

Họ và tên: Bùi Quang Hưng

MSSV: 20225849

Week 2

Assignment 1:

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 1
.text
    addi    $s0, $zero, 0x3007 # $s0 = 0 + 0x3007 = 0x3007 ;I-type
    add     $s0, $zero, $0      # $s0 = 0 + 0 = 0 ;R-type
```

Sau đó:

- Sử dụng công cụ gỡ lỗi, chạy từng lệnh và dừng lại,
- Ở mỗi lệnh, quan sát cửa sổ Registers và chú ý
 - o Sự thay đổi giá trị của thanh ghi \$s0
 - o Sự thay đổi giá trị của thanh ghi pc
- Ở cửa sổ Text Segment, hãy so sánh mã máy của các lệnh trên với khuôn dạng lệnh để chứng tỏ các lệnh đó đúng như tập lệnh đã qui định.
- Sửa lại lệnh addi như bên dưới. Chuyện gì xảy ra sau đó. Hãy giải thích

```
addi    $s0, $zero, 0x2110003d
```

Bài làm

```
1 #Laboratory Exercise 2, Assignment 1
2 .text
3     addi    $s0, $zero, 0x3007      # $s0 = 0 + 0x3007 ; I-type
4     add     $s0, $zero, $0          # $s0 = 0 + 0 ; R-type
```

Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x20103007	addi \$16,\$0,0x00003007	3: addi \$s0, \$zero, 0x3007 # \$s0 = 0 + 0x3007 ; I-type
<input type="checkbox"/>	0x00400004	0x00008020	add \$16,\$0,\$0	4: add \$s0, \$zero, \$0 # \$s0 = 0 + 0 ; R-type

- Trước khi chạy:

\$s0	16	0x00000000
pc		0x00400000

- Sau khi chạy lần 1:

\$s0	16	0x00003007
pc		0x00400004

+ \$s0 từ 0x00000000 thành 0x00003007

+ pc từ 0x00400000 thành 0x00400004

- Sau khi chạy lần 2:

\$s0	16	0x00000000
pc		0x00400008

- So sánh mã máy:

+ Lệnh thứ nhất:

addi \$16, \$0, 0x3007

I – type

opcode: 8=> 001000

rs: \$16 => 10000

rt: \$0 => 00000

imm: 0x3007 => 0011 0000 0000 0111

➔0010 0000 0001 0000 0011 0000 0000 0111 : 0x20103007

+ Lệnh thứ 2:

add \$16, \$0, \$0

R-type

opcode: 000000

rs: \$0 =>00000

rt: \$0 => 00000

rd:&16 =>10000

shamt: 0 =>00000

fn: 32 = > 100000

➔0000 0000 0000 0000 1000 0000 0010 0000 : 0x00008020

- Sau khi sửa lại lệnh addi ta thấy:


+ Đầu tiên giá trị của thanh ghi \$at bị thay đổi trước từ 0x00000000 thành 0x2110003d

+ Giải thích: Do thanh ghi chỉ có thể chứa 32 bit, mà số 0x2110003d là số 32 bit nên không thể chứa hết trong 1 câu lệnh mà phải tách ra làm 3, gán nửa đầu số vào \$at, gán nửa sau số vào \$at, gán \$at cho \$s0.

Assignment 2:

```
.text
lui    $s0,0x2110      #put upper half of pattern in $s0
ori    $s0,$s0,0x003d  #put lower half of pattern in $s0
```

Sau đó:

- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại, 
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
 - o Sự thay đổi giá trị của thanh ghi \$s0
 - o Sự thay đổi giá trị của thanh ghi \$pc
- Ở cửa sổ Data Segment, hãy click vào hộp combo để chuyển tới quan sát các byte trong vùng lệnh .text.
 - o Kiểm tra xem các byte đầu tiên ở vùng lệnh trùng với cột nào trong cửa sổ Text Segment.

Bài làm

