Brad Shea
Eric Shea
Zhanneta Plokhovska

Tomasulo Implementation, Phase II

I. Section 1

A. *System Requirements*

1. Windows OS
2. Python 2.7.14

B. *How to Compile/Run*

1. Open command prompt and navigate to folder “TomasuloAlgorithm”
2. Run tomasulo_main.py by executing the following command (replace [input_file_name] with your input filename):
    - python tomasulo_main.py [input_file_name].txt
3. Output will be displayed to the screen (open command prompt in fullscreen mode for better readability)

C. *Input File Format (see Fig. 1)*

1. Integer Adder properties have to be set in the following format:
    - int_adder [num_of_rs] [cycles_in_ex] [num_of_fus]
2. Floating Point Adder properties have to be set in the following format:
    - fp_adder [num_of_rs] [cycles_in_ex] [num_of_fus]
3. Floating Point Multiplier properties have to be set in the following format:
    - fp_multiplier [num_of_rs] [cycles_in_ex] [num_of_fus]
4. Load Store Unit properties have to be set in the following format:
    - load_store_unit [num_of_rs] [cycles_in_ex] [cycles_in_mem] [num_of_fus]
5. Register values have to be set in the following format:
    - reg [reg_name] [reg_value]
6. Memory values have to be set in the following format:
    - mem [mem_address_in_bytes] [mem_value]
7. Individual instructions are parsed by space, do not use commas
8. Input file is insensitive to order between properties/instructions/etc.
9. Input file is case-insensitive
10. Input file can handle comments that begin with “# “ and empty lines

```
# FU_name #_of_rs cycles_in_EX #_of_FUs
int_adder 2 1 1
fp_adder 3 3 1
fp_multiplier 2 20 1
# FU_name #_of_rs cycles_in_EX cycles_in_mem #_of_FUs
load_store_unit 3 1 4 1
# rob_entries #_of_rob_entries
rob_entries 10
# reg reg_name reg_value
reg R1 10
reg R2 20
reg F2 30.1
# mem mem_address mem_value
mem 4 1
mem 8 2
mem 12 3.4
# instructions
Add.d F1 F2 F3
Add R2 R1 R0
Mult.d F4 F2 F3
```

Fig. 1. Input File Format Example

D. Source Files

Table 1. Source Files Description

Source Filename	Description
tomasulo_main.py	Main file responsible for initializing tomasulo structures and controlling the flow of tomasulo algorithm
tomasulo_mem.py	Object file for Memory structure
tomasulo_arf.py	Object file for ARF structure (integer, floating point)
tomasulo_rat.py	Object file for RAT structure (integer, floating point)
tomasulo_rob.py	Object file for ROB structure
tomasulo_rs.py	Object file for RS structure (integer adder, floating point adder, floating point multiplier)
tomasulo_load_store_queue.py	Object file for Load Store Queue structure
tomasulo_timing_table.py	Object file for Timing Table structure

II. Section 2

A. *Test Case 1*

1. No dependencies exist among instructions
2. Sub.d does not have to wait after add.d to be executed since the floating point adder is pipelined

Pseudocode:

```
A = B+C;
D = B-C;
E = J*K;
F = G+H;        //FP
I = G-H;        //FP
```

Assembly code:

```
int_adder 2 1 1
fp_adder 3 3 1
fp_multiplier 2 20 1
load_store_unit 3 1 4 1
rob_entries 10
reg R1 0                //A
reg R2 2                //B
reg R3 4                //C
reg R4 0                //D
reg F1 0                //E
reg F2 0                //F
reg F3 1.1              //G
reg F4 2.5              //H
reg F5 0                //I
reg F6 4.4              //J
reg F7 5.5              //K
add R1 R2 R3            //A = B+C
sub R4 R2 R3            //D = B-C
Mult.d F1 F6 F7         //E = J*K
Add.d F2 F3 F4          //F = G+H
Sub.d F5 F3 F4          //I = G-H
```

Expected Results:

