Thực hành Kiến trúc máy tính tuần 2

Họ tên: Đỗ Hoàng Minh Hiếu

MSSV: 20225837

Assignment 1

*Sự thay đổi của các thanh ghi:

- Trước khi thực hiện câu lệnh thứ nhất, thanh ghi PC trỏ đến địa chỉ của câu lệnh đầu tiên (0x00400000), thanh ghi $s0 vẫn giữ nguyên giá trị.

- Sau khi thực hiện câu lệnh thứ nhất, thanh ghi PC trỏ đến vị trí câu lệnh thứ 2 (0x00400004), con trỏ $s0 lưu giá trị mới (0x00003007) theo source code.

- Sau khi thực hiện xong câu lệnh thứ hai, thanh ghi PC trỏ đến vị trí tiếp theo (0x00400008), con trỏ $s0 lưu giá trị mới (0x00000000) theo source code.

Bkpt	Address	Code	Basic	Source
☐	0x00400000	0x20103007	addi $16,$0,0x00003007	2: addi $s0, $zero, 0x3007
☐	0x00400004	0x00008020	add $16,$0,$0	3: add $s0, $zero, $0

*So sánh mã máy và khuôn dạng lệnh

- Lệnh 1: addi $s0, $zero, 0x3007

Opcode: 8 => 001000

$rs: $zero => $0 => 00000

$rt: $s0 =>$16 => 10000

Imm: 0x3007 => 0011 0000 0000 0111

⇨ Mã máy dạng nhị phân: 0010 0000 0001 0000 0011 0000 0000 0111
⇨ Dạng hex: 0x20103007

- Lệnh 2: add $s0, $zero, $0

Opcode: 0 => 000000

$rs: $zero => $0 => 00000

$rt: $0 => 00000

$rd: $s0 =>$16 => 10000

shamt: 0 => 00000

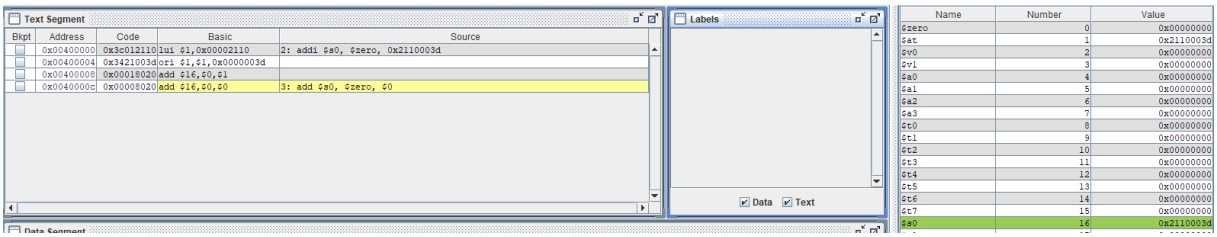
funct: 32 => 100000

⇨ Mã máy dạng nhị phân: 0000 0000 0000 0000 1000 0000 0010 0000
⇨ Dạng hex: 0x00008020

Vậy các lệnh đó đúng như tập lệnh đã quy định

*Khi sửa lại lệnh addi thành addi $s0, $zero, 0x2110003d:

Do 0x2110003d tràn số do imm là 32bit còn addi chỉ thực hiện với số 16bit. Để xử lý tình huống này, ta sử dụng lệnh *lui* với chức năng ghi một hằng số 16bit vào 2 byte cao của thanh ghi, 2 byte thấp sẽ được gán bằng 0, và sẽ gán vào $at(0x21100000). Sau đó sử dụng ori để ghép phần còn lại (0x0000003d) vào $at (0x21100000 OR 0x0000003d sẽ thành 0x2110003d). Cuối cùng dùng lệnh add để đưa về thanh ghi $s0.



