INTRODUCTION TO 8086 microprocessor ARCHITECTURE OF 8086 (PIN DIAGRAM, FUNCTIONAL BLOCK DIAGRAM) LECTURE 6
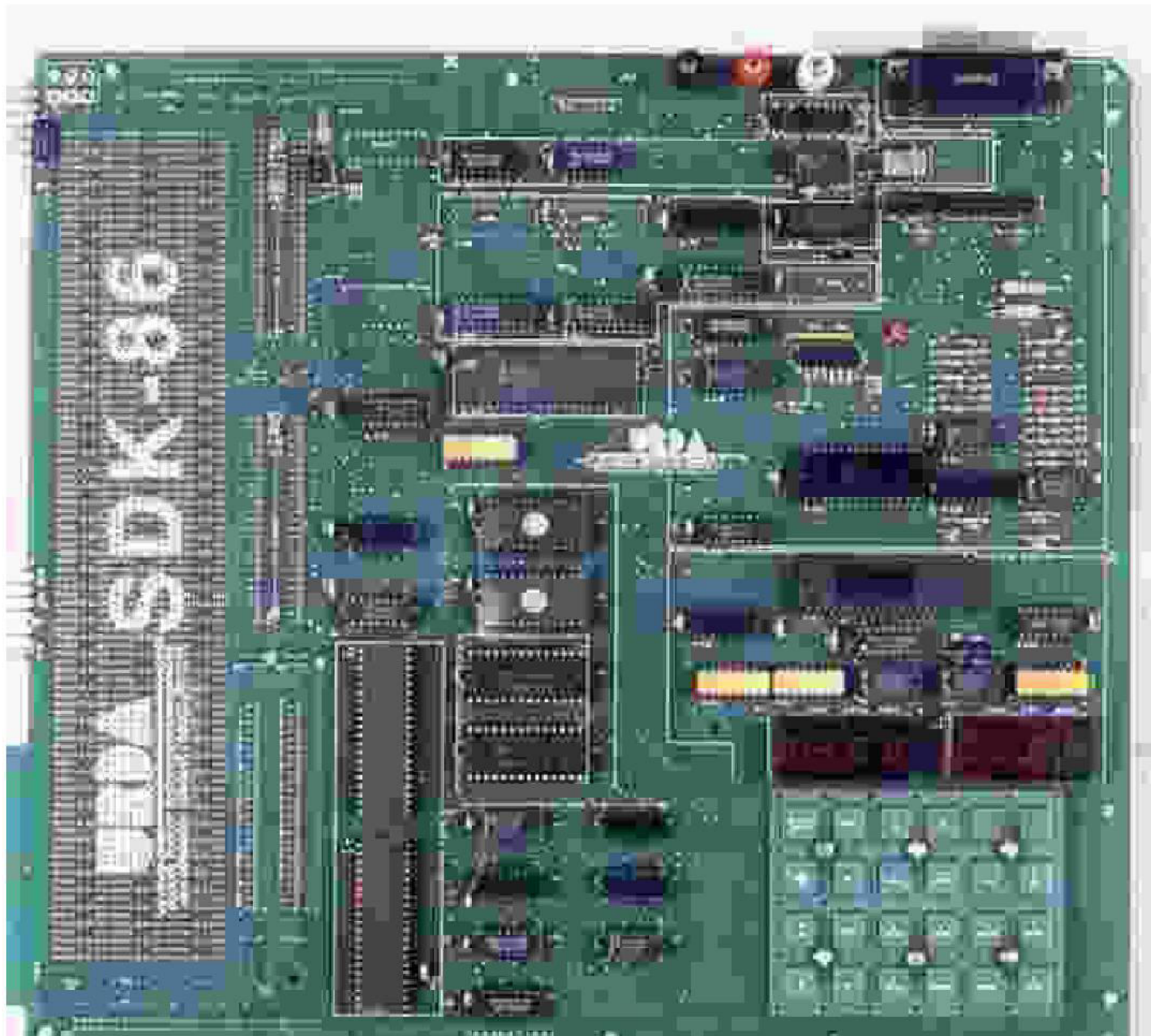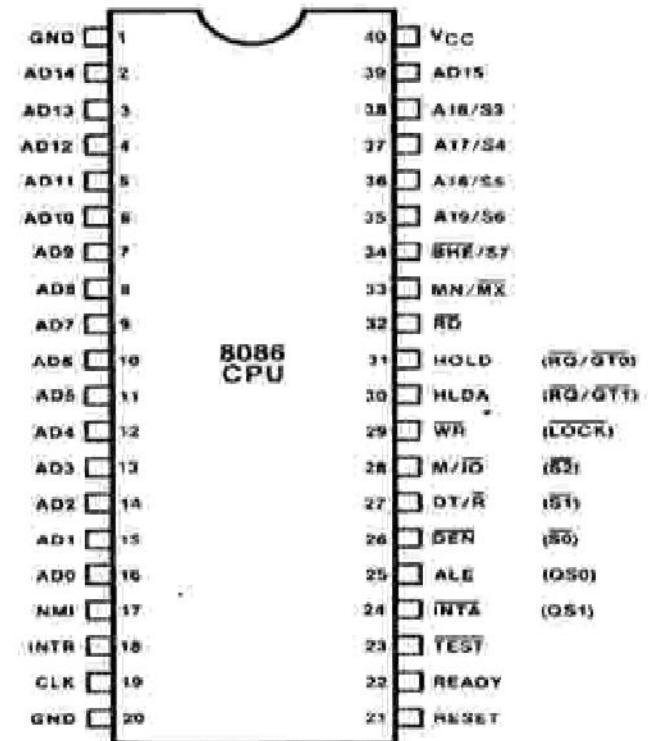
Dronacharya Group of Institutions

