Microprocessor Based System Design Laboratory (Gy)

Department of Computer Science and Technology, IIEST Shibpur

Date: August 09, 2024

Experiment No. 3: 8085 Assembly Language Programming (Continued)

Objective: To write simple programs in 8085 Assembly Language and test them at the 8085 SDK (System Development Kit).

Preamble: By this time you are familiar with

(a) writing simple 8085 assembly language programs,
(b) translating them into 8085 machine language,
(c) loading them in the RAM section of the SDK,
(d) executing them, and finally
(e) testing their effects to validate their "correctness".

Following are a few examples where simple C program segments are "*implemented*" by 8085 programs. Please consult the **2-page Instruction Set** and **16-page Instruction Set Reference Encyclopedia** for validating the program; and read the comment(s) against each instruction of the program.

Please try execution and validation of these programs.

Example 1. C statement: ***x = 5;*** */* x is an integer variable*/*
Let the memory location **2000**H be used for the variable ***x.***

Assembly Instruction	**Memory Address (Hex)**	**Machine Language (Hex)**	**Comment**
Alternative 1			
	2000	--	Memory location for variable ***x***. Note that ***x*** is not initialized.
MVI A,05H	2001	3E	Puts 5 in A.
	2002	05	
STA 2000H	2003	32	Stores contents of A in memory location 9000H
	2004	00	
	2005	20	
HLT or RST 5	2006	76 or EF	**HLT** halts 8085. You have to "**RESET**" the **kit** to make the kit working again. Check the contents of the memory location 2000H (by "**EXMEM**" key press), which should be 5. If you use **RST 5** (in place of **HLT**), the Interrupt Service Routine (**ISR,** at address **028H**) in your kit for the software interrupt instruction **RST 5** saves the contents of all the registers of the 8085 microprocessor at fixed memory locations **(see chapter 6, page 43, SDK User Manual)**. **Subsequently when you press "EXREG" key, the monitor program shows the-then contents of the registers, taking them from those locations of the memory.** Now you can check the contents of the register **A** (by "**EXREG**" key press) and the memory location 9000H (by "**EXMEM**" key press), both should be 5. Check contents of PC too.
Alternative 2			
	2000	--	Memory location for variable ***x***.
LXI H, 2000H	2001	21	Puts 9000H in HL register pair.

	2002	00	
	2003	20	
MVI M,05H	2004	36	
	2005	05	
HLT or RST 5	2006	76 or EF	See comments for this in **Alternative 1**. Check contents of 9000H location. or See comments for this in **Alternative 1**. Check contents of A, PC and 9000H.

Example 2: C Program Segment: -

int x = 0;
int y;
y = (x > 5);

Let the memory locations 2000H and 2001H be used for the variables ***x***, and ***y***, respectively.

Label	Assembly Instruction	Memory Address (Hex)	Machine Language (Hex)	Comment
	Data Area {	2000H	00	Memory location for x which is initialized to 0.
		2001H	--	Memory location for y which is **not** initialized.
		⋮		
		code may starts here		
	LDA 2000H	2100	3A	Loads contents of memory location 8000H (variable x) in register A.
		2101	00	
		2102	20	
	CPI 05	2103	FE	Compares A and 05H. This instruction sets Z flag (to 1) if (A) == 5, ie, contents of A is 5 CY flag (to 1) if (A) < 5, ie, contents of A is less than 5
		2104	05	
	LXI H, 2001H	2105	21	**HL** register pair now stores address of ***y***
		2106	01	
		2107	20	
	JC **Zero**	2108	DA	Jump to the label **ZERO** if CY is 1, that is, if A < 5
		2109	13	
		210A	21	
	JZ **Zero**	210B	CA	Jump to the label **ZERO** if Z is 1, that is, if A = 5
		210C	13	
		210D	21	
	MVI M,01H	210E	36	Stores 1 in ***y***
		210F	01	
	JMP **End**	2110	C3	
		2111	15	
		2112	21	
Zero	MVI M,00H	2113	36	Stores 0 in ***y***
		2114	00	
End	HLT or RST 5	2115	76 or EF	See comments for this in **Example 1**. Check contents of 2001H location. Or See comments for this in **Example 1**. Check contents of A, PC and 2001H.

Try your program with different values of x (say, x = 2, 9) by changing the contents of 2000H and then executing the program.

Example 3: C program segment -

```
int x = 9;
int y = 5;
if (x == y)
        z = 5;
else
        z=10;
```

where, ***x***, ***y*** and ***z*** are integer variables.

In the 8085 program (given below) variables ***x***, ***y*** and ***z*** are "implemented" as registers B, C, and D, respectively.

