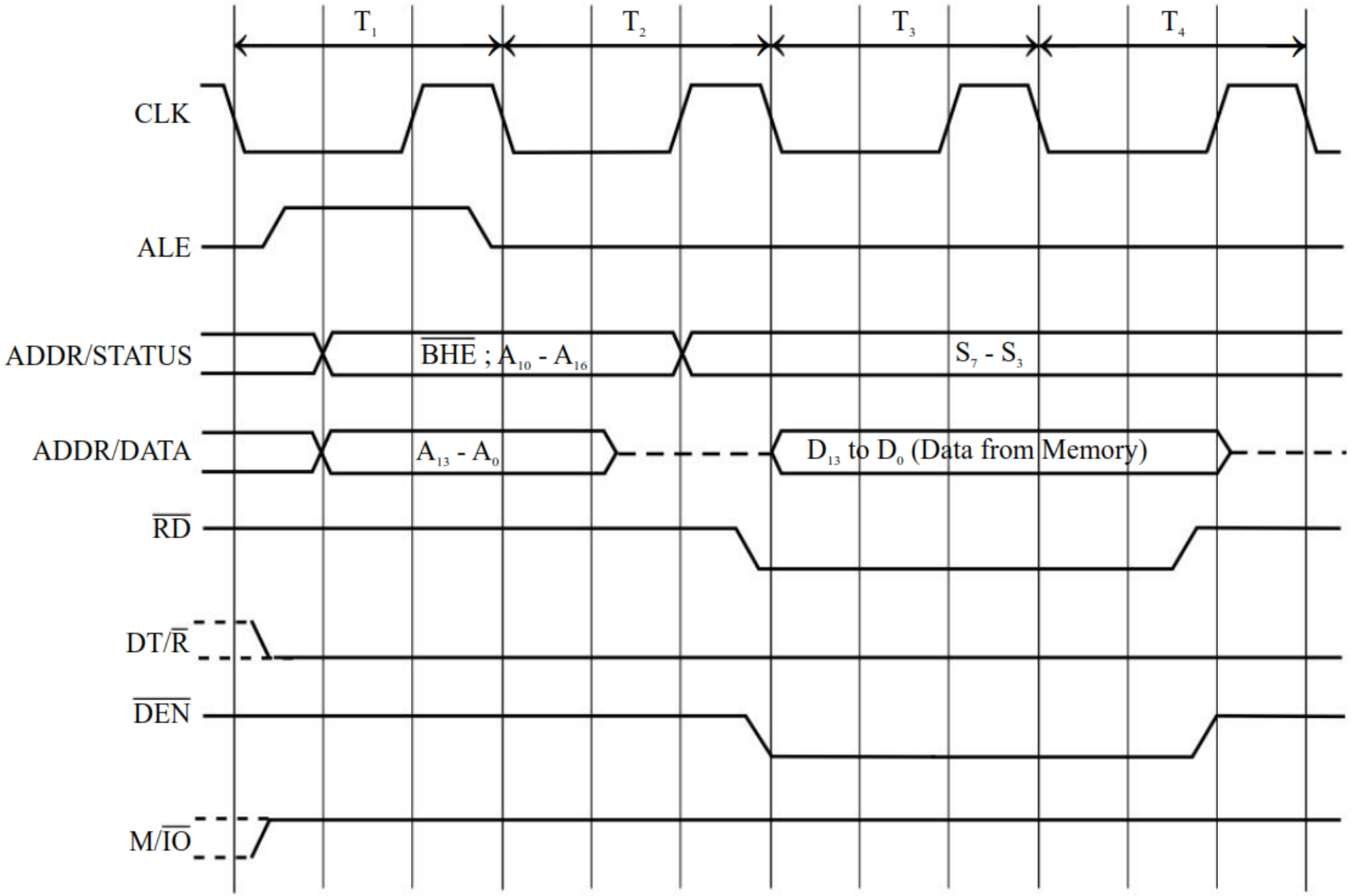
The memory or IO access time allowed by the 8086 processor with 5 MHz clock is 400 ns. If the memory or IO devices used in the system has access time more than 400 ns then wait states have to be introduced in the bus cycles, between the second and third T states (T_2 and T_3) to extend the timing of the bus cycle.