8086 MICROPROCESSOR

Pinouts

Common Signals

Name	Function	Type
AD15–AD0	Address/Data Bus	Bidirectional, 3-State
A19/S6–A16/S3	Address/Status	Output, 3-State
$\overline{BHE}$/S7	Bus High Enable/Status	Output, 3-State
MN/$\overline{MX}$	Minimum/Maximum Mode Control	Input
$\overline{RD}$	Read Control	Output, 3-State
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
VCC	+5V	Input
GND	Ground	

Minimum Mode Signals (MN/MX = V_{CC})

Name	Function	Type
HOLD	Hold Request	Input
HLDA	Hold Acknowledge	Output
$\overline{WR}$	Write Control	Output, 3-State
M/$\overline{IO}$	Memory/IO Control	Output, 3-State
DT/$\overline{R}$	Data Transmit/Receive	Output, 3-State
$\overline{DEN}$	Data Enable	Output, 3-State
ALE	Address Latch Enable	Output
$\overline{INTA}$	Interrupt Acknowledge	Output

Maximum Mode Signals (MN/MX = GND)

Name	Function	Type
$\overline{RQ}$/$\overline{GT}$1, 0	Request/Grant Bus Access Control	Bidirectional
$\overline{LOCK}$	Bus Priority Lock Control	Output, 3-State
$\overline{S2}$–$\overline{S0}$	Bus Cycle Status	Output, 3-State
QS1, QS0	Instruction Queue Status	Output

```
GND   [ 1        40 ]  VCC
AD14  [ 2        39 ]  AD15
AD13  [ 3        38 ]  A16/S3
AD12  [ 4        37 ]  A17/S4
AD11  [ 5        36 ]  A18/S5
AD10  [ 6        35 ]  A19/S6
AD9   [ 7        34 ]  BHE/S7
AD8   [ 8        33 ]  MN/MX
AD7   [ 9        32 ]  RD
AD6   [ 10   8086   31 ]  HOLD   (RQ/GT0)
AD5   [ 11   CPU    30 ]  HLDA   (RQ/GT1)
AD4   [ 12       29 ]  WR     (LOCK)
AD3   [ 13       28 ]  M/IO   (S2)
AD2   [ 14       27 ]  DT/R   (S1)
AD1   [ 15       26 ]  DEN    (S0)
AD0   [ 16       25 ]  ALE    (QS0)
NMI   [ 17       24 ]  INTA   (QS1)
INTR  [ 18       23 ]  TEST
CLK   [ 19       22 ]  READY
GND   [ 20       21 ]  RESET
```

MAXIMUM MODE PIN FUNCTIONS (e.g. $\overline{LOCK}$) ARE SHOWN IN PARENTHESES

I-46

8086 Pins

The 8086 comes in a 40 pin package which means that some pins have more than one use or are multiplexed. The packaging technology of time limited the number of pin that could be used.

