**DEPARTMENT OF**

**SCHOOL OF COMPUTING**

**College of Engineering and Technology**

**SRM Institute of Science and Technology**

MINI PROJECT REPORT

ODD Semester, 2023-2024

Lab code & Sub Name : 21CSS201T & Computer Organization and Architecture

Year & Semester : II & III

Project Title : Implement matrix multiplication using 8085 assembly language

Lab Supervisor **:** Dr. Akshaya

Team Members : 1. Gauri Gupta (RA2211026010359)

2. Neelansh Bhargava (RA2211026010360)

|  |  |  |
| --- | --- | --- |
| **Particulars** | **Max. Marks** | **Marks Obtained** |
|  |  | Gauri Gupta Neelansh Bhargava Mrinalini Vaish |
| Program and Execution | 20 |  |
| Demo verification &viva | 15 |  |
| Project Report | 05 |  |
| **Total** | **40** |  |

3. Mrinalini Vaish (RA2211026010365)

**Date :**

**Staff Name :**

**Signature :**

**Implement Matrix Multiplication using 8085 Assembly Language**

**OBJECTIVE:**

To implement Matrix Multiplication using assembly language.

**ABSTRACT:**

Two matrices can only be multiplied if their orders are of the form m × n and n × p where m, n, p ∈ Z+. In this project, we intend to multiply matrices of order 1 × n & n × 1. Later on, we may implement it for general orders.

**INTRODUCTION:**

Multiplying two matrices of order m × n and n × p where m, n, p ∈ Z+ is an O(n3) where n is the maximum of m, n, p. The project seeks to implement matrix multiplication for smaller-order matrices on an Intel 8085 Microprocessor. As you compile the program step by step using GNUSim 8085 Microprocessor you could visualize each row of the product matrix being filled. As there is no direct multiplication operation available in 8085 Instructions, we intend to multiply numbers through repeated addition methods using a loop. In order to traverse through a row in Matrix 1 & a column in Matrix 2, we first load the starting address of the row and column in stack and HL pair respectively. For traversing through rows and columns we swap the values in the HL register pair and the top of the stack and increment them. We call multiplication sub-routine as and when we require multiplication of 2 numbers.

**HARDWARE/ SOFTWARE REQUIREMENTS:**

Hardware requirements –

1. 8085 Microprocessor
2. System clock
3. Memory – RAM and ROM
4. Input/Output Devices
5. Address Bus and Data Bus
6. Logic Circuits
7. Debugging Tools

Software requirements –

1. 8085 Assembly Language Editor
2. 8085 Assembler
3. Simulator
4. Debugging Tools
5. Operating System

**ALGORITHM:**

Algorithm for Matrix Multiplication

for ( int i = 0 ; i < rowNo ; i++ ){

for ( int j = 0 ; j < colNo ; j++ ){

for ( int k = 0 ; k < p ; k++ ){

result[i][j] = result[i][j] + first[c][k]\*second[k][d];

}

}

}

Algorithm for Multiplication

int number1, number2;

while( number2 != 0 ){

number1 = number1 + number2;

number2--;

}

Matrix Multiplication Algorithm for 8085 for 1 × n & n × 1

Load HL pair with Address of 1st row and 1st column of Matrix1

Load Stack with Address of 1st row and 1st column of Matrix2

MVI E, 00H

Method: Load value in HL memory location in A register

Load value of stack in B register

Call multiply subroutine to multiply two numbers

ADD E

STA E

INX H

XCHG

INX H

JMP Method

Store the value of E in specified memory Location

Matrix Multiplication Algorithm for 8085 for 2 × 2 & 2 × 2

Load C with 2

Load D with 2

Method1: DCR C

Method: Multiply row 1 vector with column 1 vector using algo defined above DCR D

if D != 0:

if C != 0: Load HL pair with add. of Matrix1[1][1]

Call Method

if C == 0: Load HL pair with add. of Matrix2[2][1]

Call Method

if D == 0:

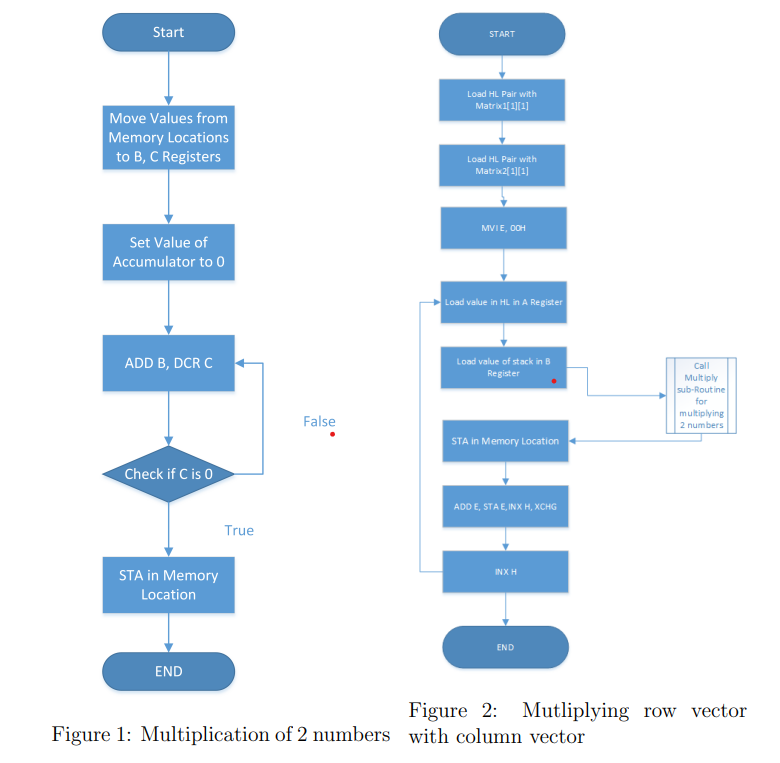
Load HL pair with add. of Matrix1[2][1]

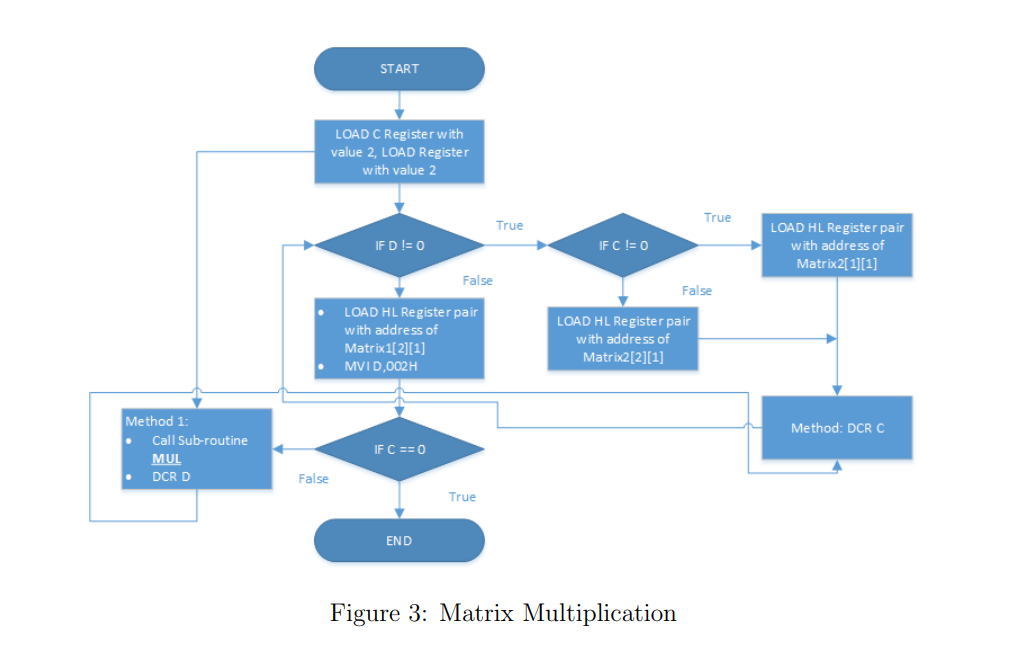
MVI D,002H

if C == 0: HLT

if C!= 0 : Call Method1

**FLOWCHART:**





**CODE:**

**Code for multiplication**

; code for multiplication of

; two numbers by repeated

; addition

; two numbers to be multiplied

; are stored in 0002H and

; 0003H,

; output is stored in 0004H

MOV B, 0002H ; Load the first number into B register

MOV C, 0003H ; Load the second number into C register

MVI A, 00H ; Initialize accumulator A to 0

LOOP:

ADD B ; Add the value in B to the accumulator A

DCR C ; Decrement the value in C

JNZ LOOP ; Jump back to LOOP if C is not zero

STA 0004H ; Store the result in memory location 0004H

**Multiplying row vector with column vector**

LXI H, 8500H ; Load the address of the row vector

PUSH 8508H ; Push the address of the column vector onto the stack

Method:

MOV M, A ; Move the value in A to the memory location pointed by H

XCHG ; Exchange H and D registers

MOV M, B ; Move the value in B to the memory location pointed by H

CALL MUL ; Call the MUL subroutine

STA 8516H ; Store the result in memory location 8516H

INX H ; Increment the address in H

XCHG ; Exchange H and D registers

INX H ; Increment the address in H

JMP Method ; Jump back to the Method

**Matrix Multiplication**

MVI C, 002H ; Initialize C with 2 (number of rows)

MVI D, 002H ; Initialize D with 2 (number of columns)

Method2:

DCR C

Method3: CALL MRC

DCR D

JNZ Method4

Method4:

ORI C, 00H

JNZ Method5

Method5:

LXI H, 8500H

JMP Method3

ORI C, 00H

JZ Method6

Method6:

LXI H, 8508H

JMP Method3

ORI D, 00H

JZ Method7

Method7:

INX H, 8508H

MVI D, 002H

ORI C, 00H

JNZ Method3

ORI C, 00H

JZ Method8

Method8:

HLT

**Final Code**

MVI C, 00 ; Initialize C to 0

LXI H, 8500 ; Load the address of the first matrix

LOOP2:

LXI D, 8600 ; Load the address of the second matrix

CALL MUL ; Call the MUL subroutine

MOV B, A ; Move the result to B

INX H ; Increment the address in H

INX D ; Increment the address in D

INX D

CALL MUL ; Call the MUL subroutine

ADD B ; Add the result to B

CALL STORE ; Call the STORE subroutine

DCX H ; Decrement the address in H

DCX D ; Decrement the address in D

CALL MUL ; Call the MUL subroutine

MOV B, A ; Move the result to B

INX H ; Increment the address in H

INX D ; Increment the address in D

INX D

ADD B ; Add the result to B

CALL STORE ; Call the STORE subroutine

MOV A, C ; Move the value of C to A

CPI 04 ; Compare A with 4

JZ LOOP1 ; If A is 4, jump to LOOP1

INX H ; Increment the address in H

JMP LOOP2 ; Jump back to LOOP2

LOOP1:

HLT ; Halt the processor

MUL: LDAX D ; Load the accumulator A with the value at the memory address pointed to by D

MOV D, A ; Move the value in A to register D

MOV H, M ; Move the value from the memory location pointed by H to register H

DCR H ; Decrement the value in H

JZ LOOP3 ; Jump to LOOP3 if H becomes zero

LOOP4: ADD D ; Add the value in D to A

DCR H ; Decrement H

JNZ LOOP4 ; Jump back to LOOP4 if H is not zero

LOOP3: MVI H,85 ; Load H with 85 (assuming it points to the first matrix)

MVI D,86 ; Load D with 86 (assuming it points to the second matrix)

RET ; Return from the subroutine

**CONCLUSIONS:**

At present, we have been successively in computing matrices of order 1 × n & n × 1 and 2 × 2 & 2 × 2.

**REFERENCES:**

1. Microprocessor Architecture, Programming, and Applications with the 8085 - S Gaonkar
2. 8080/8085 Assembly Language Programming Manual Copyright c 1977, 1978 Intel Corporation
3. http://en.wikipedia.org/wiki/Matrix multiplication