Simplified lectures in 8088 Microprocessor (4)

Presentation · November 2013

CITATIONS
0

READS
73

1 author:

Abdullatif Baba
Kuwait College of Science and Technology (Private University)
119 PUBLICATIONS **183** CITATIONS

SEE PROFILE

An Introduction to the 8088 Microprocessor (4)

Dr. Eng. Abdellatif BABA

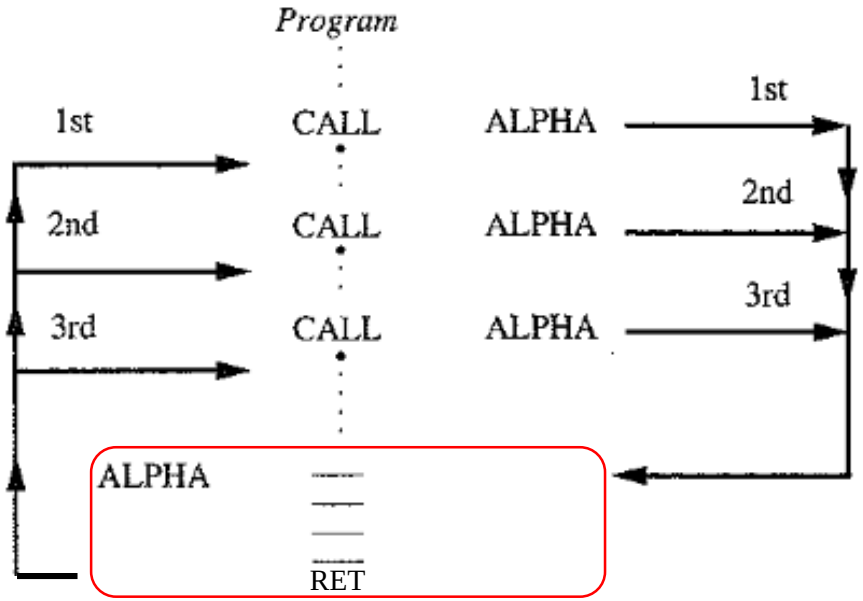
Based on "An Introduction to the Intel Family of Microprocessors" by James L. Antonakos

Flag regisetr Instructions

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-				OF	DF	IF	TF	SF	ZF	-	AF	-	PF	-	CF

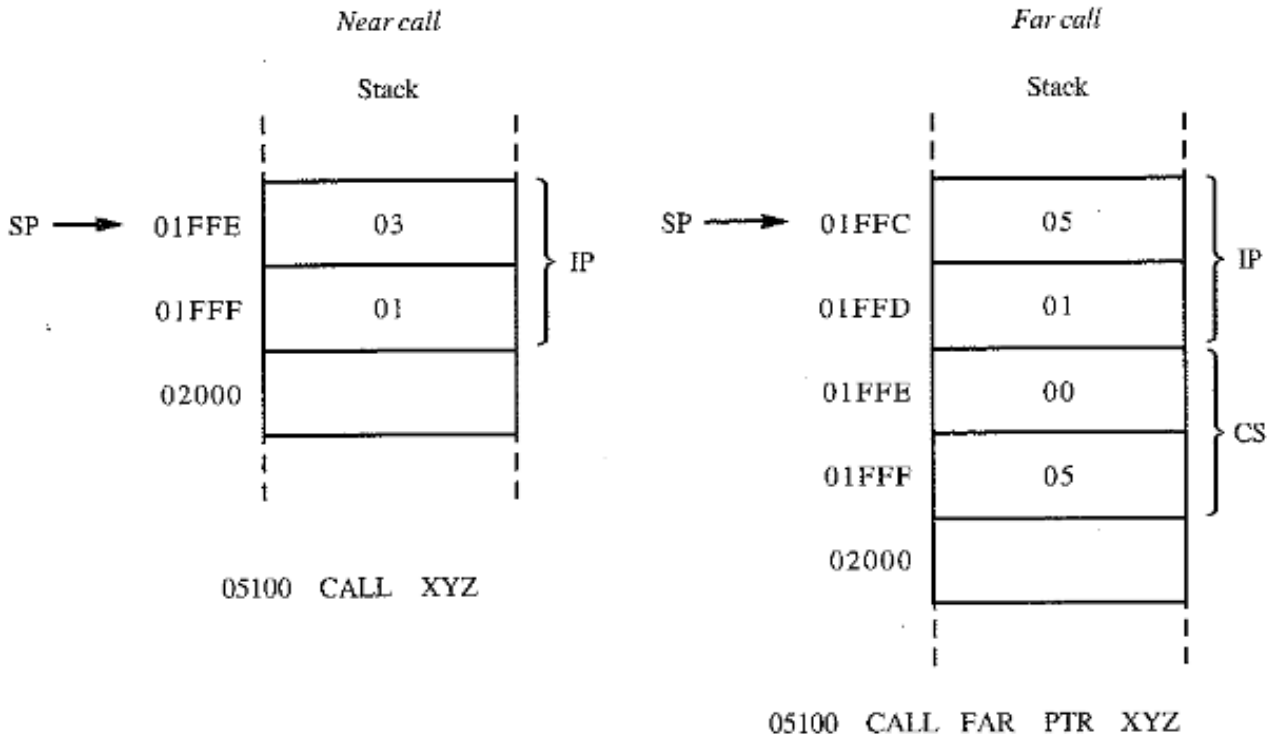
LAHF	Load the lower byte of flag register to AH
SAHF	Store AH in the lower byte of flag register
CLC	0 ⟶ CF reset Carry Flag
STC	1 ⟶ CF preset Carry Flag
CMC	$\overline{CF}$ ⟶ CF First complement Carry Flag
CLI	0 ⟶ IF reset Interrupt En. Flag
STI	1 ⟶ IF preset Interrupt En. Flag
CLD	0 ⟶ DF reset Direction Flag
STD	1 ⟶ DF preset Direction Flag

