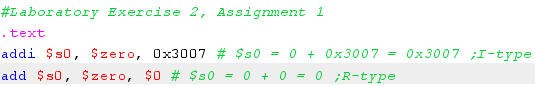
Họ và tên: Trần Quang Nam Trường Đại Học Bách Khoa Hà Nội

MSSV: 20184161 Viện CNTT&TT

**BẢN BÁO CÁO THỰC HÀNH**

**-----TUẦN 02-----**

**Bài 1.**

****

|  |  |  |  |
| --- | --- | --- | --- |
| Câu lệnh | Mã máy | Ý nghĩa | Chú thích |
| addi $s0, $zero, 0x3007  addi|$0|$16|0x3007  (Lệnh addi có mã 8) | 0x20103007  001000|00000|10000|0011000000000111 | Thực hiện phép cộng với thanh ghi nguồn là $zero và 1 hằng số, thanh ghi đích là $s0  $s0 = $zero + 0x3007 | - Đây là lệnh kiểu I với toán hạng tức thì (dữ liệu hằng số) là 0x3007  - Thanh ghi $zero luôn chứa hằng số 0  - Kết quả trả về: $s0 = 0x3007 |
| add $s0, $zero, $0  0|$0|$0|$16|0|add  (Lệnh add có mã 32) | 0x00008020  000000|00000|00000|10000|00000|100000 | Thực hiện phép cộng với thanh ghi nguồn là $zero và $s0, thanh ghi đích là $s0  $s0 = $zero + $s0 | - Đây là lệnh kiểu R  - Kết quả trả về: $s0 = 0 (do không khai báo .data nên dữ liệu $s0 từ câu lệnh trước sẽ không được lưu lại) |

* Sự thay đổi giá trị của thanh ghi $s0 như sau:

0x00003007 - sau khi thực hiện lệnh addi $s0, $zero, 0x3007

0x00000000 - sau khi thực hiện lệnh add $s0, $zero, $0

* Mã máy của các lệnh trên đúng như khuôn dạng lệnh của tập lệnh đã định
* Thanh ghi $pc lần lượt lưu lại địa chỉ của các lệnh sẽ được thực hiện, giá trị thay đổi như sau:

0x00400000 - địa chỉ của lệnh “addi $s0, $zero, 0x3007”)

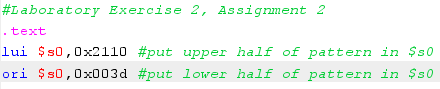
0x00400004 - địa chỉ của lệnh “add $s0, $zero, $0”)

0x00400008 - địa chỉ này không có câu lệnh để thực thi nên kết thúc chương trình

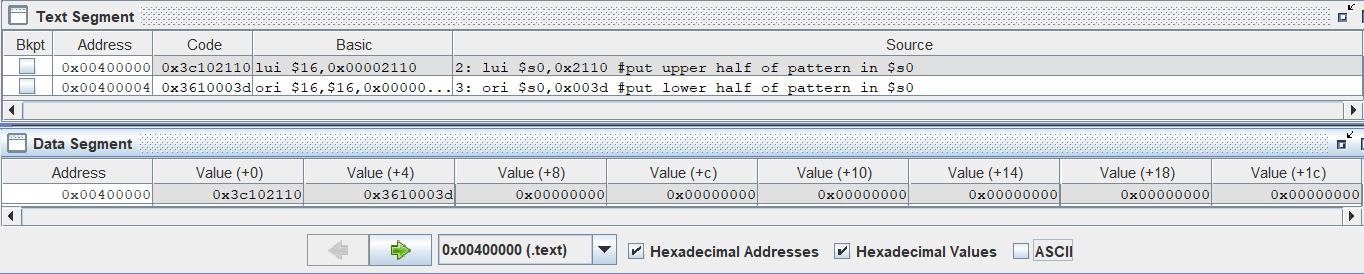
* Sau khi thay đổi sang câu lệnh mới thì từ câu lệnh hợp ngữ ban đầu đã bị tách thành 3 câu lệnh basic mới như sau:

|  |  |  |
| --- | --- | --- |
| Câu lệnh | Basic Code | Ý nghĩa |
| addi $s0, $zero, 0x2110003d | lui $1, 0x00002110 | Đưa 16 bit cao của hằng số 32-bit vào 16 bit trái của thanh ghi $at  Xóa 16 bits bên phải của $at về 0 |
| ori $1, $1, 0x0000003d | Đưa 16 bit thấp của hằng số 32-bit vào thanh ghi $at |
| add $16, $0, $1 | - Thực hiện phép cộng với thanh ghi nguồn là $0 và $1, thanh ghi đích là $16  - Kết quả trả về là hằng số 32-bit 0x2110003d |

