

# BÁO CÁO THỰC HÀNH TUẦN 1

## Assignment 1

- Lệnh `addi $s0, $zero, 0x3007`

The screenshot shows the Mars MIPS simulator interface. The **Text Segment** window displays the instruction `addi $s0, $zero, 0x3007` at address `0x00400004`. The **Data Segment** window shows memory addresses from `0x10010000` to `0x10010120`. The **Registers** window shows the state of registers, with `$s0` set to `12295` and `$pc` set to `4194308`.

⇒ `$s0 = 12295` , `$pc = 4194308`

- Lệnh `add $s0, $zero, $0`

The screenshot shows the Mars MIPS simulator interface. The **Text Segment** window displays the instruction `add $s0, $zero, $0` at address `0x00400004`. The **Data Segment** window shows memory addresses from `0x10010000` to `0x10010120`. The **Registers** window shows the state of registers, with `$s0` set to `0` and `$pc` set to `4194312`.

⇒ `$s0 = 0` , `$pc = 4194312`

So sánh mã máy:

addi \$s0, \$zero, 0x3007

- Format : I
- Op 8 : 00 0000
- imm 0x3007 : 0011 0000 0000 0111
- rs \$0 : 0 0000
- rt \$16: 1 0000
- ➔ Mã máy : 0010 0000 0001 0000 0011 0000 0000 0111
- ➔ Mã rút gọn : 0x20103007
- ➔ Trùng với khuôn dạng lệnh -> chương trình chạy đúng

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

- Format : R
- Op 0 : 00 1000
- rs 0 : 0 0000
- rt 0: 0 0000
- rd 16 : 1 0000
- shamt: 0 0000
- function 32 : 10 0000
- ➔ Mã máy : 0010 0000 0000 0000 1000 0000 0010 0000
- ➔ Mã rút gọn : 0x20008020
- ➔ Trùng với khuôn dạng lệnh -> chương trình chạy đúng

**Sau khi sửa lệnh addi \$s0, \$zero, 0x2110003d và chạy thì được chia thành các lệnh sau:**

- Lệnh thứ 1 : lui \$1,0x00002110
- Lệnh thứ 2 : ori \$1,\$1,0x0000003d
- Lệnh thứ 3 : add \$16,\$0,\$1
- Lệnh 4 như cũ: add \$16,\$0,\$0
- ➔ Vì lệnh lui là 32 bit : addi \$s0, \$zero, 0x2110003d nên phải chia thành 2 lệnh 16 bit là lui \$s0,0x2110 và ori \$s0,0x003d

## Assignment 2

- Khi chưa chạy lệnh nào:

\$s0 = 0x00000000 , \$pc = 0x00400000

- Khi chạy lệnh lui \$16,0x00002110

\$s0 = 0x21100000 , \$pc= 0x00400004

- Khi chạy lệnh ori \$16,\$16,0x0000003d

\$s0 = 0x2110003d, \$pc = 0x00400008

Data Segment								
Address	Value (+0)	Value (+4)	Value (+8)	Value (+c)	Value (+10)	Value (+14)	Value (+18)	Value (+1c)
0x00400000	0x3c102110	0x3610003d	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400020	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400040	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400060	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400080	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000a0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000c0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x004000e0	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400100	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000
0x00400120	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000	0x00000000

Lệnh lui \$s0,0x2110 : Xuất hiện ở cột Value( +0) tại thanh ghi có địa chỉ 0x3c102110

Lệnh ori \$s16,\$s16, 0x003d : Xuất hiện ở cột Value( +4) tại thanh ghi có địa chỉ 0x3610003d

## Assignment 3

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x3c012110	lui \$1,0x00002110	3: li \$s0,0x2110003d #pseudo instruction=2 basic instructions
<input type="checkbox"/>	0x00400004	0x3430003d	ori \$16,\$1,0x0000003d	
<input type="checkbox"/>	0x00400008	0x24110002	addiu \$17,\$0,0x0000...	4: li \$s1,0x2 #but if the immediate value is small, one ins