```
.text
lui $s0,0x2110 #put upper half of pattern in $s0
ori $s0,$s0,0x003d #put lower half of pattern in $s0
```

<input type="checkbox"/>	0x00400000	0x3c102110	lui \$16,0x00002110	3:	lui \$s0, 0x2110
<input type="checkbox"/>	0x00400004	0x3610003d	ori \$16,\$16,0x0000003d	4:	ori \$s0, \$s0, 0x003d

- Sau khi chạy từng lệnh, ở mỗi lệnh ta quan sát cửa sổ Register và thấy rằng:
+ Trước khi chạy:

\$s0	16	0x00000000
pc		0x00400000

- + Sau khi chạy lần 1:

\$s0	16	0x21100000
pc		0x00400004

- + Sau khi chạy lần 2:

\$s0	16	0x2110003d
pc		0x00400008

- Quan sát các byte trong vùng lệnh .text:

Assignment 3:

```
#Laboratory Exercise 2, Assignment 3
.text
li $s0,0x2110003d #pseudo instruction=2 basic instructions
li $s1,0x2 #but if the immediate value is small, one ins
```

Sau đó:

- Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment. Giải thích điều bất thường?

Bài làm

```
#Laboratory Exercise 2, Assignment 3
.text
li $s0,0x2110003d #pseudo instruction=2 basic instructions
li $s1,0x2 #but if the immediate value is small, one ins
```

<input type="checkbox"/>	0x00400000	0x3c012110	lui \$1,0x00002110	3:	li \$s0, 0x2110003d
<input type="checkbox"/>	0x00400004	0x3430003d	ori \$16,\$1,0x0000003d		
<input type="checkbox"/>	0x00400008	0x24110002	addiu \$17,\$0,0x00000002	4:	li \$s1, 0x2


- Câu lệnh li \$s0, 0x2110003d được tách thành 2 lệnh như bài số 2

- Câu lệnh li \$s1, 0x2 được biên dịch thành addiu \$17, 0, 0x00000002
- => Ta viết giả lệnh và chương trình biên dịch thành các lệnh tương ứng.

Assignment 4:

```
#Laboratory Exercise 2, Assignment 4
.text
# Assign X, Y
addi $t1, $zero, 5    # X = $t1 = ?
addi $t2, $zero, -1   # Y = $t2 = ?
# Expression Z = 2X + Y
add $s0, $t1, $t1      # $s0 = $t1 + $t1 = X + X = 2X
add $s0, $s0, $t2      # $s0 = $s0 + $t2 = 2X + Y
```

Sau đó:

- Sử dụng công cụ gỡ rối, Debug, chạy từng lệnh và dừng lại, 
- Ở mỗi lệnh, quan sát cửa sổ Register và chú ý
 - o Sự thay đổi giá trị của các thanh ghi
 - o Sau khi kết thúc chương trình, xem kết quả có đúng không?
- Ở cửa sổ Text Segment, xem các lệnh **addi** và cho biết điểm tương đồng với hợp ngữ và mã máy. Từ đó kiểm nghiệm với khuôn mẫu của kiểu lệnh I

Bài làm

