Presentation

of

Microprocessors and Microcontrollers

Unit-1:Architecture of 8086, Register Organization, Timing Diagrams, Memory Segmentation & Interrupt Vector table

By

K Venkata Siva Reddy

Assistant Professor

Department of ECE

Ravindra College of Engineering for Women

Kurnool – 518452, Andhra Pradesh, India

RCEW, Pasupula (V), Nandikotkur Road,
Near Venkayapalli, KURNOOL

Features

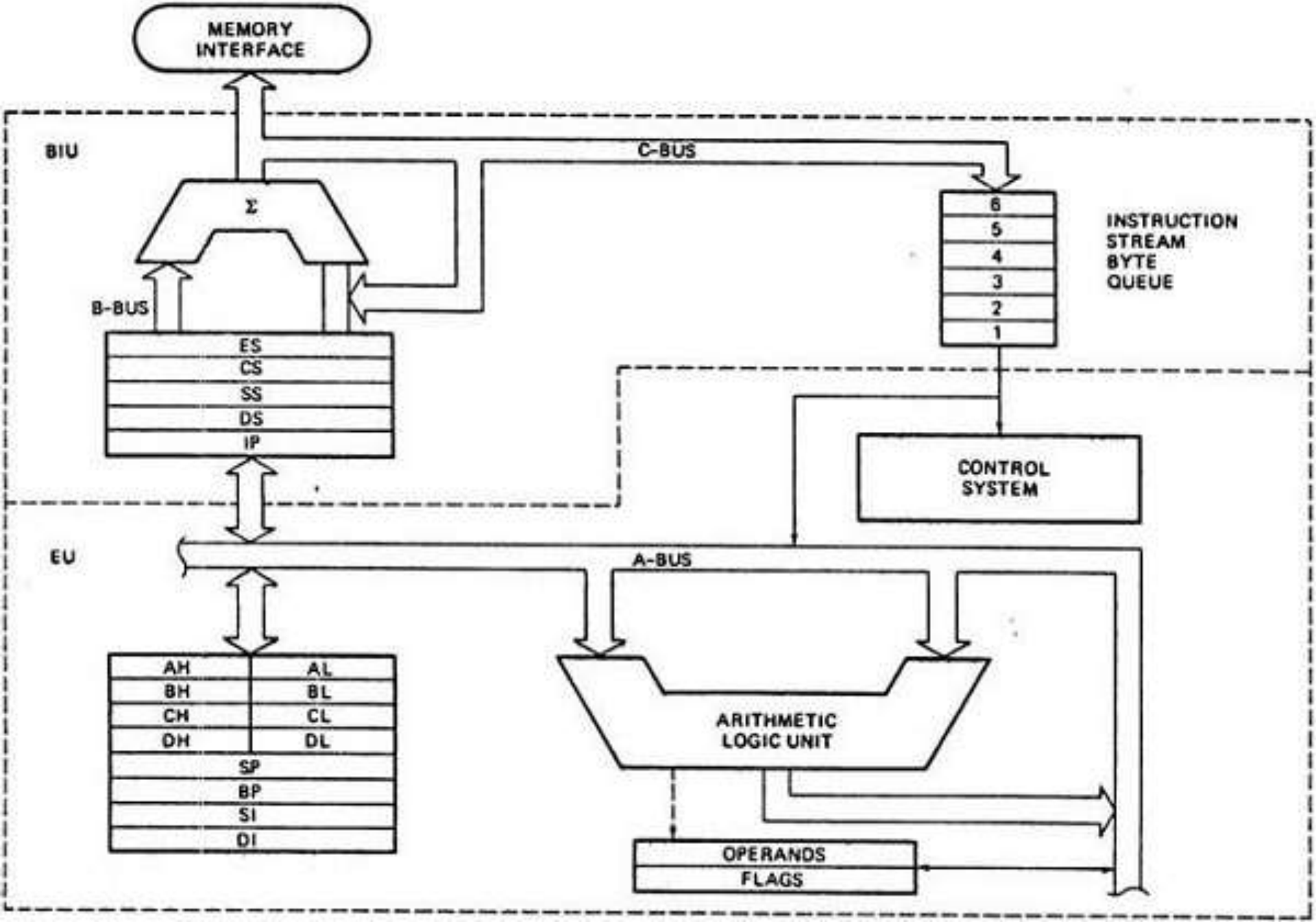
- *It is a 16-bit µp.*
- *8086 has a 20 bit address bus can access up to 2^{20} memory locations (1 MB).*
- *It can support up to 64K I/O ports.*
- *It provides 14, 16 -bit registers.*
- *Word size is 16 bits.*
- *It has multiplexed address and data bus AD0-AD15 and A16 – A19.*
- *It requires single phase clock with 33% duty cycle to provide internal timing.*

- *8086 is designed to operate in two modes, Minimum and Maximum.*
- *It can prefetches 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.*
- *Address ranges from 00000H to FFFFFH*
- *Memory is byte addressable - Every byte has a separate address.*

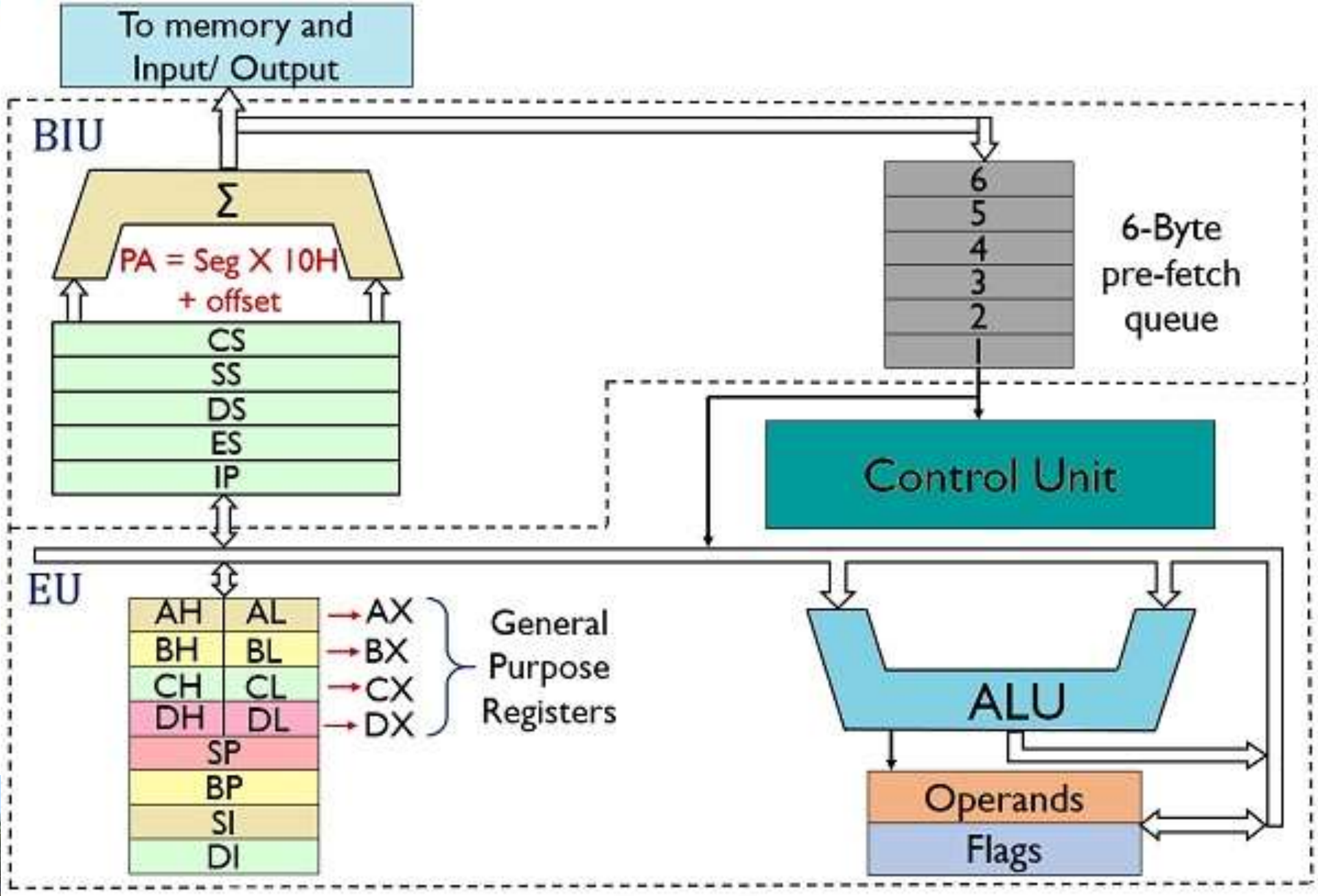
FEATURES OF 8086:

- It is introduced in 1978.
- It is first 16-bit μP.
- It is a 40 pin DIP IC
- It requires +5v Power supply
- Its clock frequency is 5MHz.
- It has 16 Data lines and 20 Address lines.
- It has multiplexed address and data bus i.e. AD0-AD15.
- It has 29,000 transistors.
- It can access 1MB of memory.
- It has 22,000 instructions.
- It has Multiply and Divide instructions.
- It has 14, 16-bit registers
- It has 16-bit ALU
- It has two stage of Pipeline i.e. Fetch & Execute.
- It operates in two modes minimum (single processor) and maximum (multi-processor).
- It has 6 byte instruction Queue.

Intel 8086 Internal Architecture



Intel 8086 Internal Architecture



Internal architecture of 8086

- 8086 has two blocks BIU and EU.
- The BIU handles all transactions of data and addresses on the buses for EU.
- 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. This results in efficient use of the system bus and system performance.

- BIU contains Instruction queue, Segment registers, Instruction pointer, Address adder.

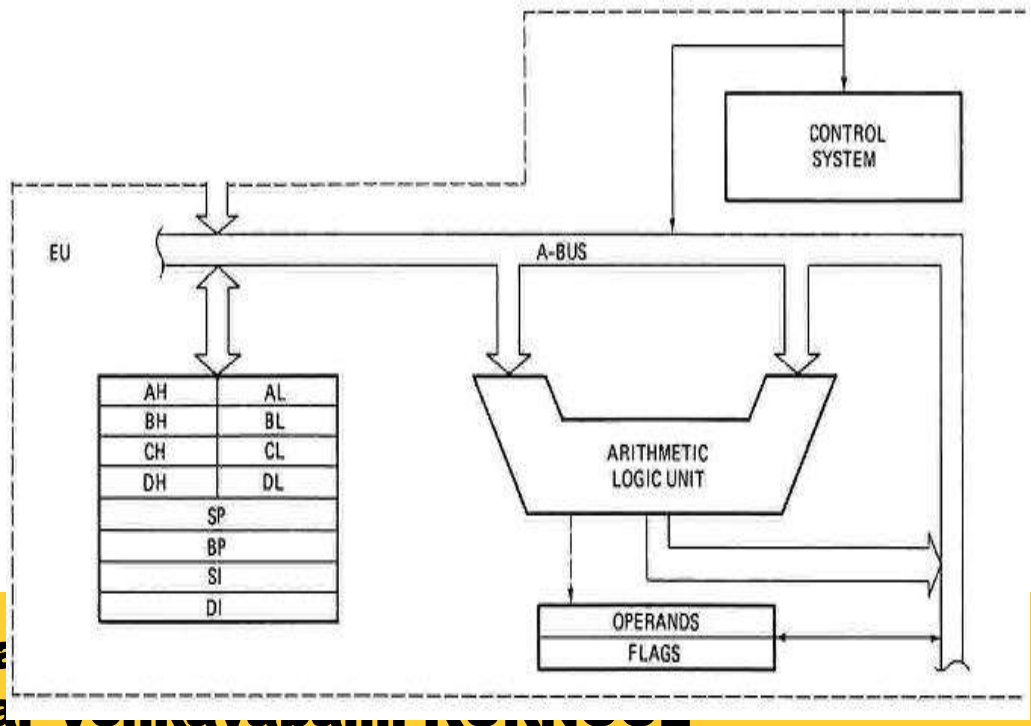
- EU contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register.

EXECUTION UNIT

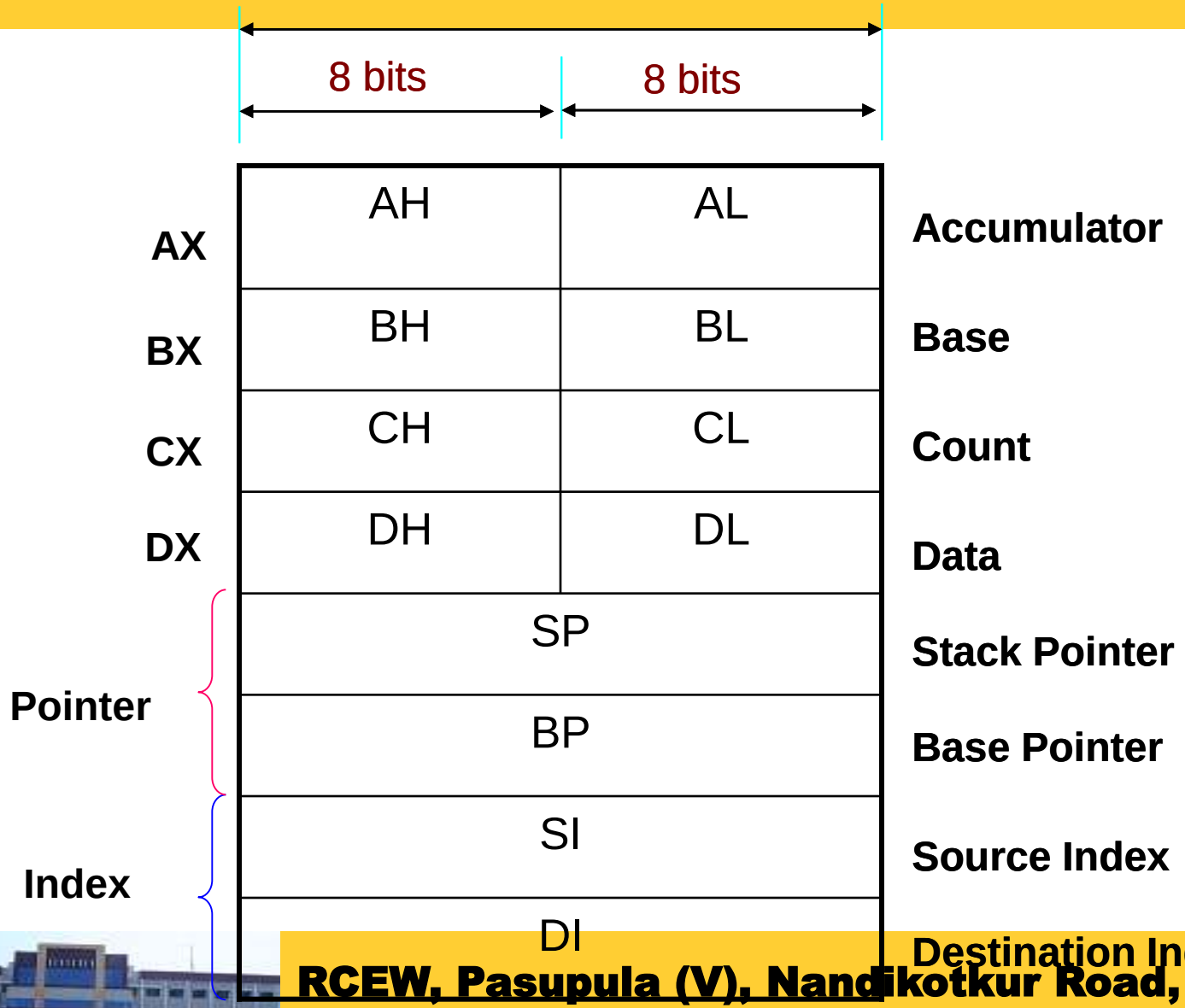
- Decodes instructions fetched by the BIU
- Generate control signals,
- Executes instructions.

The main parts are:

- Control Circuitry
- Instruction decoder
- ALU



EXECUTION UNIT – General Purpose Registers



EXECUTION UNIT – General Purpose Registers

Register	Purpose
AX	Word multiply, word divide, word I /O
AL	Byte multiply, byte divide, byte I/O, decimal arithmetic
AH	Byte multiply, byte divide
BX	Store address information
CX	String operation, loops
CL	Variable shift and rotate
DX	Word multiply, word divide, indirect I/O (Used to hold I/O address during I/O instructions. If the result is more than 16-bits, the lower order 16-bits are stored in accumulator and higher order 16-bits are stored in DX register)

Pointer And Index Registers

- used to keep offset addresses.
- Used in various forms of memory addressing.
- In the case of SP and BP the default reference to form a physical address is the Stack Segment (SS-will be discussed under the BIU)
- The index registers (SI & DI) and the BX generally default to the Data segment register (DS).

SP: Stack pointer

– Used with SS to access the stack segment

BP: Base Pointer

– Primarily used to access data on the stack

– Can be used to access data in other segments

- SI: Source Index register

  – is required for some string operations

  – When string operations are performed, the SI register points to memory locations in the data segment which is addressed by the DS register. Thus, SI is associated with the DS in string operations.

