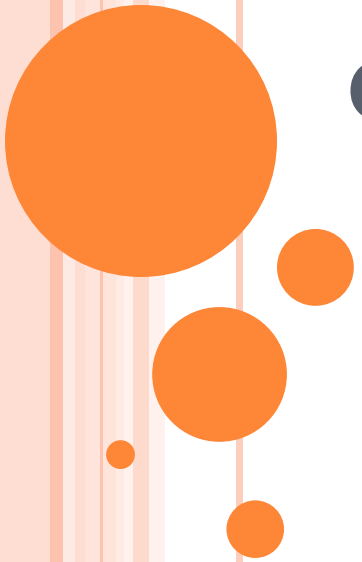


CHAPTER 2

The 8086 microprocessor architecture

CONTENTS

- 8086 architecture
  - General Information
  - Functional Diagram
  - Register Organization
  - Memory segmentation
  - Memory addresses
  - Signal descriptions
  - Minimum and Maximum mode signals
  - Read Write cycles
  - Timing diagrams
  - Interrupt structure

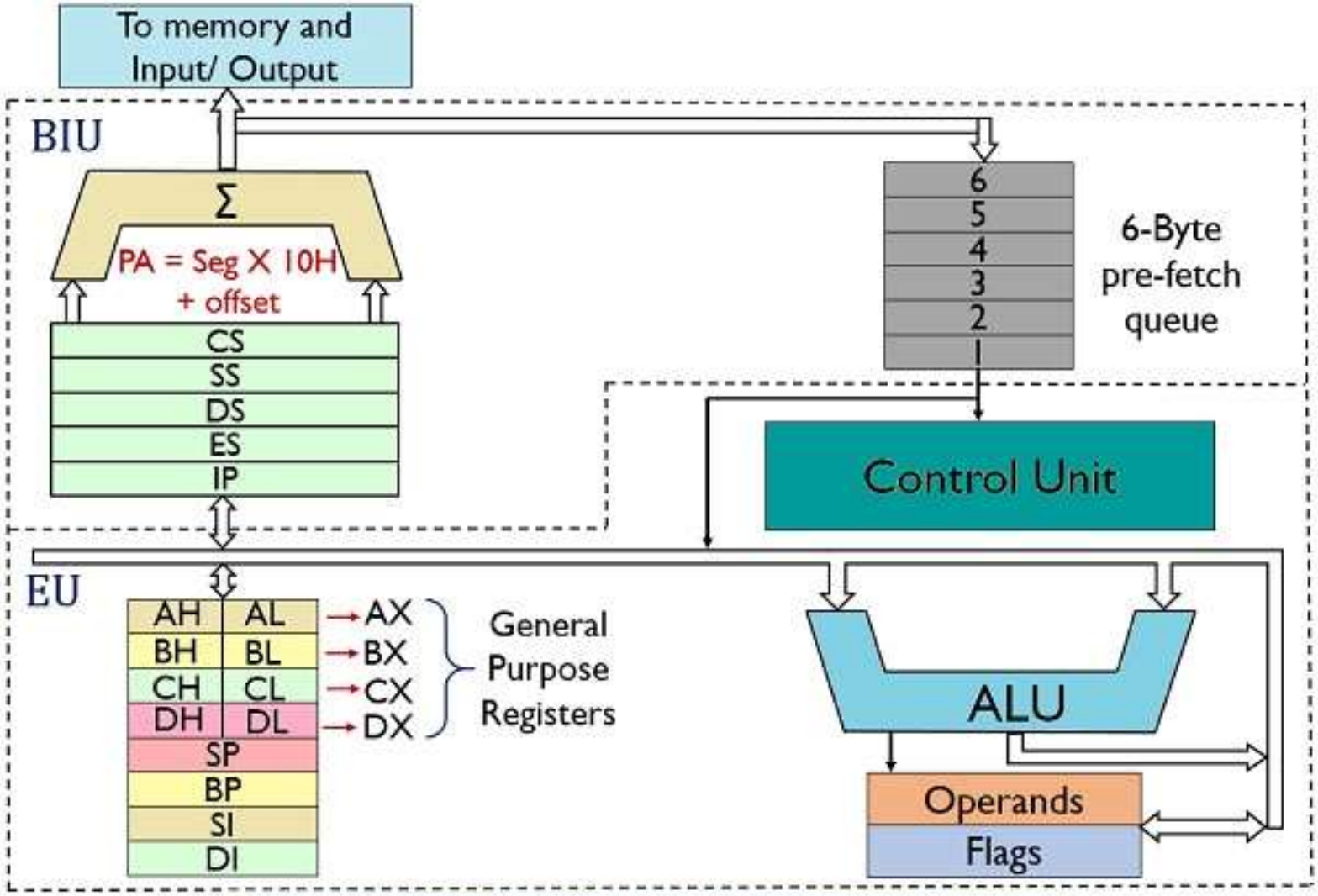
FEATURES 8086

- **16-bit Microprocessor**
  - It's ALU, internal registers works with 16 bit binary word.
- **20 bit address bus**
  - can address up to 2^{20}= 1 MB memory locations.
  - AD0- AD15 and A16 – A19.
- **16 bit data bus.**
  - It can read or write data to a memory/port either 16 bits or 8 bit at a time.
- It can support up to **2^{16} = 64K I/O** ports.
- **Fourteen** 16 -bit **registers**

FEATURES 8086 CONT..