```
#Laboratory Exercise 2, Assignment 4
.text
# Assign X, Y
addi $t1, $zero, 5 # X = $t1 = ?
addi $t2, $zero, -1 # Y = $t2 = ?
# Expression Z = 2X + Y
add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y
```

Text Segment				
Bkpt	Address	Code	Basic	Source
<input type="checkbox"/>	0x00400000	0x20090005	addi \$9,\$0,0x00000005	4: addi \$t1, \$zero, 5 # X = \$t1 = ?
<input type="checkbox"/>	0x00400004	0x200affff	addi \$10,\$0,0xffffffff	5: addi \$t2, \$zero, -1 # Y = \$t2 = ?
<input type="checkbox"/>	0x00400008	0x01298020	add \$16,\$9,\$9	7: add \$s0, \$t1, \$t1 # \$s0 = \$t1 + \$t1 = X + X = 2X
<input type="checkbox"/>	0x0040000c	0x020a8020	add \$16,\$16,\$10	8: add \$s0, \$s0, \$t2 # \$s0 = \$s0 + \$t2 = 2X + Y

- Trước khi chạy:

\$t1	9	0x00000000
\$t2	10	0x00000000
\$s0	16	0x00000000
pc		0x00400000

- Sau khi chạy lần 1:

\$t1	9	0x00000005
\$t2	10	0x00000000

\$s0	16	0x00000000
------	----	------------

pc		0x00400004
----	--	------------

- Sau khi chạy lần 2:

\$t1	9	0x00000005
------	---	------------

\$t2	10	0xffffffff
------	----	------------

\$s0	16	0x00000000
------	----	------------

pc		0x00400008
----	--	------------

- Sau khi chạy lần 3:

\$t1	9	0x00000005
------	---	------------

\$t2	10	0xffffffff
------	----	------------

\$s0	16	0x0000000a
------	----	------------

pc		0x0040000c
----	--	------------

- Sau khi chạy lần 4:

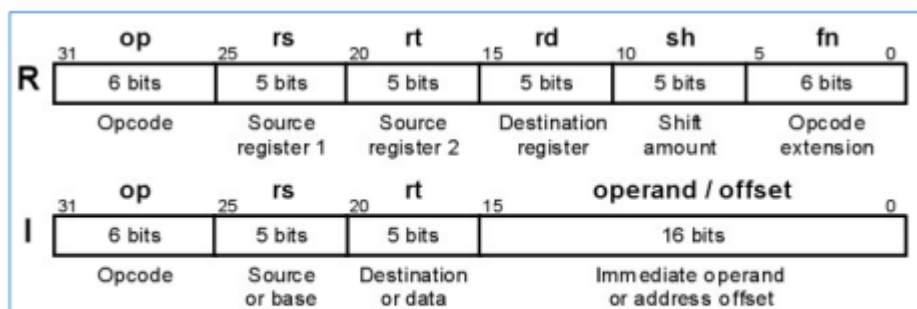
\$t1	9	0x00000005
------	---	------------

\$t2	10	0xffffffff
------	----	------------

\$s0	16	0x00000009
------	----	------------

pc		0x00400010
----	--	------------

⇒ Chương trình kết thúc ta thấy kết quả đúng.



- Ở cửa sổ Text Segment, chuyển mã máy của lệnh **add** sang hệ 2. Từ đó kiểm nghiệm với khuôn mẫu của kiểu lệnh R.

Bài làm

- Phân tích câu lệnh add:

+ add \$s0, \$t1, \$t1:

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

R-type

opcode: 000000

rs: \$9 : 01001

rt: \$9 : 01001

rd: \$16: 10000

shamt: 0 : 00000
 fn: 32 : 100000
 ⇒ 0000 0001 0010 1001 1000 0000 0010 0000 : 0x01298020
 + add \$s0, \$s0, \$t2
 add \$16, \$16, \$10
 R-type
 opcode: 000000
 rs: \$16 : 10000
 rt: \$10 : 01010
 rd: \$16 : 10000
 shamt: 0 : 00000
 fn: 32 : 100000
 ⇒ 0000 0010 0000 1010 1000 0000 0010 0000 : 0x020a8020

→ Đúng với khuôn mẫu kiểu lệnh R.

Assignment 5:

b) Thanh ghi HI và LO
 Thao tác nhân của MIPS có kết quả chứa 2 thanh ghi HI và LO, đây không phải là thanh ghi đa năng. Bit 32 đến 63 thuộc HI và 0 đến 31 thuộc LO

<div style="border: 1px solid black; width: 60px; height: 20px; display: inline-block;"></div>	X	<div style="border: 1px solid black; width: 60px; height: 20px; display: inline-block;"></div>	=	<div style="border: 1px solid black; width: 100px; height: 20px; display: inline-block;"></div>
op1		op2		<div style="display: inline-block; width: 40px; height: 15px; border-bottom: 1px solid black;"></div> hi <div style="display: inline-block; width: 40px; height: 15px; border-bottom: 1px solid black;"></div> lo


Tương tự với phép chia :

<div style="border: 1px solid black; width: 60px; height: 20px; display: inline-block;"></div>	÷	<div style="border: 1px solid black; width: 60px; height: 20px; display: inline-block;"></div>	=	<div style="border: 1px solid black; width: 100px; height: 20px; display: inline-block;"></div>
op1		op2		<div style="display: inline-block; width: 40px; height: 15px; border-bottom: 1px solid black;"></div> remainder <div style="display: inline-block; width: 40px; height: 15px; border-bottom: 1px solid black;"></div> quotient
				<div style="display: inline-block; width: 40px; height: 15px; border-bottom: 1px solid black;"></div> hi <div style="display: inline-block; width: 40px; height: 15px; border-bottom: 1px solid black;"></div> lo

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 5
.text
# Assign X, Y
addi $t1, $zero, 4    # X = $t1 = ?
addi $t2, $zero, 5    # Y = $t2 = ?
# Expression Z = 3*XY
mul   $s0, $t1, $t2    # HI-LO = $t1 * $t2 = X * Y ; $s0 = LO
mul   $s0, $s0, 3      # $s0 = $s0 * 3 = 3 * X * Y
# Z' = Z
mflo  $s1
```

