



EAST WEST UNIVERSITY

Implementation of 5x5 Matrix Multiplication using 8086 Assembly Language

CSE360 SEC 3

GROUP 02

Farhan Tanvir

ID: 2020-1-60-132

Roll: 33

Md. Arfan Ahmed

ID: 2020-1-60-139

Roll: 34

S.M. Ashiquzzaman

ID: 2017-2-60-073

Roll: 02

Submitted to,

Dr. Md. Nawab Yousuf Ali

Professor

Department of Computer Science and Engineering

East West University

Abstract - Matrix multiplication is a basic concept that is used in engineering applications such as digital image processing, digital signal processing and graph problem solving. Multiplication of huge matrices requires a lot of computation time as its complexity is $O(n^3)$. Because most engineering applications require higher computational throughputs with minimum time, many sequential and parallel algorithms are developed. The main goal of our project is the implementation of a 5x5 matrix multiplication using 8086 assembly language. To complement this project, we have used emu86 software. Two registers SI and DI have been used as address pointers of input matrices and BP register has been used to point the memory location of product matrix.

1. Introduction

Matrix multiplication is a binary operation that produces a matrix from two matrices. For matrix multiplication, the number of columns in the first matrix must be equal to the number of rows in the second matrix. The resulting matrix, known as the matrix product, has the number of rows of the first and the number of columns of the second matrix. The product of matrices A and B is denoted as AB. We have used 8086 assembly language to determine 5x5 matrix through matrix Multiplication in emu86 software. Here two registers are used as address pointers for input matrices and another register has used as address pointer for output matrix. To matrix multiplication firstly we have to store these matrices in the memory. While storing, first-row element stored in the memory first followed by second row and 3rd row elements. Here the two-dimensional array addressed by using pointers and counters. Where SI and DI registers are used as address pointers of two input matrix and CL and CH registers are used for row and column count. The BP register is used as a pointer for store the elements of output matrix.

2. Related Works

Matrix multiplication is among the most fundamental and compute-intensive operations in machine learning. Some particular examples include network theory, coordinate systems transformation, stochastic processes (like Markov chains and Markov jump processes), solution of linear systems of equations and population modeling. One of the areas of computer science in which matrix multiplication is particularly useful is graphics, since a digital image is basically a matrix to begin with: The rows and columns of the matrix correspond to rows and columns of pixels, and the numerical entries correspond to the pixels' color values. Decoding digital video, for instance, requires matrix multiplication; earlier this year, MIT researchers were able to build one of the first chips to implement the new high-efficiency video-coding standard for ultrahigh-definition TVs, in part because of patterns they discerned in the matrices it employs. In the same way that matrix multiplication can help process digital video, it can help process digital sound. A digital audio signal is basically a sequence of numbers, representing the variation over time of the air pressure of an acoustic audio signal. Many techniques for filtering or compressing digital audio signals, such as the Fourier transform, rely on matrix multiplication. Another reason that matrices are so useful in computer science is that graphs are. In this context, a graph is a mathematical construct consisting of nodes, usually depicted as circles, and edges, usually depicted as lines between them. Network diagrams and family trees are familiar examples of graphs, but in computer science they're used to represent everything from operations performed during the execution of a computer program to the relationship's characteristic of logistics problems.

3. Proposed Work

MOV SI, 1301H SI: First Matrix Pointer

MOV DI, 1401H DI: Second Matrix Pointer

MOV BP, 1501H BP: Final Matrix Pointer

MOV CL, 05H : CL = Row Count

MOV CH, 05H : CH = Column Count

MOV DH, CH : DH = Copy the value of CH

REPEAT3:

MOV BL, DH : Copy the column count in BL Register.

REPEAT2:

MOV DL, 00H : Initialized sum as zero.

MOV CH, DH : Get the column count in DH.

REPEAT1:

MOV AL, [SI] : Get one element of the Row in AL Register.

MUL [DI] : Get the product of row and column Element in AL.

