Thực hành kiến trúc máy tính

Lab 2

Họ và tên: Lê Quốc Đảng

MSSV: 20225801

Exercise 1

lab2.asm

```
1  #Laboratory Exercise 2, Assignment 1
2  .text
3  addi $s0, $zero, 0x3007 # $s0 = 0 + 0x3007 = 0x3007 ;I-type
4  add $s0, $zero, $0 # $s0 = 0 + 0 = 0 ;R-type
5
```

Sự thay đổi giá trị của:

- Thanh ghi $s0: sau lệnh thứ nhất thay đổi từ giá trị 0x00000000 thành 0x00003007. Sau lệnh thứ 2 thành 0x00000000.
- Thanh ghi $pc: sau mỗi lệnh tăng thêm 1 giá trị là 0x00000004.

-So sánh mã máy:

- Lệnh 1: Lệnh I, opcode 8 = 001000, rs 0 = 00000, rt 16 = 10000, imm 0x3007 = 0101 0000 0000 0111
  Mã máy là: 0010 0000 0001 0000 0101 0000 0000 0111
  Tương đương: 0x20103007
- Lệnh 2: Lệnh R, opcode 0 = 000000, rs 0 = 00000, rt 0 = 00000, rd 16 = 10000, funct 32 = 100000
  Mã máy là: 0000 0000 0000 0000 1000 0000 0010 0000
  Tương đương: 0x00008020
- Đúng như chạy trong ứng dụng

-Nếu sửa lại lệnh lui như bên dưới:

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

thì nó phải thực hiện them lệnh gán 32 bit vào thanh ghi $at bởi lệnh I chỉ cho phép imm có 16 bit.

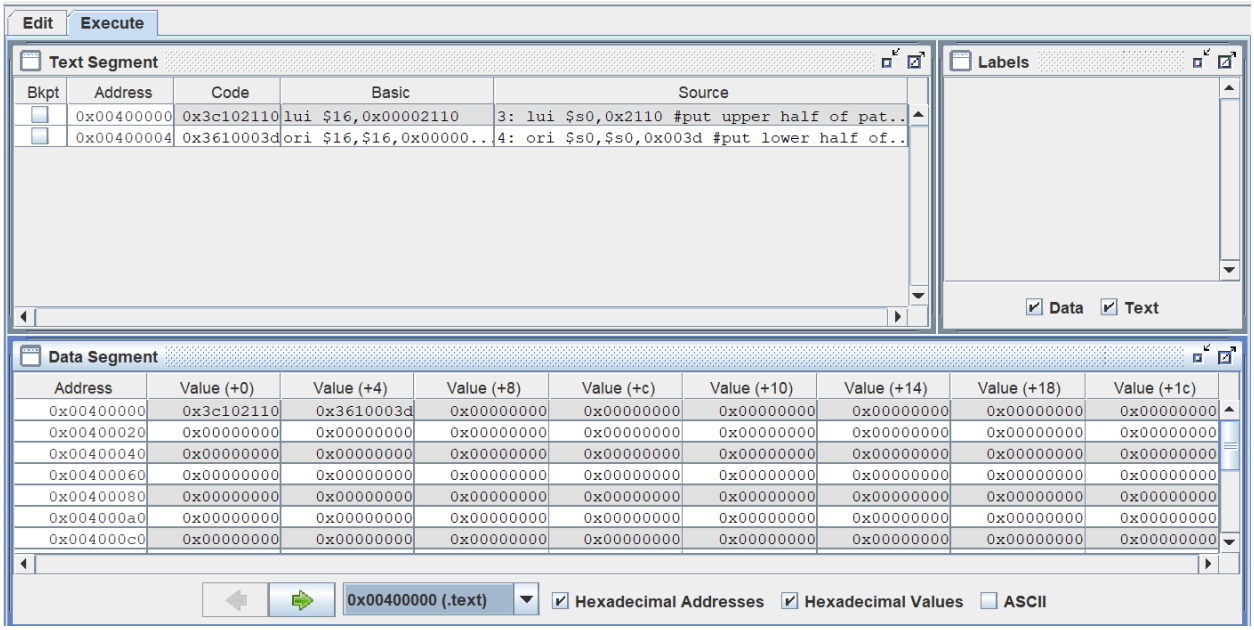
Exercise 2

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

-Sự thay đổi giá trị của:

- Thanh ghi $s0: sau lệnh thứ nhất thay đổi từ giá trị 0x00000000 thành 0x21100000. Sau lệnh thứ 2 thành 0x2110003d.
- Thanh ghi $pc: sau mỗi lệnh tăng thêm 1 giá trị là 0x00000004.