- Frequency range : **6-10 MHz**
- **Pre-fetch up to 6 instructions** bytes from memory and queues
- It requires **+5V power** supply.
- A **40 pin dual in line** package.
- **Two modes**
  - Minimum mode
  - Maximum mode
- Support **multiprogramming**

Block Diagram



Block Diagram of 8086 Microprocessor

BLOCK DIAGRAM CONT..

- **Two Blocks**
  - Bus Interface Unit(BUI)
  - Execution Unit (EU)
- **Bus Interface Unit(BUI)**
  - Contains Instruction queue, Segment registers, Instruction pointer, Address adder.
  - Performs all bus operations
- **Execution Unit (EU)**
  - Contains Control circuitry, Instruction decoder, ALU, Pointer and Index register, Flag register and general purpose register
  - Executes instructions from the instruction queue.

BLOCK DIAGRAM CONT..

- Bus Interface Unit(BUI)
  - Has 16 bit bidirectional data bus
  - Has 20 bit address bus.
  - responsible for **performing all external bus operations**.
    - Instruction fetch and Instruction queuing
    - Operand fetch and storage
    - Address relocation and Bus control.
  - Has **instruction stream queue** to implement pipeline architecture.
    - prefetching instructions are held in its FIFO queue.
  - With its 16 bit data bus, the BIU **fetches two instruction bytes in a single memory cycle**.

BLOCK DIAGRAM CONT..

- Bus Interface Unit(BUI)
  - Contains a **dedicated adder**
    - used to generate the 20 bit physical address
    - This address is formed by adding an appended 16 bit segment address and a 16 bit offset address.
    - example:
      - The physical address of the next instruction to be fetched is formed by combining the current contents of the code segment CS register and the current contents of the instruction pointer IP register.
  - The BIU is also responsible for **generating bus control signals** such as those for memory read or write and I/O read or write.

BLOCK DIAGRAM CONT..

- Execution Unit (EU)
  - responsible for **decoding and executing** all instructions.
  - **extracts instructions from the top of the queue** in the BIU, **decodes** them, **generates operands** if necessary, passes them to the BIU
  - **tests the status and control flags and updates them**
  - **If the queue is empty, the EU waits** for the next instruction byte to be fetched and shifted to top of the queue.

Register Organization

- **Fourteen registers** that are accessible to the programmer.
- It is divided into four groups.
  - Four **General purpose** registers
  - Four **Index/Pointer** registers
  - Four **Segment** registers
  - Two **Other registers**

Register Organization cont..

- Four General purpose registers
  - Accumulator -AX
  - Base -BX
  - Count -CX
  - Data -DX
- 16- bit each

15-0	15-8	7-0
AX	AH	AL
BX	BH	BL
CX	CH	CX
DX	DH	DL

REGISTER ORGANIZATION CONT..

- **AX (Accumulator):**
  - **AX** is used as **16-bit accumulator**.
  - The lower 8-bits of AX are **AL**
  - The higher 8-bits are **AH**.
  - **AL** can be used as an **8-bit accumulator.**
  - Used in arithmetic, logic and data transfer operations.
  - For multiplication and division operations, one of the numbers must be placed in AX or AL.

	15-8	7-0
AX	AH	AL
BX	BH	BL
CX	CH	CX
DX	DH	DL

REGISTER ORGANIZATION CONT..

	15-8	7-0
AX	AH	AL
BX	BH	BL
CX	CH	CX
DX	DH	DL

- **BX (Base Register):**
  - **16 bit register**,
    - BL indicates the lower 8-bits of BX and
    - BH indicates the higher 8-bits of BX.
  - Used as **address register** to form physical address in case of certain addressing modes
    - ex: indexed and register indirect.

REGISTER ORGANIZATION CONT..

- **CX (Count Register)**
  - 16 bit register
  - used as **default counter** in case of string and loop instructions.
  - can also be used **as a counter** in string manipulation and shift/rotate instruction.

	15-8	7-0
AX	AH	AL
BX	BH	BL
CX	CH	CX
DX	DH	DL

REGISTER ORGANIZATION CONT..

- **DX (Data Register):**
  - a **general purpose** register
  - used as an **implicit operand** or **destination** in case of a few instructions.
  - can also be used **as a port number** in I/O operations.
  - 16- bit
  - DH or DL

	15-8	7-0
AX	AH	AL
BX	BH	BL
CX	CH	CX
DX	DH	DL

REGISTER ORGANIZATION CONT..

