# Parallel and distribute programming

Author: Altaf khan

## Simulation matrix 4*4 fallowing assembly code 3.11 for SIMD Machine (Using c#.Net)

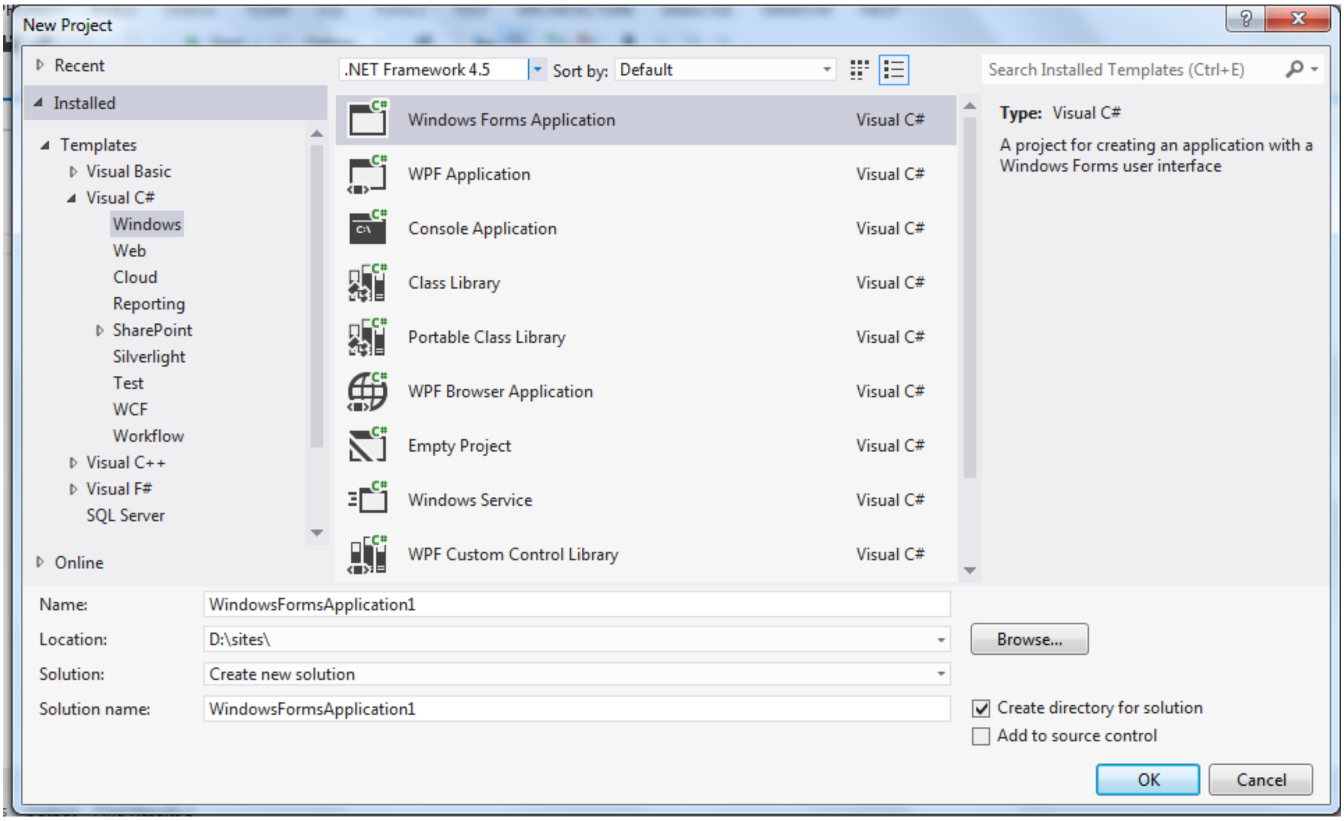
## Outline of Report

## 1. Introduction:

In this project simulation is implemented which shows that how our SIMD machine works. Matrix is a vector product when A matrix is multiply B and C is resulted matrix, so C = A*B I will show these implementation in form of simulation and I will use 4 processor which execute concurrently.

In this process. A is broad cast to all processors and B value will be fetched from B matrix each time and multiply by A value which is broadcasted to each PE. Each process execute alone and save same result in C matrix C[i,j] position.

For this Assignment I will prefer to visual studio 2012 (C#.net) tool. This is new powerful tool which is easily available in market. As compare to other C# is friendlier so I implement C# to perform above tasks. If you want to know how to make project, You can visit this link: http://msdn.microsoft.com/en-us/library/ms173077%28v=vs.90%29.aspx [2] . I will focus on my own work and try explains every point in this report. I just show the diagram in which user can easily understand how can we start new project?