Subroutine Instructions



- Subroutine is a collection of instructions CALLed from one or many other locations.
- At the end of subroutine RETurn instruction tell the micro. (go back to the next instruction where it was called from).
- Direct CALL the address of called procedure is specified inside the instruction “via label naming it”, Example: CALL XYZ.
- Indirect CALL the procedure address is contained in a register or memory location, Example: CALL BX or CALL [SI]

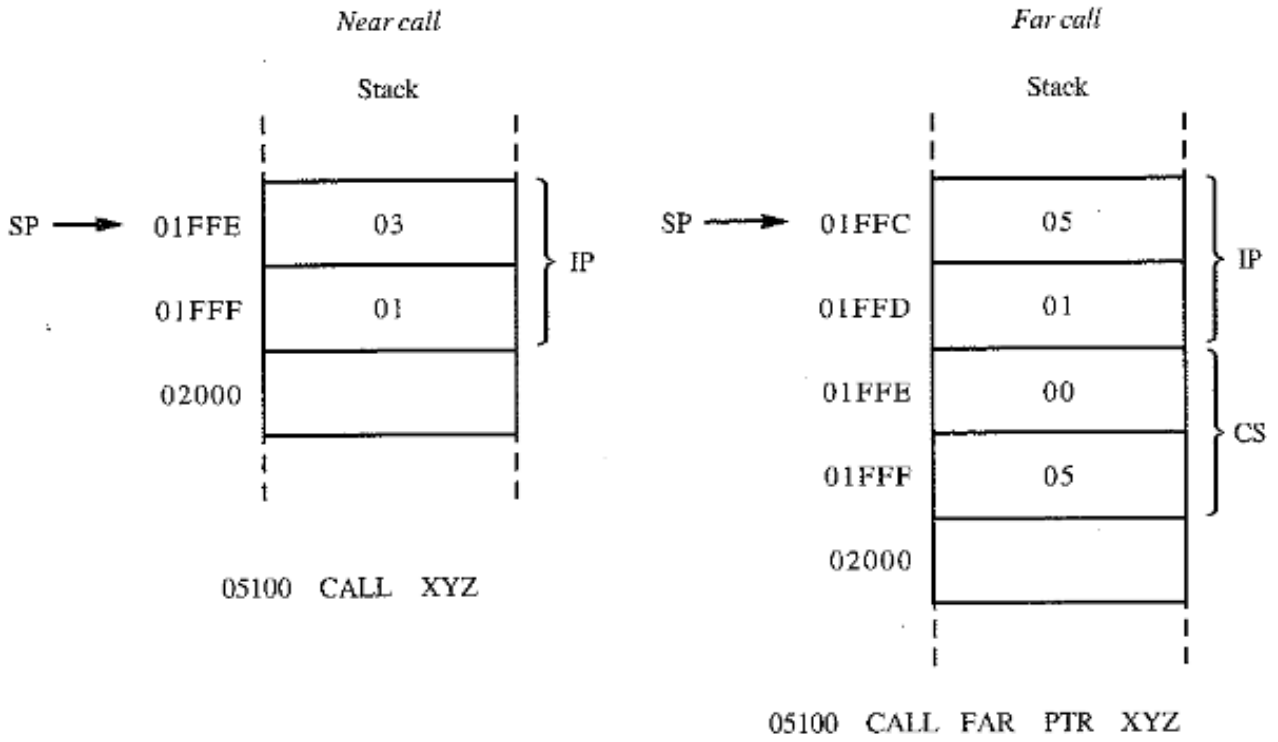
Subroutine Instructions



Subroutine may exists in the same (current) code segment or in a different one

- Near CALL pushs the address of the instruction immediately following itself onto the stack. (One word data of IP)
- Far CALL: two pushes are performed. One for the instruction offset IP and the other for the code segment CS address.