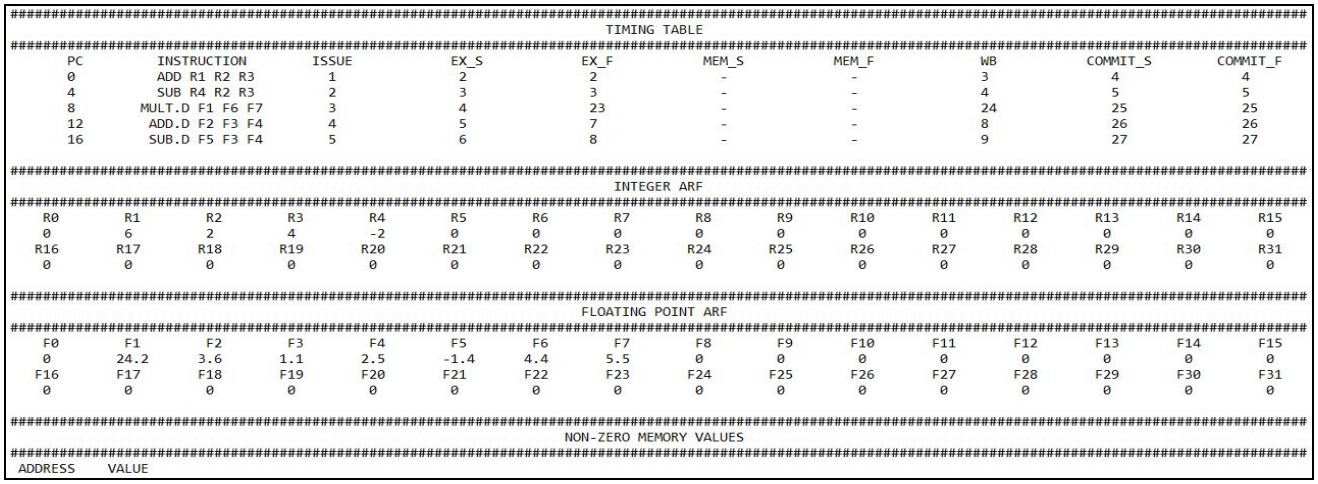
Table 2. Clock Cycle Tracking for Test Case 1

PC	Instr	Operands	ISSUE	EX	MEM	WB	COMMIT
0x00	Add	R1, R2, R3	1	2		3	4

0x04	Sub	R4, R2, R3	2	3		4	5
0x08	Mult.d	F1, F6, F7	3	4-23		24	25
0x0C	Add.d	F2, F3, F4	4	5-7		8	26
0x10	Sub.d	F5, F3, F4	5	6-8		9	27

Table 3. Register Values for Test Case 1

Register	Value (decimal)
R1	6
R2	2
R3	4
R4	-2
F1	24.2
F2	3.6
F3	1.1
F4	2.5
F5	-1.4
F6	4.4
F7	5.5



TIMING TABLE

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	ADD R1 R2 R3	1	2	2	-	-	3	4	4
4	SUB R4 R2 R3	2	3	3	-	-	4	5	5
8	MULT.D F1 F6 F7	3	4	23	-	-	24	25	25
12	ADD.D F2 F3 F4	4	5	7	-	-	8	26	26
16	SUB.D F5 F3 F4	5	6	8	-	-	9	27	27

INTEGER ARF

R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
0	6	2	4	-2	0	0	0	0	0	0	0	0	0	0	0
R16	R17	R18	R19	R20	R21	R22	R23	R24	R25	R26	R27	R28	R29	R30	R31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLOATING POINT ARF

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	24.2	3.6	1.1	2.5	-1.4	4.4	5.5	0	0	0	0	0	0	0	0
F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NON-ZERO MEMORY VALUES

ADDRESS	VALUE

Fig. 2. Official Results for Test Case 1

B. Test Case 2

1. Data dependency on F1 register; the second instruction (mult.d) cannot execute until the first instruction (mult.d) finishes its writeback stage
2. Structure dependency for integer adder rs structure (which has only 1 entry) for instructions 3 & 4 (add & sub, respectively); sub instruction has wait for the add instruction to finish its writeback stage

Pseudocode:

```
A = B*C;
D = A*B;
E = G + H;
F = G - H;
```

Assembly code:

```
int_adder 1 1 1
fp_adder 3 3 1
fp_multiplier 2 20 1
load_store_unit 3 1 4 1
rob_entries 10
reg R1 0            //E
reg R2 3            //G
reg R3 2            //H
reg R4 0            //F
reg F1 0            //A
reg F2 1.1          //B
reg F3 2.5          //C
reg F4 0.0          //D
mult.d F1 F2 F3     //A = B*C
mult.d F4 F1 F2     //D = A*B
add R1 R2 R3        //E = G+H
sub R4 R2 R3        //F = G - H
```