New Project
Recent
Installed
Templates
Visual Basic
Visual C#
Windows
Web
Cloud
Reporting
SharePoint
Silverlight
Test
WCF
Workflow
Visual C++
Visual F#
SQL Server
Online
.NET Framework 4.5
Sort by: Default
Search Installed Templates (Ctrl+E)
Windows Forms Application Visual C#
WPF Application Visual C#
Console Application Visual C#
Class Library Visual C#
Portable Class Library Visual C#
WPF Browser Application Visual C#
Empty Project Visual C#
Windows Service Visual C#
WPF Custom Control Library Visual C#
Browse...
OK
Cancel

Type: Visual C#

A project for creating an application with a Windows Forms user interface

Name: WindowsFormsApplication1

Location: D:\sites\

Solution: Create new solution

Solution name: WindowsFormsApplication1

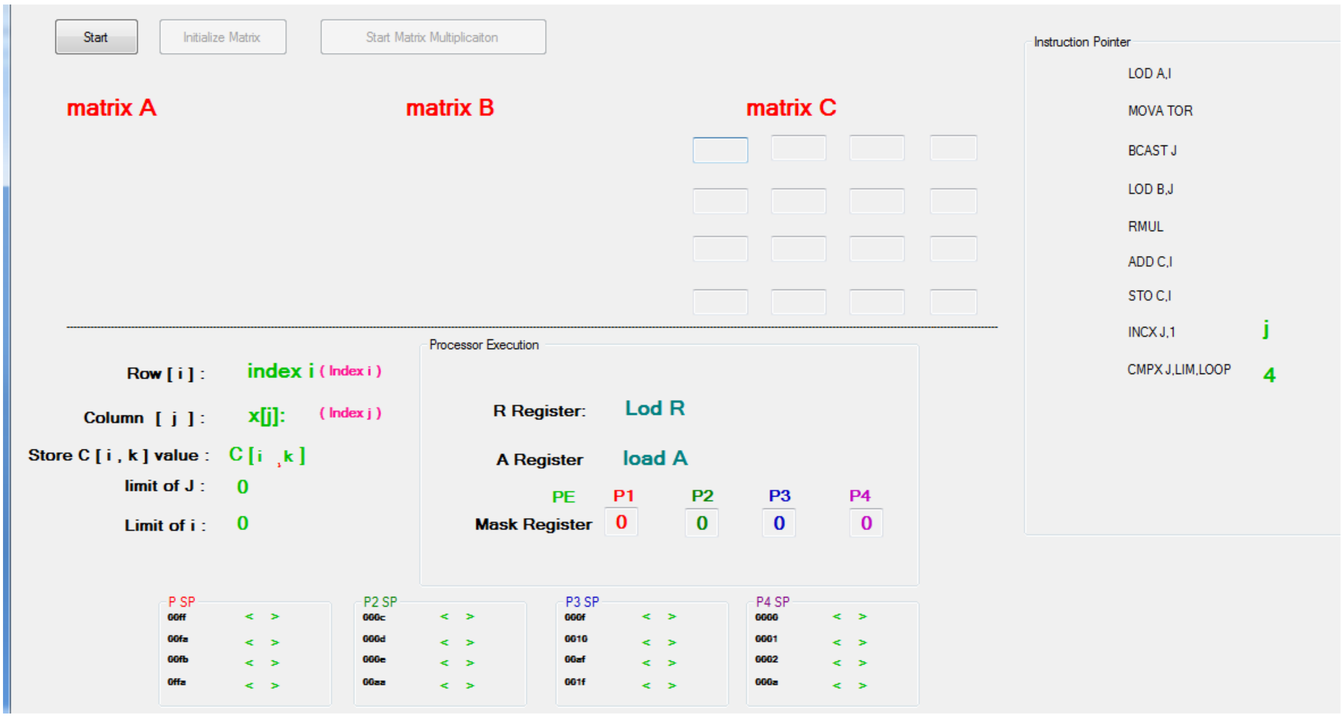
☑ Create directory for solution

□ Add to source control

**Figure 1.1** how to make new project of C# window form?

## 2. Implementation and Simulation procedure:

1. First of all create new project. Select c# and console App.
2. Add front end elements and perform matrix ‘A’ matrix ‘B’ and matrix ‘C’.
3. Create two register ‘A’ and ‘R’
    a. A is accumulator and R is routing register
4. Add processors Elements p1 to p4 which is involved to solve this solution.
5. Create I,j and limit object to handle instruction.
6. Write Assembly code on right side which is use for execution.
7. Create 3 button, Start, initialize matrix and start matrix on top of program.



```
LOD A,I
MOVA TOR
BCAST J
LOD B,J
RMUL
ADD C,I
STO C,I
INCX J,1          j
CMPX J,LIM,LOOP   4
```

Row [ i ] : index i ( Index i )

Column [ j ] : x[j]: ( Index j )