- Điều bất thường của lệnh trên xuất hiện ở lệnh li \$s1,0x2  
Vì lệnh li là 1 lệnh mở rộng nên chương trình phải đưa về lệnh cơ bản là addiu \$16,\$1,0x0000003d

## Assignment 4

### Sự thay đổi giá trị của các thanh ghi

- Lệnh 1: `addi $9,$0,0x00000005`  
Thanh ghi `$t1 = 0x00000005 = 5`
- Lệnh 2: `addi $10,$0,0xffffffff`  
Thanh ghi `$t2 = 0xffffffff = -1`
- Lệnh 3 : `add $16,$9,$9`  
Thanh ghi `$s0 = 0x0000000a = 10`
- Lệnh 4: `add $16,$16,$10`  
Thanh ghi `$s0 = 0x00000009 = 9`  
$$= 2\$t1 + \$t2$$
  
 $\Rightarrow$  Kết quả chương trình là đúng

Sự tương đồng giữa hợp ngữ và mã máy trong lệnh `addi`:

+ 16bits sau của phần mã máy giống với giá trị của lệnh `addi` thêm vào thanh ghi.

+ Vì kiểu lệnh I chỉ có thể thực hiện được với giá trị < 16bits

Kiểm nghiệm với khuôn mẫu kiểu lệnh I:

`0x20090005`: 0010 00-00 000-0 1001- 0000 0000 0000 0101

`0x200affff`: 1000 00-00 001-0 1011- 1111111111111111

Kiểm nghiệm với khuôn mẫu kiểu lệnh R:

`0x01298020`: 0000 00-01 001-0 1001- 1000 0-000 00-10 0000

`0x020a8020`: 0000 00-10 000-0 1010- 1000 0-000 00-10 0000

## Assignment 5

- Chạy lệnh `mul $16,$9,$10`

**Text Segment**

Expr	Address	Code	Basic	Source
0x00400000	0x20090004	addi \$t1, \$zero, 4	\$t1 = ?	4: addi \$t1, \$zero, 4 # X = \$t1 = ?
0x00400004	0x200a0005	addi \$t2, \$zero, 5	\$t2 = ?	5: addi \$t2, \$zero, 5 # Y = \$t2 = ?
0x00400008	0x712a0002	mul \$s0, \$t1, \$t2	\$s0 = 10	7: mul \$s0, \$t1, \$t2 # HI-LO = \$t1 * \$t2 = X * Y ; \$s0 = 10
0x0040000c	0x20010003	addi \$t1, \$zero, 3	\$t1 = ?	8: addi \$t1, \$zero, 3 # X = \$t1 = ?
0x00400010	0x72018002	mul \$s0, \$t1, \$t2	\$s0 = 30	8: mul \$s0, \$t1, \$t2 # HI-LO = \$t1 * \$t2 = X * Y ; \$s0 = 30
0x00400014	0x00080112	mflo \$s1	\$s1 = 20	10: mflo \$s1

**Registers**

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	0x00000004
\$t2	10	0x00000005
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x00000014
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x00000000
\$ap	29	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400014
\$hi		0x00000000
\$lo		0x00000014