In order to introduce wait states the READY is made **low** by an external hardware in the beginning of second T-state and then made **high** after required time delay. The processor samples READY signal at the end of second T-state of every bus cycle. If READY is **low** at this time then the processor introduces one wait state. Again it samples READY signal at the end of wait state and if READY is still **low** then it introduces another wait state. This process is continued until READY is **high**. Once READY is made **high** the processor will resume the bus activity and completes the bus cycle.

Timing Diagram

The timing diagram provides information about the various conditions (**high** state or **low** state or **high impedance** state) of the signals while a bus cycle is executed. The timing diagrams are supplied by the manufacturer of the microprocessor. The timing diagrams are essential for a system designer. Only from the knowledge of timing diagrams, the matched peripheral devices like memories, ports, etc., can be selected to form a system with a microprocessor as the CPU.

Memory Read Cycle



($\overline{WR}$ is **high** ; READY is tied **high** permanently or temporarily in the system.)

Activities during T_1 :

i) The 8086 outputs a 20-bit memory address on AD_0-AD_{15} lines and ADDR/STATUS lines.

ii) The address latch enable signal ALE is asserted **high** in the beginning of T_1 and then asserted **low** at the end of T_1. This enables the external address latches to latch the address (at the falling edge of ALE) and keep on their output lines.

iii) The direction control signal DT/$\overline{R}$ is asserted **low** to inform the external bidirectional data buffer that the processor has to receive data. (If DT/$\overline{R}$ is already **low** in the previous bus cycle then it remains **low** as such.)

iv) The M/$\overline{IO}$ signal is asserted **high** to indicate memory access. (If M/$\overline{IO}$ is already **high** then it remains **high** as such.)

v) The $\overline{BHE}$ is asserted **low** to enable the odd/upper memory bank.

Activities during T_2 :

i) The AD_0-AD_{15} lines becomes inactive.

ii) The address is withdrawn from the ADDR/STATUS lines and status signals S_7-S_3 are issued on these lines. (The $\overline{BHE}$ becomes the status signal S_7.)

iii) At the end of T_2 the read control signal $\overline{RD}$ is asserted **low** to enable the output buffer of memory. The time during which $\overline{RD}$ remains **low** is the time allowed for memory to load data in the data bus.

iv) The $\overline{DEN}$ signal is asserted **low** to enable the external bidirectional data buffers.

v) The 8086 samples READY signal during T_2. (If READY is **high** then T_3 and T_4 are executed otherwise wait states are introduced.)

Activities during T_3 :