Store C [ i , k ] value : C [ i , k ]

limit of J : 0

Limit of i : 0

R Register: Lod R

A Register load A

| PE | P1 | P2 | P3 | P4 |
|---|---|---|---|---|
| Mask Register | 0 | 0 | 0 | 0 |

| P SP | P2 SP | P3 SP | P4 SP |
|---|---|---|---|
| 00ff | 000c | 000f | 0000 |
| 00fa | 000d | 0010 | 0001 |
| 00fb | 000e | 00af | 0002 |
| 0ffa | 00aa | 001f | 000a |

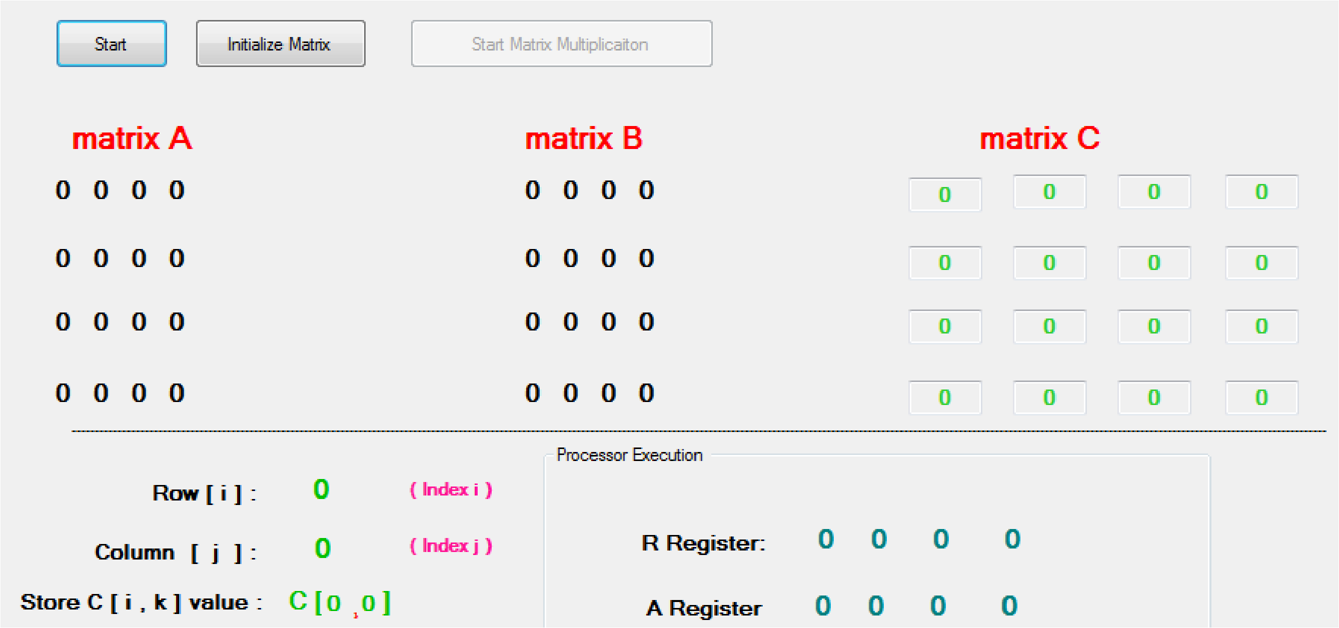
Figure 1.2 inter face of SIMD simulator for 4*4 matrix

8. Start buttion start the process. Means it initialize all value to zero.

It work fallowing these assembly instructions :

```
CLOAD   ZERO     Initialize all
 CBCAST          PE accumulators
MOVR TOA         to zero
STO     C,I      Zero the I-th row of C.
```

Start button work on above instruction. C load zero first then broadcast then move R to A. Hence shown in figure. Below



matrix A

0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

matrix B

0 0 0 0
0 0 0 0
0 0 0 0
0 0 0 0

matrix C

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Row [ i ] : 0 ( Index i )

Column [ j ] : 0 ( Index j )

