Tomasulo's algorithm Simulator

Salma Soliman 900182325
Andrew Hany 900184042

Presented to:
Dr. Cherif Salama
Computer Architecture Course
.
10/12/2020

Project Description:

==========

The goal of this project is to implement an architectural simulator capable of assessing the performance of a simplified out-of-order 16-bit RISC processor that uses Tomasulo's algorithm without speculation.

The processor is a single issue processor. It has 8 general-purpose registers R0 to R7 (16-bit each). The register R0 always contains the value 0 and cannot be changed. Memory is word addressable and uses a 16-bit address (as such the memory capacity is 128KB).

Implementation language:

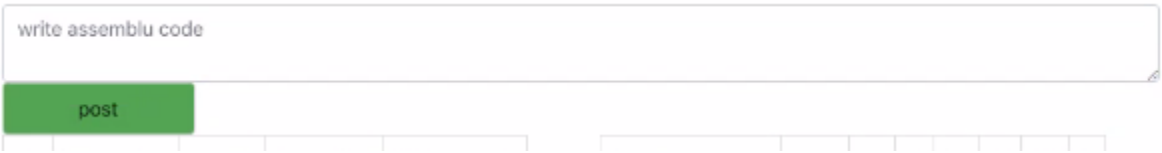
==========

JavaScript and HTML are the languages used in this project. The resulting application is a web-based application for educational and tracking purposes.

The user Guide for using the online Tomasulo's algorithm simulator:

==========

- This program supports only LW, SW, BEQ, JALR, RET, ADD, NEG, ADDI and DIV. any other provided instruction won't be processed and will cause an error. Also, they should be written in small letters.
- Each instruction should be terminated by a semicolon. An error would appear without it.
- You can write your program in this console. And then click on the post button to process the code.



- All clock cycles will appear at the top left of the screen.
- To process the next clock cycle, press on the next button in the lower right side of the screen.

- The screen contains the reservation station unit table on the right side and the registers unit table on the lower left side of the screen.
  Registers table contains the registers names, values and operation done on them.

x0	x1	x2	x3	x4	x5	x6	x7
1	2	1	6	1	1	11	1
						addi	

  Reservation station table contains the busy flag which indicates whether the unit is currently occupied by another instruction or not and the operation done on it. Vj and vk refer to the instruction operands. A refers to the immediate.

Name	busy	op	vj	vk	Qj	Qk	A
LW1	N						
LW2	N						
SW1	N						
SW2	N						
BEQ	N						
JALR/RET	N						
ADD/NEG/ADDI_1	N						
ADD/NEG/ADDI_2	N						
DIV	N						

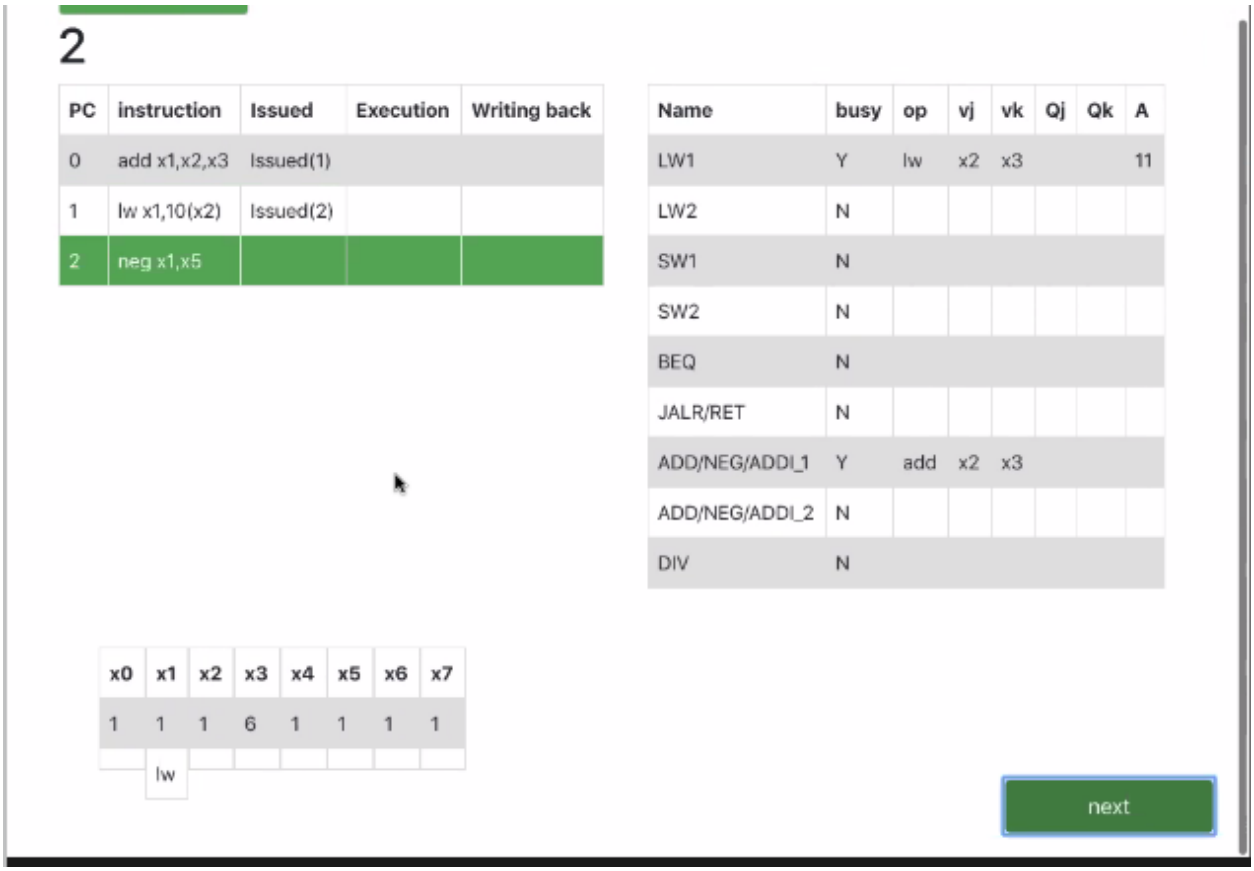
- This table shows the currently running program and the cycles for issuing, executing and writing back for each instruction.