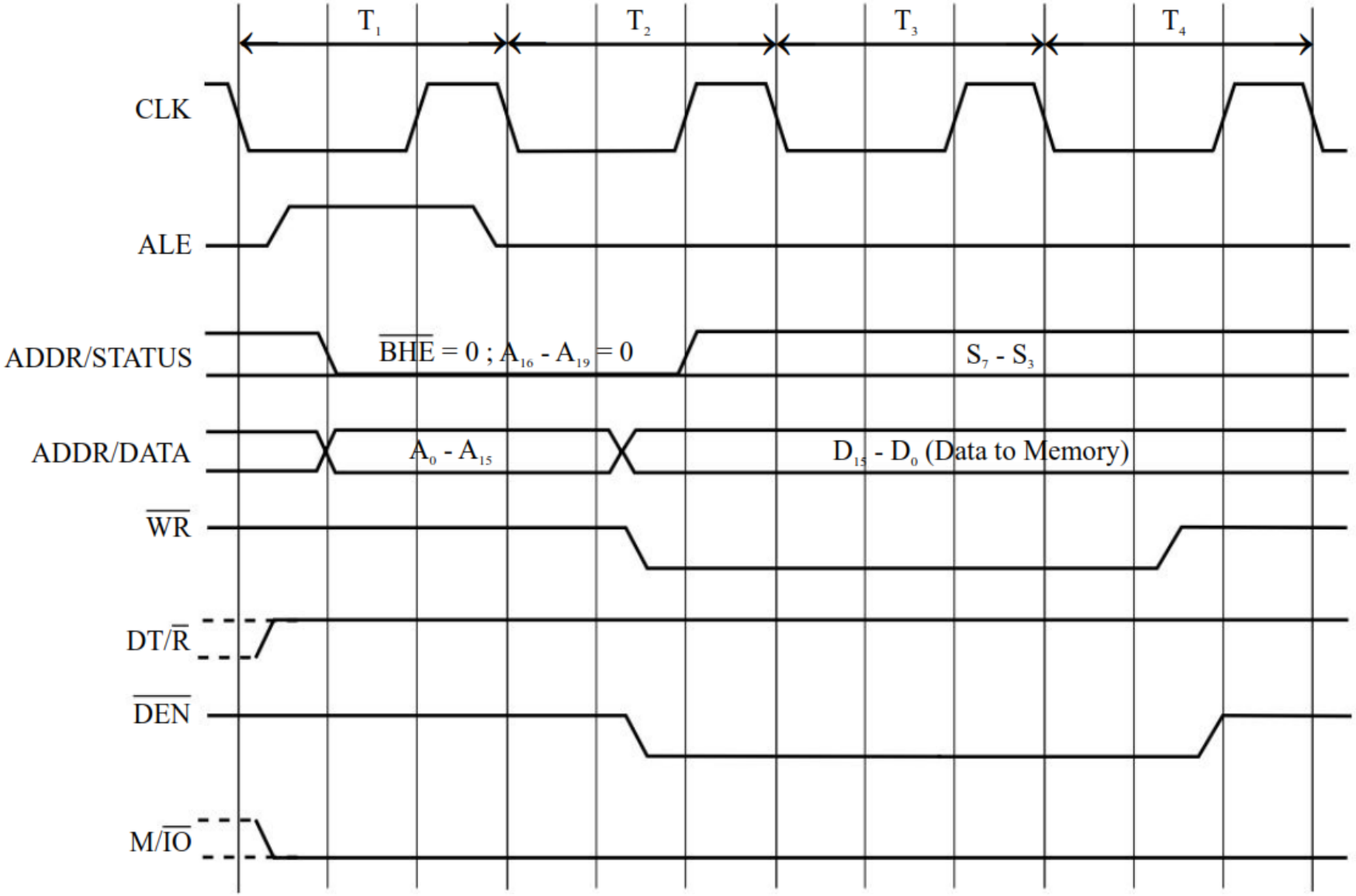
No activities are performed during T_3. The status of the signals at the end of T_2 are maintained throughout T_3.

Activities during T_4 :

i) The $\overline{RD}$ is asserted **high** and at this time (i.e., at the rising edge of $\overline{RD}$) the data is latched into 8086.

ii) The $\overline{DEN}$ is made **high** to disable the data buffer.

Memory Write Cycle



($\overline{RD}$ is **high** ; READY is tied **high** permanently or temporarily in the system.)

Activities during T_1 :

The activities during T_1 is same as that of read cycle except DT/$\bar{R}$ signal. In memory write cycle, the DT/$\bar{R}$ signal is asserted **high** to inform the external bidirectional data buffer that the processor is going to transmit data. (If DT/$\bar{R}$ is already **high** in the previous bus cycle then it remains, **high** as such.)

Activities during T_2 :

i) The address is withdrawn from AD_0-AD_{15} lines and data is output on these lines.
ii) The address is withdrawn from ADDR/STATUS lines and status signals are issued on these lines (The $\overline{BHE}$ becomes the status signal S_7.)
iii) When data is output on data bus, the control signals $\overline{WR}$ and $\overline{DEN}$ are also asserted **low** to enable the input buffer of memory and external data buffer on the data bus respectively.
iv) The 8086 samples READY signal during T_2. (If READY is **high** then T_3 and T_4 are executed otherwise wait states are introduced.)