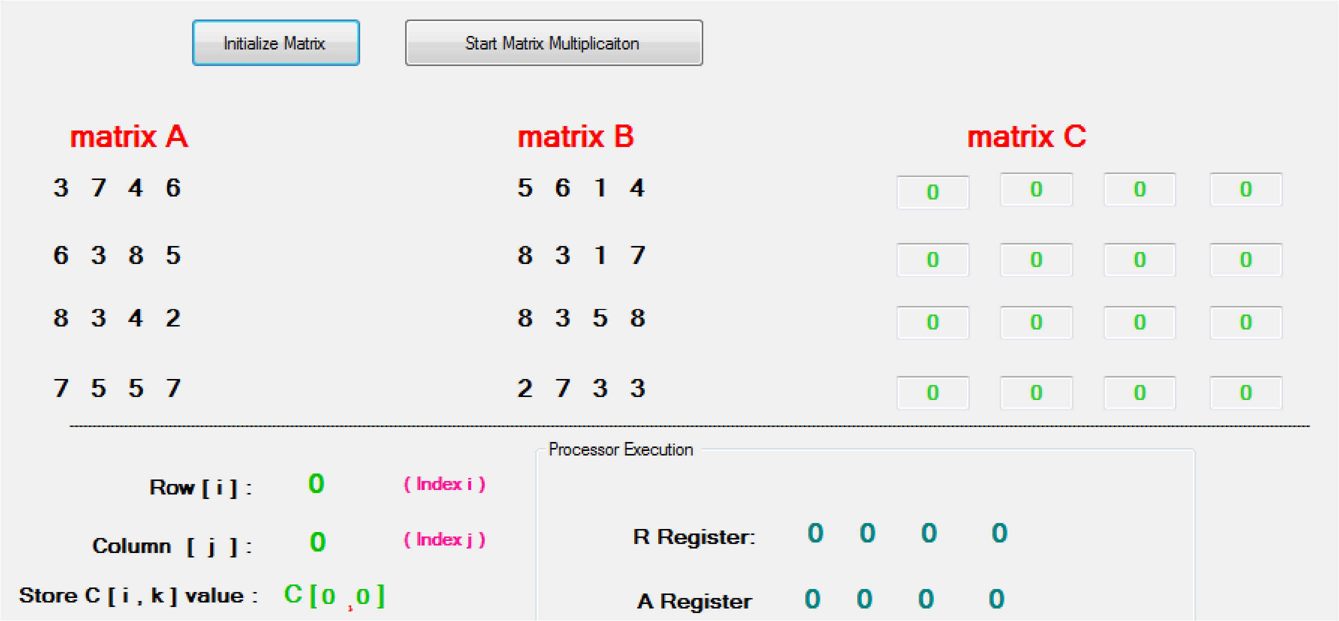
Store C [ i , k ] value : C [ 0 , 0 ]

R Register: 0 0 0 0

A Register 0 0 0 0

Figure1.3 initialize zero value to all matrix. R register is copy to A so both shows zero values.

9. Press the Initialize matrix button and assign value to matrix A and matrix B . like load matrix .



matrix A

3 7 4 6
6 3 8 5
8 3 4 2
7 5 5 7

matrix B

5 6 1 4
8 3 1 7
8 3 5 8
2 7 3 3

matrix C

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Row [ i ] : 0 ( Index i )

Column [ j ] : 0 ( Index j )

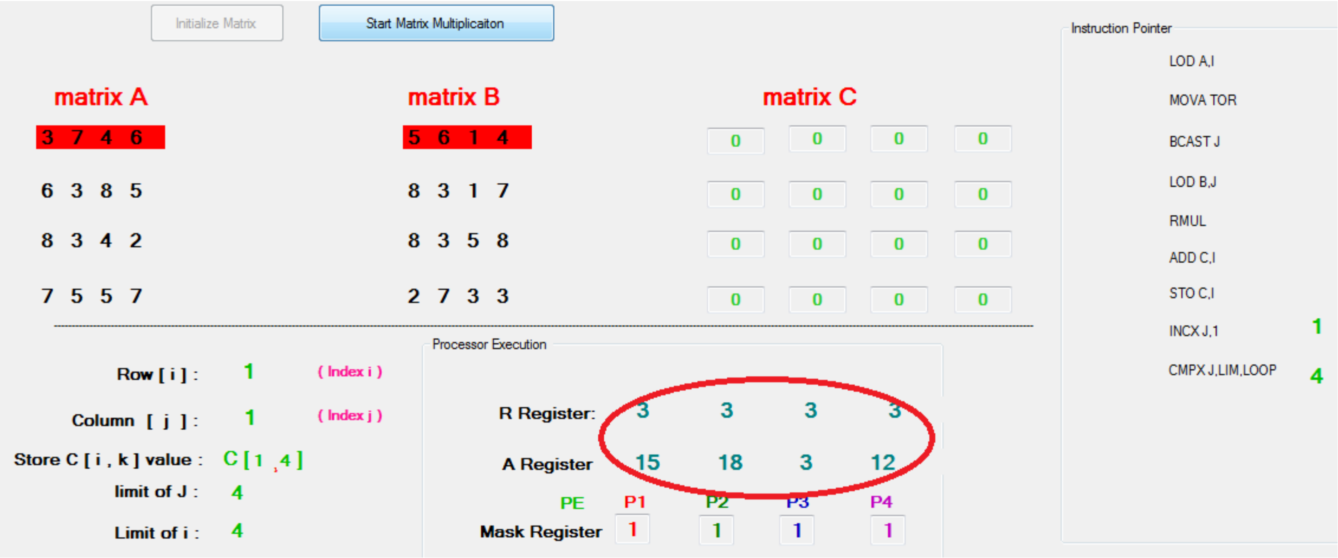
Store C [ i , k ] value : C [ 0 , 0 ]

R Register: 0 0 0 0

A Register 0 0 0 0

Figure 1.4 initialize value to matrix so now program wait for next instruction to lod value in A register.

