# Intel 8086 – Architecture, Signals and Features

**Prepared by,**

**Dr. R. Manohari**

**Dr.T.Rajalakshmi**

# Definition of Computer

- A computer is an electronic machine that accepts information (Data), processes it according to specific instructions and provides the results as new information.

- Inputs, outputs, processes and ***stores*** information
- Physical:  Keyboard, monitor, etc. – are these necessary components?

# History of Computers - Long, Long Ago

## Abacus - 3000 BC

- beads on rods to count and calculate
- still widely used in Asia!

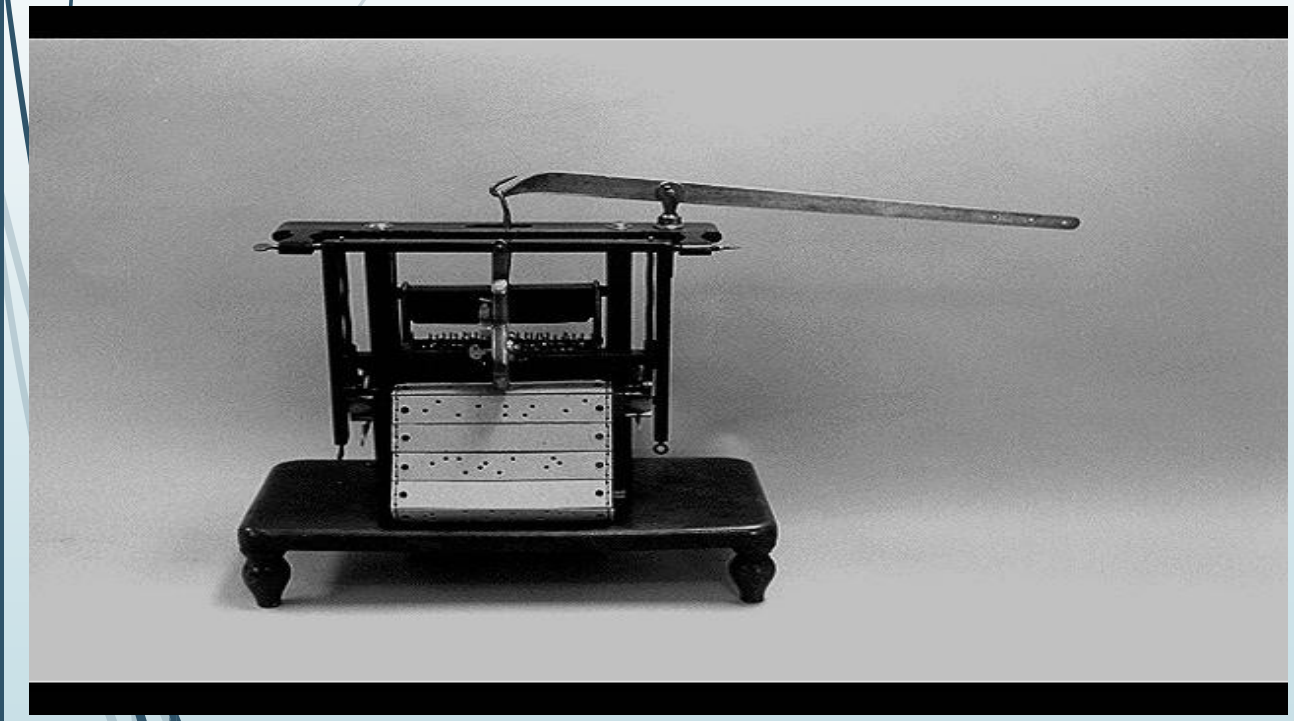
# History of Computers - Way Back When

## Slide Rule

- Slide Rule 1630
- based on Napier's rules for logarithms
- used until 1970s

# History of Computers - 19th Century

## Jacquard Loom - 1801

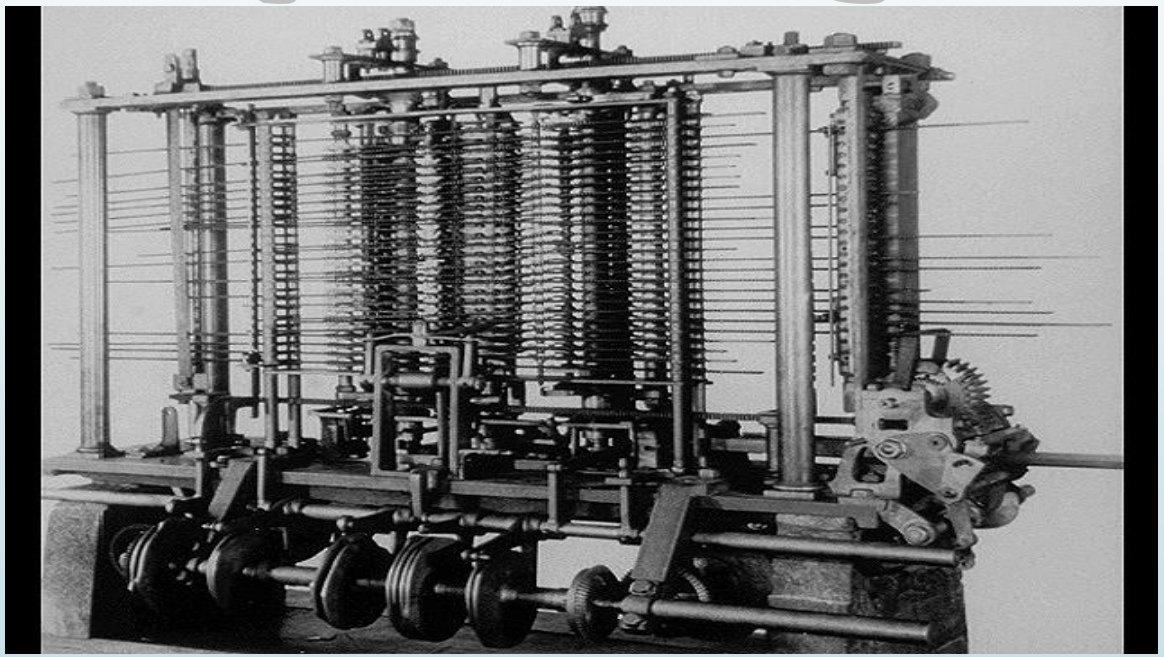


- first stored program - metal cards
- first computer manufacturing
- still in use today!

# Charles Babbage - 1792-1871

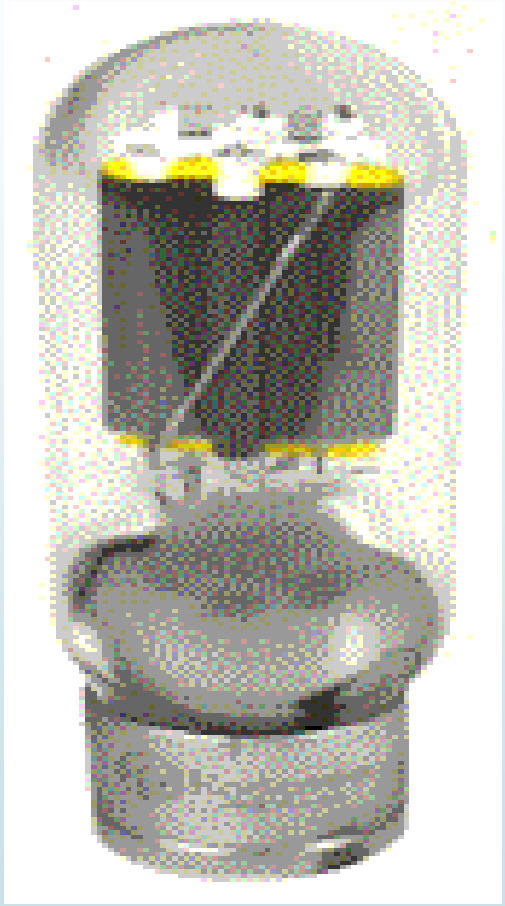
- Difference Engine c.1822
  - huge calculator, never finished
- Analytical Engine 1833
  - could store numbers
  - calculating "mill" used punched metal cards for instructions
  - powered by steam!
  - accurate to six decimal places

Analytical Engine



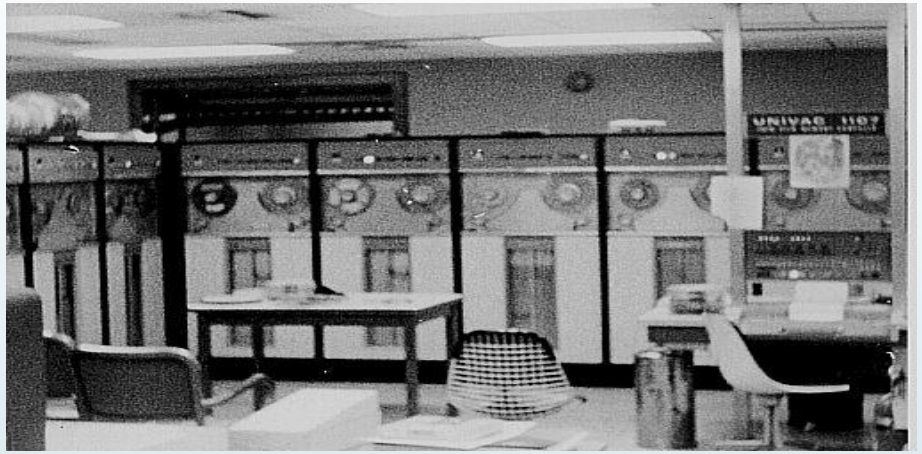
# Vacuum Tubes - 1941 - 1956

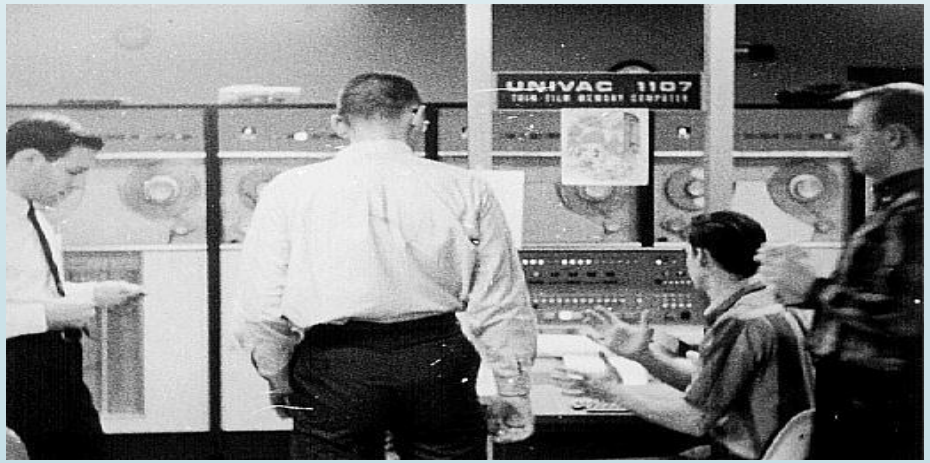
- **First Generation Electronic Computers** used Vacuum Tubes
- Vacuum tubes are glass tubes with circuits inside.
- Vacuum tubes have no air inside of them, which protects the circuitry.



# UNIVAC - 1951

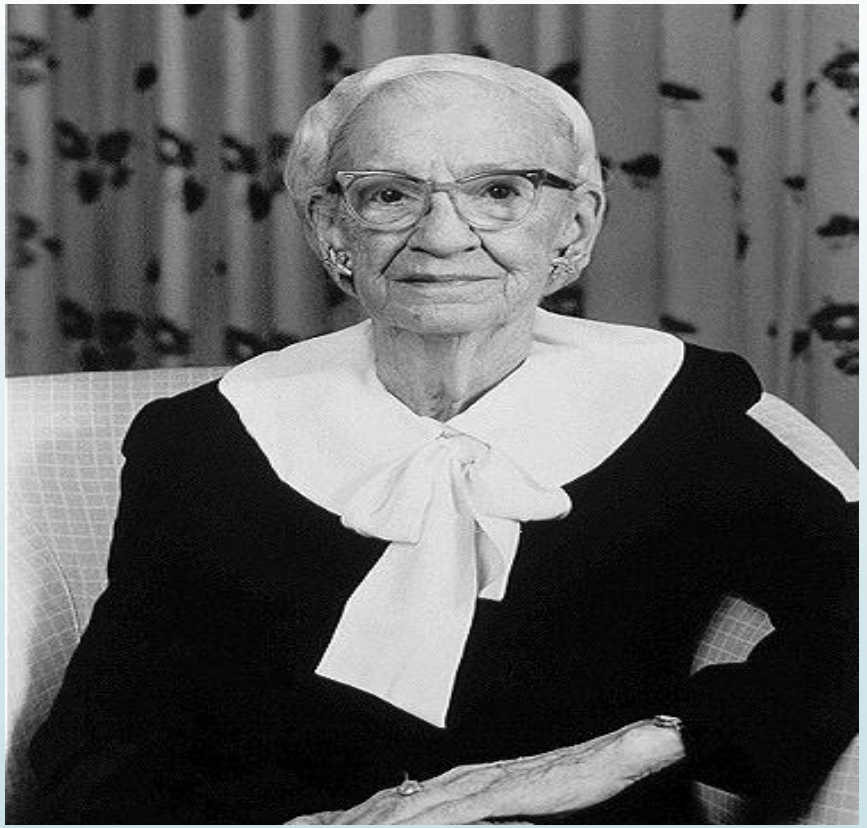
- first fully electronic **digital** computer built in the U.S.
- Created at the University of Pennsylvania
- ENIAC weighed 30 tons
- contained 18,000 vacuum tubes
- Cost a paltry $487,000



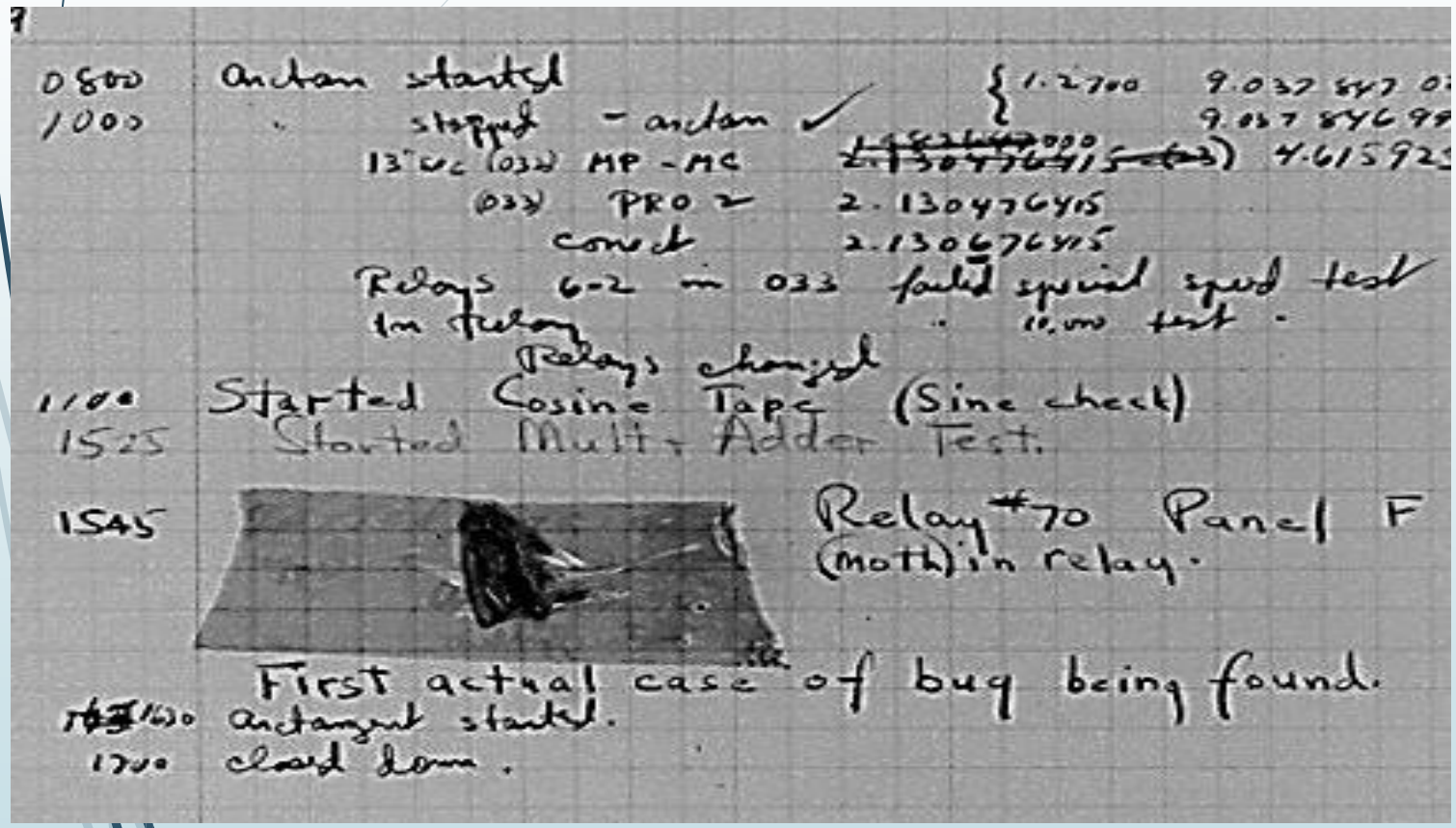


# Grace Hopper

- Programmed UNIVAC
- Recipient of Computer Science's first "Man of the Year Award"

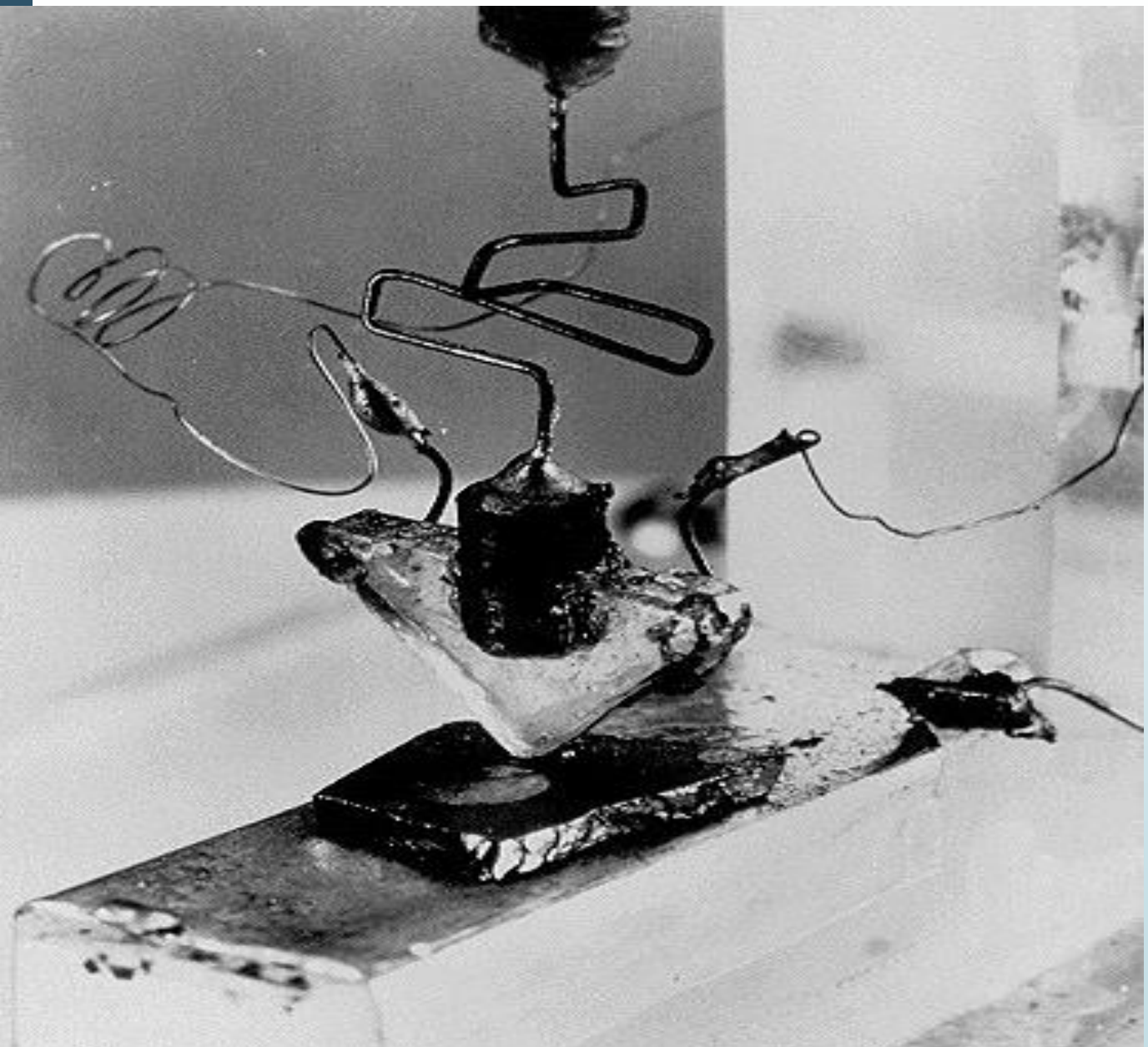


# First Computer Bug - 1945




- Relay switches part of computers
- Grace Hopper found a moth stuck in a relay responsible for a malfunction
- Called it "debugging" a computer

# First Transistor



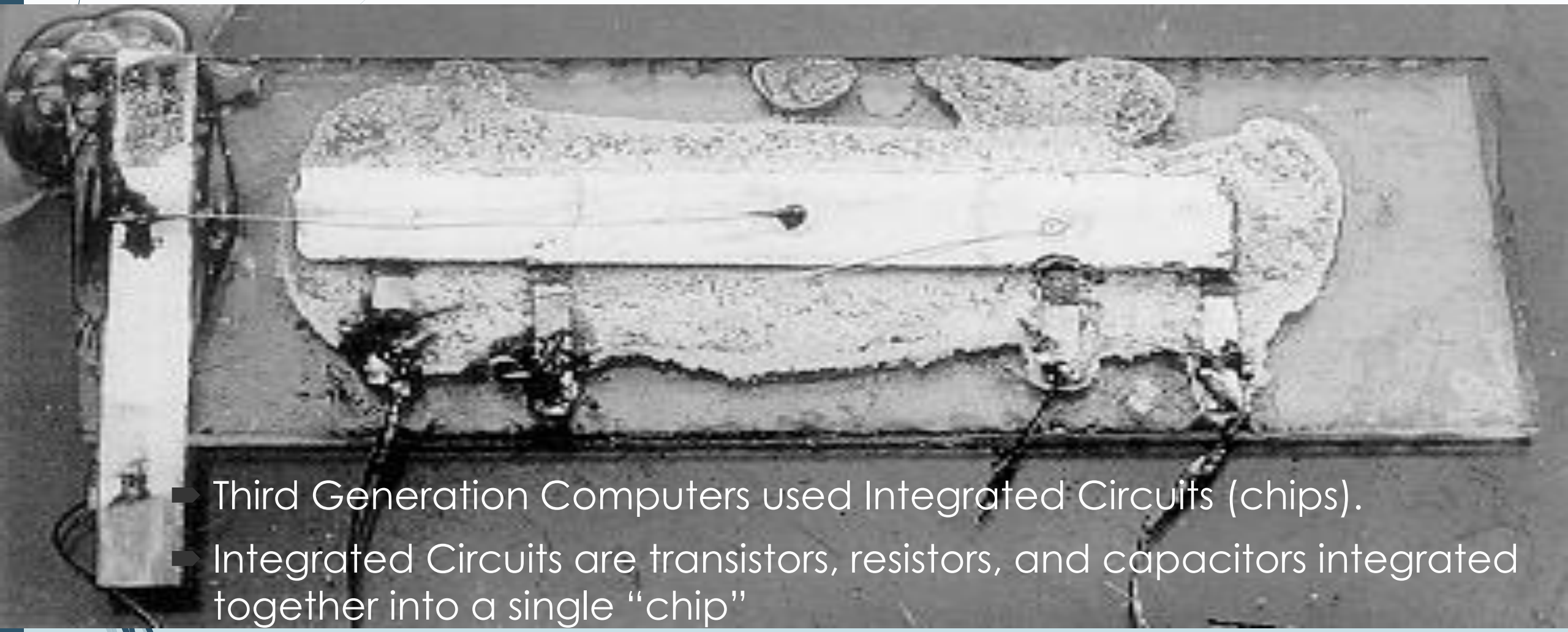
- Uses Silicon
- developed in 1948
- won a Nobel prize
- on-off switch

- Second Generation Computers used Transistors, starting in 1956

# Second Generation – 1956-1963

- 1956 – Computers began to incorporate ***Transistors***
- Replaced vacuum tubes with Transistors

# Integrated Circuits



- Third Generation Computers used Integrated Circuits (chips).
- Integrated Circuits are transistors, resistors, and capacitors integrated together into a single "chip"

# Operating System

- Software – Instructions for Computer
- Operating system is set of instructions loaded each time a computer is started
- Program is instructions loaded when needed

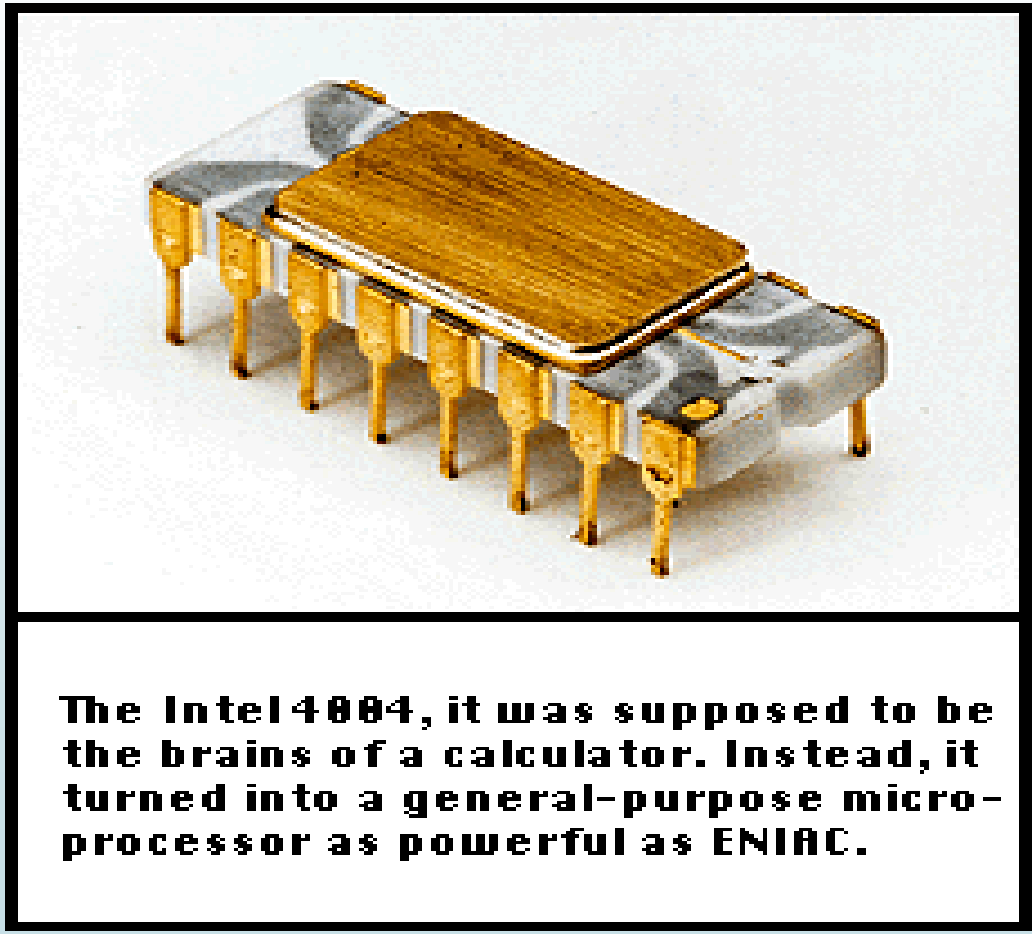
# Third Generation – 1964-1971

- 1964-1971
- Integrated Circuit
- Operating System
- Getting smaller, cheaper

# The First Microprocessor – 1971

## Intel 4004 Microprocessor

- The 4004 had 2,250 transistors
- four-bit chunks (four 1's or 0's)
- 108Khz
- Called "Microchip"



The Intel 4004, it was supposed to be the brains of a calculator. Instead, it turned into a general-purpose micro-processor as powerful as ENIAC.

# What is a Microchip?

- Very Large Scale Integrated Circuit (VLSIC)
  - Transistors, resistors, and capacitors
- 4004 had 2,250 transistors
- Pentium IV has 42 MILLION transistors
  - Each transistor 0.13 microns ($10^{-6}$ meters)

# 4th Generation – 1971-present

- MICROCHIPS!
- Getting smaller and smaller, but we are still using microchip technology

# Birth of Personal Computers - 1975

## MITS Altair

- 256 byte memory (not Kilobytes or Megabytes)
- 2 MHz Intel 8080 chips
- Just a box with flashing lights
- cost $395 kit, $495 assembled.