Activities during T_3 :

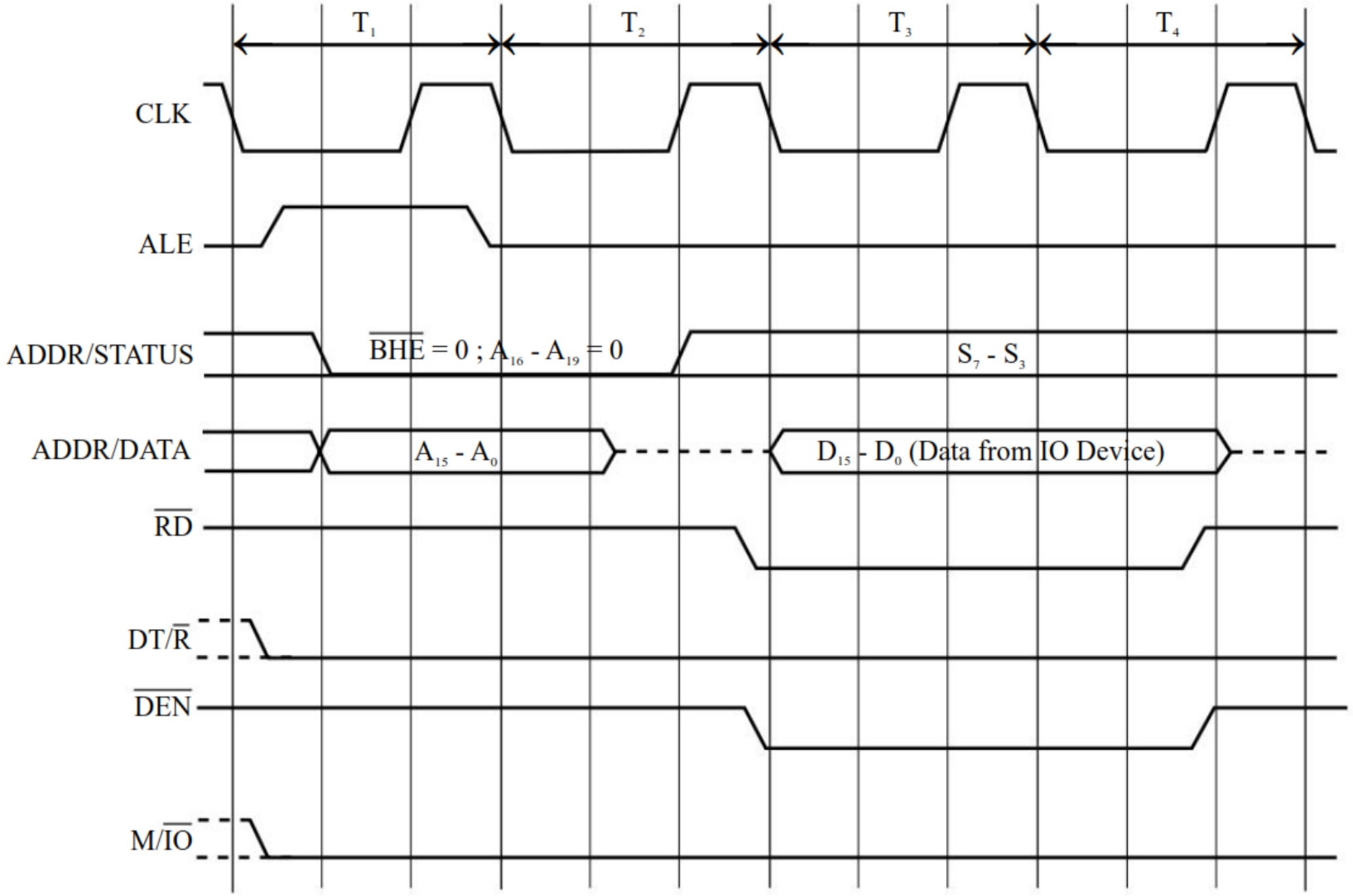
No activities are performed during T_3.The status of the signals at the end of T_2 are maintained throughout T_3.

Activities during T_4 :

i) The $\overline{WR}$ is asserted **high** and at this time (i.e., at the rising edge of $\overline{WR}$), the data is latched into memory.

ii) The $\overline{DEN}$ is made **high** to disable the data buffers.

IO Read Cycle



($\overline{WR}$ is **high** ; READY is tied **high** permanently or temporarily in the system.)

Activities during T_1 :

i) The 8086 outputs a 16-bit IO address on AD_0-AD_{15} lines. Logic **low** is output on the $\overline{BHE}$ and ADDR/STATUS lines.

ii) The ALE is asserted **high** and then **low**. This enables the external address latches to latch the address (at the falling edge of ALE) and keep on their output lines.

iii) The DT/$\overline{R}$ signal is asserted **low** to inform the external bidirectional data buffer that the processor has to receive data. (If DT/$\overline{R}$ is already **low** in the previous bus cycle then it remains **low** as such.)

iv) The M/$\overline{IO}$ signal is asserted **low** to indicate IO access. (If M/$\overline{IO}$ is already **low** then it remains **low** as such.)

