Computer Architecture Lab Report Week 2

Full name: On Quang Tung

Student ID: 20226096

Assignment 1

```
#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 khi sử dụng công cụ gỡ rối, chạy từng lệnh và dừng lại, ta nhận thấy:

- Các sự thay đổi:
  - Thanh ghi $s0 thay đổi từ 0x00000000 thành 0x00003007 sau câu lệnh đầu tiên và trở lại thành 0x00000000 sau câu lệnh thứ 2.
  - Thanh ghi PC tăng 4 byte sau mỗi câu lệnh, với giá trị khởi đầu là 0x00400000.
- Mã máy:
  - Lệnh 1: Là lệnh I, opcode: 8 => 001000, rs: 0 => 00000, rt: 16 => 10000, imm: 0x3007=> 0011 0000 0000 0111.
    Kết luận mã máy: 0010 0000 0001 0000 0011 0000 0000 0111 (0x20103007)
  - Lệnh 2: Là lệnh R, opcode: 0 => 000000, rs: 0 => 00000, rt: 0 => 00000, rd: 16 => 10000, sh: 0 => 00000, fn: 32 => 100000
    Kết luận mã máy: 0000 0000 0000 0000 1000 0000 0010 0000 (0x00008020)

  ⇨ Lệnh giống với ô code trong text segment.
- Nếu sửa lại lệnh addi thành addi $s0, $zero, 0x2110003d thì nó sẽ trở thành lệnh gán với tham số imm là quá 16 bit, từ đó lệnh này sẽ thành lệnh mở rộng. Lệnh mở rộng này sẽ biến thành 3 lệnh khác, đó là lệnh lui, ori, và add (như hình).

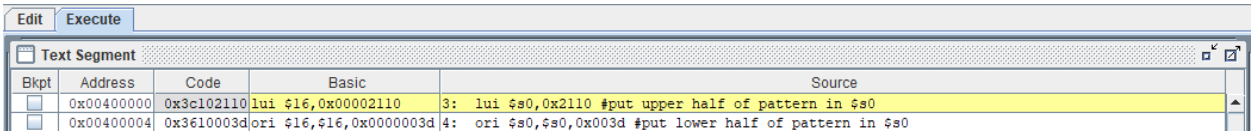
lui $1,0x00002110	3: addi $s0, $zero, 0x2110003d
ori $1,$1,0x0000003d	
add $16,$0,$1	

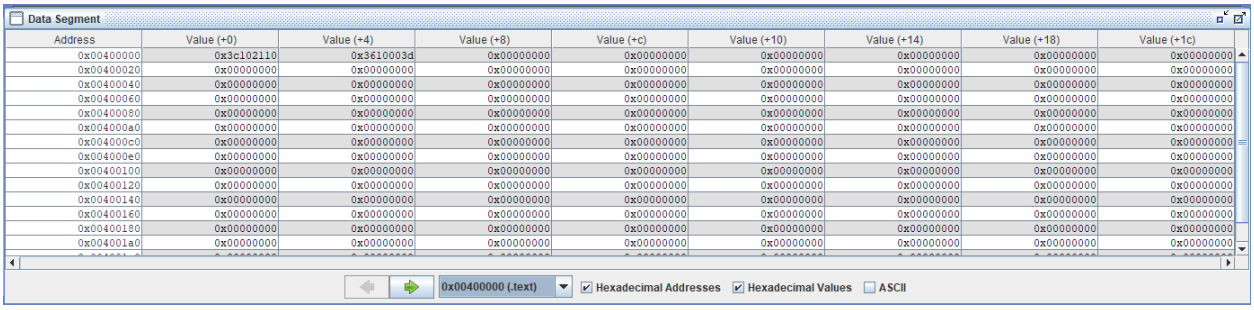
Assignment 2

```
#Laboratory Exercise 2, 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 khi sử dụng công cụ gỡ rối, chạy từng lệnh và dừng lại, ta nhận thấy:

- Các sự thay đổi:
  - Thanh ghi $s0 thay đổi từ 0x00000000 thành 0x00003007 sau câu lệnh đầu tiên và trở lại thành 0x00000000 sau câu lệnh thứ 2.
  - Thanh ghi PC tăng 0x00000004 sau mỗi câu lệnh, với giá trị khởi đầu là 0x00400000.
- Các byte đầu tiên ở vùng lệnh Address trong cửa sổ Text Segment trùng với cột code(theo hệ cơ số Hex) trong cửa sổ Text Segment ở phần Execute.