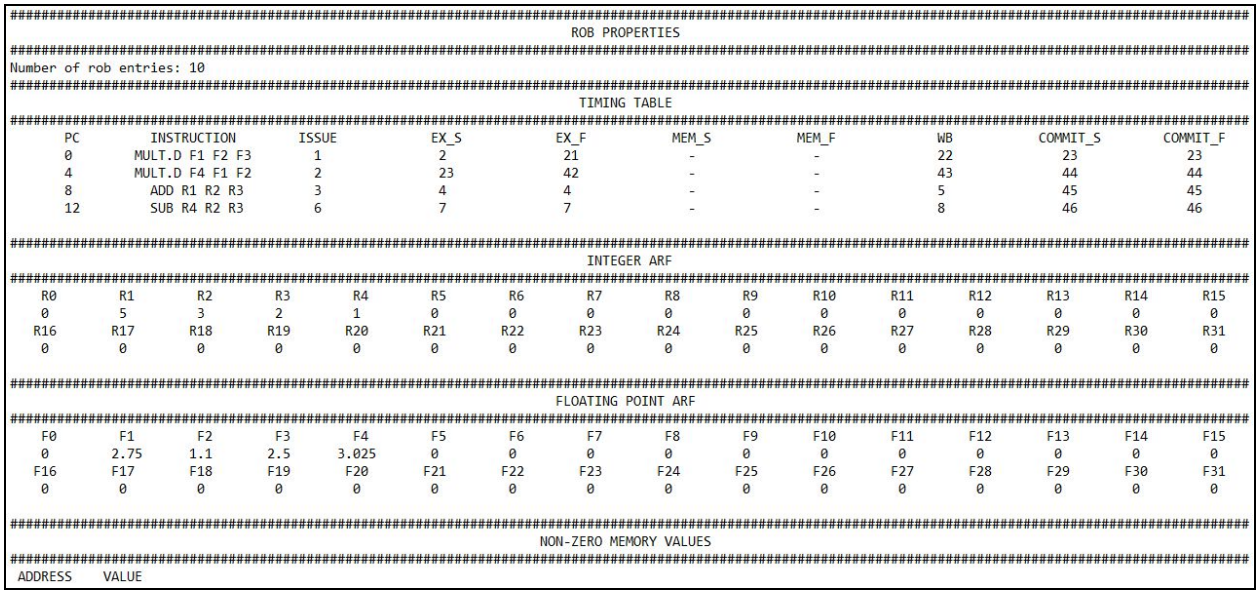
Expected Results:

Table 4. Clock Cycle Tracking for Test Case 2

PC	Instr	Operands	ISSUE	EX	MEM	WB	COMMIT
0x00	mult.d	F1, F2, F3	1	2-21		22	23
0x04	mult.d	F4, F1, F2	2	23-42		43	44
0x08	add	R1, R2, R3	3	4		5	45
0x0C	sub	R4, R2, R3	6	7		8	46

Table 5. Register Values for Test Case 2

Register	Value (decimal)
R1	5
R2	3
R3	2
R4	1
F1	2.75
F2	1.1
F3	2.5
F4	3.025



ROB PROPERTIES

Number of rob entries: 10

TIMING TABLE

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	MULT.D F1 F2 F3	1	2	21	-	-	22	23	23
4	MULT.D F4 F1 F2	2	23	42	-	-	43	44	44
8	ADD R1 R2 R3	3	4	4	-	-	5	45	45
12	SUB R4 R2 R3	6	7	7	-	-	8	46	46

INTEGER ARF

R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
0	5	3	2	1	0	0	0	0	0	0	0	0	0	0	0
R16	R17	R18	R19	R20	R21	R22	R23	R24	R25	R26	R27	R28	R29	R30	R31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLOATING POINT ARF

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	2.75	1.1	2.5	3.025	0	0	0	0	0	0	0	0	0	0	0
F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NON-ZERO MEMORY VALUES

ADDRESS VALUE

Fig. 3. Official Results for Test Case 2

C. *Test Case 3*

1. Dependency on number of RS entries available for ld/sd; the fourth ld instruction has to wait for a free entry in the RS until it can be issued. The two store instruction also have to wait for a free entry in the reservation station
2. Structure hazard for ld instructions in the memory stage; there can only be one memory stage executing at a time so each ld has to wait until the previous ld finishes its memory stage