Sau đó:

- Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment. Giải thích điều bất thường?
- Sử dụng công cụ gỡ lỗi, chạy từng lệnh và dừng lại, 
- Ở mỗi lệnh, quan sát cửa sổ Registers và chú ý
 - o Sự thay đổi giá trị của các thanh ghi, đặc biệt là **hi**, **lo**
 - o Sau khi kết thúc chương trình, xem kết quả có đúng không?

Bài làm

```
#Laboratory Exercise 2, Assignment 5
.text
# Assign X, Y
addi $t1, $zero, 4 # X = $t1 = ?
addi $t2, $zero, 5 # Y = $t2 = ?
# Expression Z = 3*XY
mul $s0, $t1, $t2 # HI-LO = $t1 * $t2 = X * Y ; $s0 = LO
mul $s0, $s0, 3 # $s0 = $s0 * 3 = 3 * X * Y
# Z' = Z
mflo $s1
```

Text Segment				
Bkpt	Address	Code	Basic	
<input type="checkbox"/>	0x00400000	0x20090004	addi \$9,\$0,0x00000004	4: addi \$t1, \$zero, 4 # X = \$t1 = ?
<input type="checkbox"/>	0x00400004	0x200a0005	addi \$10,\$0,0x00000005	5: addi \$t2, \$zero, 5 # Y = \$t2 = ?
<input type="checkbox"/>	0x00400008	0x712a8002	mul \$16,\$9,\$10	7: mul \$s0, \$t1, \$t2 # HI-LO = \$t1 * \$t2 = X * Y ; \$s0 = LO
<input type="checkbox"/>	0x0040000c	0x20010003	addi \$1,\$0,0x00000003	8: mul \$s0, \$s0, 3 # \$s0 = \$s0 * 3 = 3 * X * Y
<input type="checkbox"/>	0x00400010	0x72018002	mul \$16,\$16,\$1	
<input type="checkbox"/>	0x00400014	0x00008812	mflo \$17	10: mflo \$s1

- Ta thấy rằng câu lệnh mul \$s0, \$s0, 3 được tách thành hai câu lệnh trong cửa sổ Text Segment. Điều đó xảy ra do câu lệnh mul chỉ nhân hai ô nhớ với nhau nên thanh ghi \$at được dùng để lưu số 3.
- Trước khi chạy:

\$at	1	0x00000000
\$t1	9	0x00000000
\$t2	10	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
pc		0x00400000
hi		0x00000000
lo		0x00000000

- Sau khi chạy lần 1:

\$t1	9	0x00000004
\$t2	10	0x00000000
\$s0	16	0x00000000
\$s1	17	0x00000000
pc		0x00400004
hi		0x00000000
lo		0x00000000

- Sau khi chạy lần 2:

\$t1	9	0x00000004
\$t2	10	0x00000005
\$s0	16	0x00000000
\$s1	17	0x00000000
pc		0x00400008
hi		0x00000000
lo		0x00000000

- Sau khi chạy lần 3:

\$t1	9	0x00000004
\$t2	10	0x00000005

\$s0	16	0x00000014
\$s1	17	0x00000000
pc		0x0040000c
hi		0x00000000
lo		0x00000014

