ECE 2162 Tomasulo Algorithm

Aditya Pawar†, Manisha Mondal†, and Vivswan Shah†

*†These authors contributed equally to this work, *corresponding email: adp110@pitt.edu, manisha.mondal@pitt.edu, and vivswanshah@pitt.edu*

System Requirements:

- A system with installed Python 3.7 or higher.

Instructions to run the code:

1. Make sure you meet the system requirements.
2. Add the test code to the "$unit\_tests$" folder in a text document with a name of "$test\_\{index\}.txt$" (where index is a whole number).
3. In the "$main.py$" in the project root directory, make sure to add you $index$ number to the list at the main section at the bottom of the python file (shown in Figure 1), so that it is executed by passing to "run_test_code" function in the same file.

```
if __name__ == '__main__':
    for i in [1, 2, 3, 4, 5, 6]:
        run_test_code(i)
```

Figure 1. Code from "main.py"

4. Run "$main.py$" in the project root directory using Python 3.7 or higher.
5. The result of the executable will be display in the console and in the "$result\_\{index\}.txt$" in the "$result$" folder, given there is no error with the test code (like division by zero, accessing memory address not divisible by 4, wrong argument for any instruction etc.)

Test Code Layout:

- Sample test codes can be found in "$unit\_codes$" folder.
- *#*: create a comment in code
- *$*: `$ {name} = {value}`
  It is used to assign the value to integer register (R*), float register (F*), memory (MEM[*]), and the parameters of the program (these parameters are list in "$parameter.txt$" with their default values)
- *!*: `! {name} = {value}`
  It is used to check the value of the integer register (R*), float register (F*), memory (MEM[*]), and total number of cycles after the execution of the code.

Note: All '#', '$', and '!' must be at the beginning of the line. All '$' and '!' must be before all the instruction code.

Team Members:

- *Aditya Pawar*: Implementation of Tomasulo algorithm, Branch Prediction and Speculation recovery.
- *Manisha Modal*: Implementation of Tomasulo algorithm and Test Benching.
- *Vivswan Shah*: Implementation of Tomasulo algorithm, Memory load/store Forwarding and Speculation recovery.

```
Running Test Case 1...
```

Code

```
# Straight-line base cases where no dependencies exist among instructions

$ Mem[4] = 2.3

$ R1 = 4
$ R2 = 20
$ R3 = 12
$ R4 = 9
$ R5 = 13
$ R6 = 8
$ R7 = 5
$ R8 = 9

$ F2 = 7.3
$ F3 = 1.8
$ F4 = 3.5
$ F5 = 4.6
$ F6 = 4.5
$ F7 = 1.5
$ F8 = 2.5

! F1 = 2.3
! MEM[20] = 7.3
! R9 = 21
! R10 = 5
! R11 = 3
! R12 = 1
! F9 = 5.3
! F10 = 0.1
! F11 = 3.75

# Load/Store instructions
LD F1, 0 (R1)
SD F2, 0 (R2)

# Integer instructions
ADD R9, R3, R4
ADDI R10, R5, -8
SUB R11, R6, R7
SUBI R12, R8, 8

# Floating Addition/Subtraction Point instructions
ADDD F9, F3, F4
SUBD F10, F5, F6

# Floating Multiplication instructions
MULTD F11, F7, F8
```

Execution

Counter	Index	Instruction	Issue	Execute	Memory	Write Back	Commit	Result	Extra Info	Operands
0	0	LD F1, 0 (R1)	1	2	3-6	7	8	2.300	M: 4	['ROB1', 0, 4]
1	1	SD F2, 0 (R2)	2	3	9-12		9-12	NULL	M: 20	[7.3, 0, 20]
2	2	ADD R9, R3, R4	3	4		5	10	21		['ROB2', 12, 9]
3	3	ADDI R10, R5, -8	4	5		6	11	5		['ROB3', 13, -8]
4	4	SUB R11, R6, R7	5	6		8	12	3		['ROB4', 8, 5]
5	5	SUBI R12, R8, 8	6	7		9	13	1		['ROB5', 9, 8]
6	6	ADDD F9, F3, F4	7	8-10		11	14	5.300		['ROB6', 1.8, 3.5]
7	7	SUBD F10, F5, F6	8	9-11		12	15	0.100		['ROB7', 4.6, 4.5]
8	8	MULTD F11, F7, F8	9	10-29		30	31	3.750		['ROB8', 1.5, 2.5]