- DI: Destination Index register

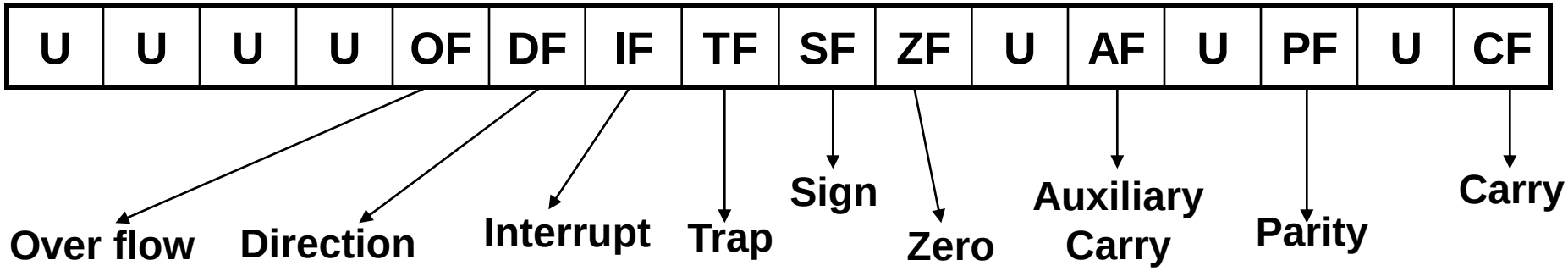
  – is also required for some string operations.

  – When string operations are performed, the DI register points to memory locations in the data segment which is addressed by the ES register. Thus, DI is associated with the ES in string operations.

- The SI and the DI registers may also be used to access data stored in arrays

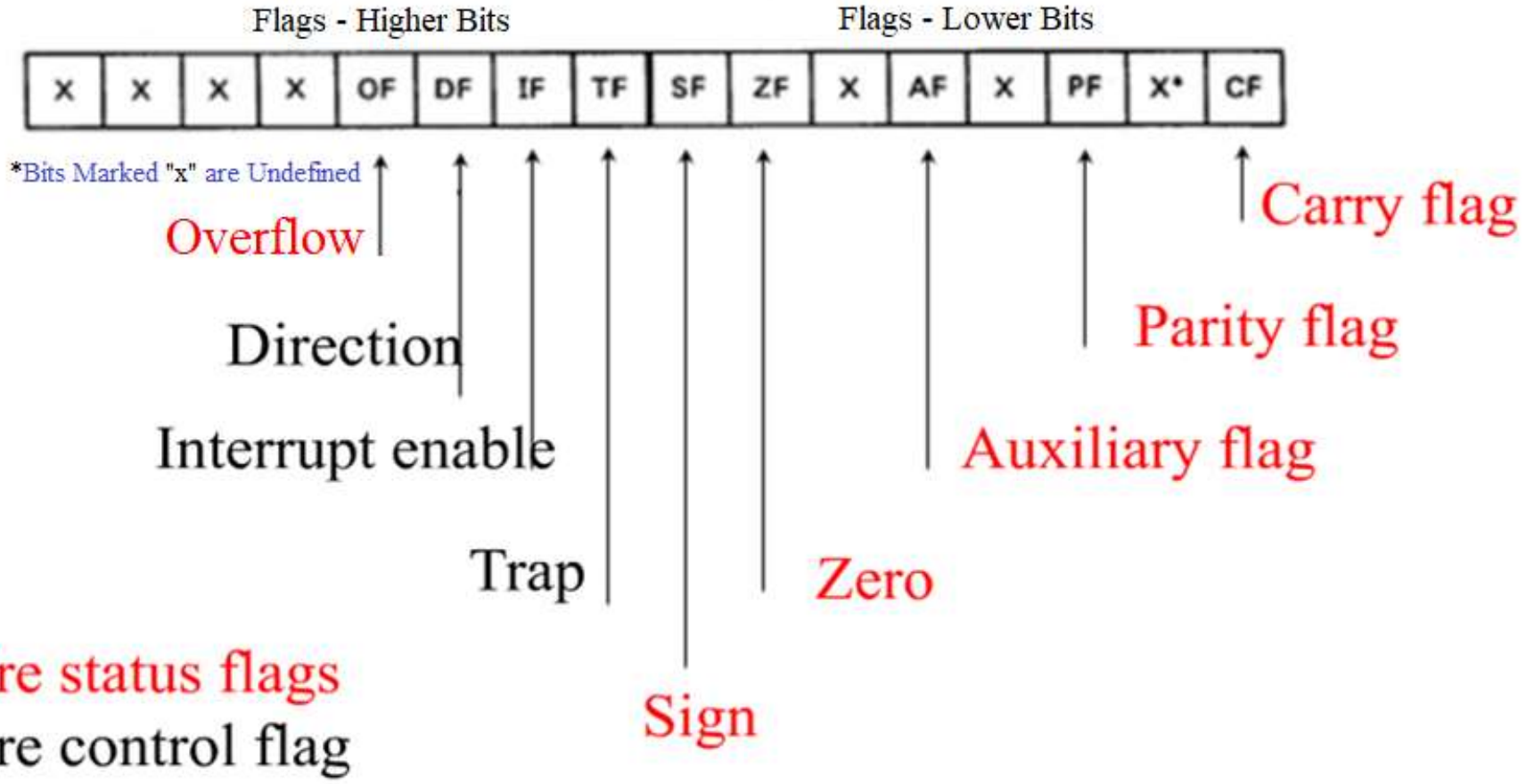
FLAG REGISTER

- A flag is a flip flop which indicates some conditions/status produced by the execution of an instruction or controls certain operations of the EU .
- In 8086 The EU contains
  - a 16 bit flag register
  - 9 of the 16 are active flags and remaining 7 are undefined.
    - 6 flags are conditional (or) status flags
    - 3 flags are control Flags



U – Unused

FLAG REGISTER



EXECUTION UNIT – Flag Register

Flag	Purpose
Carry (CF)	Holds the carry after addition or the borrow after subtraction. Also indicates some error conditions, as dictated by some programs and procedures .
Parity (PF)	PF=0;odd parity, PF=1;even parity.
Auxiliary (AF)	Holds the carry (half – carry) after addition or borrow after subtraction between bit positions 3 and 4 of the result (for example, in BCD addition or subtraction.)
Zero (ZF)	Shows the result of the arithmetic or logic operation. Z=1; result is zero. Z=0; The result is 0
Sign (SF)	Holds the sign of the result after an arithmetic/logic instruction execution. S=1; negative, S=0

Flag	Purpose
Trap (TF)	A control flag. Enables the trapping through an on-chip debugging feature.
Interrupt (IF)	A control flag. Controls the operation of the INTR (interrupt request) I=0; INTR pin disabled. I=1; INTR pin enabled.
Direction (DF)	A control flag. It selects either the increment or decrement mode for DI and /or SI registers during the string instructions.
Overflow (OF)	Overflow occurs when signed numbers are added or subtracted. An overflow indicates the result has exceeded the capacity of the Machine

Execution unit – Flag Register

- Six of the flags are status indicators reflecting properties of the last arithmetic or logical instruction.
- For example, if register AL = 7Fh and the instruction ADD AL,1 is executed then the following happen

AL = 80h

CF = 0; there is no carry out of bit 7

PF = 0; 80h has an odd number of ones

AF = 1; there is a carry out of bit 3 into bit 4

ZF = 0; the result is not zero

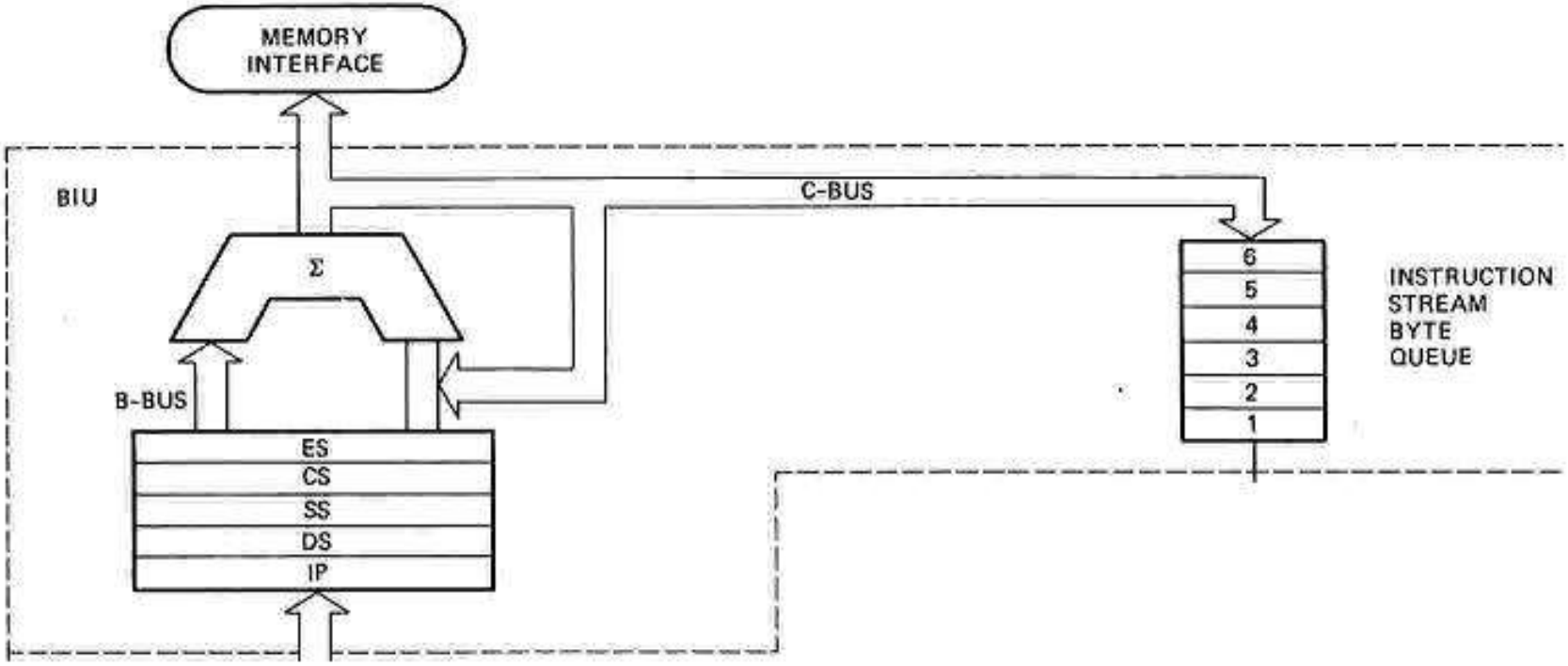
SF = 1; bit seven is one

OF = 1; the sign bit has changed

BUS INTERFACE UNIT (BIU)

Contains

- **6-byte Instruction Queue (Q)**
- **The Segment Registers (CS, DS, ES, SS).**
- **The Instruction Pointer (IP).**



THE QUEUE (Q)

- The BIU uses a mechanism known as an **instruction stream queue** to implement a ***pipeline architecture.***

- This queue permits pre-fetch of up to **6 bytes** of instruction code. Whenever the queue of the BIU is not full, it has room for at least two more bytes and at the same time the EU is not requesting it to read or write operands from memory, the BIU is free to look ahead in the program by pre-fetching the next sequential instruction.

- These pre-fetching instructions are held in its **FIFO queue**. With its **16** bit data bus, the BIU fetches two instruction bytes in a single memory cycle.

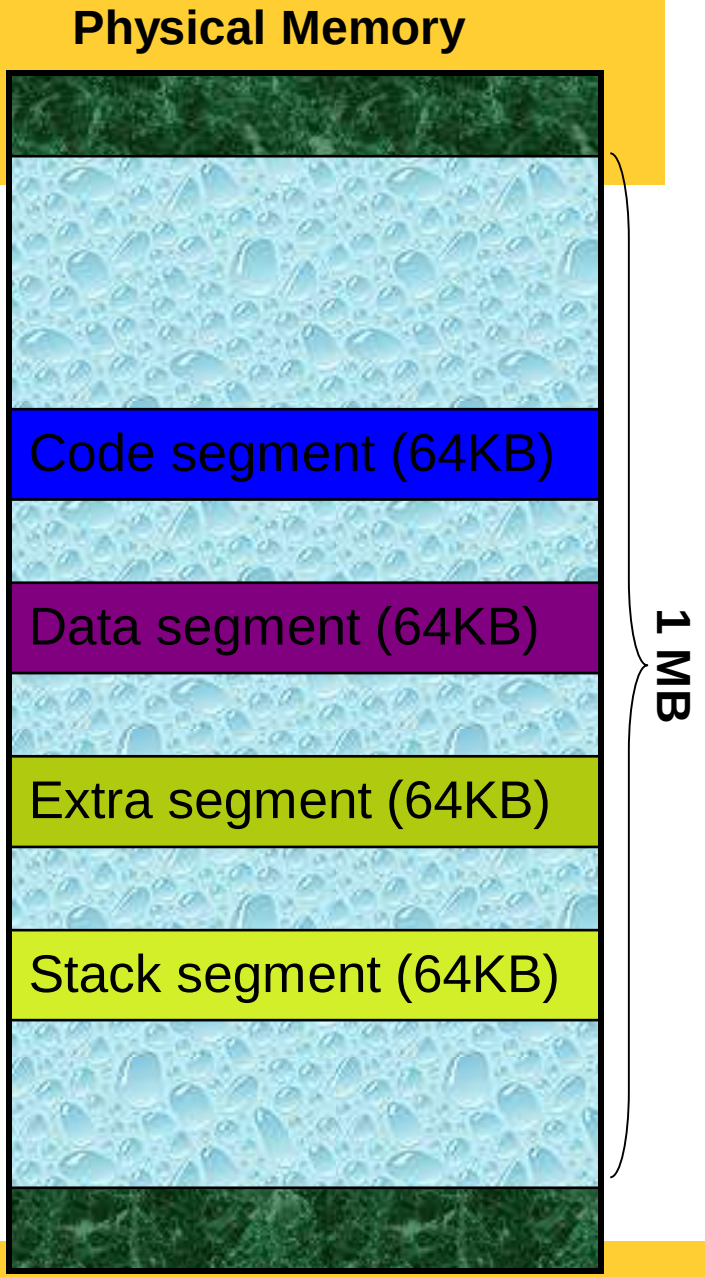
- After a byte is loaded at the input end of the queue, it automatically shifts up through the **FIFO** to the empty location nearest the output.

- The **EU accesses the queue from the output end**. It reads one instruction byte after the other from the output of the queue.

- The intervals of no bus activity, which may occur between bus cycles are known as ***Idle state*.**

Segmented Memory

- The memory in an 8086/88 based system is organized as segmented memory.
- The CPU 8086 is able to address 1Mbyte of memory.
- The Complete physically available memory may be divided into a number of logical segments.



- The size of each segment is 64 KB
- A segment is an area that begins at any location which is divisible by 16.
- A segment may be located any where in the memory
- Each of these segments can be used for a specific function.

  – Code segment is used for storing the instructions.
  – The stack segment is used as a stack and it is used to store the return addresses.
  – The data and extra segments are used for storing data byte.

* **In the assembly language programming, more than one data/ code/ stack segments can be defined. But only one segment of each type can be accessed at any time.**

- The 4 segments are Code, Data, Extra and Stack segments.
- A Segment is a 64kbyte block of memory.
- The 16 bit contents of the segment registers in the BIU actually point to the starting location of a particular segment.
- Segments may be overlapped or non-overlapped

Advantages of Segmented memory Scheme

- Allows the memory capacity to be 1Mb although the actual addresses to be handled are of 16 bit size.
- Allows the placing of code, data and stack portions of the same program in different parts (segments) of the m/y, for data and code protection.
- Permits a program and/or its data to be put into different areas of memory each time program is executed, i.e. provision for relocation may be done .
- The segment registers are used to allow the instruction, data or stack portion of a program to be more than 64Kbytes long. The above can be achieved by using more than one code, data or stack segments.

Segment registers

- In 8086/88 the processors have 4 segments registers
- Code Segment register (CS), Data Segment register (DS), Extra Segment register (ES) and Stack Segment (SS) register.
- All are 16 bit registers.
- Each of the Segment registers store the upper 16 bit address of the starting address of the corresponding segments.

CS Register

This register contains the initial address of the code segment. This address plus the offset value contained in the instruction pointer (IP) indicates the address of an instruction to be fetched for execution.

