**BÁO CÁO THỰC HÀNH KIẾN TRÚC MÁY TÍNH (IT3280) TUẦN 4**

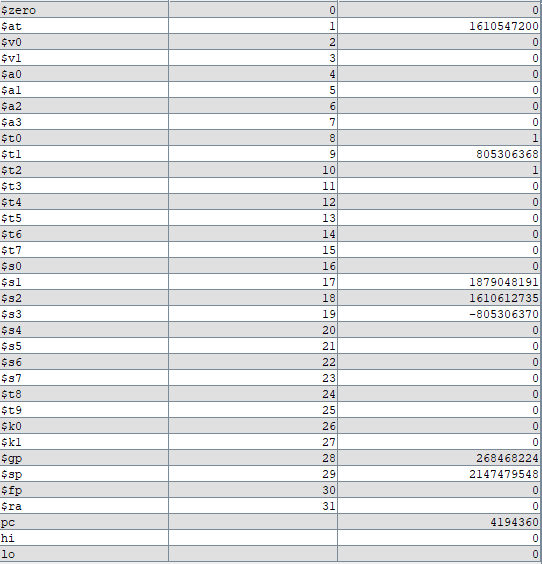
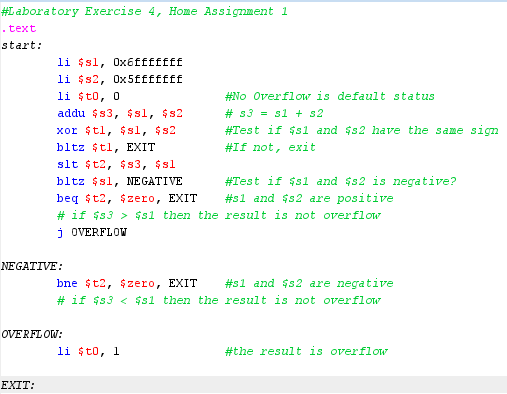
***Họ và tên: Phạm Đức Long***

***MSSV: 20225737***

Assignment 1:

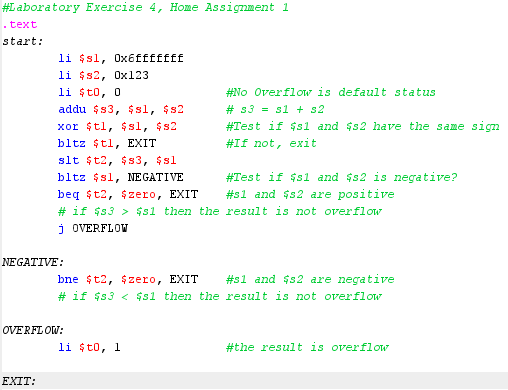
Ở bài này, ta xét 5 trường hợp của s1 và s2 bao gồm:

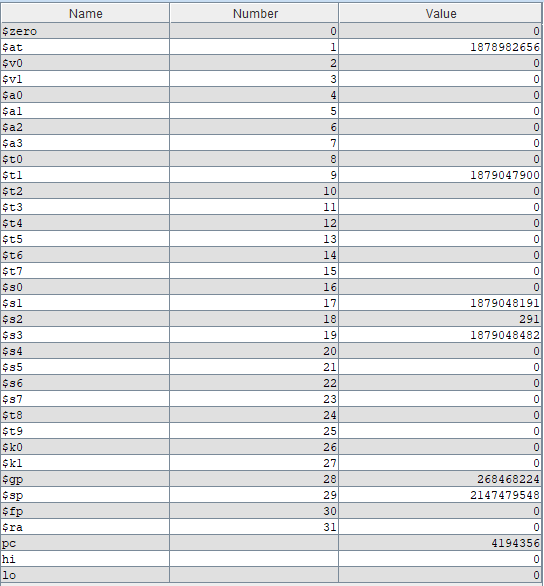
* TH1: $s1, $s2 dương; $s3 tràn:



Nhận xét: Đặt $s1 = 0x6fffffff và $s2 = 0x5fffffff khi đó $s3 = $s1 + $s2 sẽ có kết quả là một số âm, phép **xor $t1, $s1, $s2** sẽ có kết quả là một số dương, do đó chương trình sẽ tiếp tục chạy đến câu lệnh **slt $t2, $s3, $s1**.Lúc này $s3 < $s1 nên $t2 = 1, do đó chương trình sẽ thực hiện tiếp câu lệnh **j OVERFLOW**, nhảy đến nhãn OVERFLOWvà gán giá trị của thanh $t0 = 1 => tràn số.