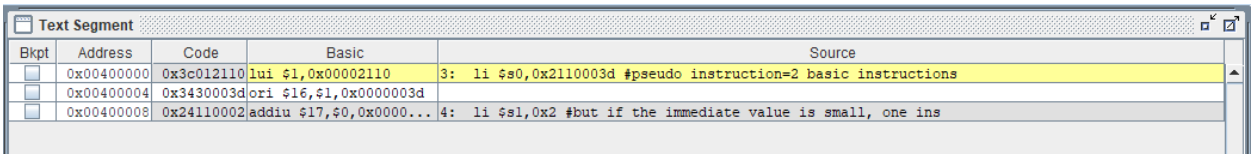




Assignment 3

```
#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
```

Kết quả của các lệnh trên:



Text Segment

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

- Ta thấy, đã xảy ra việc bất thường. Đó chính là lệnh li thứ 1 đã được biến đổi thành 2 lệnh lui và ori, và lệnh li thứ 2 biến thành lệnh addiu.
- Giải thích:
  - Lệnh li thứ 1 tách ra làm 2 lệnh lui và ori do tham số imm ở trong lệnh quá 16 bit cho phép. Nên để có thể lưu tham số này vào $16, nó đã lưu 16 bit trên vào thanh ghi $1 sau đó biến tất cả các bit dưới thành 0(lệnh lui), sau đó lệnh ori sẽ thực hiện lưu các bits còn lại vào 16 bit dưới vào thanh ghi $1.
  - Lệnh li thứ 2 biến thành lệnh addiu do nó thực hiện gán số 0x2 là loại 16 bit không dấu.

Assignment 4

```
#Laboratory Exercise 2, Assignment 4
.text
 # Assign X, Y
 addi $t1, $zero, 5 # X = $t1 = 5
 addi $t2, $zero, -1 # Y = $t2 = -1
 # Expression Z = 2X + Y
 add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X = 10
 add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y = 10 - 1 = 9
```

Sự thay đổi của các thanh ghi:

- Sau lệnh 1 thanh ghi $t1 thay đổi từ 0x00000000 thành 0x00000005
- Sau lệnh 2 thanh ghi $t2 thay đổi từ 0x00000000 thành 0xffffffff
- Sau lệnh 3 thanh ghi $s0 thay đổi từ 0x00000000 thành 0x0000000a
- Sau lệnh 4 thanh ghi $s0 thay đổi từ 0x0000000a thành 0x00000009
- Thanh ghi PC tăng 0x00000004 sau mỗi câu lệnh, với giá trị khởi đầu là 0x00400000.

Điểm tương đồng giữa hợp ngữ và mã máy trong các lệnh addi:

- Sau khi chuyển các yếu tố của lệnh loại I này(như opcode, rd, rs, rt, sh, imm) thì ta nhận lại được 1 dãy nhị phân 32 bit. Khi chuyển dãy mã máy thành thành mã HEX thì ta được kết quả giống với với cột code trong cửa sổ Text Segment ở phần Execute.
- Kiểm nghiệm lệnh addi với lệnh I:
  - addi $t1, $zero, 5 :
    opcode: 8 => 001000
    rs: $0 => 00000
    rt: $9 => 01001
    imm: 0x5
    => 0000 0000 0000 0101 0010 0000 0000 1001 0000 0000 0000 0101
    0x20090005
  - addi $t2, $zero, -1:
    opcode: 8 => 001000
    rs: $0 => 00000
    rt: $10 => 01010

