Introduction to Microprocessor Lab

Daya Sagar Baral
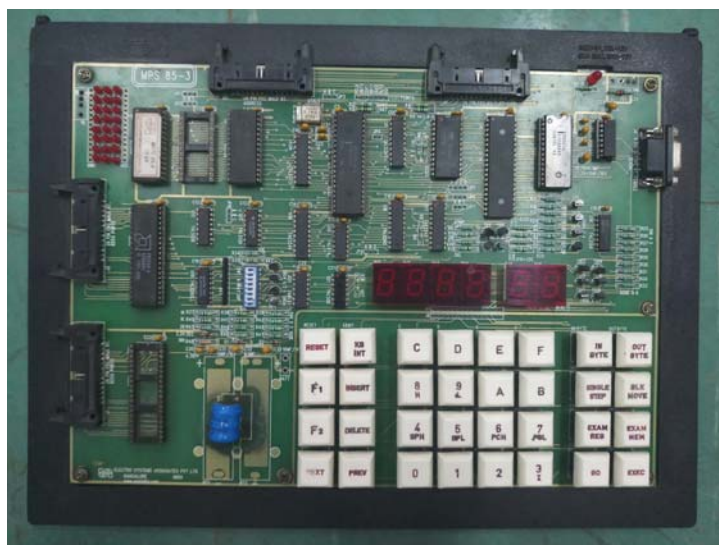
8085 Trainer Kit

Components of 8085 µP Trainer Kit

- **CPU :**
  - 8085 operated at 3.072 MHz
- **MEMORY** *:*
  - Three 28 pin JEDEC sockets offer 64K Bytes of memory
- **ROM :**
  - 16K Bytes of monitor program in 27128 EPROM
- **ROM / RAM :**
  - 16K Bytes user expansion using any EPROM/SRAM
- **RAM :**
  - 32K Bytes supplied using 62256 CMOS static RAM with battery backup

Components of 8085 µP Trainer Kit (Contd...)

- *8255 :*
  - (Two Nos) One 8255 is supplied to give 24 programmable I/O lines.
  - User can have 48 I/O lines by populating an additional 8255 in expansion socket.
- *8253 :*
  - 3 programmable interval timers
  - Timer 0 is used for implementing single step facility,
  - Timer 1 is used for generating baud clock
  - Timer 2 is available to the user.
- *8251*
  - For serial communication.
  - Supports all standard bauds from 110 to 19200.

Components of 8085 µP Trainer Kit (Contd…)

- ***8279 :***
  - To Control 32 keys keyboard and 6-digit, half-inch size, seven segment LED display.
- ***Bus Expansion :***
  - Fully de-multiplexed and buffered TTL compatible bus signals brought out through two 26 pin ribbon cable connectors for expansion.
- ***INTERFACE SIGNALS :***
  - ***Parellel I/O:*** 48 TTL compatible lines (2x8255) brought out through two 26 pin ribbon cable connectors.
  - ***Serial I/O :*** RS-232-C serial interface signals through onboard 9 pin D type female connector.

Program Types

- Machine Language Program
  - Instructions are given as machine instructions (binary instructions as interpreted by the computer by its hardware)
  - Programs are written with machine instructions (binary coded instructions)
- Assembly Language Program
  - The instructions are represented by Mnemonics (short form of word for an operation such as ADD for addition)
  - Programs are written with mnemonics.
- Low Level Language Program
  - Each statement of a program is converted to a single machine instruction
- High Level Language Program
  - Each statement of a program is converted to more than one machine instruction
  - Easy to write as it uses words from human language for keywords

Instruction Set

- Each microprocessor is equipped with components that can interpret some instruction provided to it
- A set of instructions that a microprocessor can interpret is called an instruction set
- Instruction set is provided with mnemonic of a instruction and its machine code in binary (normally in hexadecimal notation for shorter representation)
- 8085 has 74 instructions (with 246 binary values)
- Each instruction is represented by 8-bit binary value
- Instruction has operation code (or opcode) and operand.

Introduction to Microprocessor Lab by DSBaral   7
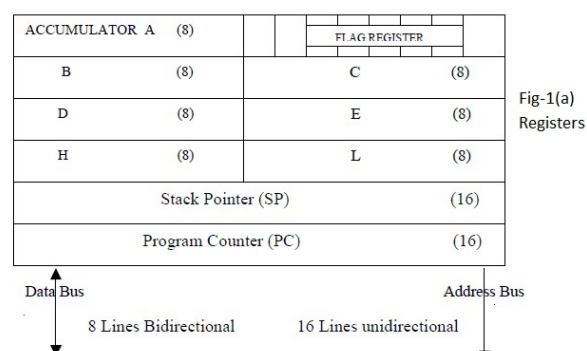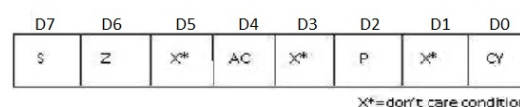