10. Lod a value to AK register and next to also broad cast to each value every time to broad cast to R register .



matrix A

| | | | |
|---|---|---|---|
| 3 | 7 | 4 | 6 |
| 6 | 3 | 8 | 5 |
| 8 | 3 | 4 | 2 |
| 7 | 5 | 5 | 7 |

matrix B

| | | | |
|---|---|---|---|
| 5 | 6 | 1 | 4 |
| 8 | 3 | 1 | 7 |
| 8 | 3 | 5 | 8 |
| 2 | 7 | 3 | 3 |

matrix C

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

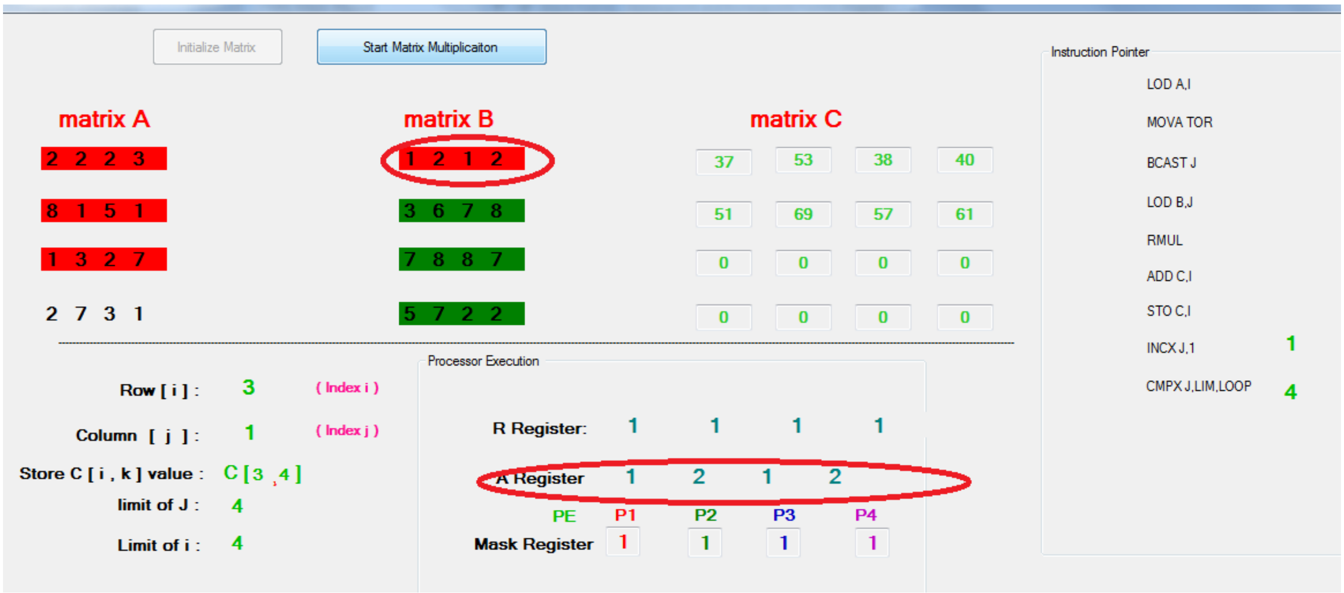
Row [ i ] : 1 ( Index i )
Column [ j ] : 1 ( Index j )
Store C [ i , k ] value : C [ 1 , 4 ]
limit of J : 4
Limit of i : 4

| | | | | |
|---|---|---|---|---|
| R Register: | 3 | 3 | 3 | 3 |
| A Register | 15 | 18 | 3 | 12 |
| PE | P1 | P2 | P3 | P4 |
| Mask Register | 1 | 1 | 1 | 1 |

LOD A,I
MOVA TOR
BCAST J
LOD B,J
RMUL
ADD C,I
STO C,I
INCX J,1 1
CMPX J,LIM,LOOP 4

Figure 1.5 lod to A[k] and add to again Ak and 3 is first value which is broadcast to each PE. (Route Register).

