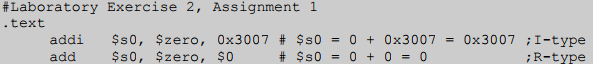
Laboratory Exercise 2

**Assignment 1: lệnh gán 16-bit**



* Sự thay đổi của thanh ghi $s0 và $pc
* Ban đầu: $s0 = 0x00000000 ; $pc = 0x00400000
* Thực hiện lệnh thứ nhất: $s0 = 0x00003007 ; $pc = 0x00400004
* Thực hiện lệnh thứ hai: $s0 = 0x00000000 ; $pc = 0x00400008
* *Mã máy các câu lệnh*

addi $s0, $zero, 0x3007

0x20103007: 001000 00000 10000 0011000000000111‬ ;I-type ‬

op: 001000 (Lệnh addi)

rs: 00000 (Thanh ghi $zero)

rt: 10000 (Thanh ghi $s0)

operan/ofset: 0011000000000111 (0x3007)

add $s0, $zero, $0

0x00008020: 000000 00000 00000 10000 00000 100000 ;R-type

op: 000000 (Chỉ dẫn R-type op luôn bằng 0)

rs: 00000 (Thanh ghi $zero)

rt: 00000 (Thanh ghi $zero)

rd: 10000 (Thanh ghi $s0)

sh: 00000 (shamt = 0)

fn: 100000 (funct = 32 mã lệnh add)

🡪 Các câu lệnh đó đúng theo tập lệnh đã quy định.

* Sửa lại lệnh lui thành :

add $s0, $zero, 0x2110003d

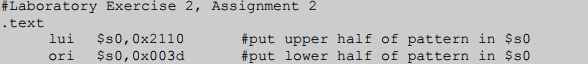
* Vì hằng số trong lệnh là hằng số 32-bit nên lệnh trên được thay thế bằng 3 lệnh :

lui $1, 0x00002110 *#copy 16 bit cao vào 16 bit trái của $at*

ori $1, $1, 0x0000003d *#đưa 16 bit thấp của hằng số vào thanh ghi $at*

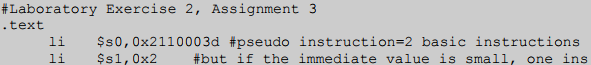
add $16, $0, $1 *#$s0 = $zero + $at*

**Assignment 2: lệnh gán số 32-bit**



* Sự thay đổi của thanh ghi $s0 và $pc
* Ban đầu: $s0 = 0x00000000 ; $pc = 0x00400000
* Thực hiện lệnh thứ nhất: $s0 = 0x21100000 ; $pc = 0x00400004
* Thực hiện lệnh thứ hai : $s0 = 0x2110003d ; $pc = 0x00400008
* Ớ cửa sổ Data Segment, quan sát các byte trong vùng lệnh .text
* Các byte đầu tiên ở vùng lệnh trùng với cột Code của cửa sổ Text Segment

**Assignment 3: lệnh gán (giả lệnh)**



* Khi biên dịch, vì hằng số là hằng số 32-bit và lệnh li là lệnh giả nên lệnh li thứ nhất tương đương với 2 lệnh cơ bản:

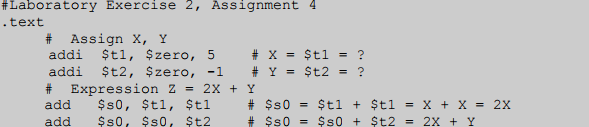
lui $1, 0x00002110

ori $16, $1, 0x0000003d

* Vì hằng số nhỏ, nên lệnh li thứ hai tương đương với một lệnh:

addiu $17, $0, 0x00000002

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

****

* Sự thay đổi của các thanh ghi:
* Sau lệnh addi đầu tiên: $t1 = 0x00000005
* Sau lệnh addi thứ 2: $t2 = 0xffffffff
* Sau lệnh add đầu tiên: $s0 = 0x0000000a
* Sau lệnh add thứ 2: $s0 = 0x00000009
* Sau khi kết thúc chương trình, $s0 lưu giá trị 9 nên kết quả chạy đúng.
* Kiểm nghiệm mã máy