**Bài 2.**

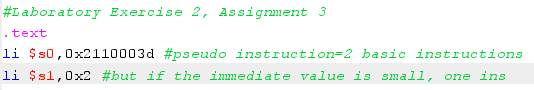


|  |  |  |
| --- | --- | --- |
| Câu lệnh | Ý nghĩa | Giá trị $s0 sau khi thực hiện lệnh |
| lui $s0,0x2110 | Đưa 16 bit cao của hằng số 32-bit vào 16 bit trái của thanh ghi $at  Xóa 16 bits bên phải của $s0 về 0 | 0x21100000 |
| ori $s0,0x003d | Đưa 16 bit thấp của hằng số 32-bit vào thanh ghi $s0 | 0x2110003d |



* Các byte đầu tiên ở vùng lệnh của cửa sổ Data Segment trùng với cột Code trong cửa sổ Text Segment

**Bài 3.**

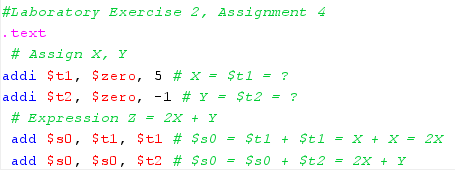
****

Lệnh gán li là lệnh giả (pseudo instruction) để lập trình viên sử dụng. Vì vậy khi chạy chương trình thì lệnh này sẽ được chuyển về ngôn ngữ máy (basic instruction) theo quy định trong tập lệnh

|  |  |  |
| --- | --- | --- |
| Lệnh giả (Pseudo Ins) | Basic Ins | Giải thích |
| li $s0,0x2110003d | lui $1, 0x00002110 | Trường hợp này thanh ghi $s0 được nạp vào hằng số 32-bit là 0x2110003d nên phải chuyển đổi lần lượt thành lệnh lui và ori |
| ori $16, $1, 0x0000003d |
| li $s1,0x2 | addiu $17, $0, 0x00000002 | Trường hợp này thanh ghi $s0 được nạp vào hằng số nhỏ, thấp hơn 16-bit là 0x00000002 nên chỉ cần chuyển đổi thành lệnh addi/ addiu |

* Vì addi/ addiu là lệnh kiểu I nên hằng số sẽ chỉ chứa được trong 16-bit. Vì vậy khi muốn nạp một giá trị hằng số cần nhiều hơn 16-bit vào thanh ghi ta không dùng được lệnh addi/ addiu nữa mà phải dùng tổ hợp lệnh lui và ori.

**Bài 4.**

****

|  |  |  |
| --- | --- | --- |
| Câu lệnh | Ý nghĩa | Kết quả |
| addi $t1, $zero, 5 | - $t1 đại diện cho biến X  - Thực hiện phép cộng: $t1 = $zero + 5 | - X = 5  - Thanh ghi $t1 có giá trị 0x00000005 |
| addi $t2, $zero, -1 | - $t2 đại diện cho biến Y  - Thực hiện phép cộng: $t2 = $zero + (-1) | - Y = -1  - Thanh ghi $t2 có giá trị 0xffffffff |
| add $s0, $t1, $t1 | Thực hiện phép cộng $s0 = $t1 + $t1 | - X + X = 10  - Thanh ghi $s0 có giá trị 0x0000000a |
| add $s0, $s0, $t2 | Thực hiện phép cộng $s0 = $s0 + $t2 | - Thanh ghi $s0 có giá trị 0x00000009 |

* Vậy kết quả sau khi thực hiện chương trình tính 2X + Y = 9 với X = 5, Y = -1 là đúng

|  |  |  |
| --- | --- | --- |
| Câu lệnh hợp ngữ | Mã máy | Kiểu lệnh |
| addi $t1, $zero, 5  addi|$0|$9|5 | 0x20090005  001000|00000|01001|0000000000000101 | I |
| addi $t2, $zero, -1  addi|$0|$10|-1 | 0x200affff  001000|00000|01010|1111111111111111 |
| add $s0, $t1, $t1  0|$9|$9|$16|0|add | 0x01298020  000000|01001|01001|10000|00000|100000 | R |
| add $s0, $s0, $t2  0|$16|$10|$16|0|add | 0x020a8020  000000|10000|01010|10000|00000|100000 |