11. Lod b and add with A and then store to C . and each time J is increased up to 4.



matrix A

| | | | |
|---|---|---|---|
| 2 | 2 | 2 | 3 |
| 8 | 1 | 5 | 1 |
| 1 | 3 | 2 | 7 |
| 2 | 7 | 3 | 1 |

matrix B

| | | | |
|---|---|---|---|
| 1 | 2 | 1 | 2 |
| 3 | 6 | 7 | 8 |
| 7 | 8 | 8 | 7 |
| 5 | 7 | 2 | 2 |

matrix C

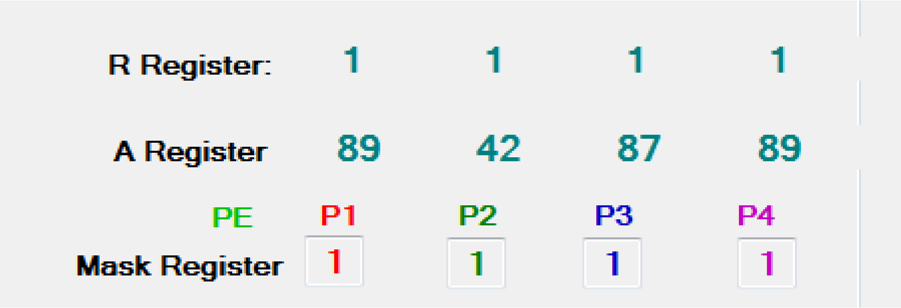
| | | | |
|---|---|---|---|
| 37 | 53 | 38 | 40 |
| 51 | 69 | 57 | 61 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Row [ i ] : 3 ( Index i )
Column [ j ] : 1 ( Index j )
Store C [ i , k ] value : C [ 3 , 4 ]
limit of J : 4
Limit of i : 4

| | | | | |
|---|---|---|---|---|
| R Register: | 1 | 1 | 1 | 1 |
| A Register | 1 | 2 | 1 | 2 |
| PE | P1 | P2 | P3 | P4 |
| Mask Register | 1 | 1 | 1 | 1 |

LOD A,I
MOVA TOR
BCAST J
LOD B,J
RMUL
ADD C,I
STO C,I
INCX J,1 1
CMPX J,LIM,LOOP 4

Figure 1.6 B lod to A[k] and then A add next calculation Rmul and result store to Ak.



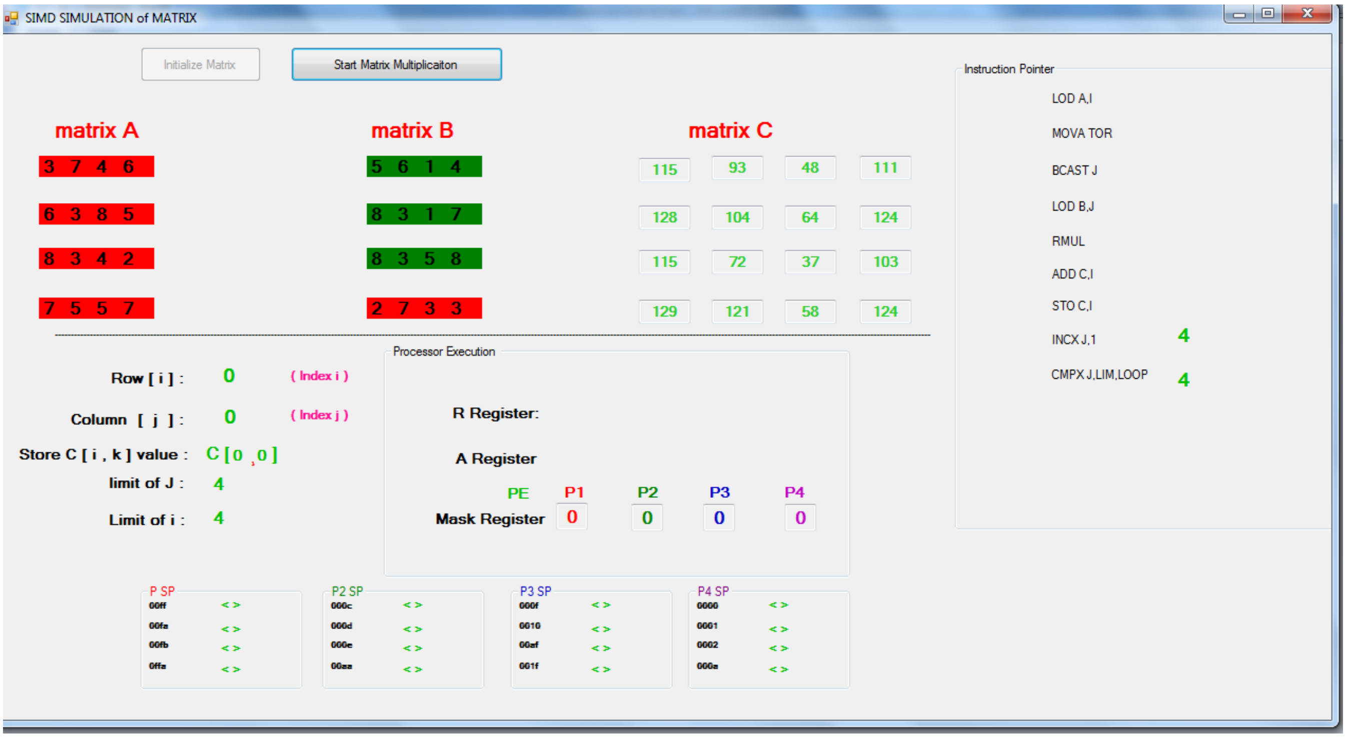
R Register: 1 1 1 1

A Register 89 42 87 89

| PE | P1 | P2 | P3 | P4 |
|---|---|---|---|---|
| Mask Register | 1 | 1 | 1 | 1 |

Figure 1.7 shows result multiply with B and store again to A and then it store to C. shown diagram 1.8.