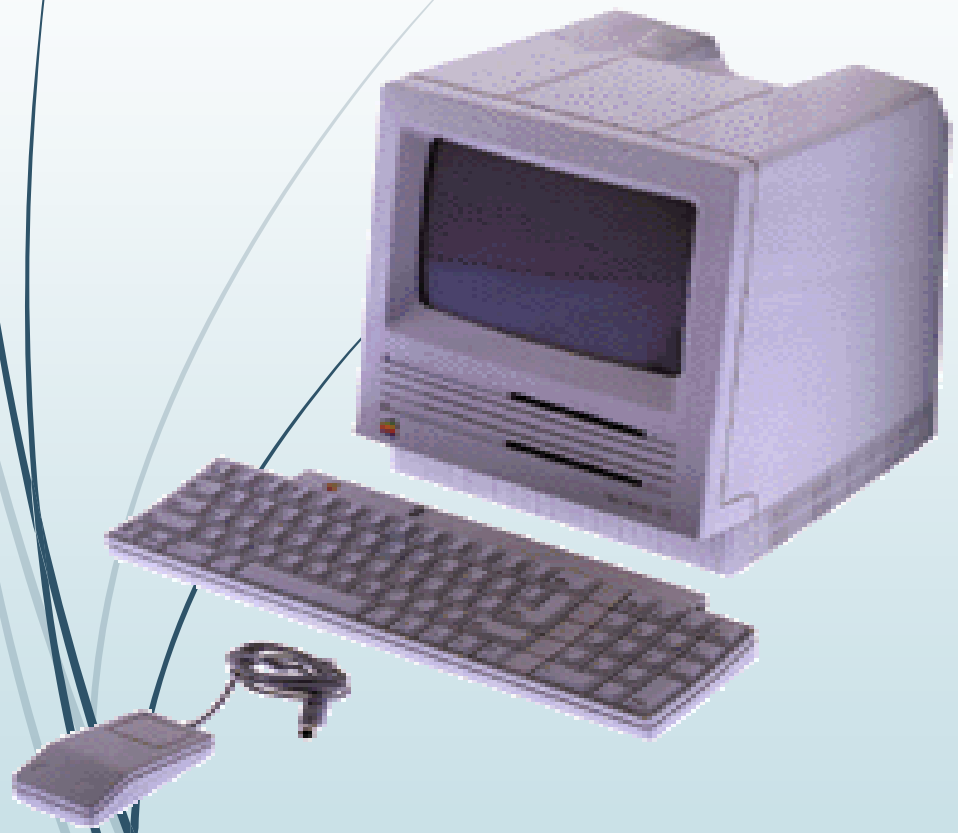
# Generations of Electronic Computers

| | **First Generation** | **Second Gen.** | **Third Gen.** | **Fourth Gen.** |
|---|---|---|---|---|
| **Technology** | Vacuum Tubes | Transistors | Integrated Circuits (multiple transistors) | Microchips (millions of transistors) |
| **Size** | Filled Whole Buildings | Filled half a room | Smaller | Tiny - Palm Pilot is as powerful as old building sized computer |

# IBM PC - 1981

- IBM-Intel-Microsoft joint venture
- First wide-selling personal computer used in business
- 8088 Microchip - 29,000 transistors
  - 4.77 Mhz processing speed
- 256 K RAM (Random Access Memory) standard
- One or two floppy disk drives

# Apple Computers



- Founded 1977
- Apple II released 1977
  - widely used in schools
- Macintosh (left)
  - released in 1984, Motorola 68000 Microchip processor
  - first commercial computer with graphical user interface (GUI) and pointing device (mouse)

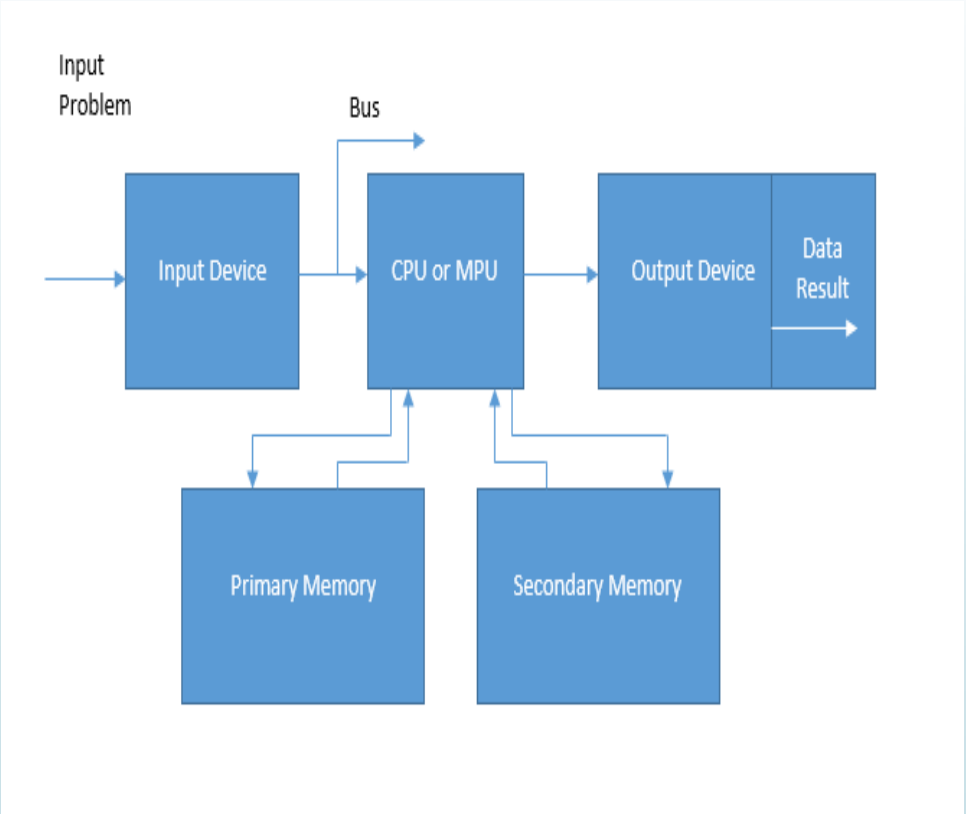
# Computers Progress

| | UNIVAC (1951-1970) (1968 vers.) | Mits Altair (1975) | IBM PC (1981) | Macintosh (1984) | Pentium IV |
|---|---|---|---|---|---|
| Circuits | Integrated Circuits | 2 Intel 8080 Microchip | Intel 8088 Microchip - 29,000 Transistors | Motorola 68000 | Intel P-IV Microchip - 7.5 million transistors |
| RAM Memory | 512 K | 265 Bytes | 256 KB | | 256 MB |
| Speed | 1.3 MHz | 2 KHz | 4.77 MHz | | 3200 MHz = 3.2 GHz |
| Storage | 100 MB Hard Drive | 8” Floppy Drive | Floppy Drive | Floppy Drives | Hard Drive, Floppy, CD-Rom |
| Size | Whole Room | Briefcase (no monitor) | Briefcase + Monitor | Two shoeboxes (integrated monitor) | Small Tower |
| Cost | $1.6 million | $750 | $1595 | ~$4000 | $1000 - $2000 |

# 1990s: Pentiums and Power Macs

- Early 1990s began penetration of computers into every niche: every desk, most homes, etc.
- Faster, less expensive computers paved way for this
- Windows 95 was first decent GUI for “PCs”
- Macs became more PC compatible - easy file transfers
- Prices have plummeted
  - $2000 for entry level to $500
  - $6000 for top of line to $1500

# Block Diagram of Microcomputer




- Block diagram of a microcomputer consist of
- **Input Device** for receiving input instruction.
- **CPU** or **MPU** for processing the instruction.
- Primary and Secondary Memory to store the information.
- **Output Device** for giving the output.
- **Data Result**

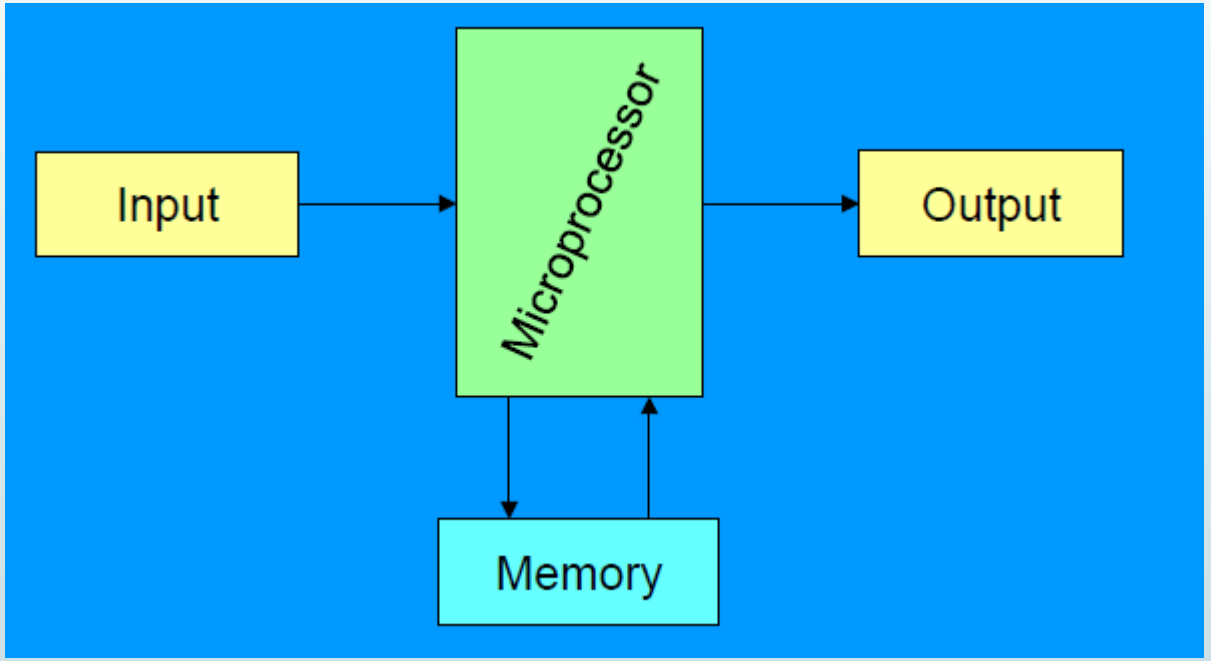
# Microprocessor Definition

- Microprocessor is a programmable device that takes in numbers, performs on them arithmetic or logical operations according to the program stored in memory and then produces other numbers as a result.

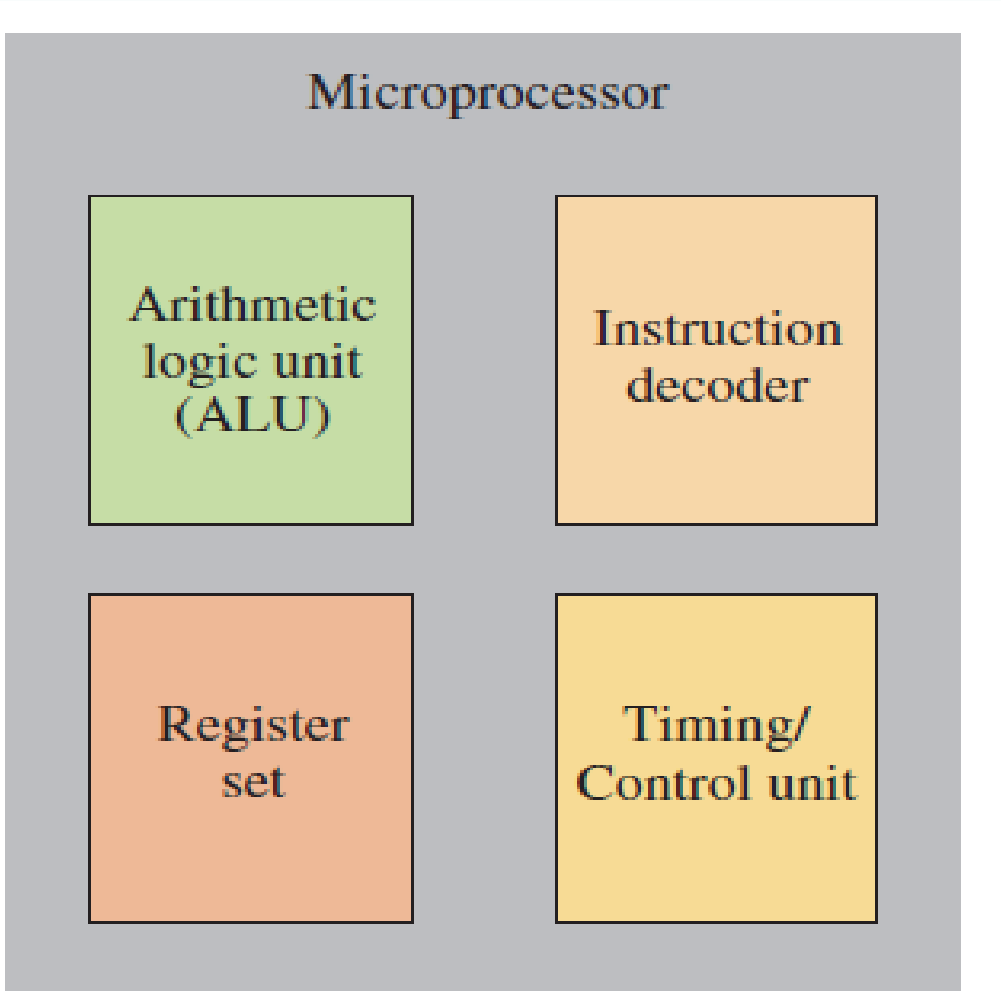
# Basic Concepts of Microprocessors

- Differences between:
  - Microcomputer –a computer with a microprocessor as its CPU. Includes memory, I/O etc.
  - Microprocessor –silicon chip which includes ALU, register circuits, instruction decoder circuits & control circuits
  - Microcontroller –silicon chip which includes microprocessor, memory & I/O in a single package

# A Microprocessor based system (fig 1)



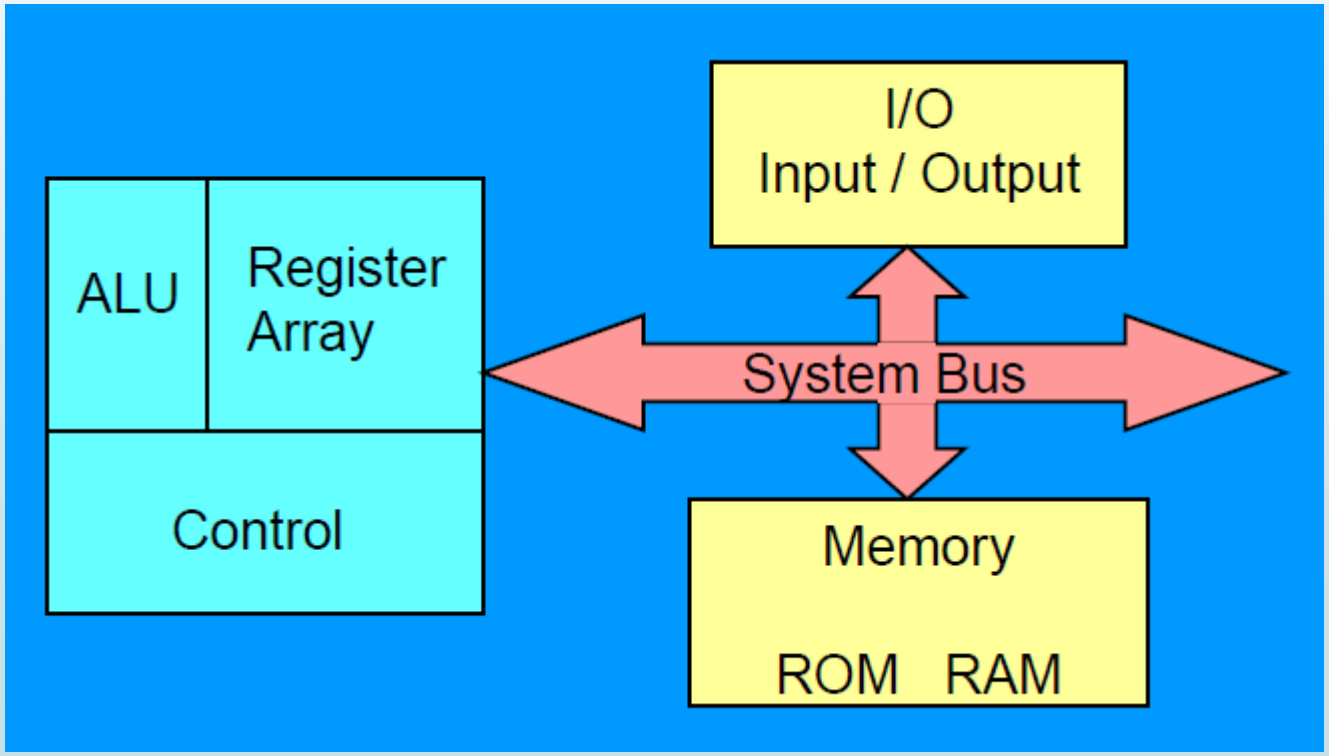
# Inside the Microprocessor




Elements of a microprocessor (CPU)

# Organization of a microprocessor-based system

- Let's expand the fig 1 a bit



**Fifth Generation** **Pentium**

**Fourth Generation**

During 1980s

Low power version of HMOS technology (HCMOS)

32 bit processors

Physical memory space $2^{24}$ bytes = 16 Mb

Virtual memory space $2^{40}$ bytes = 1 Tb

Floating point hardware

Supports increased number of addressing modes

**Intel 80386**

**Third Generation**

During 1978

HMOS technology ⇒ Faster speed, Higher packing density

16 bit processors ⇒ 40/ 48/ 64 pins

Easier to program

Dynamically relatable programs

Processor has multiply/ divide arithmetic hardware

More powerful interrupt handling capabilities

Flexible I/O port addressing

**Intel 8086** (16 bit processor)

**Second Generation**

During 1973

NMOS technology ⇒ Faster speed, Higher density, Compatible with TTL

4 / 8/ 16 bit processors ⇒ 40 pins

Ability to address large memory spaces and I/O ports

Greater number of levels of subroutine nesting

Better interrupt handling capabilities

**Intel 8085** (8 bit processor)

**First Generation**

Between 1971 – 1973

PMOS technology, non compatible with TTL

4 bit processors ⇒ 16 pins

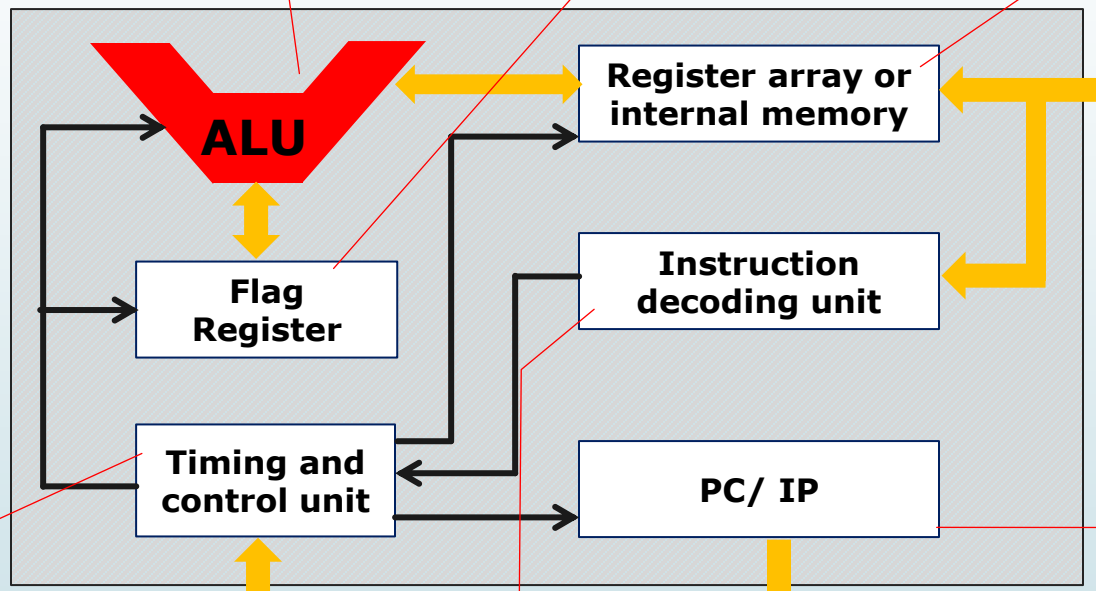
8 and 16 bit processors ⇒ 40 pins

Due to limitations of pins, signals are multiplexed

Computational Unit; performs arithmetic and logic operations

Various conditions of the results are stored as status bits called flags in flag register

Internal storage of data



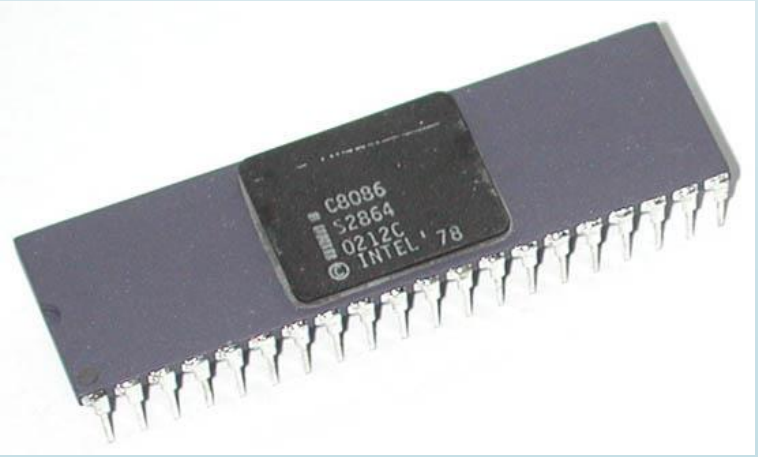

Generates the address of the instructions to be fetched from the memory and send through address bus to the memory

Generates control signals for internal and external operations of the microprocessor

Decodes instructions; sends information to the timing and control unit

# INTEL 8086

- INTEL 8086 is the first 16-bit processor released by INTEL in the year 1978
- Third Generation Microprocessors
- Designed using HMOS technology & now it is manufactured using HMOS III technology & contains approximately 29,000 transistors.
- It is available as 40-pin Dual in-line Package (DIP).



# INTEL 8086

- It is available in four versions:

1. 8086 (5 MHz)
2. 8086-2 (8 MHz)
3. 8086-4 (4 MHz)
4. 8086-1 (10 MHz)

# Main Features of 8086

- It is a 16-bit processor. It's ALU, internal registers work with 16-bit binary words.
- It has a 16 bit data bus. It can read (or write) data from (or to) a memory/port, either 16 bits or 8 bits at a time.
- It has a 20-bit address bus which means it can address up to $2^{20}$ = 1 MB memory location.
- Like 8085, 8086 too can do only fixed arithmetic.
- ✓ Intel had designed the coprocessor 8087 that can do floating point arithmetic & other complex mathematical operations.
- 8086 can work in conjunction with 8087 to do both fixed-point, floating point & other complex mathematical functions.

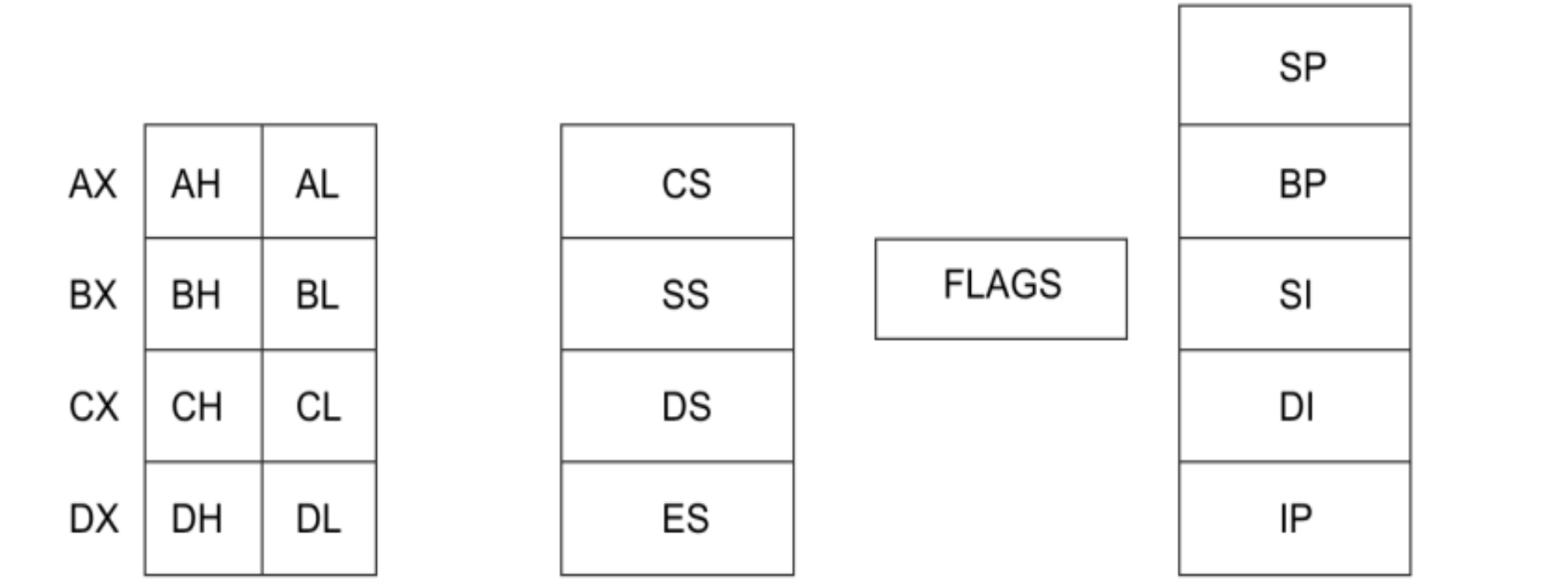
# Main Features of 8086 cont…

- It is designed to operate in two modes: minimum mode and maximum mode
- ✓ In minimum mode, 8086 processor works in a single processor environment
- ✓ In maximum mode, it works in a multiprocessor environment.
- ✓ In a multiprocessor environment, Control signals for memory & I/O are generated by an external BUS controller
- It requires +5 V power supply.
- It uses a 40 pin DIP

# Main Features of 8086 cont…

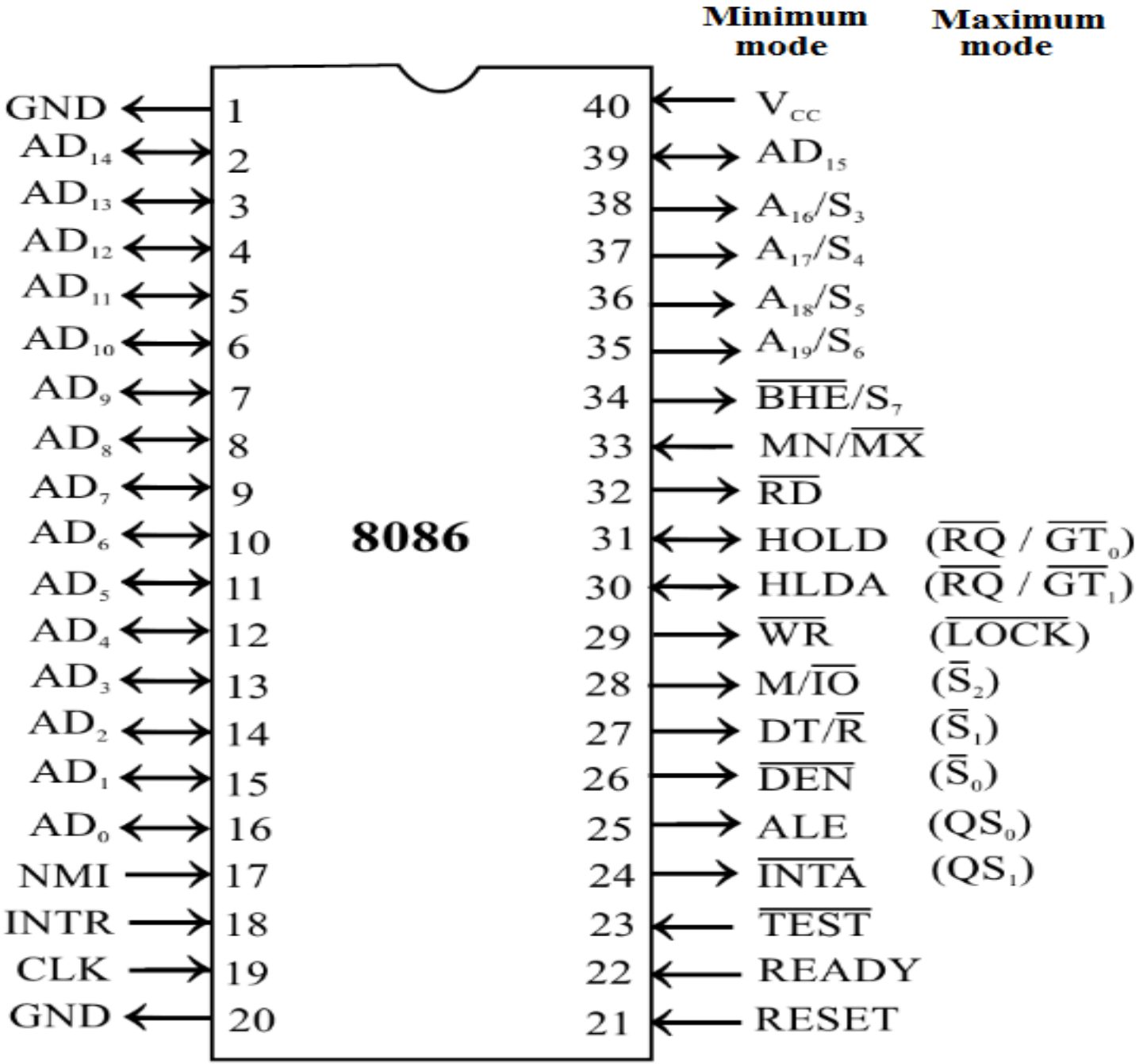
- It can pre-fetch up to six instruction bytes from memory and queues them in order to speed up instruction execution.
- 8086 has two blocks: BIU (Bus Interface Unit) & EU (Execution Unit)

