

<b>DS:</b>	<b>05D10H</b>	
<b>Offset:</b>	<b>0016H</b>	<b>+</b>
<b>Address of data item:</b>	<b>05D26H</b>	

Assume the address **05D26H** contain **4AH**, the processor now extract the **4AH** at address **05D26H** and copy it into **AL** register.

An instruction may also access more than one byte at a time

**EX:** Suppose an instruction is to store the content of the **AX** register (**0248H**) in two adjacent byte in the **DS** beginning at offset **0016H**.

The symbolic code **MOV [0016], AX**

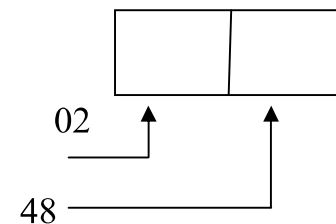
The processor stores the two byte in memory in reversed byte sequence as

Content of AX:        02        48

Offset in DS:        0017        0016

Another instruction, **MOV AX, [0016]**, subsequently could retrieve these byte by copy them from memory back into **AX**.

The operation reverses (and corrects) the byte in **AX** as:



## Number of Operands

Operands specify the value an instruction is to operate on, and where the result is to be stored. Instruction sets are classified by the number of operands used.

An instruction may have no, one, two, or three operands.

1. **Three-Operand Instruction:** In instruction that have three operands, one of the operand specifies the destination as an address where the result is to be saved. The other two operands specify the source either as addresses of memory location or constants.

**EX:**

**A=B+C**

**ADD destination, source1, source2**

**ADD A,B,C**

**EX:**

**Y=(X+D)\* (N+1)**

**ADD T1, X,**

**D ADD T2, N, 1**

**Mul Y, T1, T2**

2. **Two operand instruction :**In this type both operands specify sources. The first operand also specifies the destination address after the result is to be saved. The first operand must be an address in memory, but the second may be an address or a constant.

**ADD destination, source**

**EX: A=B+C**

**MOV A, B**

**ADD A, C**

**EX:**  $Y=(X+D)*(N+1)$

MOV T1, X

ADD T1, D

MOV Y, N

ADD Y, 1

MUL Y, T1

**3. One Operand instruction:** Some computer have only one general purpose register, usually called on Acc. It is implied as one of the source operands and the destination operand in memory instruction the other source operand is specified in the instruction as location in memory.

#### **ADD source**

**LDA** source; copy value from memory to ACC.

**STA** destination; copy value from Acc into memory.

**EX:**  $A=B+C$

LDA B

ADD C

STA A

**EX:**  $Y=(X+D)*(N+1)$

LDA X

ADD D

STA T1

LDA N

ADD 1

MUL T1

STA Y

4. **Zero Operand instruction:** Some computers have arithmetic instruction in which all operands are implied, these zero operand instruction use a stack, a stack is a list structure in which all insertion and deletion occur at one end, the element on a stack may be removed only in the reverse of the order in which they were entered. The process of inserting an item is called **Pushing**, removing an item is called **Popping**.

Computers that use Zero operand instruction for arithmetic operations also use one operand **PUSH** and **POP** instruction to copy value between memory and the stack.

**PUSH source;** Push the value of the memory operand onto the Top of the stack.

**POP destination;** POP value from the Top of the stack and copy it into the memory operand.

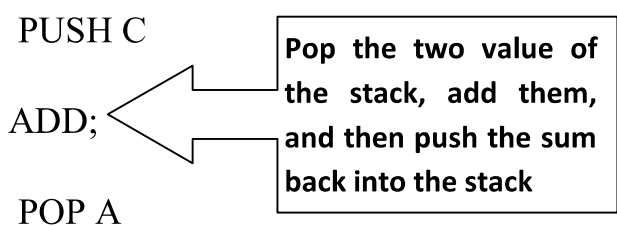
**EX:**  $A=B+C$

PUSH B

PUSH C

ADD;

POP A



Pop the two value of the stack, add them, and then push the sum back into the stack

**EX:**  $Y=(X+D)* (N+1)$

PUSH X

PUSH D

ADD

PUSH N

PUSH 1

ADD

MUL

POP Y

### **Assembly Language Instruction**

Assembly Language Instruction Assembly language instructions are provided to describe each of the basic operations that can be performed by a microprocessor. They are written using **alphanumeric symbols** instead of the 0s and 1s of the microprocessor's machine code. Program written in assembly language are called **source code**. An assembly language description of this instruction is

ADD AX, BX

In this example, the contents of BX and AX are added together and their sum is put in AX. Therefore, BX is considered to be the **source operand** and AX the **destination operand**.

Here is another example of an assembly language statement:

LOOP: MOV AX, BX ; COPY BX INTO AX

This instruction statement starts with the word LOOP. It is an address identifier for the instruction MOV AX, BX. This type of identifier is called a **label** or **tag**. The instruction is followed by "COPY BX INTO AX." This part of the statement is called **a comment**. Thus a general format for writing an assembly language statement is:

LABEL:      INSTRUCTION ;      COMMENT