

**UNIVERSITY OF SCIENCE AND TECHNOLOGY OF HANOI**

Information and Communication Technology Department



## **- REPORT LABWORK 2-**

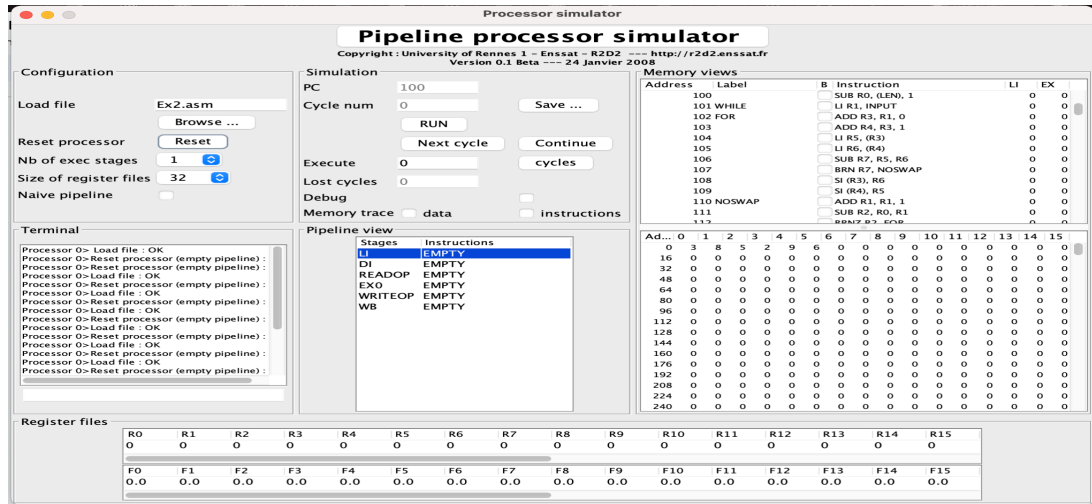
**Name : Phạm Phú Hưng**

**StudentID : BA12-81**

**Major : Cyber Security**

## I. Execute bubble sort algorithm by pipeline simulator

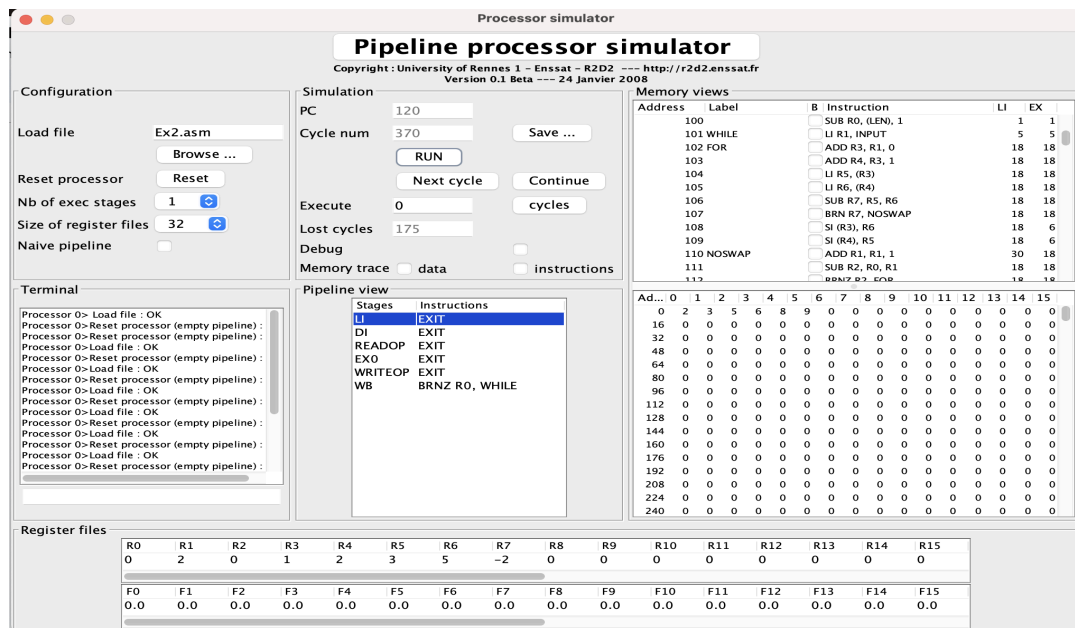
### A.Input



The input is the sequence of numbers 3,8,5,2,9,6

### B.Output

The output is the sequence of numbers 2,3,5,6,8,9



## II. Validate the correctness of program with and without naive mode

In this example,  $R5 - R6 = R7 = 3$  is wrong but the program still continues and performs the swap.