Registers

```
Integer Register -  00: 0
01: 4     02: 20    03: 12    04: 9     05: 13    06: 8     07: 5     08: 9     09: 21    10: 5     11: 3     12: 1     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Float Register -    00: 0
01: 2.30  02: 7.30  03: 1.80  04: 3.50  05: 4.60  06: 4.50  07: 1.50  08: 2.50  09: 5.30  10: 0.10  11: 3.75  12: 0     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Memory -
0004: 2.30    0020: 7.30
```

Asserts

```
True  - F1: 2.3 == 2.3000
True  - MEM[20]: 7.3 == 7.3000
True  - R9: 21 == 21
True  - R10: 5 == 5
True  - R11: 3 == 3
True  - R12: 1 == 1
True  - F9: 5.3 == 5.3000
True  - F10: 0.1 == 0.1000
True  - F11: 3.75 == 3.7500
asserts: True
```

```
Running Test Case 2...
---------------------------------------------------------------------------- Code ----------------------------------------------------------------------------
# Straight-line code where there are dependencies (true and false)

$ R1 = 3
$ R2 = 4
$ R3 = 5
$ F1 = 1.5
$ F2 = 3.5
$ F3 = 4.0

! R2 = -3
! R4 = -1
! R5 = -2
! F4 = 5
! F5 = -2.5
! F6 = 5
! F7 = 1

# Read-After-Read   (RAR:   read dependence) on F1
ADDD F4, F1, F2
SUBD F5, F1, F3

# Write-After-Read  (WAR:   anti-dependence) on R2
SUB R4, R1, R2
ADDI R2, R3, -8

# Read-After-Write  (RAW:   true dependence) on F6
ADDD F6, F1, F2
SUBD F7, F6, F3

# Write-After-Write (WAW: output dependence) on R5
ADDI R5, R1, -8
SUB R5, R1, R3

------------------------------------------------------------------------ Execution -------------------------------------------------------------------------

  Counter     Index     Instruction           Issue     Execute     Memory     Write Back     Commit     Result     Extra Info     Operands
.....................................................................................................................................................
  0           0         ADDD F4, F1, F2       1         2-4                    5              6          5.000                     ['ROB1', 1.5, 3.5]
  1           1         SUBD F5, F1, F3       2         3-5                    7              8          -2.500                    ['ROB2', 1.5, 4.0]
  2           2         SUB R4, R1, R2        3         4                      6              9          -1                        ['ROB3', 3, 4]
  3           3         ADDI R2, R3, -8       4         5                      8              10         -3                        ['ROB4', 5, -8]
  4           4         ADDD F6, F1, F2       5         6-8                    9              11         5.000                     ['ROB5', 1.5, 3.5]
  5           5         SUBD F7, F6, F3       6         10-12                  13             14         1.000                     ['ROB6', 5.0, 4.0]
  6           6         ADDI R5, R1, -8       7         8                      10             15         -5                        ['ROB7', 3, -8]
  7           7         SUB R5, R1, R3        8         9                      11             16         -2                        ['ROB8', 3, 5]

------------------------------------------------------------------------ Registers -------------------------------------------------------------------------
Integer Register -  00: 0
01: 3     02: -3    03: 5     04: -1    05: -2    06: 0     07: 0     08: 0     09: 0     10: 0     11: 0     12: 0     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Float Register -    00: 0
01: 1.50  02: 3.50  03: 4.0   04: 5.0   05: -2.50 06: 5.0   07: 1.0   08: 0     09: 0     10: 0     11: 0     12: 0     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Memory -


-------------------------------------------------------------------------- Asserts --------------------------------------------------------------------------
True  - R2: -3 == -3
True  - R4: -1 == -1
True  - R5: -2 == -2
True  - F4: 5 == 5.0000
True  - F5: -2.5 == -2.5000
True  - F6: 5 == 5.0000
True  - F7: 1 == 1.0000
asserts: True
```

```
================================================================================================================================================================
Running Test Case 3...
------------------------------------------------------------------------------ Code ----------------------------------------------------------------------------
# Straight-line code where there are forwarding among load/store instructions

$ R1 = 4
$ R2 = 8
$ R3 = 12
$ F1 = 2.1
$ F2 = 10.0
$ MEM[4] = 1
$ MEM[8] = 2
$ MEM[12] = 3

! F3 = 2
! F4 = 3
! F5 = 21
! F6 = 2

MULTD F5, F1, F2

# Get value from memory
LD F3, 0(R2)
LD F6, 0(R2)
SD F5, 12(R1)
LD F4, 0(R3)

# Get the value of F6 in a single cycle due to the previous LD for F3.
LD F6, 0(R2)