Assignment 2

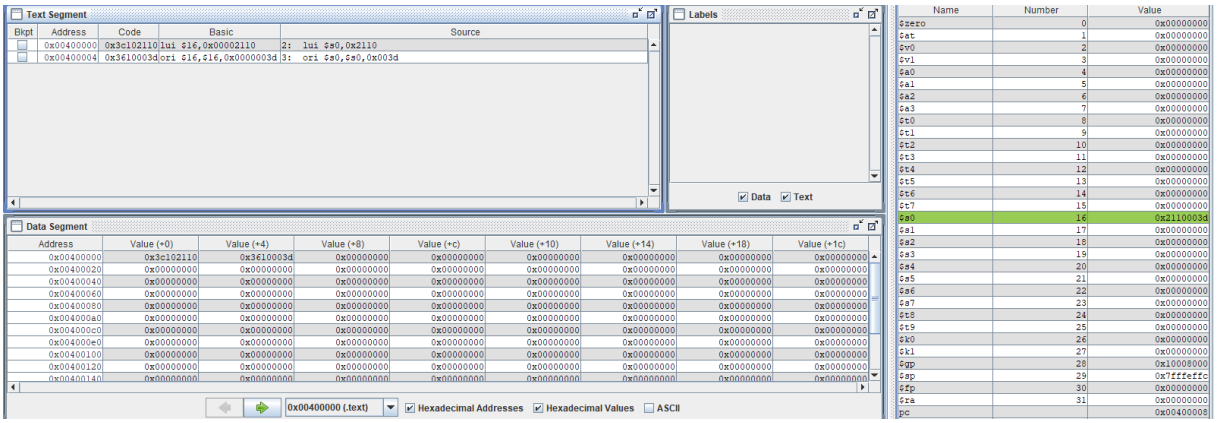
*Sự thay đổi của các thanh ghi:

- Trước khi thực hiện câu lệnh thứ nhất, thanh ghi PC trỏ đến địa chỉ của câu lệnh đầu tiên (0x00400000), thanh ghi $s0 vẫn giữ nguyên giá trị.

- Sau khi thực hiện câu lệnh thứ nhất, thanh ghi PC trỏ đến vị trí câu lệnh thứ 2 (0x00400004), con trỏ $s0 lưu giá trị mới (0x21100000) theo source code (Lệnh lui gán vào 2 byte đầu).

- Sau khi thực hiện xong câu lệnh thứ hai, thanh ghi PC trỏ đến vị trí tiếp theo (0x00400008), con trỏ $s0 lưu giá trị mới (0x2110003d) theo source code.

* Sau khi chuyển sang .text, các byte đầu tiên ở vùng lệnh trùng với cột code trong cửa sổ Text Segment.



Assignment 3

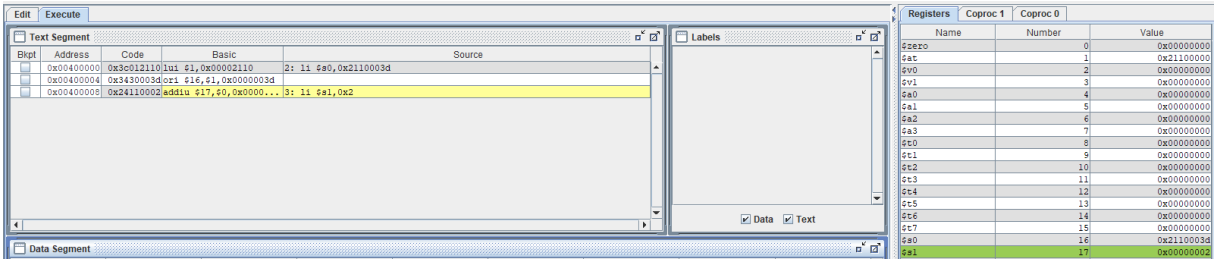
Bất thường: Lệnh đầu tiên (li $s0,0x2110003d) chuyển thành hai lệnh lui và ori. Còn lệnh thứ hai (li $s1,0x2) chuyển thành lệnh addiu.

