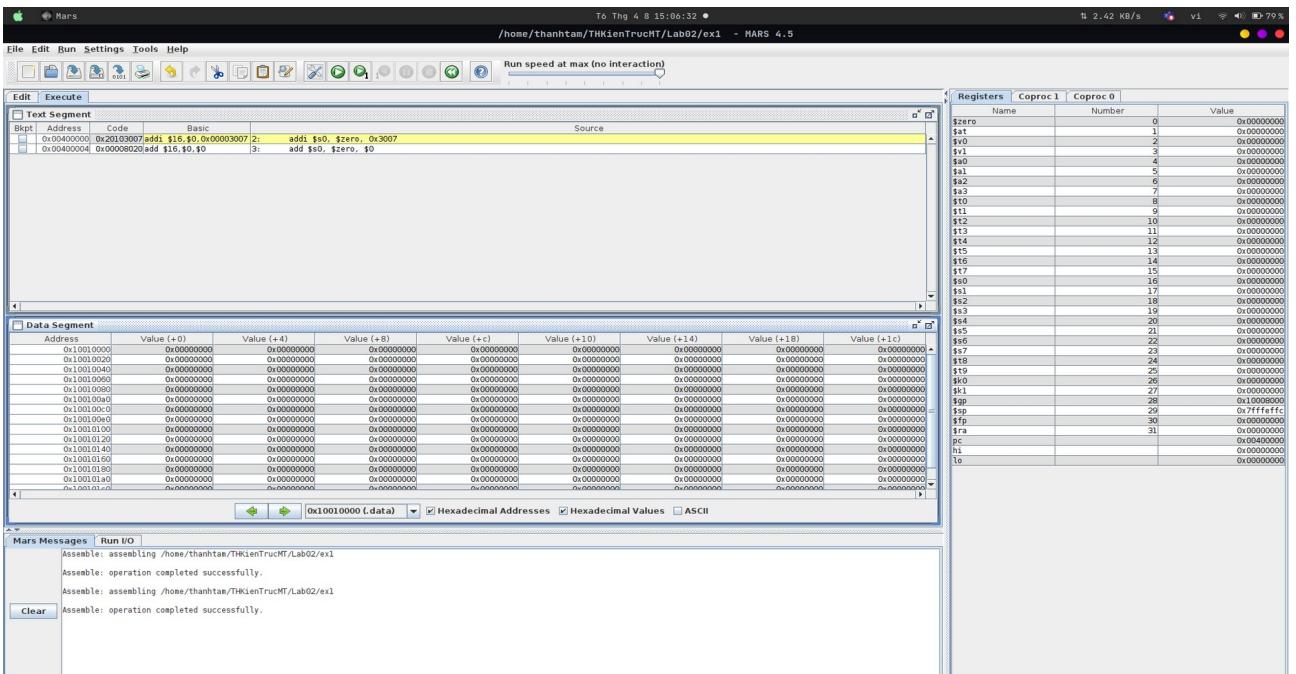
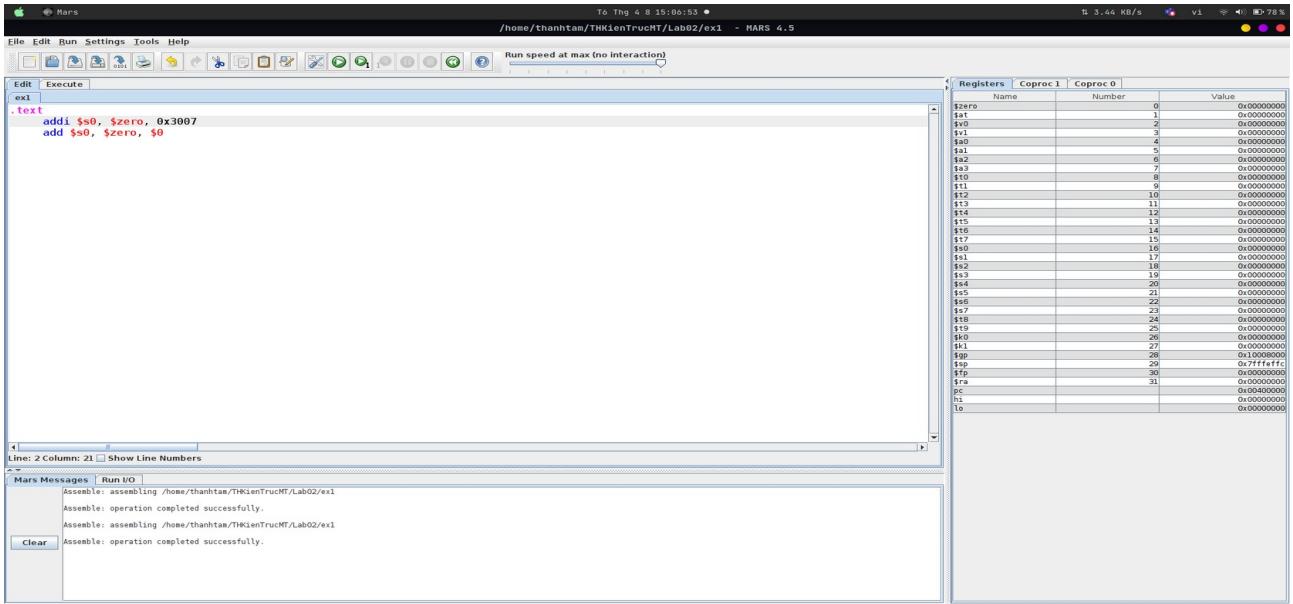


BÁO CÁO THỰC HÀNH LAB02

Bài 1:



The screenshot shows the MARS 4.5 assembly debugger interface. The top status bar displays the date and time (T6 Thg 4 8 15:16:59), the file path (/home/thanhtam/THKienTrucMT/Lab02/mips1.asm), and the version (MARS 4.5). The top menu bar includes File, Edit, Run, Settings, Tools, and Help. Below the menu is a toolbar with various icons for assembly operations. A progress bar indicates "Run speed at max (no interaction)".