3. Data dependency for sd instruction; the first sd instruction can be issued and executed but has to stall before writeback stage since it is waiting on the result of F5 from the add.d instruction

Pseudocode:

```
uint8_t A,C = 0;
uint8_t B[3] = {1 2 3};
uint8_t D[3] = {1 2 3};
uint8_t E[3];
A = B[i];        //load
C = D[i];
E[i] = A*C;      //store
```

Assembly code:

```
int_adder 2 1 1
fp_adder 3 3 1
fp_multiplier 2 20 1
load_store_unit 3 1 4 1
rob_entries 10
reg R1 0              //A
reg R2 0              //C
mem 4 3.3
mem 8 2.2
mem 12 3.4
mem 16 5.5
Ld F1 1 R0            //load mem[4]
Ld F2 2 R0            //load mem[8]
Ld F3 3 R0            //load mem[12]
Ld F4 4 R0            //load mem[16]
add.d F5 F1 F2        //
add.d F6 F5 F3        //
Sd F5 20 R0
Sd F6 24 R0           //store F6 in memory
```

Expected Results:

Table 6. Clock Cycle Tracking for Test Case 3

PC	Instr	Operands	ISSUE	EX	MEM	WB	COMMIT
0x00	Ld	F1, 1(R0)	1	2	3-6	7	8
0x04	Ld	F2, 2(R0)	2	3	7-10	11	12
0x08	Ld	F3, 3(R0)	3	4	11-14	15	16

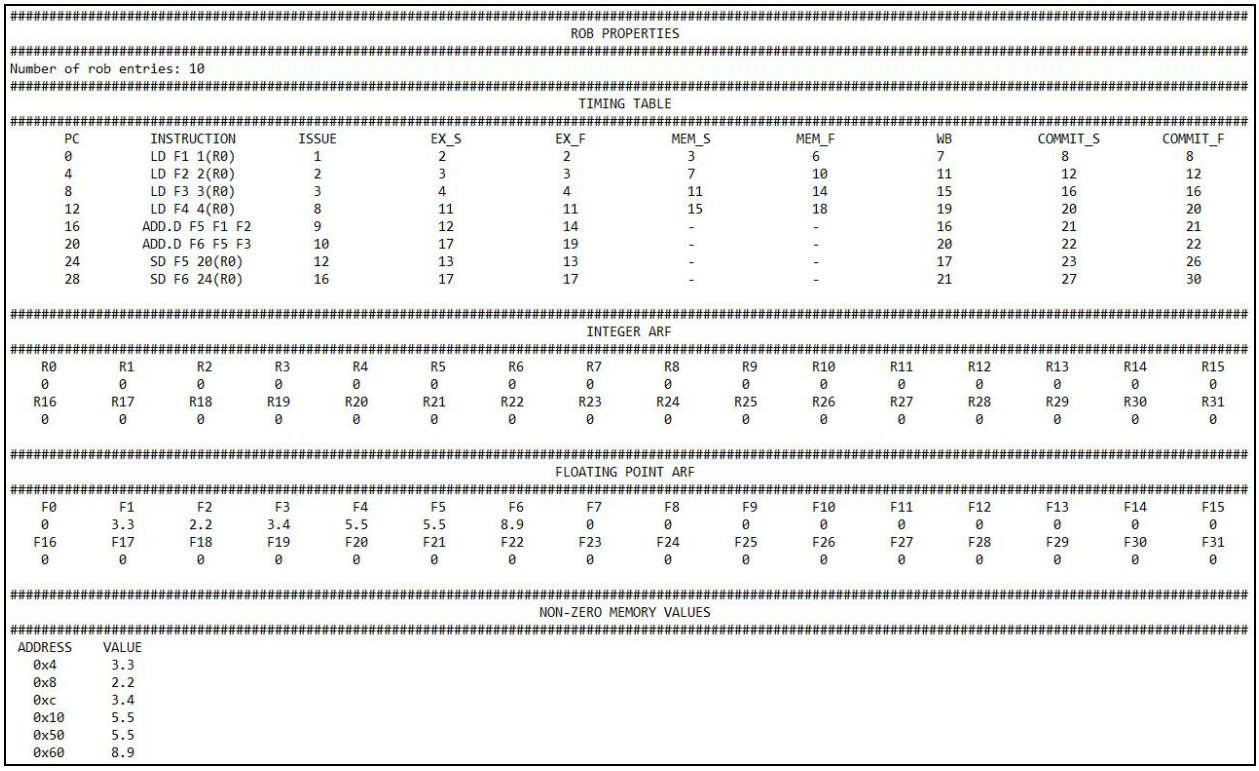
0x0C	Ld	F4, 4(R0)	8	11	15-18	19	20
0x10	Add.d	F5, F1, F2	9	12-14		16	21
0x14	Add.d	F6, F5, F3	10	17-19		20	22
0x18	Sd	F5, 20(R0)	12	13		17	23-26
0x1C	Sd	F6, 24(R0)	16	17		21	27-30