Giải thích:

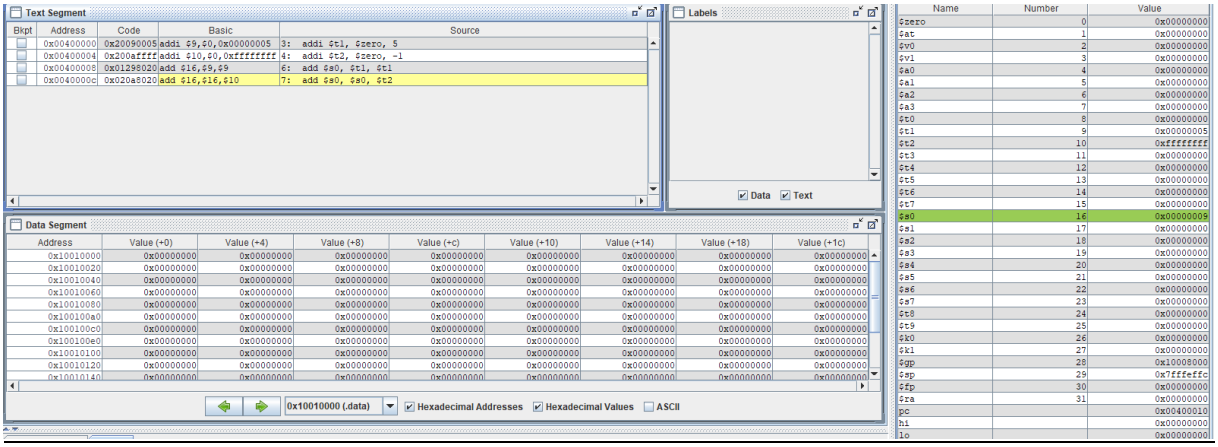
Ở câu lệnh đầu, do 0x2110003d tràn số do imm là 32bit còn addi chỉ thực hiện với số 16bit. Để xử lý tình huống này, ta sử dụng lệnh *lui* với chức năng ghi một hằng số 16bit

vào 2 byte cao của thanh ghi, 2 byte thấp sẽ được gán bằng 0, và sẽ gán tạm vào $at(0x21100000). Sau đó sử dụng ori để ghép giá trị thanh $at và phần còn lại (0x0000003d) vào $s0 (0x21100000 OR 0x0000003d sẽ thành 0x2110003d).

Ở câu lệnh thứ hai, do 0x2 là giá trị quá nhỏ, không phù hợp với 16bit, nên máy sẽ chuyển về dạng addiu để phù hợp về kiểu dữ liệu.



Assignment 4



*Sự thay đổi các thanh ghi:

- Câu lệnh addi đầu tiên được dùng để gán giá trị cần tính 5 vào thanh ghi $t1 (0x00000005), thanh ghi PC sẽ trỏ vào địa chỉ câu lệnh kế tiếp (0x00400004).

- Câu lệnh addi thứ 2 được dùng để gán giá trị cần tính -1 vào thanh ghi $t2 (0xffffffff), thanh ghi PC sẽ trỏ vào địa chỉ câu lệnh kế tiếp (0x00400008).

- Câu lệnh thứ 3 cộng giá trị ở $t1 với chính nó và lưu vào $s0 nên $s0 có giá trị là 0x0000000a, thanh ghi PC sẽ trỏ vào địa chỉ câu lệnh kế tiếp (0x0040000c).

- Cuối cùng là cộng giá trị của $s0 và $t2 với nhau và lưu vào $s0 nên $s0 cuối cùng sẽ có giá trị là (0x00000009).

*Sau khi kết thúc chương trình, kết quả ở ô $s0 là 9 nên kết quả là chính xác.

*Với 2 lệnh addi:

Phân tích source theo khuôn mẫu:

- Lệnh 1: addi $t1, $zero, 5

Opcode: 8 => 001000

$rs: $zero => $0 => 00000

$rt: $t1 =>$9 => 01001

Imm: 5 => 0000 0000 0000 0101

- ⇨ Mã máy dạng nhị phân: 0010 0000 0000 1001 0000 0000 0000 0101
- ⇨ Dạng hex: 0x20090005

- Lệnh 2: addi $t2, $zero, -1

Opcode: 8 => 001000

$rs: $zero => $0 => 00000

$rt: $t2 =>$10 => 01010

Imm: -1 => 1111 1111 1111 1111

- ⇨ Mã máy dạng nhị phân: 0010 0000 0000 1010 1111 1111 1111 1111
- ⇨ Dạng hex: 0x200affff

=> Như vậy, đúng với khuôn mẫu kiểu lệnh I đã quy định trước.