SS Register

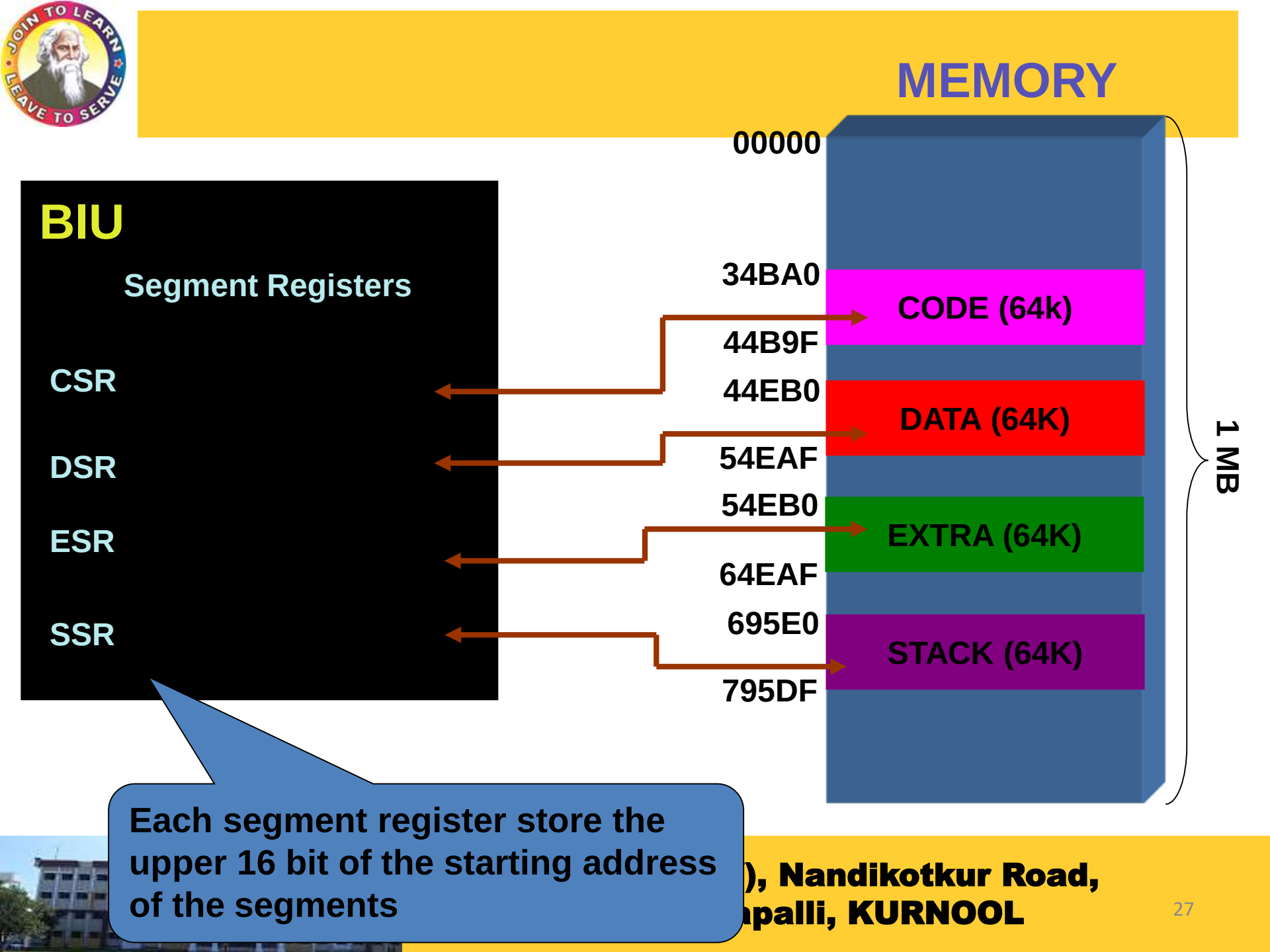
The stack segment register contains the initial address of the stack segment. This address plus the value contained in the stack pointer (SP) is used for stack operations.

DS Register

The Data segment register contains the initial address of the current data segment. This address plus the offset value in instruction causes a reference to a specific location in the data segment.

ES Register

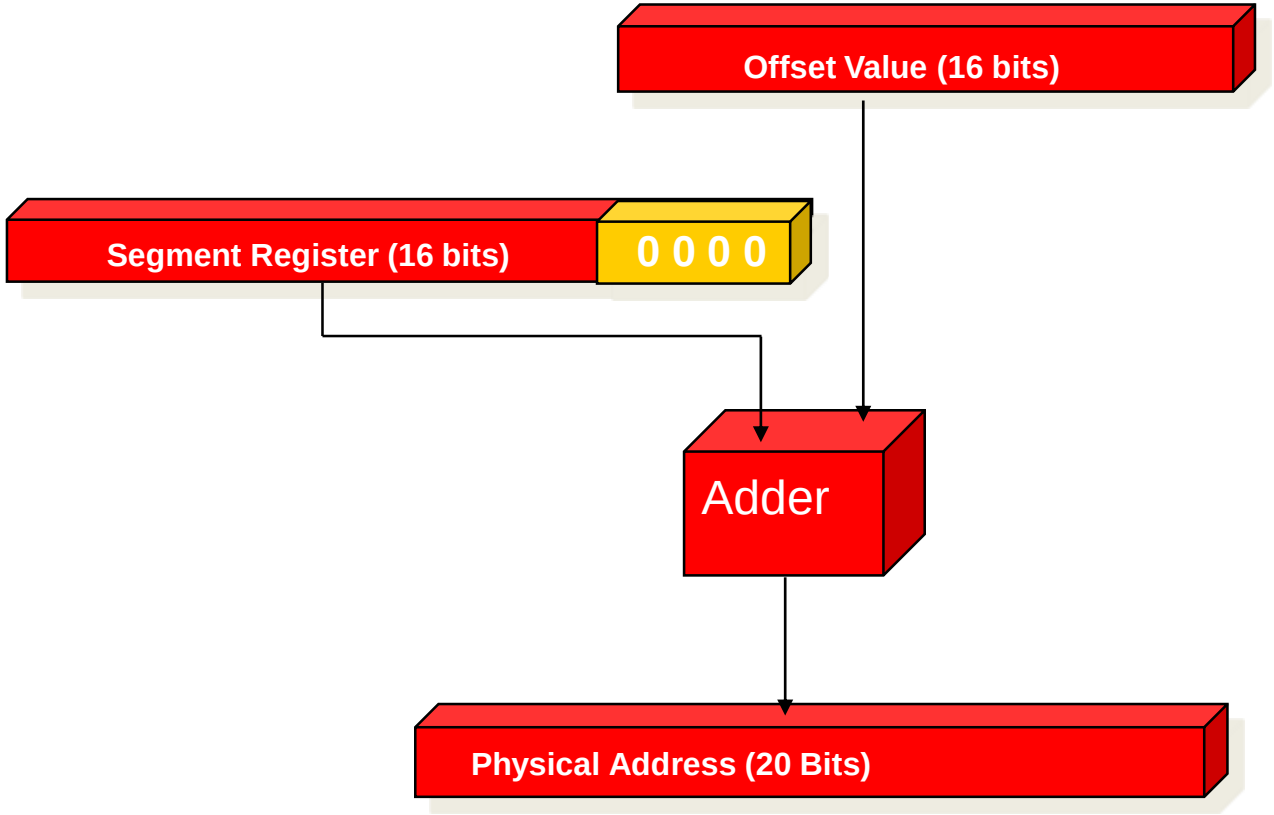
Extra segment is used by some string operations. The Extra segment register contains the initial address of the extra segment. String instructions always use the ES and DI registers to calculate the physical address for the destination.

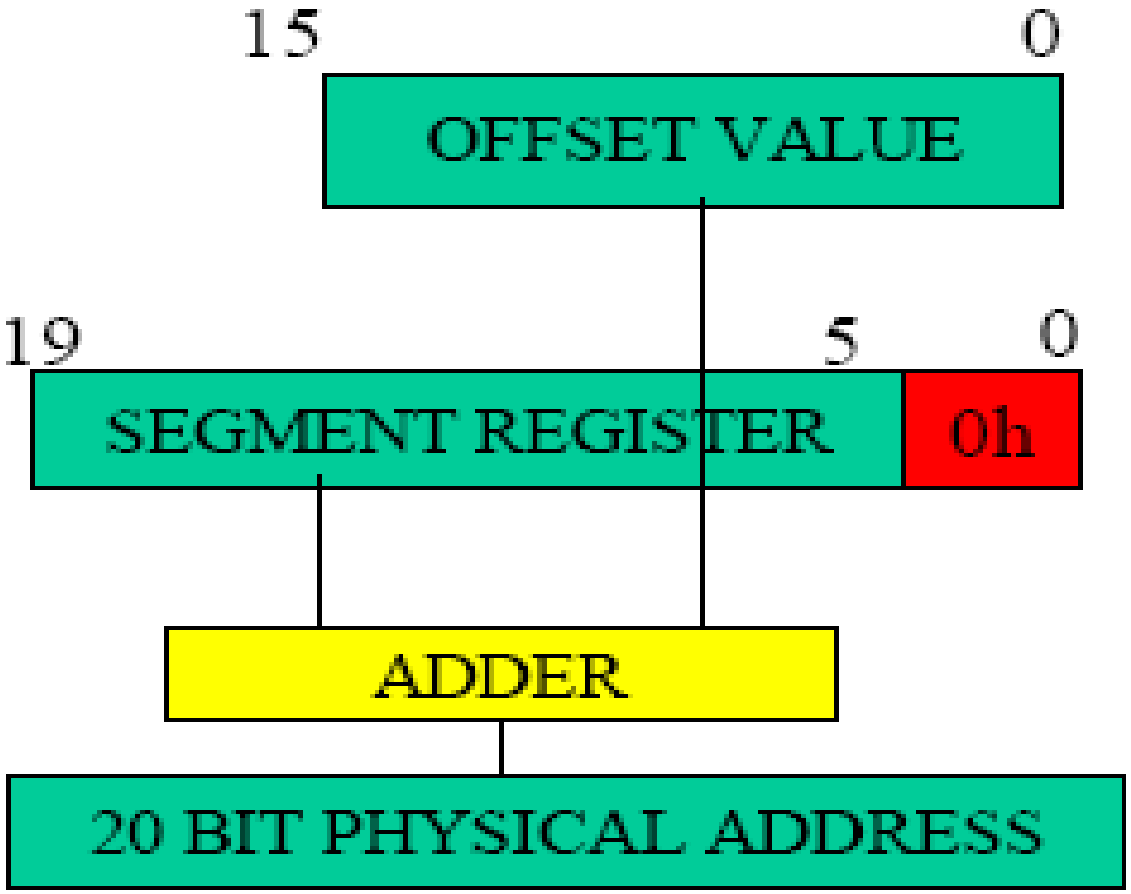

MEMORY
00000
BIU
Segment Registers
CSR
DSR
ESR
SSR
34BA0
CODE (64k)
44B9F
44EB0
DATA (64K)
54EAF
54EB0
EXTRA (64K)
64EAF
695E0
STACK (64K)
795DF
1 MB
Each segment register store the upper 16 bit of the starting address of the segments

Instruction pointer & summing block

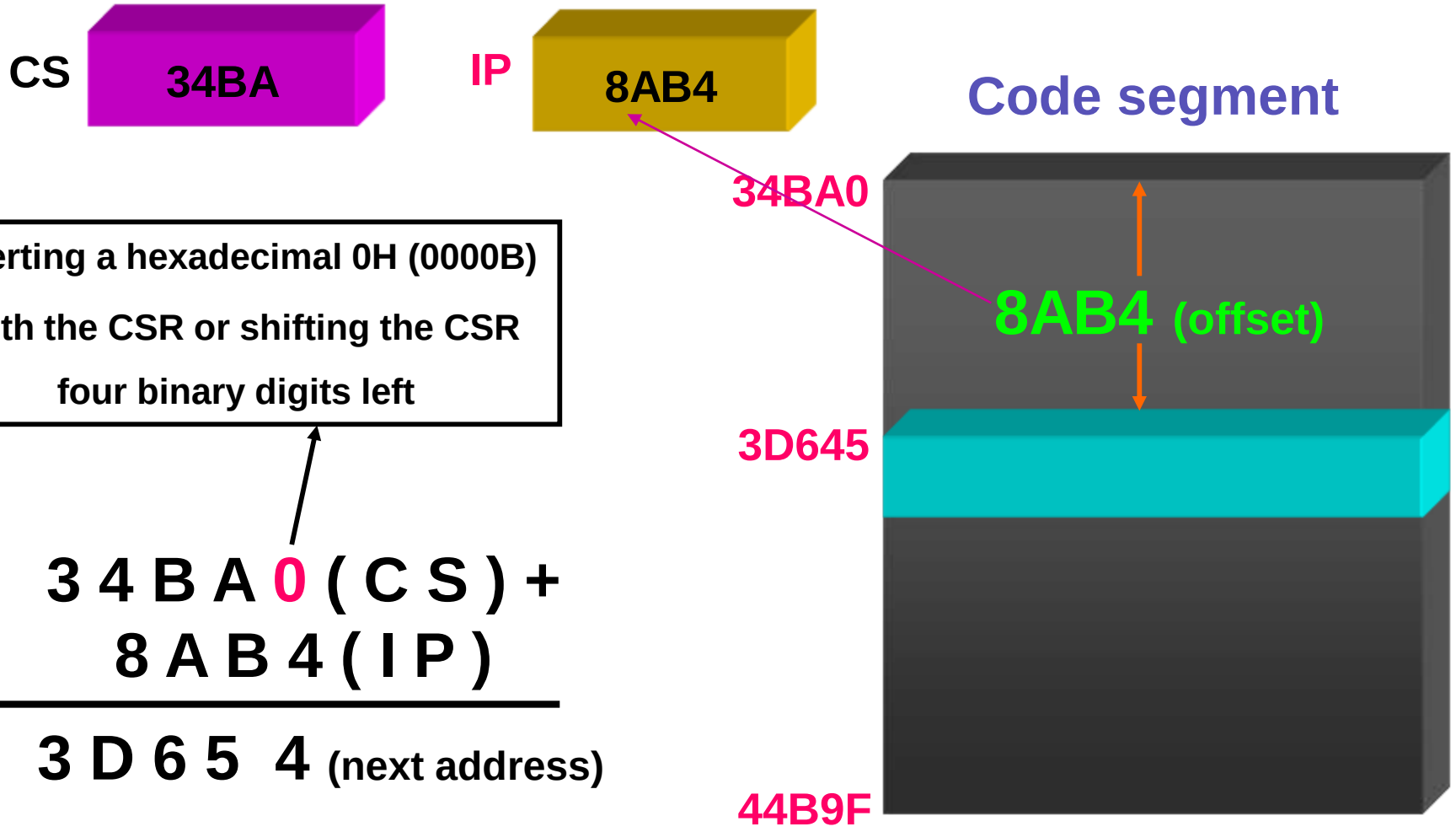
- The instruction pointer register contains a 16-bit offset address of instruction that is to be executed next.
- The IP always references the Code segment register (CS).
- The value contained in the instruction pointer is called as an offset because this value must be added to the base address of the code segment, which is available in the CS register to find the 20-bit physical address.
- The value of the instruction pointer is incremented after executing every instruction.

Memory Address Generation




15
0
OFFSET VALUE
19
5
0
SEGMENT REGISTER
0h
ADDER
20 BIT PHYSICAL ADDRESS

- **The following examples shows the CS:IP scheme of address formation:**



- **Example For Address Calculation (segment: offset)**

- If the data segment starts at location 1000h and a data reference contains the address 29h where is the actual data?