Table 7. Register Values for Test Case 3

Register	Value (decimal)
F1	3.3
F2	2.2
F3	3.4
F4	5.5
F5	5.5
F6	8.9

Table 8. Memory Values for Test Case 3

Address (bytes)	Value (decimal)
0x04	3.3
0x08	2.2
0x0C	3.4
0x10	5.5
0x50	5.5
0x60	8.9



ROB PROPERTIES

Number of rob entries: 10

TIMING TABLE

PC	INSTRUCTION	ISSUE	EX_S	EX_F	MEM_S	MEM_F	WB	COMMIT_S	COMMIT_F
0	LD F1 1(R0)	1	2	2	3	6	7	8	8
4	LD F2 2(R0)	2	3	3	7	10	11	12	12
8	LD F3 3(R0)	3	4	4	11	14	15	16	16
12	LD F4 4(R0)	8	11	11	15	18	19	20	20
16	ADD.D F5 F1 F2	9	12	14	-	-	16	21	21
20	ADD.D F6 F5 F3	10	17	19	-	-	20	22	22
24	SD F5 20(R0)	12	13	13	-	-	17	23	26
28	SD F6 24(R0)	16	17	17	-	-	21	27	30

INTEGER ARF

R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
R16	R17	R18	R19	R20	R21	R22	R23	R24	R25	R26	R27	R28	R29	R30	R31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLOATING POINT ARF

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	3.3	2.2	3.4	5.5	5.5	8.9	0	0	0	0	0	0	0	0	0
F16	F17	F18	F19	F20	F21	F22	F23	F24	F25	F26	F27	F28	F29	F30	F31
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

NON-ZERO MEMORY VALUES

ADDRESS	VALUE
0x4	3.3
0x8	2.2
0xc	3.4
0x10	5.5
0x50	5.5
0x60	8.9

Fig. 4. Official Results for Test Case 3

D. Test Case 4

1. This test case shows the forwarding of data in the queue. Since we already know what will be placed in memory location 0x4, the next two ld instructions can be forwarded this value so they do not have to retrieve it from memory

Pseudocode:

```
uint8_t A,C = 0;
uint8_t B[3] = {1 2 3};
uint8_t D[3] = {1 2 3};
uint8_t E[3];
A = B[i];          //load
C = B[i];
D = B[i];
```

Pseudocode:

```
int_adder 2 1 1
fp_adder 3 3 1
fp_multiplier 2 20 1
load_store_unit 3 1 4 1
rob_entries 10
Reg F1 7.8
```

```
Reg F2 0
Reg F3 0
mem 4 3
mem 8 2
mem 12 3.4
mem 16 5.5
Sd F1 1 R0          //store mem[4]
Ld F2 1 R0          //load mem[8]
Ld F3 1 R0          //load mem[12]
```

Expected Results:

Table 9. Clock Cycle Tracking for Test Case 4

PC	Instr	Operands	ISSUE	EX	MEM	WB	COMMIT
0x00	Sd	F1, 1(R0)	1	2		3	4-7
0x04	Ld	F2, 1(R0)	2	3	4	5	6
0x08	Ld	F3, 1(R0)	3	4	5	6	7