*Với 2 lệnh add:

- Lệnh 1: add $s0, $t1, $t1

Opcode: 0 => 000000

$rs: $t1 =>$9 => 01001

$rt: $t1 =>$9 => 01001

$rd: $s0 =>$16 => 10000

shamt: 0 => 00000

funct: 32 => 100000

- ⇨ Mã máy dạng nhị phân: 0000 0001 0010 1001 1000 0000 0010 0000
- ⇨ Dạng hex: 0x01298020

- Lệnh 2: add $s0, $s0, $t2

Opcode: 0 => 000000

$rs: $s0 => $16 => 10000

$rt: $t2 => $10 => 01010

$rd: $s0 =>$16 => 10000

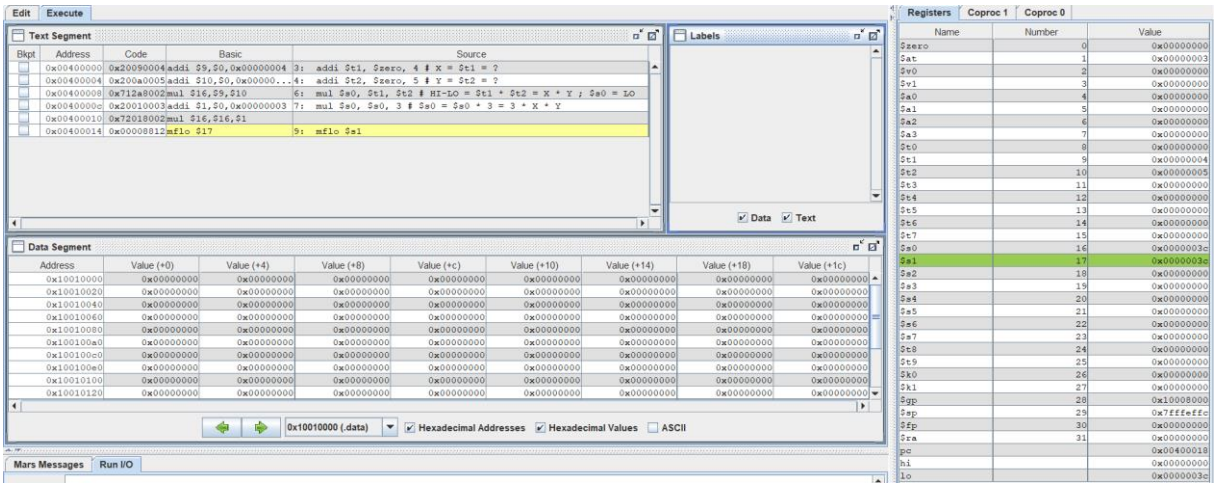
shamt: 0 => 00000

funct: 32 => 100000

- ⇨ Mã máy dạng nhị phân: 0000 0010 0000 1010 1000 0000 0010 0000
- ⇨ Dạng hex: 0x020a8020

=> Như vậy, đúng với khuôn mẫu kiểu lệnh R đã quy định trước.

Assignment 5



Bất thường: Lệnh mul $s0, $s0, 3 sau khi execute được tách ra thành addi $1, $0, 0x00000003 và mul $16, $16, $1.