- Four Index/Pointer registers
- **The index registers**
  - can be used for arithmetic operations
  - usually concerned with the **memory addressing modes** (indexed, base indexed and relative base indexed).
  - **SI (Source Index)**
    - 16-bit register.
    - used **to store the offset of source data** in data segment.
  - **DI (Destination Index)**
    - 16-bit register.
    - **to access the memory locations in Data or Extra** Segment

REGISTER ORGANIZATION CONT..

- Four Index/Pointer registers
- **Pointer Registers**
  - contains the **offset of data**(variables, labels) and instructions from their base segments
  - 8086 microprocessor contains **three pointer registers**.
  - **SP (Stack Pointer):**
    - points the program stack that means SP **stores the base address of the Stack Segment**.
  - **BP (Base Pointer):**
    - **points stack segment**.
    - can use BP **to access data in the other segments** also.

REGISTER ORGANIZATION CONT..

- Four Segment registers to access 1 megabyte of memory
  - **CS (Code Segment):**
    - a 16-bit register (64Kb)
    - where the **executable program is stored**.
    - CS register **cannot be changed directly**.
    - The CS register is **automatically updated** during far jump, far call and far return instructions.

REGISTER ORGANIZATION CONT..

- Four Segment registers to access 1 megabyte of memory
  - **Stack segment (SS)**
    - a 16-bit register
    - used for **addressing stack segment** of the memory (64kb) where stack data is stored.
    - SS register **can be changed directly** using POP instruction.

REGISTER ORGANIZATION CONT..

- Four Segment registers to access 1 megabyte of memory
  - **Data segment (DS)**
    - a 16-bit register that points the data segment of the memory (64kb)
      - where the **program data is stored**.
    - can be changed directly using POP and LDS instructions.

REGISTER ORGANIZATION CONT..

- Four Segment registers to access 1 megabyte of memory
  - **Extra segment (ES):**
    - a 16-bit register that also points the data segment of the memory (64kb)
      - where the program data is stored.
    - ES register can be changed directly using POP and LES instructions.

REGISTER ORGANIZATION CONT..

- Two Other registers
  - Instruction pointer
  - Flag register
- **IP (Instruction Pointer):**
  - holds the address of the next instruction to be fetched from memory.
  - It contains the **offset of the next word** of instruction code instead of its actual address

REGISTER ORGANIZATION CONT..

- Flag register
  - Indicates some condition produced following instruction execution
  - 16 bit register only nine of them active
    - F-Overflow flag, D – Directional flag, I- Interrupt flag, T- Trap flag, S -Sign flag , Z - zero flag, AC - auxiliary carry flag, P - parity flag & CY - carry flag

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	F	D	I	T	S	Z	X	AC	X	P	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - **control flags**
    - Enable or disable certain CPU operations.
    - The programmer can set/reset
    - D-Directional flag, I- Interrupt flag, T- Trap flag,

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	OF	D	I	T	S	Z	X	AC	X	P	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - **control flags**
    - **The Direction Flag (D):**
    - Affects the direction of moving data blocks
    - instructions as MOVS, CMPS and SCAS.
    - The flag values are 0 = up and 1 = down and
    - can be set/reset by the STD (set D) and CLD (clear D) instructions.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	F	D	I	T	S	Z	X	AC	X	PF	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - **control flags**
    - **The Interrupt Flag (I):**
      - indicates whether or not system interrupts can occur.
      - The flag values are 0 = disable interrupts or 1 = enable interrupts and
      - can be manipulated by the CLI (clear I) and STI (set I) instructions.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	F	D	I	T	S	Z	X	AC	X	PF	X	CY

REGISTER ORGANIZATION CONT..

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

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	F	D	I	T	S	Z	X	AC	X	PF	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - **status flags**
    - Which reflect the result of executing an instruction.
    - The programmer cannot set/reset these flags directly.
    - OF-Overflow flag, S -Sign flag , Z - zero flag, AC - auxiliary carry flag, P - parity flag & CY - carry flag

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	OF	D	I	T	S	Z	X	AC	X	P	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - Status flags
    - **Overflow flag bit**
      - This flag is set when the result of a signed arithmetic operation is too large to fit in the destination register
      - Overflow can occur when adding two numbers with the same sign (i.e. both positive or both negative).
      - A value of 1 = overflow and 0 = no overflow

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	F	D	I	T	S	Z	X	AC	X	PF	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - Status flags
    - **Sign flag bit**
      - Used for indicating the sign of the data in the accumulator
        - The sign flag is set if negative (1 – negative)
        - The sign flag is reset if positive (0 –positive)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	F	D	I	T	S	Z	X	AC	X	PF	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - Status flags
    - **Zero Flag**
      - Is set if result obtained after an operation is 0
      - Is set following an increment or decrement operation of that register

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	OF	D	I	T	S	Z	X	AC	X	P	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - Status flags
    - **Carry Flag**
      - Is set if there is a carry or borrow from arithmetic operation
      - 1011 0101
        + 0110 1100

        Carry 1 0010 0001

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	OF	D	I	T	S	Z	X	AC	X	P	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - Status flags
    - **Parity Flag**
      - Is set if parity is even
      - Is cleared if parity is odd

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	OF	D	I	T	S	Z	X	AC	X	P	X	CY

REGISTER ORGANIZATION CONT..