- Sau khi chạy lần 4:

\$at	1	0x00000003
\$t1	9	0x00000004
\$t2	10	0x00000005
\$s0	16	0x00000014
\$s1	17	0x00000000
pc		0x00400010
hi		0x00000000
lo		0x00000014

- Sau khi chạy lần 5:

\$t1	9	0x00000004
\$t2	10	0x00000005
\$s0	16	0x0000003c
\$s1	17	0x00000000
pc		0x00400014
hi		0x00000000
lo		0x0000003c

- Sau khi chạy lần 6:

\$t1	9	0x00000004
\$t2	10	0x00000005
\$s0	16	0x0000003c
\$s1	17	0x0000003c
pc		0x00400018
hi		0x00000000
lo		0x0000003c

➔Sau khi kết thúc chương trình ta thấy kết quả chạy đúng.

Assignment 6:

Gõ chương trình sau vào công cụ MARS.

```
#Laboratory Exercise 2, Assignment 6
.data                                # DECLARE VARIABLES
X : .word    5                      # Variable X, word type, init value =
```

*Ha Noi University of Science and Technology
School of Information and Communication Technology*


```
Y : .word    -1                    # Variable Y, word type, init value =
Z : .word                                # Variable Z, word type, no init value

.text                                # DECLARE INSTRUCTIONS
# Load X, Y to registers
la    $t8, X                        # Get the address of X in Data Segment
la    $t9, Y                        # Get the address of Y in Data Segment
lw    $t1, 0($t8)                  # $t1 = X
lw    $t2, 0($t9)                  # $t2 = Y

# Calculate the expression Z = 2X + Y with registers only
add   $s0, $t1, $t1                # $s0 = $t1 + $t1 = X + X = 2X
add   $s0, $s0, $t2                # $s0 = $s0 + $t2 = 2X + Y

# Store result from register to variable Z
la    $t7, Z                        # Get the address of Z in Data Segment
sw    $s0, 0($t7)                  # Z = $s0 = 2X + Y
```

Sau đó:

- Biên dịch và quan sát các lệnh mã máy trong cửa sổ Text Segment.
 - o Lệnh **la** được biên dịch như thế nào?
- Ở cửa sổ Labels và quan sát địa chỉ của X, Y, Z.
 - o So sánh chúng với hằng số khi biên dịch lệnh **la** thành mã máy
 - o Click đúp vào các biến X, Y, Z để công cụ tự động nhảy tới vị trí của biến X, Y, Z trong bộ nhớ ở cửa sổ Data Segment. Hãy bảo đảm các giá trị đó đúng như các giá trị khởi tạo.
- Sử dụng công cụ gỡ lỗi, chạy từng lệnh và dừng lại, 
- Ở mỗi lệnh, quan sát cửa sổ Registers và chú ý
 - o Sự thay đổi giá trị của các thanh ghi
 - o Xác định vai trò của lệnh **lw** và **sw**
- Ghi nhớ qui tắc xử lý :
 - o Đưa tất cả các biến vào thanh ghi bằng cặp lệnh **la**, **lw**
 - o Xử lý dữ liệu trên thanh ghi
 - o Lưu kết quả từ thanh ghi trở lại biến bằng cặp lệnh **la**, **sw**
- Tìm hiểu thêm các lệnh **lb**, **sb**

Bài làm

```

1  #Laboratory Exercise 2, Assignment 6
2  .data # DECLARE VARIABLES
3  X : .word 5 # Variable X, word type, init value =
4  Y : .word -1 # Variable Y, word type, init value =
5  Z : .word # Variable Z, word type, no init value
6  .text # DECLARE INSTRUCTIONS
7  # Load X, Y to registers
8  la $t8, X # Get the address of X in Data Segment
9  la $t9, Y # Get the address of Y in Data Segment
10 lw $t1, 0($t8) # $t1 = X
11 lw $t2, 0($t9) # $t2 = Y
12 # Calculate the expression Z = 2X + Y with registers only
13 add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X
14 add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y
15 # Store result from register to variable Z
16 la $t7, Z # Get the address of Z in Data Segment
17 sw $s0, 0($t7) # Z = $s0 = 2X + Y