Offset 0000 0000 0010 1001

Segment Address 0001 0000 0000 0000 0000

Required Address 0001 0000 0000 0010 1001

Segment and Offset Address register combination

- CS:IP
- SS:SP SS:BP
- DS:BX DS:SI
- DS:DI (for other than string operations)
- ES:DI (for string operations)

Summary of Registers & Pipeline of 8086 µP

EU

AX	AH	AL
BX	BH	BL
CX	CH	CL
DX	DH	DL

SP
BP
SI
DI

FLAGS

DECODER

ALU

Timing control

BIU

Fetch & store code bytes in PIPELINE

CODE OUT

PIPELINE (or) QUEUE

CODE IN

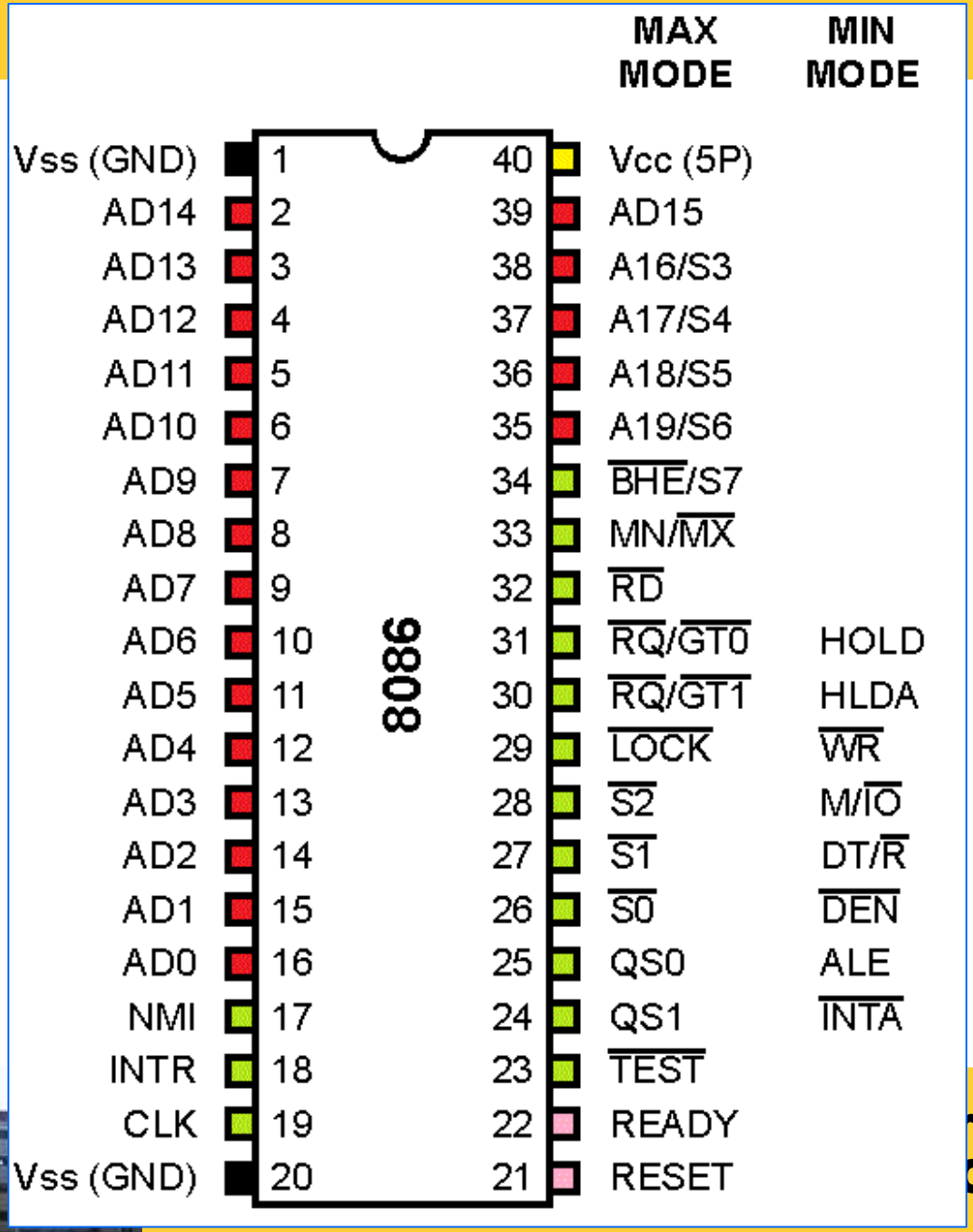
IP

CS DS ES SS

IP	BX	DI	SP
	DI		BP
	SI		

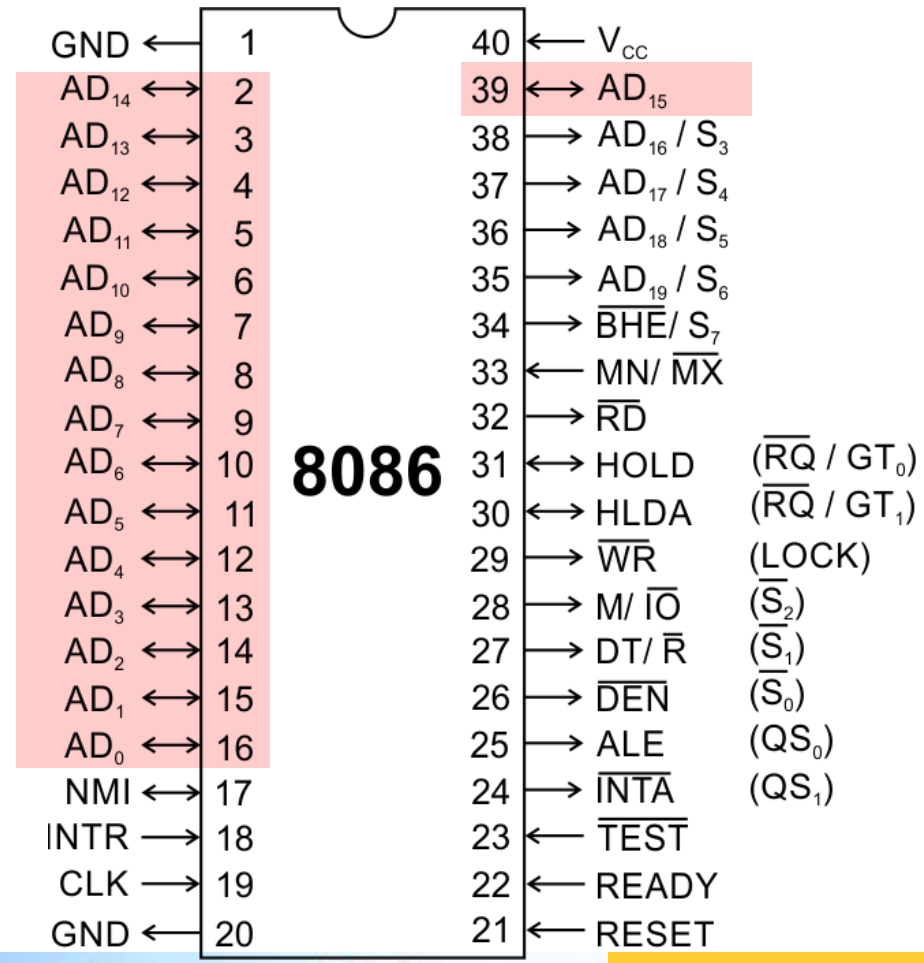
Default Assignment

PIN DIAGRAM OF 8086



Pins and Signals

Common signals



AD_0-AD_{15} (Bidirectional)

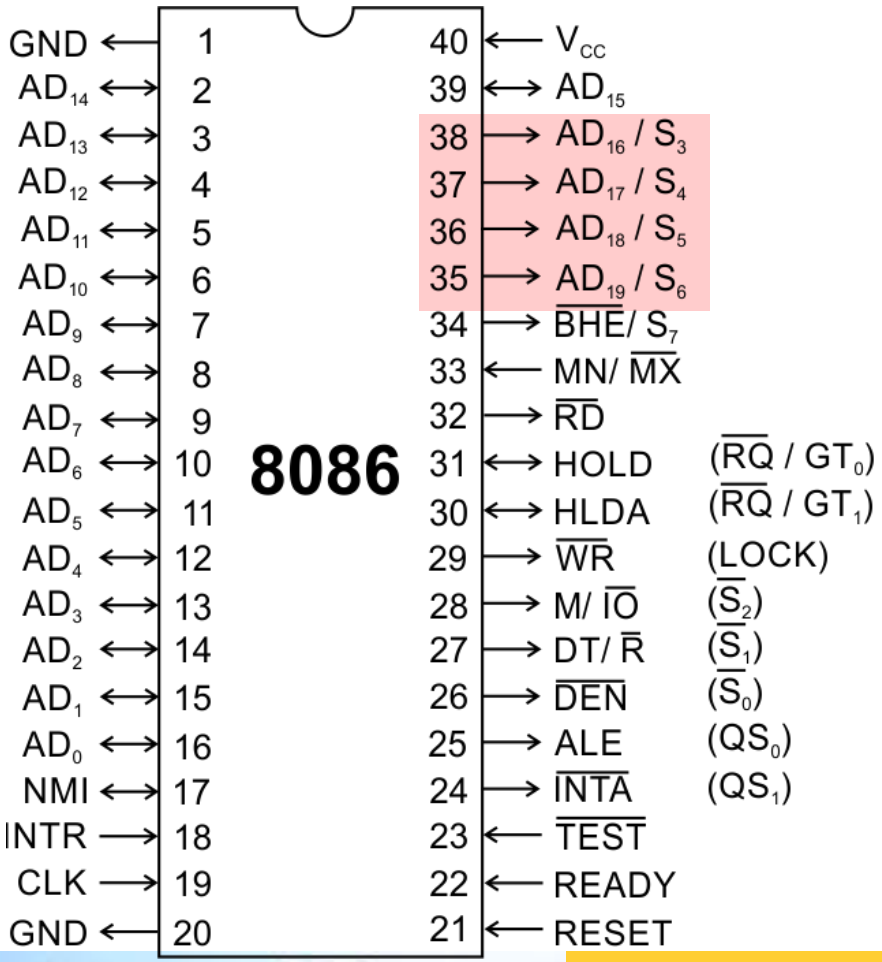
Address/Data bus

These are the time multiplexed address (memory or I/O) and data lines. Address (A0-A15) remains on the lines during T1 state, while the data (D0-D15) is available on the data bus during T2, T3, Tw and T4.

Here T1,T2, T3, T4 and Tw are the clock states of a machine cycle. Tw is a wait state

Pins and Signals

Common signals



A_{16}/S_3, A_{17}/S_4, A_{18}/S_5, A_{19}/S_6

These are the time multiplexed address and status lines. During T1 These are most significant address lines for memory operations and low line for I/O operations, Status information is available on these during T2, T3, Tw and T4. $S_5 \rightarrow$ Status of Interrupt Enable flag bit. S_4 & $S_3 \rightarrow$ Indicates which Segment Register is presently used for memory access. $S_6 \rightarrow$ Is always low

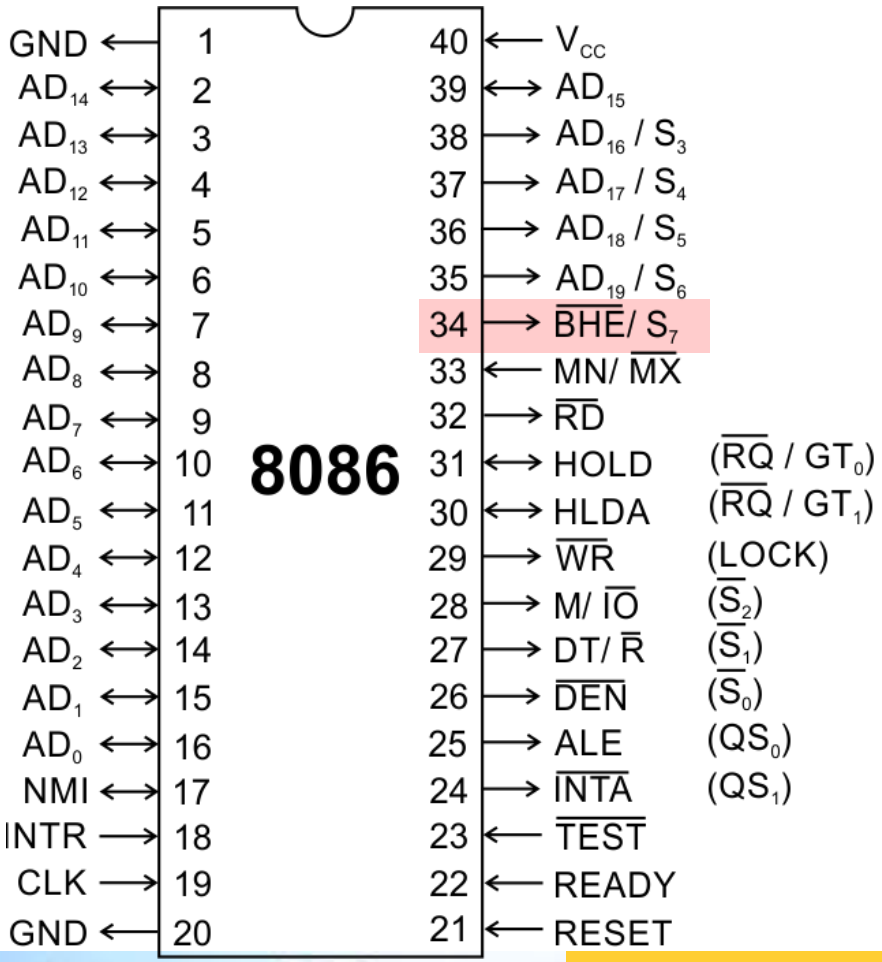
Table: Status of S_3 and S_4

S_4	S_3	*Indications*
0	0	Alternate Data
0	1	Stack
1	0	Code or none
1	1	Data

RCEW, Pasup
Near Ve

Pins and Signals

Common signals



$\overline{BHE}$ (Active Low)/S_7 (Output)

Bus High Enable/Status

$\overline{BHE}$→ It is used to indicate the transfer of data over the higher order (D15-D8) data bus. It goes low for the data transfers over D15-D8 and is used to derive chip selects of odd address memory bank or peripherals.

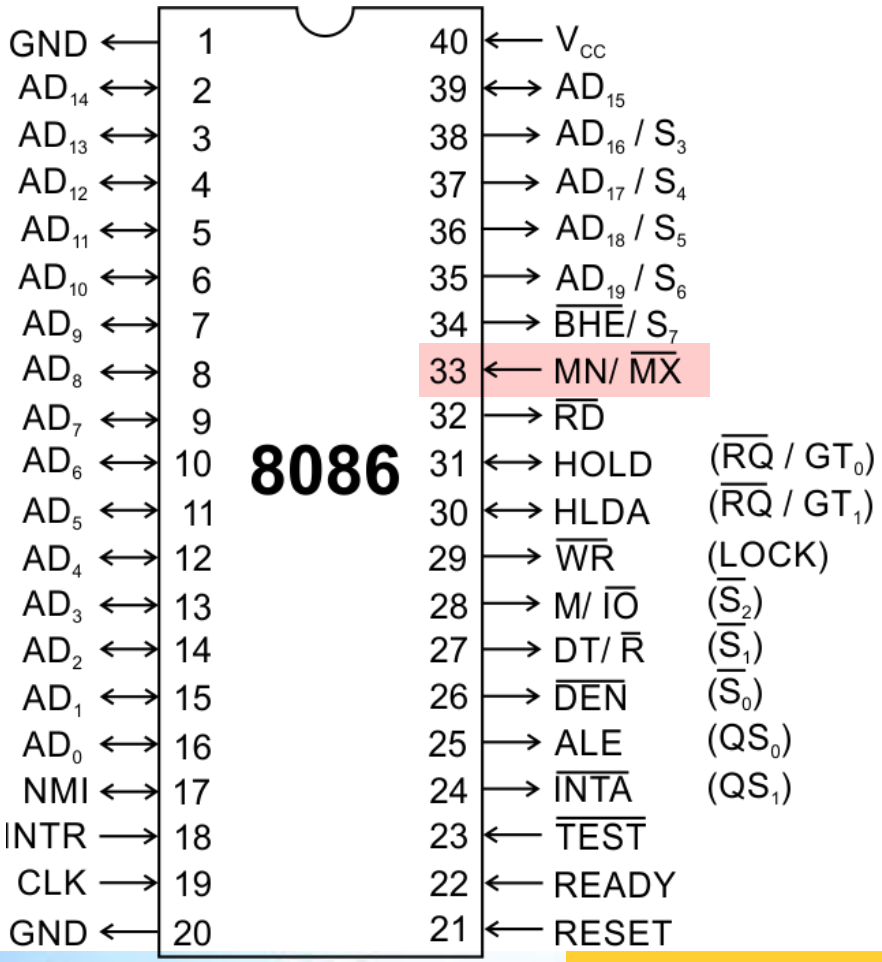
S_7→ Is not currently used

Table: *Bus High Enable and A_0*

$\overline{BHE}$	A_0	*Indication*
0	0	Whole word (2 bytes)
0	1	Upper byte from or to odd address.
1	0	Lower byte from or to even address
1	1	None

Pins and Signals

Common signals



MN/ $\overline{MX}$

MINIMUM / MAXIMUM

This pin decides the processor operating mode.

MN/ $\overline{MX}$ = 0 → Maximum (multiprocessor) mode

MN/ $\overline{MX}$ = 1 → Minimum (single processor) mode

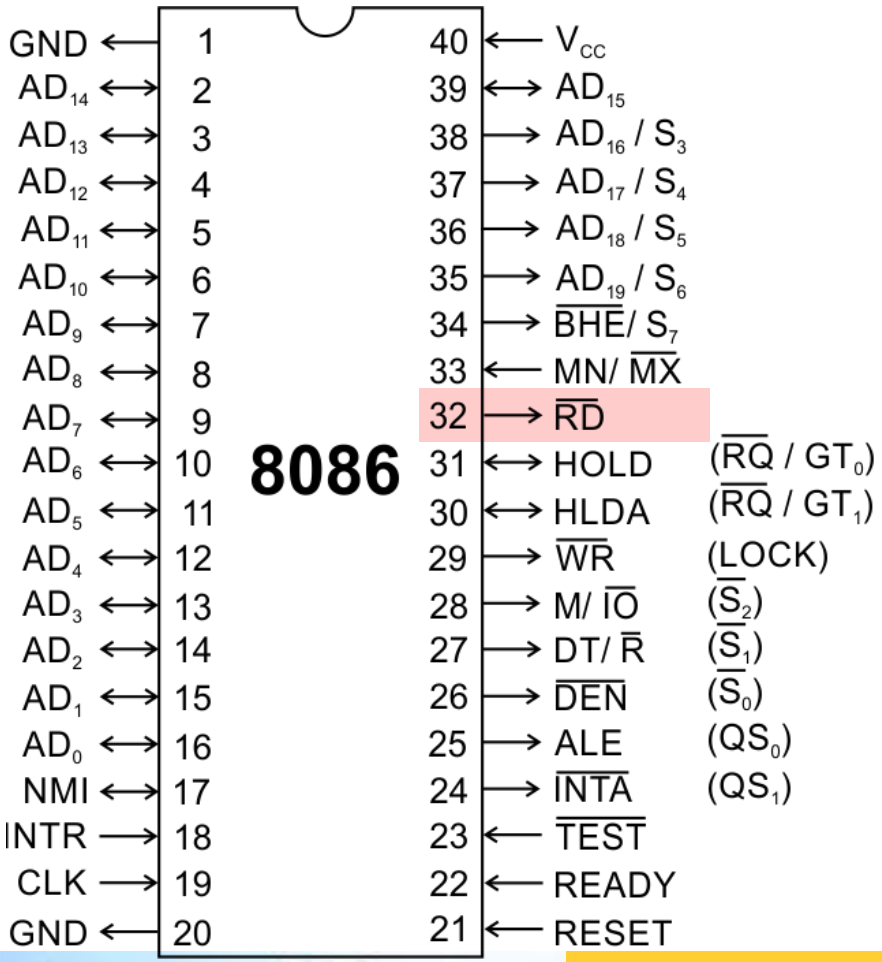
$\overline{RD}$ - Read (Active Low)

$\overline{RD}$ = 0 i.e. Low signal indicates read (memory or I/O) operation.