# Register Organization of 8086




*Register organisation of 8086*

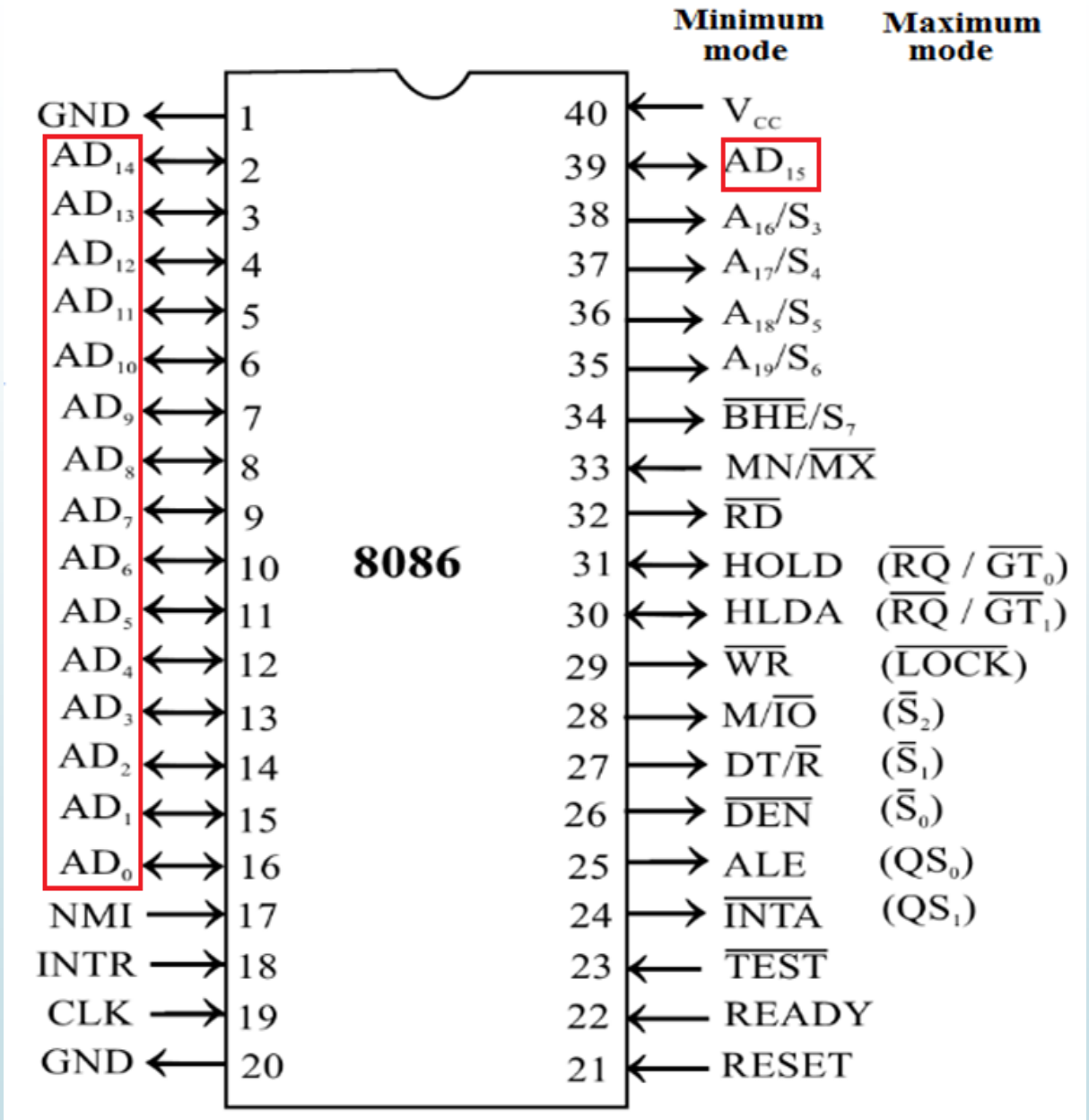
# Pin Diagram of 8086 Microprocessor




8086 pin assignments

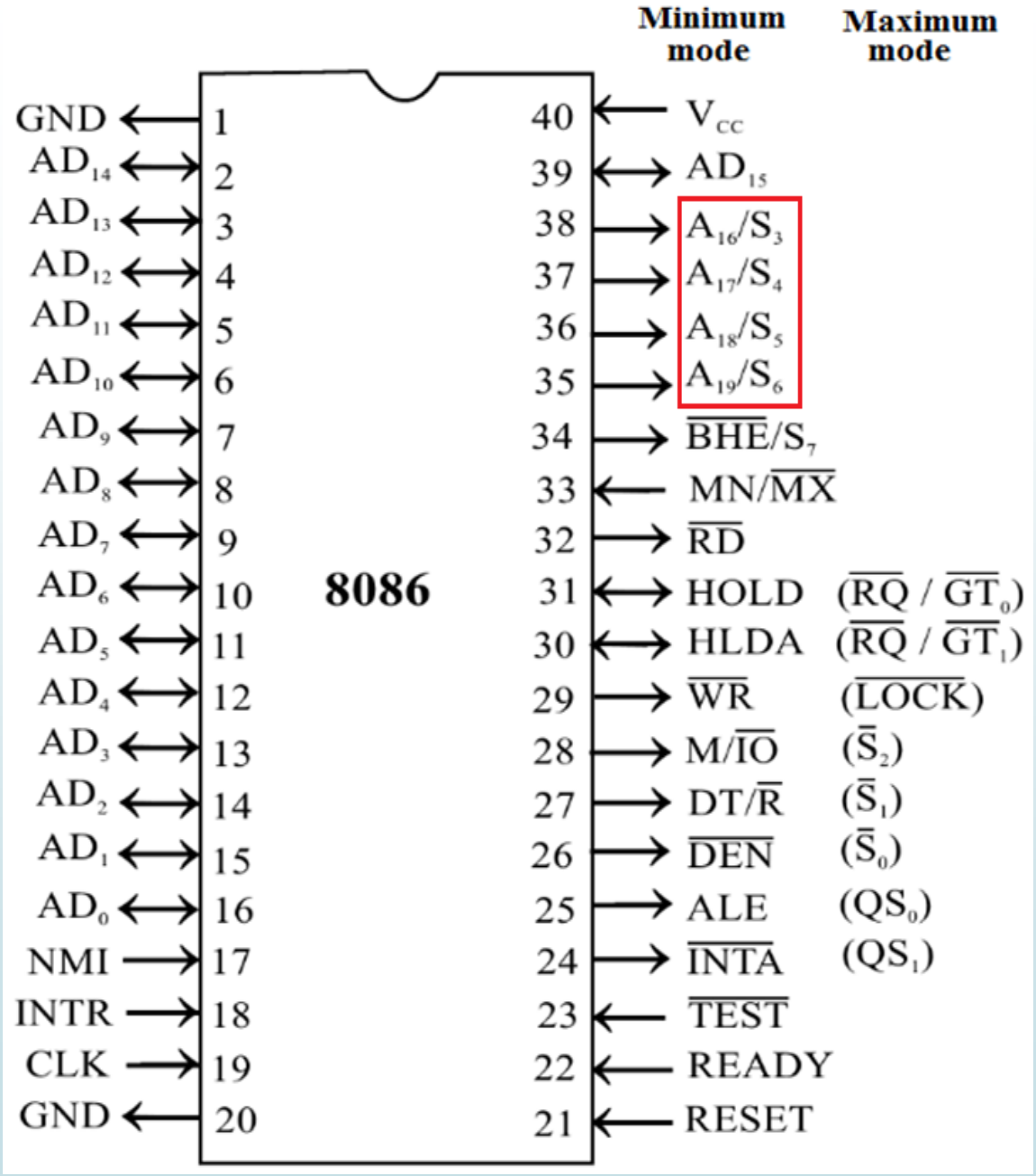
# $AD_0 – AD_{15}$ Pin 16-2, 39 (Bi-directional)

- These lines constitute the time multiplexed memory/IO address & data bus.
- It acts as address bus during the first part of machine cycle($T_1$) & data bus for the remaining part of the machine cycle



# $A_{19}/S_6$, $A_{18}/S_5$, $A_{17}/S_4$, $A_{16}/S_3$
## Pin 35-38 (Unidirectional)

- These are the time multiplexed address and status lines
- During $T_1$, they carry higher order 4-bit address.
- In the remaining clock cycles, they carry status signals.



# Cont.

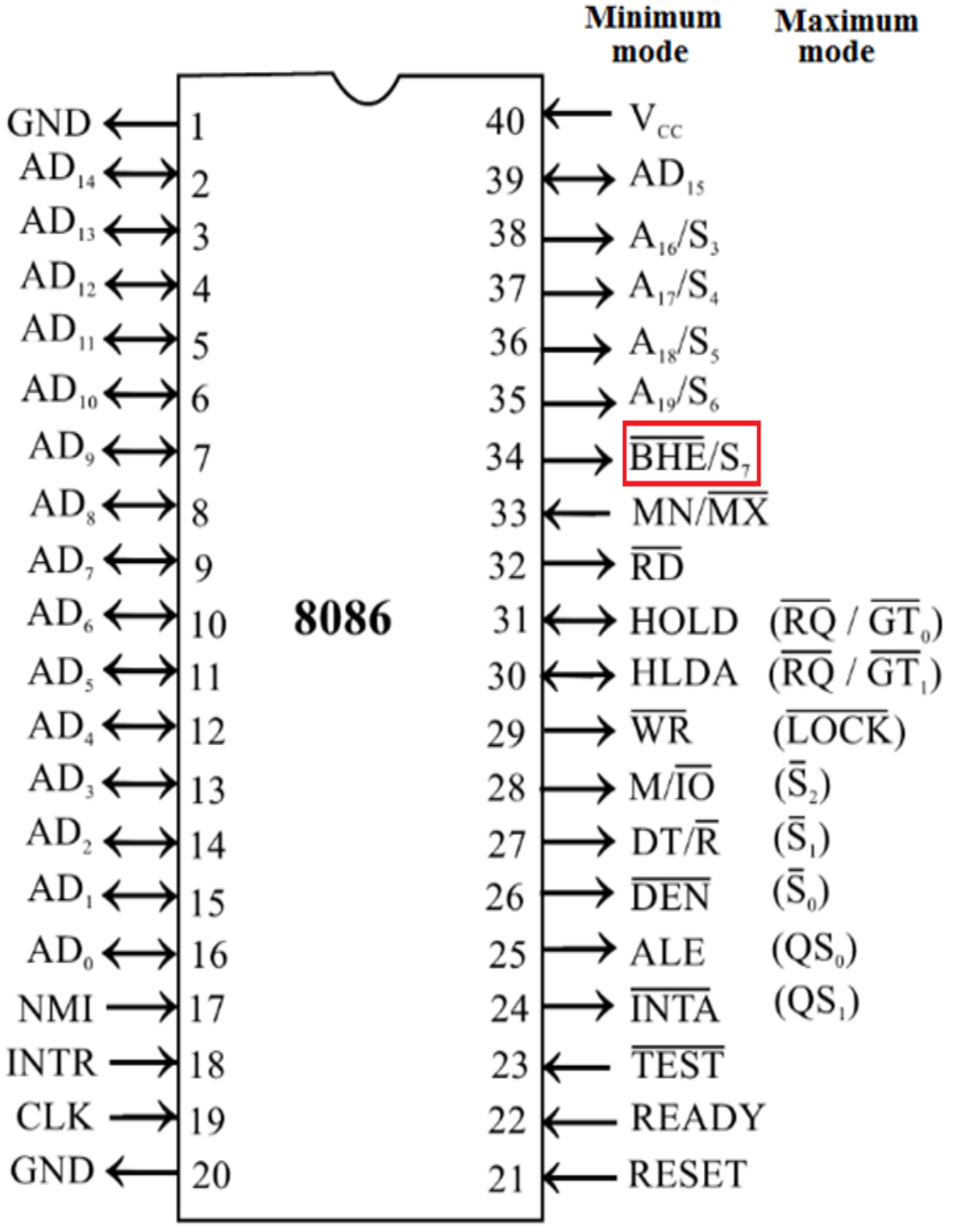
- The status signals on $S_3$ & $S_4$ specifies the segment register used for calculating the physical address
- The status line $S_5$ indicates the status of an 8086 interrupt enable flag
- The status line $S_6$ is always low

**STATUS SIGNAL DURING MEMORY SEGMENT ACCESS**

| Status signal | | Segment register |
|---|---|---|
| $S_4$ | $S_3$ | |
| 0 | 0 | Extra segment |
| 0 | 1 | Stack segment |
| 1 | 0 | Code or no segment |
| 1 | 1 | Data segment |

# $\overline{BHE}$ / $S_7$
# Pin 34 (Output)

- BHE stands for Bus High Enable.
- $\overline{BHE}$ signal is used to indicate the transfer of data over higher order data bus ($D_{15} – D_8$).
- It is multiplexed with status pin $S_7$.
- $S_7$ is not currently used



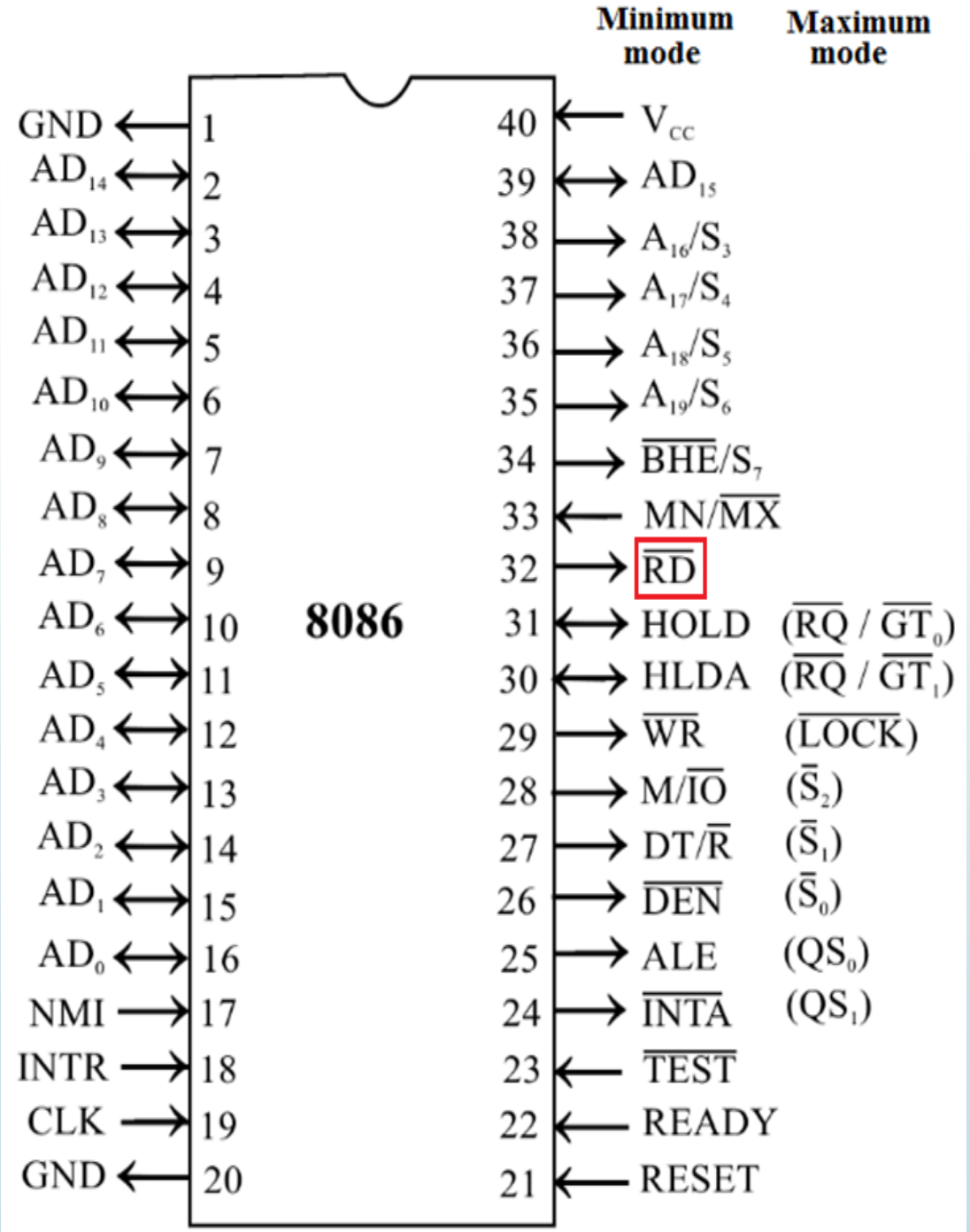
# Cont.

**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 |

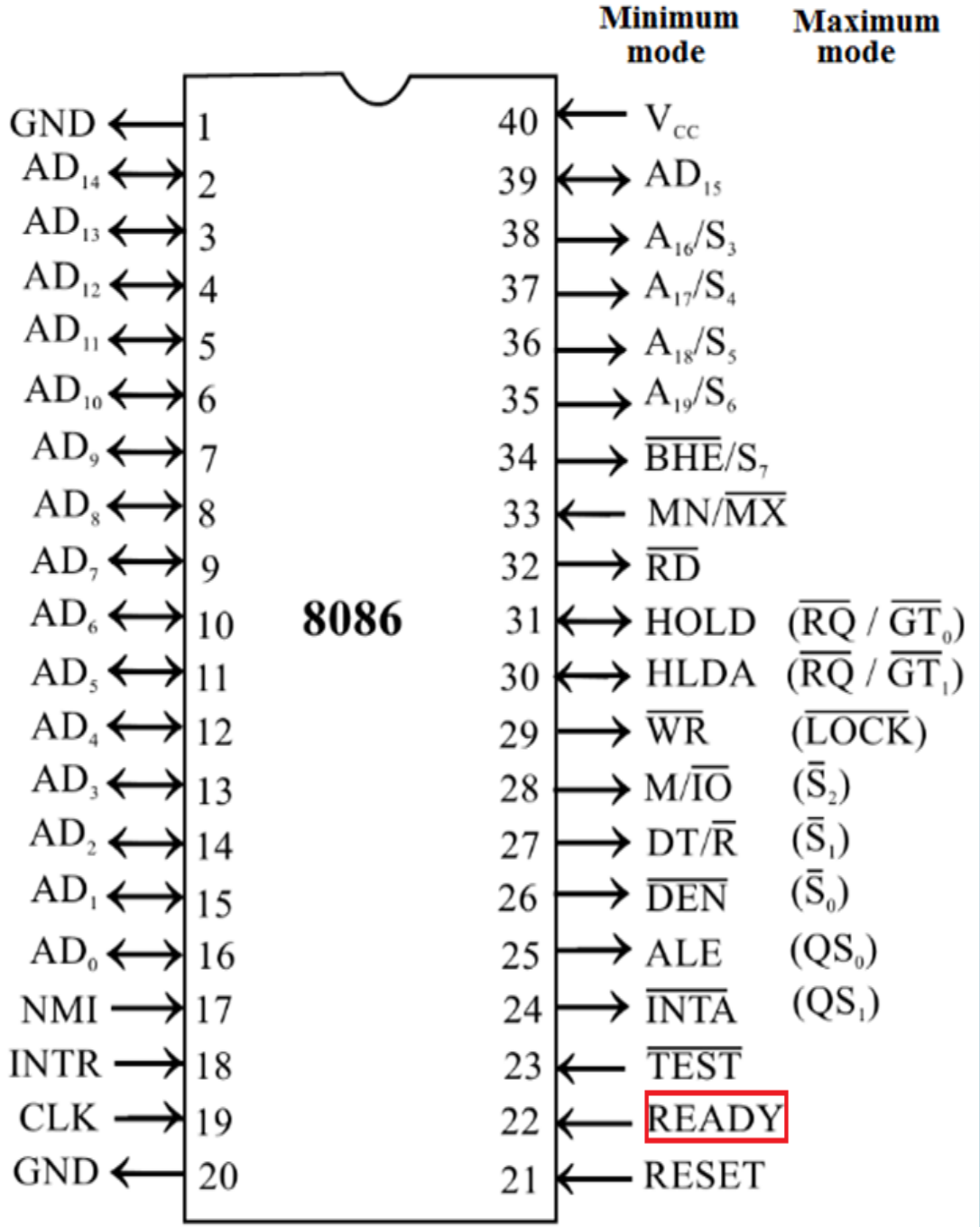
# $\overline{RD}$ (Read) Pin 32 (Output)

- It is a read signal used for read operation.
- $\overline{RD}$ is low whenever the 8086 is reading data from memory or an I/O device



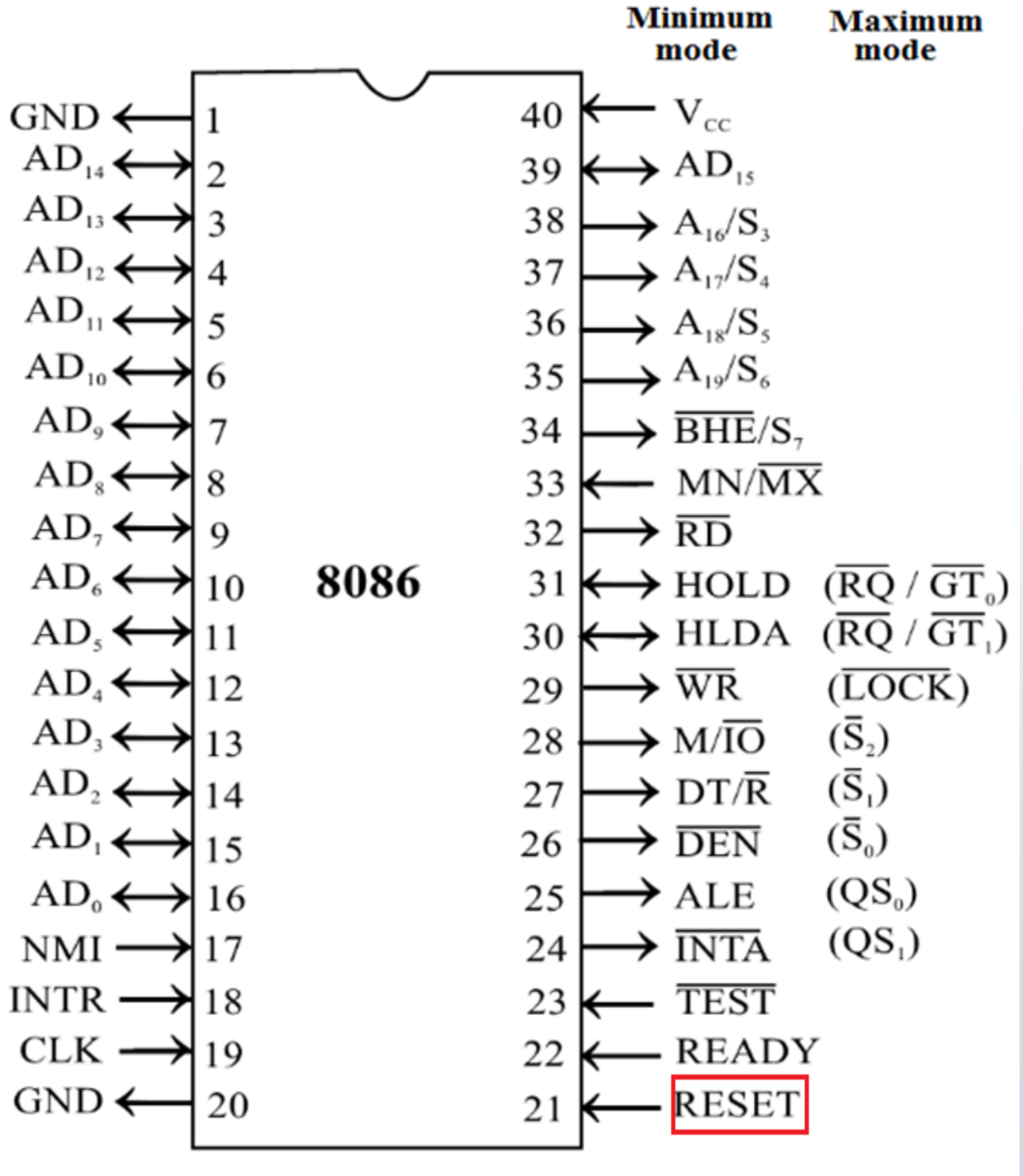
# READY
## Pin 22 (Input)

- This is an acknowledgement signal from slower I/O devices or memory.
- When high, it indicates that the device is ready to transfer data.
- When low, then microprocessor enters into wait state
- This signal is used primarily to synchronize slower peripherals with the microprocessor



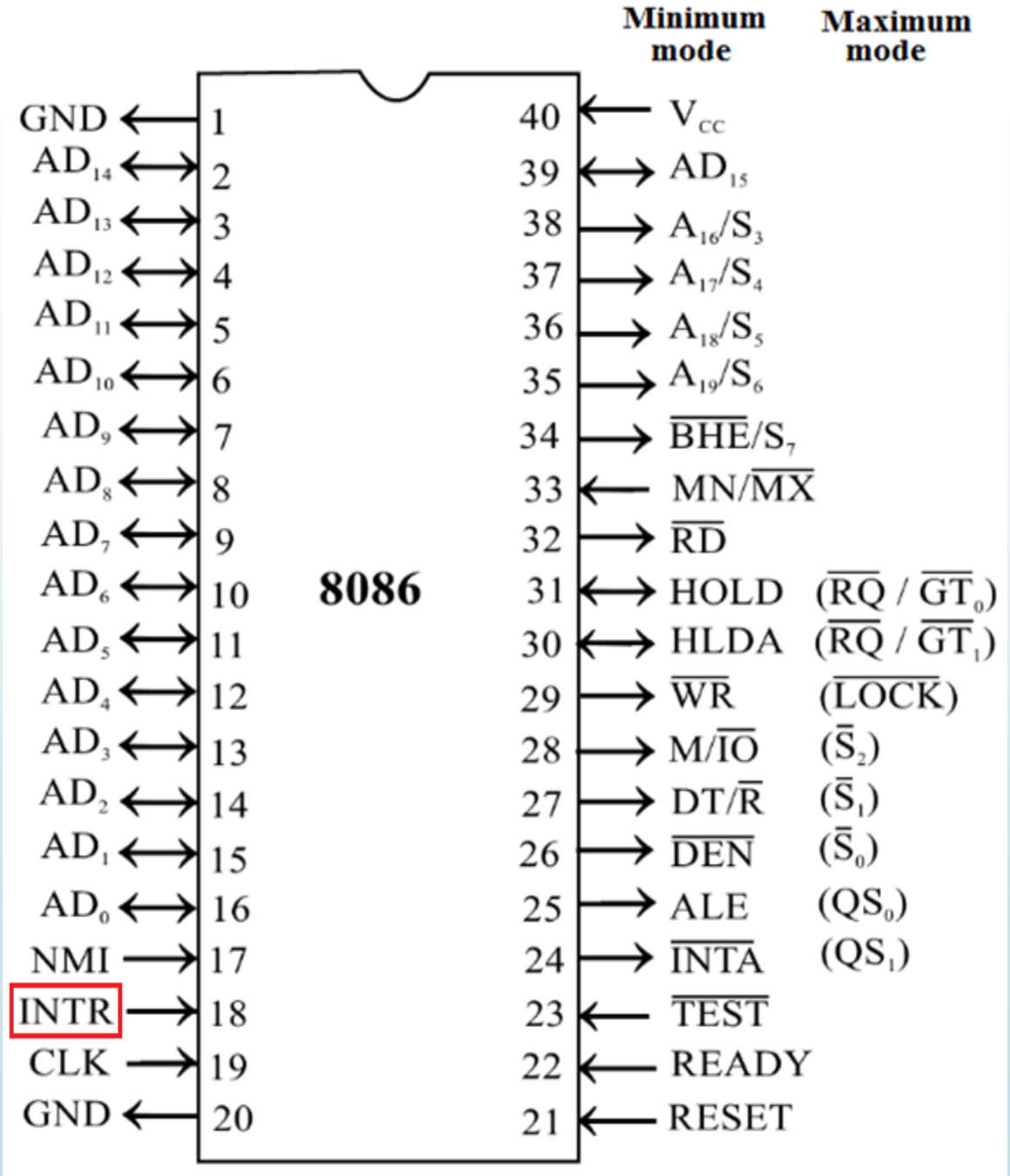
# RESET
## Pin 21 (Input)

- It is a system reset.
- It is an active high signal.
- When high, microprocessor enters into reset state and terminates the current activity and start execution from FFFF0H.
- It must be active for at least four clock cycles to reset the microprocessor.



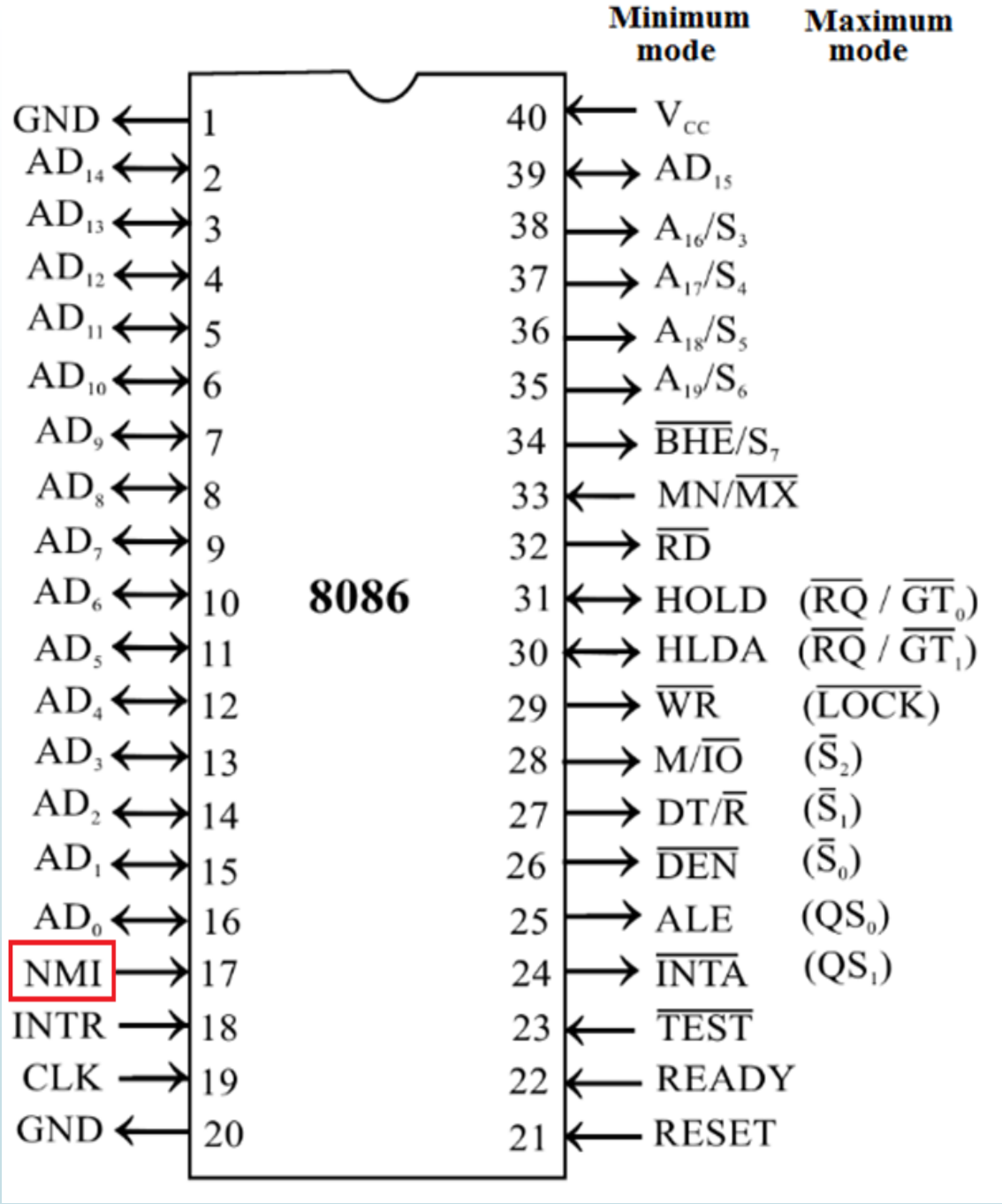
# INTR
## Pin 18 (Input)

- It is a level triggered maskable interrupt request.
- It is sampled during the last clock cycle of each instruction to determine if the processor should enter into an interrupt service routine.