Subroutine Instructions

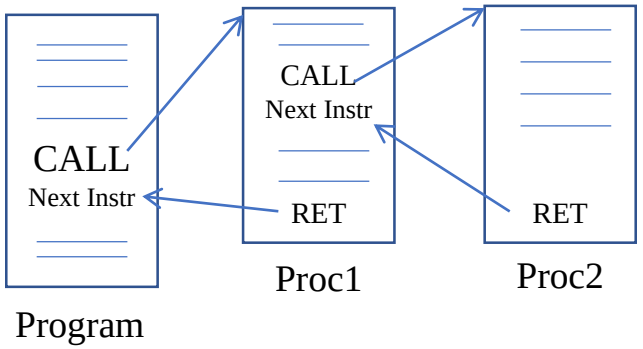


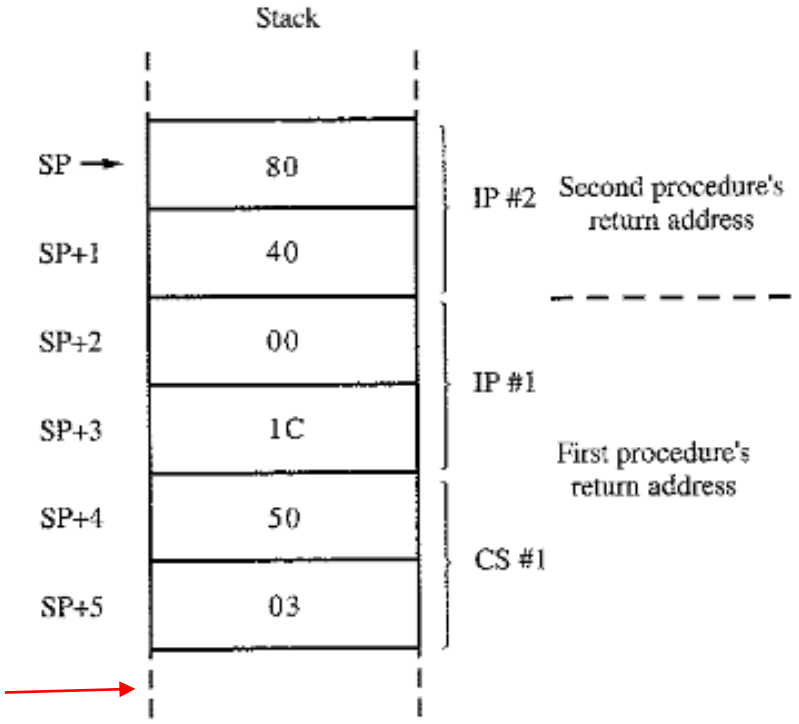
RET this instruction provides an exit from the called procedure.
RET doesn't know anything about the procedure it terminates except for its type being (Far or Near).
For a near procedure, it pops one word off the stack and write it into the IP register
For far procedure, it pops two words the first goes into the IP reg. the second goes into the CS reg.

Subroutine Instructions

In this example:

- The subroutine called first is a far procedure.
- The second subroutine is a near procedure called from inside the first procedure





The instruction that CALLed the first proc. pushed the return address into stack :
(CS = 0350H) into stack; SP = SP -2
(IP = 1C00H) into stack ; SP = SP -2
The second CALL instruction pushed the return address :
(IP = 4080H) into stack; SP = SP -2

The RET instruction in Proc2 pops 4080H off stack then place it into IP :
IP = 4080H; SP = SP +2
The second RET instruction in Proc1 pops 1C00H off the stack and place it into IP
IP = 1C00H ; SP = SP +2
Then it pops 0350H off the stack and place it into CS
CS = 0350H; SP = SP +2

String Instructions

The string: is a sequence of ASCII character codes saved in the memory.

Our tools:

CX : contains the repeat count necessary to scan all a given string.

DF : Direction Flag.

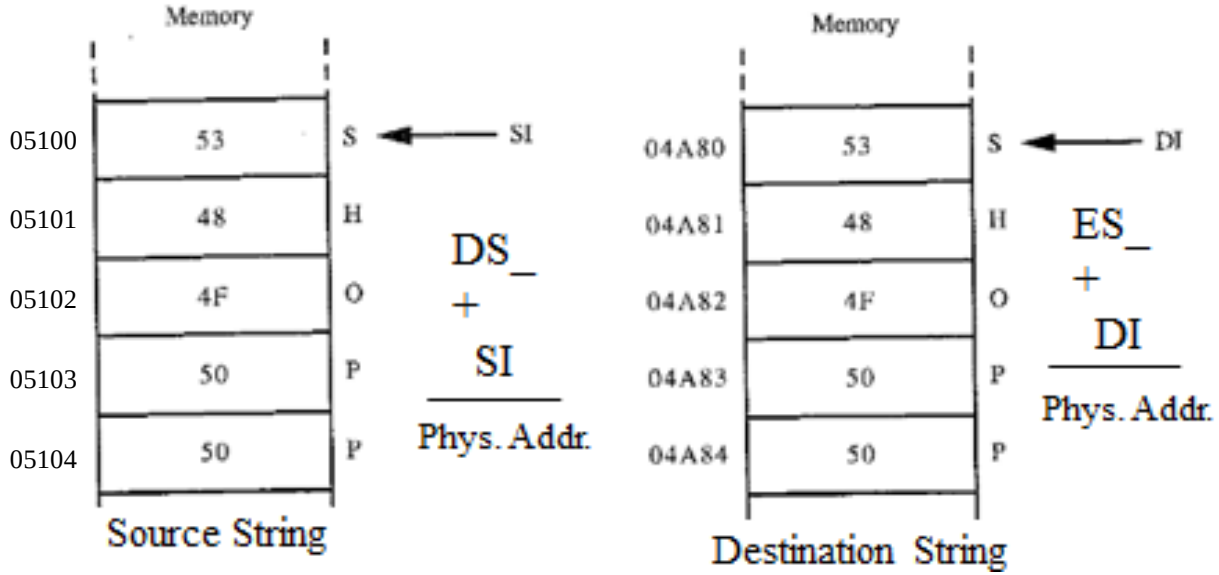
DF=0 (Instruction: CLD) SI & DI (Incremental Direction)