Activities during T_2 :

i) The AD_0-AD_{15} lines becomes inactive.

ii) The status signals S_7-S_3 are issued on ADDR/STATUS lines.

iii) At the end of T_2 the read control signal $\overline{RD}$ is asserted **low** to enable the IO device for read operation. The time during which $\overline{RD}$ remains **low** is the time allowed for IO device to load data in the data bus.

iv) The $\overline{DEN}$ signal is asserted **low** to enable the external bidirectional data buffers.

v) The 8086 samples READY signal during T_2. (If READY is **high** then T_3 and T_4 are executed otherwise wait states are introduced.)

Activities during T_3 :

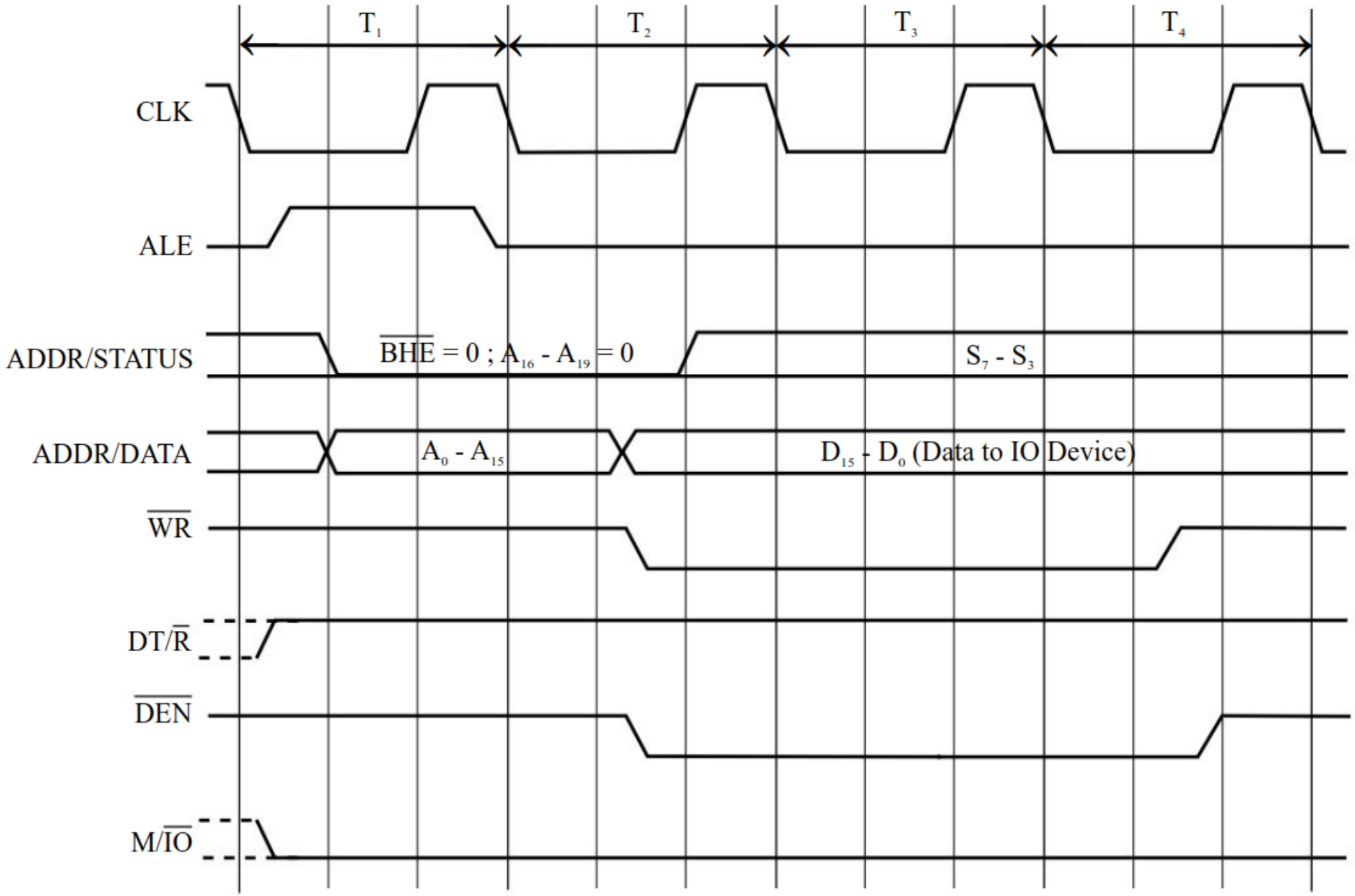
No activities are performed during T_3. The status of the signals at the end of T_2 are maintained throughout T_3.