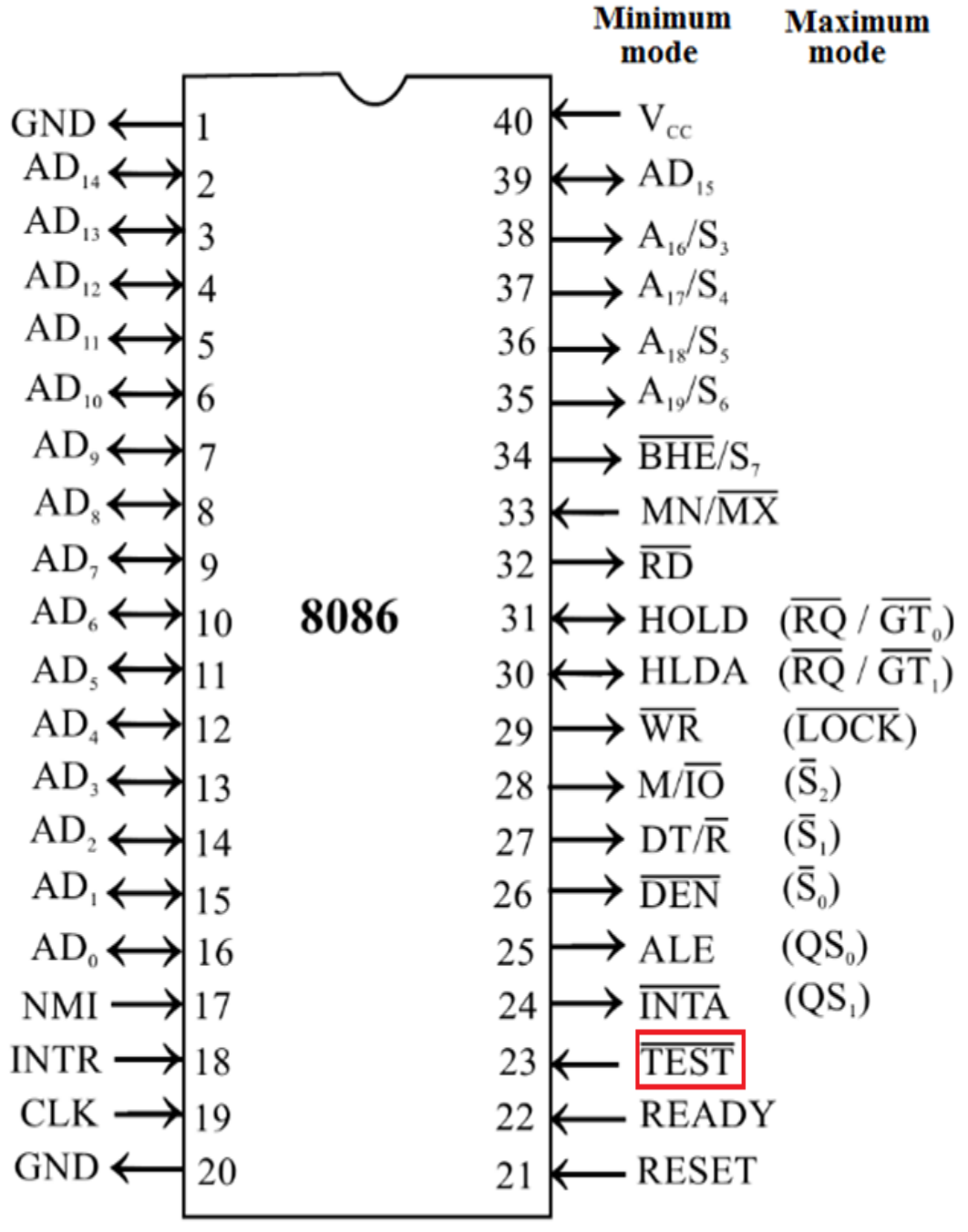
# NMI
## Pin 17 (Input)

- It is a positive edge triggered non-maskable interrupt request.
- It causs a Type 2 interrupt.



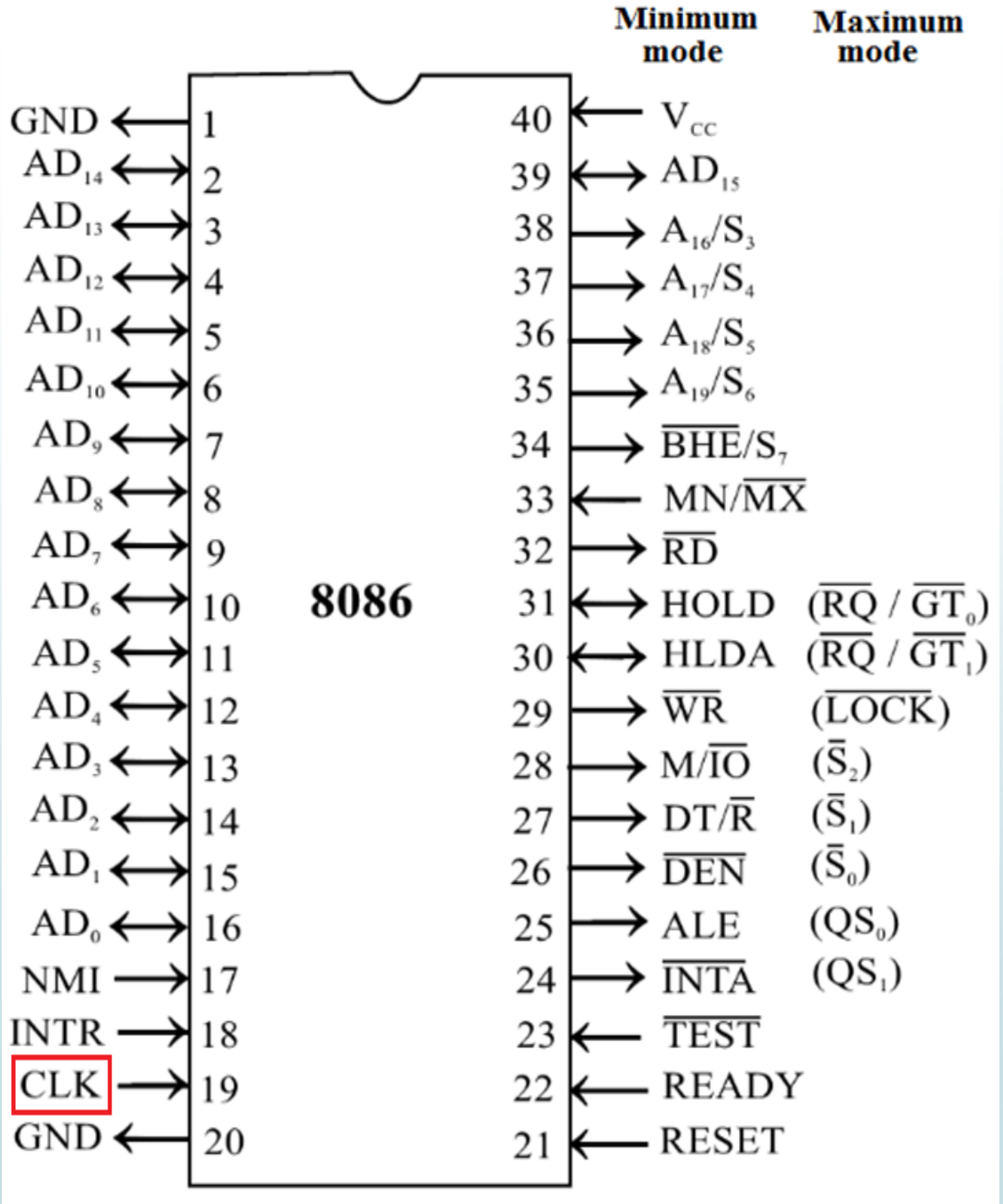
# $\overline{\text{TEST}}$ Pin 23 (Input)

- If low, execution continues else microprocessor enters into wait state
- This input is examined by the 'WAIT' instruction
- This input is synchronized internally during each clock cycle on the leading edge of the clock cycle



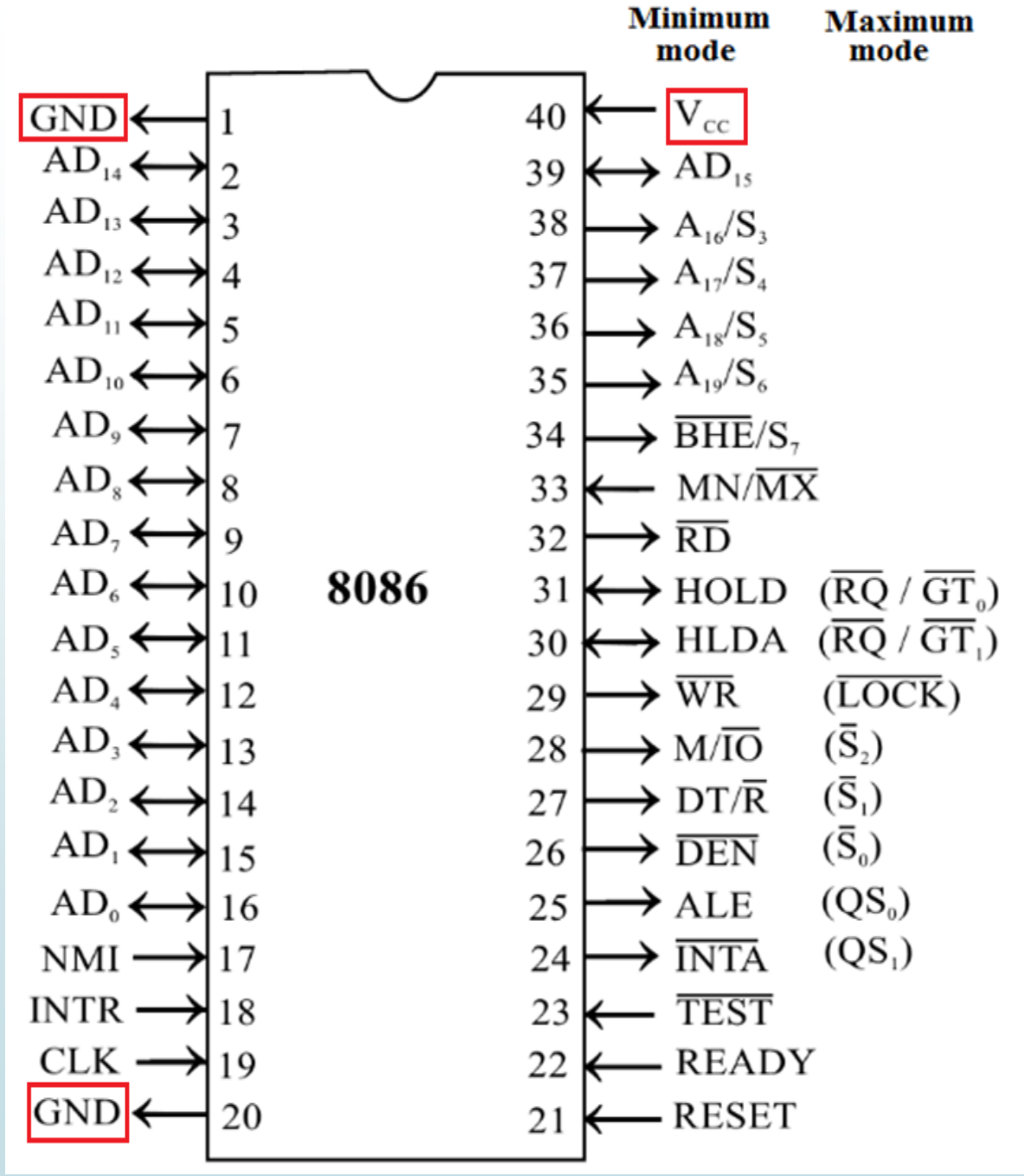
# CLK
# Pin 19 (Input)

- This clock input provides the basic timing for processor operation and bus control activity.
- Clock frequency depends on the version of 8086
- It requires clock signal ( with 33% duty cycle) from some external, crystal controlled generator to synchronize internal operations



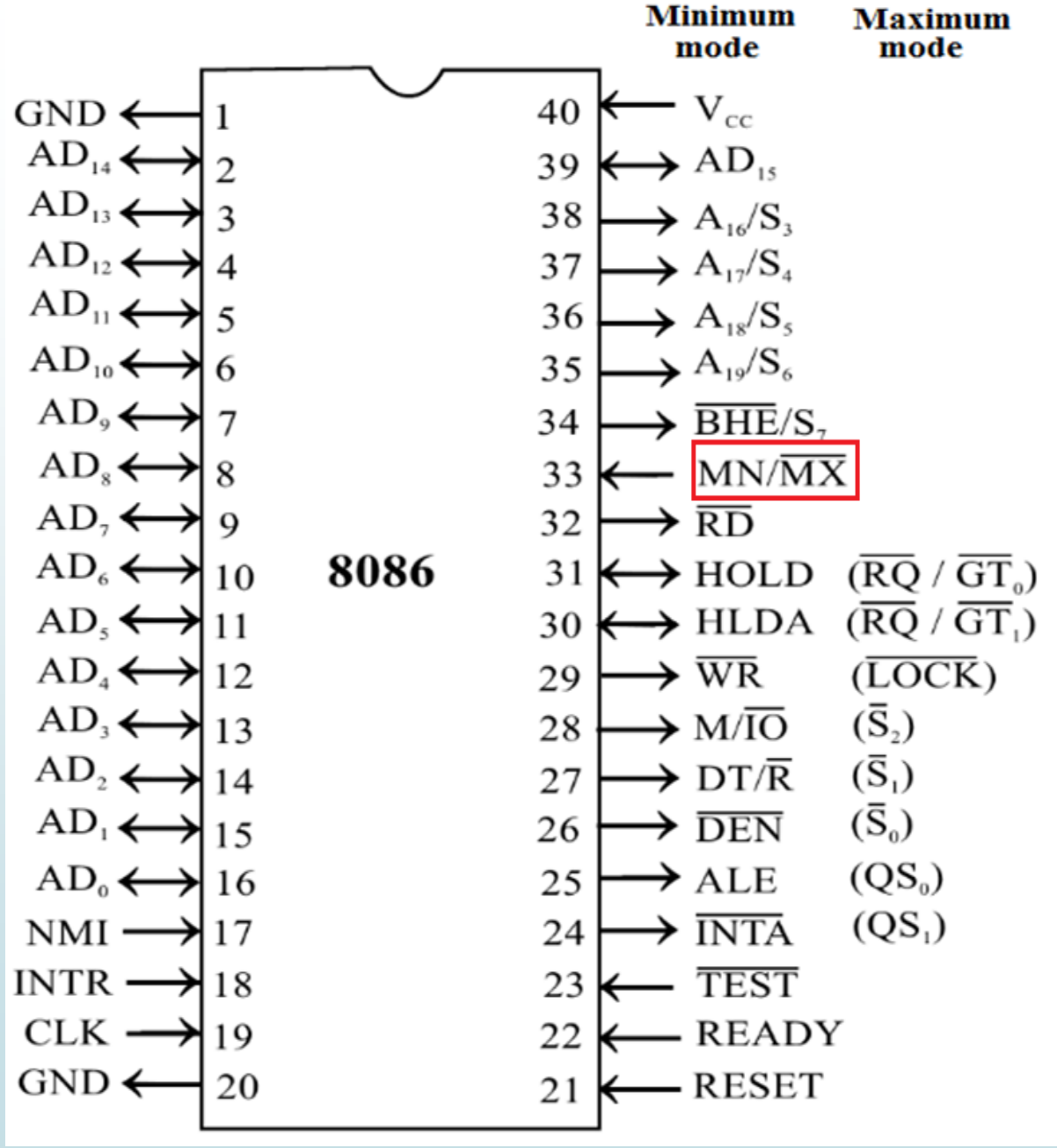
# $V_{CC}$ & GND Pin 40, 1 & 20

- $V_{CC}$: +5 V power supply for the operation of the internal circuit
- GND: ground for the internal circuit



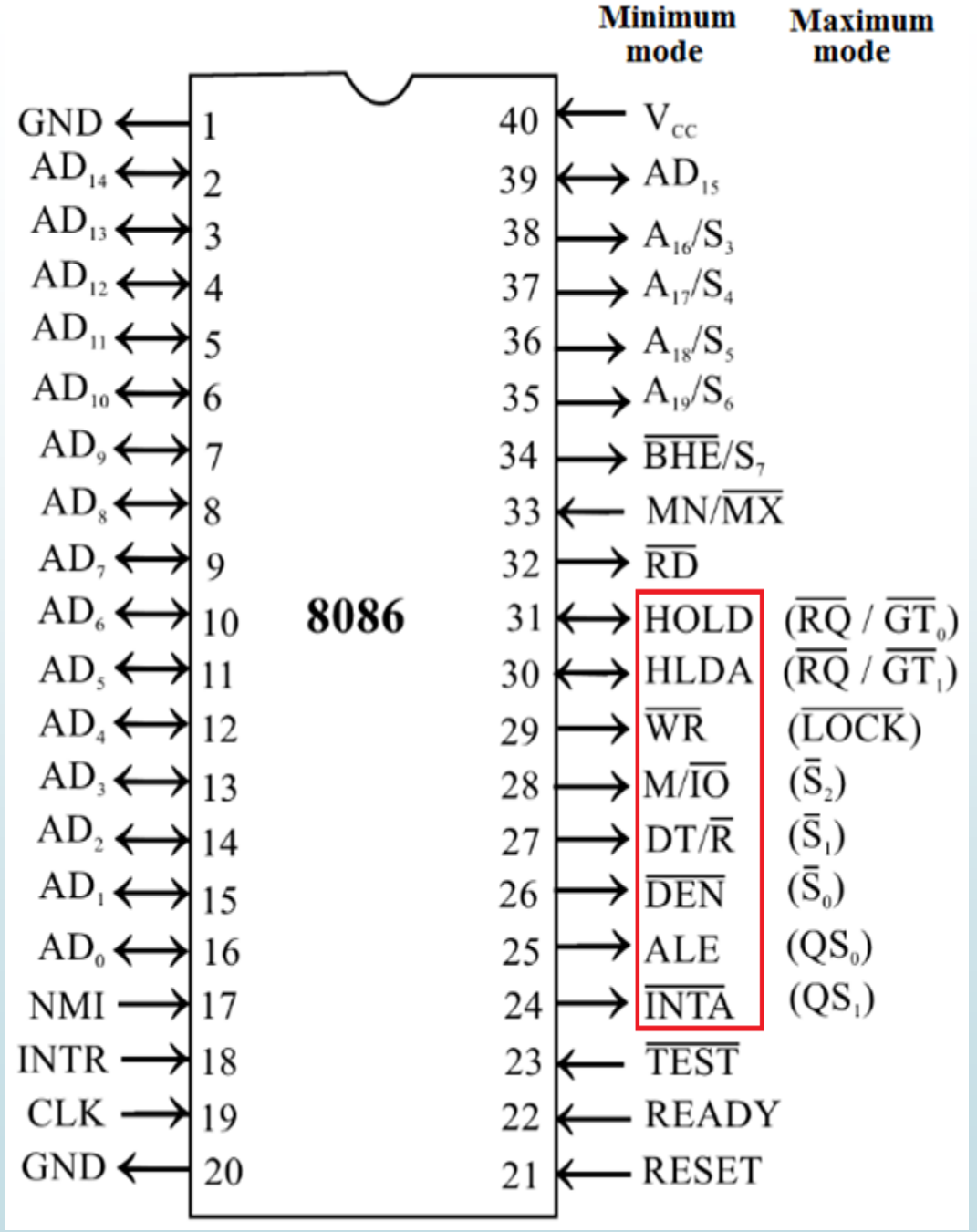
# MN / $\overline{MX}$ Pin 33 (Input)

- 8086 works in two modes:
  - ✓ Minimum Mode
  - ✓ Maximum Mode
- If MN/$\overline{MX}$ is high, it works in minimum mode.
- If MN/$\overline{MX}$ is low, it works in maximum mode.



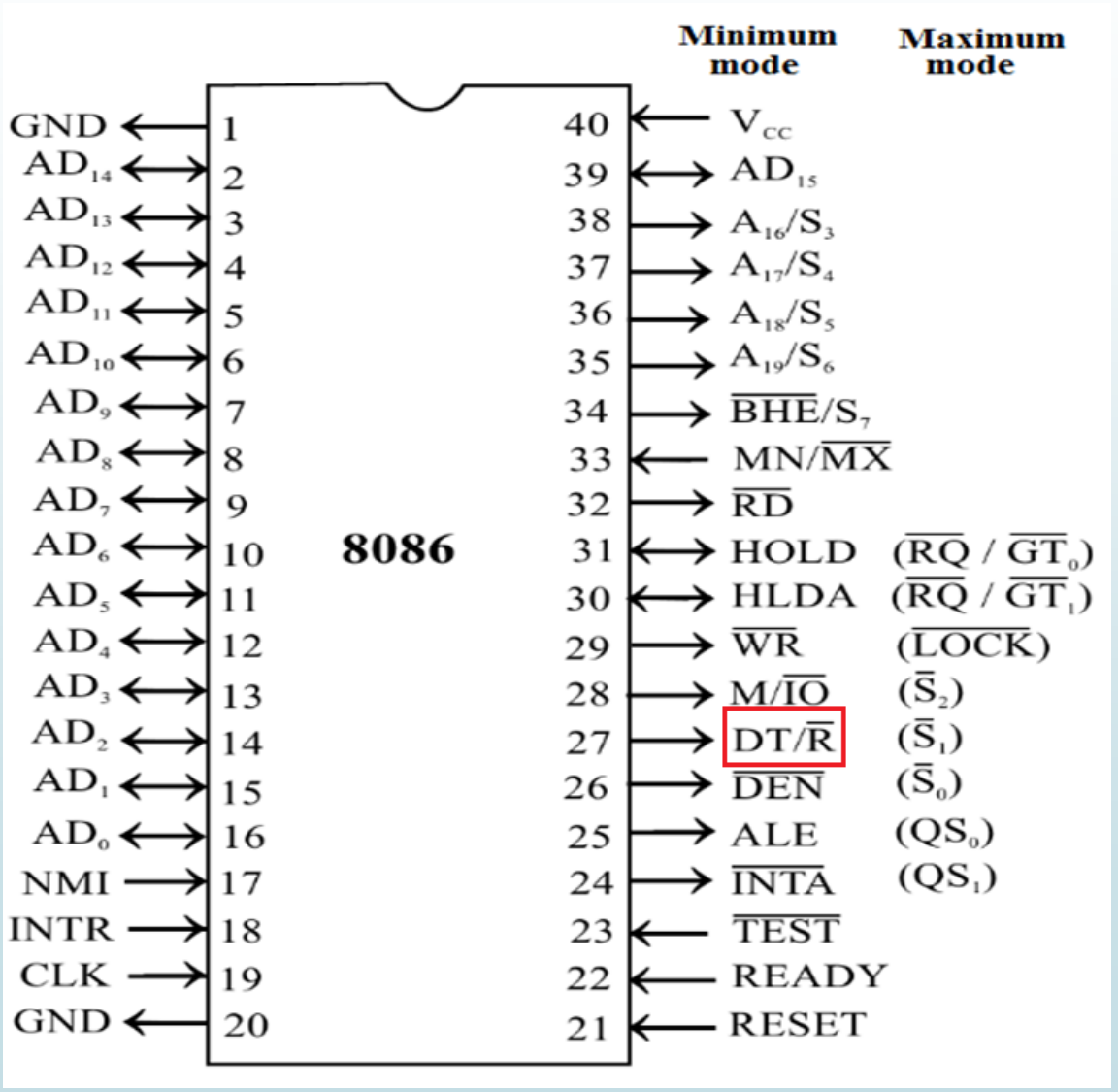
# Minimum Mode Signals

- For this mode, the MN/$\overline{MX}$ pin is tied to $V_{CC}$ (logic high)
- In this mode, the 8086 itself generates all bus control signals



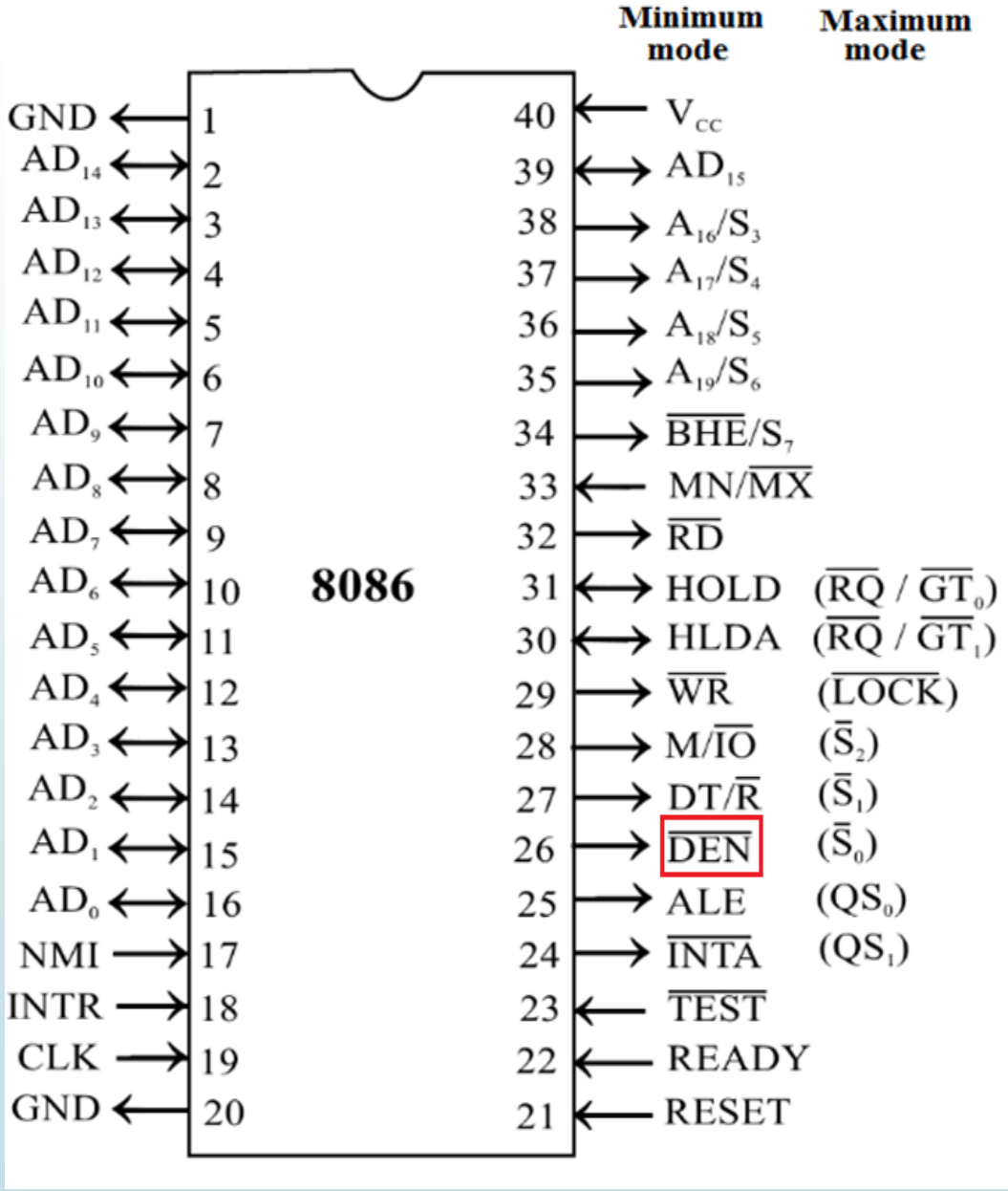
# DT/$\overline{R}$ (Data Transmit/Receive) Pin 27 (Output)

- This signal is used to decide the direction of data flow through the transceivers (bidirectional buffers)
- When it is high, data is transmitted out by the processor
- When it is low, data is received in by the processor
- It is tristated during 'hold acknowledge' cycle.