PC	instruction	Issued	Execution	Writing back
0	addi x6,x7,10	Issued(1)	executed(3)	writing back(4)
1	addi x1,x1,1	Issued(2)	executed(4)	writing back(5)
2	beq x1,x3,1	Issued(3)	executed(4)	writing back(5)
3	beq x0,x2,-1	Issued(6)		
4	addi x6,x7,10			

The simulator link:
https://andrew-hany.github.io/tomasulo_algorithm_simulation//

Sample step by step:

==========



2

PC	instruction	Issued	Execution	Writing back
0	add x1,x2,x3	Issued(1)		
1	lw x1,10(x2)	Issued(2)		
2	neg x1,x5			

Name	busy	op	vj	vk	Qj	Qk	A
LW1	Y	lw	x2	x3			11
LW2	N						
SW1	N						
SW2	N						
BEQ	N						
JALR/RET	N						
ADD/NEG/ADDI_1	Y	add	x2	x3			
ADD/NEG/ADDI_2	N						
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	1	1	6	1	1	1	1
	lw						

next

3

PC	instruction	Issued	Execution	Writing back
0	add x1,x2,x3	Issued(1)	executed(3)	
1	lw x1,10(x2)	Issued(2)		
2	neg x1,x5	Issued(3)		

Name	busy	op	vj	vk	Qj	Qk	A
LW1	Y	lw	x2	x3			11
LW2	N						
SW1	N						
SW2	N						
BEQ	N						
JALR/RET	N						
ADD/NEG/ADDI_1	Y	add	x2	x3			
ADD/NEG/ADDI_2	Y	neg	x5				
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	1	1	6	1	1	1	1

next

4

PC	instruction	Issued	Execution	Writing back
0	add x1,x2,x3	Issued(1)	executed(3)	writing back(4)
1	lw x1,10(x2)	Issued(2)	executed(4)	
2	neg x1,x5	Issued(3)		

Name	busy	op	vj	vk	Qj	Qk	A
LW1	Y	lw	x2	x3			11
LW2	N						
SW1	N						
SW2	N						
BEQ	N						
JALR/RET	N						
ADD/NEG/ADDI_1	Y	add	x2	x3			
ADD/NEG/ADDI_2	Y	neg	x5				
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	7	1	6	1	1	1	1

next

6

PC	instruction	Issued	Execution	Writing back
0	add x1,x2,x3	Issued(1)	executed(3)	writing back(4)
1	lw x1,10(x2)	Issued(2)	executed(4)	writing back(5)
2	neg x1,x5	Issued(3)	executed(5)	writing back(6)

