Mansoura University
Faculty of Computers and Information
Department of Computer Science
First Semester: 2020-2021

[CS214P] Assembly Language: Chapter 7

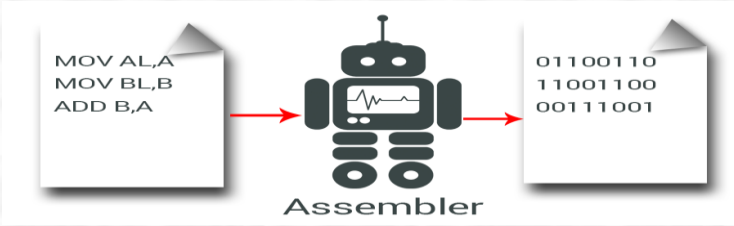
Grade: Third Year (Computer Science)

Sara El-Metwally, Ph.D.

Faculty of Computers and Information,

Mansoura University,

Egypt.



Computer Science Department
Faculty of Computers and Information
Mansoura University

Assembly Language

"Program Logic and Control"

Sara El-Metwally, Ph.D.
Faculty of Computers and Information,
Mansoura University, Egypt.

Email: sarah_almetwally4@mans.edu.eg
sara.elmetwally.2007@gmail.com

SHORT, NEAR and FAR addresses

- **Short** Address: limited to a distance of -128 (80H) to 127 (7FH) (1 byte long, same segment).
- **NEAR** Address: limited to a distance of -32768 (8000H) to 32767 (7FFFH) (2 bytes long, same segment)
- **FAR** address: (4 bytes long, another segment)

SHORT addresses

Example of Short Address

01111111	7FH	127
01111110	7EH	126
…		
…		
00000010	02H	2
00000001	01H	1
00000000	00H	0
11111111	FFH	2'com of -1
11111110	FEH	2'com of -2
…		
10000000	80H	2'com of -128

127 to -128

NEAR addresses

Example of Near Address

0111111111111111	7FFFH	32767
0111111111111110	7FFEH	32766
…		
…		
0000000000000010	0002H	2
0000000000000001	0001H	1
0000000000000000	0000H	0
1111111111111111	FFFFH	2'com of -1
1111111111111110	FFFEH	2'com of -2
…		
1000000000000000	8000H	2'com of -32768

32767 to -32768

SHORT, NEAR and FAR addresses

Instruction	SHORT	NEAR	FAR
JMP	yes	yes	Yes
Jnnn	yes	Yes (.386+)	No
LOOP	yes	no	No
CALL	N/A	yes	Yes

Backward & Forward JMP

```
L10:

JMP L10
```

Backward JUMP

```
JMP L10

L10:
```

Forward JUMP

LOOP

- **LOOPE/LOOPZ** (loop while equal/zero) continue looping as long as CX is not zero and the zero condition is set. Implicitly decrement CX by 1.

- **LOOPNE/LOOPNZ** (loop while not equal/ not zero) continue looping as long as CX is not zero and the zero condition is not set.

CMP

[label:]	CMP	R/M, R/M/I

- Only compare numeric data. CMPS for string comparing.
- The result affects AF, CF,OF, PF,SF, ZF

```
2  CMP DX,00
3  JE L10
4
5
6  L10:
```

Conditional JUMP Instructions

- **Jumps Based on Unsigned Data**

Symbol	Description	Flags Tested
JE/JZ	Jump Equal or Jump Zero.	ZF
JNE/JNZ	Jump Not Equal or Jump Not Zero.	ZF
JA/JNBE	Jump Above or Jump Not Below/Equal.	CF, ZF
JAE/JNB	Jump Above/Equal or Jump Not Below.	CF
JB/JNAE	Jump Below or Jump Not Above/Equal	CF
JBE/JNA	Jump Below/Equal or Jump Not Above	AF, CF

Conditional JUMP Instructions

- **Jumps Based on signed Data**

Symbol	Description	Flags Tested
JE/JZ	Jump Equal or Jump Zero.	ZF
JNE/JNZ	Jump Not Equal or Jump Not Zero.	ZF
JG/JNLE	Jump Greater or Jump Not Less/Equal.	OF, SF, ZF
JGE/JNL	Jump Greater/Equal or Jump Not Less.	OF, SF
JL/JNGE	Jump Less or Jump Not Greater/Equal	OF, SF
JLE/JNG	Jump Less/Equal or Jump Not Greater	OF, SF, ZF