12. This work repeatedly until i= 4.



matrix A

| | | | |
|---|---|---|---|
| 3 | 7 | 4 | 6 |
| 6 | 3 | 8 | 5 |
| 8 | 3 | 4 | 2 |
| 7 | 5 | 5 | 7 |

matrix B

| | | | |
|---|---|---|---|
| 5 | 6 | 1 | 4 |
| 8 | 3 | 1 | 7 |
| 8 | 3 | 5 | 8 |
| 2 | 7 | 3 | 3 |

matrix C

| | | | |
|---|---|---|---|
| 115 | 93 | 48 | 111 |
| 128 | 104 | 64 | 124 |
| 115 | 72 | 37 | 103 |
| 129 | 121 | 58 | 124 |

Row [ i ] : 0 ( Index i )

Column [ j ] : 0 ( Index j )

Store C [ i , k ] value : C [ 0 , 0 ]

limit of J : 4

Limit of i : 4

Processor Execution

R Register:

A Register

| PE | P1 | P2 | P3 | P4 |
|---|---|---|---|---|
| Mask Register | 0 | 0 | 0 | 0 |

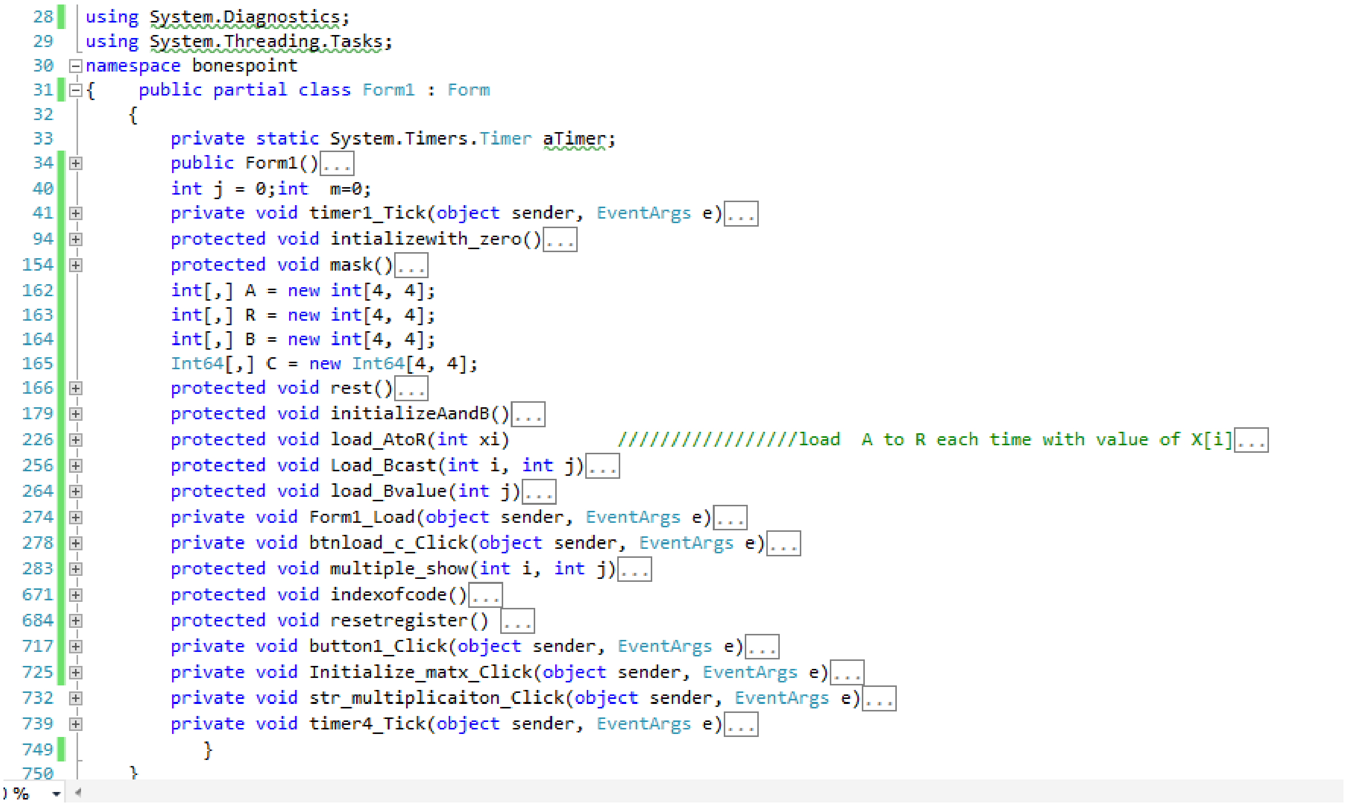
| P SP | | P2 SP | | P3 SP | | P4 SP | |
|---|---|---|---|---|---|---|---|
| 00ff | < > | 000c | < > | 000f | < > | 0000 | < > |
| 00fa | < > | 000d | < > | 0010 | < > | 0001 | < > |
| 00fb | < > | 000e | < > | 00af | < > | 0002 | < > |
| 0ffa | < > | 00aa | < > | 001f | < > | 000a | < > |