Text Segment: This window shows the assembly code. The current instruction is highlighted in yellow: `add $16,$0,$0`. The assembly code is as follows:

```
0x00000000 0x20103007 addi $16,$0,0x000003007
0x00000004 0x00000802 add $16,$0,$0
```

Data Segment: This window displays memory starting at address 0x10000000. The first few entries are:

Address	Value (+0)	Value (+4)	Value (+8)	Value (+C)	Value (+10)	Value (+14)	Value (+18)	Value (+1C)
0x10000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10000004	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10000008	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1000000C	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Registers: This window lists the processor registers. The \$t0 register is highlighted in green.

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10008000
\$sp	29	0xfffffff0
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00000008
hi		0x00000000
lo		0x00000000

Mars Messages: The log shows three "Reset: reset completed." messages and one "program is finished running (dropped off bottom)" message.

```
Reset: reset completed.  
Reset: reset completed.  
Reset: reset completed.  
... program is finished running (dropped off bottom) ..
```

The screenshot shows the MARS 4.5 assembly debugger interface. The top status bar indicates the date and time (Tô Thg 4 8 15:15:32), the file path (/home/thanhtam/THKienTrucMT/Lab02/mips1.asm), and the version (MARS 4.5). The top menu bar includes File, Edit, Run, Settings, Tools, and Help. A toolbar with various icons is located above the main window.

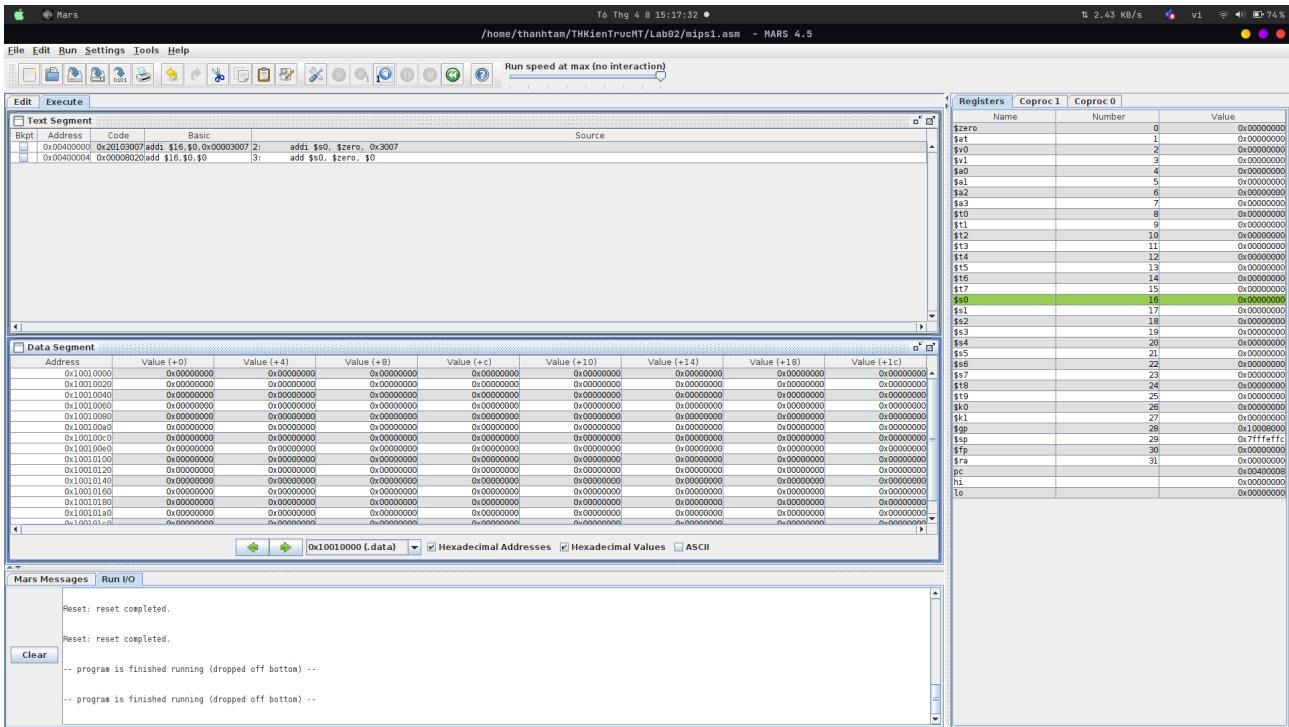
The main window displays the assembly code:

```
0x00000000 0x20130007 addi $16,$0,0x00000307
0x00000004 0x00000020 add $16,$0,$0
0x00000008 0x00000000 add $0,$0,$0
```

The Registers window shows the following register values:

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00003007
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$t1	27	0x00000000
\$gp	28	0x00008000
\$sp	29	0xfffffff1
\$fp	30	0x00000000
\$ra	31	0x00040004
pc		0x00000000
hi		0x00000000
lo		0x00000000

The bottom message window shows the message "Reset: reset completed." repeated three times, followed by "... program is finished running (dropped off bottom) ...".



*Sự thay đổi giá trị của thanh ghi:

Sự thay đổi giá trị của thanh ghi \$s0 khi chạy từng lệnh :
0x00000000 → 0x00003007 → 0x00000000

Sự thay đổi giá trị của thanh ghi \$pc khi chạy từng lệnh:

0x00400000 → 0x00400004 → 0x00400008

*So sánh mã máy của các lệnh:

addi \$s0, \$zero, 0x3007

op: 8

rs: \$zero

rt: \$s0

imm: 0x3007

0010 0000 0001 0000 0011 0000 0000 0111 => 0x20103007

add \$s0, \$zero, \$0

op:0

rd: \$s0

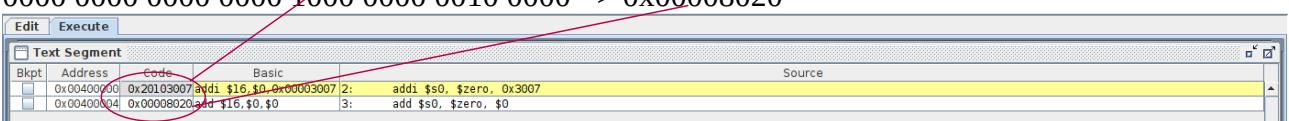
rs: \$zero

rt: \$0

shamt: 0

funct:32

0000 0000 0000 0000 1000 0000 0010 0000 => 0x00008020



*Sửa lại lệnh lui:

The screenshot shows the MARS 4.5 assembly editor interface. The assembly window displays the following code:

```
Text Segment
B0pt Address Code Basic
0x00000000 0x342100d0r1 $1,$0.0x0000003d 2: addi $s0, $zero, 0x2100003d
0x00000000 0x342100d0r1 $1,$0.0x0000003d
0x00000000 0x0000000000000000 3: add $s0, $zero, $0
```

The Registers window shows the following register values:

Name	Number	Value
\$r0	0	0x00000000
\$v0	1	0x00000000
\$v1	2	0x00000000
\$a0	3	0x00000000
\$a1	4	0x00000000
\$t0	5	0x00000000
\$t1	6	0x00000000
\$t2	7	0x00000000
\$t3	8	0x00000000
\$t4	9	0x00000000
\$t5	10	0x00000000
\$t6	11	0x00000000
\$t7	12	0x00000000
\$t8	13	0x00000000
\$t9	14	0x00000000
\$t10	15	0x00000000
\$t11	16	0x00000000
\$t12	17	0x00000000
\$t13	18	0x00000000
\$t14	19	0x00000000
\$t15	20	0x00000000
\$t16	21	0x00000000
\$t17	22	0x00000000
\$t18	23	0x00000000
\$t19	24	0x00000000
\$t20	25	0x00000000
\$t21	26	0x00000000
\$sp	27	0x10fffffc
\$gp	28	0x00000000
\$fp	29	0xffffffff
\$ra	30	0x00000000
pc	31	0x00000000
hi		0x00000000
lo		0x00000000

The Data Segment window shows memory dump starting at address 0x10010000.

Mars Messages window:

- Assemble: operation completed successfully.
- Assemble: assembling /home/thantam/THKienTrucMT/Lab02/mips1.asm
- Assemble: operation completed successfully.

Step: execution terminated due to null instruction.

Assemble: assembling /home/thantam/THKienTrucMT/Lab02/mips1.asm

Assemble: operation completed successfully.

The screenshot shows the Immunity Debugger interface with several windows open:

- Registers**: Shows CPU registers (r0-r31, rbp, rbp+1, rbp+2) with their names, numbers, and current values.
- Coproc 1** and **Coproc 0**: Shows floating-point registers (st0-st15) with their names, numbers, and current values.
- Text Segment**: Displays assembly code. The highlighted instruction is `addi $0, $zero, 0x2110003d` at address 0x00000000. The assembly pane includes columns for Address, Code, Basic, and Source.
- Data Segment**: Displays memory dump data across various segments (text, data, bss, etc.) with columns for Address, Value (+0), Value (+4), Value (+8), Value (+c), Value (+10), Value (+14), Value (+18), and Value (+1c).

The screenshot shows the Immunity Debugger interface with two main panes. The left pane displays assembly code for a specific memory location, with the instruction at address 0x00000000 highlighted in yellow. The right pane displays the CPU registers.

Registers	Coproc 1	Coproc 0
\$zero	0	0x00000000
\$at	1	0x211003d
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x211003d
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$a1	27	0x00000000
\$gp	28	0x01000000
\$sp	29	0x7fffffe
\$ra	30	0x00000000
\$pc	31	0x00000000
\$hi		0x00000000
\$lo		0x00000000

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x21100000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$t10	26	0x00000000
\$k1	27	0x00000000
\$sp	28	0x10000000
\$fp	29	0x7ffffefc
\$ra	30	0x00000000
pc	31	0x04000010
hi		0x00000000
lo		0x00000000

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x3c01210 lui \$1, 0x00002110	2:	addi \$s0, \$zero, 0x2110003d
	0x00400004	0x3421003d ori \$1, \$1, 0x0000003d	3:	add \$s0, \$s0, \$0

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010028	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010056	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010069	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001007C	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010090	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100A3	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100B6	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100C9	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100E0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100F3	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010106	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010119	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001012C	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010153	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Khi sửa lệnh lui, vì hằng số ở đây là 32 bit nên để thực hiện được lệnh addi thì phải tách thành hai lệnh basic là lui và ori.

Bài 2:

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t8	16	0x00000000
\$t9	17	0x00000000
\$k0	18	0x00000000
\$k1	19	0x00000000
\$sp	20	0x7ffffeff
\$fp	21	0x00000000
\$ra	22	0x00000000
pc	23	0x04000010
hi		0x00000000
lo		0x00000000

Text Segment

Bkpt	Address	Code	Basic	Source
	0x04000000	0x3c01210 lui \$1, 0x00002110	2:	lui \$s0, 0x2110
	0x04000004	0x3610003d ori \$16, \$16, 0x0000003d	3:	ori \$s0, 0x003d

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$t8	16	0x00000000
\$t9	17	0x00000000
\$k0	18	0x00000000
\$k1	19	0x00000000
\$sp	20	0x7ffffeff
\$fp	21	0x00000000
\$ra	22	0x00000000
pc	23	0x04000010
hi		0x00000000
lo		0x00000000

Text Segment

Bkpt	Address	Code	Basic	Source
	0x04000000	0x3c01210 lui \$1, 0x00002110	2:	lui \$s0, 0x2110
	0x04000004	0x3610003d ori \$16, \$16, 0x0000003d	3:	ori \$s0, 0x003d

Data Segment

Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x10010000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010028	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010056	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010069	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001007C	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010090	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100A3	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100B6	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100C9	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100E0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x100100F3	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010106	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010119	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x1001012C	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010140	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x10010153	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

*Chạy từng dòng lệnh và quan sát:

*Sự thay đổi của các thanh ghi:

\$s0: 0x00000000 → 0x21100000 → 0x2110003d

\$pc: 0x00400000 → 0x00400004 → 0x00400008

*Quan sát cửa sổ Data Segment:

Các byte đầu tiên ở vùng lệnh trùng với cột Code trong cửa sổ Text Segment.