Activities during T_4 :

i) The $\overline{RD}$ is asserted **high** and at this time (i.e., at the rising edge of $\overline{RD}$), the data is latched into 8086.

ii) The $\overline{DEN}$ is made **high** to disable the data buffer.

IO Write Cycle



($\overline{RD}$ is **high** ; READY is tied **high** permanently or temporarily in the system.)

The activities during IO write cycle will be same as IO read cycle except the following.

i) During T_1, DT/$\overline{R}$ is asserted **high** to inform the external bidirectional data buffer that the processor is going to transmit data.

ii) During T_2, the address is withdrawn from AD_0-AD_{15} lines and data is output on these lines. At the same time $\overline{WR}$ is asserted **low** to enable the IO device for write operation and $\overline{DEN}$ is asserted **low** to enable the data buffer on the bus. Here $\overline{RD}$ remains **high**.

iii) During T_4, $\overline{WR}$ is asserted **high** and at this time (i.e., at the rising edge of $\overline{WR}$) the data is latched into IO device.

What are the modes in which 8086 can operate?

The 8086 can operate in two modes and they are minimum (or uniprocessor) mode and maximum (or multiprocessor) mode.

What is the data and address size in 8086?

The 8086 can operate on either 8-bit or 16-bit data. The 8086 uses 20-bit address to access memory and 16-bit address to access IO devices.

What is the difference between 8086 and 8088.

The external data bus in 8086 is 16-bit and that of 8088 is 8-bit i.e., the 8086 access memory in words but 8088 access memory in bytes.

Explain the function of $M/\overline{IO}$ in 8086.

The signal $M/\overline{IO}$ is used to differentiate memory address and IO address. When the processor is accessing memory locations $M/\overline{IO}$ is asserted **high** and when it is accessing IO-mapped devices it is asserted **low**.

How is a 20-bit physical address computed in 8086?

In 8086, the 20-bit physical address is computed by multiplying a segment base address to 16_{10} and adding to a 16-bit effective address. For program codes, the CS-register will hold the segment base address and IP will hold the effective address. For data the DS/ES/SS-register will hold the segment base address and the method of calculating effective address will be specified in the instruction.

How is a clock signal generated in 8086? What is the maximum internal clock frequency of 8086?

The 8086 does not have on-chip clock generation circuit. Hence, the clock generator chip, 8284 is used to generate the required clock. The frequency of the clock generated by 8284 is thrice that of internal clock frequency of 8086. The 8284 divides the generated clock by three and modify the duty cycle to 33% and then supply as clock signal to 8086. The maximum internal clock frequency of 8086 is 5 MHz.

What is an ALE?

ALE (**A**ddress **L**atch **E**nable) is a signal used to demultiplex the address and data lines using an external latch. It is used as enable signal for the external latch.

How is the READY signal used in a microprocessor system?

The READY is an input signal that can be used by slow peripherals to get extra time in order to communicate with 8086. The 8086 will work only when READY is tied to logic **high**. Whenever READY is tied to logic **low**, the 8086 will enter a wait state. When the system has slow peripheral devices, additional hardware is provided in the system to make the READY input **low** during the required extra time while executing a bus cycle, so that the processor will remain in wait state during this extra time.

What is HOLD and HLDA? How is it used?