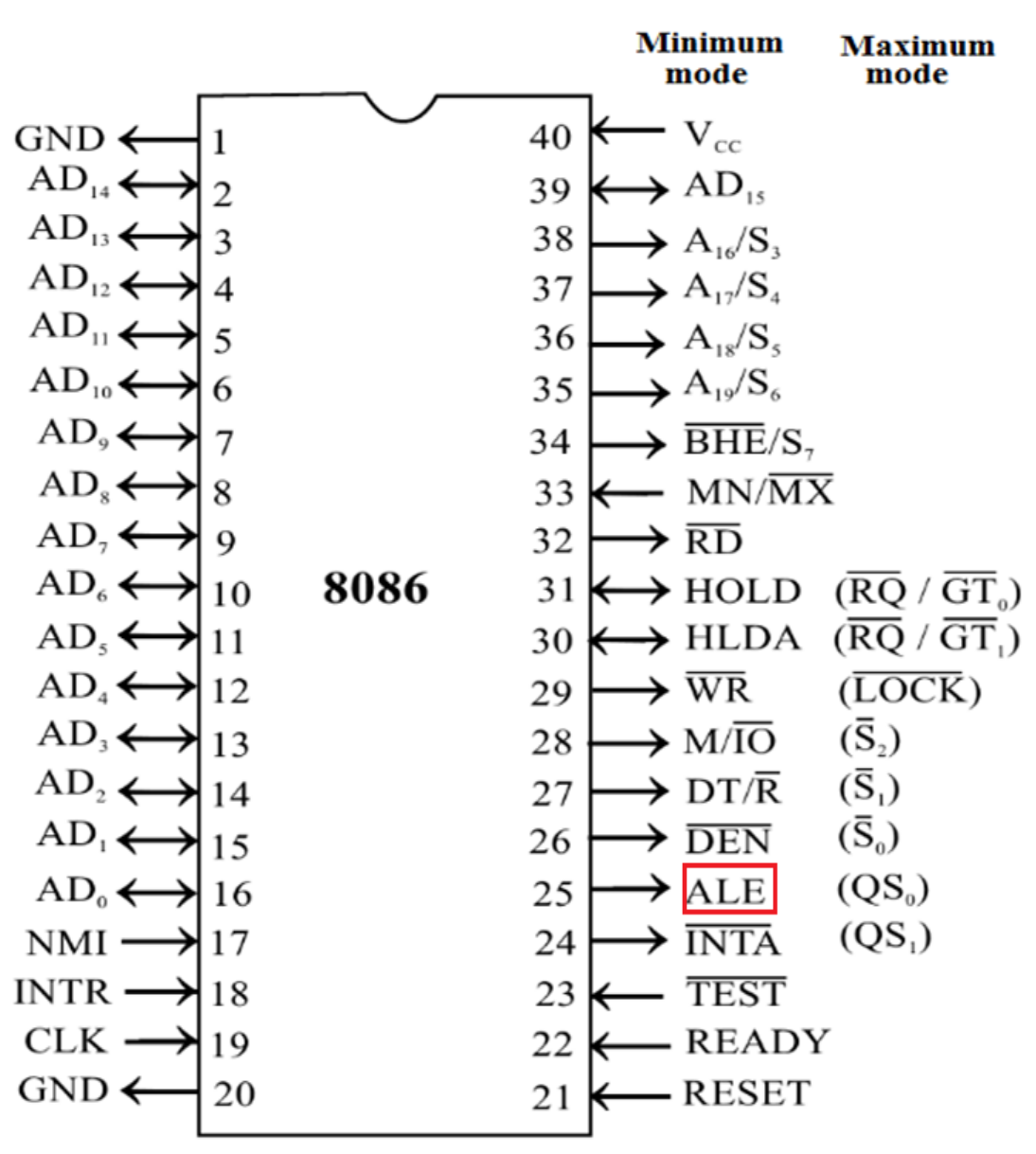
# $\overline{DEN}$ (Data Enable) Pin 26 (Output)

- This signal indicates the availability of valid data over the address/data lines.
- It is used to enable the transceivers to separate the data from the multiplexed address/data signal.
- It is active from the middle of $T_2$ until the middle of $T_4$.
- It is tristated during 'hold acknowledge' cycle.



# ALE (Address Latch Enable) Pin 25 (Output)

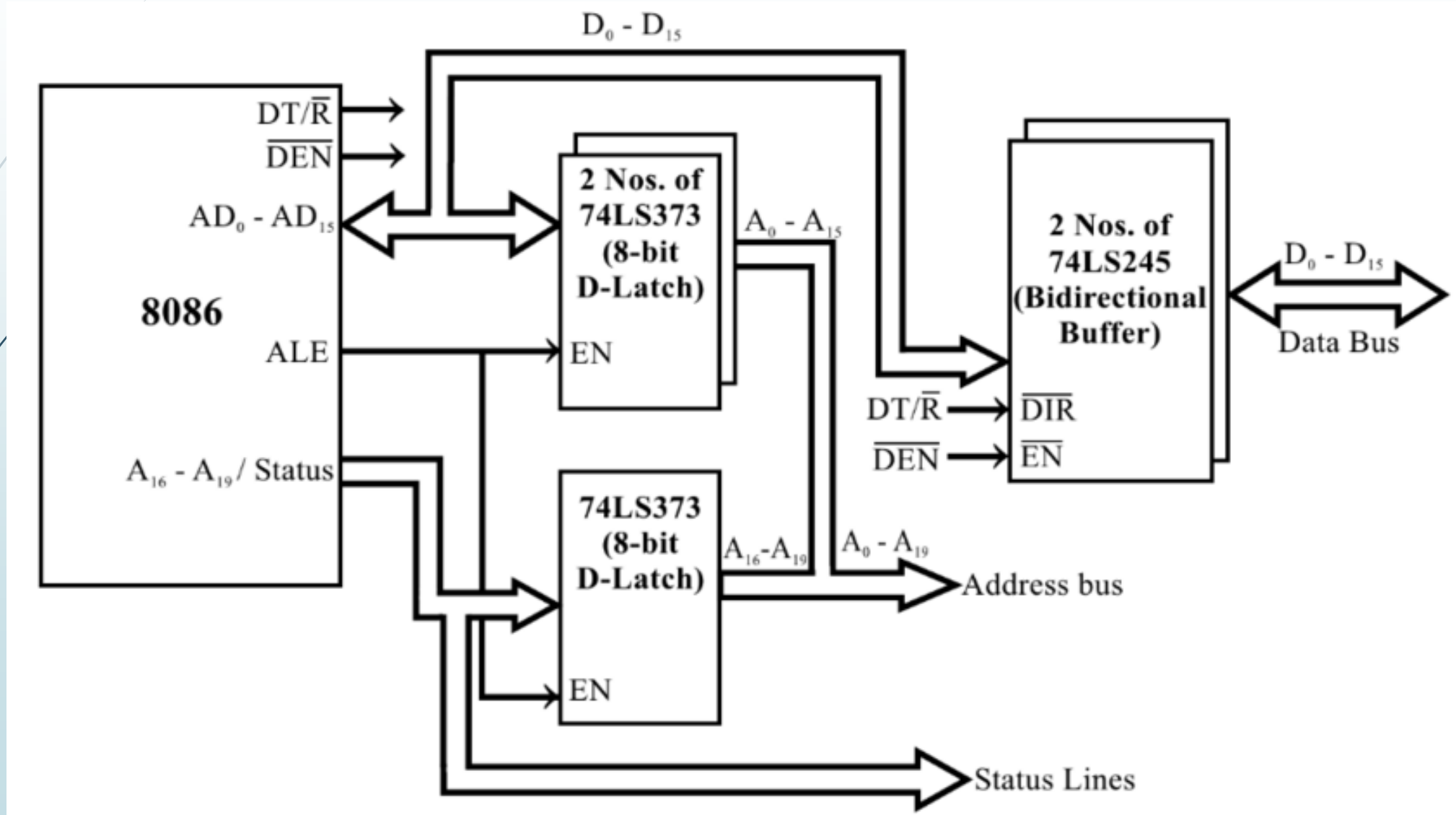
- This output signal indicates the availability of the valid address on the address/data lines.
- It is used to demultiplex the address and data lines using external latches
- That is to demultiplex the $AD_0$-$AD_{15}$ into $A_0$-$A_{15}$ & $D_0$-$D_{15}$ using external latches
- It is active high and is never tristated.



# Cont.

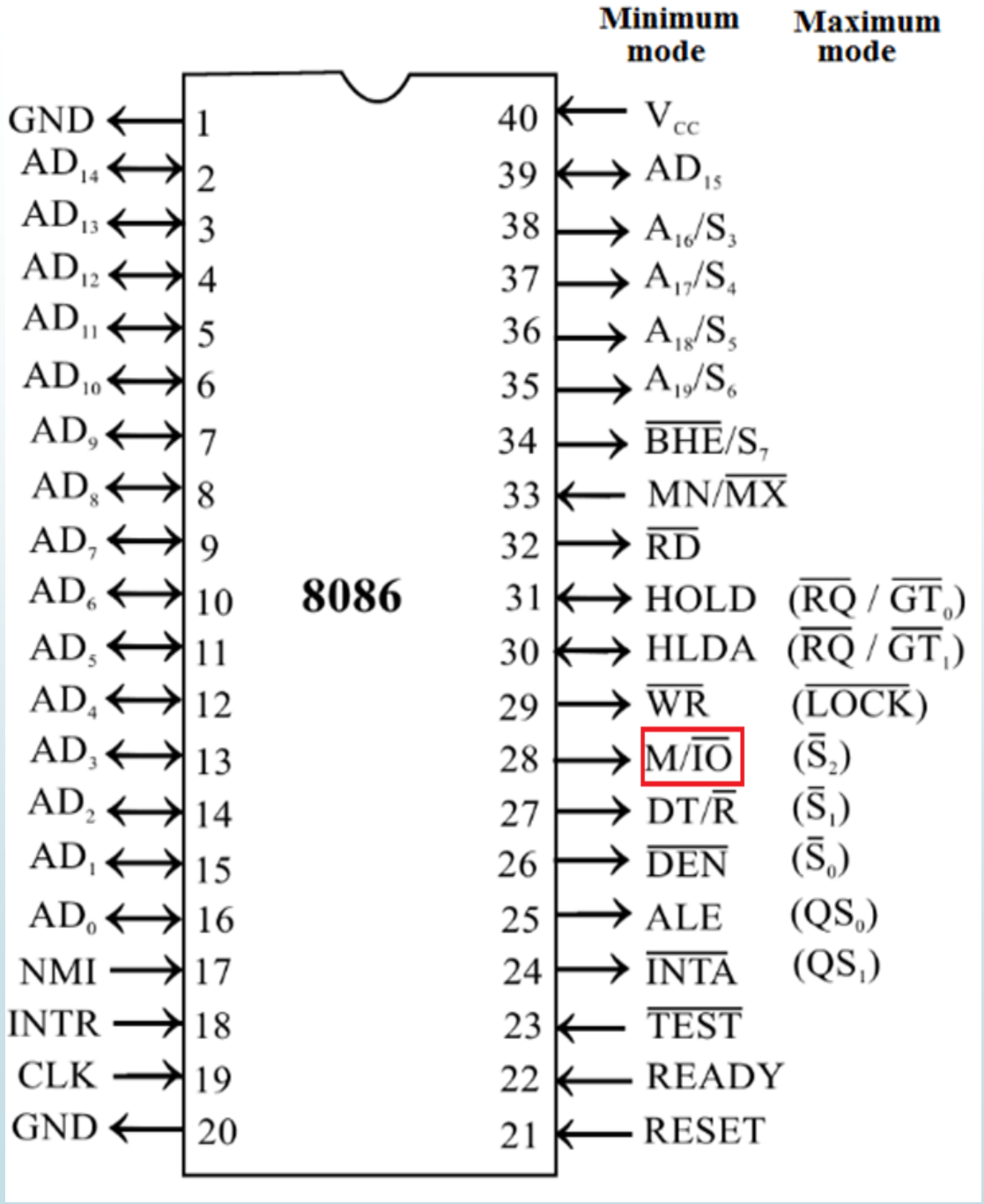
- The address bits are separated from the status bits using latches controlled by the ALE Signal

# Demultiplexing of address and data lines in 8086 processor



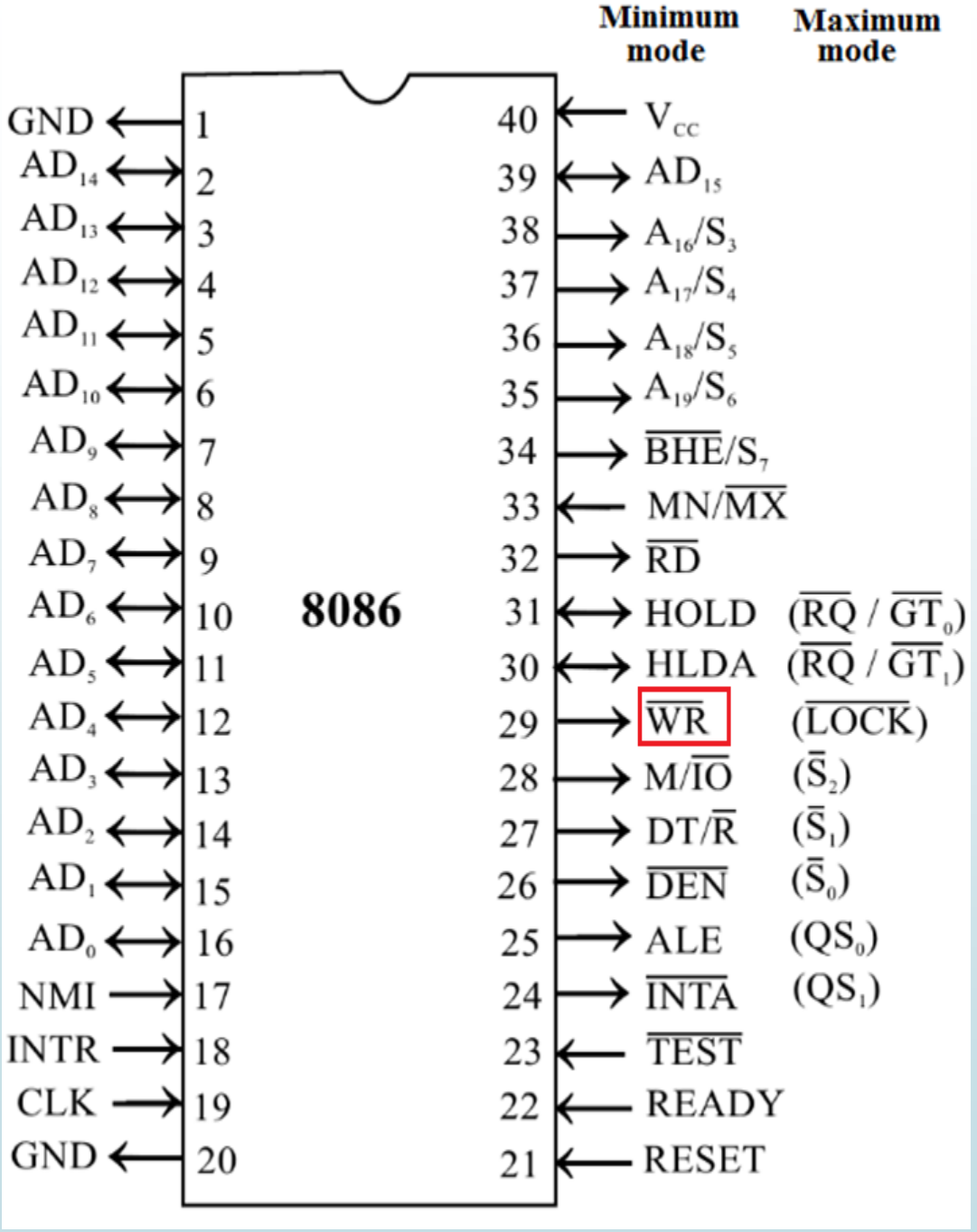
# M / $\overline{IO}$ Pin 28 (Output)

- It is used to differentiate memory access and IO access.
- When it is high, memory is accessed.
- When it is low, I/O devices are accessed.
- It is tristated during 'hold acknowledge' cycle.



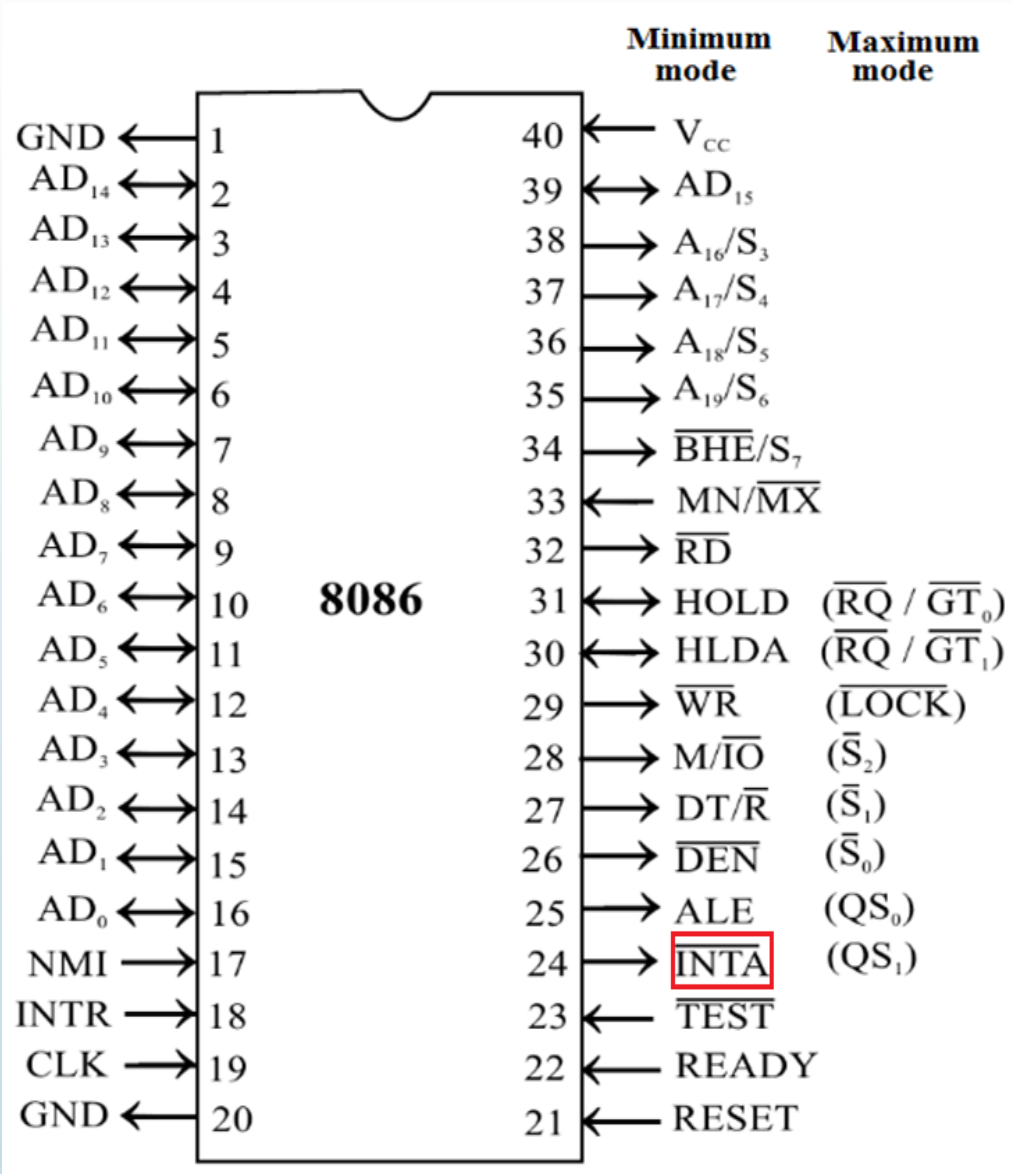
# $\overline{WR}$
# Pin 29 (Output)

- It is a write control signal
- It is asserted low whenever the processor writes data to memory or IO port



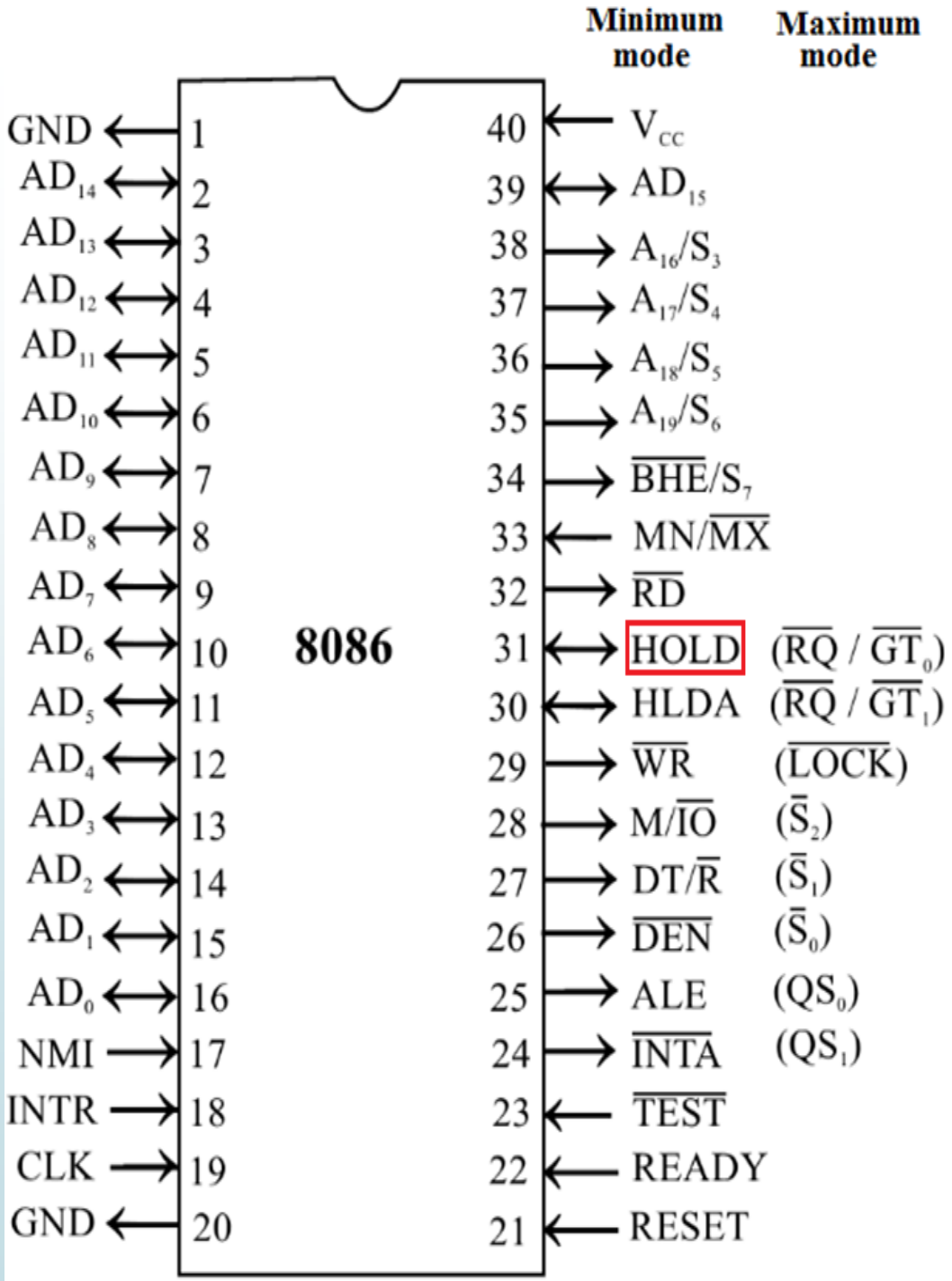
# $\overline{\text{INTA}}$ (Interrupt Acknowledge) Pin 24 (Output)

- This is an interrupt acknowledge signal.
- When microprocessor receives INTR signal, it acknowledges the interrupt by generating this signal.
- The 8086 output is asserted low on this line to acknowledge when the interrupt request is accepted by the processor
- It is active low during $T_2$, $T_3$ and $T_W$ of each interrupt acknowledge cycle



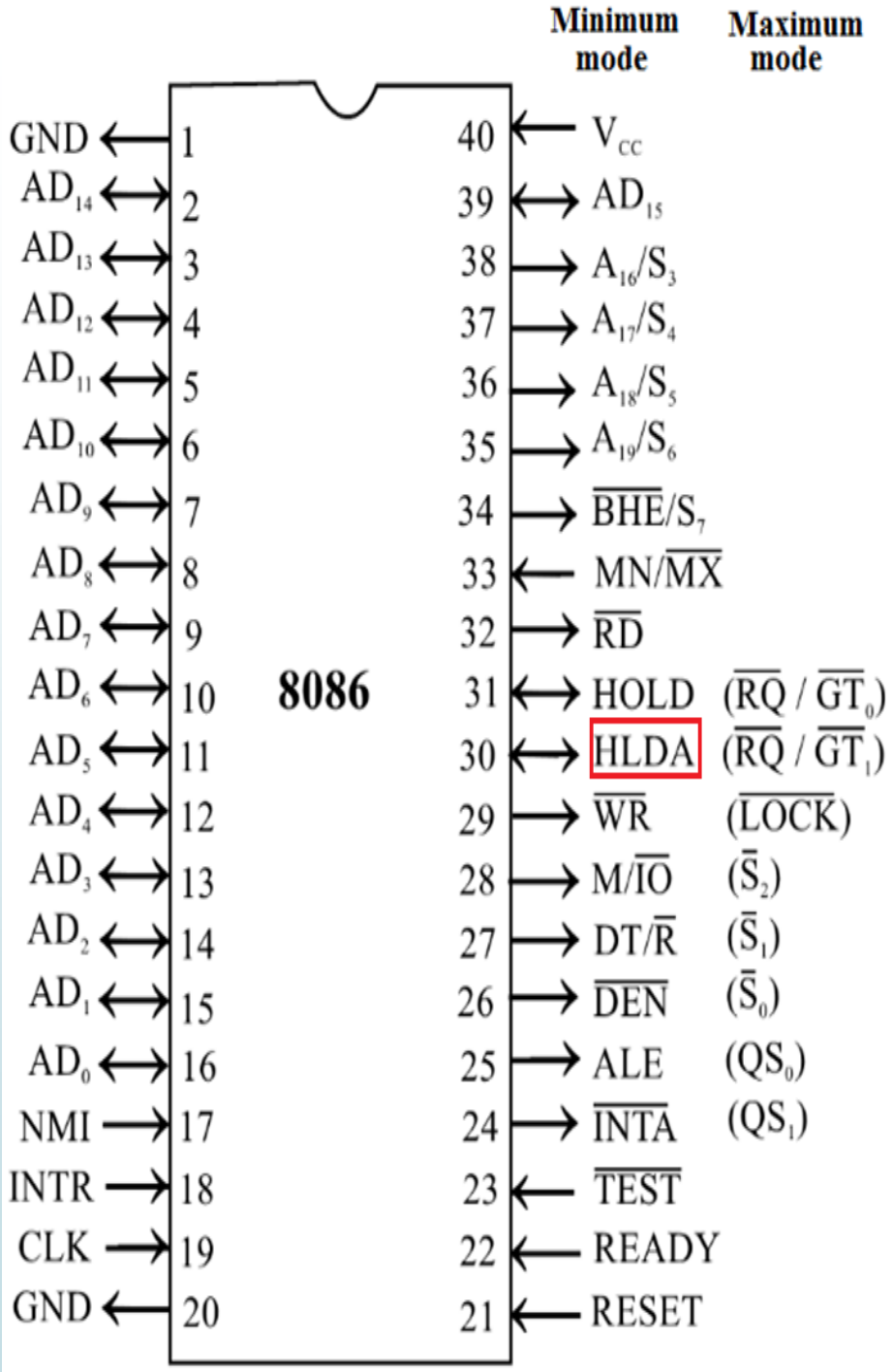
# HOLD (Hold Request) Pin 31 (Input)

- It is an input signal to the processor from other bus masters as a request to grant control of the bus.
- It is usually used by the DMA controller to get the control of the bus



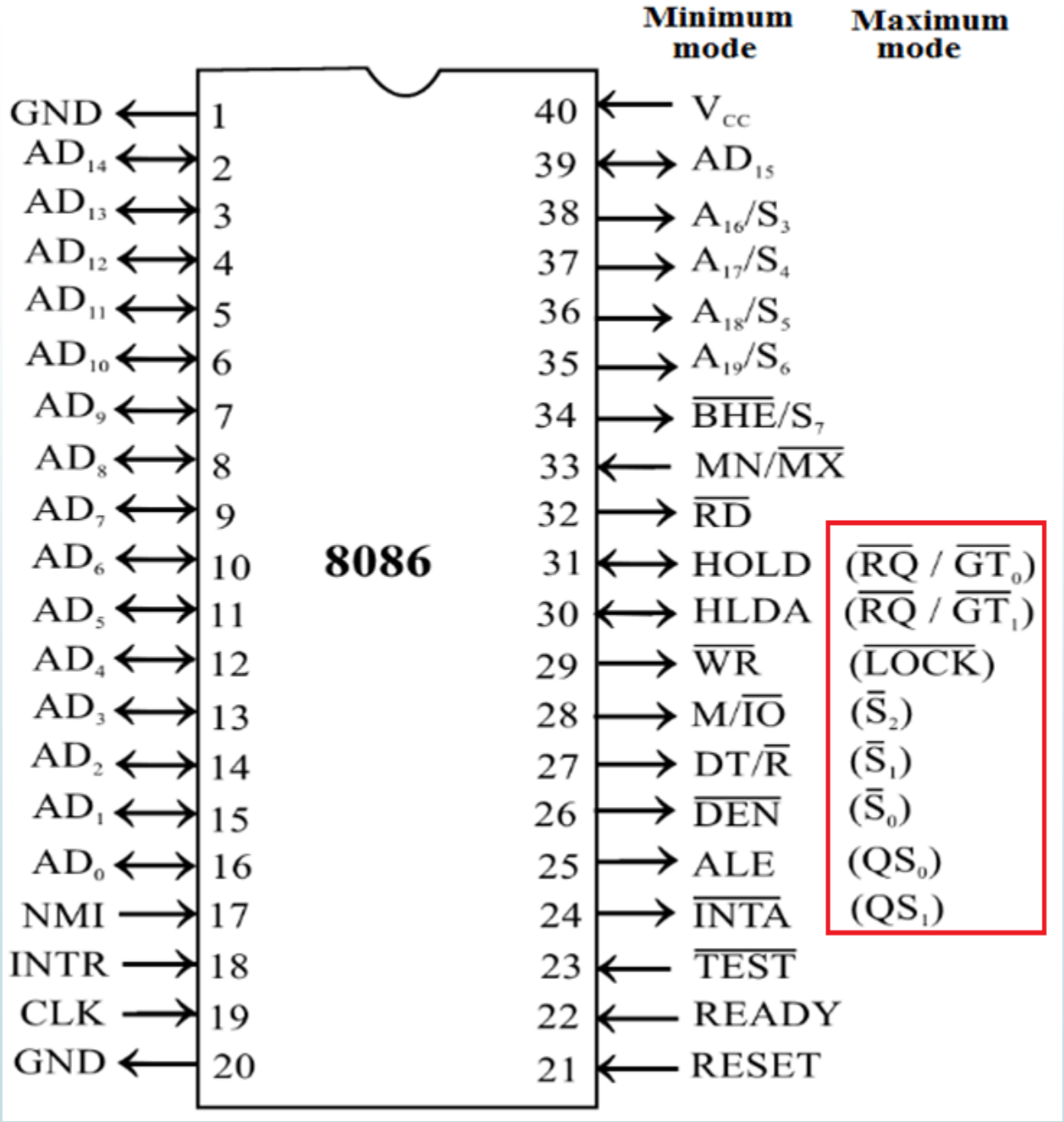
# HLDA (Hold Acknowledge) Pin 30 (Output)

- It is an acknowledge signal by the processor to the master requesting the control of the bus through HOLD.
- The acknowledge is asserted high when the processor accepts the HOLD, in the middle of the next clock cycle after completing the current bus (instruction) cycle.



