Q1. How does the 8086 differ from the 8085 microprocessor?
- 8086: 16-bit microprocessor, supports multitasking, segmented memory (1MB addressable).
- 8085: 8-bit microprocessor, simpler with 64KB memory limit.

Q2. Point out the differences between the 8086 and 8088 microprocessor?
- 8086: 16-bit data bus, faster for high-speed data transfer.
- 8088: 8-bit data bus, compatible with 8-bit peripherals.

Q3. How many bit microprocessor is the 8086 microprocessor?
- The 8086 is a 16-bit microprocessor.

Q4. Explain the internal architecture of the 8086 microprocessor?
- Two main units:
  - BIU: Handles memory, I/O, instruction prefetching, address calculations.
  - EU: Executes instructions, processes data.

Q5. What is the purpose of the BIU in the 8086?
- BIU fetches instructions, computes physical addresses, manages memory and I/O.

Q6. What does the EU do in the 8086?
- EU executes instructions, handles arithmetic, logic, and control operations.

Q7. State the number and type of registers in the 8086?
- General: AX, BX, CX, DX (16-bit, divisible).
- Segment: CS, DS, ES, SS.
- Pointer/Index: SP, BP, SI, DI.
- IP and Flags Register.

Q8. What is the purpose of segment registers?
- Segment registers divide memory into 4 segments for efficient 1MB access.

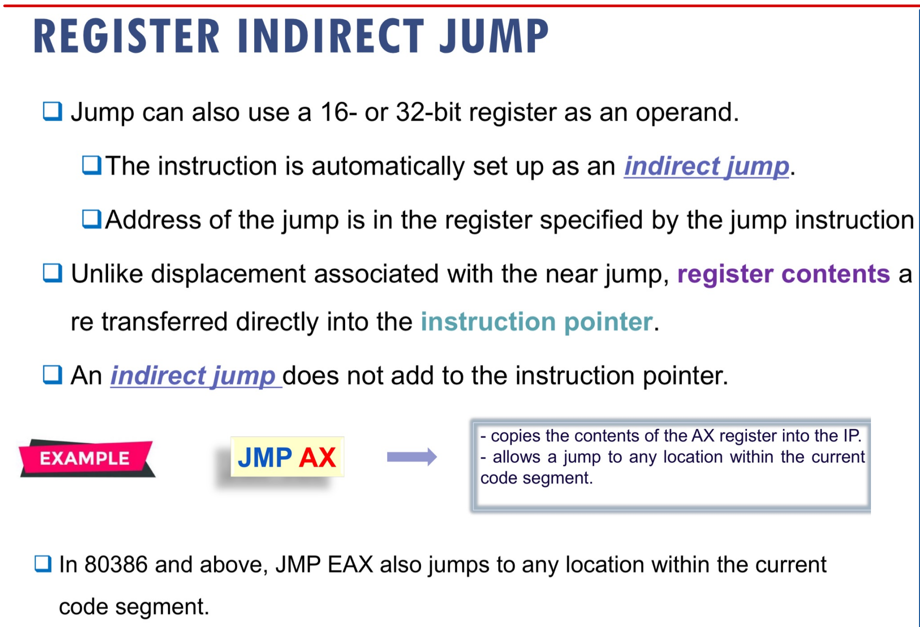
Q9. What are the index registers in 8086?
- SI (Source Index) for source memory.
- DI (Destination Index) for destination memory.

Q10. Define the number of flags present in the 8086 and name them.
- 9 flags: CF, PF, AF, ZF, SF, OF (status), TF, IF, DF (control).

Q11. What does the parity flag do?
- Indicates if the least significant byte of the result has even (PF=1) or odd (PF=0) 1s.

Q12. What is single stepping and how can it be achieved on the 8086?
- Executes one instruction for debugging by setting the Trap Flag (TF).

Q13. With the help of an example, explain how physical address is calculated.
- Physical Address = (Segment Register × 16) + Offset.
 Example: CS = 0x1234, IP = 0x5678.
 Physical Address = (0x1234 × 16) + 0x5678 = 0x179B8.



REGISTER INDIRECT JUMP

❑ Jump can also use a 16- or 32-bit register as an operand.

❑ The instruction is automatically set up as an ***indirect jump***.

❑ Address of the jump is in the register specified by the jump instruction

❑ Unlike displacement associated with the near jump, **register contents** are transferred directly into the **instruction pointer**.

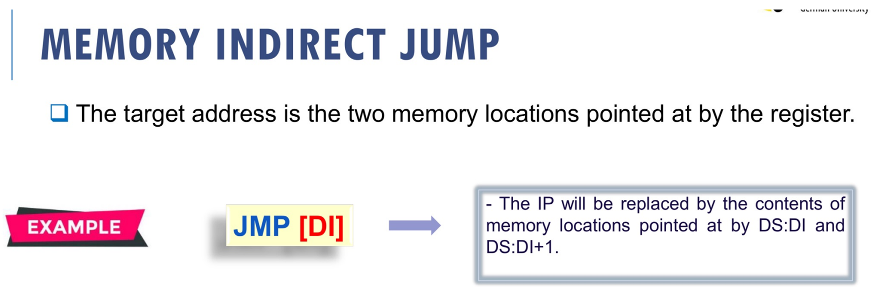
❑ An ***indirect jump*** does not add to the instruction pointer.

EXAMPLE

JMP AX

- copies the contents of the AX register into the IP.
- allows a jump to any location within the current code segment.

❑ In 80386 and above, JMP EAX also jumps to any location within the current code segment.



MEMORY INDIRECT JUMP

❑ The target address is the two memory locations pointed at by the register.

EXAMPLE

JMP [DI]

- The IP will be replaced by the contents of memory locations pointed at by DS:DI and DS:DI+1.

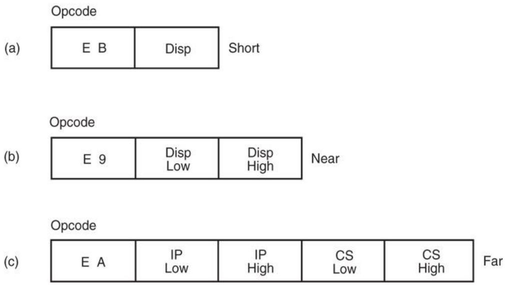
Jump

unConditional
- Short (2 Byte jump) within -128 –127 bytes from following Address
- Near (3 Byte Jump) within ± 32K bytes from Following Address
- Far (5 Byte jump) Can go between Segments (As segment is 64K)

* Short / Near intrasegment jump

* Far intersegment jump

Conditional [using flag bits



Opcode
(a)
E B
Disp
Short
Opcode
(b)
E 9
Disp Low
Disp High
Near
Opcode
(c)
E A
IP Low
IP High
CS Low
CS High
Far