In particular, the address lines 0 - 15 are multiplexed with data lines 0-15, address lines 16-19 are multiplexed with status lines. These pins are

AD0 - AD15, A16/S3 - A19/S6

The 8086 has one other pin that is multiplexed and this is BHE'/S7. BHE stands for Byte High Enable. This is an active low signal that is asserted when there is data on the upper half of the data bus.

The 8086 has two modes of operation that changes the function of some pins. The SDK-86 uses the 8086 in the minimum mode with the MN/MX' pin tied to 5 volts. This is a simple single processor mode. The IBM PC uses an 8088 in the maximum mode with the MN/MX" pin tied to ground. This is the mode required for a coprocessor like the 8087.

8086 Pins

In the <u>minimum mode</u> the following pins are available.

HOLD   When this pin is high, another master is requesting control of the local bus, e.g., a DMA controller.

HLDA   HOLD Acknowledge: the 8086 signals that it is going to float the local bus.

WR'   Write: the processor is performing a write memory or I/O operation.

M/IO'   Memory or I/O operation.

DT/R'   Data Transmit or Receive.

DEN'   Data Enable: data is on the multiplexed address/data pins.

ALE   Address Latch Enable: the address is on the address/data pins. This signal is used to capture the address in latches to establish the address bus.

INTA'   Interrupt acknowledge: acknowledges external interrupt requests.

8086 Pins

The following are pins are available in both minimum and maximum modes.

VCC + 5 volt power supply pin.

GND Ground

RD' READ: the processor is performing a read memory or I/O operation.

READY Acknowledgement from wait-state logic that the data transfer will be completed.

RESET Stops processor and restarts execution from FFFF:0. Must be high for 4 clocks. CS = 0FFFFH, IP = DS = SS = ES = Flags = 0000H, no other registers are affected.

TEST' The WAIT instruction waits for this pin to go low. Used with 8087.

NMI Non Maskable Interrupt: transition from low to high causes an interrupt. Used for emergencies such as power failure.

INTR Interrupt request: masked by the IF bit in FLAG register.

CLK Clock: 33% duty cycle, i.e., high 1/3 the time.

8086 Features

• **16-bit Arithmetic Logic Unit**

• **16-bit data bus (8088 has 8-bit data bus)**

• **20-bit address bus - 2^{20} = 1,048,576 = 1 meg**

The address refers to a byte in memory. In the 8088, these bytes come in on the 8-bit data bus. In the 8086, bytes at even addresses come in on the low half of the data bus (bits 0-7) and bytes at odd addresses come in on the upper half of the data bus (bits 8-15).