DF=1 (Instruction: STD) SI & DI (Decremental Direction)

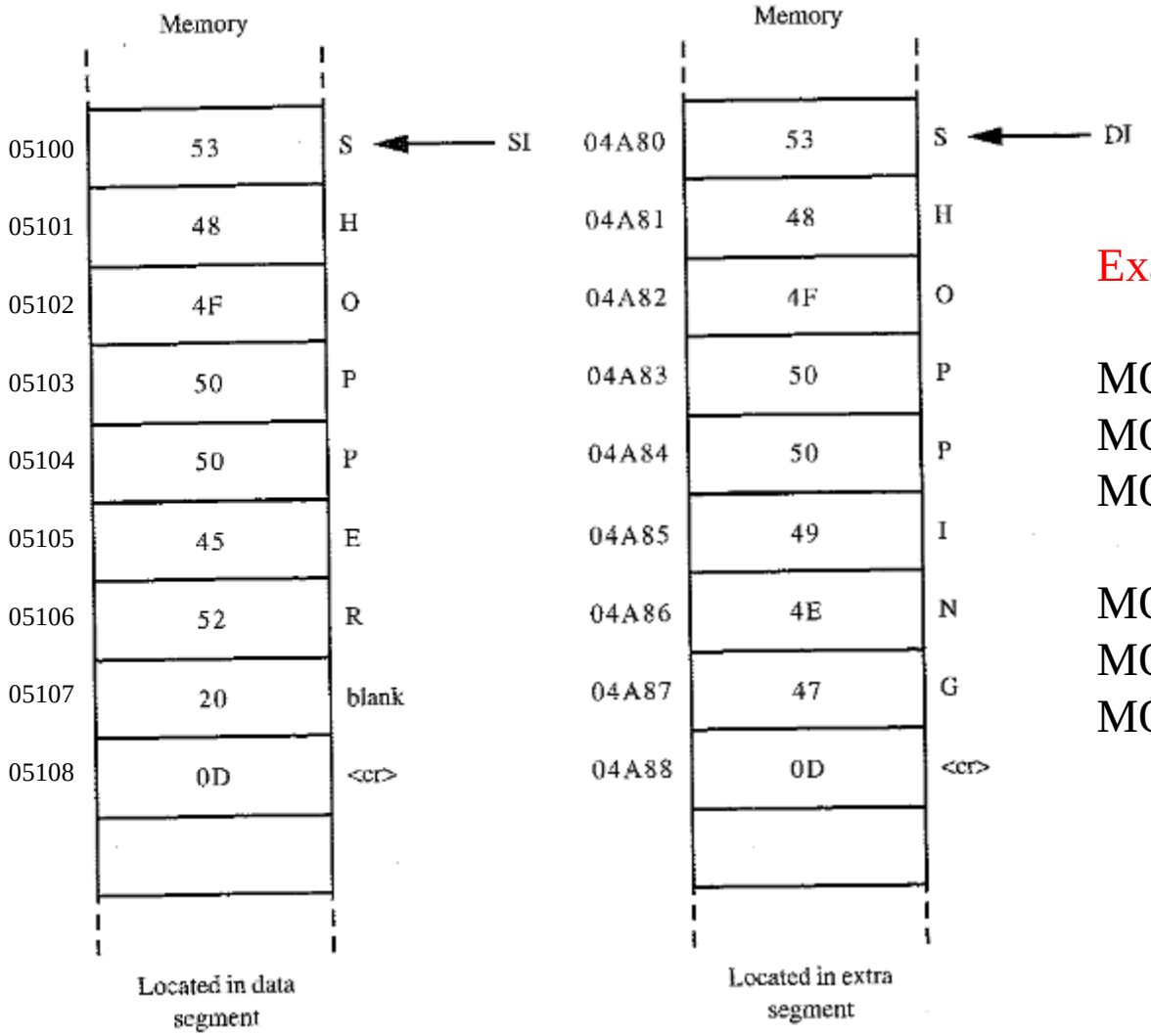
+ some strings instructions

The principle:

- The processor assumes that SI points to the first element of source string which has to be located in DS.
- It also assumes that DI points to the first element of destination string which has to be located in ES.
- Thus, before using string instructions the registers SI, DS, DI and ES have to be initialized



String Instructions



Example (Registers initialization):

MOV AX, 510H
MOV DS, AX
MOV SI, 0

MOV AX, 48AH
MOV ES, AX
MOV DI, 0

String Instructions
(Prefix)

While CX< > 0	REP		MOVS (Move String) STOS (Store String)
While ZF =1	REPE	REPZ	SCAS (Scan String) COMPS (Compare String)
While ZF =0	REPNE	REPNZ	

Thus three ways to repeat string operation:

Repeat while CX doesn't equal zero
Repeat while CX doesn't equal zero and the Zero Flag is set
Repeat while CX doesn't equal zero and the Zero Flag is cleared

String Instructions

(MOVS Destination string, Source string)

Make a copy of source string in the destination string

The name of string must be used in the operand field, DB and DW are used to define if they are byte or word strings.

