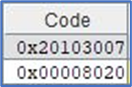
**Báo cáo thực hành tuần 2**

**1.Assignment 1: lệnh gán số 16-bit**

**-** Sự thay đổi thanh ghi $s0 và thanh ghi $pc

+ $s0: Sau lệnh 1 có giá trị = 0x00003007;Sau lệnh 2 có giá trị=0x00000000

+  $pc: Trước lệnh 1, $pc lưu địa chỉ của lệnh 1 là 0x00400000; Sau lệnh 1, $pc lưu địa chỉ của lệnh 2 là 0x00400004;Sau lệnh 2 có giá trị là 0x00400008

* Mã máy ở cửa sổ Text Segment:
* So sánh hai mã máy:

0x20103007: 001000 00000 10000 0011000000000111

opcode: 8 => Lệnh addi

rs: 0

rt: 16 => $s0

imm: 0x3007

0x00008020: 000000 00000 00000 10000 00000 100000

opcode: 0, function: 32 => Lệnh add

rs: 0

rt: 0

rd: 16 => $s0

sht: 0

* Sửa lại lệnh lui thành addi $s0, $zero, 0x2110003d

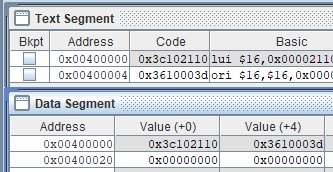
Do hằng số 0x2110003d vượt quá 16 bit nên câu lệnh trên được tách thành 2 câu lệnh "lui $s1, 0x00002110" và "ori $1, $1, 0x0000003d". Lệnh lui ghi 2110 của 0x2110003d vào 2 byte cao thanh ghi $s1 đồng thời 2 byte thấp của thanh ghi $s1 sẽ được gán bằng 0. Sau đó, chạy lệnh ori để ghi 2 byte thấp của 0x2110003d vào thanh ghi $s1. Cuối cùng gán giá trị của thanh ghi $s1 sang $s0.

**2.Assignment 2: lệnh gán số 32-bit**

- Sự thay đổi của thanh ghi $s0 và $pc

|  |  |  |
| --- | --- | --- |
|  | 0x3c102110 | 0x3610003d |
| $s0 | 0x21100000 | 0x2110003d |
| $spc | 0x00400004 | 0x00400008 |

* Byte đầu tiên của vùng lệnh trùng với cột Code trong Text Segment:



**3.Assignment 3: lệnh gán (giả lệnh)**

Graphical user interface, text, application, email

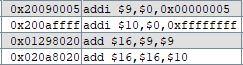
Description automatically generated

Khi chạy lệnh li $s0, 0x2110003d thì sẽ được tách thành 2 lệnh lui $1, 0x00002110 và lệnh ori $16, $1, 0x0000003d. Đầu tiên, lệnh lui ghi 2110 của 0x2110003d vào 2 byte cao thanh ghi $s1 đồng thời 2 byte thấp của thanh ghi $s1 sẽ được gán bằng 0. Sau đó, chạy lệnh ori để thực hiện phép toán OR: $16 = $1 | 0x0000003d.

Khi chạy lệnh li $s1,0x2 thì sẽ chạy lệnh addiu $17, $0, 0x00000002.

Sự khác biệt này do 0x02 có 16 bit, 0x2110003d có 32 bit (vượt quá 16 bit)

**4.Assignment 4: Tính biểu thức 2x+y**



**-** Sự thay đổi thanh ghi:

|  |  |
| --- | --- |
| $t1 | 0x00000005 |
| $t2 | 0xffffffff |
| $s0 | 0x0000000a |
| $s0 | 0x00000009 |

- Chương trình chạy đúng kết quả

- Kiểm nghiệm với khuôn mẫu :

0x20090005 : 001000 00000 01001 0000000000000101

opcode: 8 => Lệnh addi

rs: 0 => $zero

rt: 9 => $t1

imm: 0x0005

0x200affff : 001000 00000 01010 1111111111111111

opcode: 8 => Lệnh addi

rs: 0 => $zero

rt: 10 => $t2

imm: 0xffff

* Thỏa mãn kiểu lệnh I

0x01298020 : 000000 01001 01001 10000 00000 100000

opcode: 0, function: 32 => Lệnh add

rs: 9 =>$t1

rt: 9 => $t1

rd: 16 => $s0

sht: 0

0x020a8020: 000000 10000 01010 10000 00000 100000

opcode: 0, function: 32 => Lệnh add

rs: 16 => $s0

rt: 10 => $t2

rd: 16 => $s0

sht: 0

* Thỏa mãn kiểu lệnh R

**5**. **Assignment 5: phép nhân**Graphical user interface, application, Word

Description automatically generated

- Điều khác thường: Khi chạy lệnh mul $s0, $s0, 3 thì máy sẽ gọi ra 2 lệnh là addi $1, $0, 0x0000003 và lệnh mul $16, $16, $1. Vì lệnh mul thuộc kiểu R nên khi dùng theo kiểu I phải gán vào thanh ghi $1 sau đó mới thực hiện lệnh mul.

- Sự thay đổi của thanh HI và LO.

- Ở câu lệnh mul $s0, $t1, $t2 giá trị của phép nhân là 14 (hệ hexa) nằm trong khoảng 0 đến 31 nên được ghi vào thanh LO, còn thanh HI không có giá trị.

- Ở câu lệnh mul $s0, $s0, 3 giá trị của phép nhân là 3c (hệ hexa) vẫn nằm trong khoảng 0 đến 31 nên vẫn được ghi vào thanh LO, còn thanh HI không có giá trị.

- Chương trình chạy đúng

# 6.Assignment 6: tạo biến và truy cập biến

* Lệnh giả la được biên dịch thành cặp lệnh lui và ori.
* Địa chỉ của X, Y,Z tương ứng với hằng số khi biên dịch lệnh la
* Sự thay đổi giá trị của các thanh ghi

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| $t1 | $t2 | $t7 | $s0 | $t8 | $t9 |
| 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x10010000 | 0x00000000 |
| 0x00000000 | 0x00000000 | 0x00000000 | 0x00000000 | 0x10010000 | 0x10010004 |
| 0x00000005 | 0x00000000 | 0x00000000 | 0x00000000 | 0x10010000 | 0x10010004 |
| 0x00000005 | 0xffffffff | 0x00000000 | 0x00000000 | 0x10010000 | 0x10010004 |
| 0x00000005 | 0xffffffff | 0x00000000 | 0x0000000a | 0x10010000 | 0x10010004 |
| 0x00000005 | 0xffffffff | 0x00000000 | 0x00000009 | 0x10010000 | 0x10010004 |
| 0x00000005 | 0xffffffff | 0x10010008 | 0x00000009 | 0x10010000 | 0x10010004 |
| 0x00000005 | 0xffffffff | 0x10010008 | 0x00000009 | 0x10010000 | 0x10010004 |

* Vai trò của lệnh lw và sw:
* Lệnh lw load giá trị từ bộ nhớ (từ một địa chỉ xác định) rồi lưu vào thanh ghi.
* Lệnh sw lưu giá trị từ một thanh ghi vào bộ nhớ.
* Lệnh lb và sb: chức năng tương tự lw và sw nhưng sử dụng để load và store giá trị 1 byte chứ không phải 1 word (4 bytes).