Table 10. Register Values for Test Case 4

Register	Value (decimal)
F1	7.8
F2	7.8
F3	7.8

Table 11. Memory Values for Test Case 4

Address (bytes)	Value (decimal)
0x04	7.8
0x08	2.0
0x0C	3.4
0x10	5.5

```
################################################################################################################################################
                                                        ROB PROPERTIES
################################################################################################################################################
Number of rob entries: 10
################################################################################################################################################
                                                        TIMING TABLE
################################################################################################################################################
      PC        INSTRUCTION       ISSUE          EX_S          EX_F          MEM_S         MEM_F          WB          COMMIT_S       COMMIT_F
      0         SD F1 1(R0)         1             2             2             -             -             3             4             7
      4         LD F2 1(R0)         2             3             3             4             4             5             6             6
      8         LD F3 1(R0)         3             4             4             5             5             6             7             7

################################################################################################################################################
                                                        INTEGER ARF
################################################################################################################################################
    R0        R1        R2        R3        R4        R5        R6        R7        R8        R9        R10       R11       R12       R13       R14       R15
    0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
   R16       R17       R18       R19       R20       R21       R22       R23       R24       R25       R26       R27       R28       R29       R30       R31
    0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0

################################################################################################################################################
                                                        FLOATING POINT ARF
################################################################################################################################################
    F0        F1        F2        F3        F4        F5        F6        F7        F8        F9        F10       F11       F12       F13       F14       F15
    0         7.8       7.8       7.8       0         0         0         0         0         0         0         0         0         0         0         0
   F16       F17       F18       F19       F20       F21       F22       F23       F24       F25       F26       F27       F28       F29       F30       F31
    0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0

################################################################################################################################################
                                                        NON-ZERO MEMORY VALUES
################################################################################################################################################
 ADDRESS    VALUE
   0x4       7.8
   0x8       2.0
   0xc       3.4
   0x10      5.5
```

Fig. 5. Official Results for Test Case 4

E. *Test Case 5*

1. Data dependency for F1, F4, & R1 registers; they have to wait to writeback
2. Testing branch

Pseudocode:

```
for(i=0; i<2; i++)
{
        A = B*C;
        D = B + E;
}
```

Assembly code:

```
int_adder 2 1 1
fp_adder 3 3 1
fp_multiplier 2 20 1
load_store_unit 3 1 4 1
rob_entries 10
reg R1 0            //for the for loop
reg R2 2            //initialize i to 0
reg F1 0            //A
reg F2 1.1          //B
reg F3 2.5          //C
reg F4 0.0          //D
reg F5 1.5          //E
```

```
mem 4 3
mem 8 2
mem 12 3.4
mult.d F1 F2 F3         //A = B*C
add.d F4 F2 F5          //D = B + E
Addi R1 R1 1            //increment i
Bne R1 R2 -4  //check if i=2, if yes then done
Sd F4 8(R1)             //store D in memory
```

Expected Results:

Table 12. Clock Cycle Tracking for Test Case 5

PC	Instr	Operands	ISSUE	EX	MEM	WB	COMMIT
0x00	mult.d	F1, F2, F3	1	2-21		22	23
0x04	add.d	F4, F2, F5	2	3-5		6	24
0x08	add.i	R1, R1, 1	3	4		5	25
0x0C	bne	R1, R2, -4	4	6		7	26
0x00	mult.d	F1, F2, F3	7	8-27		28	29
0x04	add.d	F4, F2, F5	8	9-11		12	30
0x08	add.i	R1, R1, 1	9	10		11	31
0x0C	bne	R1, R2, -4	10	12		13	32
0x10	Sd	F4, 8(R1)	13	14		15	33-36

