Lương Văn Khanh 20225728

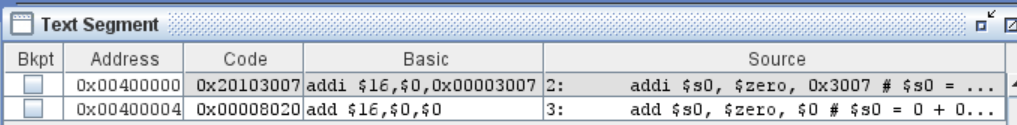
**EXERCISE 2**

Ass1:

* Quan sát cửa sổ Registers :

+ Sự thay đổi của giá trị thanh ghi $s0 : từ 0 tăng lên 0x3007 rồi về 0

+ Sự thay đổi của giá trị thanh ghi pc :từ 0x00400000 lên 0x00400004 rồi lên 0x00400008

* Cửa sổ Text Segment : 

|  |  |  |
| --- | --- | --- |
|  | addi $s0, $zero, 0x3007 | add $s0, $zero, $0 |
| Op code | 8 -> 001000 | 0->000000 |
| Rs | 0 -> 00000 | 0->00000 |
| Rt | 16 -> 10000 | 0->00000 |
| Rd |  | 16->10000 |
| Imm | 0x3007->0011000000000111 |  |
| Shamt |  | 0->00000 |
| Funct |  | 32->100000 |
| Code | 0010 0000 0001 0000 0011 0000 0000 0111 | 0000 0000 0000 0000 1000 0000 0010 0000 |
| Hex | 0x20103007 | 0x00008020 |

=>Đúng cấu trúc khuôn lệnh

-Sửa lại lệnh addi $s0, $zero, 0x2110003d

A screenshot of a computer

Description automatically generated

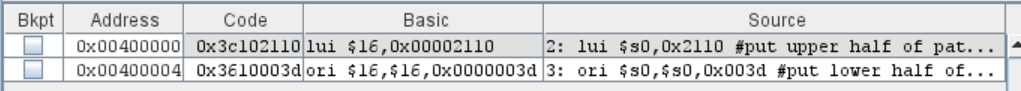
+Lệnh addi lúc trước chỉ dùng 1 lệnh basic

+Lệnh addi bây giờ sử dụng 2 lệnh là :

* lui để đưa 2byte đầu vào $s0
* ori để đưa 2byte sau vào $s0 thông qua biến phụ là $1

Lý do: theo khuôn lệnh I- Type, giá trị imm chỉ chứa được tối đa 16 bit – giá trị 0x2110003d chứa 32 bit nên MIPS cung cấp lệnh lui để cộng 16 bit đầu của imm vào 2 byte cao của $s0 và lệnh ori để cộng 16 bit còn lại vào 2 byte thấp

Ass2:



Sự thay đổi của thanh ghi $s0

* $s0 : 0x00000000 thành 0x21100000 ,do lui lấy 0x2110 gán vào nửa đầu $s0
* $0 : 0x21100000 thành  0x2110003d, do ori lấy 0x003d gán vào nửa sau $s0

Sự thay đổi của thanh ghi pc

* Thanh ghi pc thay đổi từ 0x00400000 thành 0x00400004 thành  0x00400008

A screenshot of a computer

Description automatically generated

Các byte đầu tiên trong vùng code trùng với value trong vùng data segment theo thứ tự từ trái qua phải, từ trên xuống dưới

Ass3:

A screenshot of a computer

Description automatically generated

* Lệnh li đầu tiên là lệnh giả, sử dụng 2 câu lệnh kết hợp
* Lệnh li tiếp theo chỉ sử dụng 1 câu lệnh basic

Lí do: Do khuôn mẫu của các lệnh kiểu I chỉ chứa được tối đa 16 bit

* Giá trị 0x2110003d chứa 32 bit nên lệnh li phải gán lần lượt 16 bit đầu và 16 bit cuối vào $s0
* Giá trị 0x2 chỉ chứa 1 bit, trong khoảng giá trị chứa được của 16 bit nên li sẽ gán trực tiếp giá trị 0x2 vào $s0

Ass4:

A screenshot of a computer

Description automatically generated

Mã máy : 0x20090005

001000|00000|01001|0000000000000101

I-Type

Opcode :  8=> addi

rs: 0 => $0

rt: 9=> $9

imm = 0x0005

Mã máy: 0x200affff

001000|00000|01010|1111111111111111

I-Type

Opcode: 8=> addi

Rs : 0 => $0

Rt: 10 => $10

Imm = 0xffff

Đúng với khuôn mẫu câu lệnh I

Mã máy :0x01298020

000000|01001|01001|10000|00000|100000

Opcode: 0=> R-Type

Rs : 9=> $9

Rt : 9 =>$9

Rd: 16=> $16

Shamt : 0

Funct: 32=> add

Mã máy: 0x020a8020

000000|10000|01010|10000|00000|100000

Opcode: 0=>R-Type

Rs : 16 => $16

Rt : 10 => $10

Rd : 16 => $16

Shamt: 0

Funct: 32 => add

Đúng với khuôn mẫu của kiểu lệnh R

Ass5:

A screenshot of a computer

Description automatically generated

A screenshot of a table

Description automatically generated

- Lệnh mul với 1 số bị thay thế bằng 1 lệnh khác, do lệnh mul là R-format, ta lại truyền vào là I-format, do đó chương trình phải them 1 lệnh khác để chuyển 0x00000003 thành 1 biến rồi mới thực hiện

|  |  |  |
| --- | --- | --- |
| Trạng thái | hi | lo |
| Ban đầu | 0x00000000 | 0x00000000 |
| Chạy add 1 | 0x00000000 | 0x00000000 |
| Chạy add 2 | 0x00000000 | 0x00000000 |
| Chạy add 3 | 0x00000000 | 0x00000014 |
| Chạy add 4 | 0x00000000 | 0x0000003c |

Kết quả cho ra đúng với phép toán: 3\*4\*5=60(0x0000003c)

Ass6:

Lệnh la được dịch thành 2 lệnA screenshot of a computer

Description automatically generatedh là lui và ori

A screenshot of a computer

Description automatically generated

Hợp ngữ **:Lui $1, 000010001**

**0x3c011001**

Mã máy: **001111|00000|00001|0000 0000 0001 0001**

Khi biên dịch la thì các hằng số trong mã máy trùng với địa chỉ của các biến X,Y,Z trong cửa sổ Lable

Sự thay đổi của các thanh ghi:

* Thanh ghi $t8 từ 0 thành **0x10010000** là địa chỉ của X
* Thanh ghi $t9 từ 0 thành **0x10001004** là địa chỉ của Y
* Thanh ghi $t1 từ 0 thành **5** là giá trị của X
* Thanh ghi $t2 từ 0 thành **-1** là giá trị của Y
* Thanh ghi $t7 từ 0 thành **0x100010008** là địa chỉ của Z => thanh ghi $t7 đang chứa địa chỉ của biến Z

Vai trò của lệnh lw là **đưa giá trị từ bộ nhớ vào thanh ghi**

Vai trò của lệnh sw là **đưa giá trị từ thanh ghi để lưu vào bộ nhớ**