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

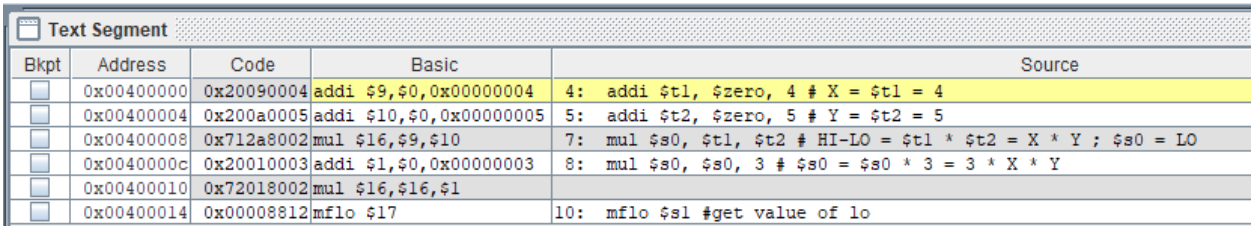
- Chuyển mã máy của lệnh add sang hệ 2:
  0x01298020 => 0000 0001 0010 1001 1000 0000 0010 0000
  Opcode: 000000 => 0
  rs: 01001 => $9
  rt: 01001 => $9
  rd: 10000 => $16
  sh: 00000
  fn: 100000
  ⇨ add $16, $9, $9

  0x020a8020
  0000 0010 0000 1010 1000 0000 0010 0000
  Opcode: 000000 => 0
  rs: 10000 => $16
  rt: 01010 => $10
  rd: 10000 => $16
  sh: 00000
  fn: 100000
  ⇨ add $16, $16, $10

Assignment 5

```
#Laboratory Exercise 2, Assignment 5
.text
 # Assign X, Y
 addi $t1, $zero, 4 # X = $t1 = 4
 addi $t2, $zero, 5 # Y = $t2 = 5
 # 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 #get value of lo
```

Kết quả của các lệnh trên:



Text Segment

Bkpt	Address	Code	Basic	Source
	0x00400000	0x20090004	addi $9,$0,0x00000004	4: addi $t1, $zero, 4 # X = $t1 = 4
	0x00400004	0x200a0005	addi $10,$0,0x00000005	5: addi $t2, $zero, 5 # Y = $t2 = 5
	0x00400008	0x712a8002	mul $16,$9,$10	7: mul $s0, $t1, $t2 # HI-LO = $t1 * $t2 = X * Y ; $s0 = LO
	0x0040000c	0x20010003	addi $1,$0,0x00000003	8: mul $s0, $s0, 3 # $s0 = $s0 * 3 = 3 * X * Y
	0x00400010	0x72018002	mul $16,$16,$1	
	0x00400014	0x00008812	mflo $17	10: mflo $s1 #get value of lo

- Ta thấy, đã xảy ra việc bất thường. Đó chính là lệnh mul thứ 2 đã được biến đổi thành 2 lệnh addi và mul.
- Giải thích:
  Vì lệnh mul không hỗ trợ việc nhân thanh ghi với một số, nên nó phải tách thành lệnh addi để lưu 1 số vào thanh ghi $1(như ảnh) và dùng lệnh mul sau khi tách để nhân 2 thanh ghi đó với nhau.

Sự thay đổi các thanh ghi:

- Thanh ghi $t1, $t2 dùng để gán giá trị 4 và 5, thanh ghi lo thay đổi thành 0x0000003c, thanh ghi hi không thay đổi.
- Thanh ghi $at lưu giá trị tạm thời 0x00000003 khi thực hiện phép nhân với số 3.