Bài 3:

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$t1	1	0x2110003d
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x2110003d
\$s1	17	0x00000002
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$t10	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x0040000c
hi		0x00000000
lo		0x00000000

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$t1	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$t10	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x0040000c
hi		0x00000000
lo		0x00000000

Hàng số 0x2110003d là 32 bit nên lệnh li thứ nhất để thực hiện được cần phải tách thành 2 lệnh basic là lui và ori.

Bài 4:

Registers Coproc 1 Coproc 0

Name	Number	Value
\$zero	0	0x00000000
\$t1	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$t10	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffeffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x0040000c
hi		0x00000000
lo		0x00000000

The screenshot shows the Immunity Debugger interface with two main panes. The top pane, titled "Text Segment", displays assembly code. The bottom pane, titled "Data Segment", displays memory dump data. The registers pane is also visible on the right.

Registers	Name	Number	Value
\$zero	0		0x00000000
\$at	1		0x00000000
\$v0	2		0x00000000
\$v1	3		0x00000000
\$t0	4		0x00000000
\$t1	5		0x00000000
\$a2	6		0x00000000
\$a3	7		0x00000000
\$t0	8		0x00000000
\$t1	9		0x00000000
\$t2	10		0x00000000
\$t3	11		0x00000000
\$t4	12		0x00000000
\$t5	13		0x00000000
\$t6	14		0x00000000
\$t7	15		0x00000000
\$s0	16		0x00000000
\$s1	17		0x00000000
\$s2	18		0x00000000
\$s3	19		0x00000000
\$t8	20		0x00000000
\$t9	21		0x00000000
\$t10	22		0x00000000
\$t11	23		0x00000000
\$t12	24		0x00000000
\$t13	25		0x00000000
\$t0	26		0x00000000
\$t1	27		0x00000000
\$t2	28		0x00000000
\$sp	29		0x7f7ffefc
\$fp	30		0x00000000
\$ra	31		0x00000000
\$pc			0x00000000
\$hi			0x00000000
\$lo			0x00000000

*Chạy từng lệnh:

The screenshot shows the QEMU debugger interface with three main windows:

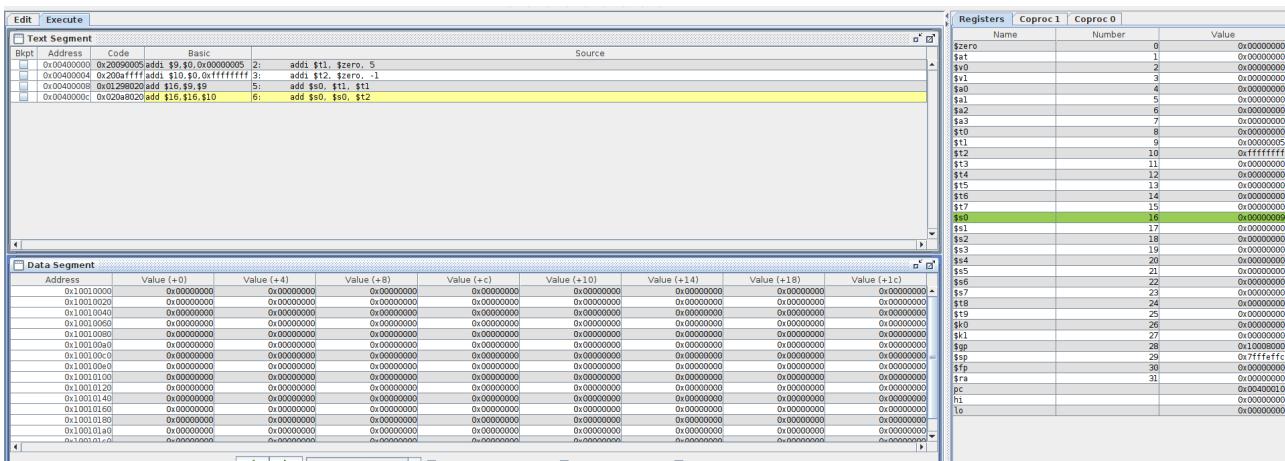
- Text Segment**: Displays assembly code in the CPU register window. The assembly instructions are as follows:
 - 0x04000000: addi \$10,\$0,0x00000005
 - 0x04000004: addi \$10,\$0,0xffffffff
 - 0x04000008: add \$10,\$0,\$9
 - 0x0400000c: add \$10,\$0,\$11
 - 0x04000010: add \$10,\$0,\$12
- Registers**: Shows CPU register values. The \$r10 register has a value of 0x00000000.
- Data Segment**: Shows memory dump starting at address 0x10000000. The first 16 bytes of memory are filled with the value 0x00000000.

Edit Execute

Text Segments

The screenshot shows the Immunity Debugger interface with two windows open:

- Assembly Window:** Shows assembly code starting at address 0x00400000. The code includes instructions like add \$1, \$zero, \$5 and add \$10, \$10, \$10.
- Data Segment Window:** Shows memory dump from address 0x00400000 to 0x00401000. It lists values for registers and memory locations, including \$t1 through \$t20, \$s1 through \$s10, and \$a1 through \$a10.



Kết quả bằng 9 → chạy đúng kết quả.