**Processor simulator**

## Pipeline processor simulator

Copyright : University of Rennes 1 - Enssat - R2D2 --- <http://r2d2.enssat.fr>  
Version 0.1 Beta ---- 24 Janvier 2008

**Configuration**

Load file:

Reset processor:

Nb of exec stages:

Size of register files:

Naive pipeline: ☐

**Simulation**

PC:

Cycle num:

Execute:

Lost cycles:

Debug: ☐

Memory trace: ☐ data ☐ instructions

**Memory views**

Address	Label	B	Instruction	LI	EX
100			SUB R0, (LEN), 1	1	1
101	WHILE		LI R1, INPUT		
102	FOR		ADD R3, R1, 0	1	0
103			ADD R4, R3, 1	1	0
104			LI R5, (R3)	1	0
105			LI R6, (R4)		
106			SUB R7, R5, R6		
107			BRN R7, NOSWAP		
108			SI (R3), R6		
109			SI (R4), R5		
110	NOSWAP		ADD R1, R1, 1		
111			SUB R2, R0, R1		
112			BRNZ R2, FOR		

**Terminal**

```

Processor 0> Load file : OK
Processor 0>Reset processor (empty pipeline) :
Processor 0>Reset processor (empty pipeline) :
Processor 0>Load file : OK
Processor 0>Reset processor (empty pipeline) :
Processor 0>Load file : OK
Processor 0>Data trace for dinero : valid
Processor 8>Data trace for dinero : invalid
Processor 8>Reset processor (empty pipeline) :
Processor 0>Load file : OK
Processor 0>

```

**Pipeline view**

Stages	Instructions
LI	LI R5, (R3)
DI	ADD R4, R3, 1
READOP	ADD R3, R1, 0
EX0	LI R1, INPUT
WRITEOP	SUB R0, (LEN), 1
WB	EMPTY

**Register files**

R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

**Processor simulator**

## Pipeline processor simulator

Copyright : University of Rennes 1 - Enssat - R2D2 --- <http://r2d2.enssat.fr>  
Version 0.1 Beta ---- 24 Janvier 2008

**Configuration**

Load file:

Reset processor:

Nb of exec stages:

Size of register files:

Naive pipeline: ☐

**Simulation**

PC:

Cycle num:

Execute:

Lost cycles:

Debug: ☐

Memory trace: ☐ data ☐ instructions

**Memory views**

Address	Label	B	Instruction	LI	EX
112			BRNZ R2, FOR		
113			SUB R0, R0, 1		
114			BRNZ R0, WHILE		
115	EXIT		EXIT		
116			EXIT		
117			EXIT		
118			EXIT		
119			EXIT		
120			EXIT		
121			EXIT		
122			EXIT		
123			EXIT		
124			EXIT		

**Terminal**

```

Processor 0> Load file : OK
Processor 0>Reset processor (empty pipeline) :
Processor 0>Reset processor (empty pipeline) :
Processor 0>Load file : OK
Processor 0>Reset processor (empty pipeline) :
Processor 0>Load file : OK
Processor 0>Data trace for dinero : valid
Processor 8>Data trace for dinero : invalid
Processor 8>Reset processor (empty pipeline) :
Processor 0>Load file : OK
Processor 0>Reset processor (empty pipeline) :
Processor 0>Load file : OK
Processor 0>

```

**Pipeline view**

Stages	Instructions
LI	SI (R4), R5
DI	SI (R3), R6
READOP	BRN R7, NOSWAP
EX0	SUB R7, R5, R6
WRITEOP	EMPTY
WB	LI R6, (R4)

**Register files**

R0	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
5	0	0	0	1	3	8	0	0	0	0	0	0	0	0	0

F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

In this example, R5 is loaded with the value from memory at the address contained in R3, and then R5's value is used to store into another memory location at the address contained in R4. The second instruction depends directly on the result of the first:

### B. Case when a branch is executed