```

Text Segment				
Bkpt	Address	Code	Basic	
<input type="checkbox"/>	0x00400000	0x3c011001	lui \$1,0x00001001	8: la \$t8, X # Get the address of X in Data Segment
<input type="checkbox"/>	0x00400004	0x34380000	ori \$24,\$1,0x00000000	
<input type="checkbox"/>	0x00400008	0x3c011001	lui \$1,0x00001001	9: la \$t9, Y # Get the address of Y in Data Segment
<input type="checkbox"/>	0x0040000c	0x34390004	ori \$25,\$1,0x00000004	
<input type="checkbox"/>	0x00400010	0x8f090000	lw \$9,0x00000000(\$24)	10: lw \$t1, 0(\$t8) # \$t1 = X
<input type="checkbox"/>	0x00400014	0x8f2a0000	lw \$10,0x00000000(\$25)	11: lw \$t2, 0(\$t9) # \$t2 = Y
<input type="checkbox"/>	0x00400018	0x01298020	add \$16,\$9,\$9	13: add \$s0, \$t1, \$t1 # \$s0 = \$t1 + \$t1 = X + X = 2X
<input type="checkbox"/>	0x0040001c	0x020a8020	add \$16,\$16,\$10	14: add \$s0, \$s0, \$t2 # \$s0 = \$s0 + \$t2 = 2X + Y
<input type="checkbox"/>	0x00400020	0x3c011001	lui \$1,0x00001001	16: la \$t7, Z # Get the address of Z in Data Segment
<input type="checkbox"/>	0x00400024	0x342f0008	ori \$15,\$1,0x00000008	
<input type="checkbox"/>	0x00400028	0xadf00000	sw \$16,0x00000000(\$15)	17: sw \$s0, 0(\$t7) # Z = \$s0 = 2X + Y

- Lệnh **la** được tách thành lệnh **lui** và **ori**.
- Ở cửa sổ Labels, địa chỉ của ba biến X, Y, Z giống với hằng số khi biên dịch lệnh **la** thành mã máy

Labels	
Label	Address ▲
mips6.asm	
X	0x10010000
Y	0x10010004
Z	0x10010008

- Sau khi chạy lần 1:

\$at	1	0x10010000
pc		0x00400004

+ Thanh \$at từ 0x00000000 thay đổi thành 0x10010000

- Sau khi chạy lần 2:

\$t8	24	0x10010000
pc		0x00400008

+ Thanh \$t8 thay đổi từ 0x00000000 thành 0x10010000

- Sau khi chạy lần 3:

\$at	1	0x10010000
pc		0x0040000c

+ Thanh \$at giữ nguyên giá trị.

- Sau khi chạy lần 4:

\$t9	25	0x10010004
pc		0x00400010

+ Thanh \$t9 thay đổi giá trị từ 0x00000000 thành 0x10010004

- Sau khi chạy lần 5:

\$t1	9	0x00000005
pc		0x00400014

+ Thanh \$t1 thay đổi giá trị từ 0x00000000 thành 0x00000005

- Sau khi chạy lần 6:

\$t2	10	0xffffffff
pc		0x00400018

+ Thanh \$t2 thay đổi giá trị từ 0x00000000 thành 0xffffffff.

- Sau khi chạy lần 7:

\$s0	16	0x0000000a
pc		0x0040001c

+ Thanh \$s0 thay đổi từ 0x00000000 thành 0x0000000a.

- Sau khi chạy lần 8:

\$s0	16	0x00000009
pc		0x00400020

+ Thanh \$s0 thay đổi từ 0x0000000a thành 0x00000009

- Sau khi chạy lần 9, 10:

\$t7	15	0x10010008
pc		0x00400028

+ Thanh \$t7 thay đổi từ 0x00000000 thành 0x10010008.

- Vai trò của lệnh **lw** và **sw**:

+ **lw**: Tải word từ bộ nhớ

+ **sw**: Lưu trữ word trong bộ nhớ