- Flag register
  - Status flags
    - **Auxiliary Carry Flag**
      - Is set if there is a carry out of bit 3
      - Is reset if there is no carry out of bit 3

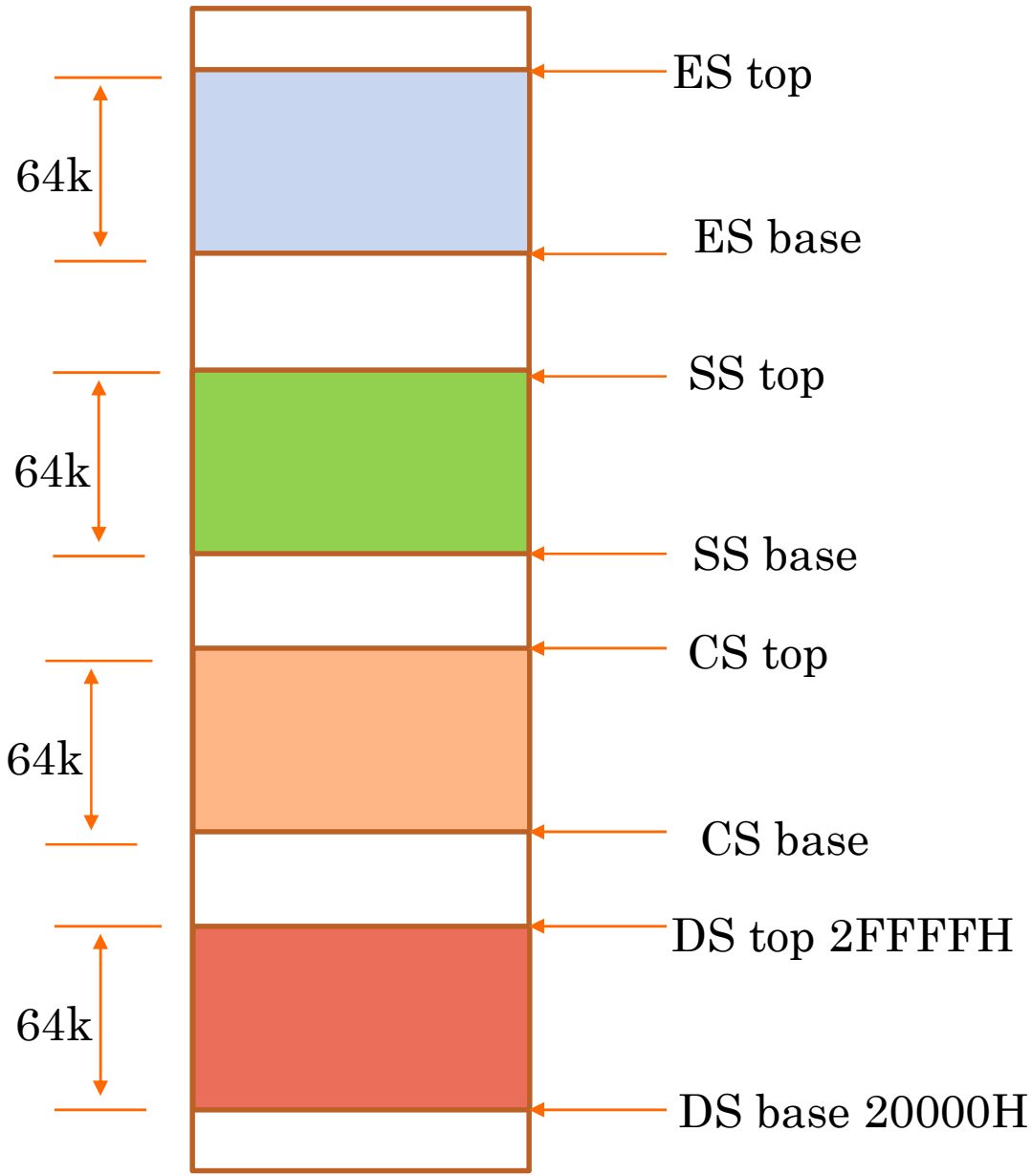
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	X	X	X	X	OF	D	I	T	S	Z	X	AC	X	P	X	CY

CHAPTER 2- LECTURE 2

MEMORY ORGANIZATION

- Linear addressing
  - One linear address for the entire memory
- Segmented addressing
  - **Small groups of logical memory section**
    - Code segment
    - Data segment
    - Stack segment
    - Extra segment
  - **Segment register holds starting/base** address of particular segment
  - **Offset** value shall be given through **pointer/index** register