It is an output signal.

Pins and Signals

Common signals



TEST

TEST input pin is tested (examined) by the 'WAIT' instruction.

8086 processor will enter into a wait state after execution of the WAIT instruction and will resume execution only when the TEST input pin goes low.

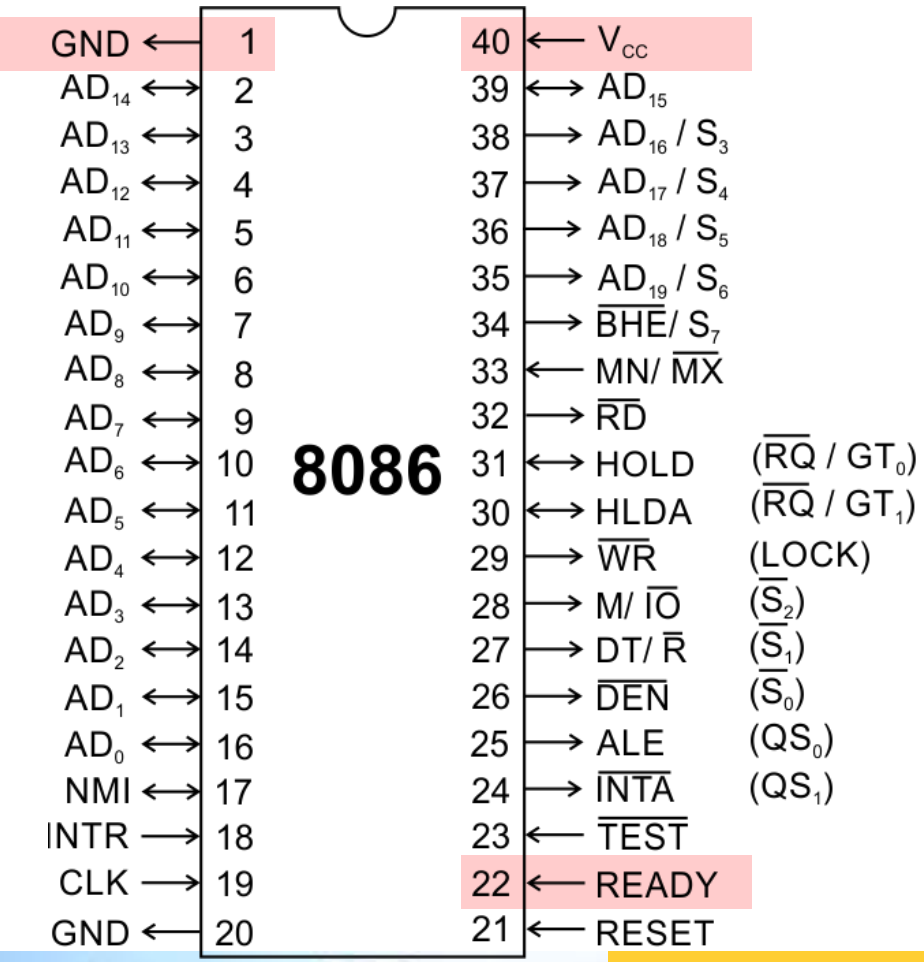
READY

This is the acknowledgement from the slow devices or memory that they have completed the data transfer.

The signal is active high.

Pins and Signals

Common signals



RESET (Input)

Causes the processor to immediately terminate the current activity. It restarts the execution when RESET is low

The signal must be active HIGH for at least four clock cycles.

VCC

+5v Power Supply pin

GND

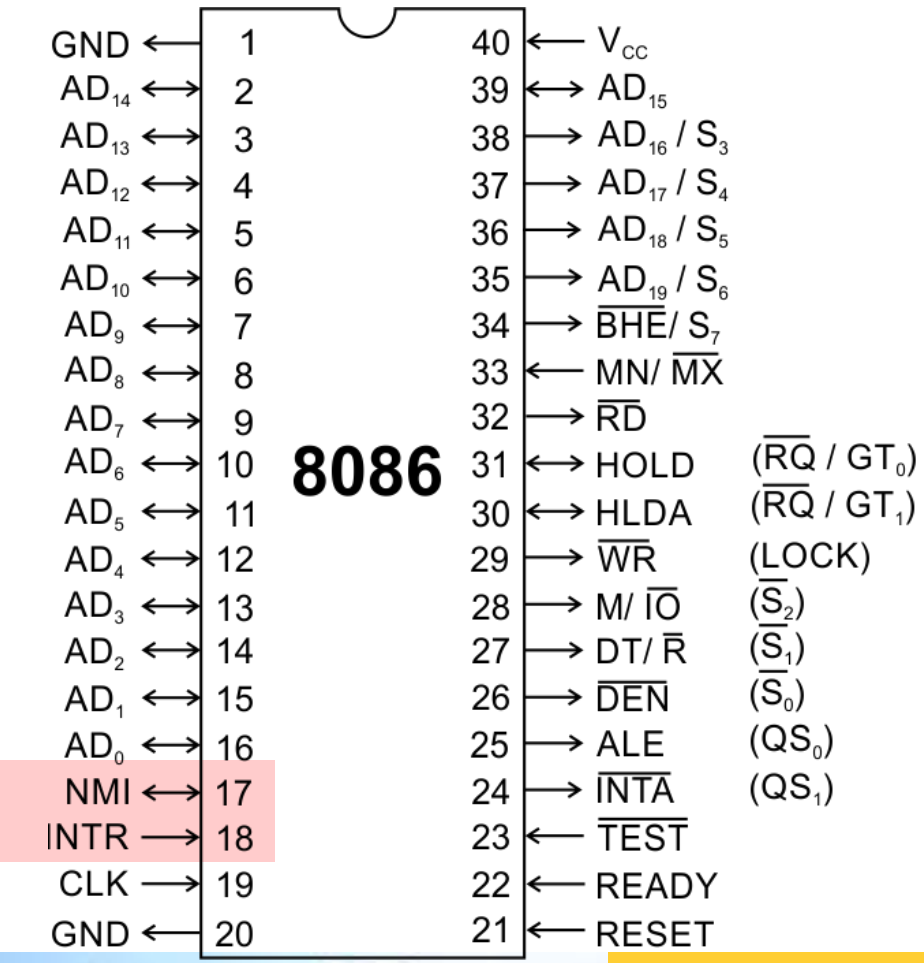
Ground pin

CLK

The clock input provides the basic timing for processor operation and bus control activity. Its an asymmetric square wave with 33% duty cycle. Clock input frequency is 5MHz

Pins and Signals

Common signals



INTR Interrupt Request

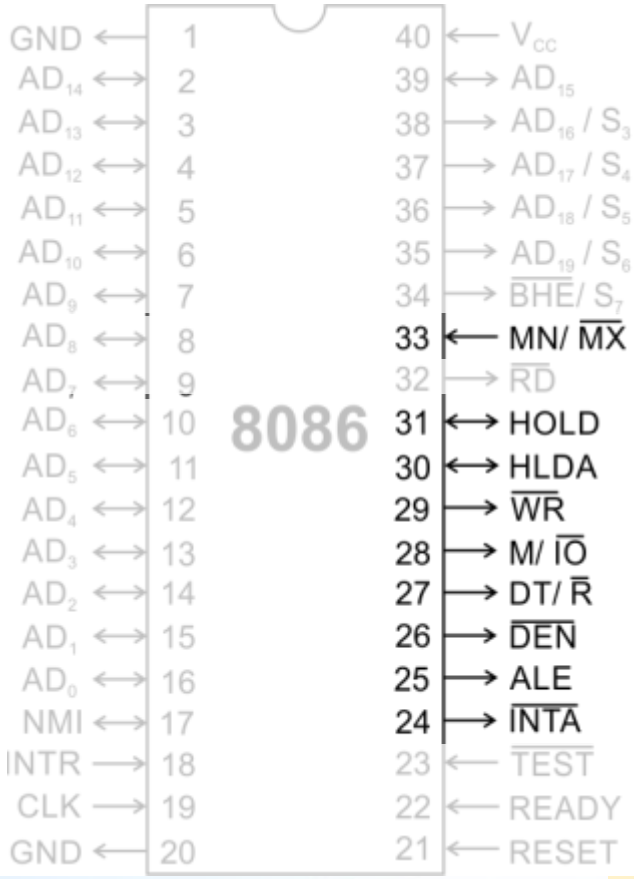
- **This is a level triggered input. If this is high, indicates the availability of the interrupt on this pin.**
- **This signal is active high.**
- **This can be disabled by Interrupt Enable flag bit (IF) by making IF=0**

NMI
Non Maskable Interrupt

- **This is an edge-triggered input which is a Type2 Interrupt.**
- **There is no disable option.**
- **A transition from low to high initiates the NMI interrupt**

Pins and Signals

Min/ Max Pins



The 8086 microprocessor can work in two modes of operations : Minimum mode and Maximum mode.

In the minimum mode of operation the microprocessor do not associate with any co-processors and can not be used for multiprocessor systems.

In the maximum mode the 8086 can work in multi-processor or co-processor configuration.

Minimum or maximum mode operations are decided by the pin MN/$\overline{MX}$ (Active low).

When this pin is high 8086 operates in minimum mode otherwise it operates in Maximum mode.

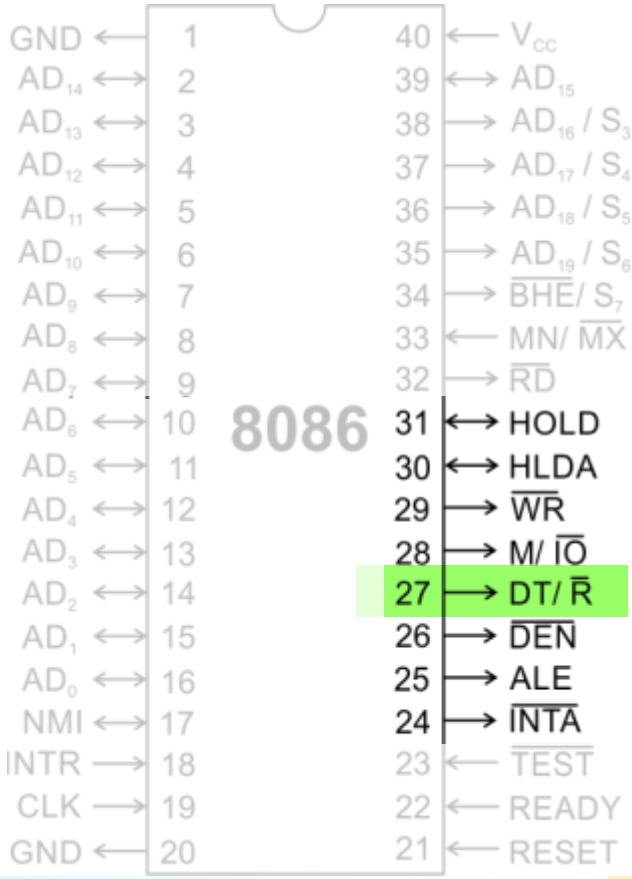
Pins and Signals

Minimum mode signals

Pins 24 -31

For minimum mode operation, the MN/$\overline{MX}$ is tied to VCC (logic high)

8086 itself generates all the bus control signals



DT/$\overline{R}$ — (Data Transmit/ Receive) Output signal used to decide the direction of data flow through the data transceivers. When the processor sends out data, this signal is high and when it is receiving data, this signal is low

$\overline{DEN}$ — (Data Enable) It is used to enable the transceivers to separate the data from multiplexed address/data lines. This signal is active low

ALE — (Address Latch Enable) It is used to enable the latches to separate the address from multiplexed address/data lines. This signal is active high

M/$\overline{IO}$ — Used to differentiate memory access and I/O access. For memory reference instructions, it is high. For IN and OUT instructions, it is low.

$\overline{WR}$ — Write control signal, asserted low Whenever processor writes data to memory or I/O port

$\overline{INTA}$ — (Interrupt Acknowledge) When the interrupt request is accepted by the processor, the output is low on this line

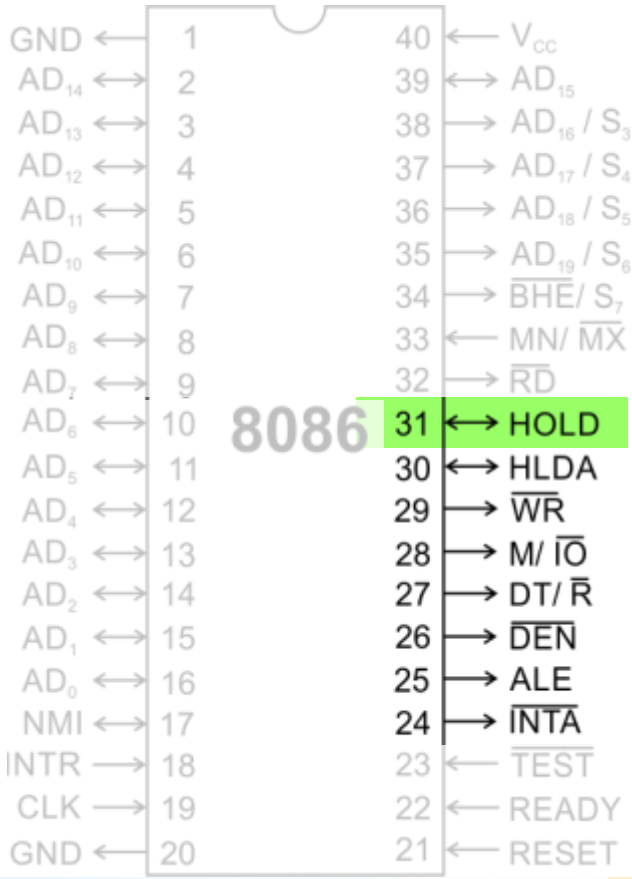
Pins and Signals

Minimum mode signals

Pins 24 -31

For minimum mode operation, the MN/$\overline{MX}$ is tied to VCC (logic high)

8086 itself generates all the bus control signals



HOLD — **When the HOLD line goes high, it indicates to the processor that another master requesting the bus access.**

Usually used by the DMA controller to get the control of the bus.

HLDA — **(Hold Acknowledge) Acknowledge signal by the processor to the bus master requesting the control of the bus through HOLD.**

The acknowledge is asserted high, when the processor receives HOLD request high.

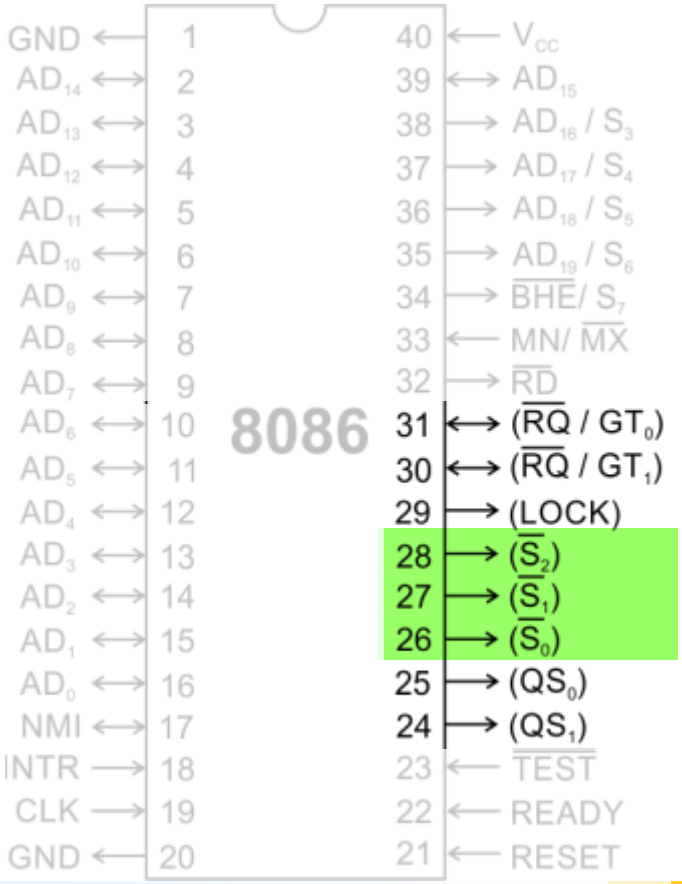
When the processor detects the HOLD line low, it lowers the HLDA signal.