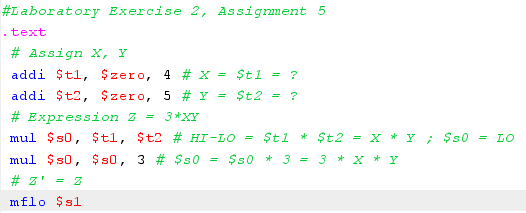
* Vậy khuôn mẫu của cả 2 kiểu lệnh I và R đều chính xác

Mỗi câu lệnh trong hợp ngữ đều được đánh một mã số riêng, khi chuyển về ngôn ngữ máy thì mã số đó được chuyển về dạng nhị phân, ví dụ

- Lệnh add có mã 32 (100000 trong hệ cơ số 2)

- Lệnh addi có mã 8 (1000 trong hệ cơ số 2)

**Bài 5.**



|  |  |  |
| --- | --- | --- |
| Câu lệnh | Giải thích | Kết quả |
| addi $t1, $zero, 4 | - $t1 đại diện cho biến X  - Thực hiện phép cộng: $t1 = $zero + 4 | - X = 4  - Thanh ghi $t1 có giá trị 0x00000004 |
| addi $t2, $zero, 5 | - $t2 đại diện cho biến Y  - Thực hiện phép cộng: $t2 = $zero + 5 | - Y = 5  - Thanh ghi $t2 có giá trị 0x00000005 |
| mul $s0, $t1, $t2  (1) | Thực hiện phép nhân $s0 = $t1 \* $t2 (\*) | - X\*Y = 20  - Thanh ghi $hi không thay đổi giá trị  - Thanh ghi $lo có giá trị 0x00000014  - Thanh ghi $s0 có giá trị 0x00000014 |
| mul $s0, $s0, 3  (2) | Thực hiện phép nhân $s0 = $s0 \* 3 | - 3\*X\*Y = 60  - Thanh ghi $at có giá trị 0x00000003  - Thanh ghi $hi không thay đổi giá trị  - Thanh ghi $lo có giá trị 0x0000003c  - Thanh ghi $s0 có giá trị 0x0000003c |
| mflo $s1 | Dịch chuyển giá trị từ thanh ghi $lo đến thanh ghi $s1 | - Thanh ghi $s1 có giá trị 0x0000003c |

* Vậy kết quả sau khi thực hiện chương trình tính 3\*X\*Y = 60 với X = 4, Y = 5 là đúng
* Ở lệnh nhân đầu tiên (1): mul $s0, $t1, $t2

Nhân 2 giá trị được gán ở $t1 và $t2 với nhau (4\*5), kết quả nhận được là 20 (14 trong hệ 16) nên chỉ dùng 2 bit trong thanh ghi $lo để ghi lại giá trị, không cần dùng đến thanh ghi $hi

* Ở lệnh nhân sau (2): mul $s0, $s0, 3

Gấp ba giá trị $s0 nhận được ở lệnh nhân (1) (3\*20) và ghi đè lại kết quả lên thanh ghi $s0

Vì vậy, để thực hiện lệnh này máy cần thực hiện tổ hợp lệnh mới là:

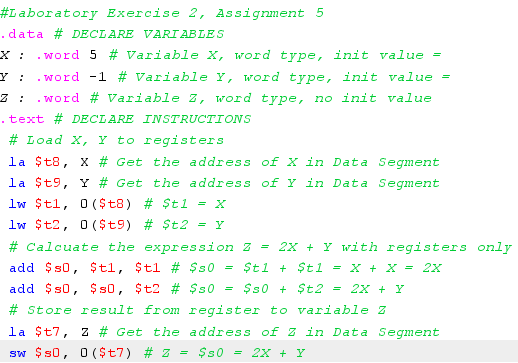
- addi $1, $0, 0x00000003 - gán giá trị tạm thời cho thanh ghi $at là 3

- mul $16, $16, $1 - thực hiện phép nhân $at\*$s0 (3\*20)

Lúc này, kết quả là 60 (3c trong hệ 16) được lưu vào thanh ghi $lo, sau đó lấy giá trị từ thanh ghi $lo ghi đè vào thanh $s0. Giá trị thanh ghi $s0 lúc này sẽ thay đổi từ 20 thành 60

Cần phải làm như vậy vì ta cần giữ lại giá trị ban đầu của thanh ghi $s0 là 20 để thực hiện phép nhân 3, vì vậy ta sử dụng thanh ghi $at để tạo 1 giá trị tạm thời là 3

**Bài 6.**

****

* Lệnh la có tác dụng lấy địa chỉ của một biến được khai báo từ phần .data và ghi vào thanh ghi đích