ADD DL, AL : Add the product to SUM

INC SI : Increment the first matrix pointer

ADD DI, 05 : DI point the next Element of the Element in AL.

DEC CH : Decrement column count

JNZ REPEAT1 : Repeat Instructions until CH = 0

MOV [BP], DL : Store an Element of final matrix Element in AL.

INC BP : Increment the final matrix pointer

SUB SI, 05H : Make SI to point to first element of the Row again

SUB DI, 19H : Make DI to point first element of the column again.

INC DI : Increment the second matrix Pointer.

DEC BL : Decrement column count

JNZ REPEAT2 : Repeat instructions until BL= 0

ADD SI, 05H : Make SI point to first element of second Row of first matrix.

MOV DI, 1401H : Make DI point to the first element of second matrix.

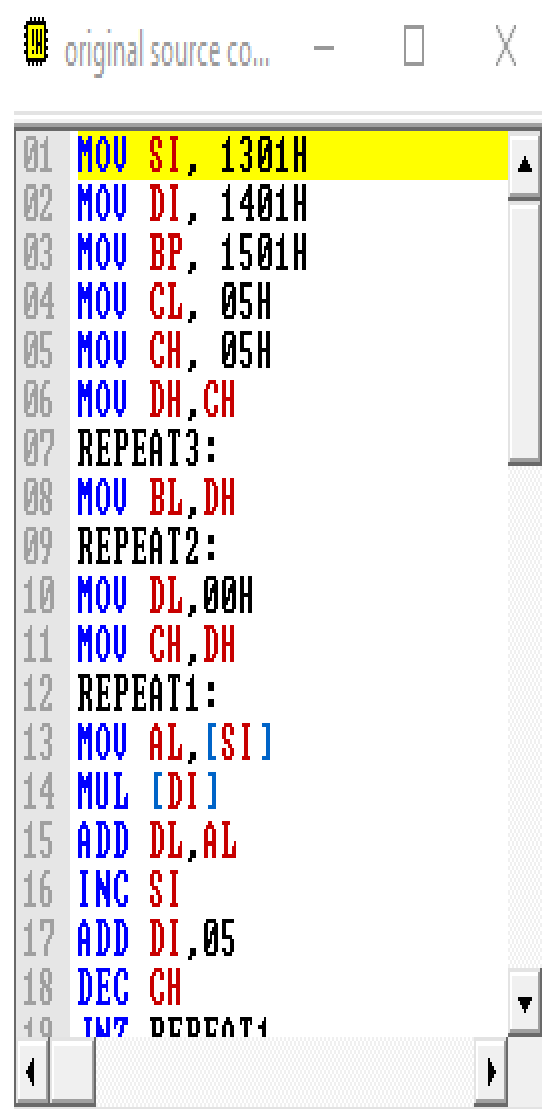
DEC CL : Decrement Row count

JNZ REPEAT3 : Repeat the instructions until CL= 0

HLT

4. Experimental result

First of all, we put the elements of first matrix on memory location 1301 to 1325.



```
01 MOV SI, 1301H
02 MOV DI, 1401H
03 MOV BP, 1501H
04 MOV CL, 05H
05 MOV CH, 05H
06 MOV DH, CH
07 REPEAT3:
08 MOV BL, DH
09 REPEAT2:
10 MOV DL, 00H
11 MOV CH, DH
12 REPEAT1:
13 MOV AL, [SI]
14 MUL [DI]
15 ADD DL, AL
16 INC SI
17 ADD DI, 05
18 DEC CH
19 JNZ REPEAT1
HLT
```

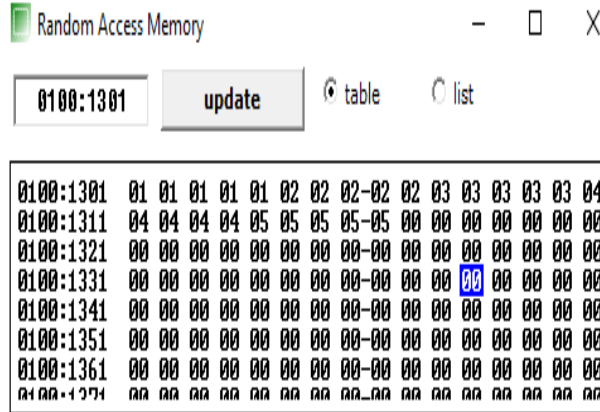


Figure 01: 1st Matrix Inputs

Secondly, we have put the elements of second matrix on memory location 1401 to 1425.