STRINGA DB ‘SHOPPER ‘,0DH

STRINGB DB ‘SHOPPING‘,0DH

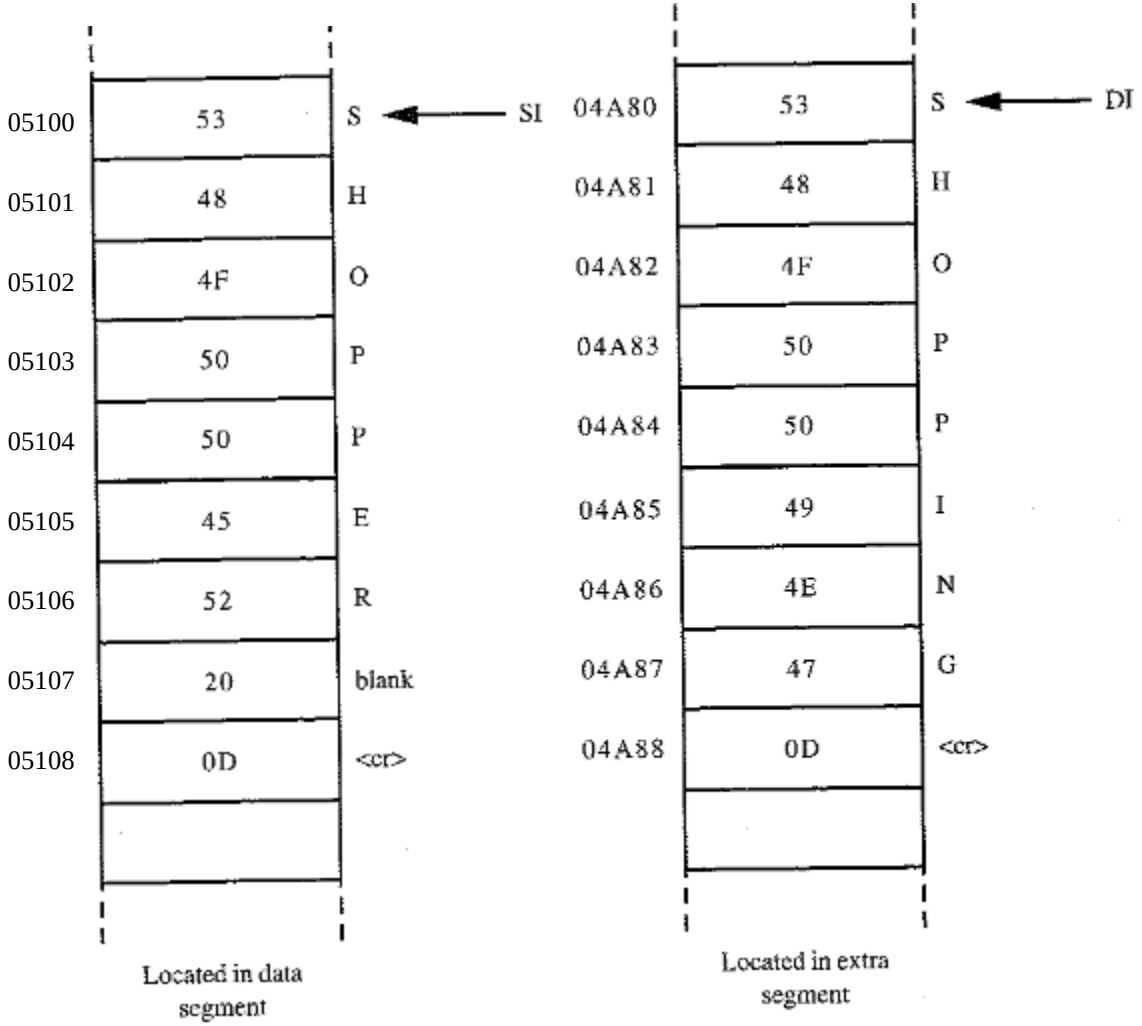
STRINGY DW 6566H

MOVS STRINGB, STRINGA

MOVSB/ MOVSW

Note: 0D carriage back ACII Code

0A New line ACII Code



String Instructions

(MOVS Destination string, Source string)

Example: Give the instructions to make a copy of the SHOPPER string. The starting address of destination string is 3000H. Incremental direction is supposed during the string operation. CX=9

```
MOV     AX,510H     ;source string segment-address
MOV     DS,AX
SUB     SI,SI       ;source string offset
MOV     AX,300H     ;destination string segment-address
MOV     ES,AX
SUB     DI,DI       ;destination string offset
CLD                 ;auto increment
REP                 ;repeat while CX <> 0
MOVSB               ;copy string
```

String Instructions

(CMPS Destination string, Source string)

