Department of Information and Communication Engineering

CSE-3102: Microprocessor and Interfacing Lab

Experiment No. 04:

Program control instructions in assembly language.

Objectives:

To load programs containing program control instructions to MDA-8086, execute the program in single step mode and verify the results. Some program control instructions are discussed below.

JUMP Commands:

Sometimes it is necessary to go from one line of program to another line without executing some intermediate lines. For this Jump commands are used. We can explain this with a simple example.

```
MOV AX, 3254H
MOV BX, 1F4BH
MOV CX, 412AH
ADD AX, CX
JMP S7
SUB AX, BX
S7: AND AX, BX
HLT
```

In this example S5 is a level. As we can see in fifth line JMP command is used. It makes the program to go from fifth line to S7 level that is seventh line. So sixth line will not be executed. There are two types of Jump commands. These are (i) Conditional jump and (ii) Unconditional Jump. Previous example is an unconditional jump. Conditional Jumps are like if statements. If some flags are affected only then these jump instructions executed. We can look at the following example,

```
MOV AX, 125BH
MOV BX, 125BH
MOV CX, 412AH
SUB AX, BX
JZ S5
DIV BX
S5: AND AX,CX
HLT
```

Clearly observe the code. In fourth line subtraction operation is performed. As both AX and BX have same value. Their subtracted value is 0. So ZF is set to 1. In fifth line JZ S5 is written. It means if ZF = 1 then go to S5, otherwise continue. As ZF = 1, program moves to seventh line. This is a conditional Jump. Some other conditional jumps are listed below.

MNEMONIC	CONDITION TESTED	"JUMP IF ... "
JA/JNBE	(CF or ZF)=O	above/not below nor equal
JAE/JNB	CF=O	above or equal/not below
JB/JNAE	CF=I	below/not above nor equal
JBE/JNA	(CF or ZF)= 1	below or equal/not above
JC	CF=I	carry
JE/JZ	ZF=I	equal/zero

JG/JNLE	((SF xor OF) or ZF) = 0	greater/not less nor equal
JGE/JNL	(SF xor OF)=O	greater or equal/not less
JL/JNGE	(SF xor OF) = 1	less/not greater nor equal
JLE/JNG	((SF xor OF) or ZF) = 1	less or equal/not greater
JNC	CF=O	not carry
JNE/JNZ	ZF=O	not equal/not zero
JNO	OF=O	not overflow
JNP/JPO	PF=O	not parity/parity odd
JNS	SF=O	not sign
JO	OF=I	overflow
JP/JPE	PF= 1	parity/parity equal
JS	SF= 1	sign
JCXZ	CX=0	Register cx = 0

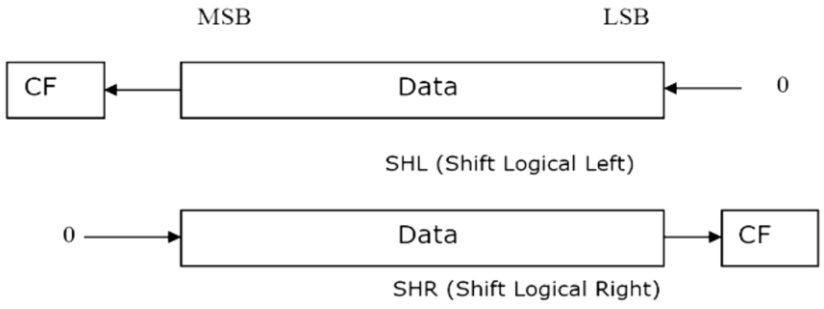
Note: "above" and "below" refer to the relationship of two unsigned values; "greater" and "less" refer to the relationship of two signed values.