8085 Programming Architecture

Fig-1(a) Registers

Fig-1(b) Flags

X*=don't care condition

Introduction to Microprocessor Lab by DSBaral   8

8085 Microprocessor Instruction Set

| ARITHMETIC AND LOGICAL GROUP | DATA TRANSFER GROUP | BRANCH CONTROL GROUP | I/O AND MACHINE CONTROL | ASSEMBLER REFERENCE (Cont.) |

(Instruction set reference chart — opcodes for Add, Increment, Logical, Subtract, Decrement, Specials, Double Add, Rotate, Arith & Logical Immediate, Move, Move Immediate, Load Immediate, Load/Store, Jump, Call, Return, Restart, Stack Ops, Input/Output, Control, New Instructions (8085 Only), Assembler Reference, Pseudo Instruction, Operators, Constant Definition.)

Translating Assembly Language Program to Machine Language

- Manual/Hand Assembling
  - Write an assembly language program using mnemonics
  - From the instructions find the binary code for that mnemonic
  - Write the address for that instruction
  - Write the binary code for the mnemonics
  - Write the operands for the instruction
  - Find the size of that instruction and calculate the address for next instruction
  - Repeat the process for all the mnemonics.

Translating Assembly Language Program to Machine Language (Contd..)

- Automatic Assembling
  - Assembler software is used to convert the assembly language program to machine language
  - Write a program in a text editor
  - The assembly language program file is provided as input to the assembler
  - The assembler generates the machine language file
  - If any library is used linking is done to generate the executable file

An Example of Assembly Language Program

- A simple assembly language program
  - MVI A, 34
  - MVI B, 79
  - ADD B
  - HLT
- Program to display the output
  - MVI A, 80
  - OUT 43
  - MVI A, 34
  - MVI B, 79
  - ADD B
  - OUT 40
  - HLT

Manual Assembling of Example Programs

Assembling Language Program		Machine Language Program		
START:	MVI A, 34	8000	3E	34
	MVI B, 79	8002	06	79
	ADD B	8004	80	
	RST 5	8005	EF	

Manual Assembling of Example Programs (Contd…)

Assembling Language Program		Machine Language Program		
START:	MVI A, 80	8000	3E	80
	OUT 43	8002	D3	43
	MVI A, 34	8004	3E	34
	MVI B, 79	8006	06	79
	ADD B	8008	80	
	OUT 40	8009	D3	40
	RST 5	800B	EF	

Program Entry Process in Trainer Kit/Simulators

- After writing and manually assembling the program do the following steps to enter the program into the memory of the trainer kit
  - Press key EXAM MEM/SET MEM
  - Enter program start address first
  - Press key NEXT/INR
  - Enter instruction/data
  - Press key NEXT/INR
  - Enter instruction/data
  - Press key NEXT/INR
  - After finishing all the instruction/data press key RESET

Check the Entered Program in Trainer Kit/Simulators

- After entering the program in trainer kit do the following steps to check whether the entry is done correctly or not
  - Press key EXAM MEM/SET MEM
  - Enter program start address
  - Press key NEXT/INR
  - Check whether instruction/data in that address is correct or not
  - Press key NEXT/INR
  - Check whether instruction/data in that address is correct or not
  - Press key NEXT/INR
  - After checking all the locations press key RESET

Executing the Entered Program in Trainer Kit

- After entering and checking the entered program in trainer kit program can be executed
- Program execution can be done in two modes a) Full speed mode b) Single step mode
- To execute the program in full speed do the following steps
  - Press key GO
  - Enter the beginning address of the program
  - Press key EXEC

Executing the Entered Program in Trainer Kit (Contd…)

- To execute the program in single step do the following steps
  - Press key SINGLE STEP
  - Press key EXEC
  - Press key EXAM REG
  - Press respective key A or B to see the value of the register
  - Press key EXEC
  - Press key SINGLE STEP
  - Press key NEXT
  - Press key EXEC
  - Press key EXAM REG
  - Press respective key A or B or EXAM MEM to see the value of the register or memory
  - Press key EXEC
  - Press key SINGLE STEP to resume single step operation
  - After completing the execution press key RESET

Executing the Entered Program in Simulators

- Different simulators have different ways provided for program execution.
- For full speed execution simulators may provide different sequences.
- Some simulators provide an area where we enter the starting address of the program and press run on full speed or step by step mode
- Some simulators provide the exact sequence as in the physical single board computer (or trainer kit)
- For single step (step by step) execution simulators may provide menu item or a button to set the execution in single step mode.
  - We can provide the starting address of the program then press the step by step key (or menu item) or some key sequence in keyboard of a computer one at time.