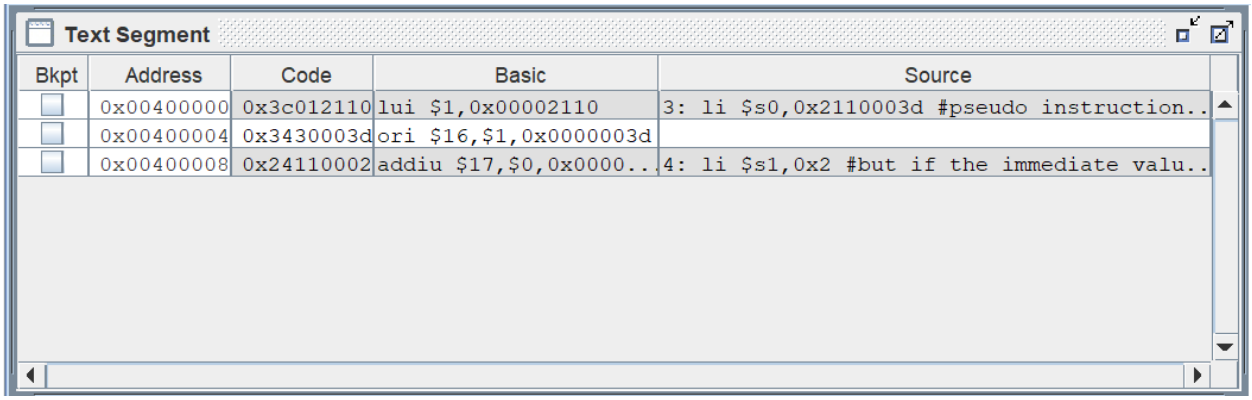
-Các byte đầu tiên ở vùng lệnh trùng với cột code trong cửa sổ Text Segment.



Exercise 3

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

-Kết quả biên dịch:



-Điều bất thường ở đây là lệnh 1 chuyển từ li thành 2 lệnh là lui và ori, còn lệnh thứ 2 chuyển từ li sang lệnh addiu.

-Giải thích:

- Lệnh thứ 1: imm có giá trị là 32 bit nên phải tách thành 2 lệnh để mỗi lệnh lưu 16 bit.
- Lệnh thứ 2: bởi 0x2 là loại 16 bit có dấu nên chuyển thành lệnh addiu.

Exercise 4

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

-Sự thay đổi giá trị của

- Thanh ghi $t1: thay đổi từ 0x00000000 sang 0x00000005.
- Thanh ghi $t2: thay đổi từ 0x00000000 sang 0xffffffff.
- Thanh ghi $s0: thay đổi từ 0x00000000 sang 0x0000000a sau lệnh add thứ nhất, sang 0x00000009 sau lệnh add thứ 2.
- Thanh ghi $pc: tăng thêm 0x00000004 sau mỗi lệnh.

-Sau khi kết thúc chương trình, kết quả đúng.

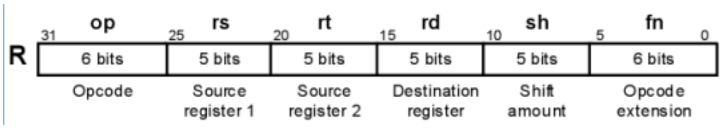
- Ở cửa số Text Segment, xem các lệnh **addi** và cho biết điểm tương đồng với hợp ngữ và mã máy: 2 lệnh addi trên có chung 001000 00000. Nó tương ứng với tên lệnh (addi) và thanh ghi nguồn $zero. Các bit còn lại khác nhau tương ứng với các thành phần còn lại của câu lệnh.

Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20090005	addi $9,$0,0x00000005	4: addi $t1, $zero, 5 # X = $t1 = ?
	0x00400004	0x200affff	addi $10,$0,0xffffff..	5: addi $t2, $zero, -1 # Y = $t2 = ?
	0x00400008	0x01298020	add $16,$9,$9	7: add $s0, $t1, $t1 # $s0 = $t1 + $t1 =..
	0x0040000c	0x020a8020	add $16,$16,$10	8: add $s0, $s0, $t2 # $s0 = $s0 + $t2 =..