Name	busy	op	vj	vk	Qj	Qk	A
LW1	N						
LW2	N						
SW1	N						
SW2	N						
BEQ	N						
JALR/RET	N						
ADD/NEG/ADDI_1	N						
ADD/NEG/ADDI_2	Y	neg	x5				
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	-1	1	6	1	1	1	1

next

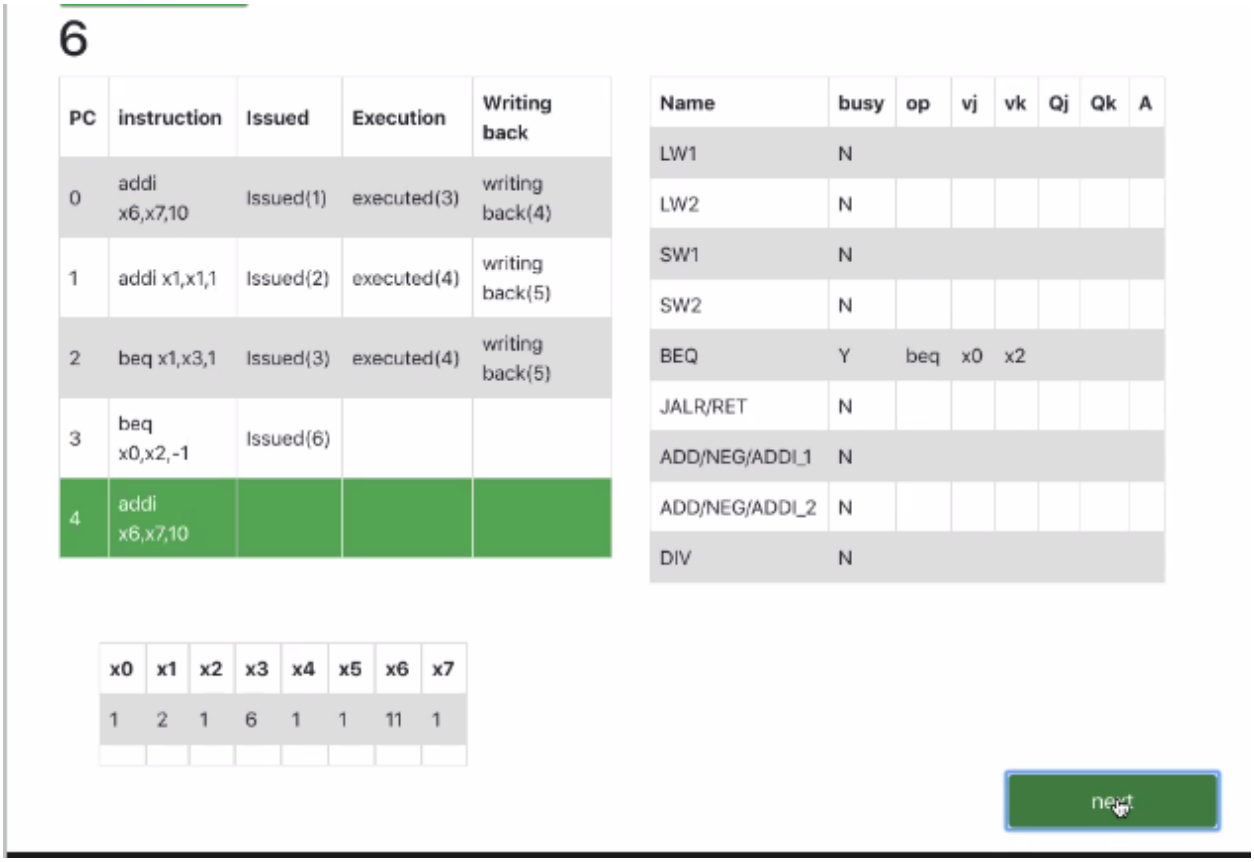
Looping test:

==========

We tried to perform the logic of looping by making the immediate of the BEQ a negative immediate. We initialized x3 with 6 and x1 with 1.
The program is :

```
Addi x6,x7,10
Addi x1,x1,1
Beq x1,x3,1
Beq x0,x2,-1    # looping backward to addi again
Addi x6,x7,10
```

The first BEQ is responsible for exiting the loop and doing the last addi instruction when x1 equals to x3.
The second BEQ is responsible for looping backward to the second addi to increment the x1 five times in a row until being equal with x3.