MEMORY ORGANIZATION



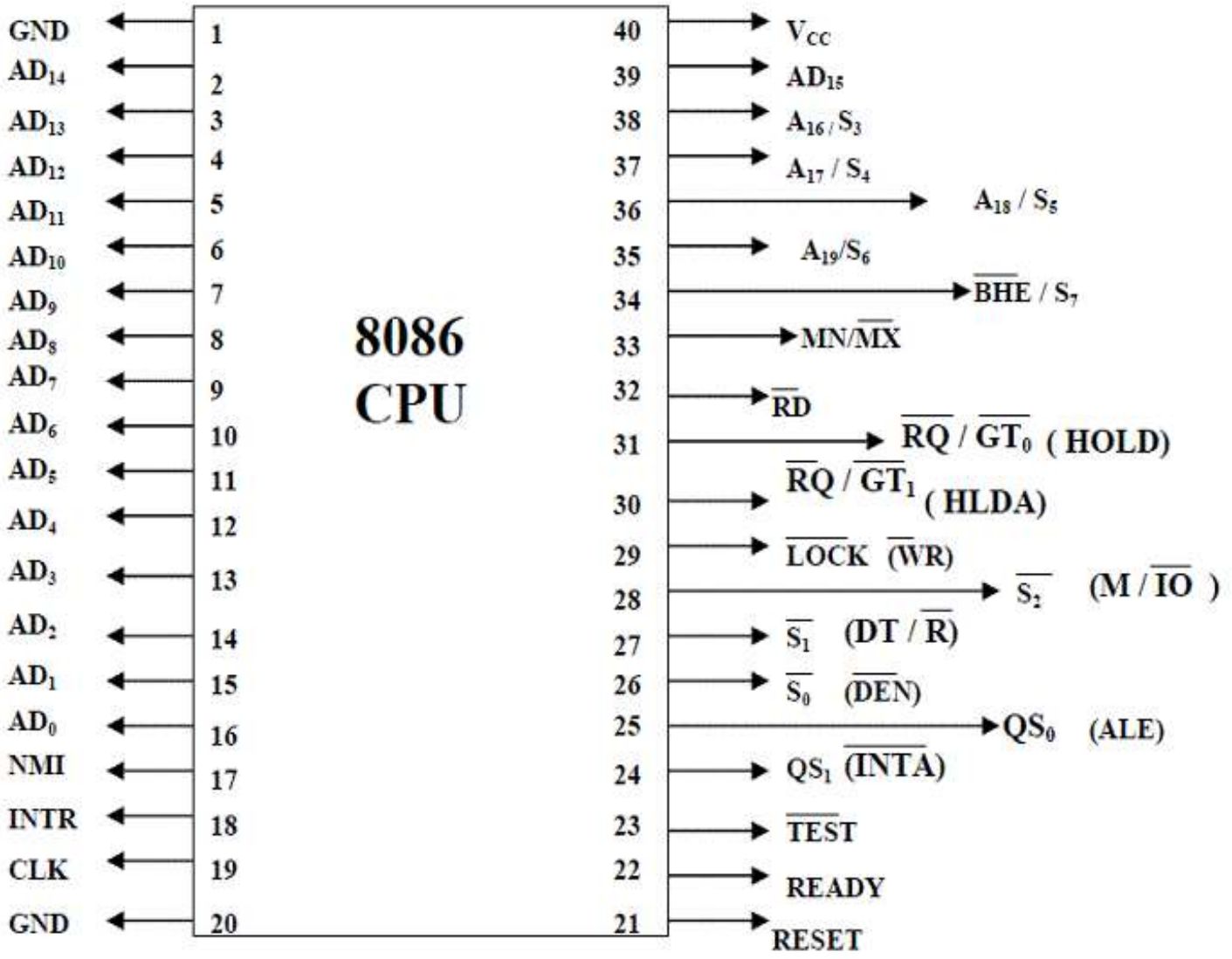
DEFAULT REGISTER VALUES

Type of memory reference	Default segment	Alternate segments	offset
Instruction fetch	CS	None	IP
Stack operation	SS	None	SP, BP
General data	DS	CS, ES, SS	Effective address
String source	DS	CS, ES, SS	SI
String destination	ES	None	DI
BX, as pointer	DS	CS, ES, SS	Effective address
BP, as pointer	SS	CS, ES, SS	Effective address

MODE OF OPERATION

- **Minimum mode**
  - All the control signals are given out by the microprocessor.
  - single microprocessor
  - Operated by strapping its MN/MX pin to logic 1.
- **Maximum mode**
  - more than one microprocessor in the system configuration.
  - the processor derives the status signal S2, S1, S0
    - **bus controller derives the control signal** using this status information.
  - operated by strapping the MN/MX pin to ground.

PIN DESCRIPTION



CONTROL SIGNALS

- Three groups
  - Minimum mode signals
  - Maximum mode signals.
  - Common signal (minimum and maximum) mode.