Compare two string CMPSB, CMPSW

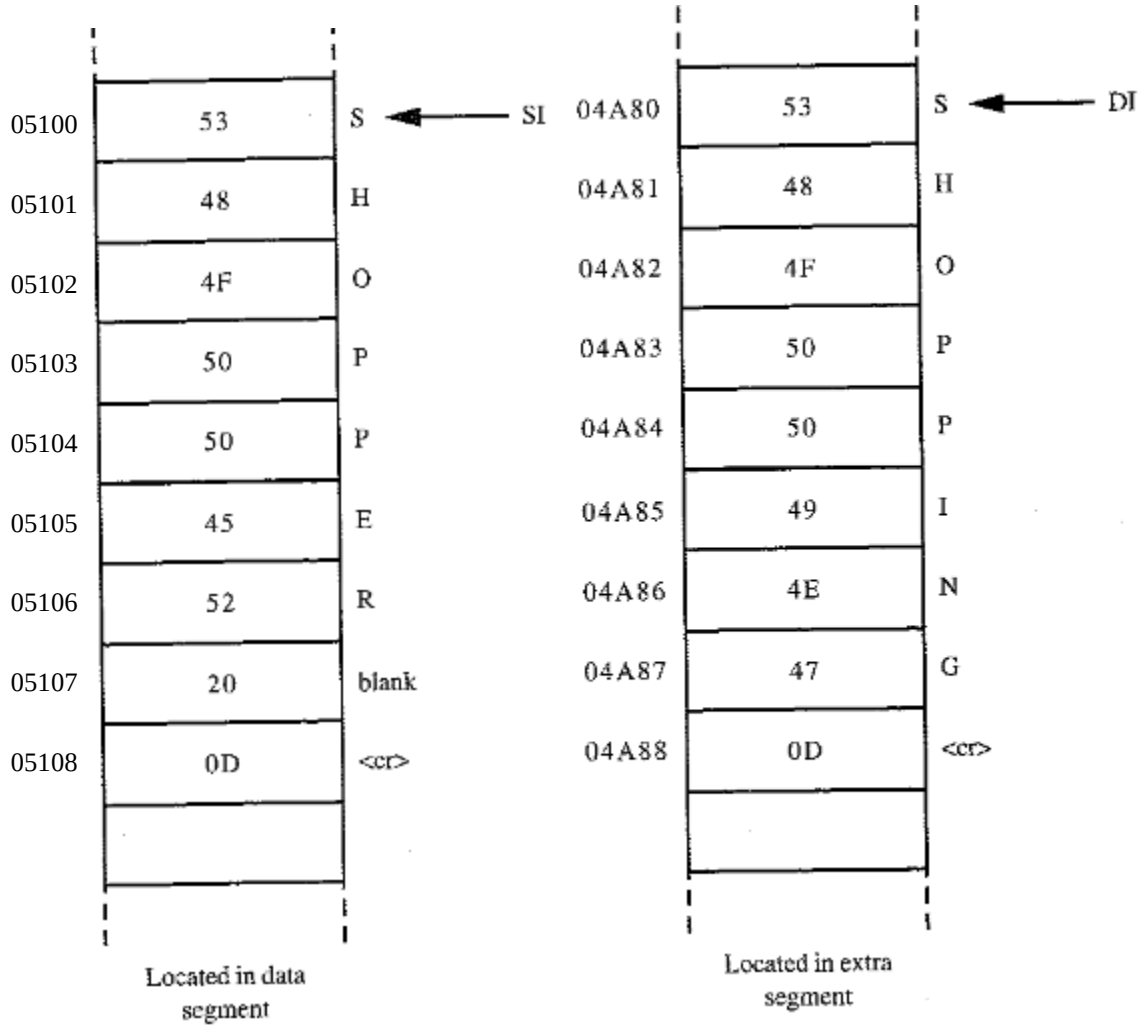
Example:

```
CLD
MOV CX, 4
REPZ CMPS STRINGA, STRTINGB
```

In this case the operation will be repeated 4 times until CX=0

```
CLD
MOV CX, 8
REPZ CMPS STRINGA, STRTINGB
```

In this case the operation will be stopped at the 6th byte.
'I' and 'E' are different thus ZF = 0



String Instructions

(SCAS Destination string)

Scan byte or word string SCASB, SCASW

Comparing each String element pointed by ES:DI with the value saved in AL or AX.

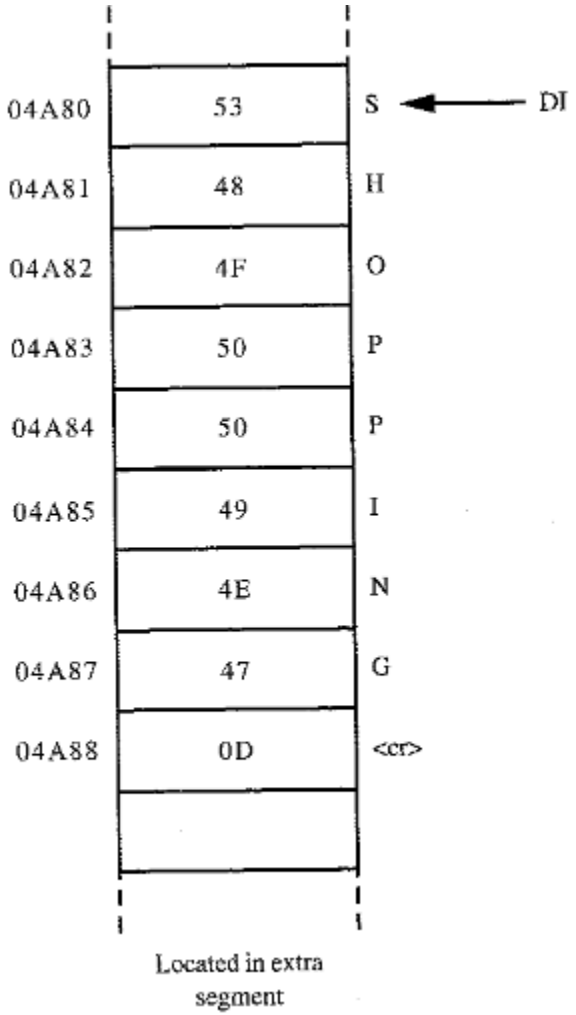
Example:

```
CLD
MOV CX, 9
MOV AL, 4EH
REPNZ SCASB
```