Bởi vì địa chỉ của biến được khai báo là 1 số 32-bit nên khi biên dịch lệnh la, máy sẽ chạy tổ hợp lệnh lui và ori, sử dụng thanh ghi $at làm bộ nhớ tạm thời. Sau khi đã nạp được địa chỉ cơ sở của biến vào thanh ghi $at thì sẽ tực hiện phép tính địa chỉ của biến rồi lưu địa chỉ đó vào thanh ghi đích

|  |  |  |
| --- | --- | --- |
| Biến | Địa chỉ | Thanh ghi và giá trị lưu trữ trong thanh ghi |
| X | 0x10010000 | $t8 0x10010000 |
| Y | 0x10010004 | $t9 0x10010004 |
| Z | 0x10010008 | $t7 0x10010008 |

* Sự thay đổi giá trị các thanh ghi

|  |  |  |
| --- | --- | --- |
| Câu lệnh | Ý nghĩa | Giá trị của các thanh ghi |
| la $t8, X | Lấy địa chỉ của biến X và ghi vào thanh ghi $t8 | - Thanh ghi $at có giá trị 0x10010000  - Thanh ghi $t8 có giá trị 0x10010000 |
| la $t9, Y | Lấy địa chỉ của biến Y và ghi vào thanh ghi $t9 | - Thanh ghi $at có giá trị 0x10010000  - Thanh ghi $t9 có giá trị 0x10010004 (mỗi word lưu trong 4 byte nhớ) |
| lw $t1, 0($t8) | Gán giá trị của $t1 bằng giá trị của X là 5 | - Thanh ghi $t1 có giá trị 0x00000005 |
| lw $t2, 0($t9) | Gán giá trị của $t2 bằng giá trị của Y là -1 | - Thanh ghi $t2 có giá trị 0xffffffff |
| add $s0, $t1, $t1 | Thực hiện phép cộng:  $s0 = $t1 + $t1 = 5 + 5 = 10 | - Thanh ghi $s0 có giá trị 0x0000000a |
| add $s0, $s0, $t2 | Thực hiện phép cộng:  $s0 = $s0 + $t2 = 10 + (-1) = 9 | - Thanh ghi $s0 có giá trị 0x00000009 |
| la $t7, Z | Lấy địa chỉ của biến Z và ghi vào thanh ghi $t7 | - Thanh ghi $at có giá trị 0x10010000  - Thanh ghi $t7 có giá trị 0x10010008 |
| sw $s0, 0($t7) | Gán giá trị của thanh ghi $s0 vào biến Z, lúc này Z = 9 |  |

* Lệnh lw được sử dụng để đọc word dữ liệu 32-bit từ bộ nhớ và đưa vào thanh ghi, cấu trúc lệnh như sau:

lw rt, imm(rs)

rs: thanh ghi chứa địa chỉ cơ sở (base address)

imm (immediate): hằng số (offset)

=> địa chỉ của word dữ liệu cần đọc = địa chỉ cơ sở + hằng số

rt: thanh ghi đích, chứa word dữ liệu được đọc vào

* Lệnh sw được sử dụng để ghi word dữ liệu 32-bit từ thanh ghi đưa ra bộ nhớ, cấu trúc lệnh như sau:

sw rt, imm(rs)

rt: thanh ghi nguồn, chứa word dữ liệu cần ghi ra bộ nhớ

rs: thanh ghi chứa địa chỉ cơ sở (base address)

imm (immediate): hằng số (offset)

=> địa chỉ nơi ghi word dữ liệu = địa chỉ cơ sở + hằng số

* Lệnh lb được sử dụng để đọc một byte từ bộ nhớ và mở rộng thành số có dấu 32-bit rồi gán giá trị đó vào thanh ghi đích. Bit đầu tiên là 1 thì đó là số dương, bit đầu tiên là 0 thì đó là số âm. Cấu trúc lệnh như sau:

lb rt, offset(rs)

rt: thanh ghi đích, chứa dữ liệu được đọc vào

rs: thanh ghi chứa địa chỉ cơ sở (base address)

=> địa chỉ nơi lưu dữ liệu cần đọc = địa chỉ cơ sở + hằng số (offset)

* Lệnh sb được sử dụng để lưu 8-bit thấp từ thanh ghi nguồn vào bộ nhớ. Cấu trúc lệnh như sau:

sb rt, offset(rs)

rt: thanh ghi nguồn, chứa 8-bit dữ liệu cần ghi ra bộ nhớ

rs: thanh ghi chứa địa chỉ cơ sở (base address)

=> địa chỉ nơi ghi dữ liệu = địa chỉ cơ sở + hằng số (offset)