1. Common signal for minimum and maximum mode
   - **AD15-AD0:**
     - Multiplexed address and data signals
     - Address signal: T1 state
     - Data signal : T2, T3, Tw and T4 state.

CONTROL SIGNALS

Common signal for minimum and maximum mode

- **A19/S6, A18/S5, A17/S4, and A16/S3:**
  - Time multiplexed **address and status** lines.
  - Higher order Address (A19:A16) : during T1
  - Status signal (S6:S3)
  - During I/O operations: these signals are low.
  - S5 Interrupt enable flag
  - S6 always low
  - The S4 and S3 :current in use segment registers

S4	S3	Register
0	0	ES
0	1	SS
1	0	CS
1	1	DS

CONTROL SIGNALS

Common signal for minimum and maximum mode

- **/BHE/S7:** Bus High Enable
  - Active low
  - **Transfer of data over the higher order** (AD15-AD8) data bus.
  - **BHE is low during T1 for read, write and interrupt acknowledge cycles**, whenever a byte is to be transferred on higher byte of data bus.
  - S7 not defined yet

CONTROL SIGNALS

Common signal for minimum and maximum mode

- **/RD – Read:**
  - processor is performing memory or I/O **read operation**.
  - RD is **active low** and shows the state for T2, T3, Tw of any read cycle.
  - The signal remains **tri-stated during the hold acknowledge**.
- **READY:**
  - This is the **acknowledgement from the slow device or memory** that they have completed the data transfer.
  - the signal is **active high**.

CONTROL SIGNALS

Common signal for minimum and maximum mode

- **INTR-Interrupt Request:**
  - triggered input
  - **sampled during the last clock cycles** of each instruction to determine the availability of the request.
    - If any interrupt request is pending, the processor enters the interrupt acknowledge cycle.
  - **Maskable** .
  - active high

CONTROL SIGNALS

Common signal for minimum and maximum mode

- **TEST:**
  - examined by a '**WAIT**' instruction.
  - If the TEST pin goes low, execution will continue, else the processor remains in an idle state.
  - The input is synchronized internally during each clock cycle on leading edge of clock.
- **CLK- Clock Input:**
  - The clock input provides the basic timing for processor operation and bus control activity.
  - It's an asymmetric square wave with 33% duty cycle.

CONTROL SIGNALS

2. minimum mode operation

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

CONTROL SIGNALS

minimum mode operation

- **INTA – Interrupt Acknowledge:**
  - This signal is used as a **read strobe for interrupt acknowledge cycles**.
  - when it goes low, the processor has accepted the interrupt.
- **ALE – Address Latch Enable:**
  - This output signal indicates the availability of the valid address on the address/data lines, and is connected to latch enable input of latches.
  - This signal is active high and is never tri-stated.

CONTROL SIGNALS

- *minimum mode operation*
- **DT/R – Data Transmit/Receive:**
  - direction of data flow through the transceivers (bidirectional buffers).
  - When the processor **sends** out data, this signal is **high** and
  - when the processor is **receiving** data, this signal is **low**.
- **DEN – Data Enable:**
  - indicates the **availability of valid data** over the address/data lines.
  - It is used to enable the transceivers (bidirectional buffers)
    - to separate the data from the multiplexed address/data signal.
  - It is active from the middle of T2 until the middle of T4.
  - This is tri-stated during 'hold acknowledge' cycle.

CONTROL SIGNALS

- *minimum mode operation*
- **HOLD, HLDA- Acknowledge:**
  - When the HOLD line goes high; it indicates to the processor that another master is requesting the bus access.
  - The processor, after receiving the HOLD request, issues the hold acknowledge signal on HLDA pin, in the middle of the next clock cycle after completing the current bus cycle.
  - At the same time, the processor floats the local bus and control lines.

CONTROL SIGNALS

3. maximum mode operation

- S2, S1, and S0 – Status Lines:
  - reflect the type of operation, being carried out by the processor.

S2	S1	S0	Status
0	0	0	Interrupt Acknowledgment
0	0	1	Read IO
0	1	0	Write IO
0	1	1	HALT
1	0	0	Code access
1	0	1	Read memory
1	1	0	Write Memory
1	1	1	Passive

CONTROL SIGNALS

- *maximum mode operation*
  - **LOCK**:
    - Active low output signal
    - Prevent other processor from requesting system bus gain
    - activated by the 'LOCK' prefix instruction and remains active until the completion of the next instruction.
  - **RQ/GT0, RQ/GT1**
    - Request/Grant bi-directional pin
    - Other processor use this to request system bus gain from CPU
    - CPU use it to send acknowledgement
    - RQ/GT0 has high priority than RQ/GT1

WRITE CYCLE TIMING DIAGRAM FOR MINIMUM MODE

- The working of the minimum mode configuration system can be better described in terms of the timing diagrams rather than qualitatively describing the operations.
- The opcode fetch and read cycles are similar.
- Hence the timing diagram can be categorized in two parts,
  - the first is the timing diagram for read cycle and
  - the second is the timing diagram for write cycle.