*Sự thay đổi của các thanh ghi:

\$t1 : 0x00000000 → 0x00000005

\$t2 : 0x00000000 → 0xffffffff

\$s0 : 0x00000000 → 0x0000000a → 0x00000009

*Kiểm nghiệm với khuôn mẫu của kiểu lệnh I:

addi \$9, \$0, 0x00000005

op: 8

rs:\$0

rt:\$9

imm: 0x00000005

0010 0000 0000 1001 0000 0000 0000 0101 => 0x20090005

addi \$10, \$0, 0xffffffff

op: 8

rs: \$0

rt: \$10

imm: 0xffffffff

0010 0000 0000 1010 1111 1111 1111 1111 => 0x200affff

*Kiểm nghiệm với khuôn mẫu của kiểu lệnh R:

add \$16, \$9, \$9

op:0

rs:\$9

rt:\$9

rd: \$16

sh:0

fn:32

0000 0001 0010 1001 1000 0000 0010 0000 => 0x01298020

add \$16, \$16, \$10

op:0

rs: \$16

rt: \$10

rd: \$16

sh:0

fn:32

0000 0010 0000 1010 1000 0000 0010 0000 => 0x020a8020

Bài 5:

Registers [Coproc 1 | Coproc 0]

Name	Number	Value
\$zero	0	0x00000000
\$at	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$t0	7	0x00000000
\$t1	8	0x00000000
\$t2	9	0x00000000
\$t3	10	0x00000000
\$t4	11	0x00000000
\$t5	12	0x00000000
\$t6	13	0x00000000
\$t7	14	0x00000000
\$t8	15	0x00000000
\$t9	16	0x00000000
\$s0	17	0x00000000
\$s1	18	0x00000000
\$s2	19	0x00000000
\$s3	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$sp	28	0x10000000
\$fp	29	0x7fffffeffc
\$ra	30	0x00000000
\$pc	31	0x00000000
\$t0	32	0x00000000
\$t1	33	0x00000000
\$t2	34	0x00000000
\$t3	35	0x00000000
\$t4	36	0x00000000
\$t5	37	0x00000000
\$t6	38	0x00000000
\$t7	39	0x00000000
\$t8	40	0x00000000
\$t9	41	0x00000000
\$s0	42	0x00000000
\$s1	43	0x00000000
\$s2	44	0x00000000
\$s3	45	0x00000000
\$s5	46	0x00000000
\$s6	47	0x00000000
\$s7	48	0x00000000
\$t8	49	0x00000000
\$t9	50	0x00000000
\$k0	51	0x00000000
\$k1	52	0x00000000
\$sp	53	0x10000000
\$fp	54	0x7fffffeffc
\$ra	55	0x00000000
\$pc	56	0x00000000
\$t0	57	0x00000000
\$t1	58	0x00000000
\$t2	59	0x00000000
\$t3	60	0x00000000
\$t4	61	0x00000000
\$t5	62	0x00000000
\$t6	63	0x00000000
\$t7	64	0x00000000
\$t8	65	0x00000000
\$t9	66	0x00000000
\$s0	67	0x00000000
\$s1	68	0x00000000
\$s2	69	0x00000000
\$s3	70	0x00000000
\$s5	71	0x00000000
\$s6	72	0x00000000
\$s7	73	0x00000000
\$t8	74	0x00000000
\$t9	75	0x00000000
\$k0	76	0x00000000
\$k1	77	0x00000000
\$sp	78	0x10000000
\$fp	79	0x7fffffeffc
\$ra	80	0x00000000
\$pc	81	0x00000000
\$t0	82	0x00000000
\$t1	83	0x00000000
\$t2	84	0x00000000
\$t3	85	0x00000000
\$t4	86	0x00000000
\$t5	87	0x00000000
\$t6	88	0x00000000
\$t7	89	0x00000000
\$t8	90	0x00000000
\$t9	91	0x00000000
\$s0	92	0x00000000
\$s1	93	0x00000000
\$s2	94	0x00000000
\$s3	95	0x00000000
\$s5	96	0x00000000
\$s6	97	0x00000000
\$s7	98	0x00000000
\$t8	99	0x00000000
\$t9	100	0x00000000
\$k0	101	0x00000000
\$k1	102	0x00000000
\$sp	103	0x10000000
\$fp	104	0x7fffffeffc
\$ra	105	0x00000000
\$pc	106	0x00000000
\$t0	107	0x00000000
\$t1	108	0x00000000
\$t2	109	0x00000000
\$t3	110	0x00000000
\$t4	111	0x00000000
\$t5	112	0x00000000
\$t6	113	0x00000000
\$t7	114	0x00000000
\$t8	115	0x00000000
\$t9	116	0x00000000
\$s0	117	0x00000000
\$s1	118	0x00000000
\$s2	119	0x00000000
\$s3	120	0x00000000
\$s5	121	0x00000000
\$s6	122	0x00000000
\$s7	123	0x00000000
\$t8	124	0x00000000
\$t9	125	0x00000000
\$k0	126	0x00000000
\$k1	127	0x00000000
\$sp	128	0x10000000
\$fp	129	0x7fffffeffc
\$ra	130	0x00000000
\$pc	131	0x00000000
\$t0	132	0x00000000
\$t1	133	0x00000000
\$t2	134	0x00000000
\$t3	135	0x00000000
\$t4	136	0x00000000
\$t5	137	0x00000000
\$t6	138	0x00000000
\$t7	139	0x00000000
\$t8	140	0x00000000
\$t9	141	0x00000000
\$s0	142	0x00000000
\$s1	143	0x00000000
\$s2	144	0x00000000
\$s3	145	0x00000000
\$s5	146	0x00000000
\$s6	147	0x00000000
\$s7	148	0x00000000
\$t8	149	0x00000000
\$t9	150	0x00000000
\$k0	151	0x00000000
\$k1	152	0x00000000
