Microprocessor based System Design Laboratory (GY)

Department of Computer Science and Technology, IIEST Shibpur

Date: August 2, 2024

Experiment No. 2: Familiarization with 8085 Instruction Set

Objective: To be familiar with the Instruction Set of 8085 Microprocessor through writing simple programs and testing them at the 8085 SDK (System Development Kit).

Preamble: By this time you are familiar with the SDK you are using in the laboratory. **Please ensure** that, **through the Monitor program** of the SDK, using the **keypad** and **display section**, you can now **interactively**

i. using "**EXMEM**" key (command) examine the contents of main memory and modify its contents (the RAM part, say, 2000H onward) to load a 8085 machine language program,
ii. using "**GO**" key (command) execute a program which is already available in the main memory of the SDK, and test the effect of the program for validating its expected result.

In general developing a program in the SDK, **for the time being**, will involve the following steps.

i. Write down the program in 8085 assembly language (shown below) in your notebook.
ii. Translate the assembly language program into machine language (shown below) by hand in in your notebook.
iii. Load the machine language program in the available RAM section of your SDK (using "**EXMEM**" key)
iv. Execute the program (using "**GO**" key)
v. If the effect of the program is not as expected go to step i.

For example, the following table contains a 8085 program (in **assembly** as well as **machine language**) with documentation (comments).

Label (Name)	**Assembly Instruction**	**Memory Address (Hex)**	**Machine Language (Hex)**	**Comment**
	MVI B, 22H	2000	06	Register B is initialized with 22H
		2001	22	
	MVI C, 82H	2002	0E	Register C is initialized with 82H
		2003	82	
	MOV A, B	2004	78	Contents of B (that is, 22H) is copied to A
	STA 2040H	2005	32	Contents of A is stored (copied) to Memory location 2040H
		2006	40	8085 is Little Endian, lower order byte of 2040H, that is, 40H comes first before 20H, the higher order byte.
		2007	20	
	RST 5	2008	EF	RST 5 saves the current contents of PC into stack and loads PC with 5 x 8 = 40, that is, 28H. Hence, after RST 5 instruction, 8085 will execute the instruction at location 28H which, in the SDK, falls inside the ROM (containing the Monitor program). **Actually, in your kit, the monitor program at 28H saves the CURRENT contents of the Registers somewhere in the RAM (see chapter 6, page 43, SDK User Manual) so that, subsequently when you press "EXREG" key, the monitor program can show the-then**

				contents of the registers, taking them from those locations of the memory.
After execution of the above program (GO-2-0-0-0-●) you may check the contents of the memory location @2040H (should contain 22H) as well as contents of registers A (should contain 22H), B (should contain 22H) and C (should contain 82H).				

In today's laboratory class, **EACH OF YOU** have to write 8085 programs as specified below **in the format shown in the previous table**.

For each program,

i. you write the assembly code first,
ii. and then translate it to machine language using the Instruction Set provided to you.
iii. You can now load, execute and test the program.

Get the program, that you have written in your notebook, signed by your teacher.

If you want to check the contents of the Registers after execution of your program, then end your program with "RST 5" instruction. After executing your program press the "EXREG" key; the monitor program would then show what the registers contained immediately before execution of RST 5 instruction.

However, after execution of your program, if you want to check the contents of some memory locations only, then you may end your program with "HLT" (halt) instruction. After executing your program, reset the SDK ("RESET" key), press the "EXMEM" key; and check the contents of the concerned memory location.

Assignment Programs

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

1. Add **51**H to the 8-bit number available in a register (say, **B**)
2. Add **51**H to the 8-bit number available in a memory location (say, **2000**H)
3. Add two 8-bit numbers available in 2 registers (say, **B** and **C**) and store their sum in a 3rd register (say, **D**).
4. Add two 8-bit numbers available in 2 memory locations (say, **2000**H and **2001**H) and store their sum in a 3rd memory location (say, **2002**H).
5. Swap the contents of 2 registers (say, **B** and **C**).
6. Swap the contents of 2 memory locations (say, **2000**H and **2111**H).
7. Storing the "**difference**" between two registers (say, **B** and **C**) in another register (say, **D**).
   [Hint: if (**B**) > (**C**), then **D** contains (**B**) – (**C**). Otherwise **D** contains (**C**) – (**B**). The "if-else" type of statement can be achieved using "**compare**" and "**jump**" type of instructions of 8085]

8. Adding contents of 10 consecutive memory locations (starting from address **2000**H) and storing the sum at address **200A**H. This is, as if, adding the sum of elements of an array of 10 integers.

   [Hint: A "loop" ("while", "for", etc) can be achieved through "if-else" and "goto" (that is, "jump") statements also.]

9. The starting address (say, **2100**H) of an array of 8-bit integers is available at address **2000**H (that is, location@**2000**H contains **00**H, and location@**2001**H contains **21**H). The number of elements of the array is available at **2002**H. Find the sum of the elements of the array and store the sum at address **2003**H.
10. Refer to program 9 above. Storing sum of elements in 1 Byte is likely to lead to overflow and wrong result. Modify the program so that the sum is computed as a 2-Byte number in a register-pair (say **DE**) and finally stored in the memory using 2 locations (say, @ **2003**H – **2004**H).