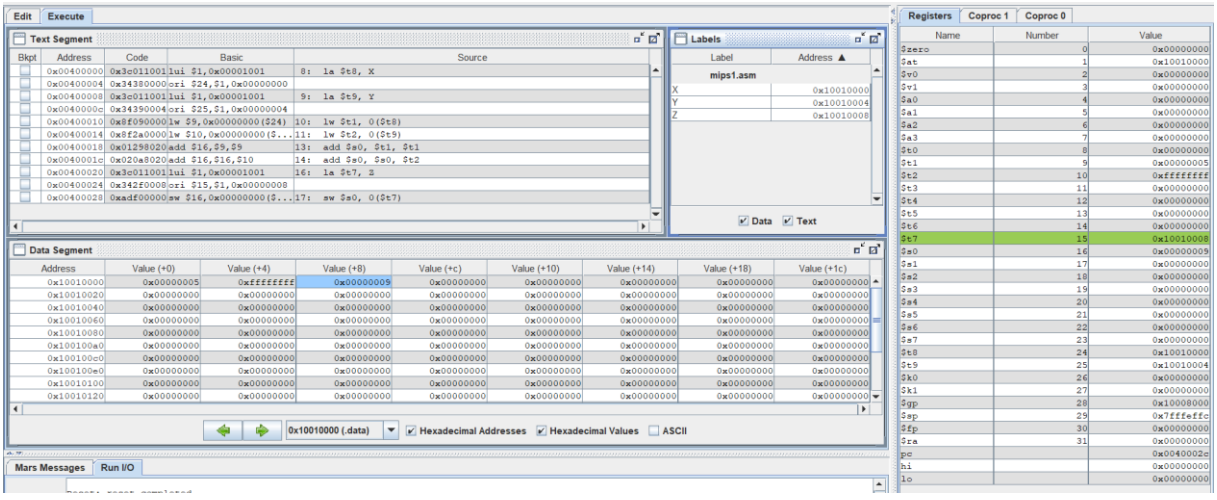
Giải thích: Do mul là lệnh thuộc khuôn dạng lệnh R, dạng mul $rd, $rs, $rt, không thể dùng immediate (3) được. Cho nên tách thành 2 lệnh. Lệnh addi $1, $0, 0x00000003 là gán tạm vào thanh ghi $1, sau đó hàm mul mới thực hiện phép nhân.

*Sự thay đổi thanh ghi hi, lo:

- Sau khi thực hiện lệnh mul $s0, $t1, $t2: Thanh ghi lo chứa kết quả phép nhân giá trị trong hai thanh ghi $t1 và $t2. (0x14).

- Sau khi thực hiện xong lệnh mul $s0, $s0, 3, khi đó thanh ghi lo thay đổi giá trị thành 0x3c (Kết quả phép nhân).

Assignment 6



- Lệnh ‘la’ được biên dịch thành 2 lệnh ‘lui’ và ‘ori’.

*Sự thay đổi các thanh ghi:

- Khi lệnh la $t8, X được thực thi, đầu tiên, địa chỉ sẽ được lưu tạm thời vào biến $at, thay đổi giá trị $at thành địa chỉ của X(0x10010000). Sau đó đưa về biến $t8, thay đổi thành giá trị địa chỉ X thanh ghi PC thay đổi thành 0x00400008.

- Khi lệnh la $t9, Y thực thi, đầu tiên, địa chỉ sẽ được lưu tạm thời vào biến $at, thay đổi giá trị $at thành (0x10010000). Sau đó $t8 thay đổi thành giá trị địa chỉ Y(0x10010004), thanh ghi PC thay đổi thành 0x00400010.

- Hai lệnh lw $t1, 0($t8) và lw $t2, 0($t9) làm thay đổi giá trị thanh ghi $t1 và $t2 lần lượt thành giá trị biến X(0x00000005) và Y (0xffffffff).

- Lệnh add $s0, $t1, $t1 thay đổi giá trị $s0 thành 0x0000000a, thanh ghi PC thay đổi thành 0x0040001c.

- Lệnh add $s0, $s0, $t2 thay đổi giá trị $s0 thành 0x00000009, thay đổi PC thành 0x00400020.

- Lệnh la $t7, Z thực thi, đầu tiên, giá trị $at thay đổi thành 0x10010000, sau đó, thay đổi hàm $t7 thành 0x10010008 (địa chỉ của Z), thanh ghi PC thay đổi thành 0x00400028.

- Lệnh sw $s0, 0($t7) đưa gán giá trị cho biến Z bằng giá trị của thanh ghi $s0, thanh ghi PC thay đổi thành 0x0040002c.

*Như vậy, vai trò của lệnh lw là tải dữ liệu từ biến vào một thanh ghi; vai trò của sw là lưu một dữ liệu từ thanh ghi vào biến chỉ định.