\$sp	153	0x10000000
\$fp	154	0x7fffffeffc
\$ra	155	0x00000000
\$pc	156	0x00000000
\$t0	157	0x00000000
\$t1	158	0x00000000
\$t2	159	0x00000000
\$t3	160	0x00000000
\$t4	161	0x00000000
\$t5	162	0x00000000
\$t6	163	0x00000000
\$t7	164	0x00000000
\$t8	165	0x00000000
\$t9	166	0x00000000
\$s0	167	0x00000000
\$s1	168	0x00000000
\$s2	169	0x00000000
\$s3	170	0x00000000
\$s5	171	0x00000000
\$s6	172	0x00000000
\$s7	173	0x00000000
\$t8	174	0x00000000
\$t9	175	0x00000000
\$k0	176	0x00000000
\$k1	177	0x00000000
\$sp	178	0x10000000
\$fp	179	0x7fffffeffc
\$ra	180	0x00000000
\$pc	181	0x00000000
\$t0	182	0x00000000
\$t1	183	0x00000000
\$t2	184	0x00000000
\$t3	185	0x00000000
\$t4	186	0x00000000
\$t5	187	0x00000000
\$t6	188	0x00000000
\$t7	189	0x00000000
\$t8	190	0x00000000
\$t9	191	0x00000000
\$s0	192	0x00000000
\$s1	193	0x00000000
\$s2	194	0x00000000
\$s3	195	0x00000000
\$s5	196	0x00000000
\$s6	197	0x00000000
\$s7	198	0x00000000
\$t8	199	0x00000000
\$t9	200	0x00000000
\$k0	201	0x00000000
\$k1	202	0x00000000
\$sp	203	0x10000000
\$fp	204	0x7fffffeffc
\$ra	205	0x00000000
\$pc	206	0x00000000
\$t0	207	0x00000000
\$t1	208	0x00000000
\$t2	209	0x00000000
\$t3	210	0x00000000
\$t4	211	0x00000000
\$t5	212	0x00000000
\$t6	213	0x00000000
\$t7	214	0x00000000
\$t8	215	0x00000000
\$t9	216	0x00000000
\$s0	217	0x00000000
\$s1	218	0x00000000
\$s2	219	0x00000000
\$s3	220	0x00000000
\$s5	221	0x00000000
\$s6	222	0x00000000
\$s7	223	0x00000000
\$t8	224	0x00000000
\$t9	225	0x00000000
\$k0	226	0x00000000
\$k1	227	0x00000000
\$sp	228	0x10000000
\$fp	229	0x7fffffeffc
\$ra	230	0x00000000
\$pc	231	0x00000000
\$t0	232	0x00000000
\$t1	233	0x00000000
\$t2	234	0x00000000
\$t3	235	0x00000000
\$t4	236	0x00000000
\$t5	237	0x00000000
\$t6	238	0x00000000
\$t7	239	0x00000000
\$t8	240	0x00000000
\$t9	241	0x00000000
\$s0	242	0x00000000
\$s1	243	0x00000000
\$s2	244	0x00000000
\$s3	245	0x00000000
\$s5	246	0x00000000
\$s6	247	0x00000000
\$s7	248	0x00000000
\$t8	249	0x00000000
\$t9	250	0x00000000
\$k0	251	0x00000000
\$k1	252	0x00000000
\$sp	253	0x10000000
\$fp	254	0x7fffffeffc
\$ra	255	0x00000000
\$pc	256	0x00000000
\$t0	257	0x00000000
\$t1	258	0x00000000
\$t2	259	0x00000000
\$t3	260	0x00000000
\$t4	261	0x00000000
\$t5	262	0x00000000
\$t6	263	0x00000000
\$t7	264	0x00000000
\$t8	265	0x00000000
\$t9	266	0x00000000
\$s0	267	0x00000000
\$s1	268	0x00000000
\$s2	269	0x00000000
\$s3	270	0x00000000
\$s5	271	0x00000000
\$s6	272	0x00000000
\$s7	273	0x00000000
\$t8	274	0x00000000
\$t9	275	0x00000000
\$k0	276	0x00000000
\$k1	277	0x00000000
\$sp	278	0x10000000
\$fp	279	0x7fffffeffc
\$ra	280	0x00000000
\$pc	281	0x00000000
\$t0	282	0x00000000
\$t1	283	0x00000000
\$t2	284	0x00000000
\$t3	285	0x00000000
\$t4	286	0x00000000
\$t5	287	0x00000000
\$t6	288	0x00000000
\$t7	289	0x00000000
\$t8	290	0x00000000
\$t9	291	0x00000000
\$s0	292	0x00000000
\$s1	293	0x00000000
\$s2	294	0x00000000
\$s3	295	0x00000000
\$s5	296	0x00000000
\$s6	297	0x00000000
\$s7	298	0x00000000
\$t8	299	0x00000000
\$t9	300	0x00000000
\$k0	301	0x00000000
\$k1	302	0x00000000
\$sp	303	0x10000000
\$fp	304	0x7fffffeffc
\$ra	305	0x00000000
\$pc	306	0x00000000
\$t0	307	0x00000000
\$t1	308	0x00000000
\$t2	309	0x00000000
\$t3	310	0x00000000
\$t4	311	0x00000000
\$t5	312	0x00000000
\$t6	313	0x00000000
\$t7	314	0x00000000
\$t8	315	0x00000000
\$t9	316	0x00000000
\$s0	317	0x00000000
\$s1	318	0x00000000
\$s2	319	0x00000000
\$s3	320	0x00000000
\$s5	321	0x00000000
\$s6	322	0x00000000
\$s7	323	0x00000000
\$t8	324	0x00000000
\$t9	325	0x00000000
\$k0	326	0x00000000
\$k1	327	0x00000000
\$sp	328	0x10000000
\$fp	329	0x7fffffeffc
\$ra	330	0x00000000
\$pc	331	0x00000000
\$t0	332	0x00000000

