Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Project Report

on

"Find the square of the given numbers from memory location 6800H and store the result from memory location 8000H. 5"

Submitted by:

Prajwol Lamichhane (26)

Prakash Shrestha (50)

Abhay Raut Chettri (43)

Ekta Chaudhary (11)

Submitted to:

Dr. Gajendra Sharma

Associate Professor

Department of Computer Science and Engineering

Submission Date: 15/07/2018

CERTIFICATION

PROJECT REPORT

ON

“Find the square of the given numbers from memory location 6800H and store the result from memory location 8000H. ”

Approved by:

Dr. Gajendra Sharma

Associate Professor

_____	_____	_____
(Signature)	(Name)	(Date)

Abstract

This project allows us to know and understand assembly language by simulating the operation of 8085 microprocessor chip. The program allows an eight-bit data to store in memory. This sort of program is heavily used in PLC and lower level bit manipulation. Beginner assembly programmers can understand and code this program to get and understanding of how microprocessor typically functions.

Acknowledgement

This academic project work has been completed with suggestion, guidance & help from many individuals. We are deeply indebted to those who have contributed in this developmental study.

Forwarding the vote of thankfulness, we record our appreciation to our esteemed Kathmandu University for having involved us in practical based syllabus. We would like to sincerely thank our project supervisor Dr. Gajendra Sharma, our professor, for providing his invaluable guidance and comments in every step of our project.

The support of our colleagues and others who helped us directly or indirectly in our project cannot be forgotten at this juncture.

Table of Contents

Table of Figures

Chapter 1: Introduction

A microprocessor is a computer processor which incorporates the functions of a computer's central processing unit (CPU) on a single integrated circuit (IC), or at most a few integrated circuits. A microprocessor is a multipurpose, programmable, clock driven, register based, digital-integrated circuit that accepts binary data as input, processes it according to instructions and provides results as output. Microprocessors contain both combinational logic and sequential digital logic. Microprocessors operate on numbers and symbols represented in the binary numeral system.

An assembly language is a low-level programming language for microprocessors and other programmable devices. It is not just a single language, but rather a group of languages. Assembly code is converted into executable machine code by a utility program referred to as an assembler. Assembly language uses a mnemonic to represent each low-level machine instruction or opcode.

1.1 Objectives

- To understand assembly language.
- To understand and execute many binary information.
- To learn to store data in assembly programming language.

1.2 8085 Microprocessor

Intel 8085("eighty-eighty-five") is an 8-bit microprocessor designed by Intel in 1977 using NMOS technology.

It has the following configuration:

- 8-bit data bus
- 16-bit address bus, which can address upto 64KB
- A 16-bit program counter
- A 16-bit stack pointer
- Six 8-bit registers arranged in pairs: BC DE, HL

1.3 Instruction Set of Intel 8085 Microprocessor

An Instruction is a binary pattern designed inside a microprocessor to perform a specific function. The instruction set of a microprocessor is the group of the instructions that determines what functions of the microprocessor can perform. The programmer can execute a program in assembly language using these instructions. These instructions can be classified into the following five functional categories:

1. Data Transfer Operations
2. Arithmetic Operations
3. Logical Operations
4. Branching Operations
5. Machine Control Operations

1.3.1 Data Transfer Operations

Instructions, which copies data from one register to another register, from memory to register or register to memory, without modifying the contents of the source. These instructions copy data from source to destination. Examples are: MOV, MVI, LXI, LDA, STA etc. For example, when MOV A, B is executed the content of the register B is copied into the register A, and the content of register B remains unaltered. Similarly, when LDA 2500 is executed the content of the memory location 2500 is loaded into the accumulator. But the content of the memory location 2500 remains unaltered.

Opcode	Operand	Explanation of Instruction	Description
MOV	**Rd, Rs** **M, Rs** **Rd, M**	Copy from source(Rs) to destination(Rd)	This instruction copies the contents of the source register into the destination register; the contents of the source register are not altered. If one of the operands is a memory location, its location is specified by the contents of the HL registers.

			Example: MOV B, C or MOV B, M
MVI	**Rd, data** **M, data**	Move immediate 8-bit	The 8-bit data is stored in the destination register or memory. If the operand is a memory location, its location is specified by the contents of the HL registers. Example: MVI B, 57H or MVI M, 32H

1.3.2 Arithmetic Operations

These instructions perform arithmetic operations such as addition, subtraction, increment or decrement of the content of a register or memory. Examples are: ADD, SUB, INR, DAD etc.