Instruction Pointer

LOD A,I
MOVA TOR
BCAST J
LOD B,J
RMUL
ADD C,I
STO C,I
INCX J,1 4
CMPX J,LIM,LOOP 4

Figure 1.8 shows execution of full matrix result 4*4. End of result all result store in C matrix and all variable become.

13. Cs code file is here. First of all I shows the function list.



```
using System.Diagnostics;
using System.Threading.Tasks;
namespace bonespoint
{     public partial class Form1 : Form
    {
        private static System.Timers.Timer aTimer;
        public Form1()...
        int j = 0;int  m=0;
        private void timer1_Tick(object sender, EventArgs e)...
        protected void intializewith_zero()...
        protected void mask()...
        int[,] A = new int[4, 4];
        int[,] R = new int[4, 4];
        int[,] B = new int[4, 4];
        Int64[,] C = new Int64[4, 4];
        protected void rest()...
        protected void initializeAandB()...
        protected void load_AtoR(int xi)          ////////////////load  A to R each time with value of X[i]...
        protected void Load_Bcast(int i, int j)...
        protected void load_Bvalue(int j)...
        private void Form1_Load(object sender, EventArgs e)...
        private void btnload_c_Click(object sender, EventArgs e)...
        protected void multiple_show(int i, int j)...
        protected void indexofcode()...
        protected void resetregister() ...
        private void button1_Click(object sender, EventArgs e)...
        private void Initialize_matx_Click(object sender, EventArgs e)...
        private void str_multiplicaiton_Click(object sender, EventArgs e)...
        private void timer4_Tick(object sender, EventArgs e)...
            }
    }
```

Figure 1.9 illustrate line of code which is use behind the above program.

14. First of all timer1_tick() function is use for represent slow the program .
**15.** Declare integer matrix A .B  size 4 * 4 but C is int64 but because C is resultant matrix it may be use for  storing large value  **.**
**16. Initializewith _zero():** this function is used for initialize zero to matrix  same work for broadcast zero to all PE and store also into the C matrix.
17. **Mask** is use for processor enable and disable .
18.  Reset is simple for initialize again to all with zero . Simple reset function.
19. InitializeAandB() , this function use for Initialize value to  A and B matrix.
20. LodAtoR() function simple load A value to R register  during execution,
21. loadBcast() is use for brocast j value to R register , each value of A is broadcasted by this function.
22. LoadBvalue() this function is specially for load B value into Ak register.
23. multipleMatrixShow() , this function properties is that it has done multiplication with B and store to Ak to C matrix .
24. index ofcode() function illustrate to show arrows and index value which is incremented during simulation.
25. Last 4 are button which use to call above all program in sequence. _click() function are button which work on click events. All code is written on project CD.

In below memory location is shown in which data store , load A and then B and both multiply with each other**. Mask register** show when program execute all processor s[k] bit become one and all are executed when execution is finish every all PE mask disable .

Row and column is checked with I to 4 and j is also start 1 to 4. Limit is 4 for all iterations. This assembly code which is implemented here.

## 3. Assembly code :

```
 CLOAD      ZERO                     Initialize all
 CBCAST                              PE accumulators
 MOVR      TOA                       to zero
 STO       C,I                       Zero the I-th row of C
 LDXI      J,0                       Initialize the column index
 LDX       LIM,N                     Loop limit is the matrix size
 LOOP:     LOD          A,I          Fetch the row I of A
 MOVA      TOR                       and set up for routing
 BCAST     J                         Broadcast A[I,J] to all PEs
 LOD B,    J                        Get row J of B and perform
RMUL                                 A[I,J]xB[J,k] for all k
ADD        C,I                       C[I,k]←C[I,k]+A[I,J]xB[J,k]
 STO       C,I                       for all k
INCX       J,1                        Increment column of A
CMPX     J,LIM,LOOP                   Loop if row of C not complete
```

Program 3-11. Matrix multiply SIMD assembly code for one row of the product matrix " [1]

## 4. Reference:

1. Book :fundamental of Parallel and distributed programming.(edition: 2003)
2. http://msdn.microsoft.com. (date : 5/20/2014)
3. http://stackoverflow.com/.