The screenshot shows the Immunity Debugger interface with three main panes:

- Assembly Pane:** Displays assembly code for the current program. The code includes instructions like add, mul, and mflo. A yellow highlight covers the instruction at address 0x04000008, which is a multiplication operation.
- Registers Pane:** Shows the CPU registers. The \$t2 register is highlighted in green and contains the value 0x00000005.
- Data Segment Pane:** Displays memory dump data starting at address 0x10000000.

The screenshot shows the QEMU debugger interface with two main windows:

- Text Segment**: Displays assembly code for the basic section. The code includes additions and multiplications of \$1.0 and \$16.0, followed by an mflo instruction.
- Data Segment**: Displays a memory dump of the .data section, showing various memory locations filled with zeros.

The screenshot shows the Immunity Debugger interface with three main panes:

- Registers** pane: Shows CPU registers (r0 to r31) with their names, numbers, and current values.
- Registers** pane: Shows Coprocessor registers (COP0 to COP2) with their names, numbers, and current values.
- Data Segment** pane: Shows memory dump and search features for the current assembly context.

Text Segment				Registers	Coproc 1	Coproc 0
Blpt	Address	Code	Base	Name	Number	Value
	0x00400000	0x40000000 addi \$8, \$zero, 4	0: 0x0000000000000004	\$zero	0	0x00000000
	0x00400004	0x20000000 addi \$10, \$0, 0x00000005	2: 0x0000000000000005	\$at	1	0x00000000
	0x00400008	0x71208002 mul \$1, \$10	3: 0x0000000000000005	\$v0	2	0x00000000
	0x0040000c	0x20012000 addi \$1, \$0, 0x00000001	5: 0x0000000000000001	\$v1	3	0x00000000
	0x00400000	0x20010003 addi \$1, \$0, 0x00000003	6: 0x0000000000000003	\$v0	4	0x00000000
	0x00400010	0x72018002 mul \$16, \$1	mul \$50, \$0, 3	\$v1	5	0x00000000
	0x00400014	0x00008812 mflo \$17	8: 0x0000000000000001	\$v2	6	0x00000000
				\$a3	7	0x00000000
				\$t0	8	0x00000000
				\$t1	9	0x00000000
				\$t2	10	0x00000005
				\$t3	11	0x00000000
				\$t4	12	0x00000000
				\$t5	13	0x00000000
				\$t6	14	0x00000000
				\$t7	15	0x00000000
				\$s0	16	0x00000003
				\$s1	17	0x00000000
				\$s2	18	0x00000000

* Sự thay đổi của các thanh ghi:

\$t1 : 0x00000000 → 0x00000004

\$t2: 0x00000000 → 0x00000005

\$lo: 0x00000000 → 0x00000014 → 0x0000003c

\$at: 0x00000000 → 0x00000003

\$s0: 0x00000000 → 0x0000003c

\$s1: 0x00000000 → 0x0000003c

Thanh ghi hi không thay đổi giá trị vì kết quả dưới 32 bit, kết quả ở đây được ghi vào thanh ghi lo.

Bài 6:

Name	Number	Value
\$zero	0	0x00000000
\$t1	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

Name	Number	Value
\$zero	0	0x00000000
\$t1	1	0x00000000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x00000000
\$t9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

* Lệnh la được biên dịch bằng cách tách thành 2 lệnh basic là lui và ori

* Ở cửa sổ Data Segment :

Địa chỉ của X = \$t8

Địa chỉ của Y = \$t9

Địa chỉ của Z = \$t7

Tương ứng với hằng số khi biên dịch lệnh la thành mã máy.

Name	Number	Value
\$zero	0	0x00000000
\$t1	1	0x10010000
\$v0	2	0x00000000
\$v1	3	0x00000000
\$a0	4	0x00000000
\$a1	5	0x00000000
\$a2	6	0x00000000
\$a3	7	0x00000000
\$t0	8	0x00000000
\$t1	9	0x00000005
\$t2	10	0xffffffff
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x10010008
\$s0	16	0x00000009
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$t8	24	0x10010000
\$t9	25	0x10010004
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x10000000
\$sp	29	0x7fffffc
\$fp	30	0x00000000
\$ra	31	0x00000000
pc		0x00400002
hi		0x00000000
lo		0x00000000

Label	Address
X	0x10010000
Y	0x10010004
Z	0x10010008

Data Text

Bkpt	Address	Code	Basic	Source
	0x04000000	0x3c010001 lui \$1, \$00001001	7: la \$t8, X	
	0x04000004	0x34380000 ori \$24, \$1, \$00000000	8: la \$t9, Y	
	0x04000008	0x34380000 ori \$24, \$1, \$00000000	9: lw \$t1, 0(\$t8)	
	0x04000012	0x01298020 add \$16, \$9, \$9	10: lw \$t2, 0(\$t9)	
	0x04000016	0x020a0020 add \$16, \$16, \$10	12: add \$s0, \$t1, \$t1	
	0x04000020	0x3c010001 lui \$1, \$00001001	13: add \$s0, \$s0, \$12	
	0x04000024	0x342f0008 ori \$15, \$1, \$00000008	15: la \$t7, Z	
	0x04000028	0xadff0000 sw \$16, \$00000000(\$15)	16: sw \$s0, 0(\$t7)	

*Chạy từng dòng lệnh và quan sát:

The screenshot shows the Mars 32-bit CPU Emulator interface. On the left, the **Text Segment** window displays assembly code with specific lines highlighted in yellow. The highlighted lines are:

```

    7: la $t8, X
    8: la $t9, Y
    9: lw $t1, 0($t8)
    10: lw $t2, 0($t9)
    11: add $t0, $t1, $t2
    12: add $t0, $t0, $t0
    13: add $t0, $t0, $t0
    14: la $t7, Z
    15: sw $t0, 0($t7)
  
```

The **Registers** window on the right shows the state of the processor registers. The **Labels** window in the center lists labels and their addresses.

Name	Number	Value
\$zero	0	0x00000000
\$t0	1	0x10000000
\$t1	2	0x00000000
\$t2	3	0x00000000
\$t3	4	0x00000000
\$t4	5	0x00000000
\$t5	6	0x00000000
\$t6	7	0x00000000
\$t7	8	0x00000000
\$t8	9	0x00000000
\$t9	10	0x00000000
\$t10	11	0x00000000
\$t11	12	0x00000000
\$t12	13	0x00000000
\$t13	14	0x00000000
\$t14	15	0x00000000
\$t15	16	0x00000000
\$t16	17	0x00000000
\$t17	18	0x00000000
\$t18	19	0x00000000
\$t19	20	0x00000000
\$t20	21	0x00000000
\$t21	22	0x00000000
\$t22	23	0x00000000
\$t23	24	0x00000000
\$t24	25	0x00000000
\$t25	26	0x00000000
\$t26	27	0x00000000
\$t27	28	0x00000000
\$t28	29	0x7f11fffc
\$t29	30	0x00000000
\$t30	31	0x00000000
pc		0x00400004
hi		0x00000000
lo		0x00000000

This screenshot shows the same assembly code and register state as the previous one, but with different memory dump results. The **Data Segment** window now shows the memory starting at address 0x10000000. The first few bytes are 0x00000005 followed by 0xffffffff.