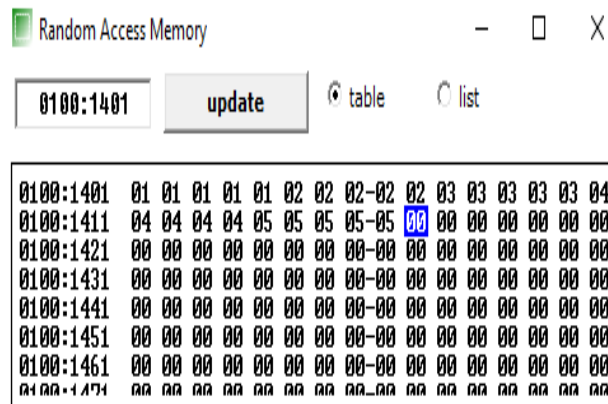


Figure 02: 2nd Matrix inputs

5. Conclusion

Matrix multiplication is repeatedly used in programs to represents a graphical data structure, which is used to store multiple vectors and also it is used in many applications like solving linear equations and more. Lots of research has been done on multiplying matrices using a minimum number of operations. To fulfill our goal of implementing matrix multiplication using 8086

After Updating the values in memory location, the output matrix has stored in memory location 1501H to 1525H.

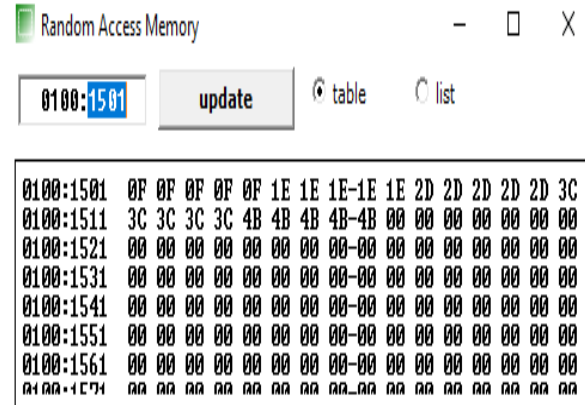
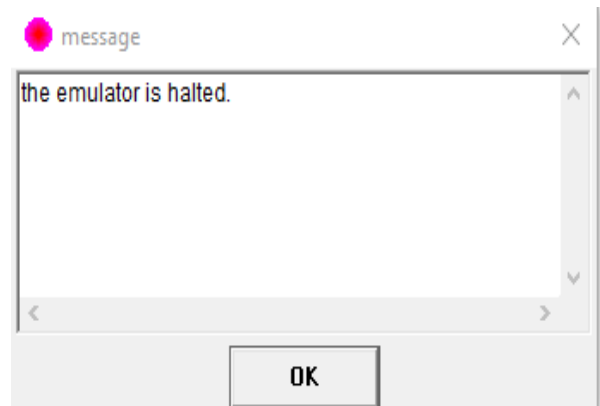


Figure 03: Output Matrix.



assembly language on emu86, we have used two-dimensional array addressed by using pointers and counters and few registers like SI, DI for Input matrices memory location, BP for product matrix's memory location and CH, CL, DH, BL for row count and column count.

References

1. Sagar, R. (2021, 9 14). *Developers corner. from Analyticsindiamag*: [Can We Speed Up Matrix Multiplication? \(analyticsindiamag.com\)](https://analyticsindiamag.com/can-we-speed-up-matrix-multiplication/)

2. Hardesty, L. (2013, 12 6). *From MIT NEWS: Explained: Matrices | MIT News | Massachusetts Institute of Technology*