Pins and Signals

Maximum mode signals

For maximum mode operation, the MN/$\overline{MX}$ is grounded (logic low)

Pins 24 -31 are reassigned



$\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$

Status signals: These are status lines which indicate the type of operation, being carried out by the processor

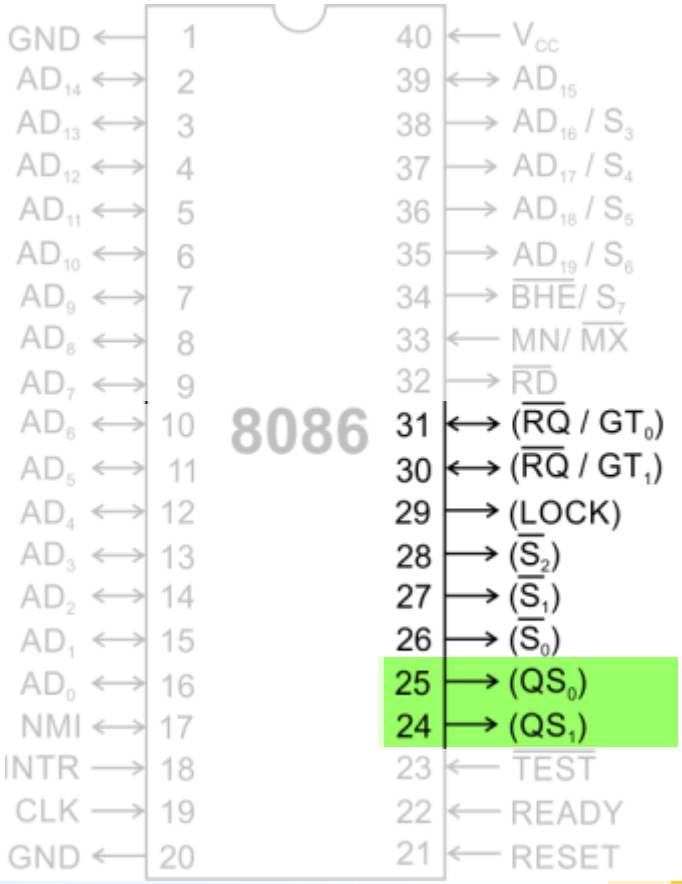
Status Signal			Indication
$\overline{S}_2$	$\overline{S}_1$	$\overline{S}_0$	
0	0	0	Interrupt acknowledge
0	0	1	Read I/O port
0	1	0	Write I/O port
0	1	1	Halt
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write memory
1	1	1	Passive/Inactive

Pins and Signals

Maximum mode signals

For maximum mode operation, the MN/$\overline{MX}$ is grounded (logic low)

Pins 24 -31 are reassigned



$\overline{QS_0}$, $\overline{QS_1}$ **Queue Status These lines provides the status of queue.**

The queue status can be used by external device to track the internal status of the queue in 8086.

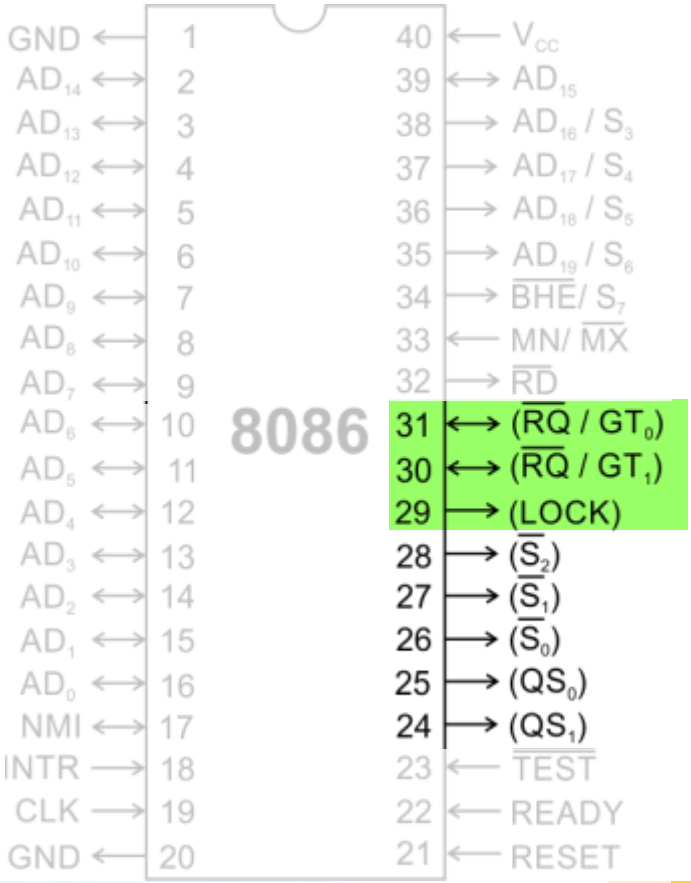
Queue status		Queue operation
QS_1	QS_0	
0	0	No operation
0	1	First byte of an opcode from queue
1	0	Empty the queue
1	1	Subsequent byte from queue

Pins and Signals

Maximum mode signals

For maximum mode operation, the MN/$\overline{MX}$ is grounded (logic low)

Pins 24 -31 are reassigned



$\overline{RQ}/\overline{GT_0}$, $\overline{RQ}/\overline{GT_1}$ — **Bus Request/Bus Grant** These pins are used by 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 $\overline{RQ}/\overline{GT_0}$ will have higher priority than $\overline{RQ}/\overline{GT_1}$

$\overline{LOCK}$ — The Low signal on this pin indicates the accessing system bus is prevented by other system bus masters.

It is activated by 'LOCK' prefix instruction and remains active until the completion of next instruction

MEMORY ORGANIZATION and MEMORY BANKS ACCESSING

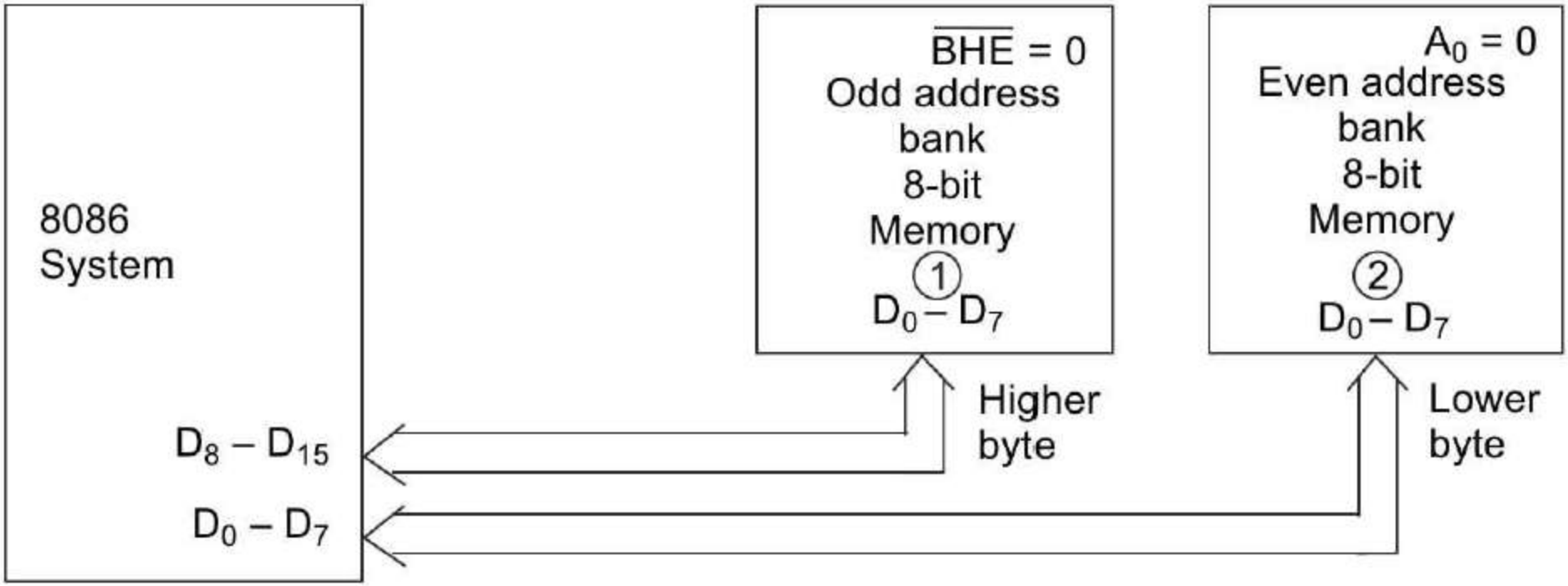


Fig. *Physical Memory Organisation*

TIMING and CONTROL SIGNALS

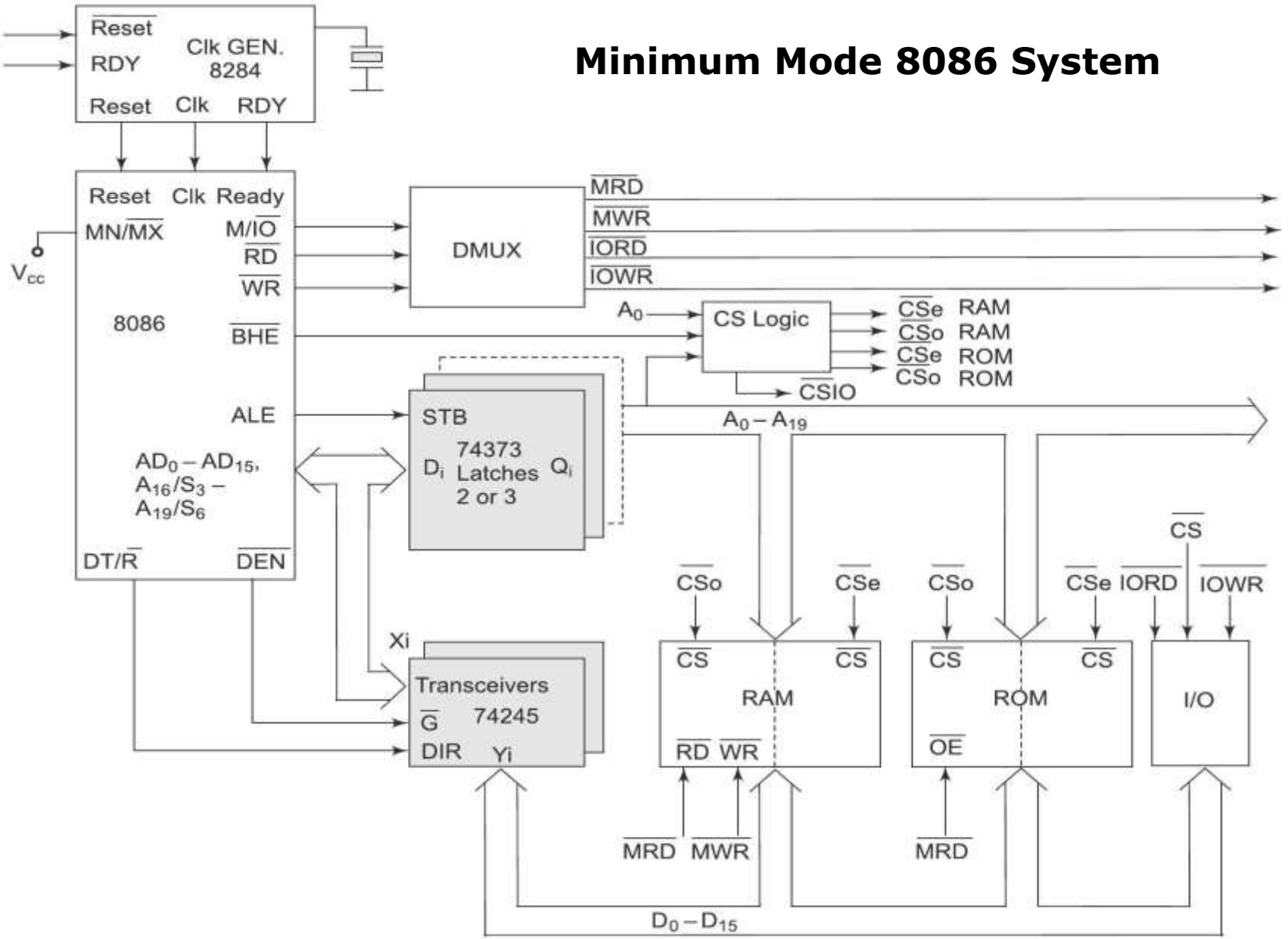
In Minimum Mode

- ALE
- $\overline{BHE}$
- M/$\overline{IO}$
- DT/$\overline{R}$
- $\overline{RD}$
- $\overline{WR}$
- $\overline{DEN}$
- HOLD
- HLDA

In Maximum Mode

- ALE
- $\overline{BHE}$
- $\overline{S0}$
- $\overline{S1}$
- $\overline{S2}$
- DT/$\overline{R}$
- $\overline{DEN}$
- $\overline{RQ}$/$\overline{GT}$