Table 13. Register Values for Test Case 5

Register	Value (decimal)
R1	2
R2	2
F1	2.75
F2	1.1
F3	2.5
F4	2.6
F5	1.5

Table 14. Memory Values for Test Case 5

Address (bytes)	Value (decimal)
0x04	3.0
0x08	2.0
0x12	3.4
0x32	2.6

```
################################################################################################################################################
                                                                ROB PROPERTIES
################################################################################################################################################
Number of rob entries: 10
################################################################################################################################################
                                                                TIMING TABLE
################################################################################################################################################
      PC        INSTRUCTION        ISSUE          EX_S          EX_F         MEM_S         MEM_F           WB       COMMIT_S       COMMIT_F
       0       MULT.D F1 F2 F3         1             2            21             -             -           22             23             23
       4        ADD.D F4 F2 F5         2             3             5             -             -            6             24             24
       8         ADDI R1 R1 1          3             4             4             -             -            5             25             25
      12          BNE R1 R2 -4         4             6             6             -             -            7             26             26
       0       MULT.D F1 F2 F3         7             8            27             -             -           28             29             29
       4        ADD.D F4 F2 F5         8             9            11             -             -           12             30             30
       8         ADDI R1 R1 1          9            10            10             -             -           11             31             31
      12          BNE R1 R2 -4        10            12            12             -             -           13             32             32
      16          SD F4 8(R1)         13            14            14             -             -           15             33             36

################################################################################################################################################
                                                                INTEGER ARF
################################################################################################################################################
    R0        R1        R2        R3        R4        R5        R6        R7        R8        R9       R10       R11       R12       R13       R14       R15
     0         2         2         0         0         0         0         0         0         0         0         0         0         0         0         0
   R16       R17       R18       R19       R20       R21       R22       R23       R24       R25       R26       R27       R28       R29       R30       R31
     0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0

################################################################################################################################################
                                                                FLOATING POINT ARF
################################################################################################################################################
    F0        F1        F2        F3        F4        F5        F6        F7        F8        F9       F10       F11       F12       F13       F14       F15
     0      2.75       1.1       2.5       2.6       1.5         0         0         0         0         0         0         0         0         0         0
   F16       F17       F18       F19       F20       F21       F22       F23       F24       F25       F26       F27       F28       F29       F30       F31
     0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0

################################################################################################################################################
                                                                NON-ZERO MEMORY VALUES
################################################################################################################################################
 ADDRESS    VALUE
   0x4       3.0
   0x8       2.0
   0xc       3.4
  0x20       2.6
```

Fig. 6. Official Results for Test Case 5

F. *Test Case 6*

1. Structural dependency for limited amount of mult.d reservation stations. If there are only three RS entries for mult.d, then the fourth Mult.d instruction will have to wait until one of the Mult.d finishes writeback stage to free up RS entry

Pseudocode:

A = B*C;
D = E*F;
G = H*I;
J = K*L;

Assembly code:

```
int_adder 2 1 1
fp_adder 3 3 1
fp_multiplier 3 20 1
load_store_unit 3 1 4 1
rob_entries 10
reg F1 0              //A
reg F2 1.1            //B
reg F3 2.5            //C
reg F4 0.0            //D
reg F5 1.5            //E
reg F6 1.5            //F
reg F7 0              //G
reg F8 3.5            //H
reg F9 3.5            //I
reg F10 0             //J
reg F11 5.5           //K
reg F12 5.5           //L
mult.d F1 F2 F3           //A = B*C
mult.d F4 F5 F6           //D = E*F
mult.d F7 F8 F9           //G = H*I
mult.d F10 F11 F12        //J = K*L
```

Expected Results:

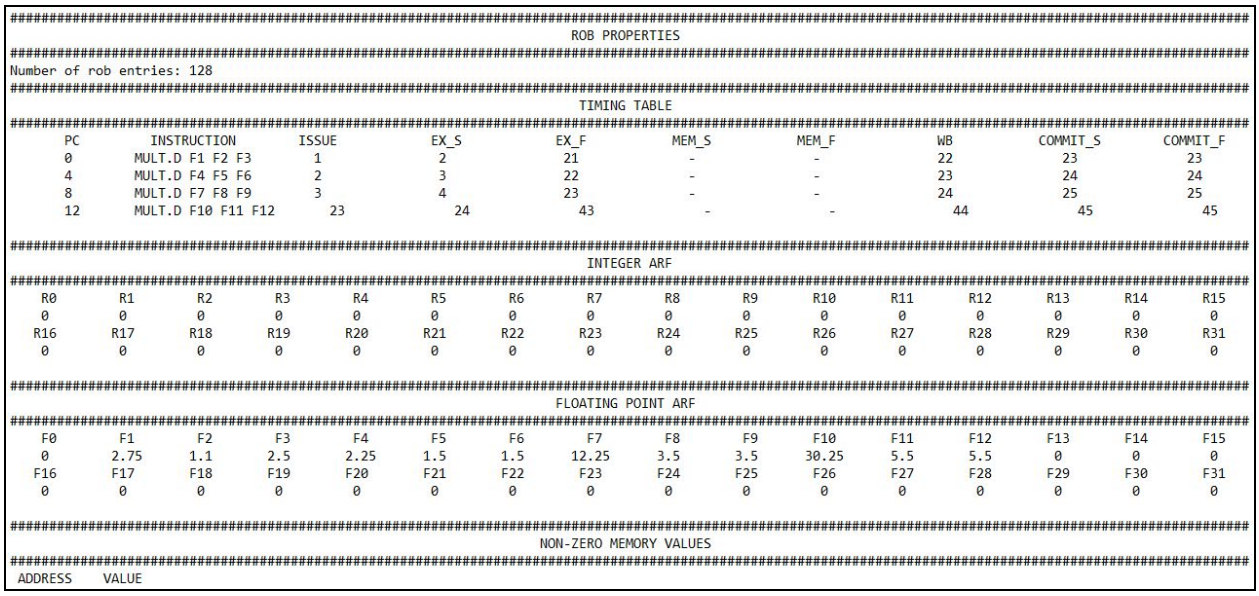
Table 15. Clock Cycle Tracking for Test Case 6