The HOLD and HLDA signals are used for the **D**irect **M**emory **A**ccess (DMA) type of data transfer. This type of data transfers are achieved by employing a DMA controller in the system. When DMA is required the DMA controller will place a **high** signal on the HOLD pin of 8086. When HOLD input is asserted **high,** the processor will enter a wait state and drive all its tristate pins to **high impedance** state and send an acknowledgement signal to DMA controller through HLDA pin. Upon receiving the acknowledgement signal, the DMA controller will take control of the bus and perform DMA transfer and at the end it asserts HOLD signal **low**. When HOLD is asserted **low,** the processor will resume its execution.

What is pipelined architecture?

In pipelined architecture, the processor will have a number of functional units and the execution time of the functional units are overlapped. Each functional unit works independently most of the time.

What are the functional units available in 8086 architecture?

The **B**us **I**nterface **U**nit (BIU) and **E**xecution **U**nit (EU) are the two functional units available in 8086 architecture.

List the segment registers of 8086.

The segment registers of 8086 are **C**ode **S**egment (CS), **D**ata **S**egment (DS), **S**tack **S**egment (SS) and **E**xtra **S**egment (ES) registers.

What is the difference between segment register and general purpose register?

Segment registers are used to store 16-bit segment base address of the four memory segments. General purpose registers are used as the source or destination register during data transfer and computation, as pointers to memory and as counters.

What is a queue? How is queue implemented in 8086?

A data structure which can be accessed on the basis of first-in-first-out is called queue. The 8086 has six numbers of 8-bit FIFO registers which are used as instruction queue.

What is a flag?

Flag is a flip-flop used to store the information about the status of the processor and the status of the instruction executed most recently.

Write the flags of 8086.

The 8086 has nine flags and they are:

1. Carry Flag (CF)
2. Parity Flag (PF)
3. Auxiliary carry Flag (AF)
4. Zero Flag (ZF)
5. Sign Flag (SF)
6. Overflow Flag (OF)
7. Trace Flag (TF) (or Single step trap)
8. Interrupt Flag (IF)
9. Direction Flag (DF)

What are control bits?

The flags TF, IF and DF of 8086 are used to control the processor operation and so they are called control bits.

Write the special functions carried by the general purpose registers of 8086.

The special functions carried by the registers of 8086 are the following :

Register	Name of the register	Special function
AX	16-bit Accumulator	Stores the 16-bit result of certain arithmetic and logical operations.
AL	8-bit Accumulator	Stores the 8-bit result of certain arithmetic and logical operations.
BX	Base register	Used to hold the base value in base addressing mode to access memory data.
CX	Count register	Used to hold the count value in SHIFT, ROTATE and LOOP instructions.
DX	Data register	Used to hold data for multiplication and division operations.
SP	Stack pointer	Used to hold the offset address of top of stack memory.
BP	Base pointer	Used to hold the base value in base addressing using stack segment register to access data from stack memory.
SI	Source Index	Used to hold the index value of source operand (data) for string instructions.
DI	Destination Index	Used to hold the index value of destination operand (data) for string instructions.

How is memory organized in 8086?

The 1MB physical memory space of 8086 is organized as two banks of 512 kb each. The two banks are known as odd (or upper) bank and even (or lower) bank. The odd bank is enabled using $\overline{BHE}$ and the even bank is enabled using the address line A_o.

What is a bus cycle?

A bus cycle is the basic external operation performed by the processor. It is also known as processor cycle or machine cycle. To execute an instruction, the processor will run one or more bus cycles in a particular order.

List the bus cycles of 8086?

The various bus cycles of 8086 are:

(i) Memory read cycle
(ii) Memory write cycle
(iii) IO read cycle
(iv) IO write cycle
(v) Interrupt acknowledge cycle

What is T-state?

T-state is the time period of the internal clock signal of the processor. The time taken by the processor to execute a machine cycle is expressed in T-state.

What is the need for timing diagram?

The timing diagram provides information regarding the status of various signals, when a bus cycle is executed. The knowledge of timing diagram is essential for system designer to select matched peripheral devices like memories, latches, ports, etc., to form a microprocessor system.

What operation is performed during the first T-state of every bus cycle in 8086?

In 8086, during the first T-state of every bus cycle the address is latched into the external latches using ALE signal.

When does the 8086 processor check for an interrupt?

The 8086 checks for an interrupt in the last T-state of the last bus cycle of an instruction. (i.e., at the end of an instruction execution).