**Mars Messages**

```

Reset: reset completed.
Reset: reset completed.
Reset: reset completed.

```

hi= 0x00000000, lo = 0x00000014 = 20

- Chạy lệnh mul \$16,\$16,\$1

**Text Segment**

Expr	Address	Code	Basic	Source
0x00400000	0x20090004	addi \$t1, \$zero, 4	\$t1 = ?	4: addi \$t1, \$zero, 4 # X = \$t1 = ?
0x00400004	0x200a0005	addi \$t2, \$zero, 5	\$t2 = ?	5: addi \$t2, \$zero, 5 # Y = \$t2 = ?
0x00400008	0x712a0002	mul \$s0, \$t1, \$t2	\$s0 = 10	7: mul \$s0, \$t1, \$t2 # HI-LO = \$t1 * \$t2 = X * Y ; \$s0 = 10
0x0040000c	0x20010003	addi \$t1, \$zero, 3	\$t1 = ?	8: addi \$t1, \$zero, 3 # X = \$t1 = ?
0x00400010	0x72018002	mul \$s0, \$t1, \$t2	\$s0 = 30	8: mul \$s0, \$t1, \$t2 # HI-LO = \$t1 * \$t2 = X * Y ; \$s0 = 30
0x00400014	0x00080112	mflo \$s1	\$s1 = 20	10: mflo \$s1

**Registers**

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	0x00000004
\$t2	10	0x00000005
\$t3	11	0x00000000
\$t4	12	0x00000000
\$t5	13	0x00000000
\$t6	14	0x00000000
\$t7	15	0x00000000
\$s0	16	0x0000003c
\$s1	17	0x00000000
\$s2	18	0x00000000
\$s3	19	0x00000000
\$s4	20	0x00000000
\$s5	21	0x00000000
\$s6	22	0x00000000
\$s7	23	0x00000000
\$s8	24	0x00000000
\$s9	25	0x00000000
\$k0	26	0x00000000
\$k1	27	0x00000000
\$gp	28	0x00000000
\$ap	29	0x7fffffc0
\$fp	30	0x00000000
\$ra	31	0x00000000
\$pc		0x00400014
\$hi		0x00000000
\$lo		0x0000003c

**Mars Messages**

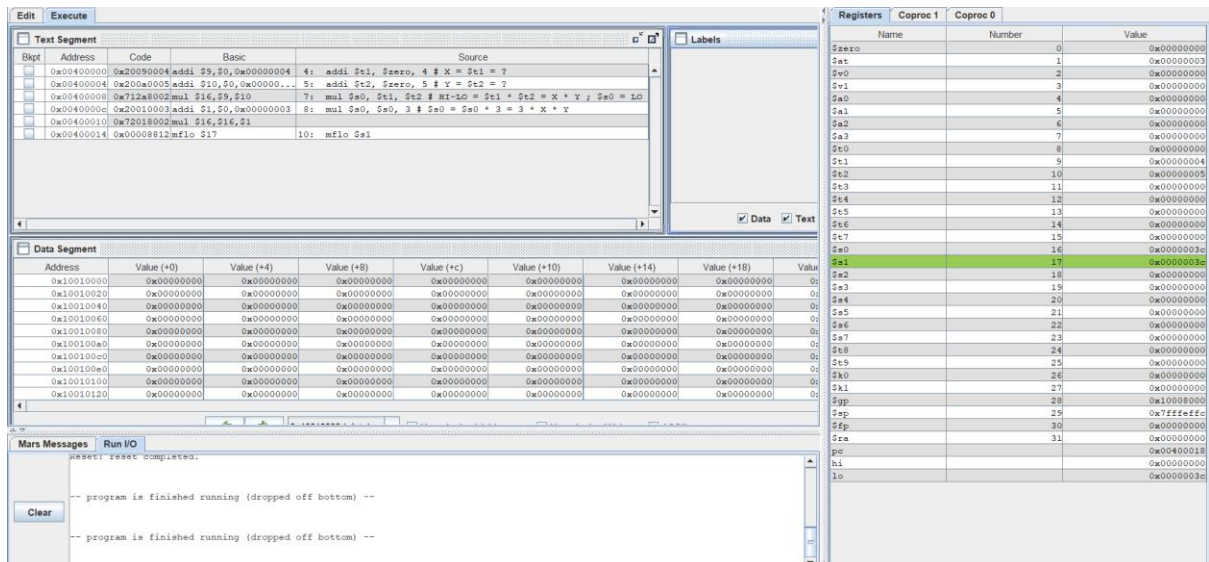
```

Reset: reset completed.
Reset: reset completed.
Reset: reset completed.