WRITE CYCLE TIMING DIAGRAM FOR MINIMUM MODE

- The read cycle begins in T1 with the assertion of address latch enable (ALE) signal and also M / IO signal. During the negative going edge of this signal, the valid address is latched on the local bus.
- The BHE and A0 signals address low, high or both bytes. From T1 to T4 , the M/IO signal indicates a memory or I/O operation.
- At T2, the address is removed from the local bus and is sent to the output. The bus is then tri-stated. The read (RD) control signal is also activated in T2.

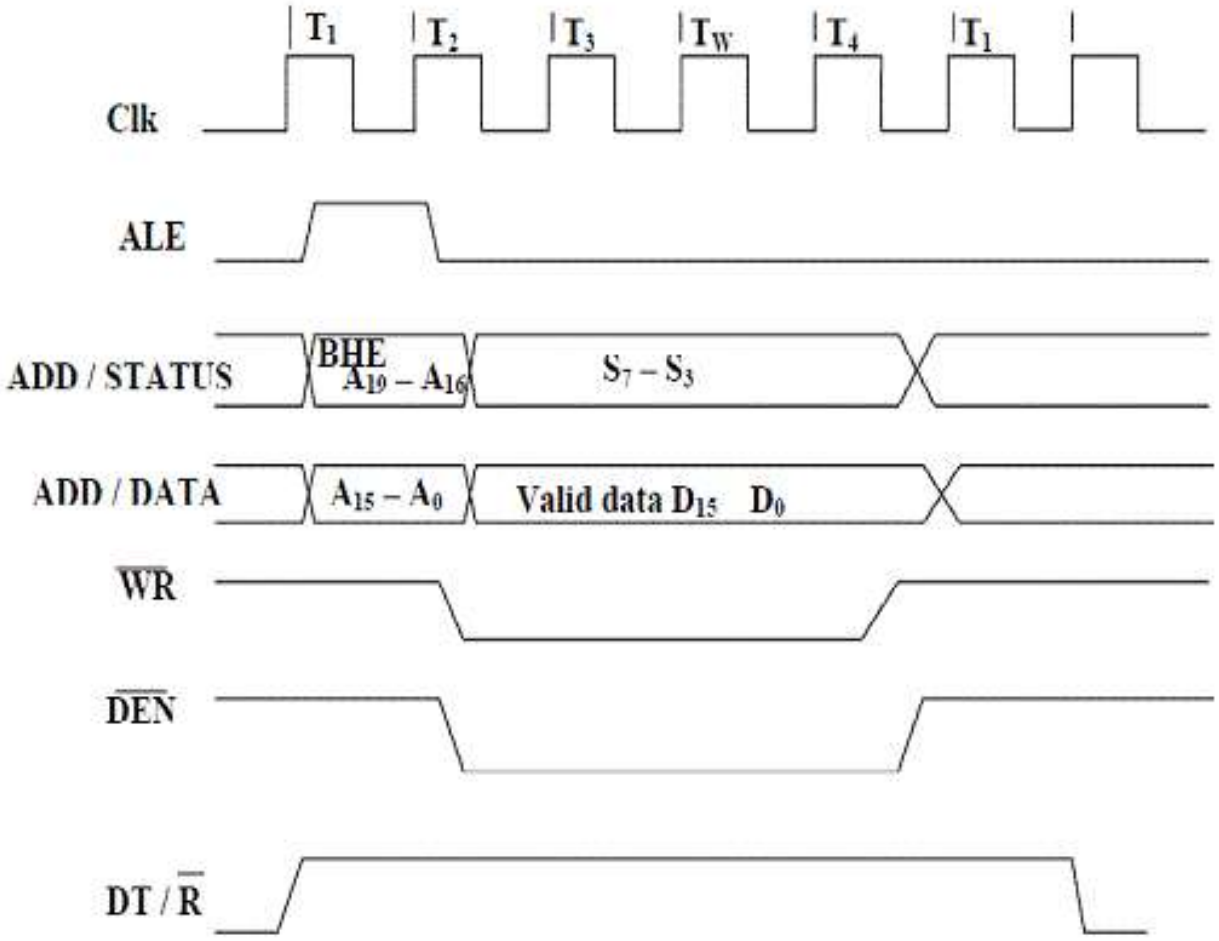
WRITE CYCLE TIMING DIAGRAM FOR MINIMUM MODE

- The read (RD) signal causes the address device to enable its data bus drivers. After RD goes low, the valid data is available on the data bus
- The addressed device will drive the READY line high. When the processor returns the read signal to high level, the addressed device will again tristate its bus drivers.
- A write cycle also begins with the assertion of ALE and the emission of the address. The M/IO signal is again asserted to indicate a memory or I/O operation. In T2, after sending the address in T1, the processor sends the data to be written to the addressed location.