PC	Instr	Operands	ISSUE	EX	MEM	WB	COMMIT
	mult.d	F1, F2, F3	1	2-21		22	23
	mult.d	F4, F5, F6	2	3-22		23	24
	mult.d	F7, F8, F9	3	4-23		24	25
	mult.d	F10, F11, F12	23	24-43		44	45

Table 16. Register Values for Test Case 6

Register	Value (decimal)
F1	2.75
F2	1.1
F3	2.5

F4	2.25
F5	1.5
F6	1.5
F7	12.25
F8	3.5
F9	3.5
F10	30.25
F11	5.5
F12	5.5



```
################################################################################################################################################################
                                                                ROB PROPERTIES
################################################################################################################################################################
Number of rob entries: 128
################################################################################################################################################################
                                                                TIMING TABLE
################################################################################################################################################################
      PC         INSTRUCTION        ISSUE          EX_S          EX_F          MEM_S         MEM_F          WB          COMMIT_S       COMMIT_F
      0        MULT.D F1 F2 F3        1             2             21             -             -             22            23             23
      4        MULT.D F4 F5 F6        2             3             22             -             -             23            24             24
      8        MULT.D F7 F8 F9        3             4             23             -             -             24            25             25
      12       MULT.D F10 F11 F12      23            24            43             -             -             44            45             45

################################################################################################################################################################
                                                                INTEGER ARF
################################################################################################################################################################
    R0        R1        R2        R3        R4        R5        R6        R7        R8        R9        R10       R11       R12       R13       R14       R15
    0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0
   R16       R17       R18       R19       R20       R21       R22       R23       R24       R25       R26       R27       R28       R29       R30       R31
    0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0

################################################################################################################################################################
                                                                FLOATING POINT ARF
################################################################################################################################################################
    F0        F1        F2        F3        F4        F5        F6        F7        F8        F9        F10       F11       F12       F13       F14       F15
    0         2.75      1.1       2.5       2.25      1.5       1.5       12.25     3.5       3.5       30.25     5.5       5.5       0         0         0
   F16       F17       F18       F19       F20       F21       F22       F23       F24       F25       F26       F27       F28       F29       F30       F31
    0         0         0         0         0         0         0         0         0         0         0         0         0         0         0         0

################################################################################################################################################################
                                                                NON-ZERO MEMORY VALUES
################################################################################################################################################################
 ADDRESS     VALUE
```

Fig. 7. Official Results for Test Case 6

III. Responsibilities of Group Members

A. <u>Brad Shea</u> (main developer of python classes for tomasulo structures)

1. Developed python classes for tomasulo structures in Python:
    - tomasulo_mem.py
    - tomasulo_arf.py
    - tomasulo_rat.py
    - tomasulo_rob.py
    - tomasulo_rs.py
    - tomasulo_load_store_queue.py

- tomasulo_timing_table.py

2. Ran a second verification round of final results
3. Held group discussions with teammates to acquire a common understanding of tomasulo algorithm and resolve issues

B. *Eric Shea (main developer of test cases)*
   1. Developed all test cases and hand-derived timing cycle and register output data
   2. Debugged hand-derived test cases with actual results
   3. Held group discussions with teammates to acquire a common understanding of tomasulo algorithm and resolve issues

C. *Zhanneta Plokhovska (main developer of tomasulo algorithm flow)*
   1. Developed the main part of the code in Python:
      - tomasulo_main.py
   2. Debugged the main flow of the algorithm using test cases written by Eric
   3. Held group discussions with teammates to acquire a common understanding of tomasulo algorithm and resolve issues