Project 2 Report

Tomasulo Algorithm Simulation

CSCE 3301

Computer Architecture
Fall 2022

Farah Kabesh 900191706
Karim Sherif 900191474
Omar Fayed 900191831

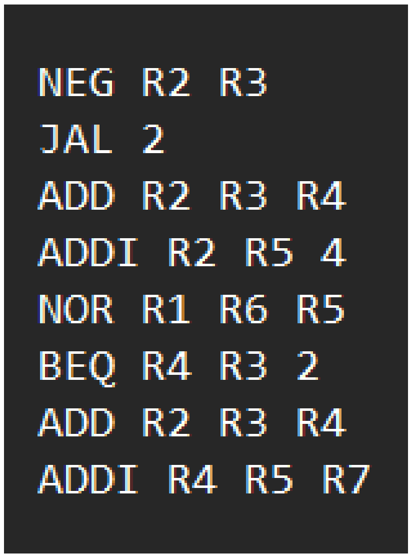
Table of Contents:

Description of Implementation:

First we implemented a function to parse an input file if the user chooses to take the instructions from an input file. Alternatively, in the main function we leave a second option for the user to input the instructions one after another in the terminal. When the user chooses how he wants to input the instructions the rest of the function and algorithms are called. The algorithm depends on three main functions that run the code. Issuing the instruction then executing the instruction and finally writing the result. These functions are called in the main function in a do while loop to run the program.

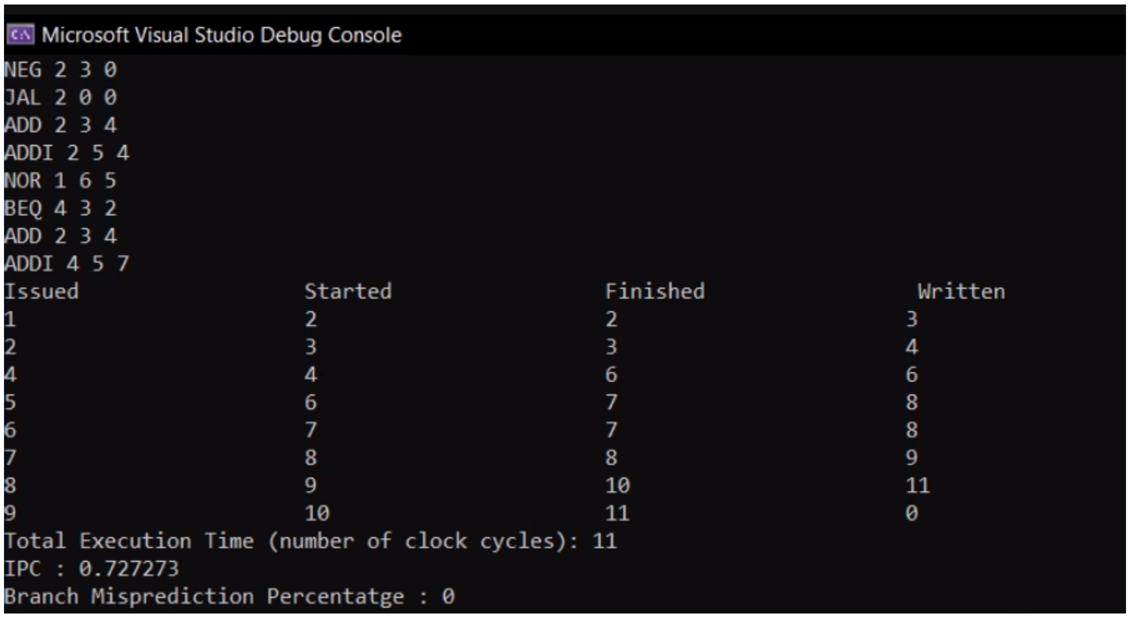
Results Obtained From Simulation of Each Assembly Program:

Program:



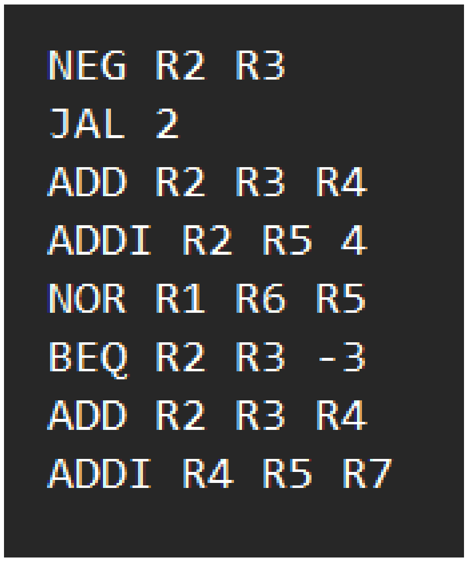
```
NEG R2 R3
JAL 2
ADD R2 R3 R4
ADDI R2 R5 4
NOR R1 R6 R5
BEQ R4 R3 2
ADD R2 R3 R4
ADDI R4 R5 R7
```

Simulation Result:



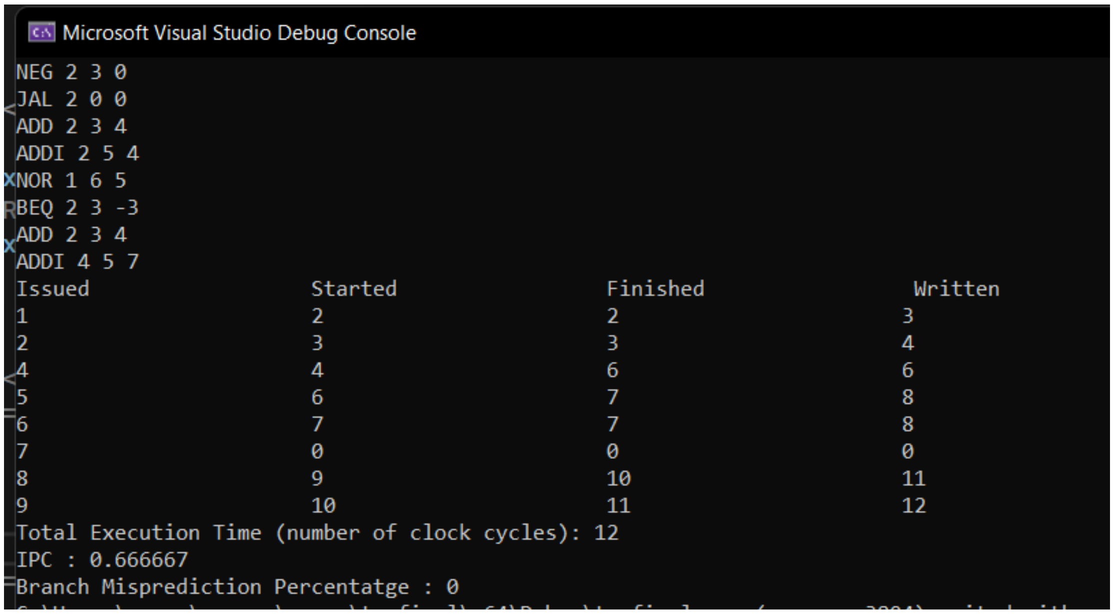
```
NEG 2 3 0
JAL 2 0 0
ADD 2 3 4
ADDI 2 5 4
NOR 1 6 5
BEQ 4 3 2
ADD 2 3 4
ADDI 4 5 7
Issued                  Started                 Finished                 Written
1                       2                       2                       3
2                       3                       3                       4
4                       4                       6                       6
5                       6                       7                       8
6                       7                       7                       8
7                       8                       8                       9
8                       9                       10                      11
9                       10                      11                      0
Total Execution Time (number of clock cycles): 11
IPC : 0.727273
Branch Misprediction Percentatge : 0
```

Program:



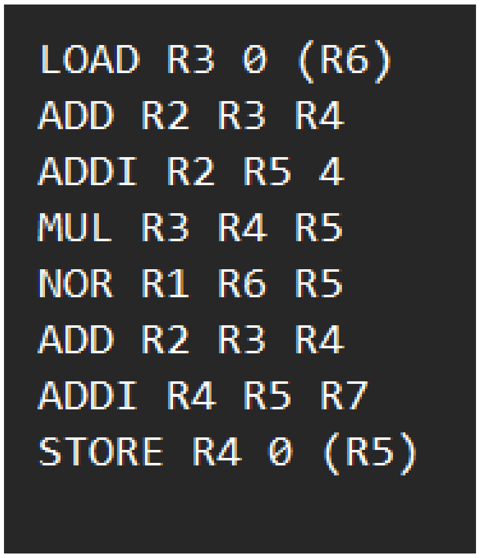
```
NEG R2 R3
JAL 2
ADD R2 R3 R4
ADDI R2 R5 4
NOR R1 R6 R5
BEQ R2 R3 -3
ADD R2 R3 R4
ADDI R4 R5 R7
```

Simulation Result:



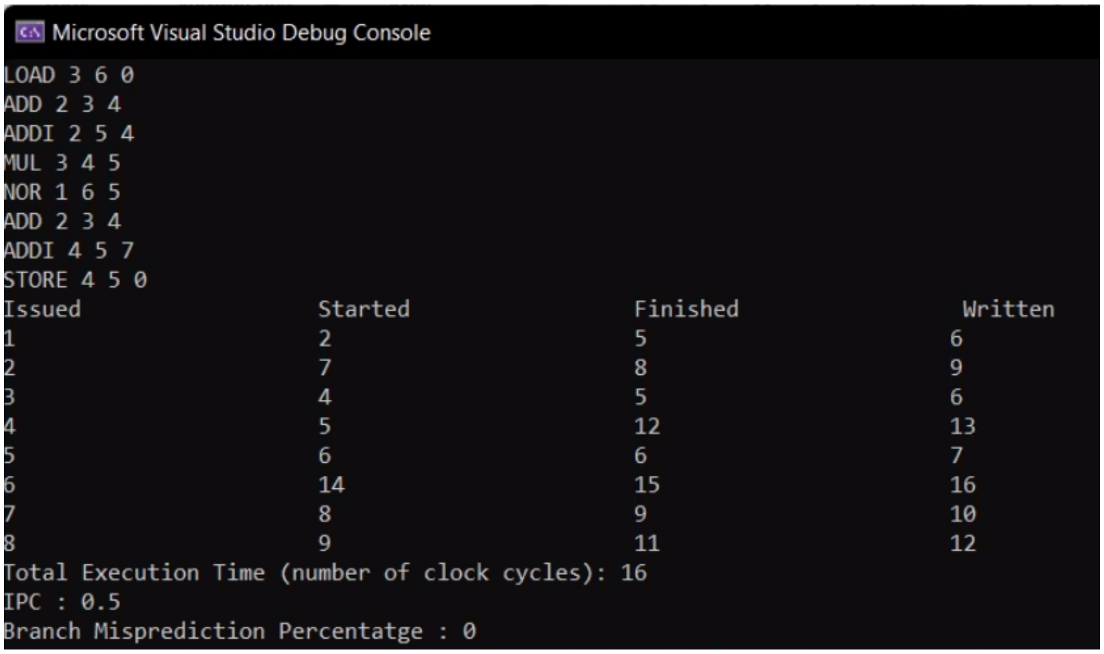
```
NEG 2 3 0
JAL 2 0 0
ADD 2 3 4
ADDI 2 5 4
NOR 1 6 5
BEQ 2 3 -3
ADD 2 3 4
ADDI 4 5 7
Issued                  Started                 Finished                 Written
1                       2                       2                       3
2                       3                       3                       4
4                       4                       6                       6
5                       6                       7                       8
6                       7                       7                       8
7                       0                       0                       0
8                       9                       10                      11
9                       10                      11                      12
Total Execution Time (number of clock cycles): 12
IPC : 0.666667
Branch Misprediction Percentatge : 0
```

Program:



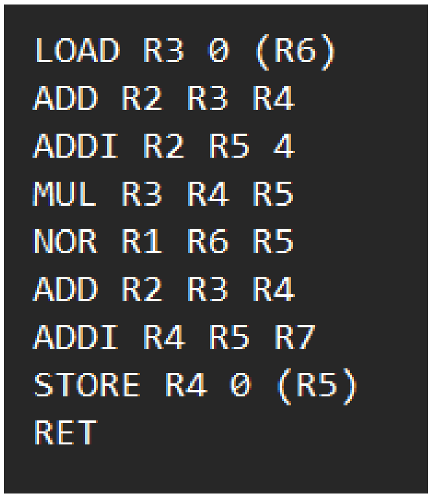
```
LOAD R3 0 (R6)
ADD R2 R3 R4
ADDI R2 R5 4
MUL R3 R4 R5
NOR R1 R6 R5
ADD R2 R3 R4
ADDI R4 R5 R7
STORE R4 0 (R5)
```

Simulation Result:



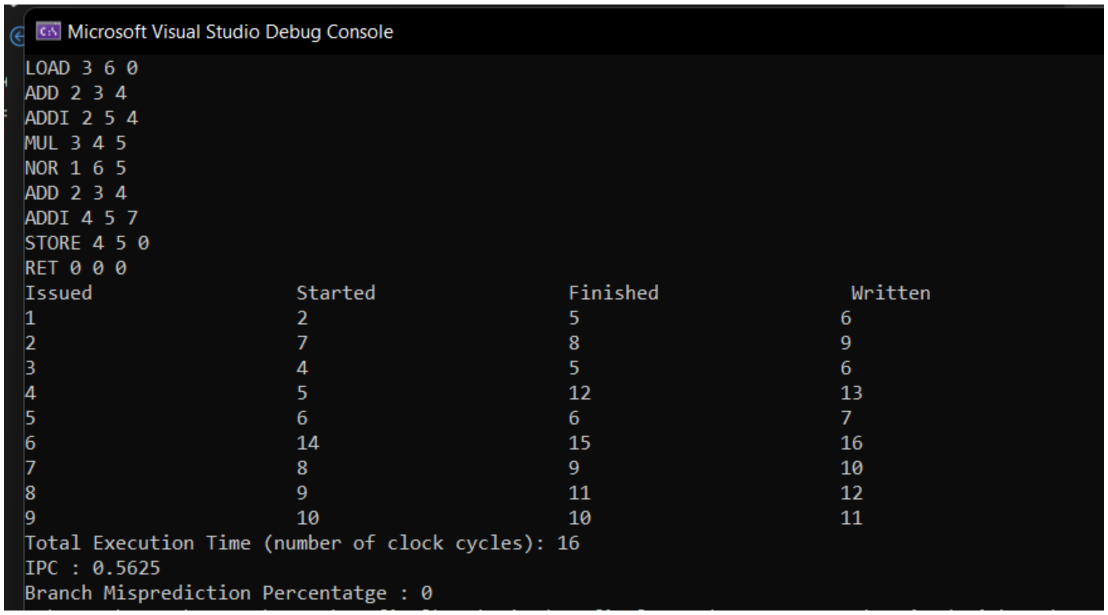
```
Microsoft Visual Studio Debug Console
LOAD 3 6 0
ADD 2 3 4
ADDI 2 5 4
MUL 3 4 5
NOR 1 6 5
ADD 2 3 4
ADDI 4 5 7
STORE 4 5 0
Issued                  Started                 Finished                 Written
1                       2                       5                        6
2                       7                       8                        9
3                       4                       5                        6
4                       5                       12                       13
5                       6                       6                        7
6                       14                      15                       16
7                       8                       9                        10
8                       9                       11                       12
Total Execution Time (number of clock cycles): 16
IPC : 0.5
Branch Misprediction Percentatge : 0
```

Program:



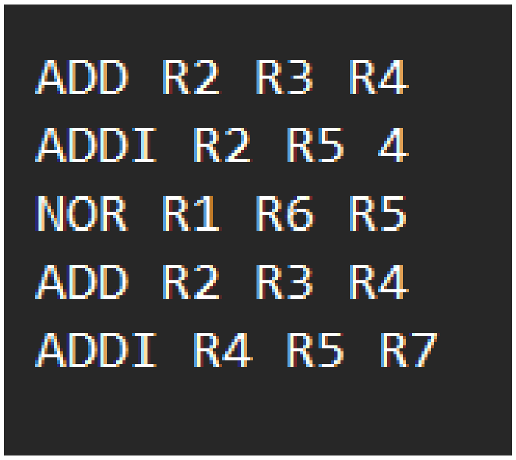
```
LOAD R3 0 (R6)
ADD R2 R3 R4
ADDI R2 R5 4
MUL R3 R4 R5
NOR R1 R6 R5
ADD R2 R3 R4
ADDI R4 R5 R7
STORE R4 0 (R5)
RET
```

