**Laboratory Exercise 2**

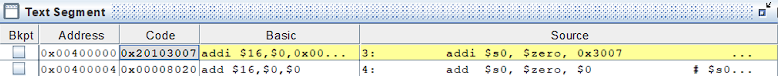
Họ và tên: Đỗ Mạnh Phương MSSV:20225660

**Assignment 1 : Lệnh gán số 16-bit**

A screenshot of a computer

Description automatically generated

* Sự thay đổi của thanh ghi $s0:
  + Ban đầu thanh ghi $s0 có giá trị 0x00000000.
  + Sau lệnh addi, giá trị thanh ghi đổi thành 0x00003007
  + Sau lệnh add, giá trị thanh ghi đổi thành 0x00000000
* Sự thanh đổi của thanh ghi pc:
  + Ban đầu thanh ghi pc có giá trị 0x00400000
  + Sau lệnh addi, giá trị thanh ghi trở thành 0x00400004
  + Sau lệnh add, gái trị thanh ghi trở thành 0x00400008
* Ở cửa sổ Text Segment, dễ dàng nhận thấy mã máy của các lệnh trên trùng với khuôn dạng lệnh. Điều đó chứng tỏ các lệnh đó đúng như các tập lệnh đã quy định.



A screenshot of a computer

Description automatically generated

* Khi sửa lại lệnh lui như đề bài yêu cầu, ta thấy lệnh addi đã được tách thành 3 lệnh là lui, ori và add.
  + Giải thích: Ở câu hỏi trước, chúng ta làm việc với các số 16-bit. Đây cũng là giới hạn hằng số của các lệnh thao tác với hằng số. Vì vậy khi gặp 1 số 32-bit là 0x2110003d, MIPS đã tự động cung cấp lệnh lui với chức năng ghi hằng số 16-bit vào 2 byte cao của thanh ghi và 2 byte thấp sẽ được gán thành 0.

**Assignment 2: lệnh gắn số 32-bit**

A screenshot of a computer

Description automatically generated

* Sự thay đổi của thanh ghi $s0:
  + Ban đầu thanh ghi $s0 có giá trị 0x00000000
  + Sau lệnh lui, $s0 có giá trị 0x21100000
  + Sau lệnh ori, $s0 có giá trị 0x2110003d
* Sự thay đổi của thanh ghi pc:
  + Ban đầu thanh ghi pc có giá trị 0x00400000
  + Sau lệnh lui, giá trị pc đổi thành 0x00400004
  + Sau lệnh ori, giá trị của pc đổi thành 0x00400008
* Các byte đầu tiên của vùng lệnh trùng với cột Code trong cửa sổ Text Segment.

A white rectangular box with black text

Description automatically generated 

**Assignment 3: lệnh gán (giả lệnh)**

A screenshot of a computer program

Description automatically generated

* Ở đây ta thấy lệnh li $s0,0x2110003d bị chia thành 2 lệnh lui và ori, còn lệnh li $s1,0x2 biến thành lệnh addiu.
* Giải thích:
  + Lệnh li là 1 lệnh gán thực hiện gán giá trị ngay lập tức vào thanh ghi. Vì đây là 1 giả lệnh nên khi chạy nó sẽ được MIPS biên dịch thành 1 hoặc nhiều lệnh cơ bản.
  + Ở lệnh li $s0,0x2110003d; giá trị 0x2110003d được gán lập tức vào thanh ghi $s0. Vì giá trị này có độ dài 32-bit nên sẽ được tách thành 2 lệnh là lui và ori.
  + Ở lệnh li $s1,0x2 lập tức được gán vào thanh ghi $s1. Vì giá trị này có độ dài 16-bit và đồng thời không có dấu nên được hiểu là lệnh addiu.