Conditional JUMP Instructions

- **Special Arithmetic Test.**

Symbol	Description	Flags Tested
JCXZ	Jump if CX is Zero	None
JC	Jump Carry (same as JB)	CF
JNC	Jump No Carry	CF
JO	Jump Overflow	OF
JNO	Jump No Overflow	OF
JP/JPE	Jump Parity or Jump Parity Even	PF

Conditional JUMP Instructions

- **Special Arithmetic Test.**

Symbol	Description	Flags Tested
JNP/JPO	Jump No Parity or Jump Parity Odd.	PF
JS	Jump Sign (negative)	SF
JNS	Jump No Sign (positive)	SF

CALL

[label:]	CALL	Proc-name
[label:]	RET[n]	[immediate]

- Purpose of call instruction to transfer control to a called procedure.
- RET, returns from the called procedure.
- RETN for near returns.
- RETF for far returns.

CALL

```
.CODE
Main PROC FAR
CALL B10

MOV AH,4CH
INT21H
Main ENDP

B10 PROC NEAR
CALL C10

RET
B10 ENDP

C10 PROC NEAR
RET

C10 ENDP
END Main
```

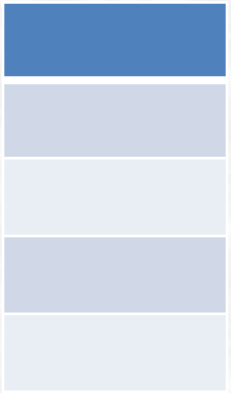
```
using System;

namespace ConsoleApplication
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("I am being Called from Program.cs");
            Console.ReadLine();
        }
    }
}
```

CALL

```
.STACK 64
        .CODE
0000    Main PROC FAR
0000    CALL B10
0003
0006    MOV AH,4CH
        INT21H
        Main ENDP

        B10 PROC NEAR
0008    CALL C10

000B    RET
        B10 ENDP

000C    C10 PROC NEAR
000D    RET

        C10 ENDP
        END Main
```

SP=64 = 0040H

SP Dec by 2



STACK

Hex value:
0040 – 0002 = **3E**

CALL

```
.STACK 64
        .CODE
0000    Main PROC FAR
0000    CALL B10
0003
0006    MOV AH,4CH
        INT21H
        Main ENDP

        B10 PROC NEAR
0008    CALL C10

000B    RET
        B10 ENDP

000C    C10 PROC NEAR
000D    RET

        C10 ENDP
        END Main
```

SP=003EH

003E	0300
003C	0B00

STACK

CALL

```
.STACK 64
        .CODE
0000    Main PROC FAR
0000    CALL B10
0003
0006    MOV AH,4CH
        INT21H
        Main ENDP

        B10 PROC NEAR
0008    CALL C10

000B    RET
        B10 ENDP

000C    C10 PROC NEAR
000D    RET

        C10 ENDP
        END Main
```

SP=003EH

003E	0300
003C	0B00

STACK

CALL

```
.STACK 64
        .CODE
0000    Main PROC FAR
0000    CALL B10
0003
0006    MOV AH,4CH
        INT21H
        Main ENDP

        B10 PROC NEAR
0008    CALL C10

000B    RET
        B10 ENDP

000C    C10 PROC NEAR
000D    RET

        C10 ENDP
        END Main
```

SP=003EH

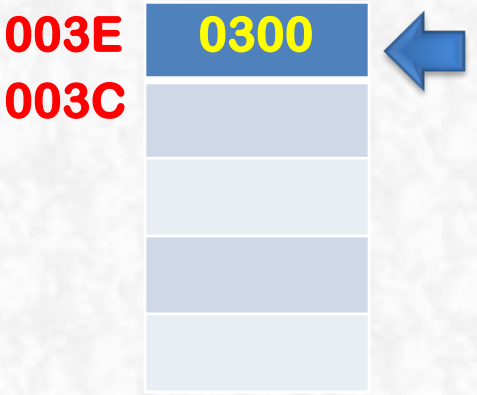
003E	0300
003C	0B00

STACK

CALL

```
.STACK 64
        .CODE
0000    Main PROC FAR
0000    CALL B10
0003
0006    MOV AH,4CH
        INT21H
        Main ENDP

        B10 PROC NEAR
0008    CALL C10

000B    RET
        B10 ENDP

000C    C10 PROC NEAR
000D    RET

        C10 ENDP
        END Main
```