AL

4E

The scan will be allowed because there is no match until the memory arriving the address 04A86H. Where ZF = 1



String Instructions

(LODS Source string)

Load byte or word string LODSB, LODSW

Load the current String element pointed by DS:SI to the accumulator AL or AX and automatically advance SI to point to the next element ±1 byte or ±2 bytes

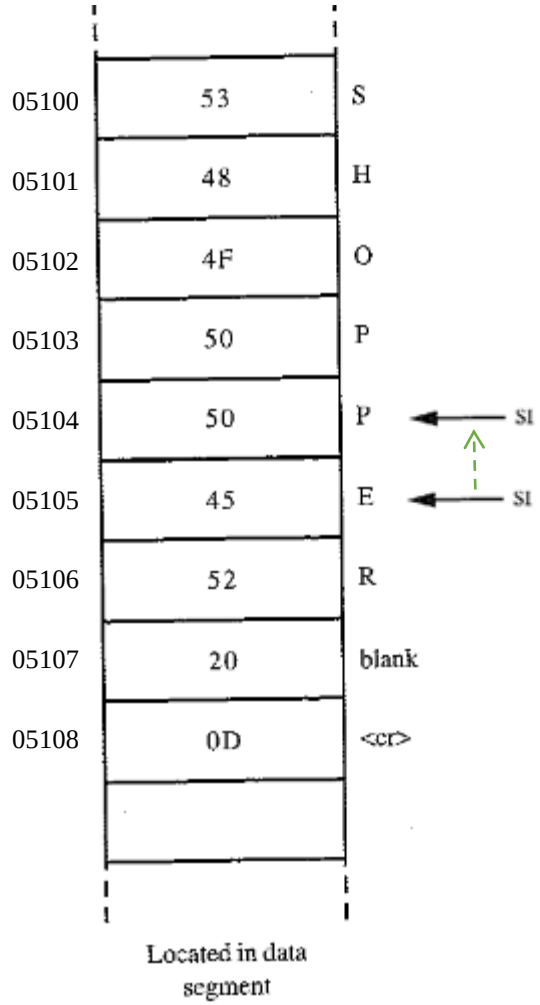
Example:

```
STD
LODSB
```

AL

45

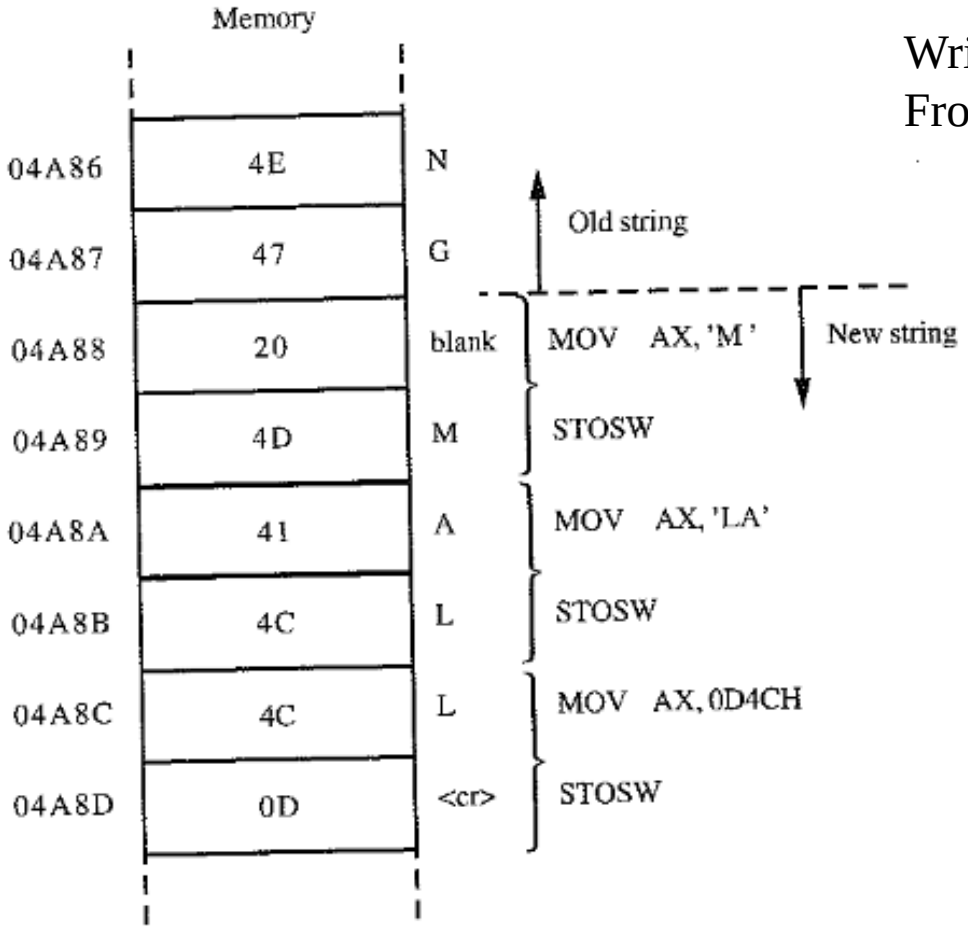
SI = SI -1



String Instructions

(STOS Destination string)

Store byte or word string STOSB, STOSW



Write elements of string into memory.
From AL or AX to a memory pointed by ES:DI

Example:

Modify the ‘SHOPPING’ by adding

the word MALL to it

```
MOV     AX,4A8H
MOV     ES,AX
MOV     DI,8
CLD
MOV     AX,’M  ‘
STOSW
MOV     AX,’LA’
STOSW
MOV     AX,0D4CH
STOSW
```

Example 4

Write a program which determines the sum of two numbers of 16 bits.
The address of the first number is 200H
The address of the second number is 202H
Save the result in the address 204H

```
MOV AX,[200 H]
ADD AX,[202 H]
MOV [204 H],AX
HLT
```

Example 5

Write a program to divide a number that allocates 4 bytes of a memory starting from the address 100H, by another number of 2 bytes starting from the memory address 200H.
Save the result in the address 300H and the residual in 302H.

```
MOV AX, [0100H]
MOV DX, [0102H]
MOV BX, [0200H]
DIV BX
MOV [0300H] , AX
MOV [0302H] , DX
HLT
```

Example 6

Write a program which determines the following arithmetic expression

$$y = x^2 - 2x + 6$$

x is stored in AL and the result y has to be returned in AX

```
MOV   BL, AL  ; Save a copy of input value
MUL   BL      ; Compute x^2
XCHG  DX,AX   ; Save temporary result
MOV   AL,2
MUL   BL      ; Compute 2x
SUB   DX,AX   ; Comute x^2-2x
XCHG  DX,AX   ; Get current result into AX
ADD   AX,6    ; Compute x^2 - 2x + 6
```

Example 7

Write a program to determine the length of block of memory starting from the address 100H. The block of memory has to be finished at any memory location that contains 77H. The maximal possible length of that upper mentioned block is FFH. Store the result at the memory location 200H.