Note that there is no 8085 instruction for comparing registers B and C. Comparison is always with register A. Contents of B can be copied to A, so that, A and C can be compared later to do the job.

Label	**Assembly Instruction**	**Memory Address (Hex)**	**Machine Language (Hex)**	**Comment**
Start:	MVI B, 09H	2000	06	Initialize B (that is, variable ***x***) with 9
		2001	09	
	MVI C, 05H	2002	0E	Initialize C (that is, variable ***y***) with 5
		2003	05	
	MOV A, B	2004	78	Contents of B is copied to A.
	CMP C	2005	B9	Compares A and C. This instruction sets Z flag (to 1) if (A) == (C), ie, contents of A and C are same sets CY flag (to 1) if (A) < (C), ie, contents of A is less than C
	JZ MakeD5	2006	CA	If Z flag is 1, ie, in this case, (B)==(C) then jump to label **MakeD5** (Address 880AH), where D will be made 5, ie, D=5
		2007	0E	
		2008	20	
MakeD10:	MVI D,0AH	20809	16	The **else-part** of the C program segment. Make D=10.
		200A	0A	
	JMP End	200B	C3	Following instruction implements the if-part of the C program segment. Hence, jump to End (Address 880CH) to skip that part.
		200C	10	
		200D	20	
MakeD5:	MVI D,05H	200E	16	The **if-part** of the C program segment. Make D=5.
		200F	05	
End:	RST 5	2010	EF	**See the comments for this in Example 1 above. Note HLT instruction here (in place of RST 5 will not let you check the-then values of x, y, and z (ie, registers B, C, and D)**
After execution of the above program (GO-2-0-0-0-●) you may check the-then contents of registers A, B, C, D, PC, etc to validate the program.				
Change and try the above program for different values of *x* and *y* (that is, registers B and C)				

- **Please follow the tabular format of the above examples while writing your programs for the problems given below.**

- **You may, however, defer writing comments until you have a correct running program.**

In today's laboratory class **each of you** have to

1. **try the example programs given above**
2. write 8085 programs as specified below **in the format shown in the examples**. For each program, you **(i)** write the assembly code first, **(ii)** and then translate it to machine language using the Instruction Set Table provided to you, **(iii)** and finally you load, execute, and test the program.
3. **Get the program, that you have written in your notebook, signed by your teacher. Take a photo of your program and submit it at your Google Classroom.** The problems statements are available as classwork in the Classroom.

Programs

[Consult the **2-page Instruction Set** and **16-page Instruction Set Reference Encyclopedia** for choosing appropriate Instructions while writing programs]

1. To do what the following C program segment does. [Similar to Example 3. above]

   ```
   int x = 5;
   int y = 15;
   int z;
   z = (2*x > y)? 10: 5;
   ```

   where, ***x***, ***y*** and ***z*** are integer variables. Use memory locations to implement these variables (say, 2000H, 2001H, 2002H for ***x***, ***y***, and ***z***, respectively.) Please note that, as per C language, ***(2*x >= y)? 10: 5*** is a conditional expression, and its value is ***10*** if the condition ***(2*x >= y)*** is true; otherwise its value is ***5.***

   Modify your program for different initial values for *x* and *y*.

2. To do what the following C program segment does. [Similar to Example 2. above]

   ```
   int x = 9;
   int y = 7;
   int z;
   z = (x & 5); /* bitwise anding */
   ```

   Let the memory locations 2000H, 2001H, and 2002H be used for the variables ***x***, y and ***z***, respectively.

   Change and run your program for different initial values of *x and y*.

3. Repeat assignment 1. above with relational expression "2*x < y" (in place of "2*x > y").
4. Repeat assignment 1. above with relational expression "2*x == y" (in place of "2*x > y").
5. Repeat assignment 1. above with relational expression "2*x != y" (in place of "2*x > y").
6. Repeat assignment 1. above with relational expression "2*x >= y" (in place of "2*x > y").
7. Repeat assignment 1. above with relational expression "2*x <= y" (in place of "2*x > y").
8. To do what the following C program segment does.

   ```
   char ch='C', dept[10];
   switch (ch) {
           case 'A': strcpy(dept, "ARCH";
                     break;
           case 'C': strcpy(dept, "CST";
                     break;
   ```

```
        default: strcpy(dept, ”IIESTS”;
    }
```

where, ***ch*** is a character variable mapped to memory location 2000H, and ***dept[10]*** is a character array mapped to memory locations 2001H to 200AH in the memory. **Please note that characters are stored as their respective ASCII values.** That is ’A’ is stored as 65 (41H) and ”CST” needs 4 Bytes to store 67, 83, 84, 0 (that is, 43H, 53H, 54H, 00H).

Change and run your program for different initial values of *ch*.