6

PC	instruction	Issued	Execution	Writing back
0	addi x6,x7,10	Issued(1)	executed(3)	writing back(4)
1	addi x1,x1,1	Issued(2)	executed(4)	writing back(5)
2	beq x1,x3,1	Issued(3)	executed(4)	writing back(5)
3	beq x0,x2,-1	Issued(6)		
4	addi x6,x7,10			

Name	busy	op	vj	vk	Qj	Qk	A
LW1	N						
LW2	N						
SW1	N						
SW2	N						
BEQ	Y	beq	x0	x2			
JALR/RET	N						
ADD/NEG/ADDI_1	N						
ADD/NEG/ADDI_2	N						
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	2	1	6	1	1	11	1

next

post

8

PC	instruction	Issued	Execution	Writing back
0	addi x6,x7,10	Issued(1)	executed(3)	writing back(4)
1	addi x1,x1,1			
2	beq x1,x3,1			
3	beq x0,x2,-1			
4	addi x6,x7,10			

Name	busy	op	vj	vk	Qj	Qk	A
LW1	N						
LW2	N						
SW1	N						
SW2	N						
BEQ	N						
JALR/RET	N						
ADD/NEG/ADDI_1	N						
ADD/NEG/ADDI_2	N						
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	2	1	6	1	1	11	1
						addi	

next

26

PC	instruction	Issued	Execution	Writing back
0	addi x6,x7,10	Issued(1)	executed(3)	writing back(4)
1	addi x1,x1,1	Issued(23)	executed(25)	writing back(26)
2	beq x1,x3,1	Issued(24)	executed(25)	writing back(26)
3	beq x0,x2,-1			
4	addi x6,x7,10			

Name	busy	op	vj	vk	Qj	Qk	A
LW1	N						
LW2	N						
SW1	N						
SW2	N						
BEQ	Y	beq	x1	x3			
JALR/RET	N						
ADD/NEG/ADDI_1	N						
ADD/NEG/ADDI_2	N						
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	5	1	6	1	1	11	1
						addi	

next

30

PC	instruction	Issued	Execution	Writing back
0	addi x6,x7,10	Issued(1)	executed(3)	writing back(4)
1	addi x1,x1,1	Issued(30)		
2	beq x1,x3,1			
3	beq x0,x2,-1			
4	addi x6,x7,10			

Name	busy	op	vj	vk	Qj	Qk	A
LW1	N						
LW2	N						
SW1	N						
SW2	N						
BEQ	N						
JALR/RET	N						
ADD/NEG/ADDI_1	Y	addi	x1	1			
ADD/NEG/ADDI_2	N						
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	5	1	6	1	1	11	1
	addi					addi	

next

37

PC	instruction	Issued	Execution	Writing back
0	addi x6,x7,10	Issued(1)	executed(3)	writing back(4)
1	addi x1,x1,1	Issued(30)	executed(32)	writing back(33)
2	beq x1,x3,1	Issued(31)	executed(32)	writing back(33)
3	beq x0,x2,-1			
4	addi x6,x7,10	Issued(34)	executed(36)	writing back(37)

Name	busy	op	vj	vk	Qj	Qk	A
LW1	N						
LW2	N						
SW1	N						
SW2	N						
BEQ	N						
JALR/RET	N						
ADD/NEG/ADDI_1	N						
ADD/NEG/ADDI_2	N						
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	6	1	6	1	1	11	1

next

Proof of flushing after BEQ:

==========

7

PC	instruction	Issued	Execution	Writing back
0	addi x6,x7,10	Issued(1)	executed(3)	writing back(4)
1	addi x1,x1,1	Issued(2)	executed(4)	writing back(5)
2	beq x1,x3,1	Issued(3)	executed(4)	writing back(5)
3	beq x0,x2,-1	Issued(6)	executed(7)	
4	addi x6,x7,10	Issued(7)		