The 8086 can read a 16-bit word at an even address in one operation and at an odd address in two operations. The 8088 needs two operations in either case.
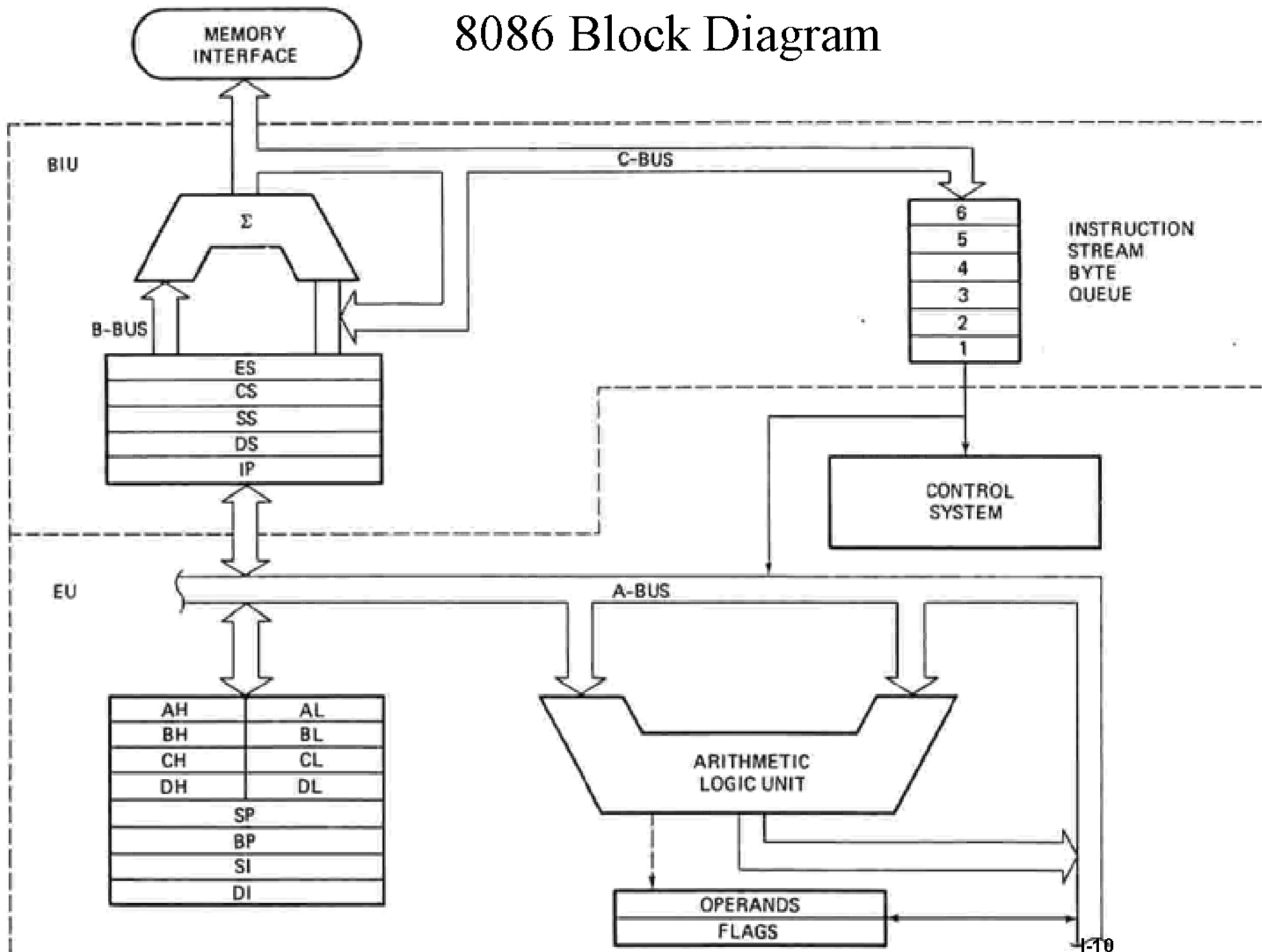
The least significant byte of a word on an 8086 family microprocessor is at the lower address.

8086 Architecture

• The 8086 has two parts, the Bus Interface Unit (BIU) and the Execution Unit (EU).

• The BIU fetches instructions, reads and writes data, and computes the 20-bit address.

• The EU decodes and executes the instructions using the 16-bit ALU.

• The BIU contains the following registers:

    IP - the Instruction Pointer
    CS - the Code Segment Register
    DS - the Data Segment Register
    SS - the Stack Segment Register
    ES - the Extra Segment Register

The BIU fetches instructions using the CS and IP, written CS:IP, to contruct the 20-bit address. Data is fetched using a segment register (usually the DS) and an effective address (EA) computed by the EU depending on the addressing mode.

8086 Block Diagram

MEMORY INTERFACE

BIU

C-BUS

Σ

B-BUS

ES
CS
SS
DS
IP

INSTRUCTION STREAM BYTE QUEUE

6
5
4
3
2
1

CONTROL SYSTEM

EU

A-BUS

AH	AL
BH	BL
CH	CL
DH	DL
SP	
BP	
SI	
DI	

ARITHMETIC LOGIC UNIT

OPERANDS

FLAGS

I-10

8086 Architecture

The EU contains the following 16-bit registers:

AX - the Accumulator
BX - the Base Register
CX - the Count Register
DX - the Data Register

SP - the Stack Pointer   \ defaults to stack segment
BP - the Base Pointer   /
SI - the Source Index Register
DI - the Destination Register

These are referred to as general-purpose registers, although, as seen by their names, they often have a special-purpose use for some instructions.

The AX, BX, CX, and DX registers can be considers as two 8-bit registers, a High byte and a Low byte. This allows byte operations and compatibility with the previous generation of 8-bit processors, the 8080 and 8085. 8085 source code could be translated in 8086 code and assembled. The 8-bit registers are:

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

Flag Register

❑ Flag register contains information reflecting the current status of a microprocessor. It also contains information which controls the operation of the microprocessor.