SYSTEM TIMING DIAGRAMS



SYSTEM TIMING DIAGRAMS

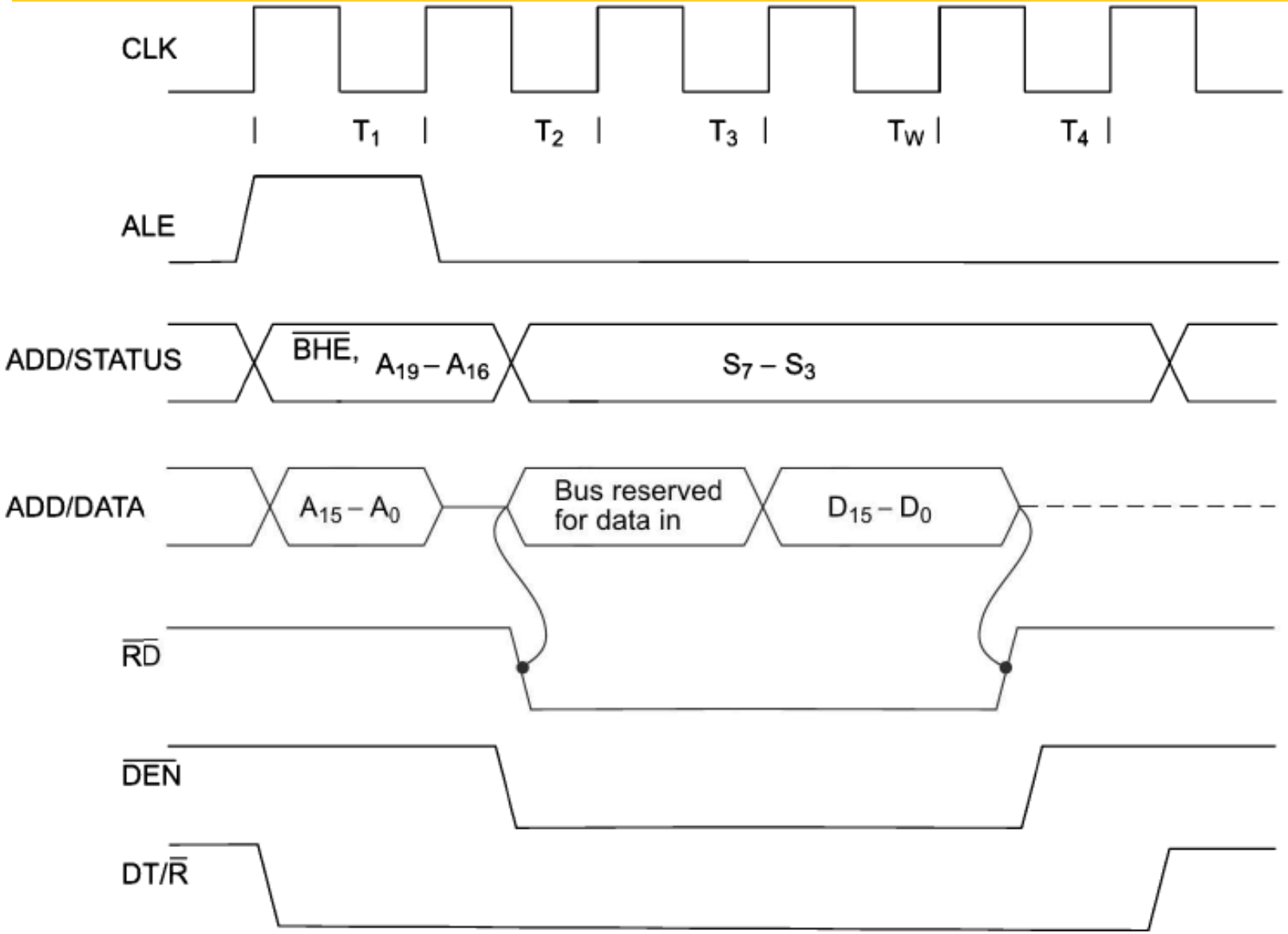


Fig. *Read Cycle Timing Diagram for Minimum Mode*

SYSTEM TIMING DIAGRAMS

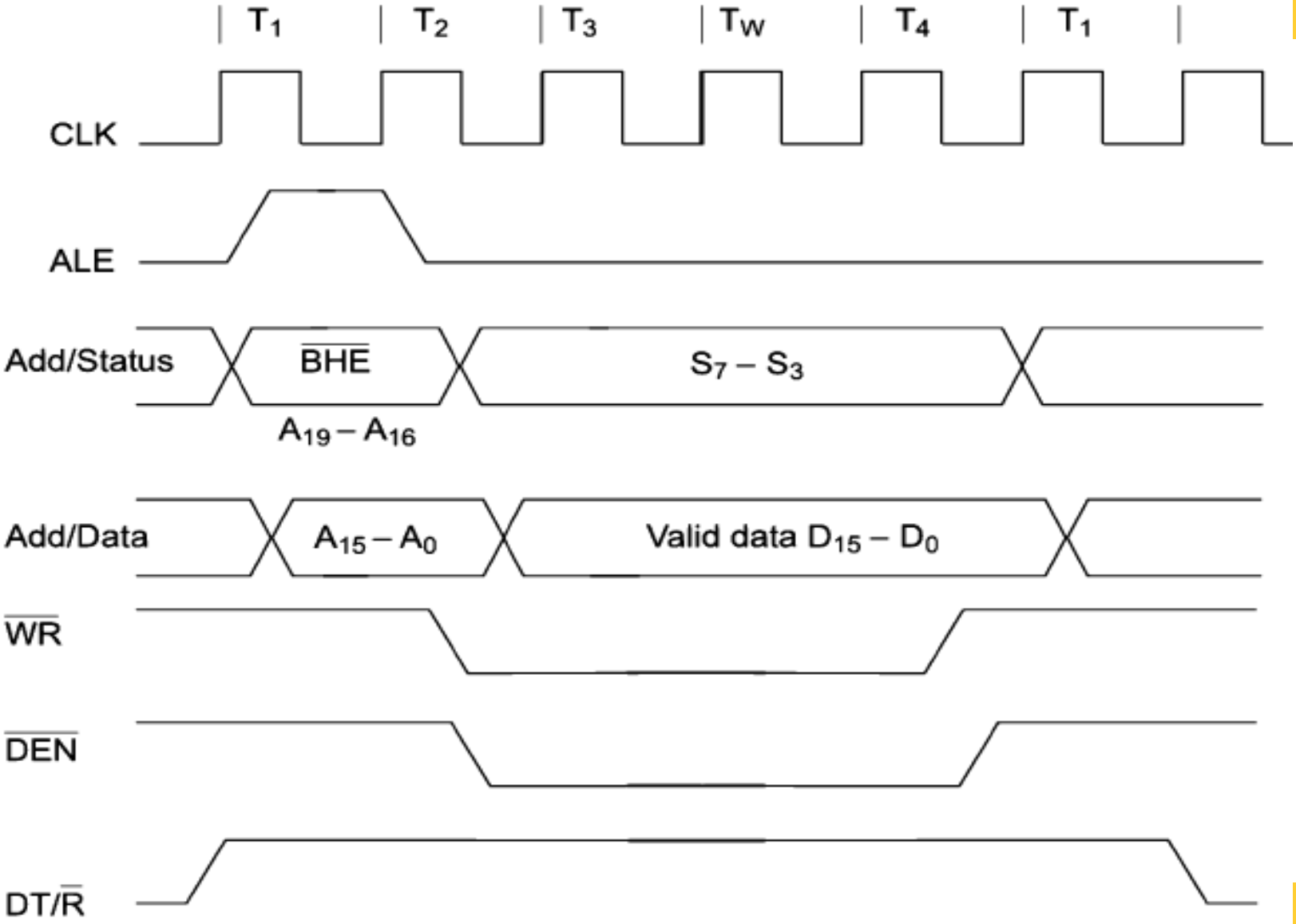


Fig. *Write Cycle Timing Diagram for Minimum Mode Operation*

SYSTEM TIMING DIAGRAMS

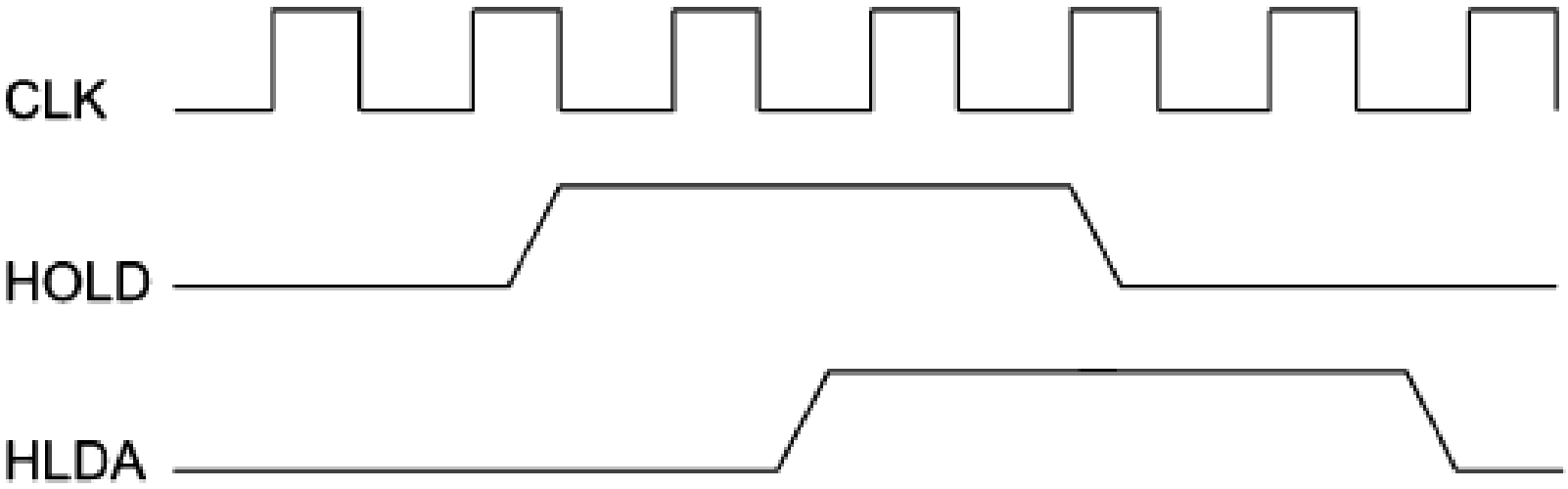
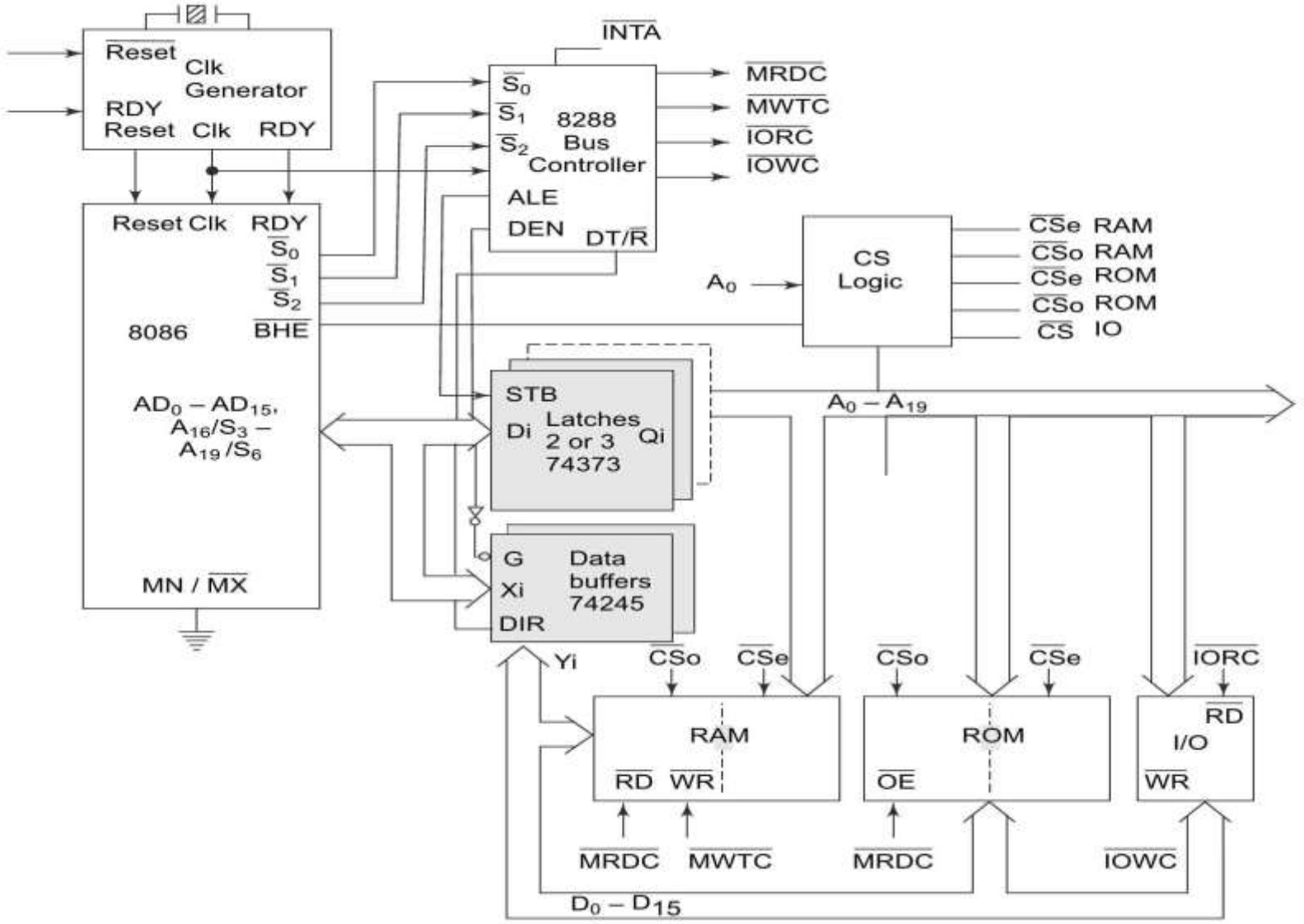