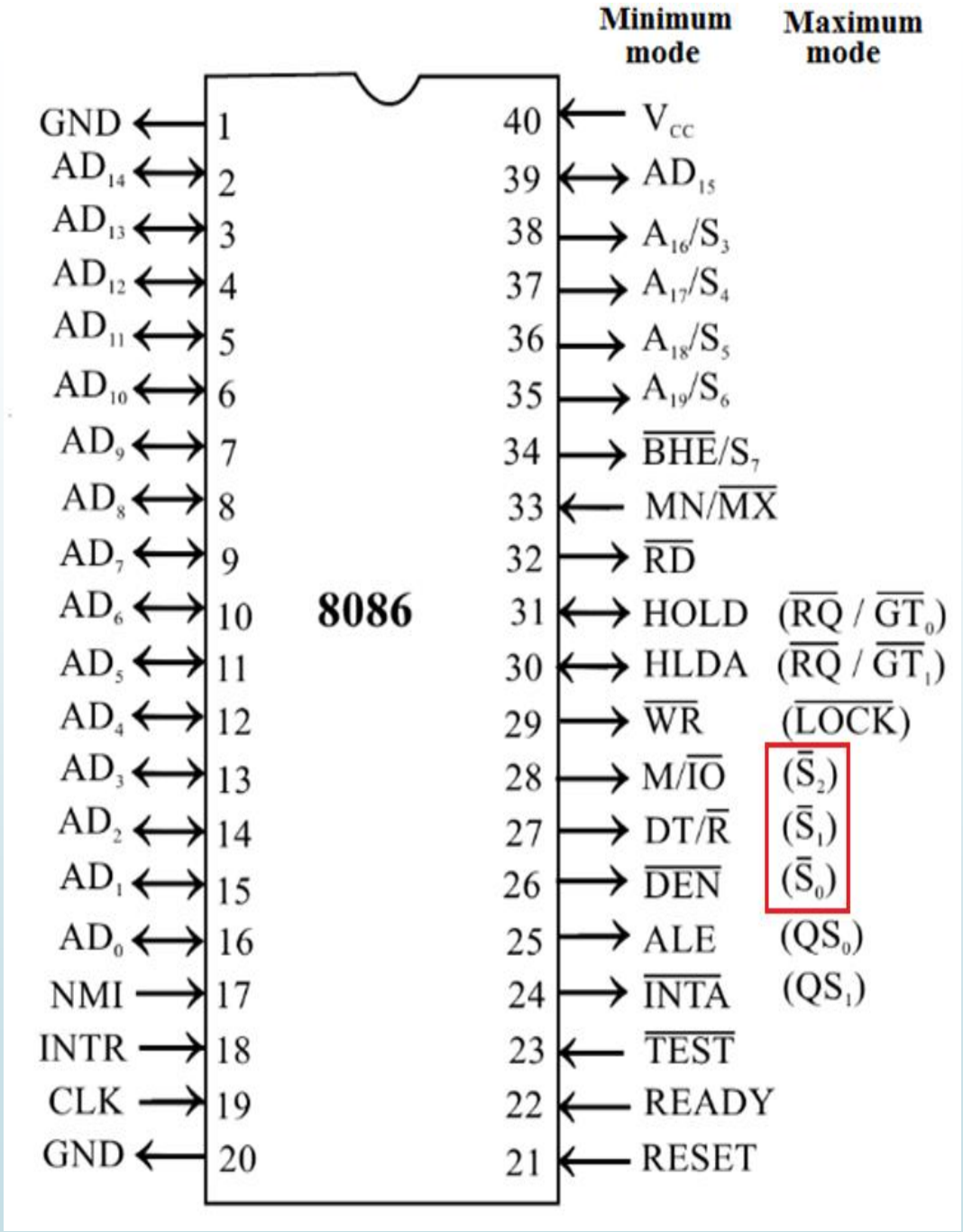
# Maximum Mode Signals

- For this mode, the MN/$\overline{MX}$ pin is tied to GND (logic low)



# $\overline{S_0}$, $\overline{S_1}$, $\overline{S_2}$ - Status Lines Pin 26, 27, 28 (Output)

- These status signals indicate the type of operation being carried out by the microprocessor.
- They are used by the 8288 bus controller to generate bus timing and control signals
- These become active during T4 of the previous cycle and remain active during T1 and T2 of the current bus cycle.
- The status lines return to passive state during T3 of the current bus cycle.

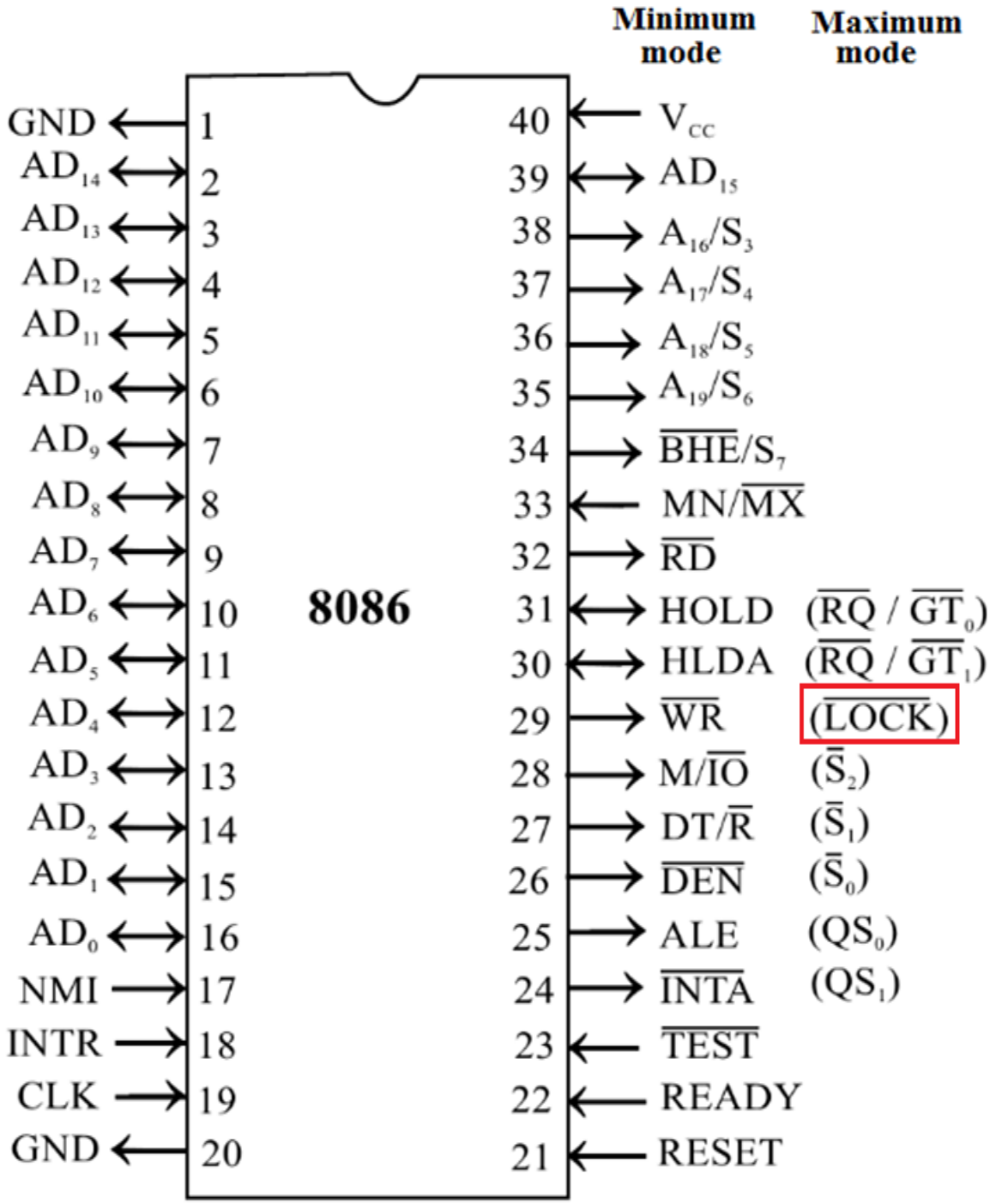


# Status Signals ($\overline{S}_0$, $\overline{S}_1$, $\overline{S}_2$) during various Machine Cycles

| Status signal | | | Machine cycle |
|---|---|---|---|
| $\overline{S}_2$ | $\overline{S}_1$ | $\overline{S}_0$ | |
| 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 |

# $\overline{\text{LOCK}}$
# Pin 29 (Output)

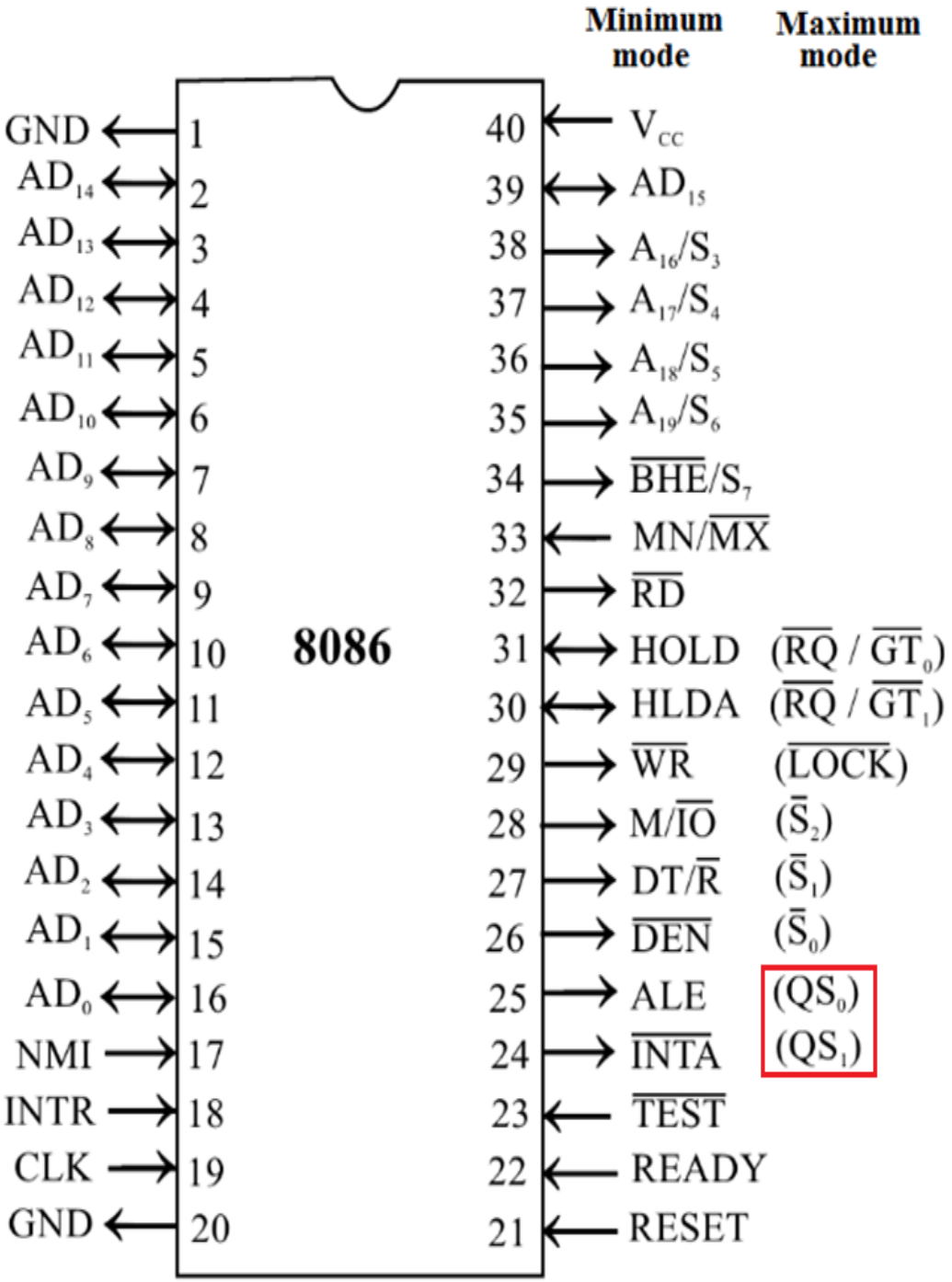
- This output pin indicates that other system bus masters will be prevented from gaining the system bus, while this signal is low.
- When it goes low, all interrupts are masked.
- This pin is activated by using LOCK prefix on any instruction and remains active until the completion of the next instruction.



# $QS_1$ and $QS_0$ (Queue Status) Pin 24 and 25 (Output)

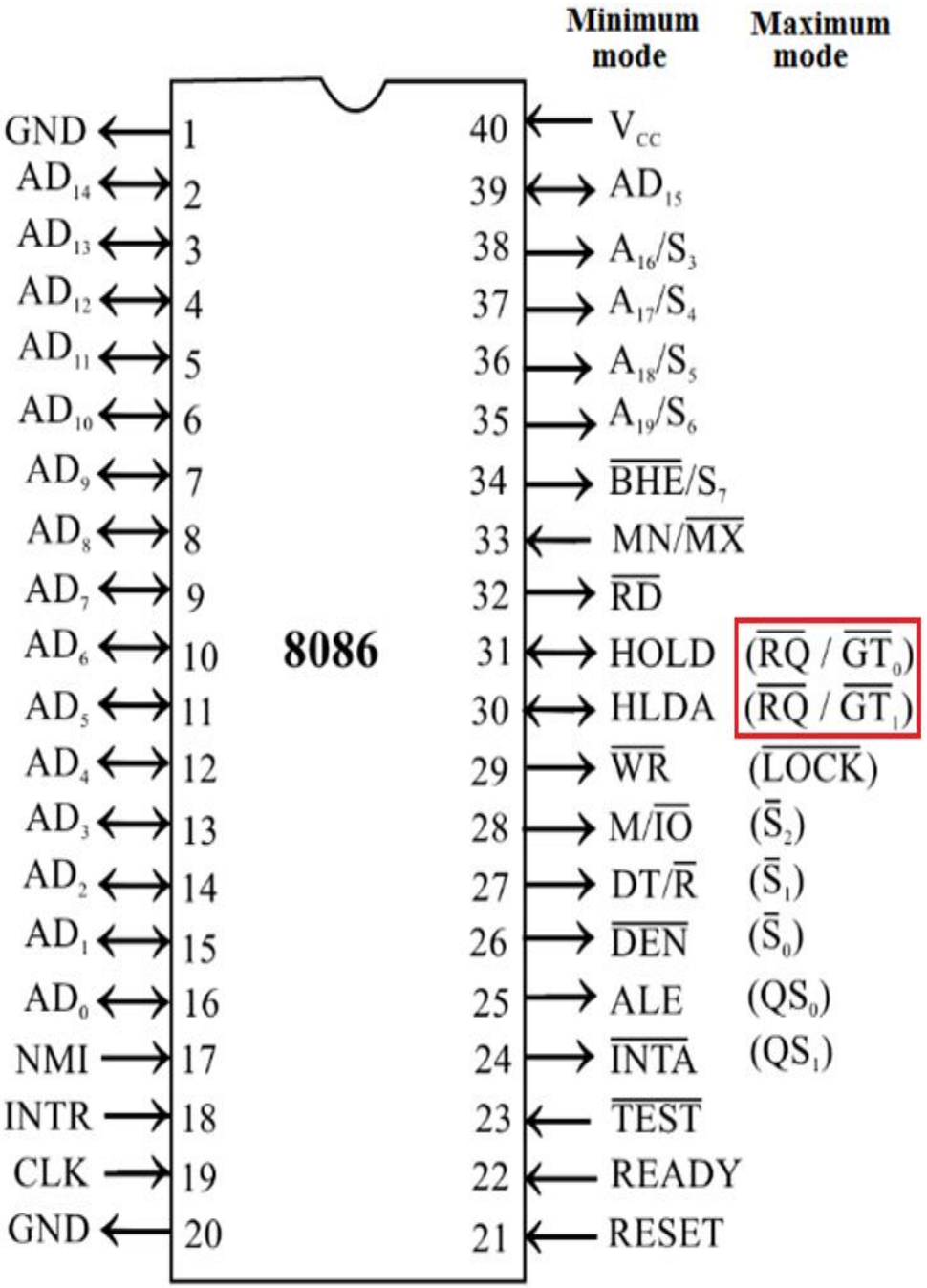
- These pins provide the status of instruction queue.

| 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 |

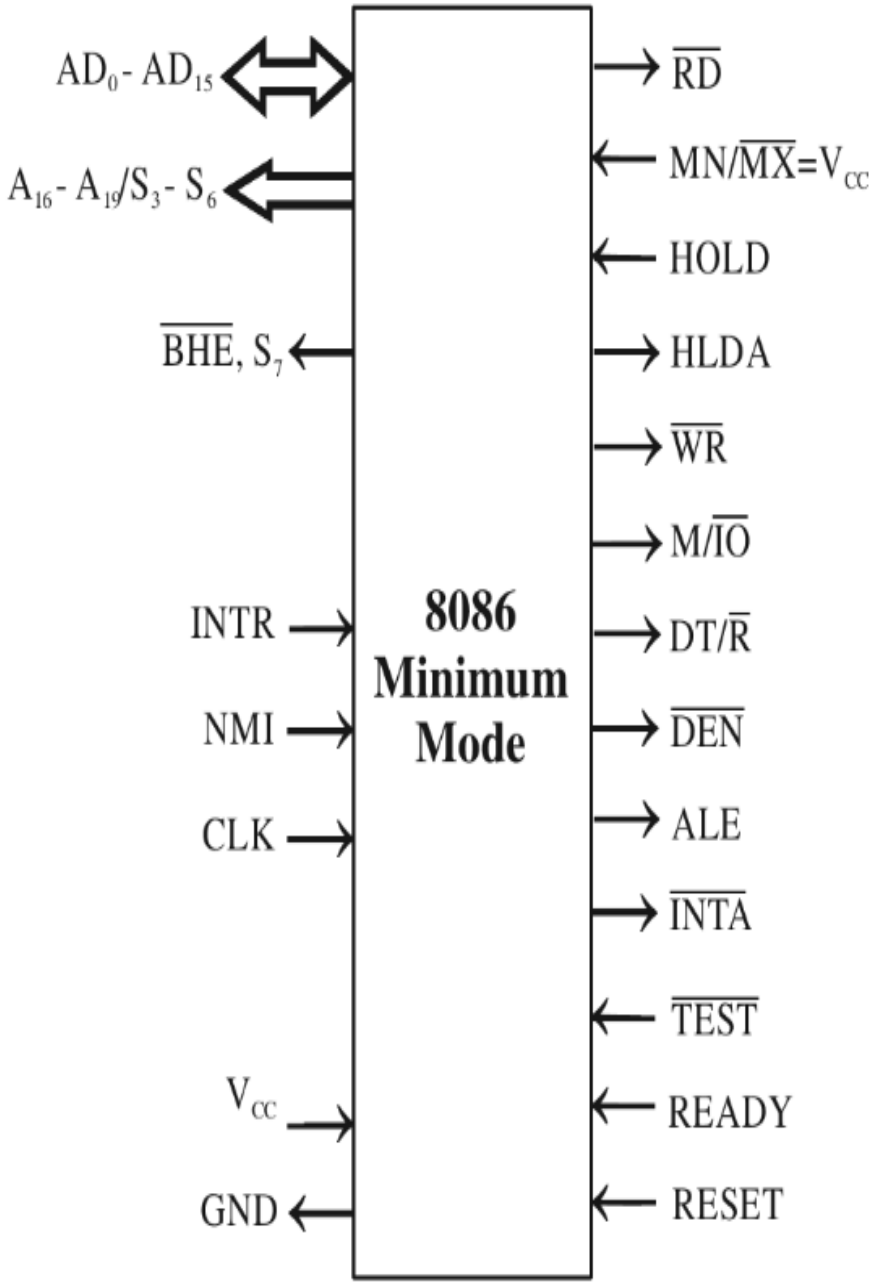


# $\overline{RQ}/\overline{GT}_1$ and $\overline{RQ}/\overline{GT}_0$ Pin 30 and 31 (Bi-directional)

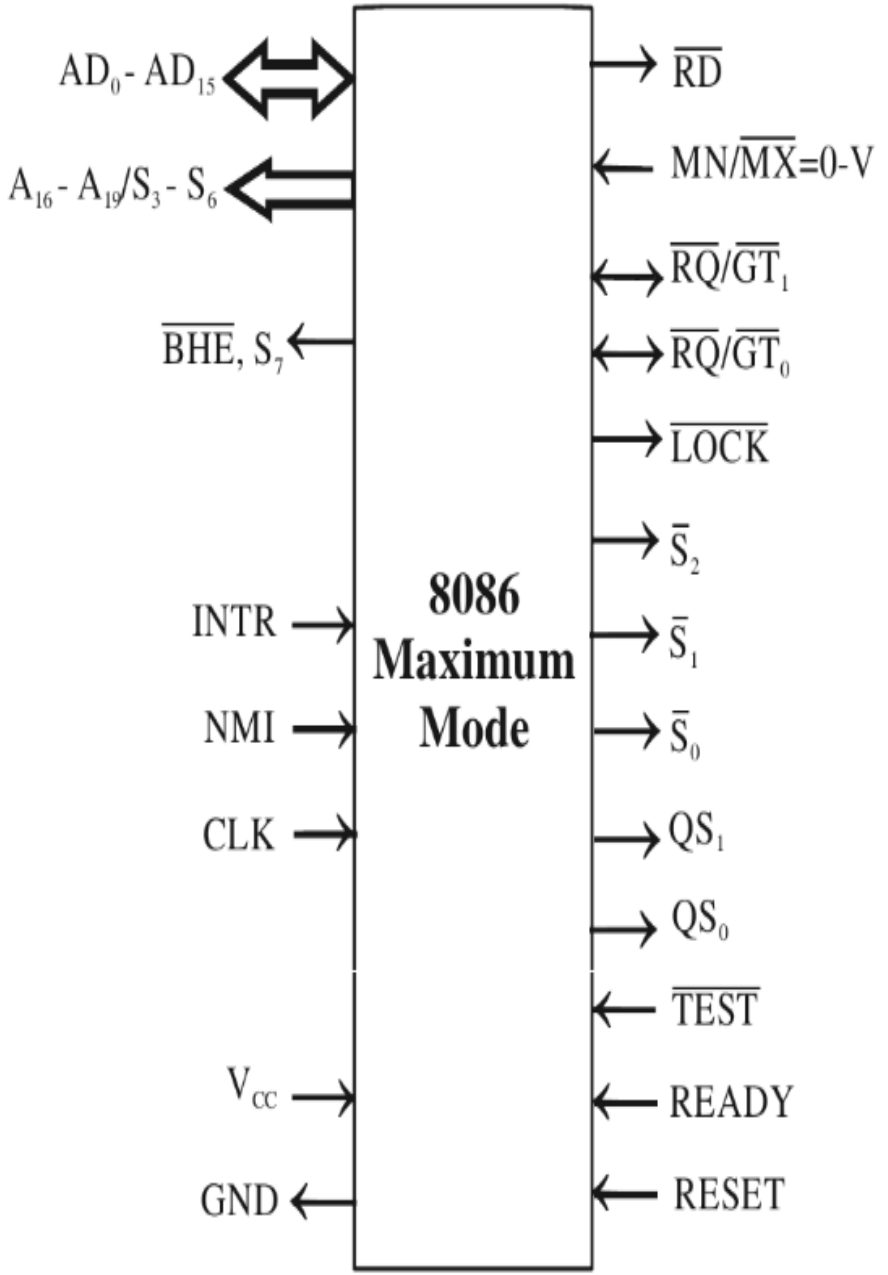
- These are bus request/bus grant pins
- Other local bus masters request the processor through these lines to release the system bus.
- After receiving the request, processor sends acknowledge signal on the same lines.
- $\overline{RQ}/\overline{GT}_0$ has higher priority than $\overline{RQ}/\overline{GT}_1$.
- These pins have internal pull-up resistors and may be left unconnected.



# Two modes of 8086




8086-Minimum mode




8086-Maximum mode.

# Common Signals in both modes

| 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 |

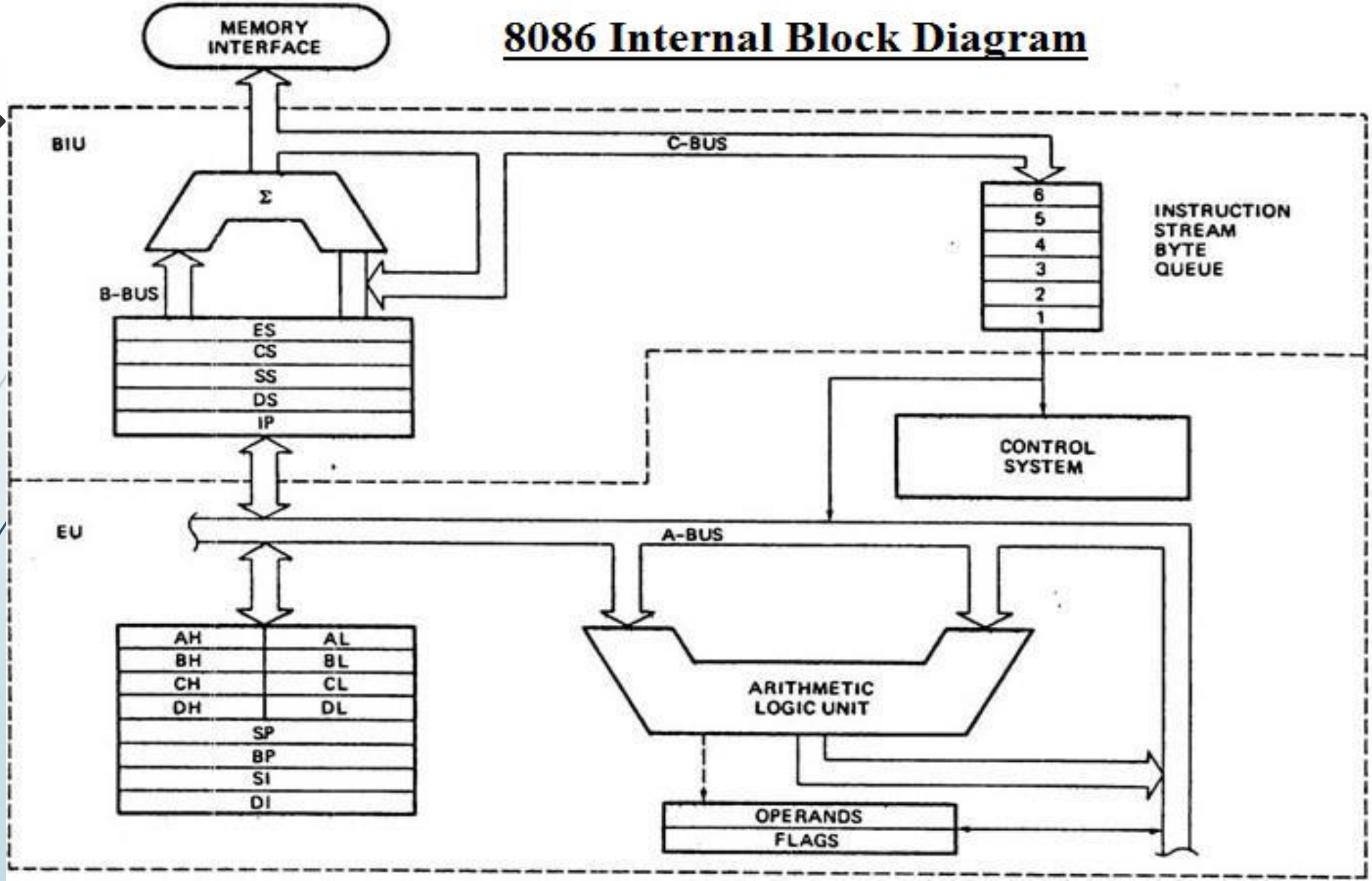
## MINIMUM MODE SIGNALS [MN/$\overline{MX}$ = $V_{cc}$ (Logic high)]

| 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 |

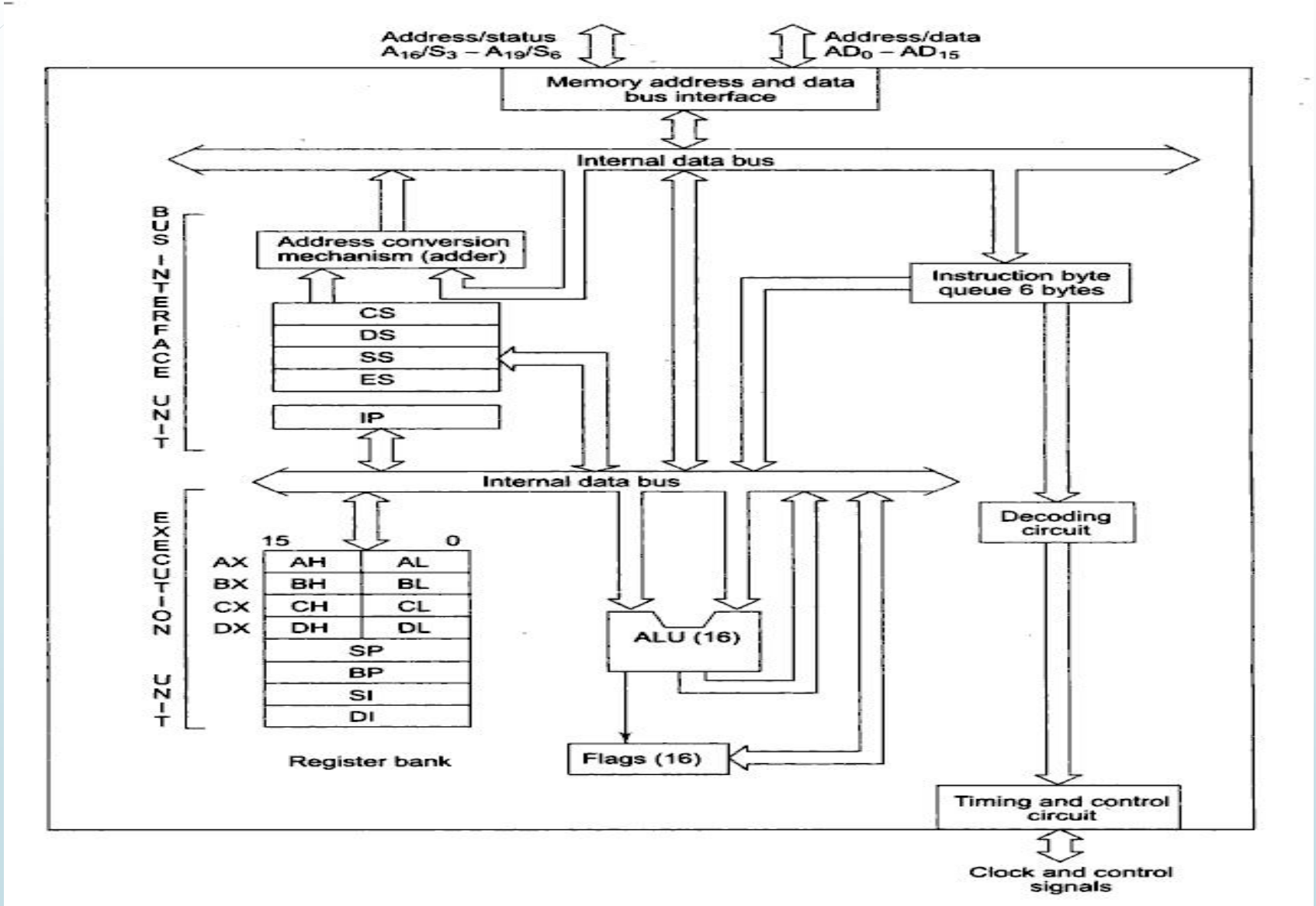
## MAXIMUM MODE SIGNALS [MN/$\overline{MX}$ = Ground(Logic low)]

| 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 |

# 8086 Internal Block Diagram



# Architecture of 8086



# 8086 Internal Architecture

- The complete architecture is divided into two parts:

1. Bus Interface Unit
2. Execution Unit

# 8086 Internal Architecture

- BIU: It sends out addresses, fetches instructions from memory, reads data from ports and memory and writes data to ports and memory.
- EU: It tells the BIU where to fetch instructions or data from, decodes instructions and executes instructions

# Execution Unit

- It contains:

1. **Control Circuitry**
2. **Instruction Decoder**
3. **Arithmetic Logic Unit (ALU)**
4. **Flag Register**
5. **General Purpose Registers**
6. **Pointers and Index Registers**

# Control Circuitry, Instruction Decoder, ALU

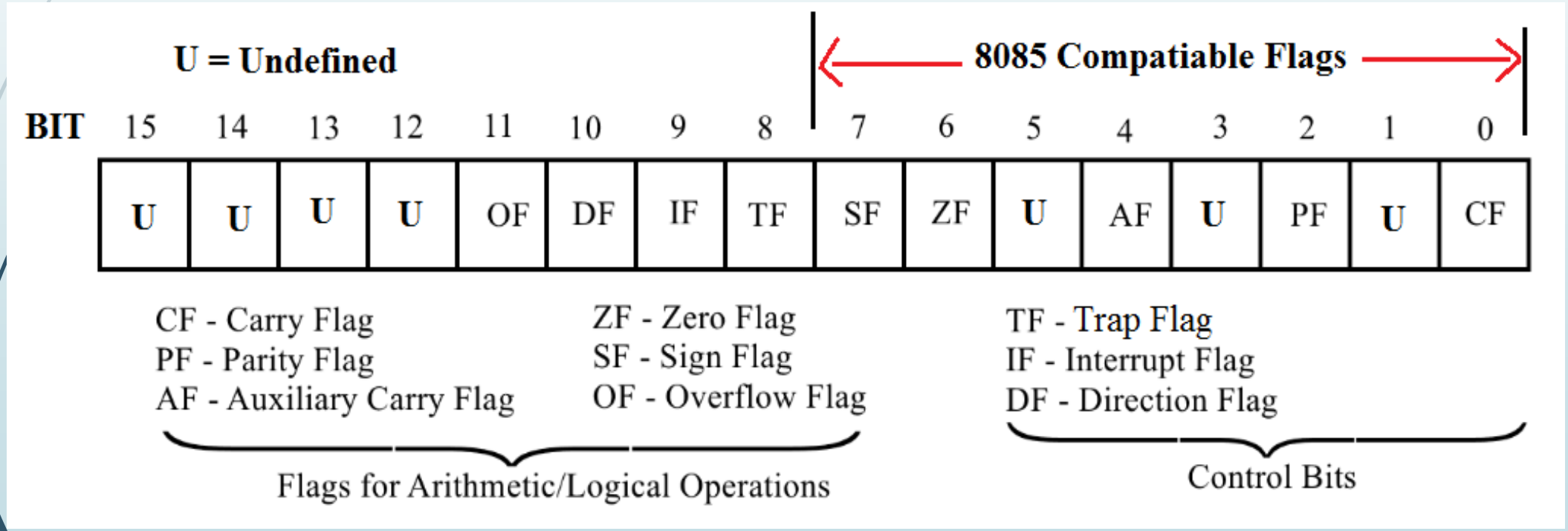
- ✓ **Control Circuitry** in the EU directs the internal operations
- ✓ A **Instruction decoder** translates instructions fetched from memory into a series of actions which the EU carries out
- ✓ **16 bit arithmetic logic unit** can add, subtract, AND, OR, XOR, increment, decrement, complement or shift binary numbers

# Flag Register

- A flag is a flip-flop that indicates some condition produced by the execution of an instruction or controls certain operations of the EU
- A 16-bit flag register in the EU contains nine active flags in the flag register

# Flag Register

- 8086 has 9 flags and they are divided into two categories:
  - Condition Flags
  - Control Flags



# Condition Flags

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

- **Carry Flag (CF):** This flag is set if there is a carry / borrow after an integer arithmetic.

- **Auxiliary Carry Flag (AF):** If an operation performed in ALU generates a carry / borrow from lower nibble (i.e. $D_0 – D_3$) to upper nibble (i.e. $D_4 – D_7$), then AF is set. It is used in BCD Addition.

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

# Condition Flags

- **Zero Flag (ZF):** It is set; if the result of arithmetic or logical operation is zero else it is reset.
- **Sign Flag (SF):** In sign magnitude format, the sign of number is indicated by MSB bit. If the result of operation is negative, sign flag is set.
- **Overflow Flag (OF):** It occurs when signed numbers are added or subtracted. An OF indicates that the result has exceeded the capacity of machine that is result is out of range

# Control Flags

- Control flags are set or reset deliberately to control the operations of the execution unit. Control flags are as follows:
- **Trap Flag (TP):**
  - It is used for single stepping through a program
  - It allows user to execute one instruction of a program at a time for debugging.
  - When trap flag is set, program can be run in single step mode.

# Control Flags

- **Interrupt Flag (IF):**
  - It is an interrupt enable / disable flag.
  - If it is set, the INTR interrupt of 8086 is enabled and if it is reset then INTR is disabled.

# Control Flags

- **Directional Flag (DF):**
  - It is used with string operations.
  - If it is set, string bytes are accessed from higher memory address to lower memory address.
  - When it is reset, the string bytes are accessed from lower memory address to higher memory address.

# General Purpose Registers

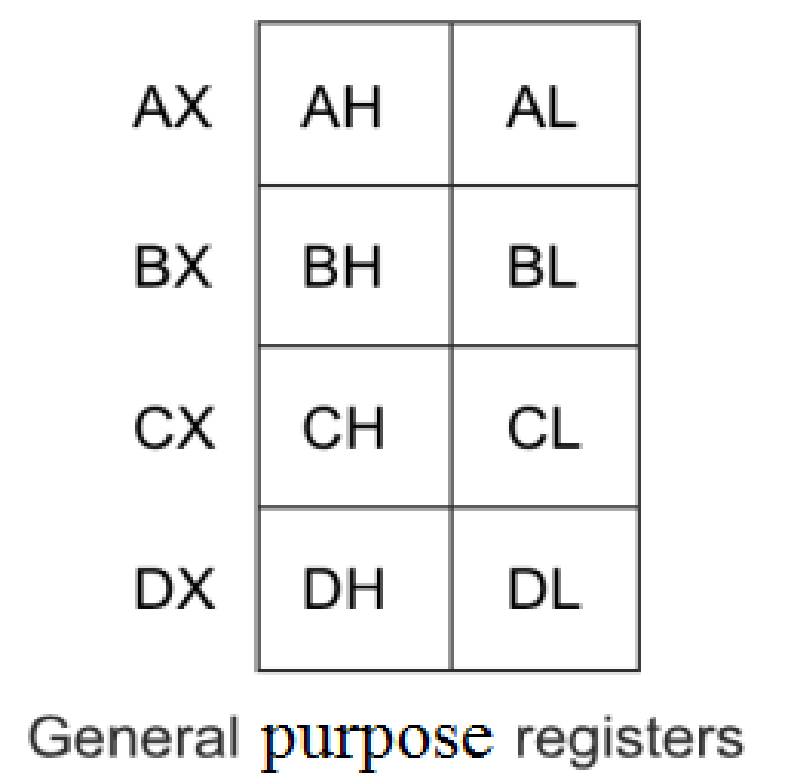
- EU has eight general purpose registers, labeled **AH, AL, BH, BL, CH, CL, DH & DL**
- These registers can be used individually for temporary storage of 8-bit data.
- The AL register is also called the accumulator
- Certain pairs of these general purpose registers can be used together to store 16-bit data words

✓ **AH-AL pair --- AX register**

✓ **BH-BL pair --- BX register**

✓ **CH-CL pair --- CX register**

✓ **DH-DL pair --- DX register**




General purpose registers

# General Purpose Registers

- **AX Register**: AX register is also known as accumulator register that stores operands for arithmetic operation like divided, rotate.
- **BX Register**: This register is used as an offset storage for forming physical addresses in case of certain addressing modes.
- **CX Register**: It is used as a default counter in case of string and loop instructions.
- **DX Register**: DX register is used to contain I/O port address for I/O instruction.

# Pointer and Index Registers

- Stack Pointer (SP)
- Base Pointer (BP)
- Source Index (SI)
- Destination Index (DI)

# Pointer and Index Registers

- Stack Pointer (SP):
  - The function of SP is same as the function of SP in Intel 8085.
  - It stores the address of top element in the stack.
- BP, SI & DI are used

✓ for temporary data storage just as the general purpose registers

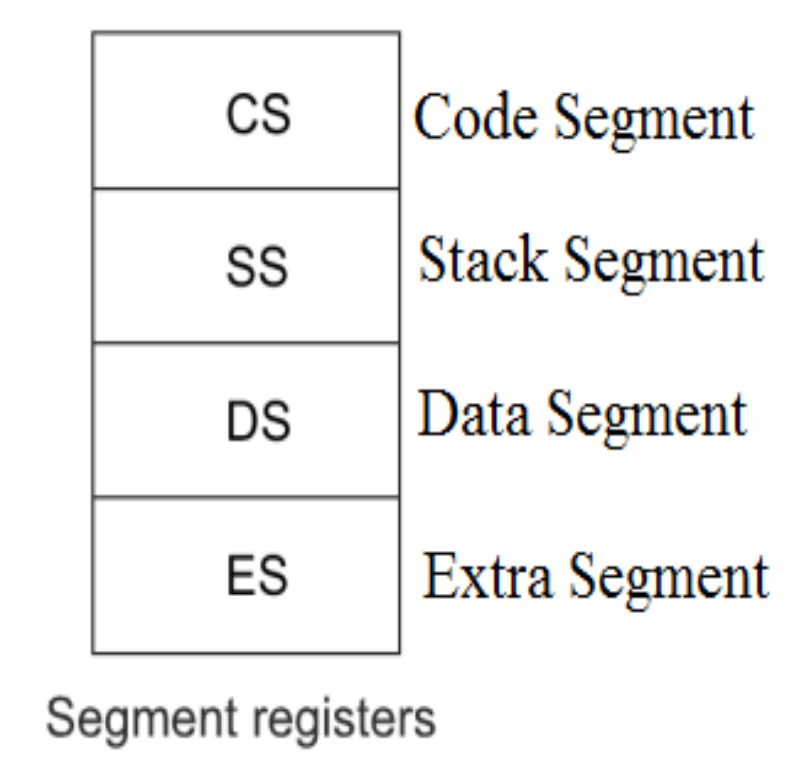
✓ in memory address computation. (later)

# Bus Interface Unit

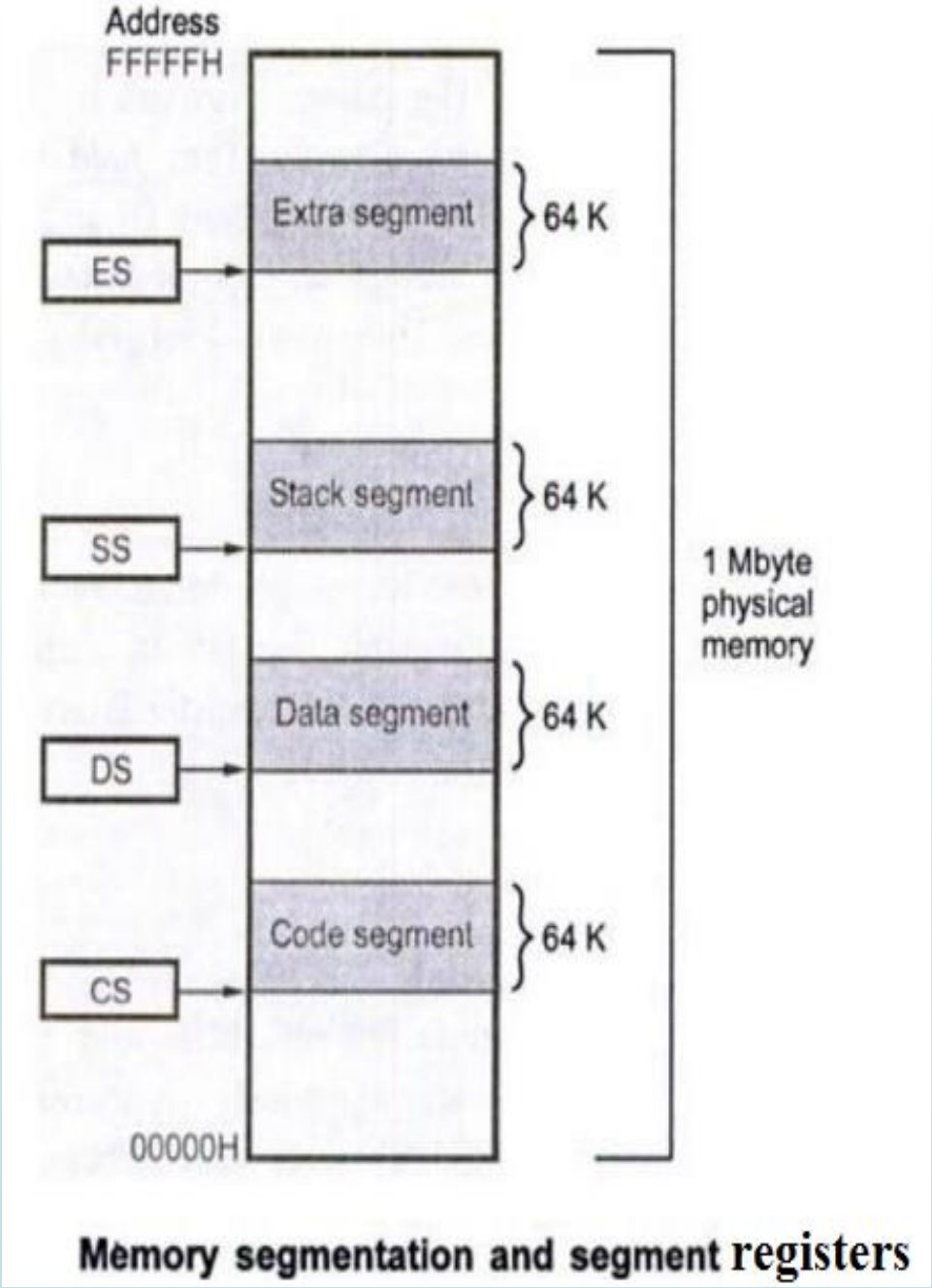
- **BIU contains**

1. **Instruction Byte Queue (6 bytes long)**
2. **Segment Registers**
3. **Instruction Pointer**
4. **Circuit for physical address calculations**

# Segment Registers

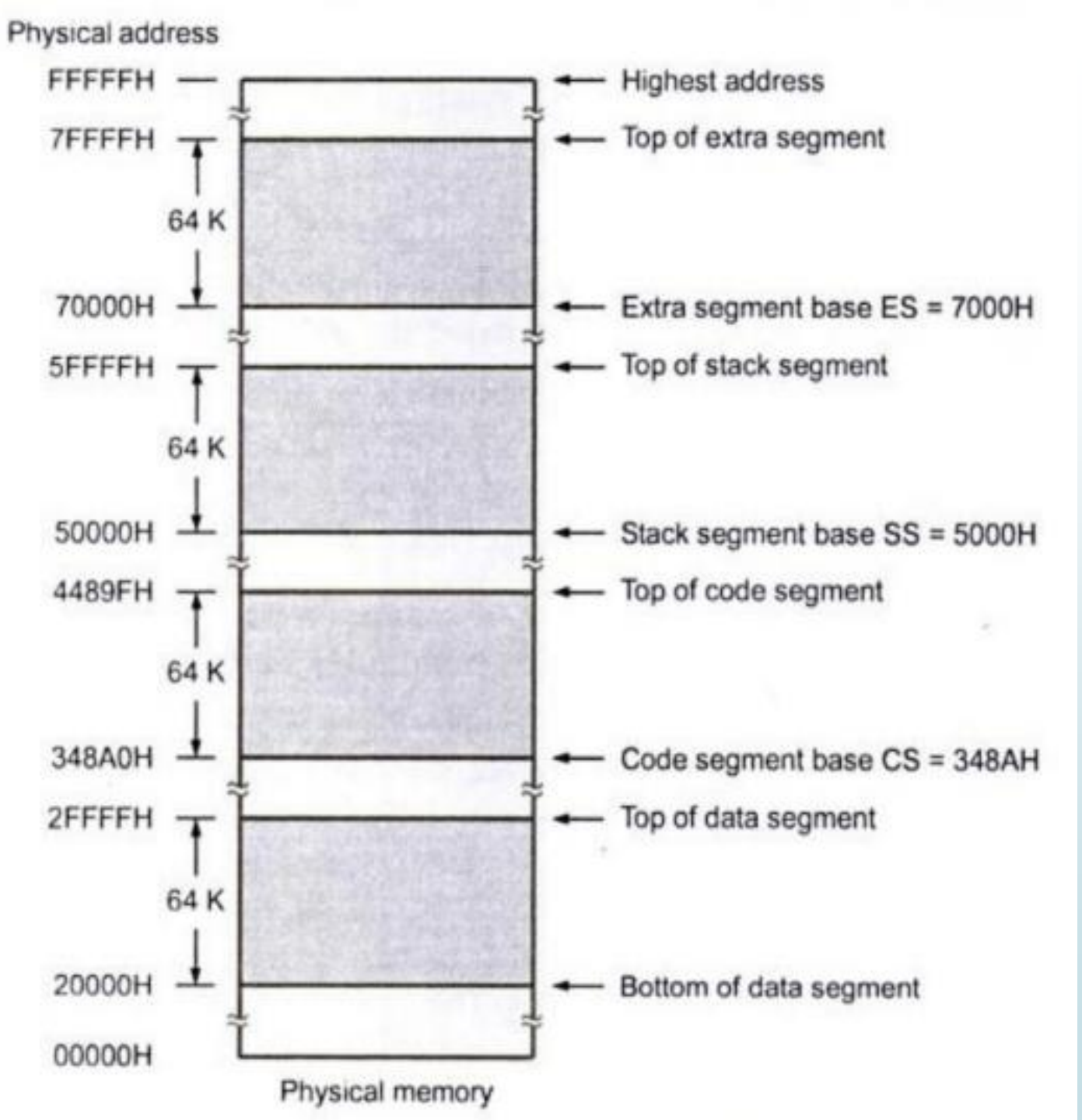



- Unlike 8085, the 8086 addresses a segmented memory
- A segment register points to the starting address of a memory segment.



# Memory Segmentation

- One way four 64-Kbyte segments might be positioned within the 1-Mbyte address space of an 8086 as shown in the figure



# Cont.

- The complete physical address which is **20-bits** long is generated using **segment registers** and **offset registers**, each 16 bits long.

| | | | | |
|---|---|---|---|---|
| Segment address | ⟶ | 1005H | | |
| Offset address | ⟶ | 5555H | | |
| Segment address | ⟶ | 1005H | ⟶ | 0001 0000 0000 0101 |
| Shifted by 4 bit positions left | | | ⟶ | 0001 0000 0000 0101 0000 |
| | | | + | |
| Offset address | | | ⟶ | 0101 0101 0101 0101 |
| Physical address | | | ⟶ | 0001 0101 0101 1010 0101 |
| | | | | 1 5 5 A 5 |

# Segment Registers

- Four Segment registers in the BIU are used **to hold the upper 16 bits of the starting addresses** of four memory segments that the 8086 is working with at a particular time

# Segment Registers

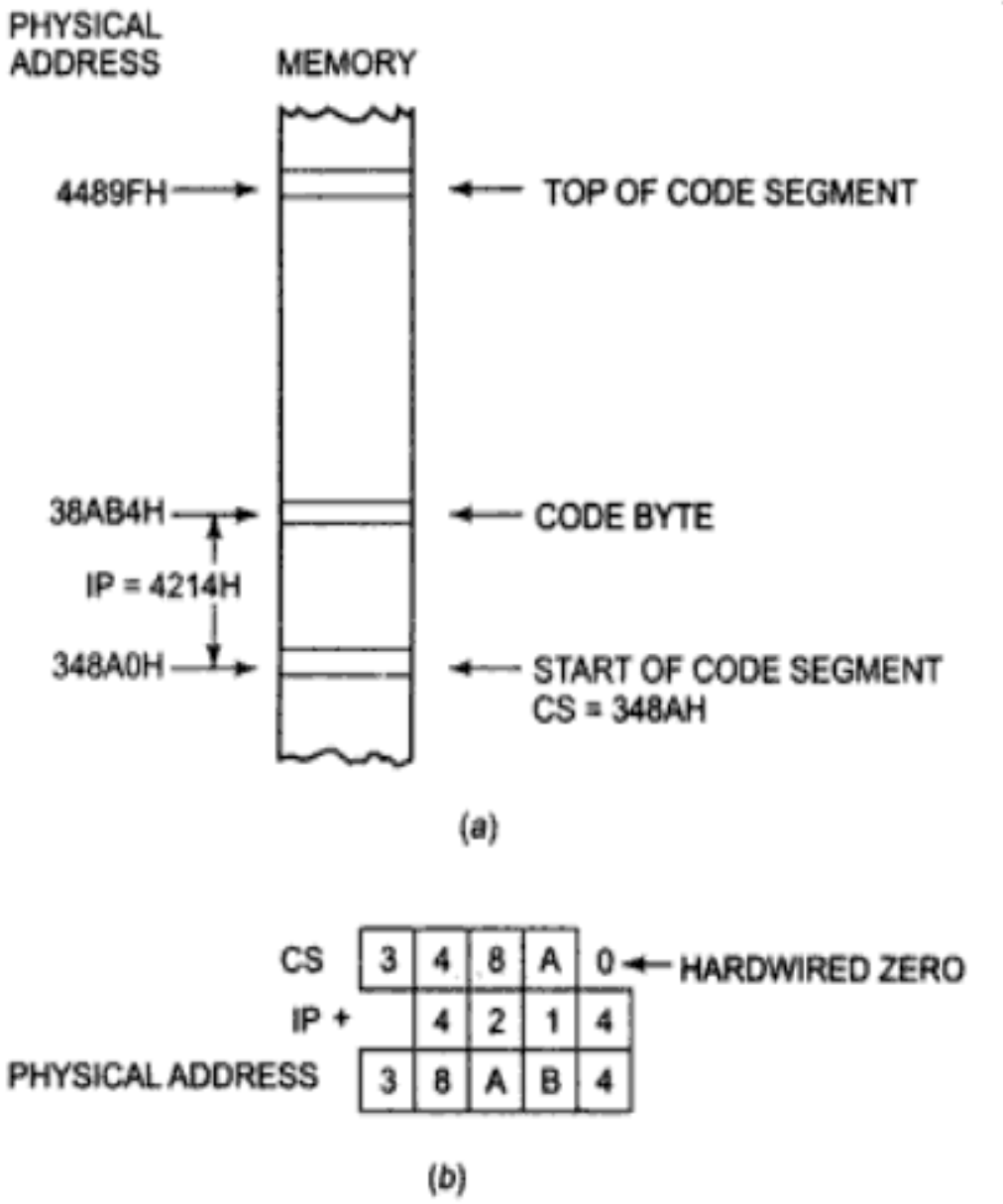
- **Code Segment Register**: Used to hold the upper 16 bits of the starting address of code segment from which the BIU is currently fetching instruction code bytes
- **Data Segment Register & Extra Segment Register**: Used to hold the upper 16 bits of the starting addresses of data segment and extra segment that are used for data
- **Stack Segment Register**: Used to hold the upper 16 bits of the starting addresses of stack segment which is used to store stack data

# Base Pointer (BP), Source Index (SI), Destination Index (DI) Registers

- BP : We can use it instead of SP for accessing the stack using based addressing mode.
- ✓ In this case, the 20-bit physical stack address is calculated from BP and SS
- SI : It can be used to hold the offset of source data in the data segment
- ✓ In this case, the 20-bit physical data address is calculated from SI and DS
- DI: It is used to store the offset of destination in data or extra segment
- ✓ String instructions always use ES and DI to determine the 20-bit physical address for the destination

# Instruction Pointer (IP)

- The Instruction Pointer (IP) in 8086 acts as a Program Counter.
- IP register holds the 16-bit address or offset of the next code byte within the code segment
- ✓ The value contained in the IP is referred to as an offset because the 20-bit physical code byte address is calculated from IP and CS



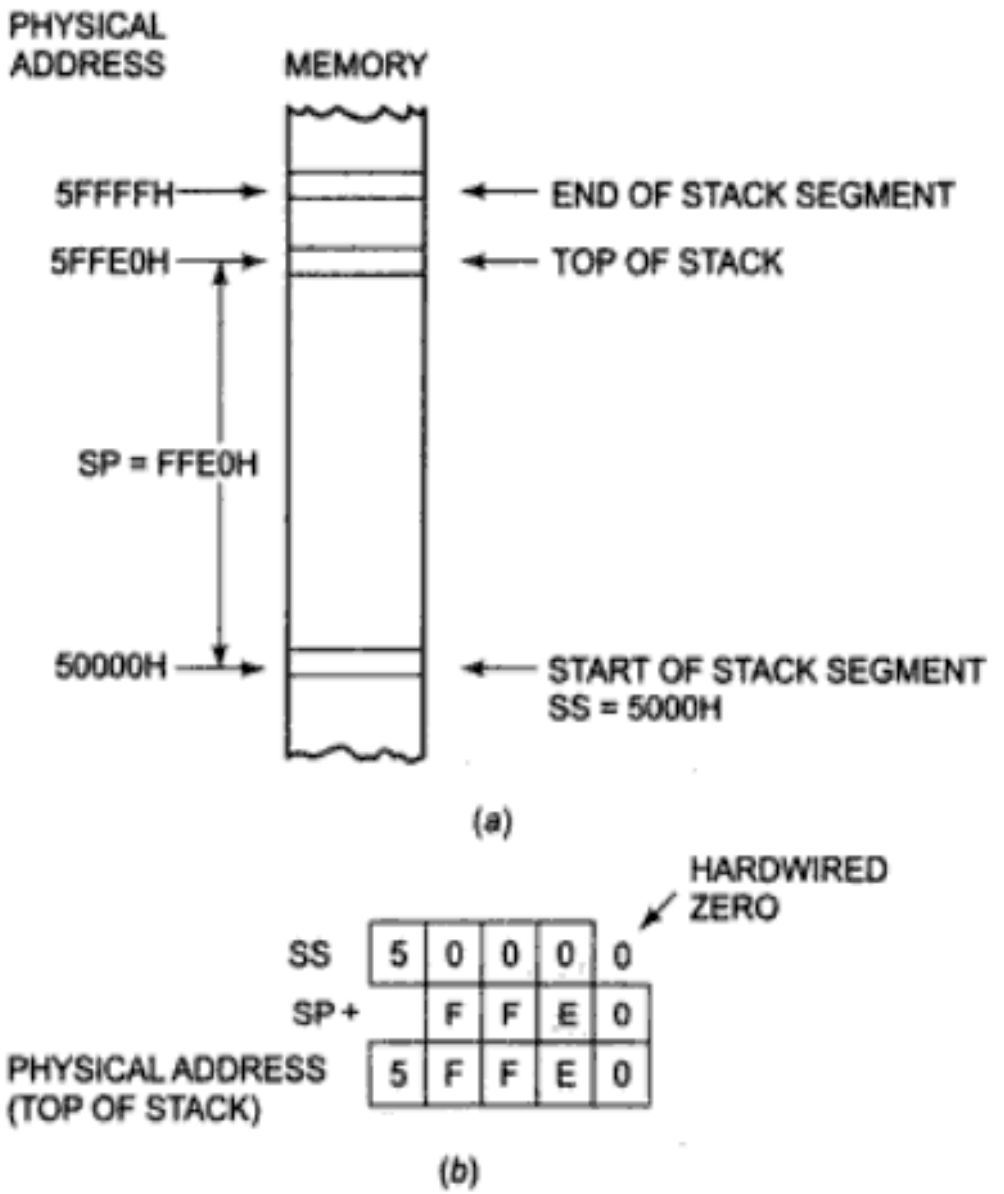

*Addition of IP to CS to produce the physical address of the code byte, (a) Diagram, (b) Computation.*

# Cont.

- The instruction pointer register contains a 16-bit offset which tells, where in that 64-Kbyte code segment the next instruction byte is to be fetched from.