Simulation Result:



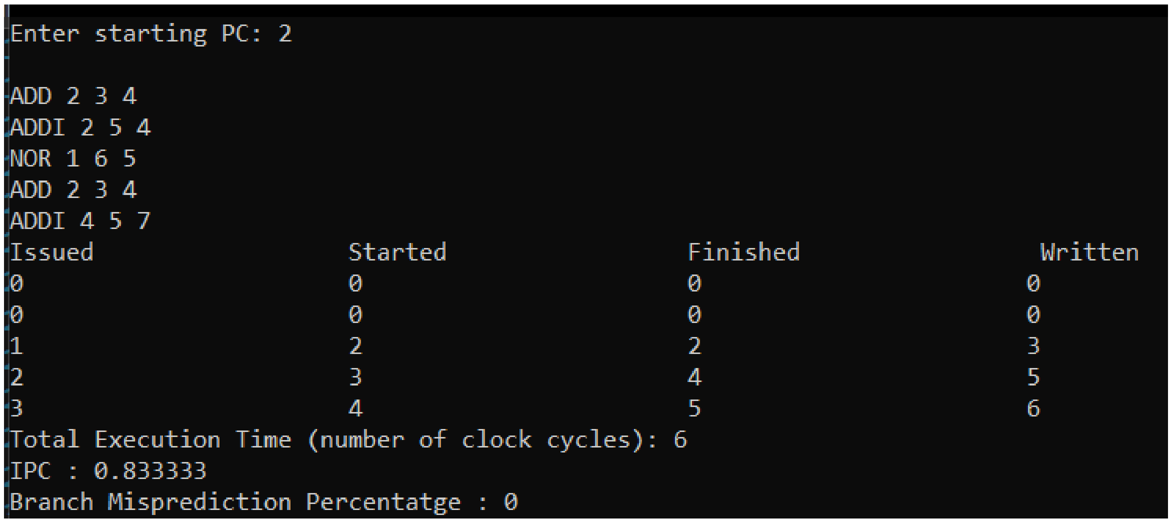
```
LOAD 3 6 0
ADD 2 3 4
ADDI 2 5 4
MUL 3 4 5
NOR 1 6 5
ADD 2 3 4
ADDI 4 5 7
STORE 4 5 0
RET 0 0 0
Issued                  Started                 Finished                 Written
1                       2                       5                        6
2                       7                       8                        9
3                       4                       5                        6
4                       5                       12                       13
5                       6                       6                        7
6                       14                      15                       16
7                       8                       9                        10
8                       9                       11                       12
9                       10                      10                       11
Total Execution Time (number of clock cycles): 16
IPC : 0.5625
Branch Misprediction Percentatge : 0
```

Program:



```
ADD R2 R3 R4
ADDI R2 R5 4
NOR R1 R6 R5
ADD R2 R3 R4
ADDI R4 R5 R7
```

Simulation Result (user specifies its starting address):



```
Enter starting PC: 2

ADD 2 3 4
ADDI 2 5 4
NOR 1 6 5
ADD 2 3 4
ADDI 4 5 7
Issued                  Started                 Finished                 Written
0                       0                       0                       0
0                       0                       0                       0
1                       2                       2                       3
2                       3                       4                       5
3                       4                       5                       6
Total Execution Time (number of clock cycles): 6
IPC : 0.833333
Branch Misprediction Percentatge : 0
```

Observations on Results:

By looking at the simulation results of the tested assembly programs above, we can clearly see that Tomasulo's algorithm provides in-order issuing, out-of-order execution (since execution start and execution end are both out of order) as well as out-of-order completion.