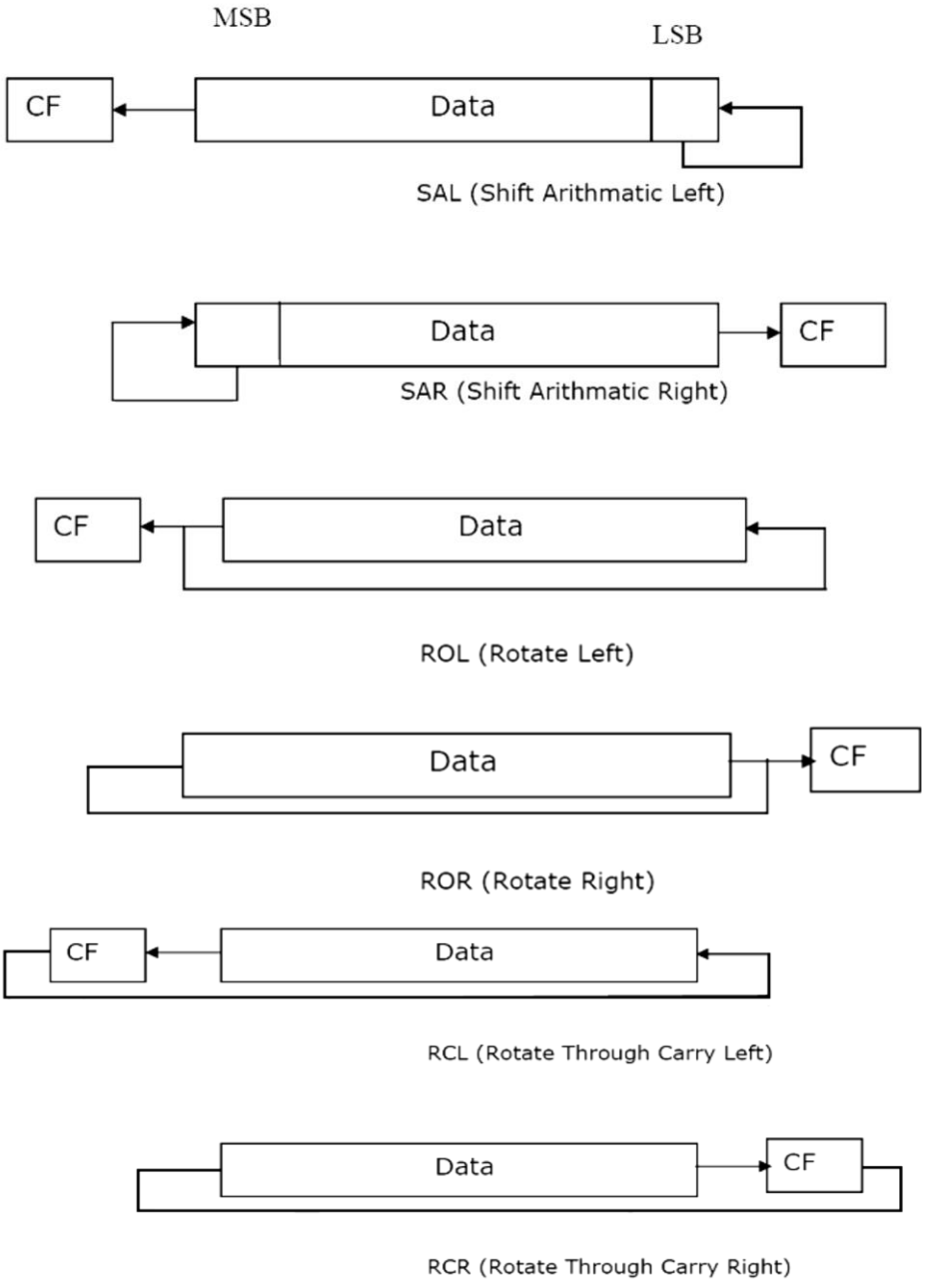
Program 1:

```
CODE SEGMENT
        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
        ORG 1000H
        MOV AX, 7A24H
        MOV BX, 95A3H
        ADD AX, BX
        JC NSTU
        ICE: OR AX, 23H
        JNZ LAST
        NSTU: MOV CX, 0FC7H
        SUB AX,CX
        JZ ICE
        LAST: HLT
CODE ENDS
        END
```

Shift and Rotate command:

Shift and Rotate commands are used to convert a number to another form where some bits are shifted or rotated. Basic difference between shift and rotate is shift command makes “fall of” bits at the end of register. Where rotate command makes “Wrap around” at the end of the register. There are both arithmetic (SAL and SAR) and logical (SHL and SHR) shift instructions. Graphical operations for these commands are shown below.





Some simple codes can be given to clarify the idea.

```
MOV CL,03H ;
MOV AX,02F3H        ; In binary 0000 0010 1111 0011
SHR AX,CL           ; In binary 0000 0000 0101 1110
```

In this procedure, SHR commands inserts 0's from left side. Each time a 0 is inserted right most bit is vanished from register content.

```
MOV CL,03H ;
MOV AX,82F3H        ; In binary 1000 0010 1111 0011
SAR AX,CL           ; In binary 1111 0000 0101 1110
```

In this procedure, SAR command inserts MSB content from left side. Each time it is inserted right most bit is vanished from register content.

```
MOV CL,03H ;
MOV AX,82F3H        ; In binary 1000 0010 1111 0011
ROR AX,CL           ; In binary 0111 0000 0101 1110
```

In this case, ROR instruction picks up the LSB and inserts it as MSB and so on.