Fig. *Bus Request and Bus Grant Timings in Minimum Mode System*

Maximum Mode 8086 System



Maximum Mode 8086 System

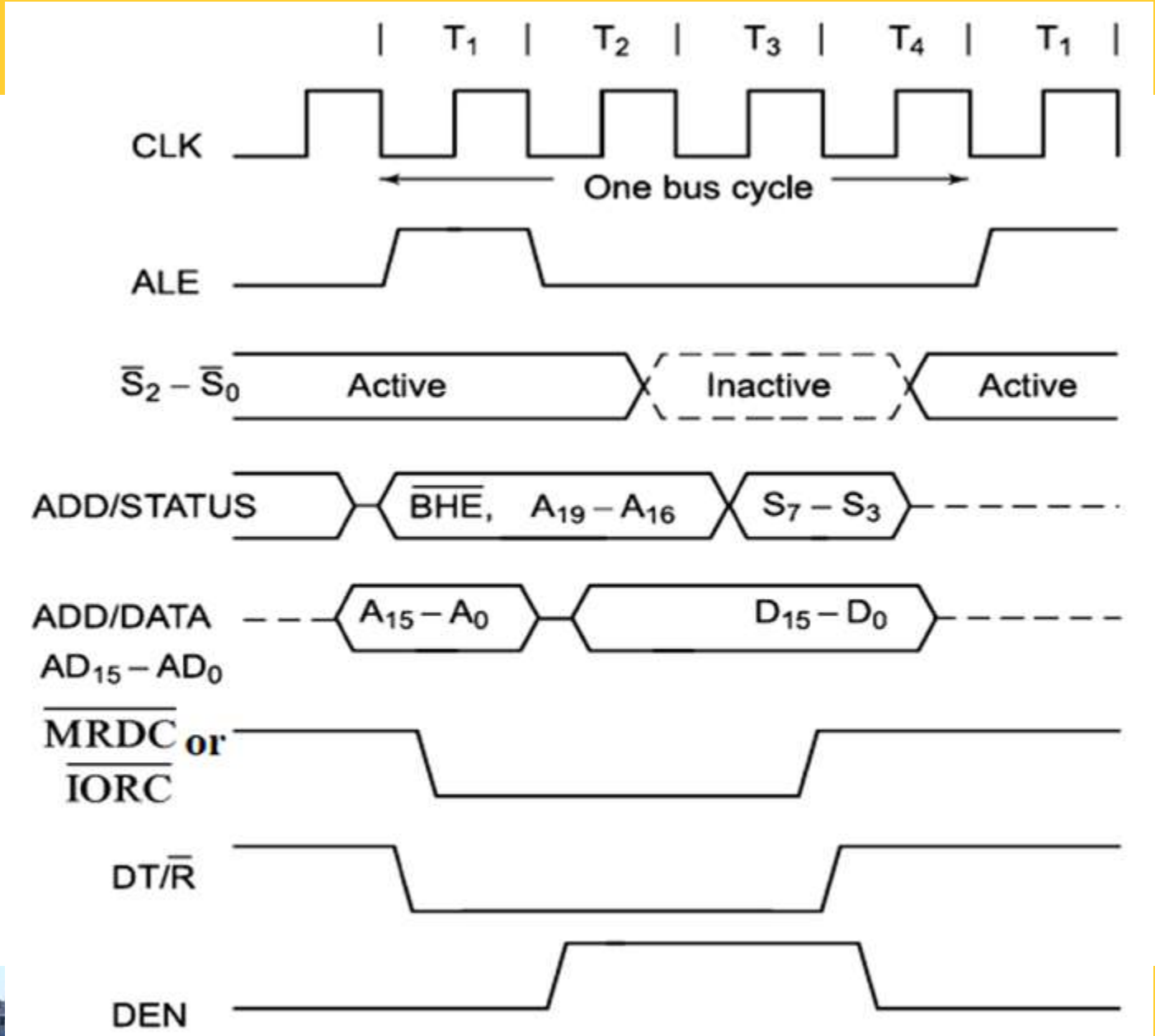


Fig. *Read Timing in Maximum Mode*

Maximum Mode 8086 System

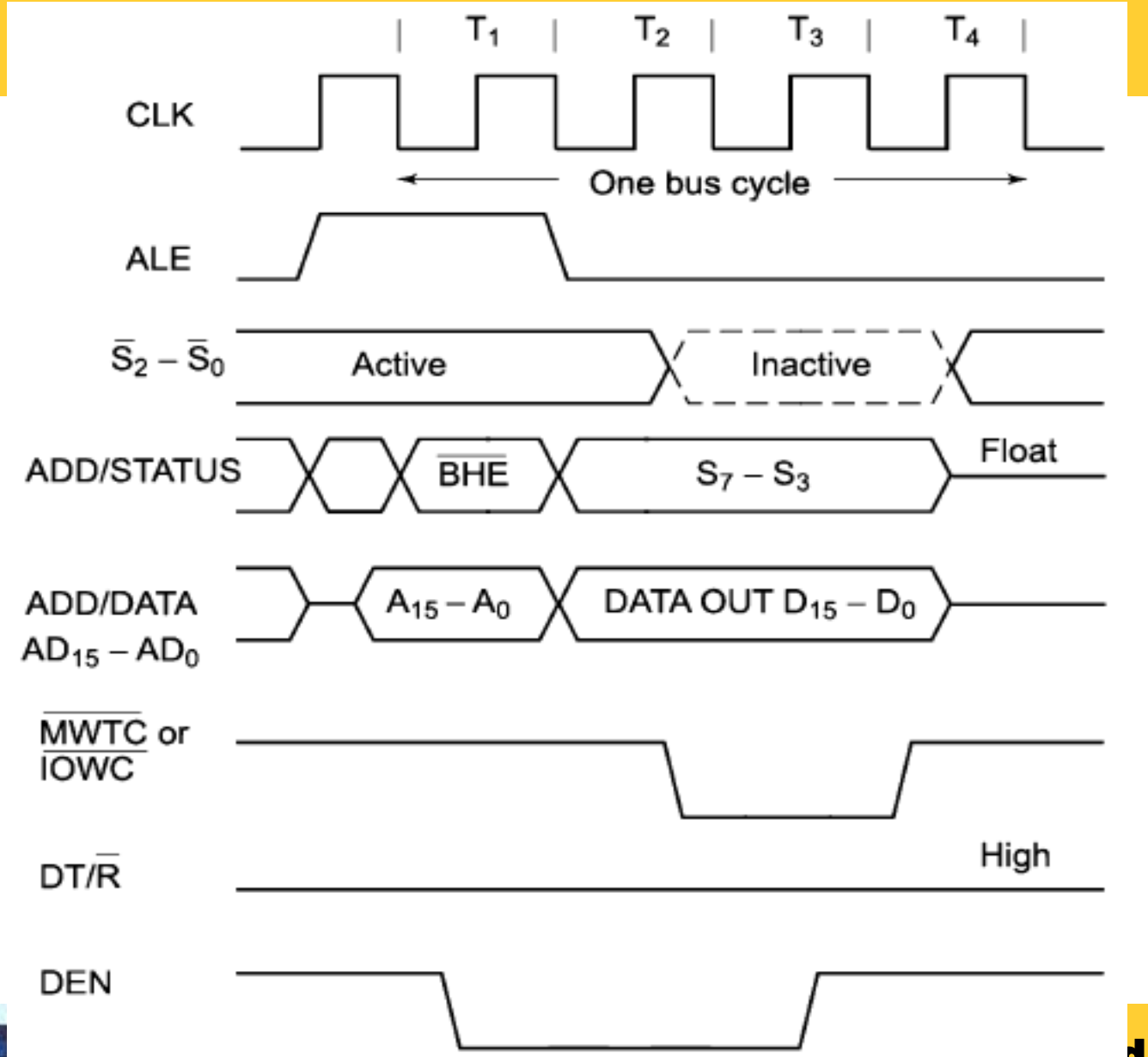


Fig. *Write Timing in Maximum Mode*

Maximum Mode 8086 System

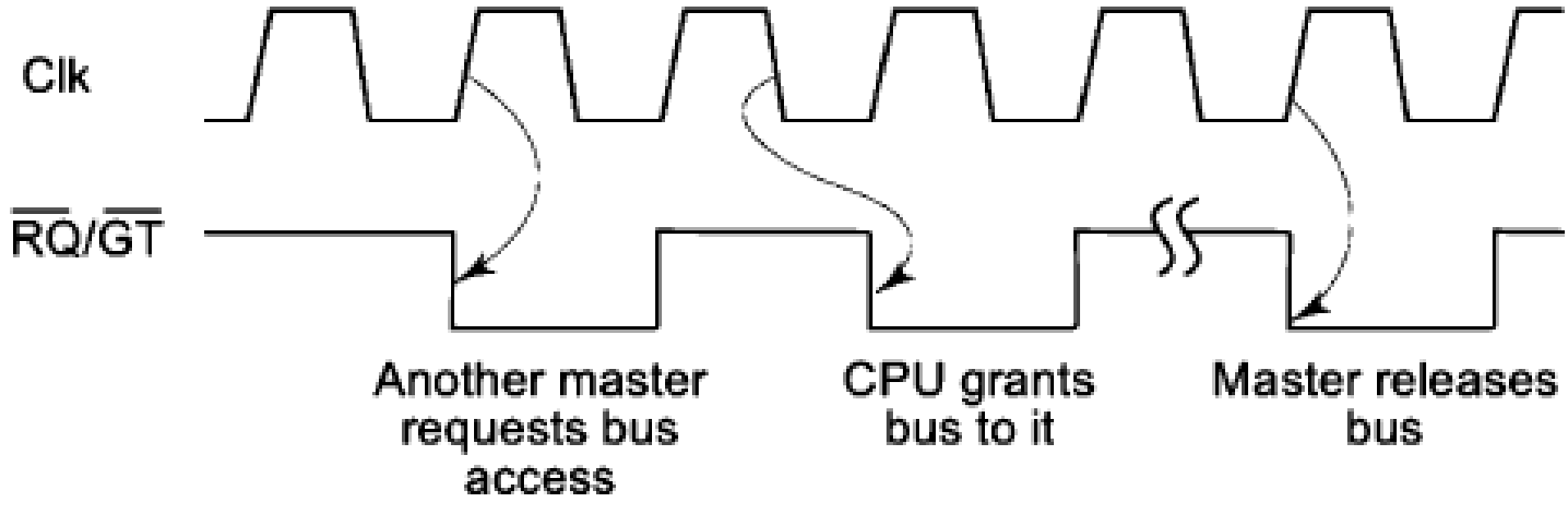


Fig. $\overline{RQ}/\overline{GT}$ *Timings in Maximum Mode*

INTERRUPT STRUCTURE OF 8086 and INTERRUPT VECTOR TABLE

‘INTERRUPT’ DEFINED

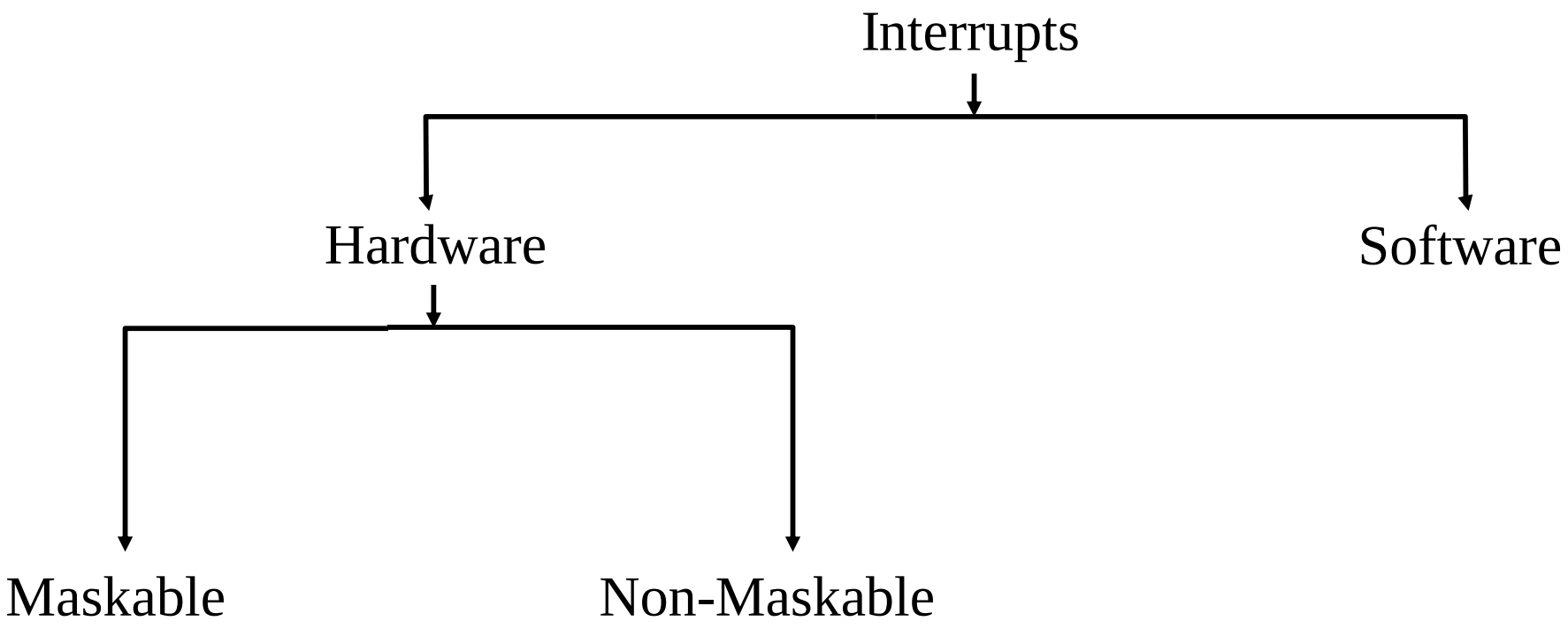
An interrupt is a mechanism by which an I/O device or an instruction can suspend the normal execution of processor and get itself serviced.

(or)

An interrupt is a signal to the processor generated by hardware or software indicating an event/a task/a device that needs processor service

ISR: *Generally, a particular task/service routine/small program is assigned to that interrupt signal and this program is called interrupt service routine (ISR).*

CLASSIFICATION OF INTERRUPTS



TASK HANDLING

Event/Task/Service routine can handle by two methods.

They are: 1) Polling method

2) Interrupt method

1) Polling method is continuous checking of condition or status of device by program or device to see what state they are in. 100% CPU Load

2) In interrupt method continuous checking of condition or status is not required because in this if condition or status met the required condition/state then automatically a signal is send to processor. <0.1% CPU Load

Waiting for an Event: Family Vacation

Polling

Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?

ula (V), Nandikotkur Road,
nkayapalli, KURNOOL

An engineering example...

Waiting for an Event: Family Vacation

Polling

Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?
Are we there yet?

Interrupts

Wake me up when we get there...

ula (V), Nandikotkur Road,
nkayapalli, KURNOOL

An engineering example...

8086 INTERRUPT SOURCES

An 8086 interrupt can come from any one of the following three sources:

a) *An external signal applied to the Non-Maskable Interrupt* ***(NMI)*** *pin* ***(or)*** *to the interrupt* ***(INTR)*** *pin. An interrupt caused by a signal applied to one of these inputs is called* ***hardware interrupt****.*

b) *The execution of the instruction INT n, where n is the interrupt type that can take any value between 00H and FFH. This is called* ***software interrupt****.*

c) *An error condition such as* ***divide-by-0****, which is produced in the 8086 by the execution of the* ***DIV/IDIV*** *instruction or the* ***trap interrupt****.*

8086 INTERRUPT PROCESSING

If an interrupt has been requested, the 8086 processes it by performing the following series of steps:

a) *Pushes the content of the flag register onto the stack to preserve the status of the program by decrementing the stack pointer (SP) by 2*

b) *Disables the INTR interrupt by clearing IF in the flag register*

c) *Resets TF in the flag register, to disable the single step or trap interrupt*

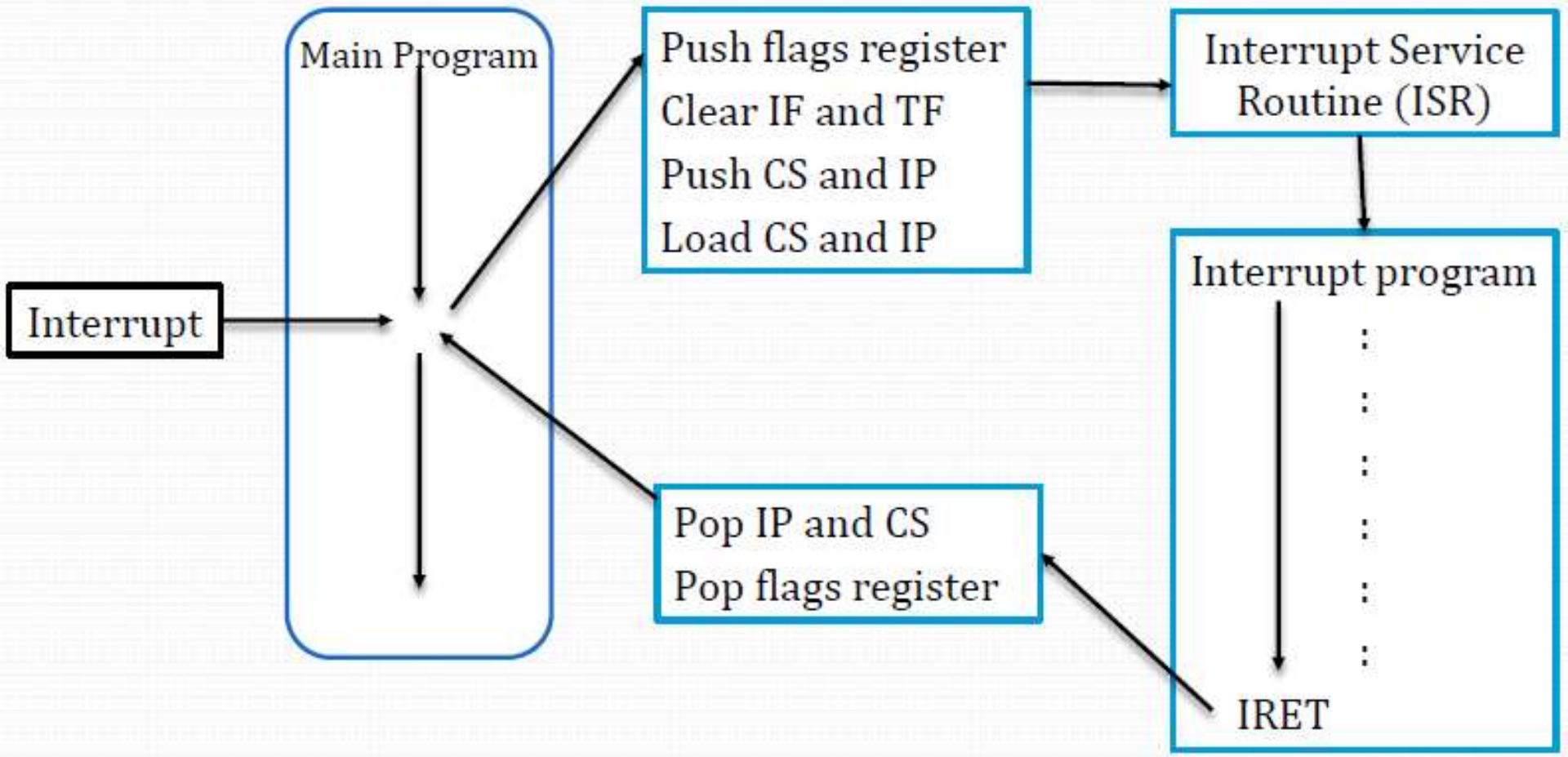
8086 INTERRUPT PROCESSING Cont....

d) *Pushes the content of the code segment (CS) register onto the stack by decrementing SP by 2*

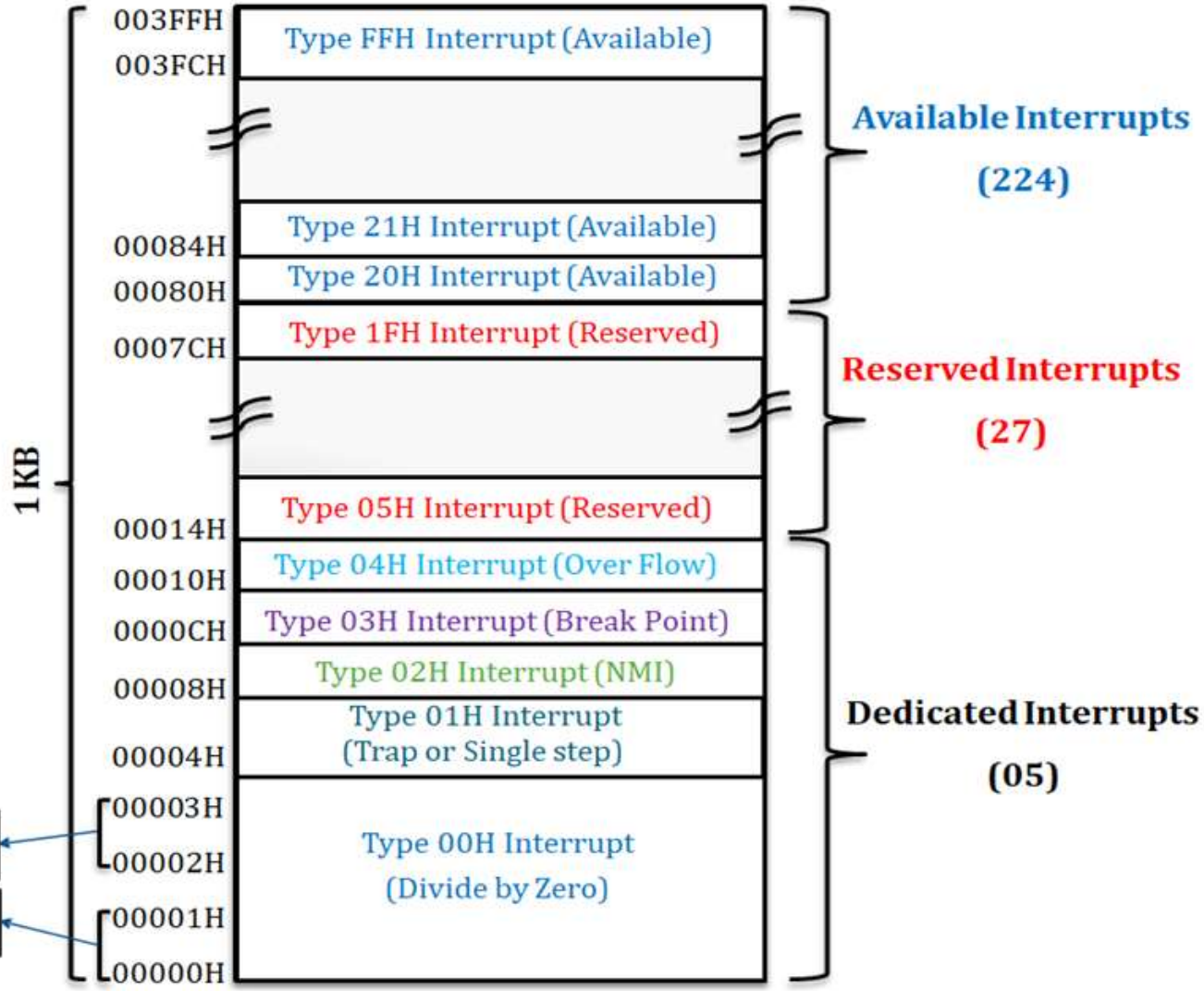
e) *Pushes the content of the instruction pointer (IP) onto the stack by decrementing SP by 2*

f) *Next Jumps to* ***ISR Location*** *by loading CS and IP with ISR address and CPU starts the execution of ISR*

INTERRUPT PROCESSING - 8086



INTERRUPT VECTOR TABLE - 8086



256 Interrupts Of 8086 are Divided in To 3 Groups

1) Type 00 to Type 04 interrupts

These are used for fixed operations and hence are called ***Dedicated interrupts***

2) Type 05 to Type 31 interrupts

These interrupts are ***Reserved interrupts***

3) Type 32 to Type 255 interrupts

These interrupts are ***User defined interrupts***

DEDICATED INTERRUPTS – INT 00 to INT 04

- *Type – 0 :- Divide by Zero Error Interrupt*

  Quotient is large cant be fit in al/ax or divide by zero

- *Type –1 :- Single step or Trap Interrupt*

  Used for executing the program in single step mode by setting trap flag.

- *Type – 2 :- Non-Maskable Interrupt*

  This interrupt is used for executing ISR of NMI pin (positive edge signal), NMI can't b masked by S/W.

- *Type – 3 :- One-byte INT instruction interrupt*

  Used for providing break points in the program

- *Type – 4 :- Over flow Interrupt*

Used to handle any overflow error after signed arithmetic.