15													0
—	NT	IOPL	OF	DF	IF	TF	SF	ZF	—	AF	—	PF	—	CF

➢ Control Flags

IF: Interrupt enable flag
DF: Direction flag
TF: Trap flag

➢ Status Flags

CF: Carry flag
PF: Parity flag
AF: Auxiliary carry flag
ZF: Zero flag
SF: Sign flag
OF: Overflow flag
NT: Nested task flag
IOPL: Input/output privilege level

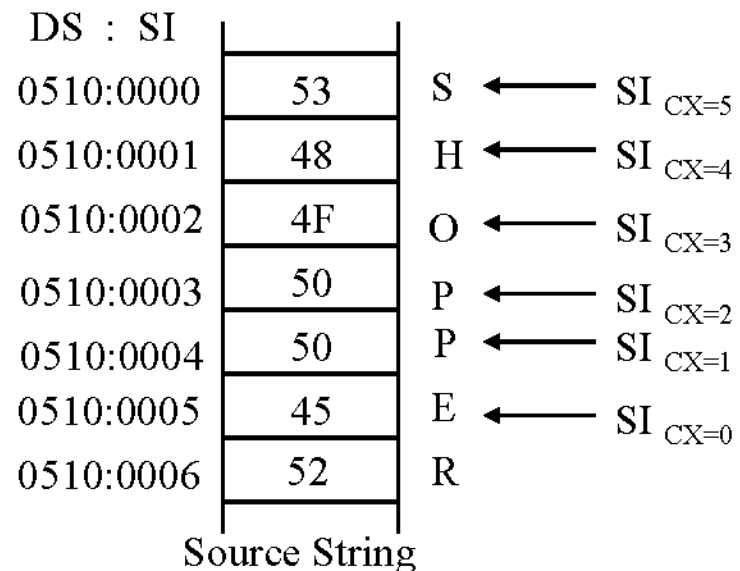# Flags Commonly Tested During the Execution of Instructions

❑ There are five flag bits that are commonly tested during the execution of instructions

     — Sign Flag (Bit 7), SF: 0 for positive number and 1 for negative number

     — Zero Flag (Bit 6), ZF: If the ALU output is 0, this bit is set (1); otherwise, it is 0

     — Carry Flag (Bit 0), CF: It contains the carry generated during the execution

     — Auxiliary Carry, AF: Depending on the width of ALU inputs, this flag
           (Bit 4)   bit contains the carry generated at bit 3 (or, 7, 15) of the 8088 ALU

     — Parity Flag (bit2), PF: It is set (1) if the output of the ALU has even number of ones; otherwise it is zero

Direction Flag

❑ Direction Flag (DF) is used to control the way SI and DI are adjusted during the execution of a string instruction

— DF=0, SI and DI will auto-increment during the execution; otherwise, SI and DI auto-decrement

— Instruction to set DF: **STD**; Instruction to clear DF: **CLD**

— Example:

CLD
MOV CX, 5
REP MOVSB

At the beginning of execution,
DS=0510H and SI=0000H

DS : SI		
0510:0000	53	S ← SI $_{CX=5}$
0510:0001	48	H ← SI $_{CX=4}$
0510:0002	4F	O ← SI $_{CX=3}$
0510:0003	50	P ← SI $_{CX=2}$
0510:0004	50	P ← SI $_{CX=1}$
0510:0005	45	E ← SI $_{CX=0}$
0510:0006	52	R

Source String

MAX and MIN Modes

- In minmode, the 9 signals correspond to **control signals** that are needed to operate memory and I/O devices connected to the 8088.

- In maxmode, the 9 signals change their functions; the 8088 now requires the use of the **8288 bus controller** to generate memory and I/O read/write signals.

Why MIN and MAX modes?

- Minmode signals can be directly decoded by memory and I/O circuits, resulting in a system with minimal hardware requirements.

- Maxmode systems are more complicated, but obtain the new signals that allow for bus grants (e.g. DMA), and the use of an 8087 coprocessor.