Assignment 6

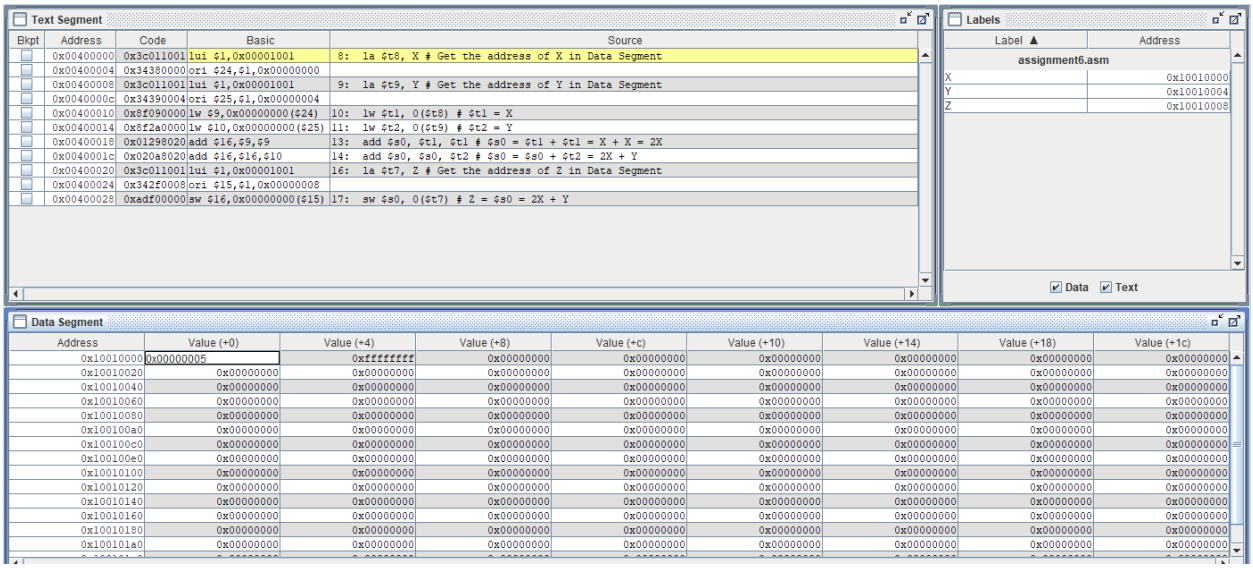
```
#Laboratory Exercise 2, Assignment 6
.data # DECLARE VARIABLES
X : .word 5 # Variable X, word type, init value =
Y : .word -1 # Variable Y, word type, init value =
Z : .word # Variable Z, word type, no init value
.text # DECLARE INSTRUCTIONS
 # Load X, Y to registers
 la $t8, X # Get the address 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
```

Lệnh la được biên dịch thành 2 lệnh lui và ori để gán 1 số 32 bit như hình ảnh bên dưới:

Basic	Source
lui $1,0x00001001	8: la $t8, X # Get the address of X in Data Segment
ori $24,$1,0x00000000	
lui $1,0x00001001	9: la $t9, Y # Get the address of Y in Data Segment
ori $25,$1,0x00000004	

Ở cửa sổ Labels và quan sát địa chỉ của X, Y, Z:

- Khi biên dịch lệnh la thành mã mãy, các biến X, Y, Z là bằng nhau.
- Khi chạy chương trình, ta thấy giá trị X, Y, Z đúng với giá trị khởi tạo (như hình)



- Sự thay đổi của các thanh ghi:
  - Thanh ghi $at thay đổi từ 0x00000000 thành 0x10010000 sau lệnh 1.
  - Thanh ghi $t8 thay đổi từ 0x00000000 thành 0x10010000 sau lệnh 2.
  - Thanh ghi $t9 thay đổi từ 0x00000000 thành 0x10010004 sau lệnh 3.
  - Thanh ghi $t1 thay đổi từ 0x00000000 thành 0x00000005 sau lệnh 4.
  - Thanh ghi $t2 thay đổi từ 0x00000000 thành 0xffffffffffff sau lệnh 5.
  - Thanh ghi $s0 thay đổi từ 0x00000000 thành 0x0000000a sau lệnh 6, từ 0x0000000a thành 0x00000009 sau lệnh 7.
  - Thanh ghi $t7 thay đổi từ 0x00000000 thành 0x10010008 sau lệnh 8.
  - Thanh ghi PC tăng 0x00000004 sau mỗi câu lệnh, với giá trị khởi đầu là 0x00400000.
- Vai trò lệnh lw và sw:
  - Lệnh lw: Lấy địa chỉ của biến kiểu word và lưu vào 1 thanh ghi
  - Lệnh sw: Lấy địa chỉ của biến kiểu word lưu vào bộ nhớ