This screenshot shows the assembly code and register state again, but with a significant change in the memory dump. The **Data Segment** window shows the memory starting at address 0x10000000. The first few bytes are now 0x00000005 followed by 0x00000000. This indicates a write operation has occurred at address 0x10000000.

The screenshot shows the QEMU debugger interface with several windows open:

- Text Segment**: Displays assembly code from address 0x00400000 to 0x0040028. The code includes instructions like lui, la, add, and sw.
- Registers**: Shows the CPU registers (r0-r31, fp, pc) and their values.
- Labels**: Shows labels and their addresses, such as mips6.asm and various \$z labels.
- Data Segment**: Displays memory dump information for address 0x10010000, including raw hex data and ASCII representation.

The screenshot shows the QEMU debugger interface with several windows open:

- Registers**: Shows registers r0 to r31, all initialized to 0.
- Labels**: Shows labels defined in the mips6.asm file, including \$t0 through \$t9.
- Text Segment**: Shows assembly code with addresses from 0x00000000 to 0x00000028. The code includes instructions like la, add, and sw.
- Data Segment**: Shows memory starting at address 0x10010000 with various values assigned to different memory locations.

The screenshot displays the Mars Simulation Environment interface with several windows open:

- Text Segment**: Shows assembly code for the mips6.asm file. The code includes instructions like la, add, and sw.
- Registers**: Shows the state of various registers (r0 to r31, \$t0 to \$t9, \$k0 to \$k1, \$pc, \$n1, \$lo) with their names, numbers, and values.
- Data Segment**: Shows memory dump starting at address 0x00010000, displaying values for memory locations from 0x00010000 to 0x0001003f.
- Mars Messages**: Displays two log entries indicating the program has finished running.

The figure shows a screenshot of the QEMU debugger interface, specifically the GDB-like monitor. It consists of two main windows side-by-side.

Left Window (mips6 assembly):

- Text Segment:** Shows assembly code with labels X, Y, Z, and various memory locations. The assembly includes instructions like `la`, `lw`, `add`, and `sw`.
- Data Segment:** Shows memory dump from address 0x10010000 to 0x1001000f. It lists values for each byte, including 0xffffffff at address 0x10010005.

Right Window (Registers and Coprocessors):

- Registers:** Shows the state of general-purpose registers (\$zero-\$#) and floating-point registers (\$fp-\$lo).
- Coproc 1:** Shows the state of coprocessor 1 registers (\$at-\$t0).
- Coproc 0:** Shows the state of coprocessor 0 registers (\$t1-\$t9).

Mars Messages and Run I/O:

- Mars Messages:** Displays the message "... program is finished running (abnormal exit)." and the command `q`.
- Run I/O:** Buttons for running the program.

The screenshot shows the QEMU debugger interface with four main panes:

- Text Segment:** Displays assembly code for the `mips64.asm` file. The code includes instructions like `la`, `lw`, `add`, and `sw`. A yellow selection highlights the instruction at address `0x00400028`.
- Labels:** Shows a list of labels and their addresses. Labels include `X`, `Y`, `Z`, `lui`, `sw`, `add`, `lw`, `la`, `add`, and `sw`.
- Registers:** Displays the CPU registers. The `zero` register is at address 0. Other registers range from \$t0 to \$t14.
- Data Segment:** Displays memory dump segments for `l.data`, `l.bss`, and `l.heap`. It shows memory addresses from `0x10010000` to `0x10030000` with values corresponding to the assembly code.

The screenshot shows the following windows:

- Text Segment**: Displays assembly code with columns for Blkpt, Address, Code, Basic, and Source. The source code includes instructions like `la $t8, X`, `la $t9, Y`, `lw $t1, 0($t8)`, `lw $t2, 0($t9)`, `add $t0, $t1, $t2`, and `sw $s0, 0($t7)`.
- Data Segment**: Shows memory starting at address 0x10010000 with various data values.
- Labels**: Lists labels from the assembly code: `mips6.asm`, `X`, `Y`, `Z`, `$zero`, `$t0`, `$t1`, `$t2`, `$t3`, `$t4`, `$t5`, `$t6`, `$t7`, `$t8`, `$t9`, `$t10`, `$t11`, `$t12`, `$t13`, `$t14`, `$t15`, `$t16`, `$t17`, `$t18`, `$t19`, `$t20`, `$t21`, `$t22`, `$t23`, `$t24`, `$t25`, `$t26`, `$t27`, `$t28`, `$t29`, `$t30`, `$t31`, `$r0`, `pc`, and `hi`.
- Registers**: Shows the state of registers \$zero through \$t31, \$r0, pc, and hi.

The screenshot shows the same interface as the first one, but with the assembly code modified to include `lui $t1, 0x00000001` before the `lw` instructions. The labels and register values remain identical.

*Sự thay đổi giá trị các thanh ghi :

\$at : 0x00000000 → 0x10010000

\$t8 : 0x00000000 → 0x10010000

\$t9 : 0x00000000 → 0x10010004

\$t1 : 0x00000000 → 0x00000005

\$t2 : 0x00000000 → 0xffffffff

\$s0 : 0x00000000 → 0x0000000a → 0x00000009

\$t7 : 0x00000000 → 0x10010008

- Vai trò của lệnh lw và sw :

lw \$rt, imm(\$rs) : gán giá trị của thanh ghi \$rs vào thanh ghi \$rt (\$rt = M[\$rs+imm])

ở đây lw gán \$t1 = \$t8 và \$t2 = \$t9

sw \$rt, imm(\$rs) : gán giá trị của thanh ghi \$rt vào thanh ghi \$rs (M[\$rs+imm] = \$rt)

- Các lệnh lb, sb:

lb : chép 1 byte tại vị trí trong bộ nhớ RAM vào byte thấp của thanh ghi.

sb : lưu một byte thấp trong thanh ghi vào vị trí trong bộ nhớ RAM.