BÀI THỰC HÀNH TUẦN 2
KIẾN TRÚC MÁY TÍNH

Họ và tên: Đinh Huy Dương

MSSV: 20215020

Bài 1:

Code:

```
#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
```

a, Quá trình Debug:

$s0: Thay đổi từ 0x0000 3007 đến 0x0000 0000

Lệnh `addi $s0, $zero, 0x3007`:

Cộng giá trị của thanh ghi $s0 = $0 + 0x3007 = 0x3007

Lệnh `add $s0, $zero, $0`:

Cộng giá trị của thanh ghi $s0 = $0 + 0 = 0

$pc: Bắt đầu từ 0x0040 0000, qua từng câu lệnh tăng lên 4: 0x0040 0004 -> 0x0040 0008

Thanh ghi $pc là bộ đếm chương trình, mỗi lần đi qua một lệnh thanh $pc sẽ tăng lên với độ dài của lệnh ở dạng byte, trong trường hợp này, mỗi lệnh ở MIPS có độ dài 32 bit tương đương với 4 byte nên sẽ tăng lên 4 qua mỗi câu lệnh

b, Chuyển đổi lệnh sang mã máy:

Source: `addi $s0, $zero, 0x3007`

Basic: addi $16, $0, 0x00003007

addi $rt, $rs, imm

Op: 8 -> 001000

$rt: 10000

$rs: 00000

imm: 0011 0000 0000 0111

001000 00000 10000 0011 0000 0000 0111

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

Kiểm tra đúng với Mã Máy

==========

Source: `add $s0, $zero, $0`

Basic: add $16, $0, $0

add $rd, $rs, $rt

Op: 000000

$rd: 10000

$rs: 00000

$rt: 00000

$sh: 00000

$fn: 100000

000000 00000 00000 10000 00000 100000

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

Kiểm tra đúng với Mã Máy

c, Thay giá trị imm trong lệnh addi thành `0x2110003d` (32 bit)

Mà giá trị imm (offset) trong câu lệnh I chỉ chứa tối đa 16 bit

Nên máy sẽ thực hiện 3 lệnh thay lệnh addi:

```
lui    $at, 0x00002110
ori   $at, $at, 0x0000003d
add   $s0, $zero, $at
```

Máy sẽ chia đôi giá trị `0x2110003d` và gán 1 nửa đầu (0x2110) vào thanh ghi tạm thời qua lệnh lui (do lệnh lui cũng thuộc tập I nên chỉ tối đa 16 bit), và lệnh ori để thực hiện phép OR cho $at và nửa sau của 0x211003d (0x003d)

Từ đó giá trị của $at sẽ mang hoàn toàn giá trị `0x2110003d`, sử dụng lệnh add để gán giá trị của thanh ghi $at vào trong $s0 ban đầu

Bài 2:

a, Code:

```
#Laboratory Exercise 2, Assignment 2
.text
        lui   $s0,0x2110       #put upper half of pattern in $s0
        ori   $s0,0x003d       #put lower half of pattern in $s0
```

Thanh ghi $s0 sẽ nhận 1 nửa giá trị đầu (16 bit) của giá trị 0x2110003d qua lệnh lui và qua lệnh ori $s0 sẽ thực hiện phép OR với nửa giá trị sau (0x003d) để gán giá trị 0x2110003d vào $s0.

$s0 qua các câu lệnh: 0x21000000 -> 0x2100003d

b, Trên cửa sổ Data Segment, ta có thể quan sát được các byte của lệnh trong từng ô địa chỉ lệnh. Ta thấy được tại ô địa chỉ 0x00400000 giá trị byte tương ứng là 0x3c102110, tương ứng với mã (Code) của câu lệnh lui trong Text Segment. Bên cạnh cột đó là địa chỉ (+4) 0x00400004 là giá trị byte của mã của lệnh ori trong Text Segment.

Bài 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
```

Tại cửa sổ Thực thi, lệnh li sẽ biến thành các lệnh khác nhau tùy vào tham số

```
		lui   $at, 0x00002110
		ori   $s0, $at, 0x0000003d
		addiu $s1, $zero, 0x00000002
```

Lệnh li đầu tiên là gán 1 giá trị 32 bit vào trong thanh ghi $s0, nên máy khi dịch máy sẽ chuyển thành 2 lệnh lui và ori như bài 2, còn lại khi li gán giá trị nhỏ, máy sẽ chuyển thành câu lệnh addiu (là lệnh add với các số không dấu)

Bài 4:

a,

```
#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
```

Khi chạy chương trình, Gán biến X (=5) và Y (=-1) lần lượt cho các thanh ghi $t1 (=0x0000 0005) và $t2 (=ffffffff). Khi đó gán kết quả tại thanh ghi $s0, lần lượt cộng X+X ($t1+$t1) và cộng thêm Y (+$t2) để ra được kết quả.

Kiểm nghiệm kết quả tại $s0 =9 đúng.

b, Lệnh:

```
addi $t1, $zero, 5
```

```
Basic: addi $9, $0, 0x00000005
```

addi $rt, $rs, imm

```
op: 8 -> 001000

$rt: 01001

$rs: 00000

imm: 00000000000101

I: OP | rs | rt | offset
```

➔ 001000 00000 01001 0000 0000 0000 0101
➔ 0010 0000 0000 1001 0000 0000 0000 0101
➔ 0x20090005

Đúng với khuôn mẫu kiểu I

c, Tương tự cũng kiểm chứng

khuôn mẫu kiểu R: OP | rs | rt | rd | sh |fn qua lệnh

```
add $s0, $t1, $t1
```

add $16, $9, $9

có Code: 0x01298020

= 0000 0001 0010 1001 1000 0000 0010 0000

= 000000 01001 01001 10000 00000 100000

➔ op = 0
rs =9
rt =9 -> $t1
rd = 16 -> $s0
sh = 0
fn =32

Kiểm chứng khuôn mẫu chính xác

Bài 5:

```
#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
```

Khi thực hiện phép nhân mul, lần đầu tiên sẽ ra kết quả 20 và gán vào $s0 với giá trị 0x14 (vẫn bé hơn 32 bit), và tại lo cũng gán giá trị 0x14 từ phép nhân. Tuy nhân tại lần nhân tiếp theo khi 3*XY = 60 (lớn hơn 32 bit), lệnh mul sẽ phải thêm lệnh addi để gán giá trị 0x3 vào trong thanh ghi tạm thời $at, rồi mới nhân $s0

để gán vào lo và $s0.

Bài 6:

Lệnh la được biên dịch thành lệnh lui (dùng thanh ghi $at) và ori để gán địa chỉ của 1 biến với một thanh ghi tương ứng

Khai báo các biến X, Y, Z theo kiểu Word, khi đó ta có thể thấy các biến trên cửa sổ Label

Sau đó lệnh lw sẽ gán giá trị theo kiểu word của X và Y vào trong thanh ghi $t8 và $t9 tương ứng, và lệnh sw sẽ lưu giá trị thanh ghi vào bộ nhớ