Opcode	Operand	Explanation of Instruction	Description
ADD	**R** **M**	Add register or memory, to accumulator	The contents of the operand (register or memory) are added to the contents of the accumulator and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the addition. Example: ADD B or ADD M
SUB	**R** **M**	Subtract register or memory from accumulator	The contents of the operand (register or memory) are subtracted from the contents of the accumulator, and the result is stored in the accumulator. If the operand is a memory location, its location is specified by the contents of the HL registers. All flags are modified to reflect the result of the subtraction. Example: SUB B or SUB M

Opcode	Operand	Explanation of Instruction	Description
CMP	**R** **M**	Compare register or memory with accumulator	The contents of the operand (register or memory) are M compared with the contents of the accumulator. Both contents are preserved. The result of the comparison is shown by setting the flags of the PSW as follows: if (A) < (reg/mem): carry flag is set if (A) = (reg/mem): zero flag is set if (A) > (reg/mem): carry and zero flags are reset Example: CMP B or CMP M

1.3.3 Logical Operations

The Instructions perform various logical operations such as AND, OR, compare, rotate etc. with the contents of the accumulator. Examples are: ANA, XRA, ORA, CMP, and RAL etc.

XRA	**R** **M**	Exclusive OR register or memory with accumulator	The contents of the accumulator are Exclusive ORed with M the contents of the operand (register or memory), and the result is placed in the accumulator. If the operand is a memory location, its address is specified by the contents of HL registers. S, Z, P are modified to reflect the result of the operation. CY and AC are reset. Example: XRA B or XRA M

1.3.4 Branching Operations

This group of instructions alters the sequence of program execution either conditionally or unconditionally. This includes the instructions for conditional and unconditional jump, subroutine call and return, and restart. Examples are: JMP, JC, JZ, CALL, CZ, RST etc.

Opcode	Operand	Explanation of Instruction	Description
NOP	**none**	No operation	No operation is performed. The instruction is fetched and decoded. However, no operation is executed. Example: NOP
HLT	**none**	Halt and enter wait state	The CPU finishes executing the current instruction and halts any further execution. An interrupt or reset is necessary to exit from the halt state. Example: HLT

Opcode	Operand	Explanation of Instruction	Description
JMP	**16-bit address**	Jump unconditionally	The program sequence is transferred to the memory location specified by the 16-bit address given in the operand. Example: JMP 2034H or JMP XYZ

1.3.5 Machine Control Operations

These instructions include the instructions for input/output ports, stack and machine control.
Examples are: IN, OUT, PUSH, POP, and HLT etc.

Chapter 2: Project Description and Implementation

2.1 Fetch and execution model

Whenever microprocessor has to execute any instruction, the instruction is first received by the instruction register (8-bit) through A/D (address/data) bus. The instruction is then passed on to the instruction decoder where it is split, decoded and then passed to the control unit to generate necessary control signals for its execution. Besides doing this timing and control unit also checks whether any internal or external interrupt has occurred. If there is any kind of interrupt request this unit stops the execution of normal sequence of instructions to respond to it (interrupt) by generating the required control signals.

To execute any instruction µP has to pursue the following steps:

1) Identify the memory location from where the instruction is to be fetched

2) Decode the instruction using instruction decoder

3) Perform the function specified by the decoded instruction

All these operations are performed within a given time interval, which is provided by the clock of the system. With reference to the above-mentioned operations certain terms can be defined. The definitions are as follows:

1) **Instruction cycle:** Time required for completing the execution of an instruction is known as instruction cycle. The 8085 instructions cycle consists of one to six machines cycles or operations.

2) **Machine cycle:** It is the time required for completing a single operation. This operation can be accessing memory for read/write operation or accessing I/O device. There can be 3 to 6 clock periods or T-states in a machine cycle.

3) **T-states or clock cycles/periods (CLK):** T-state is equivalent to one clock period It is the time in which only a subdivision of the operation can be performed. The total number of T-states determines the size of the machine cycle required to perform an operation.