SP=003EH



STACK

Passing Parameters

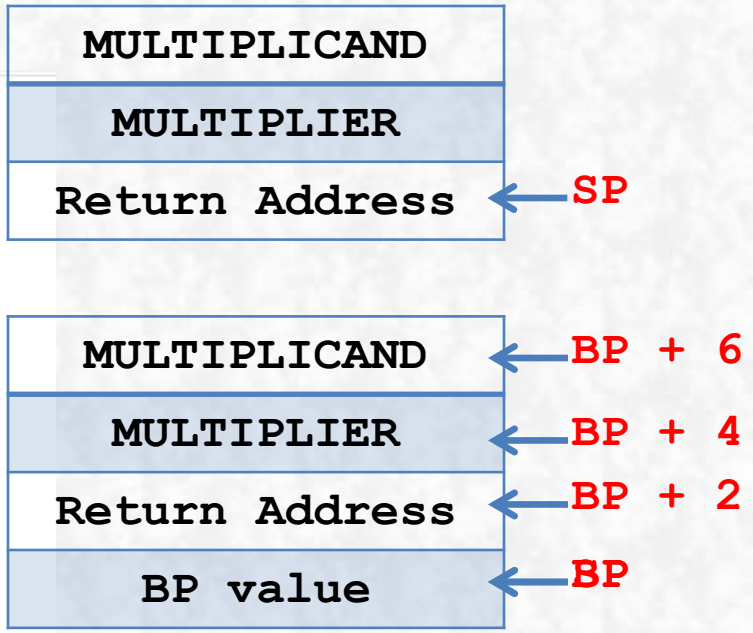
- **Passing Parameters by Value.**

```
1  MOV AX, MULTIPLICAND
2  MOV BX, MULTIPLIER
3  CALL M3MULT
4
5  M3MULT PROC NEAR
6         MUL BX
7         RET
8  M3MULT ENDP
9
10
```

Passing Parameters

- Passing Parameters by Value.

```
1  PUSH MULTIPLICAND
2  PUSH MULTIPLIER
3  CALL M3MULT
4
5  M3MULT  PROC NEAR
6          PUSH BP
7          MOV BP,SP
8          MOV AX,[BP+6]
9          MUL WORD PTR [BP+4]
10         POP BP
11         RET 4
12 M3MULT  ENDP
13
14
```



Passing Parameters

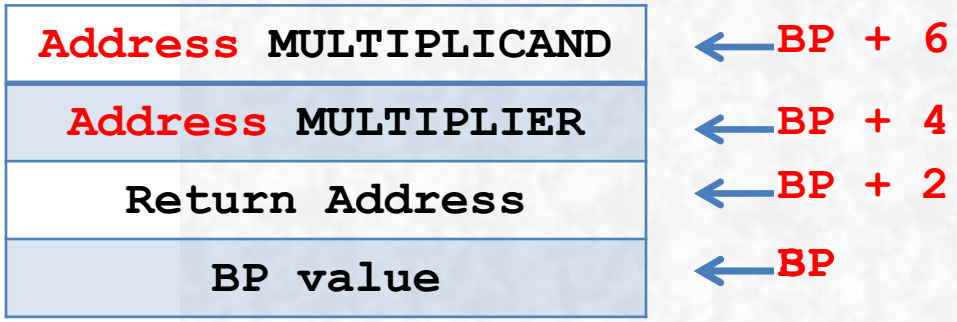
- Passing Parameters by Reference.

```
1  LEA BX, MULTIPLICAND
2  LEA SI, MULTIPLIER
3  CALL M30MULT
4
5  M30MULT PROC NEAR
6  MOV AX, [BX]
7  MUL WORD PTR [SI]
8  RET
9  M30MULT ENDP
10
```

Passing Parameters

- **Passing Parameters by Reference.**

```
1  PUSH OFFSET MULTIPLICAND
2  PUSH OFFSET MULTIPLIER
3  CALL M30MULT
4  M30MULT PROC NEAR
5  PUSH BP
6  MOV BP,SP
7  MOV BX, [BP+6]
8  MOV DI, [BP+4]
9  MOV AX, [BX]
10 MUL WORD PTR [DI]
11 POP BP
12 RET 4
13 M30MULT ENDP
```



Passing Parameters

- A common practice for a called procedure:

```
1  PUSH BP
2  MOV BP, SP
3  PUSHF
4  PUSHA
5
6  ......
7  .....
8
9  POPA
10 POPF
11 POP BP
12
13
```

BOOLEN OPERATIONS

Self Study!!