The 9 pins (min)

- **ALE: address latch enable (AD0 – AD7)
- **DEN: data enable (connect/disc. buffer)
- **WR: write (writing indication)
- *HOLD
- *HDLA: hold acknowledge
- *INTA: interrupt acknowledge
- IO/M: memory access or I/O access
- DT/R: data transmit / receive (direction)
- SSO: status

The 9 pins (max)

- S0, S1, S2: status
- *RQ/GT0, RQ/GT1: request/grant
- *LOCK: locking the control of the sys. bus
- *QS1, QS0: queue status (tracking of internal instruction queue).
- HIGH

Instruction Types

❑ **Data transfer instructions**

❑ **String instructions**

❑ **Arithmetic instructions**

❑ **Bit manipulation instructions**

❑ **Loop and jump instructions**

❑ **Subroutine and interrupt instructions**

❑ **Processor control instructions**

Addressing Modes

Addressing Modes	*Examples*
❑ Immediate addressing	MOV AL, 12H
❑ Register addressing	MOV AL, BL
❑ Direct addressing	MOV [500H], AL
❑ Register Indirect addressing	MOV DL, [SI]
❑ Based addressing	MOV AX, [BX+4]
❑ Indexed addressing	MOV [DI-8], BL
❑ Based indexed addressing	MOV [BP+SI], AH
❑ Based indexed with displacement addressing	MOV CL, [BX+DI+2]

Exceptions

❑ String addressing

❑ Port addressing   (*e.g.* IN AL, 79H)

Data Transfer Instructions

❑ **MOV** *Destination, Source*

— Move data from source to destination; *e.g.* **MOV [DI+100H], AH**

— It does not modify flags

➢ For 80x86 family, directly moving data from one memory location to another memory location is not allowed

   MOV [SI], [5000H]   ✖

➢ When the size of data is not clear, assembler directives are used

   MOV [SI], 0   ❓

   ▪ **BYTE PTR**   MOV BYTE PTR [SI], 12H
   ▪ **WORD PTR**   MOV WORD PTR [SI], 12H
   ▪ **DWORD PTR**   MOV DWORD PTR [SI], 12H

➢ You can not move an immediate data to segment register by MOV
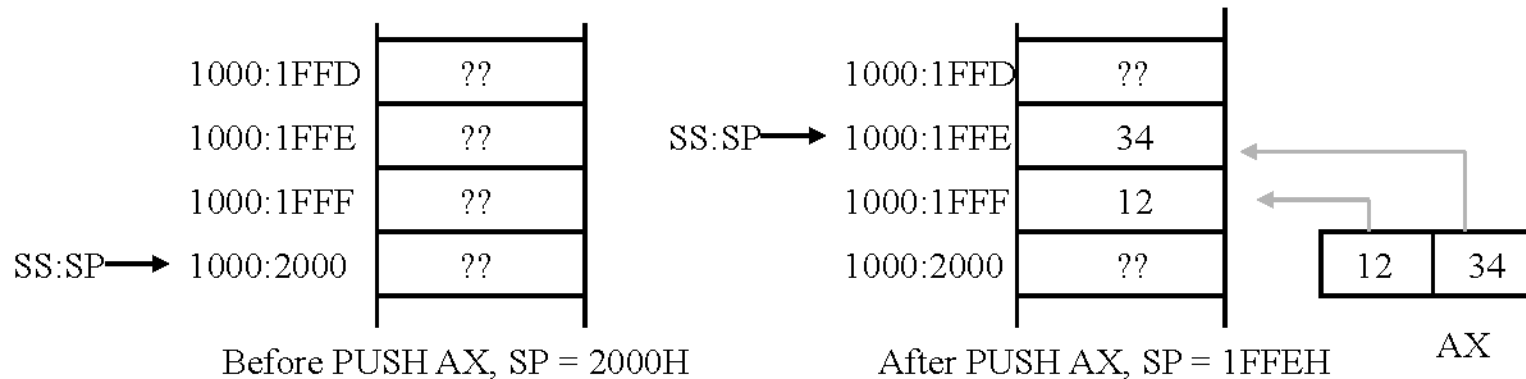
   MOV DS, 1234H   ✖

Instructions for Stack Operations

❑ **What is a Stack ?**

  — A stack is a collection of memory locations. It always follows the rule of
    last-in-firs-out
  — Generally, SS and SP are used to trace where is the latest date written into stack