---------------------------------------------------------------------------- Execution ---------------------------------------------------------------------------

  Counter     Index     Instruction           Issue     Execute     Memory     Write Back     Commit     Result     Extra Info     Operands
..................................................................................................................................................
  0           0         MULTD F5, F1, F2      1         2-21                   22             23         21.000                    ['ROB1', 2.1, 10.0]
  1           1         LD F3, 0(R2)          2         3           4-7        8              24         2          M: 8           ['ROB2', 0, 8]
  2           2         LD F6, 0(R2)          3         4           8          9              25         2          M: 8           ['ROB3', 0, 8]
  3           3         SD F5, 12(R1)         4         5           26-29                     26-29      NULL       M: 16          [21.0, 12, 4]
  4           4         LD F4, 0(R3)          5         6           9-12       13             27         3          M: 12          ['ROB4', 0, 12]
  5           5         LD F6, 0(R2)          6         7           13         14             28         2          M: 8           ['ROB5', 0, 8]

------------------------------------------------------------------------------ Registers -----------------------------------------------------------------------------
Integer Register -  00: 0
01: 4     02: 8     03: 12    04: 0     05: 0     06: 0     07: 0     08: 0     09: 0     10: 0     11: 0     12: 0     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Float Register -    00: 0
01: 2.10  02: 10.0  03: 2.0   04: 3.0   05: 21.0  06: 2.0   07: 0     08: 0     09: 0     10: 0     11: 0     12: 0     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Memory -
0004: 1       0008: 2       0012: 3       0016: 21.0

------------------------------------------------------------------------------ Asserts -----------------------------------------------------------------------------
True  - F3: 2 == 2.0000
True  - F4: 3 == 3.0000
True  - F5: 21 == 21.0000
True  - F6: 2 == 2.0000
asserts: True
================================================================================================================================================================

================================================================================================================================================================
Running Test Case 4...
------------------------------------------------------------------------------ Code ----------------------------------------------------------------------------
# Straight-line code where there are structure hazards in reservation stations and functional unit

# Modifying the number of Integer adder RS to show structure hazard resolution.
$ float_multiplier_rs = 2
$ integer_adder_rs = 2

$ R4 = 16
$ R5 = 4
$ R6 = 2
$ F4 = 3.4
$ F5 = 5.3
$ F6 = 2.8

! F13 = 18.02
! F14 = 9.52
! F15 = 14.84
! R10 = 20
! R11 = 14
! R12 = 6

MULTD F13, F4, F5
MULTD F14, F4, F6
MULTD F15, F5, F6

ADD R10, R4, R5
SUB R11, R4, R6
ADD R12, R5, R6

---------------------------------------------------------------------------- Execution ---------------------------------------------------------------------------

  Counter     Index     Instruction               Issue     Execute     Memory     Write Back     Commit     Result     Extra Info     Operands
..................................................................................................................................................
  0           0         MULTD F13, F4, F5         1         2-21                   22             23         18.020                    ['ROB1', 3.4, 5.3]
  1           1         MULTD F14, F4, F6         2         3-22                   23             24         9.520                     ['ROB2', 3.4, 2.8]
  2           2         MULTD F15, F5, F6         23        24-43                  44             45         14.840                    ['ROB3', 5.3, 2.8]
  3           3         ADD R10, R4, R5           24        25                     26             46         20                        ['ROB4', 16, 4]
  4           4         SUB R11, R4, R6           25        26                     27             47         14                        ['ROB5', 16, 2]
  5           5         ADD R12, R5, R6           27        28                     29             48         6                         ['ROB6', 4, 2]

------------------------------------------------------------------------------ Registers -----------------------------------------------------------------------------
Integer Register -  00: 0