WRITE CYCLE TIMING DIAGRAM FOR MINIMUM MODE

- The data remains on the bus until middle of T4 state. The WR becomes active at the beginning of T2 (unlike RD is somewhat delayed in T2 to provide time for floating).
- The BHE and A0 signals are used to select the proper byte or bytes of memory or I/O word to be read or write.
- The M/IO, RD and signals indicate the type of data transfer as specified in table below.
- Tw – **wait state** time between T3, T4

Write Cycle Timing Diagram for Minimum Mode



Write Cycle Timing Diagram for Minimum Mode

DMA INTERFACE SIGNALS:

- DMA interface of minimum mode consist of the HOLD and HLDA signals.
- When an external device wants to take control of the system bus, switching HOLD to the logic 1 level.
- At the completion of the current bus cycle, the 8086 enters the hold state.
- In the hold state, signal lines AD0 through AD15, A16/S3 through A19/S6, BHE, M/IO, DT/R, RD, WR, DEN and INTR are all in the high Z state.

INTERRUPT SIGNALS:

- interrupt request (INTR) and interrupt acknowledge (INTA).
- INTR is an input to the 8086 that can be used by an external device to signal that it need to be serviced.
  - Logic 1 at INTR represents an active interrupt request.
  - When an interrupt request has been recognized by the 8086, it indicates this fact to external circuit with pulse to logic 0 at the INTA output.
  - the TEST input is also related to the external interrupt interface. Execution of a WAIT instruction causes the 8086 to check the logic level at the TEST input.

INTERRUPT SIGNALS CONT..

- If the logic 1 is found, the MPU suspend operation and goes into the idle state.
- The 8086 no longer executes instructions; instead it repeatedly checks the logic level of the TEST input waiting for its transition back to logic 0.
- As TEST switches to 0, execution resume with the next instruction in the program. This feature can be used to synchronize the operation of the 8086 to an event in external hardware.

INTERRUPT SIGNALS CONT..

- There are two more inputs in the interrupt interface:
  - the non-maskable interrupt NMI and the reset interrupt RESET.
- On the 0-to-1 transition of NMI control is passed to a non-maskable interrupt service routine.
- The RESET input is used to provide a hardware reset for the 8086.
  - Switching RESET to logic 0 initializes the internal register of the 8086 and initiates a reset service routine.

INTERRUPTS

- **Hardware Interrupts** (External Interrupts)
  - Two pins that allow interrupt requests,
    - INTR and NMI
  - One pin that acknowledges, INTA, the interrupt requested on INTR and NMI
  - INTR
    - is a maskable hardware interrupt.
    - The interrupt can be enabled/disabled using STI/CLI instructions or
    - using more complicated method of updating the FLAGS register with the help of the POPF instruction.

INTERRUPTS

- **Hardware Interrupts** (External Interrupts).
  - support hardware interrupts through:
    - NMI is a non-maskable interrupt.
    - Interrupt is processed in the same way as the INTR interrupt.
    - address of the NMI processing routine is stored in location 0008h.
    - This interrupt has higher priority than the maskable interrupt.
    - Ex: NMI, INTR.

INTERRUPTS

- **Software Interrupts**
  - Internal Interrupts and Instructions
  - Software interrupts can be caused by:
    - **INT instruction**
    - INT <interrupt number> instruction - any one interrupt from available 256 interrupts.
      - **INTO instruction** - interrupt on overflow
      - **Single-step interrupt** - generated if the TF flag is set. This is a type 1 interrupt.
        - When the CPU processes this interrupt it clears TF flag before calling the interrupt processing routine.
      - **Processor exceptions**: Divide Error (Type 0), Unused Opcode (type 6) and Escape opcode (type 7).
    - Software interrupt processing is the same as for the hardware interrupts.

INTERRUPTS

- **Functions associated with INT00 to INT04**
  - **INT 00 (divide error)**
    - INT00 is invoked by the microprocessor whenever there is an attempt to **divide a number by zero**.
    - ISR is responsible for displaying the message "Divide Error" on the screen
  - **INT 01**
    - For **single stepping** the trap flag must be 1
    - After execution of each instruction, 8086 automatically jumps to 00004H to fetch 4 bytes for CS: IP of the ISR.
    - The job of ISR is to dump the registers on to the screen
  - **INT 02 (Non maskable Interrupt)**
    - When ever NMI pin of the 8086 is activated by a high signal (5v), the CPU Jumps to physical memory location 00008 to fetch CS:IP of the ISR associated with NMI

INTERRUPTS

- **INT 03 (break point)**
  - A break point is used **to examine the CPU and memory** after the execution of a group of Instructions.
  - It is **one byte instruction** whereas other instructions of the form "INT nn" are 2 byte instructions
- **INT 04 (Signed number overflow)**
  - There is an instruction associated with this INT 0 (interrupt on overflow).
  - If INT 0 is placed after a signed number arithmetic as IMUL or ADD the CPU will activate INT 04 if 0F = 1.
  - In case where 0F = 0, the INT 0 is not executed but is bypassed and acts as a NOP