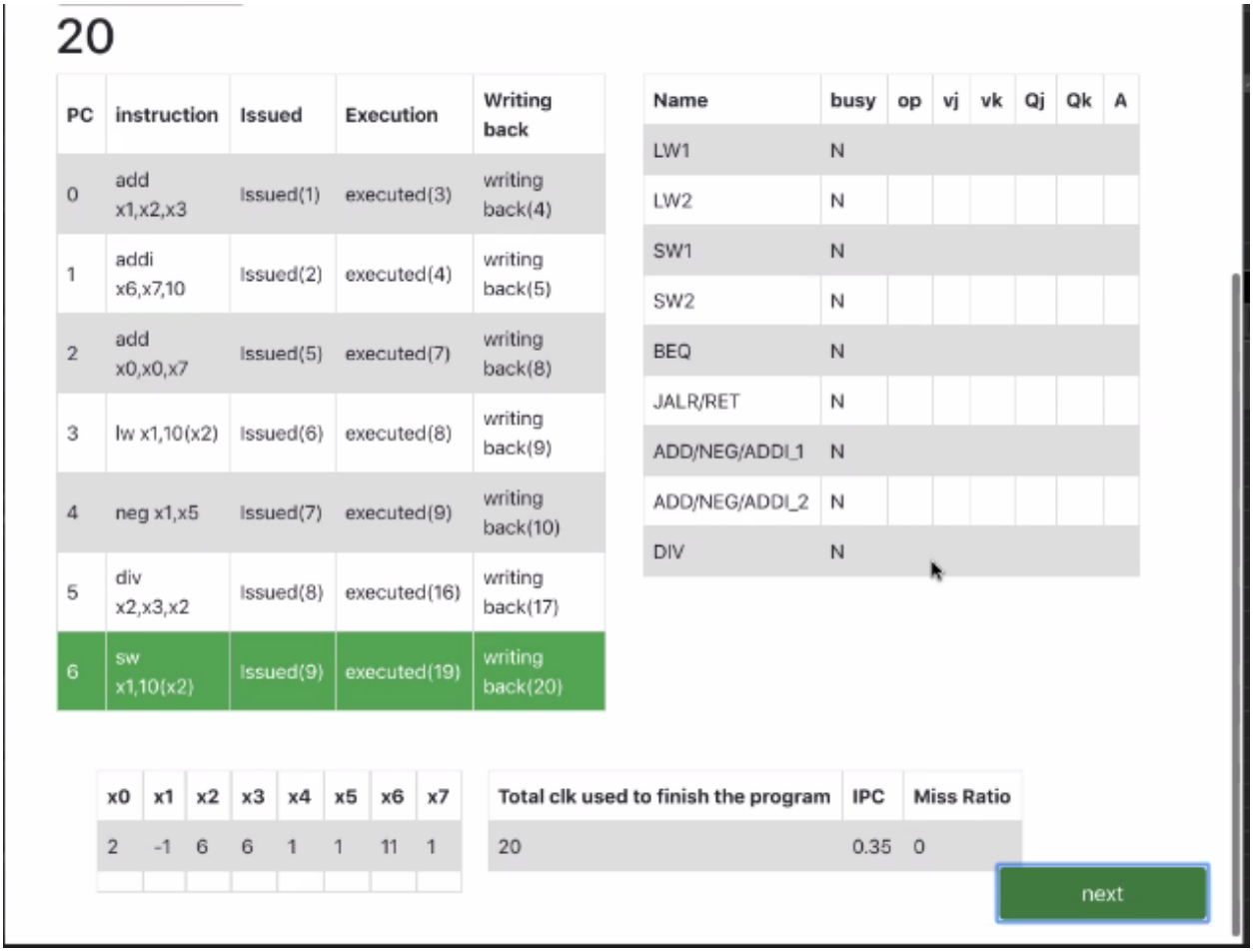
Name	busy	op	vj	vk	Qj	Qk	A
LW1	N						
LW2	N						
SW1	N						
SW2	N						
BEQ	Y	beq	x0	x2			
JALR/RET	N						
ADD/NEG/ADDI_1	Y	addi	x7	10			
ADD/NEG/ADDI_2	N						
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
1	2	1	6	1	1	11	1
						addi	

next

Test case that tests some instructions:

==========



20

PC	instruction	Issued	Execution	Writing back
0	add x1,x2,x3	Issued(1)	executed(3)	writing back(4)
1	addi x6,x7,10	Issued(2)	executed(4)	writing back(5)
2	add x0,x0,x7	Issued(5)	executed(7)	writing back(8)
3	lw x1,10(x2)	Issued(6)	executed(8)	writing back(9)
4	neg x1,x5	Issued(7)	executed(9)	writing back(10)
5	div x2,x3,x2	Issued(8)	executed(16)	writing back(17)
6	sw x1,10(x2)	Issued(9)	executed(19)	writing back(20)

Name	busy	op	vj	vk	Qj	Qk	A
LW1	N						
LW2	N						
SW1	N						
SW2	N						
BEQ	N						
JALR/RET	N						
ADD/NEG/ADDI_1	N						
ADD/NEG/ADDI_2	N						
DIV	N						

x0	x1	x2	x3	x4	x5	x6	x7
2	-1	6	6	1	1	11	1

Total clk used to finish the program	IPC	Miss Ratio
20	0.35	0

next

Test case for jalr:

==========