```

hi= 0x00000000, lo = 0x0000003c = 60

- Chạy lệnh mflo \$17

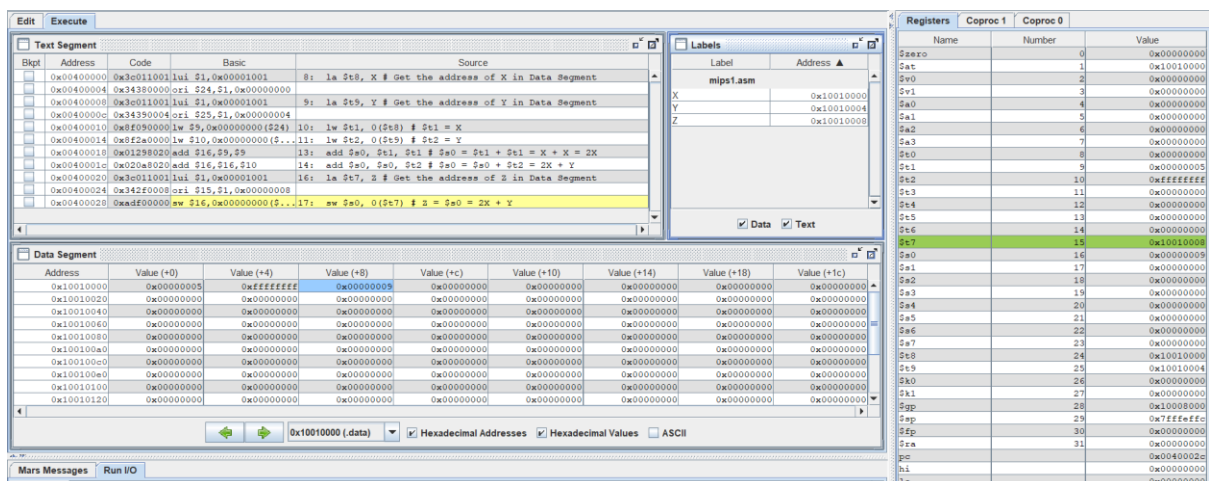


\$s1 = 0x0000003c = 60

- ⇒ Kết quả chương trình chạy đúng
- Điều bất thường là khi sử dụng mul, ta không thể dùng hằng số mà cần phải sử dụng thêm lệnh `addi $1, $0, 0x00000003` để ghi giá trị 3 cho thanh ghi \$at

## Assignment 6

- Lệnh la được biên dịch thành 2 lệnh : `lui $1, 0x00001001` và `ori $24, $1, 0x00000000`
- Ở cửa sổ Label, địa chỉ của X, Y, Z đúng với các giá trị khởi tạo



Vai trò của lệnh `lw`:

Chép 1 word (4bytes) tại vị trí trong bộ nhớ ram vào thanh ghi

Vai trò của lệnh sw:

Lưu 1 word trong thanh ghi vào bộ nhớ RAM

Lệnh lb và sb:

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

+ sb: Lưu 1 byte thấp trong thanh ghi vào vị trí trong bộ nhớ RAM

The screenshot displays a MIPS simulator interface with the following components:

- Text Segment:** A table of assembly instructions with columns for Dispt, Address, Code, Basic, and Source. The instructions include `addi $t1, $zero, 4`, `addi $t2, $zero, 5`, `mul $s0, $t1, $t2`, `addi $t1, $t0, 3`, and `mflw $s1`.
- Data Segment:** A table showing memory addresses and their corresponding values for various registers and memory locations.
- Registers:** A table listing registers (e.g., \$zero, \$t1, \$t2, \$s0, \$s1) and their current values.
- Mips Messages:** A log showing messages such as "Reset: reset completed."