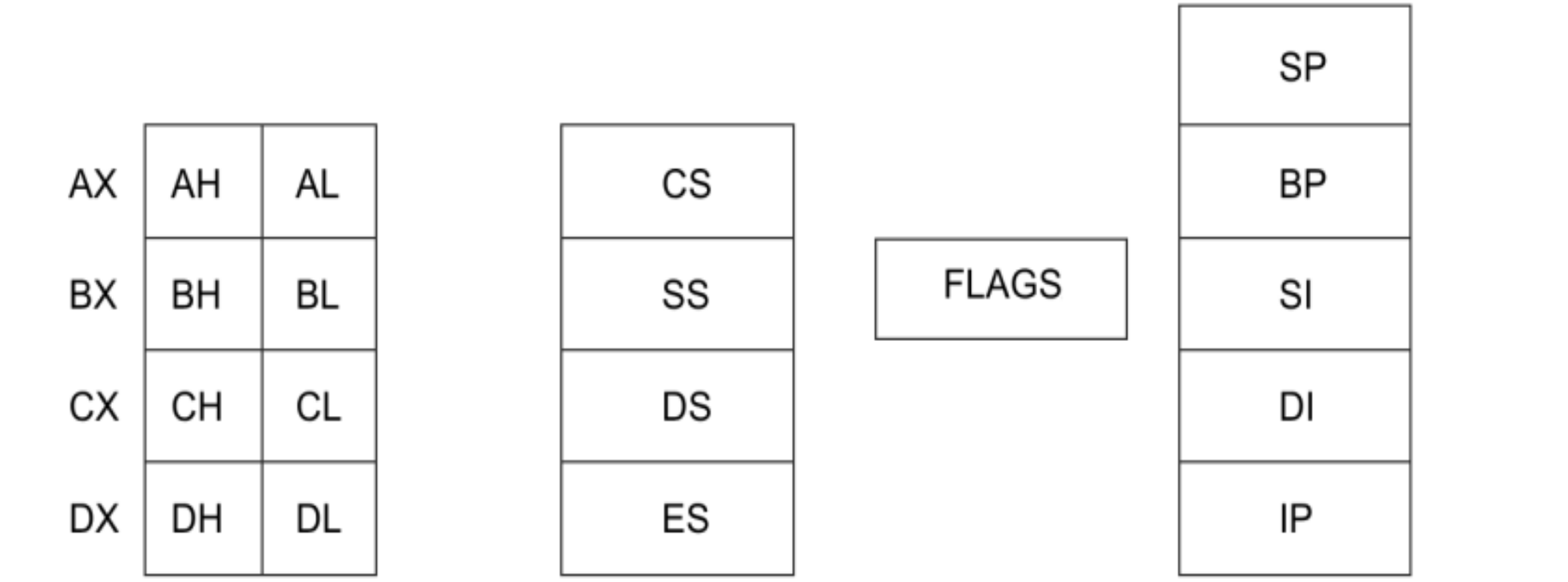
# Stack Pointer Register

- The SP register in the EU holds the 16-bit offset from the start of the stack segment to the top of the stack
- For stack operation, the content of the SS register and the content of the SP register are used to produce the 20-bit physical address.



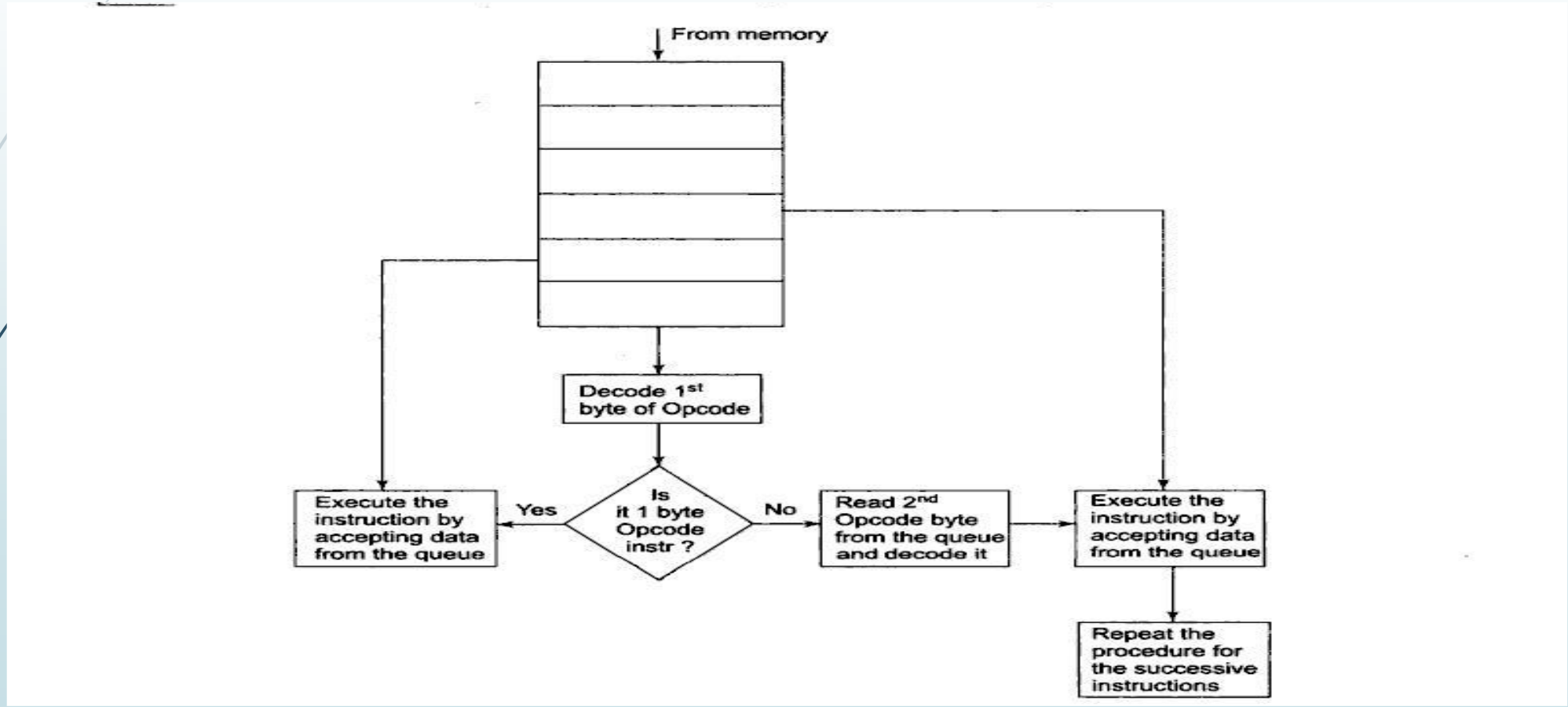

*Addition of SS and SP to produce the physical address of the top of the stack. (a) Diagram, (b) Computation.*

# Register Organization of 8086




*Register organisation of 8086*

# The Queue Operation



# Queue Operation cont…

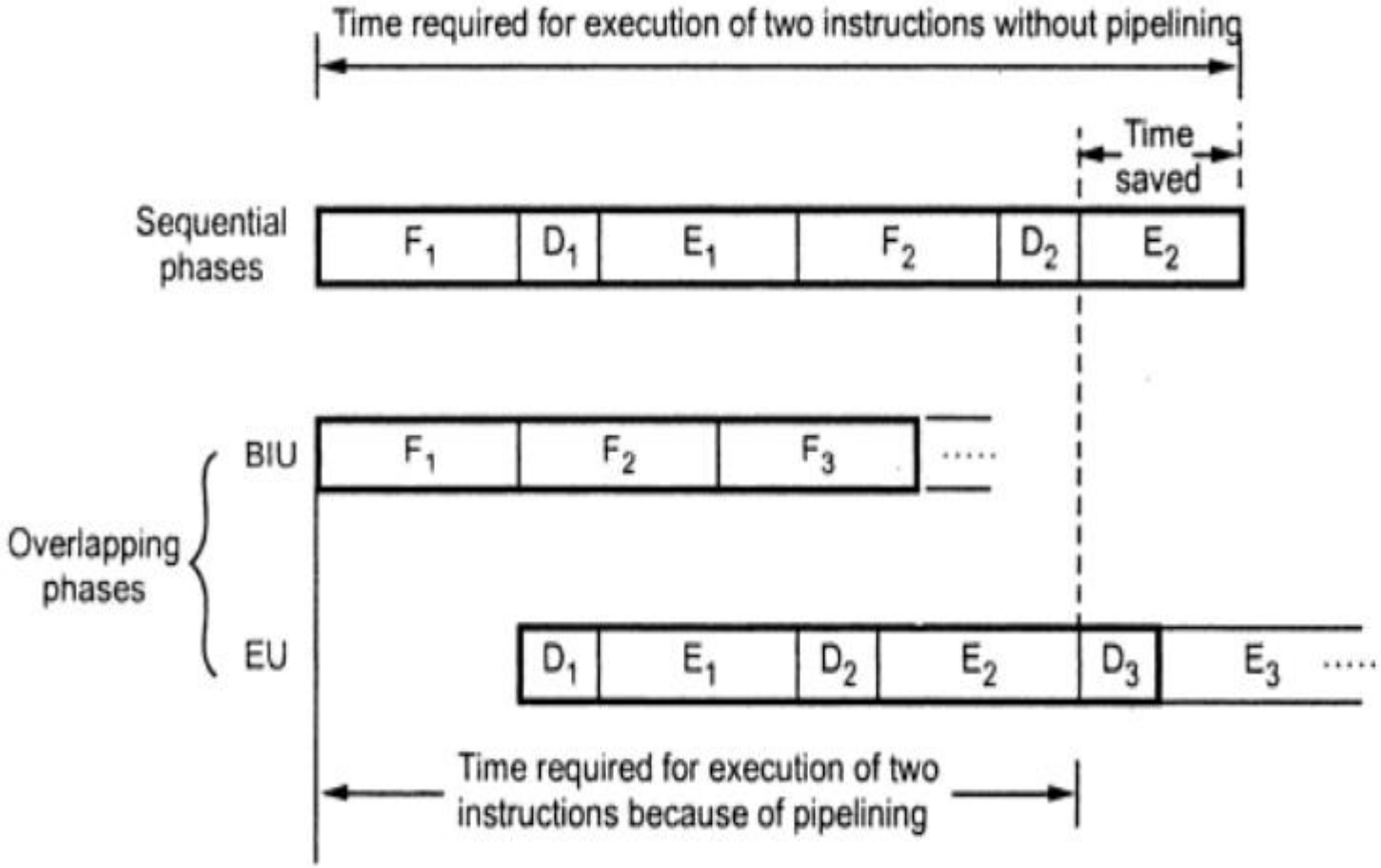
- The 8086 has a 6-byte instruction prefetch queue.
- Thus even the largest (6-byte) instruction can be prefetched from the memory and stored in the prefetched queue.
- This results in a faster execution of the instruction.
- In the beginning, the CS:IP is loaded with the required address from which the execution is to be started.
- Initially, the queue will be empty and 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.
- When the first byte from queue goes for decoding and interpretation, one byte in the queue becomes empty and subsequently the queue is updated.
- The microprocessor does not perform the next fetch operation till at least two bytes of the instruction queue are emptied.

# Queue Operation cont…

- 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.
- The next byte after the instruction is completed is again the first opcode byte of the next instruction.
- A similar procedure is repeated till the complete execution of the program.
- The fetch operation of the next instruction is overlapped with the execution of the current instruction.
- While the execution unit is busy in executing an instruction, after it is completely decoded, the bus interface unit may be fetching the bytes of the next instruction from memory, depending upon the queue ststus.

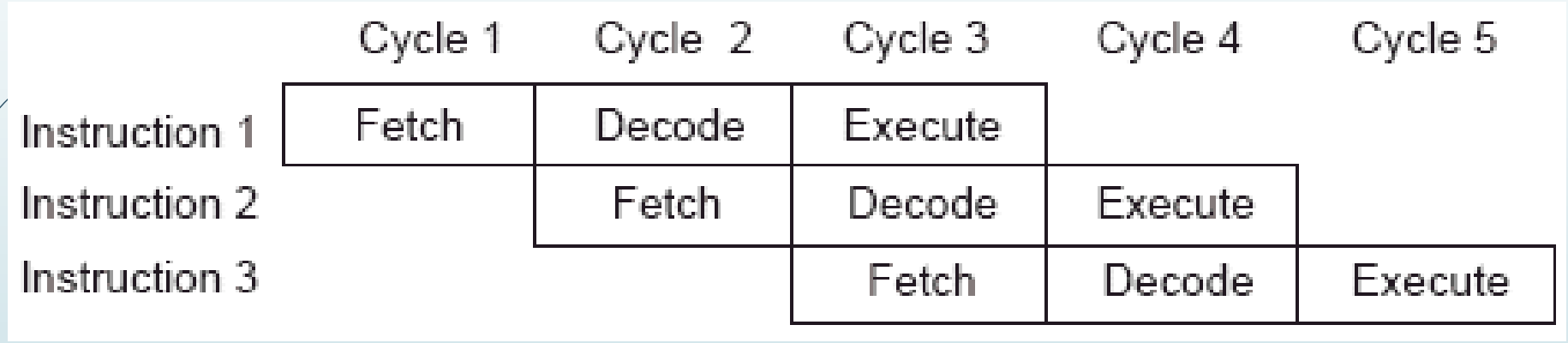
# Instruction Byte Queue

- To increase the execution speed, BIU fetches as many as six instruction bytes ahead to time from memory.
- All six bytes are then held in first-in-first-out 6-byte register called instruction queue.
- Queue operates on the principle first in first out (FIFO)
- Fetching the next instruction while the current instruction executes is called pipelining

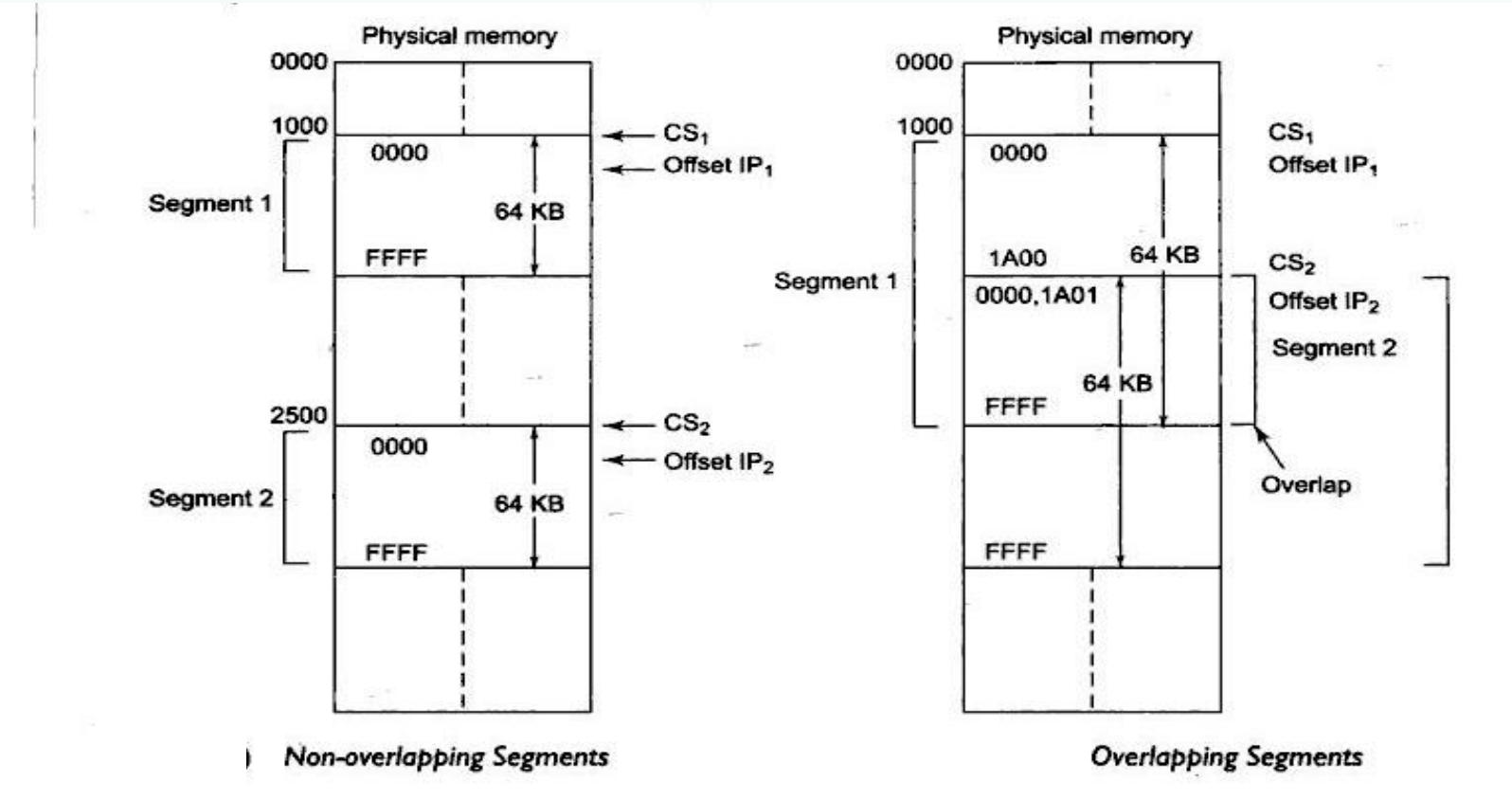



Pipelining

# Pipelining


Cycle 1
Cycle 2
Cycle 3
Cycle 4
Cycle 5
Instruction 1
Fetch
Decode
Execute
Instruction 2
Fetch
Decode
Execute
Instruction 3
Fetch
Decode
Execute

# Memory Segmentation



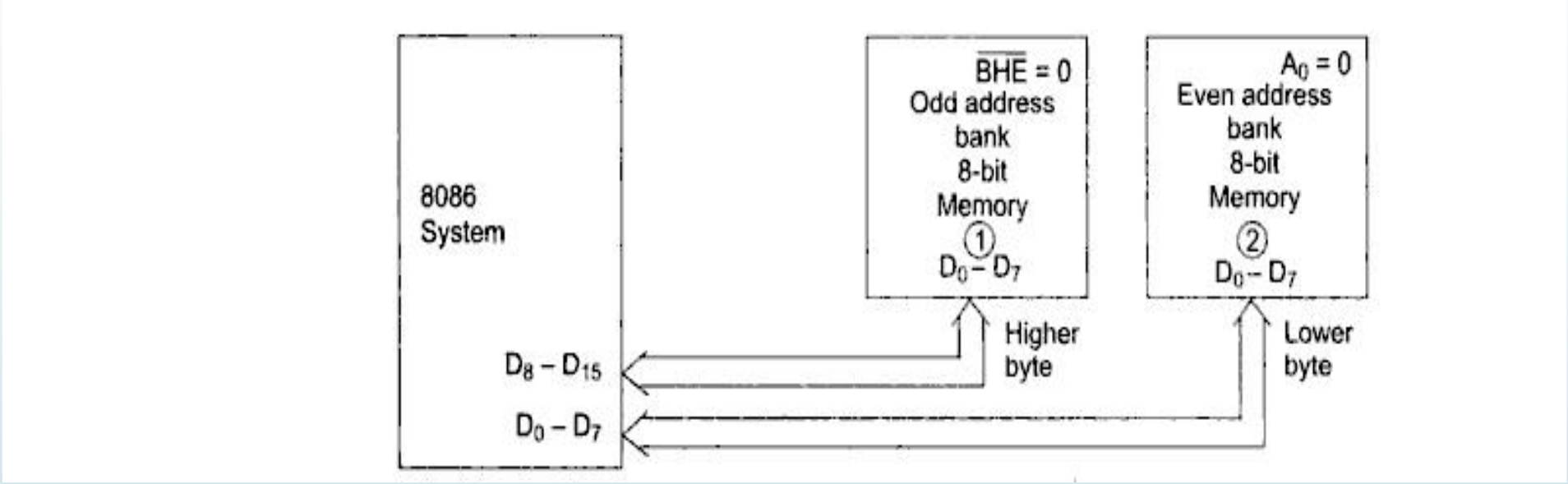
# Memory Segmentation cont…

- The CPU 8086 is able to address 1Mbytes of physical memory.
- The complete 1Mbytes memory can be divided into 16 segments, each of 64Kbytes size.
- The address of the segments may be assigned as 0000H t0 F000H respectively.
- The offset address values are from 0000H t0 FFFFH so that the physical addresses range from 00000H to FFFFFH.
- In the above said case, the segments are called non-overlapping segments.
- If another segment starts before 64Kbytes of the first segment, the two segments are said to be overlapping segments.
- The area of memory from the start of the second segment to the possible end of the first segment is called an overlapped segment area.

# Memory Segmentation cont…

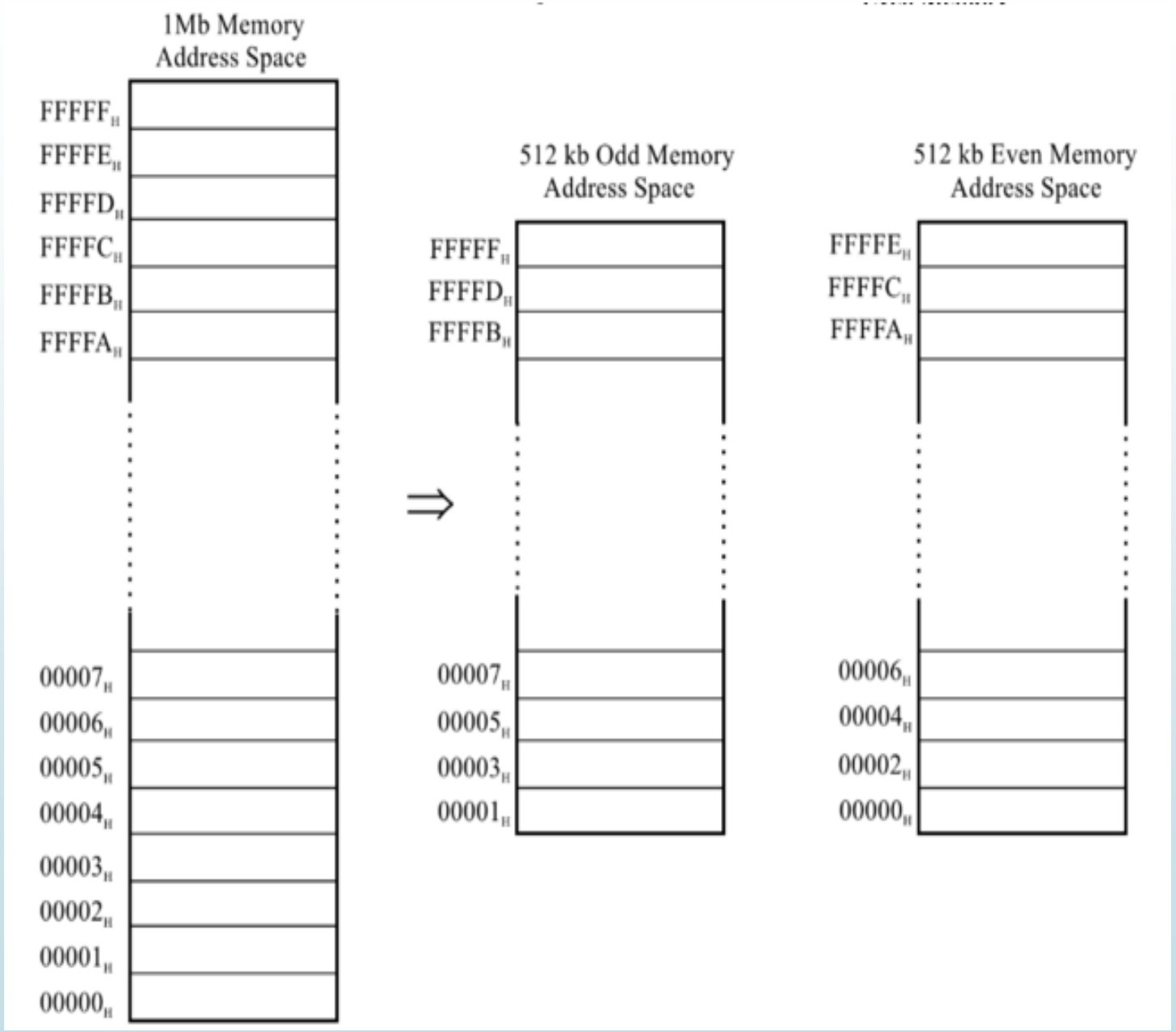
- The location lying in the overlapped area may be addressed by the same physical address generated from two different sets of segment and offset addresses.
- In the overlapped area locations Physical Address = $CS_1+IP_1 = CS_2+IP_2$
- The main advantages of the segmented memory scheme are as follows:
  - Allows the memory capacity to be 1Mbytes 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 (segment) of memory, for data and code protection
  - Permits a program and/or its data to be put into different areas of memory each time the program is executed, i.e. provision for relocation is done.

# Physical Memory Organisation

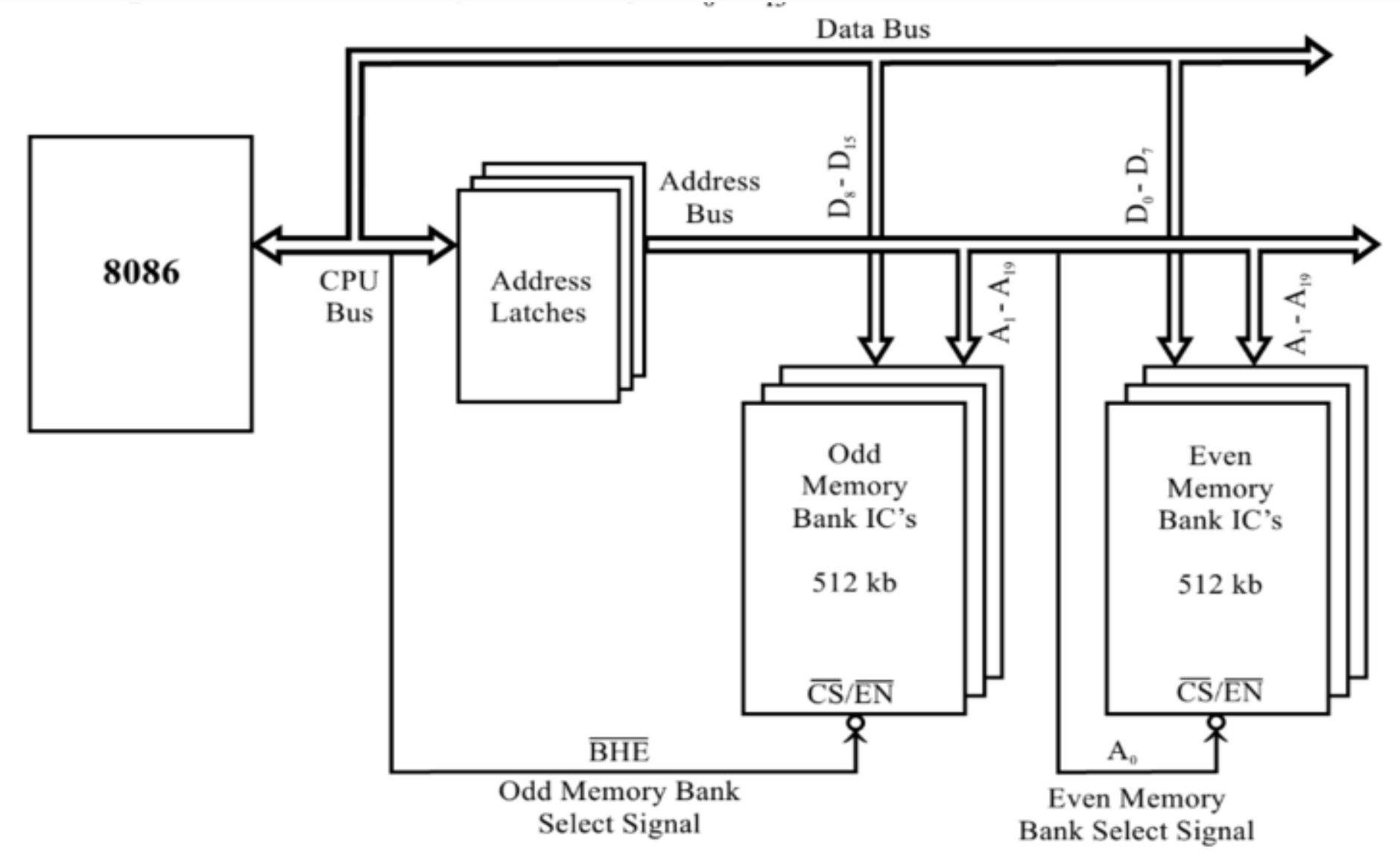



- Certain locations in memory are reserved for specific CPU operations. The locations from FFFF0H to FFFFFH are reserved for operations including jump to initialisation programme and I/O-processor initialisation.
- The locations 00000H to 003FFH are reserved for *interrupt vector table*.

# Organization of even and odd memory banks in an 8086 based system



# Addressing Memory



# Status of $A_0$ & $\overline{BHE}$ during Memory Access

| Memory bank | Operand type | Status of | | Data lines used for memory access | No. of bus cycle |
|---|---|---|---|---|---|
| | | $A_0$ | $\overline{BHE}$ | | |
| Even | Byte | 0 | 1 | $D_0$ - $D_7$ | One |
| Odd | Byte | 1 | 0 | $D_8$ - $D_{15}$ | One |
| Even, Odd | Word | 0 | 0 | $D_0$ - $D_{15}$ | One |
| Odd | Word | 1 | 0 | $D_8$ - $D_{15}$ | First cycle |
| Even | | 0 | 1 | $D_0$ - $D_7$ | Second cycle |

# References


- Advanced Microprocessors and Peripherals by A K Ray & K M Bhurchandi
- Microprocessors and Interfacing by Douglas V Hall & SSSP Rao