01: 0     02: 0     03: 0     04: 16    05: 4     06: 2     07: 0     08: 0     09: 0     10: 20    11: 14    12: 6     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Float Register -    00: 0
01: 0     02: 0     03: 0     04: 3.40  05: 5.30  06: 2.80  07: 0     08: 0     09: 0     10: 0     11: 0     12: 0     13: 18.02 14: 9.52  15: 14.84 16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Memory -


------------------------------------------------------------------------------ Asserts -----------------------------------------------------------------------------
True  - F13: 18.02 == 18.0200
True  - F14: 9.52 == 9.5200
True  - F15: 14.84 == 14.8400
True  - R10: 20 == 20
True  - R11: 14 == 14
True  - R12: 6 == 6
asserts: True
================================================================================================================================================================
```

```
Running Test Case 5...
---------- Code ----------
# Straight line code with a loop added at the end for BNE

$ Mem[0] = 2.3

$ R3 = 12
$ R4 = 9
$ R20 = 10
$ R21 = 16
$ F7 = 1.5
$ F8 = 2.5

! R20 = 16
! F1 = 2.3
! R9 = 21
! F11 = 3.75

# code should loop 3 times.
Loop: ADDI R20, R20, 2
LD F1, 0 (R1)
ADD R9, R3, R4
MULTD F11, F7, F8
BNE R20, R21, Loop

---------- Execution ----------

  Counter     Index     Instruction                Issue     Execute     Memory     Write Back     Commit     Result     Extra Info     Operands
.....................................................................................................................................................
  0           0         ADDI R20, R20, 2           1         2                      3              4          12                        ['ROB1', 10, 2]
  1           1         LD F1, 0 (R1)              2         3           4-7        8              9          2.300      M: 0           ['ROB2', 0, 0]
  2           2         ADD R9, R3, R4             3         4                      5              10         21                        ['ROB3', 12, 9]
  3           3         MULTD F11, F7, F8          4         5-24                   25             26         3.750                     ['ROB4', 1.5, 2.5]
  4           4         BNE R20, R21, -5           5         6                                     27         0          P: False       [12, 16, -5]
  5           0         ADDI R20, R20, 2           8         9                      10             28         14                        ['ROB5', 12, 2]
  6           1         LD F1, 0 (R1)              9         10          11         12             29         2.300      M: 0           ['ROB6', 0, 0]
  7           2         ADD R9, R3, R4             10        11                     13             30         21                        ['ROB7', 12, 9]
  8           3         MULTD F11, F7, F8          11        12-31                  32             33         3.750                     ['ROB8', 1.5, 2.5]
  9           4         BNE R20, R21, -5           12        13                                    34         0          P: True        [14, 16, -5]
  10          0         ADDI R20, R20, 2           13        14                     15             35         16                        ['ROB9', 14, 2]
  11          1         LD F1, 0 (R1)              14        15          16         17             36         2.300      M: 0           ['ROB10', 0, 0]
  12          2         ADD R9, R3, R4             15        16                     18             37         21                        ['ROB11', 12, 9]
  13          3         MULTD F11, F7, F8          26        27-46                  47             48         3.750                     ['ROB12', 1.5, 2.5]
  14          4         BNE R20, R21, -5           27        28                                    49         5          P: True        [16, 16, -5]
~ 15          0         ADDI R20, R20, 2           28        None                   None           None       None                      ['ROB13', 16, 2]

---------- Registers ----------
Integer Register -  00: 0
01: 0     02: 0     03: 12    04: 9     05: 0     06: 0     07: 0     08: 0     09: 21    10: 0     11: 0     12: 0     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 16    21: 16    22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Float Register -    00: 0
01: 2.30  02: 0     03: 0     04: 0     05: 0     06: 0     07: 1.50  08: 2.50  09: 0     10: 0     11: 3.75  12: 0     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Memory -
0000: 2.30

---------- Asserts ----------
True  - R20: 16 == 16
True  - F1: 2.3 == 2.3000
True  - R9: 21 == 21
True  - F11: 3.75 == 3.7500
asserts: True
```

```
Running Test Case 6...
```