Program 2:

```
CODE SEGMENT
        ASSUME  CS:CODE,DS:CODE,ES:CODE,SS:CODE
        ORG 1000H
        MOV AX, 0055H
        MOV DX, 0505H
        MOV CL, 3
        SAL AX, CL
        SAR DX, CL
        MOV CL, 2
        ROR AX, CL
        ROL DX, CL
        STC
        RCL AL, CL
        RCR DX, CL
        HLT
CODE ENDS
        END
```

Experiment Requirements:

1. 8086 microprocessor kit.
2. Assembler “MASM” and loader “LOD186”.
3. WinComm.

Experiment Procedures:

1. Write the program 1 in notepad and save the file as “filename.asm”. Place this file in the folder where “masm.exe” exists.
2. Go to command prompt and execute “masm.exe”. You will see the following message

```
Microsoft (R) Macro Assembler Version 5.10
Copyright (C) Misrosoft Corp 1981, 1988. All right reserved.

Source filename [.ASM]:
```

3. Follow the procedure given below to prepare machine code for your program:

```
Source filename [.ASM]:          filename Press ENTER
Object filename [filename.OBJ]:  Press ENTER
Source listing [NUL.LST]:        filename Press ENTER
Cross reference [NUL.CRF]:       Press ENTER
```

4. Execute “LOD186.exe”. You will see the following message

```
Paragon LOD186 Loader-Version 4.0h
Copyright (C) 1983 - 1986 Microtec Research Inc.
ALL RIGHT RESERVED.
Object/Command File [.OBJ]:
```

5. Follow the procedure given below to prepare HEX (ABS) file for your program:

```
Object/Command File [.OBJ]:          filename Press ENTER
Output Object File [C:filename.ABS]: Press ENTER
Map Filename [C:NUL.MAP]:            Press ENTER
```

**LOAD COMPLETE

6. Turn on the 8086 microprocessor kit
7. Open the "Wincomm" window. Press "L" then "Enter". You will see the following message:

```
** Serial Monitor 1.0 **
** Midas 335-0964/5 **
8086 >L Press ENTER
Down load start !!
```

8. Strike PgUp or F3 key of your keyboard. A new window will appear. Locate the "filename.ABS" file and open it.
9. You will observe that file download has started. A message like the following one will be shown:

```
:14100000B800008ED88EC0BB00208B078A6F028A4F038BEBB6
:101014003E8B5604268B76068B7E088B1E0A20CCCC
:0E20000012345678ABCDF0146853B1C41020E2
:00000001FF
OK completed !!
```

10. After loading the program, execute it in single step mode. Fill up the data table and verify the results.
11. Follow procedure 1 to 10 for program 2.

Data Table:

Program 1:

Offset Address	Instruction / Mnemonics	AX	BX	CX	DX	Set Flag Bit(s)	IP
	Initial Status						
	MOV AX, 7A24H						
	MOV BX, 95A3H						
	ADD AX, BX						
	JC NSTU						
	ICE : OR AX,23H						
	JNZ LAST						
	NSTU : MOV CX,0FC7H						
	SUB AX,CX						
	JZ ICE						
	LAST: HLT						

Program 2:

Offset Address	Instruction / Mnemonics	AX	BX	CX	DX	Set Flag Bit(s)	IP
	Initial Status						
	MOV AX, 0055H						
	MOV DX, 0505H						
	MOV CL, 3						
	SAL AX, CL						
	SAR DX, CL						
	MOV CL, 2						
	ROR AX, CL						
	ROL DX, CL						
	STC						
	RCL AL, CL						
	RCR DX, CL						

Report:

1. Discuss the effect of each instruction/ mnemonics that is used in this program.

References:

1. User's manual of MDA-8086 microprocessor kit, Midas Engineering, www.midaseng.com

2. "Assembly Language Programming and Organization of the IBM PC", Ytha Yu and Charles Marut, Mitchell McGraw-Hill.

Prepared by

Md. Sabbir Ejaz

Lecturer/ Dept. of ICE/ NSTU