addi $t1, $zero, 5 # X = $t1 = 5

0x20090005 : 001000 00000 01001 0000000000000101

op: 001000 (Lệnh addi)

rs: 00000 ($zero)

rt: 01001 ($9 ⬄ $t1)

opran/offset: 0000000000000101 (Biểu diễn nhị phân 16 bit của 5)

addi $t2, $zero, -1 # Y = $t2 = -1

0x200affff: 001000 00000 01010 1111111111111111

op: 001000 (Lệnh addi)

rs: 00000 ($zero)

rt: 01001 ($10 ⬄ $t2)

opran/offset: 1111111111111111 (Biểu diễn nhị phân 16 bit của -1)

add $s0, $t1, $t1 # $s0 = $t1 + $t1 = X + X = 2X

0x01298020: 000000 01001 01001 10000 00000 100000

op: 000000 (Chỉ dẫn R-type op luôn bằng 0)

rs: 01001 (Thanh ghi $9 ⬄ $t1)

rt: 01001 (Thanh ghi $9 ⬄ $t1)

rd: 10000 (Thanh ghi $16 ⬄ $s0)

sh: 00000 (shamt = 0)

fn: 100000 (funct = 32 mã lệnh add)

add $s0, $s0, $t2 # $s0 = $s0 + $t2 = 2X + Y

0x020a8020: 000000 10000 01010 10000 00000 100000

op: 000000 (Chỉ dẫn R-type op luôn bằng 0)

rs: 10000 (Thanh ghi $16 ⬄ $s0)

rt: 01010 (Thanh ghi $10 ⬄ $t2)

rd: 10000 (Thanh ghi $16 ⬄ $s0)

sh: 00000 (shamt = 0)

fn: 100000 (funct = 32 mã lệnh add)

**Assignment 5: phép nhân**



* Điều bất thường: câu lệnh ***mul $s0, $s0, 3*** trong source được chuyển thành 2 câu lệnh

addi $1, $0, 0x00000003

mul $16, 16, $s1

* Do lệnh mul basic yêu cầu toán hạng là 2 thanh ghi, không chạy với hằng số nên hằng số sẽ được lưu vào thanh ghi $1 trước khi thực hiện nhân thanh ghi $1 với $s0.
* Sự thay đổi giá trị các thanh ghi:
* Thanh ghi *hi* luôn bằng 0, thanh ghi *lo* thay đổi giá trị và luôn bằng *$s0* bằng kết quả phép nhân.
* Kết quả chính xác, sau phép nhân ***mul $s0, $t1, $t2***, giá trị $s0 = 20; sau phép nhân ***mul $s0, $s0, 3*** giá trị $s0 = 60.

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



* Lệnh la được biên dịch thành 2 lệnh lui và ori
* Địa chỉ của mỗi biến được tách làm 2 phần 16 bit cao và 16 bit thấp mỗi phần tương ứng với hằng số truyền vào lệnh la sau biên dịch 16 bit cao cho lệnh lui, 16 bit thấp cho lệnh ori

Giá trị khởi tạo X = 5; Y = -1; Z = 0; Đảm bảo đúng giá trị khởi tạo mong muốn.

* Sự thay đổi giá trị của các thanh ghi:

$at: 0 -> 0x10010000 -> 0x10010000 -> 0x10010000 -> 0x10010000

$t8: 0 -> 0x00000000 -> 0x10010000

$t9: 0 -> 0x00000000 -> 0x00000000 -> 0x00000000 -> 0x10010004

$t1: 0 -> 0 -> 0 -> 0 -> 0 -> 5

$t1: 0 -> 0 -> 0 -> 0 -> 0 -> 0 -> -1

$s0: 0 -> … -> 10 -> 9

$t7: 0 -> .. -> 0x10010008

* Vai trò:

lw: load giá trị từ bộ nhớ chính vào thanh ghi

sw: lưu giá trị từ thanh ghi sang bộ nhớ chính