❑ **PUSH** *Source*

  — Push data (**word**) onto stack
  — It does not modify flags
  — For Example: PUSH AX   (assume ax=1234H, SS=1000H, SP=2000H
                                        before PUSH AX)

1000:1FFD	??		1000:1FFD	??	
1000:1FFE	??	SS:SP→ 1000:1FFE	34	
1000:1FFF	??	1000:1FFF	12	
SS:SP→ 1000:2000	??	1000:2000	??	12	34

Before PUSH AX, SP = 2000H   After PUSH AX, SP = 1FFEH   AX

➤ Decrementing the stack pointer during a push is a standard way of implementing stacks in hardware
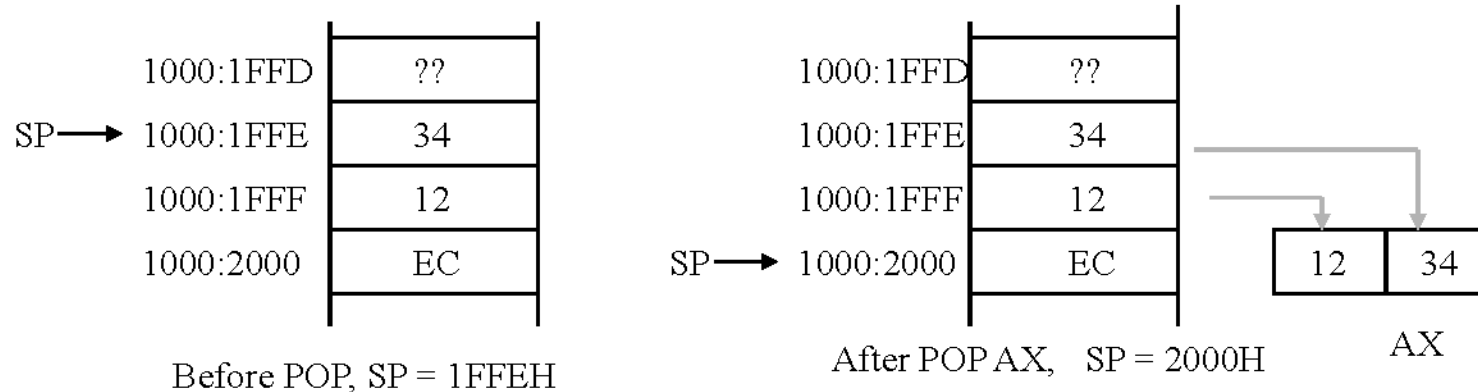
Instructions for Stack Operations

❑ **PUSHF**

— Push the values of the flag register onto stack
— It does not modify flags

❑ **POP** *Destination*

— Pop word off stack
— It does not modify flags
— For example: **POP   AX**

1000:1FFD	??
SP→ 1000:1FFE	34
1000:1FFF	12
1000:2000	EC

Before POP, SP = 1FFEH

1000:1FFD	??
1000:1FFE	34
1000:1FFF	12
SP→ 1000:2000	EC

12	34

AX

After POP AX,   SP = 2000H

❑ **POPF**

— Pop word from the stack to the flag register
— It modifies all flags

Data Transfer Instructions

❑ SAHF

— Store data in AH to the low 8 bits of the flag register
— It modifies flags: AF, CF, PF, SF, ZF

❑ LAHF

— Copies bits 0-7 of the flags register into AH
— It does not modify flags

❑ LDS *Destination Source*

— Load 4-byte data (pointer) in memory to two 16-bit registers
— Source operand gives the memory location
— The first two bytes are copied to the register specified in the destination operand; the second two bytes are copied to register DS
— It does not modify flags

❑ LES *Destination Source*

— It is identical to LDS except that the second two bytes are copied to ES
— It does not modify flags

Data Transfer Instructions

❑ LEA *Destination Source*

— Transfers the offset address of source (must be a memory location) to the destination register
— It does not modify flags

❑ XCHG *Destination Source*

— It exchanges the content of destination and source
— One operand must be a microprocessor register, the other one can be a register or a memory location
— It does not modify flags

❑ XLAT

— Replace the data in AL with a data in a user defined look-up table
— BX stores the beginning address of the table
— At the beginning of the execution, the number in AL is used as the index of the look-up table
— It does not modify flags