**Assignment 4: tính biểu thức 2x + y = ?**

**A screenshot of a computer

Description automatically generated**

* Sự thay đổi giá trị của các thanh ghi
  + Ở lệnh addi $t1, $zero, 4; giá trị $t1 có giá trị là 0x00000004
  + Ở lệnh addi $t2, $zero, 5; giá trị $t2 có giá trị là 0x00000005
  + Ở lệnh mul $s0, $t1, $t2; giá trị lo có giá trị là 0x00000014, giá trị $s0 cũng vậy, giá trị hi là 0x00000000
  + Ở lệnh mul $s0,$s0,3; giá trị lo có giá trị 0x0000003c, giá trị $s0 cũng vậy, giá trị hi không thay đổi
  + Ở lệnh mflo $s1, giá trị của $s1 chính là giá trị của $s0
* Kết quả đúng với các giá trị x,y tương ứng.
* Lệnh addi $t1, $zero, 5 :
  + Có mã 0x20090005.
  + Với mã nhị phân tương ứng là: “0010 00|00 000|0 1001| 0000 0000 0000 0101”
* Dựa vào mã nhị phân ta thấy:
  + op: “0010 00(8)” tương ứng với opcode của lệnh addi.
  + rs: “00 000” tương ứng với thanh ghi $zero.
  + rt: “0 1001” tương ứng với thanh ghi $t1
  + immediate: “0000 0000 0000 0101” tương ứng với giá trị hằng số 5.
* Lệnh addi $t2, $zero, -1 :
  + Có mã 0x200affff
  + Với mã nhị phân tương ứng là: “0010 00|00 000|0 1010| 1111 1111 1111 1111”
* Dựa vào mã nhị phân ta thấy:
  + op: “0010 00(8)” tương ứng với opcode của lệnh addi.
  + rs: “00 000” tương ứng với thanh ghi zero.
  + rt: “0 1010” tương ứng với thanh ghi $t2.
  + immediate: “1111 1111 1111 1111” tương ứng với hằng số -1.
* Nhận xét: Ta thấy hợp ngữ có những điểm tương đồng với mã máy và lệnh addi tuân theo khuôn mẫu kiểu lệnh I.
* Lệnh add $16, $9, $9 (add $s0, $t1, $t1) có:
  + op: “000000”
  + rd: “10000”
  + rs: “10001”
  + rt: “10001”
  + shamt: “00000”
  + fn: “100000”
  + Mã nhị phân tương ứng: 0000 0001 0010 1001 1000 0000 0010 0000.
  + Mã hex: 0x01298020.
* Lệnh add $16, $16, $10(add $s0, $s0, $t2) có:
  + op: “000000”
  + rd: “10000”
  + rs: “10000”
  + rt: “01010”
  + shamt: “00000”
  + fn: “100000”
  + Mã nhị phân tương ứng: 0000 0010 0000 1010 1000 0000 0010 0000
  + Mã hex: 0x020a8020
* Kết luận: lệnh add phù hợp với R-format.

**Assignment 5: phép nhân**

**A screenshot of a computer

Description automatically generated**

* Ta thấy lệnh mul được chuyển thành các lệnh khác do lệnh mul là R-format nhưng khi gõ lệnh lại là I-format nên cần thêm một câu lệnh để máy có thể thực hiện được.
* Sự thay đổi giá trị của các thanh ghi :
  + Ở lệnh addi $t1, $zero, 4; giá trị $t1 có giá trị là 0x00000004
  + Ở lệnh addi $t2, $zero, 5; giá trị $t2 có giá trị là 0x00000005
  + Ở lệnh mul $s0, $t1, $t2; giá trị lo có giá trị là 0x00000014, giá trị $s0 cũng vậy, giá trị hi là 0x00000000
  + Ở lệnh mul $s0,$s0,3; giá trị lo có giá trị 0x0000003c, giá trị $s0 cũng vậy, giá trị hi không thay đổi
  + Ở lệnh mflo $s1, giá trị của $s1 chính là giá trị của $s0
* Kết quả đúng với các giá trị x,y tương ứng.

**Assignment 6: tạo biến và truy cập biến**

**A screenshot of a computer

Description automatically generated**

* Lệnh la do chỉ lưu được địa chỉ 16 bit mà địa chỉ của biến là 32 lên la sẽ thực hiện 2 câu lệnh lui và ori để lưu địa chỉ của biến.

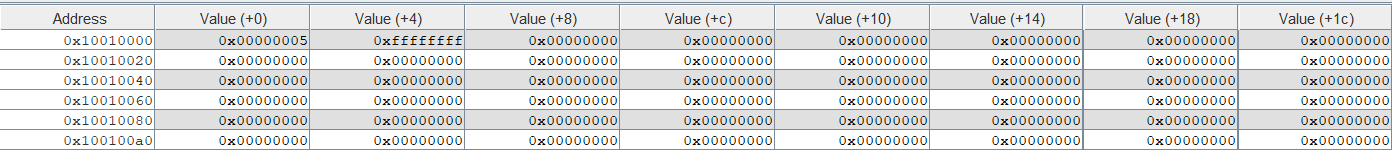
**A screenshot of a computer

Description automatically generated**

* Ta có các giá trị của X,Y và Z sau 3 lần chạy lệnh la như sau:
  + X: A screenshot of a table

    Description automatically generated
  + Y: A table with numbers and symbols

    Description automatically generated
  + Z:



* Sự thay đổi giá trị của các thanh ghi:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trạng thái | $s0 | $t1 | $t2 | $t7 | $t8 | $t9 |
| Ban đầu | 0x000000 00 | 0x000000 00 | 0x000000 00 | 0x000000 00 | 0x000000 00 | 0x000000 00 |
| Chạy lần 1 |  |  |  |  | 0x100100 00 |  |
| Chạy lần 2 |  |  |  |  |  | 0x100100 04 |
| Chạy lw 1 |  | 0x000000 05 |  |  |  |  |
| Chạy lw 2 |  |  | 0xffffffff |  |  |  |
| Chạy add 1 | 0x000000 0a |  |  |  |  |  |
| Chạy add 2 | 0x000000 09 |  |  |  |  |  |
| Chạy la |  |  |  | 0x100100 08 |  |  |
| Chạy sw |  |  |  |  |  |  |

* Vai trò của các lệnh lw, sw: lw dùng để load word hay đọc word dữ liệu 32-bit từ bộ nhớ đưa vào thanh ghi, sw dùng để store word hay để ghi word dữ liệu 32-bit từ thanh ghi đưa ra bộ nhớ.