-Kiểm nghiệm 2 lệnh I:

- Lệnh addi $t1, $zero, 5:
  Opcode 8 = 001000, rs 0 = 00000, rt 9 = 01001, imm 5 = 0000 0000 0000 1001
  Mã máy: 0010 0000 0000 1001 0000 0000 0000 10001
  Tương đương: 0x20090005
- Lệnh addi $t2, $zero, -1:
  Opcode 8 = 001000, rs 0 = 00000, rt 10 = 01010, imm -1 = 1111 1111 1111 1111
  Mã máy: 0010 0000 0000 1010 1111 1111 1111 1111
  Tương đương: 0x200affff



-Lệnh add

- 0x01298020
  Mã nhị phân: 0000 0001 0010 1001 1000 0000 0010 0000
  Opcode 000000 => lệnh kiểu R
  rs 01001 : 9 => $9 => $t1
  rt 01001 : 9 => $9 => $t1
  rd 10000 : 16 => $16 => $s0
  sh 00000
  funct 100000 : 32 => lệnh add

⇨ Add $s0, $t1, $t1

- 0x020a8020
  Mã nhị phân: 0000 0010 0000 1010 1000 0000 0010 0000
  Opcode 000000 => lệnh kiểu R
  rs 10000 : 16 => $16 => $s0
  rt 01010 : 10 => $10 => $t2

rd 10000 : 16 => $16 => $s0
sh 00000
funct 100000 : 32 => lệnh add

⇨ Add $s0, $s0, $t2

Exercise 5

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

-Điều bất thường: lệnh mul thứ nhất khác lệnh mul thứ 2, lệnh mul thứ 2 phải tách thành 2 lệnh addi, mul.

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20090004	addi $9,$0,0x00000004	4: addi $t1, $zero, 4 # X = $t1 = ?
	0x00400004	0x200a0005	addi $10,$0,0x00000..	5: addi $t2, $zero, 5 # Y = $t2 = ?
	0x00400008	0x712a8002	mul $16,$9,$10	7: mul $s0, $t1, $t2 # HI-LO = $t1 * $t..
	0x0040000c	0x20010003	addi $1,$0,0x00000003	8: mul $s0, $s0, 3 # $s0 = $s0 * 3 = 3 ..
	0x00400010	0x72018002	mul $16,$16,$1	
	0x00400014	0x00008812	mflo $17	10: mflo $s1

-Giải thích: lệnh mul thực hiện phép nhân giữa các thanh ghi. Mà lệnh thứ 2 thực hiện phép nhân với 1 số nên phải gán số đó vào thanh ghi $1 rồi thực hiện phép nhân.

-Sự thanh đổi của các thanh ghi

Bước nhảy	Thanh ghi	Trước bước nhảy	Sau bước nhảy
1	$t1	0x00000000	0x00000004
2	$t2	0x00000000	0x00000005
3	$s0	0x00000000	0x00000014
	lo	0x00000000	0x00000014
4	$at	0x00000000	0x00000003
5	$s0	0x00000014	0x0000003c
	lo	0x00000014	0x0000003c
6	$s1	0x00000000	0x0000003c

Thanh ghi pc: tăng thêm 0x00000004 sau mỗi lệnh.

-Kết thúc chương trình, kết quả đúng.

Exercise 6

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

-So sánh chúng với hằng số khi biên dịch lệnh la thành mã máy: 16 bit đầu của X trở thành hằng số cho lệnh lui, 16 bit sau của X trở thành hằng số cho lệnh ori. Y tương tự.

-Lệnh la được biên dịch thành 2 lệnh lui và ori gắn số 32 bit vào thanh ghi.

-Sự thay đổi của các thanh ghi

Bước nhảy	Thanh ghi	Trước bước nhảy	Sau bước nhảy
1	$at	0x00000000	0x10010000
2	$t8	0x00000000	0x10010000
3	$at	0x00000000	0x10010000
4	$t9	0x00000000	0x10010004
5	$t1	0x00000000	0x00000005
6	$t2	0x00000000	0xffffffff
7	$s0	0x00000000	0x0000000a
8	$s0	0x0000000a	0x00000009
9	$at	0x10010000	0x10010000
10	$st7	0x00000000	0x10010008

Thanh ghi pc tăng thêm 0x00000004 sau mỗi lệnh.

-Lệnh lw lấy giá trị có kích thước 4 byte từ vùng dữ liệu vào thanh ghi thông qua địa chỉ thanh ghi và địa chỉ vùng nhớ. Lệnh sw lưu dữ liệu từ thanh ghi vào bộ nhớ thông qua địa chỉ thanh ghi và địa chỉ vùng nhớ.

-Tìm hiểu thêm các lệnh lb, sb: lệnh lb và sb tương tự như lệnh lw và sw. lb và sb khác lw và sw ở lượng dữ liệu sử dụng. Lệnh lb và sb sử dụng 1 byte (8 bit) còn lệnh lw và sw sử dụng 4 byte (32 bit).