```
        MOV SI,   0100H
        MOV AL, 77H
        MOV DL, 00H
        MOV CX, FFH
NEXT:   CMP AL , [SI]
        JZ ADCNT
        INC DL
        INC SI
        LOOP NEXT
        JMP END
ADCNT:  INC DL
END:    MOV [0200H],DL
        HLT
```

Example 8

Write a program to determine the biggest number saved in a block of unsigned numbers. The length of block is saved in the address 200H. The block is starting from the address 201H. Save the result in the address 100H. Save the address of biggest number in 101H.

```
        MOV SI,0202H
        MOV CL , [SI-2]
        MOV CH, 0
        DEC CX
        MOV AL , [SI-1]
        MOV DX , 0201H
NEXT:   CMP AL , [SI]
        JAE SKIP
        MOV AL , [SI]
        MOV DX , SI
SKIP:   INC SI
        LOOP NEXT
        MOV [0100H], AL
        MOV [0101H], DX
        HLT
```

Example 9

A group of signed numbers are already saved in a memory starting from the address 201H. Their length of their block is saved in the address 200H. Write a program to determine the number of negative numbers and save the result in the memory address 300H .

```
        MOV CL , [0200H]
        MOV CH, 00H
        MOV DL , 00H
        MOV SI, 0201H
NEXT:   MOV AL , [SI]
        TEST AL , 80H   (AND logic operation without saving result)
        JZ SKIP
        INC DL
SKIP:   INC SI
        LOOP NEXT
        MOV [0300H], DL
        HLT
```

Example 10

Write a program to read 100 temperature degrees from the input address 2233H and save all these readings in a memory starting with the address 200H

```
        MOV CX , 64H
        MOV SI, 0200H
        MOV DX , 2233H
NEXT:   IN AL , DX
        MOV [SI] , AL
        INC SI
        LOOP NEXT
        HLT
```

Example 11

Write a program to make a block of numbers in decremental order. The block length is 20H. It is starting from the address 200H.

```
        MOV SI , 0200H
        MOV CX , 0020H
FIRST:  MOV DI , SI
        INC DI
NEXT:   MOV AL , [SI]
        CMP AL , [DI]
        JAE SKIP
        XCHG AL , [DI]
        XCHG AL , [SI]
SKIP:   INC DI
        CMP DI , 0220H
        JNZ NEXT
        INC SI
        LOOP FIRST
        HLT
```