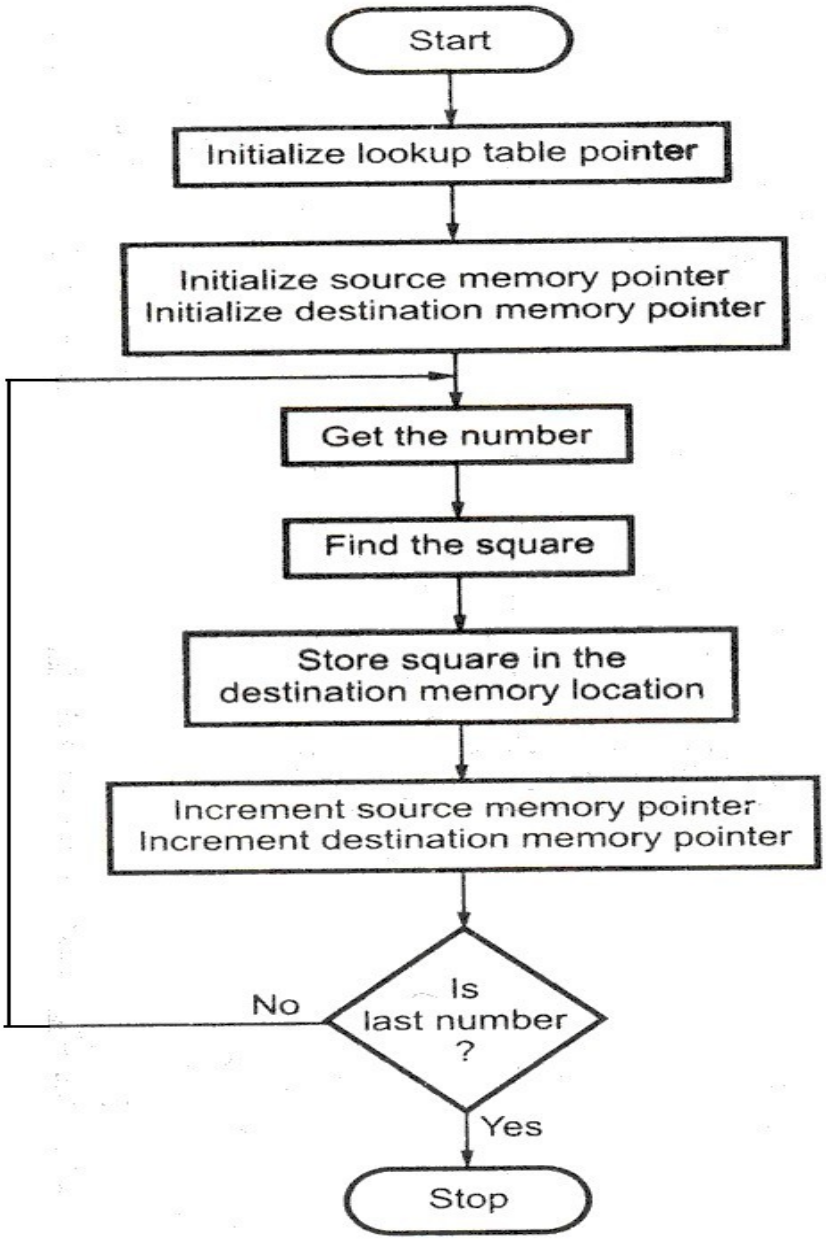
2.2 Tools used:

Programming language: Assembly Language

Software: 8085 simulator

OS: windows 10

2.3 Flowchart



2.4 Code

Program 1: DIRECT ADDRESSING METHOD

Sample problem

2200H = 4H

2201H= 9AH

2202H= 52H

2203H= 89H

2204H= 3FH

Result = 89H + 3FH = C8H

2300H= H Lower byte

2301H = H Higher byte

```
1.          LXI H, 6900H      :"Initialize lookup table pointer"
2.          LXI D, 6800H      :"Initialize source memory pointer"
3.          LXI B, 8000H      :"Initialize destination memory pointer"
4.          BACK: LDAX D      :"Get the number"
5.          MOV L, A          :"A point to the square"
6.          MOV A, M          :"Get the square"
7.          STAX B            :"Store the result at destination memory location"
8.          INX D             :"Increment source memory pointer"
9.          INX B             :"Increment destination memory pointer"
10.         MOV A, C
11.         CPI 05H           :"Check for last number"
12.         JNZ BACK          :"If not repeat"
13.         HLT               :"Terminate program execution"
```

Program 2: INDIRECT ADDRESSING METHOD

HLT: Terminate program execution

#The result of both programs will be the same.

2.5 Program Execution

Once the program is executed, the program is loaded on to the memory. As seen in the figure below, each of instruction is stored in a certain memory address which will be executed.

Figure 2: Screenshot of Direct Addressing

CHAPTER 3: Conclusion

Through this project, the basic understanding of assembly language was obtained. The various instruction sets were familiarized from basic logical instruction sets to advance low-level bit manipulation. An assembly language is a low-level programming language for microprocessors and other programmable devices. An assembly language implements a symbolic representation of the machine code needed to program a given CPU architecture.

Assembly language is also known as assembly code. The term is often also used synonymously with 2GL. An assembly language is the most basic programming language available for any processor. With assembly language, a programmer works only with operations that are implemented directly on the physical CPU.A microprocessor having separate bidirectional instruction and data busses is disclosed which allows the fetching of instructions from a program memory to be overlapped with the execution of instructions previously fetched. Program instructions are stored in an internal read-only-memory and/or in an external read-only-memory. Variable data is stored in an internal register array. During a given machine cycle, a data word in the register array can be transferred to an arithmetic-logic unit by a bidirectional data bus. The result of the operation performed by the arithmetic-logic unit can be transferred by the data bus back to the register array and stored in the selected location during the same machine cycle.

Reference

1. Coffron, J. (1988). *Microprocessor programming, troubleshooting, and interfacing the Z80, 8080, and 8085*. Englewood Cliffs, N.J.: Prentice-Hall.
2. Routt, W. (2007). *Microprocessor architecture, programming, and systems featuring the 8085*. Clifton Park, NY: Thomson Delmar Learning.
3. Kani, A. (2013). *8085 microprocessor and its applications*. New Delhi: Tata McGraw-Hill.
4. Blackstone, W., Christian, E., Chitty, J., Lee, T., Hovenden, J. and Ryland, A. (1893). *Commentaries on the laws of England*. Philadelphia: J.B. Lippincott.
5. www.tutorialspoint.com. (2018). Microprocessor Tutorial. [online] Available at: https://www.tutorialspoint.com/microprocessor