Code

```
# simple loop demonstrating functionality of branch equations, branch prediction
# and speculation recovery.

$ R1 = 4
$ R2 = 12
$ R3 = 8
$ R4 = 4

! R1 = 12
! R4 = 8
! R5 = 0
! R6 = 0
! R7 = 24

LoopA: ADDI R1, R1, 2
BNE R1, R2, LoopA
ADDI R4, R4, 4
BEQ R3, R4, LoopB
ADDI R5, R5, 5
ADD R6, R5, R1
LoopB: ADD R7, R1, R2
```

Execution

Counter	Index	Instruction	Issue	Execute	Memory	Write Back	Commit	Result	Extra Info	Operands
0	0	ADDI R1, R1, 2	1	2		3	4	6		['ROB1', 4, 2]
1	1	BNE R1, R2, -2	2	4			5	0	P: False	[6, 12, -2]
~~2~~	~~2~~	~~ADDI R4, R4, 4~~	~~3~~	~~None~~		~~None~~	~~None~~	~~None~~		~~['ROB2', 4, 4]~~
~~3~~	~~3~~	~~BEQ R3, R4, 2~~	~~4~~	~~None~~			~~None~~	~~None~~	~~P: False~~	~~[8, R4 -> ROB2, 2]~~
4	0	ADDI R1, R1, 2	6	7		8	9	8		['ROB3', 6, 2]
5	1	BNE R1, R2, -2	7	9			10	0	P: True	[8, 12, -2]
6	0	ADDI R1, R1, 2	8	10		11	12	10		['ROB4', 8, 2]
7	1	BNE R1, R2, -2	9	12			13	0	P: True	[10, 12, -2]
8	0	ADDI R1, R1, 2	10	13		14	15	12		['ROB5', 10, 2]
9	1	BNE R1, R2, -2	11	15			16	2	P: True	[12, 12, -2]
~~10~~	~~0~~	~~ADDI R1, R1, 2~~	~~12~~	~~None~~		~~None~~	~~None~~	~~None~~		~~['ROB6', 12, 2]~~
~~11~~	~~1~~	~~BNE R1, R2, -2~~	~~13~~	~~None~~			~~None~~	~~None~~	~~P: True~~	~~[R1 -> ROB6, 12, -2]~~
~~12~~	~~0~~	~~ADDI R1, R1, 2~~	~~14~~	~~None~~		~~None~~	~~None~~	~~None~~		~~['ROB7', R1 -> ROB6, 2]~~
~~13~~	~~1~~	~~BNE R1, R2, -2~~	~~15~~	~~None~~			~~None~~	~~None~~	~~P: True~~	~~[R1 -> ROB7, 12, -2]~~
14	2	ADDI R4, R4, 4	17	18		19	20	8		['ROB8', 4, 4]
15	3	BEQ R3, R4, 2	18	20			21	6	P: False	[8, 8, 2]
~~16~~	~~4~~	~~ADDI R5, R5, 5~~	~~19~~	~~None~~		~~None~~	~~None~~	~~None~~		~~['ROB9', 0, 5]~~
~~17~~	~~5~~	~~ADD R6, R5, R1~~	~~20~~	~~None~~		~~None~~	~~None~~	~~None~~		~~['ROB10', R5 -> ROB9, 12]~~
18	6	ADD R7, R1, R2	22	23		24	25	24		['ROB11', 12, 12]

Registers

```
Integer Register -  00: 0
01: 12    02: 12    03: 8     04: 8     05: 0     06: 0     07: 24    08: 0     09: 0     10: 0     11: 0     12: 0     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Float Register -    00: 0
01: 0     02: 0     03: 0     04: 0     05: 0     06: 0     07: 0     08: 0     09: 0     10: 0     11: 0     12: 0     13: 0     14: 0     15: 0     16: 0
17: 0     18: 0     19: 0     20: 0     21: 0     22: 0     23: 0     24: 0     25: 0     26: 0     27: 0     28: 0     29: 0     30: 0     31: 0     32: 0

Memory -
```

Asserts

```
True  - R1: 12 == 12
True  - R4: 8 == 8
True  - R5: 0 == 0
True  - R6: 0 == 0
True  - R7: 24 == 24
asserts: True
```