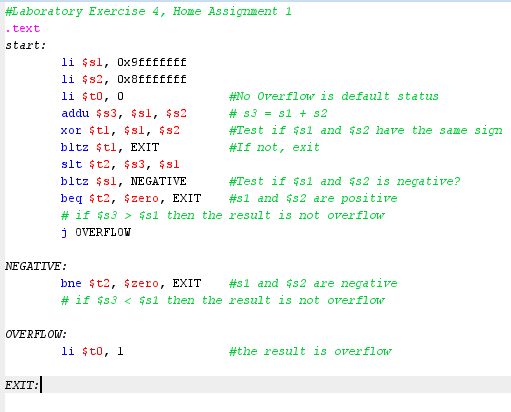
* TH2: $s1, $s2 dương; $s3 không tràn:

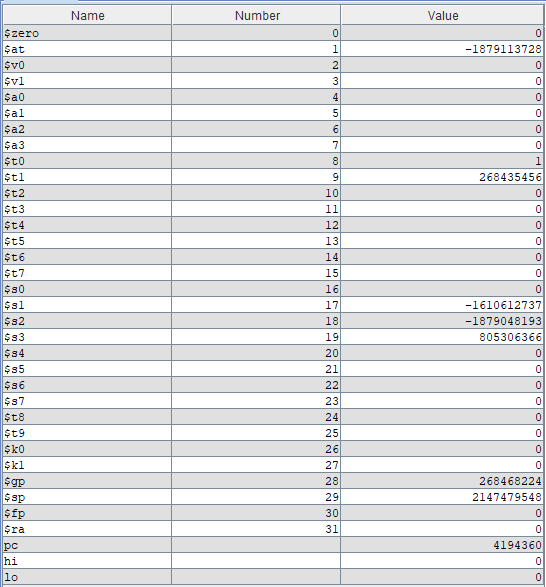




Nhận xét: Đặt $s1 = 0x6fffffff và $s2 = 0x123 khi đó $s3 = $s1 + $s2 sẽ có kết quả là một số dương, phép **xor $t1, $s1, $s2** sẽ có kết quả là một số dương, do đó chương trình sẽ tiếp tục chạy đến câu lệnh **slt $t2, $s3, $s1**.Nhưng lúc này $s3 > $s1 nên $t2 = 0, do đó chương trình sẽ thực hiện tiếp câu lệnh **beq $t2, $zero, EXIT**, nhảy đến nhãn EXIT, giá trị của thanh $t0 = 0 => không xảy ra tràn số.

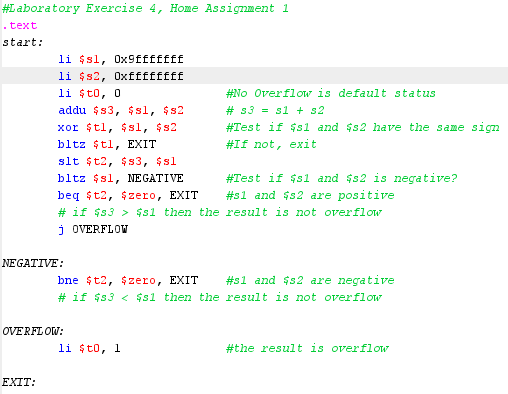
* TH3: $s1, $s2 âm, $s3 tràn:

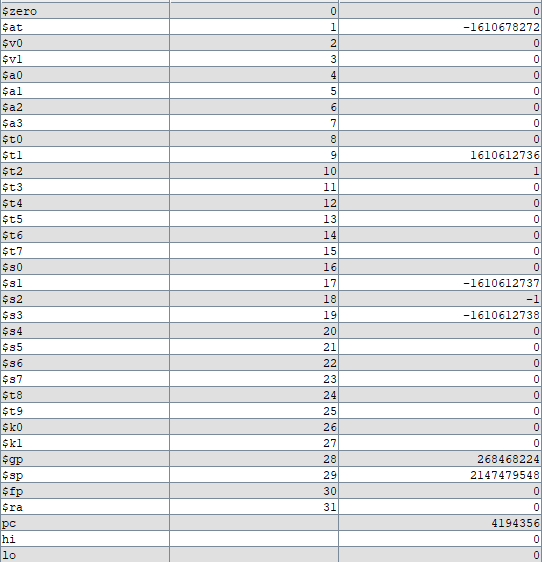




Nhận xét: Đặt $s1 = 0x9fffffff và $s2 = 0x8fffffff khi đó $s3 = $s1 + $s2 sẽ có kết quả là một số dương, phép **xor $t1, $s1, $s2** sẽ có kết quả là một số dương, do đó chương trình sẽ tiếp tục chạy đến câu lệnh **slt $t2, $s3, $s1**.Lúc này $s3 > $s1 nên $t2 = 0, mặt khác chương trình sẽ chuyển qua nhãn NEGATIVE do lệnh **bltz $s1, NEGATIVE**. Lúc này chương trình sẽ kiểm tra $t2 có bằng 0 hay không, nhưng $t2 = 1 nên chương trình sẽ nhảy xuống nhãn OVERFLOW và gán $t0 = 1 => xảy ra tràn dấu.

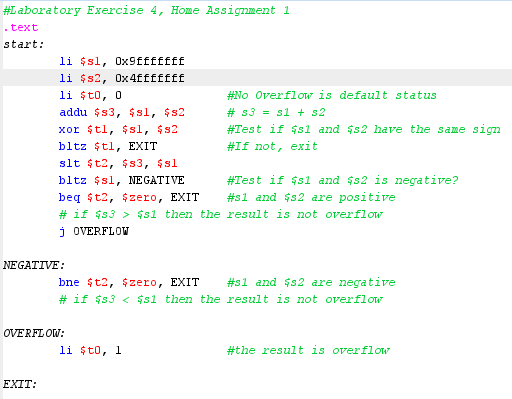
* TH4: $s1, $s2 âm, $s3 không tràn:

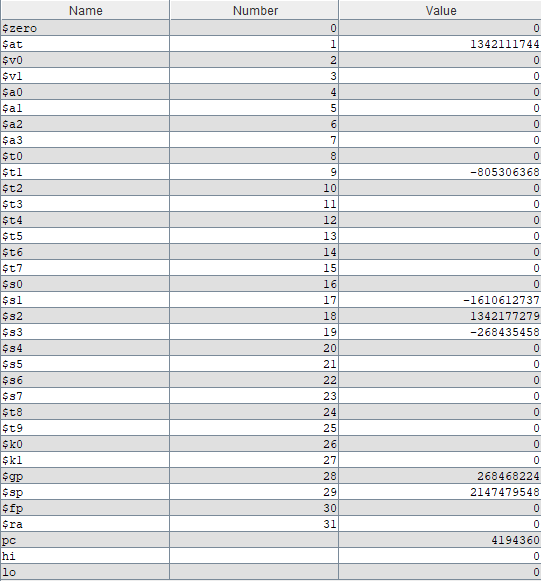




Nhận xét: Đặt $s1 = 0x9fffffff và $s2 = 0xffffffff khi đó $s3 = $s1 + $s2 sẽ có kết quả là một số âm, phép **xor $t1, $s1, $s2** sẽ có kết quả là một số dương, do đó chương trình sẽ tiếp tục chạy đến câu lệnh **slt $t2, $s3, $s1**.Lúcnày $s3 < $s1 nên $t2 = 1, nhưng sau đó chương trình sẽ kiểm tra ra $s1 là số âm qua lệnh bltz $s1, NEGATIVE và chuyển xuống nhãn NEGATIVE. Lúc này chương trình sẽ thực hiện lệnh bne $t2, $zero, EXIT để kiểm tra xem $t2 != 0 không và $t2 có khác 0, nên chương trình sẽ chuyển đến nhãn EXIT, $t0 = 0 => Không xảy ra tràn dấu.

* TH5: $s1 dương, $s2 âm ( TH $s1 và $s2 trái dấu nói chung):





Nhận xét: Gán $s1 = 0x9fffffff, $s2 = 0x4fffffff, lúc này $s3 = $s1 + $s2 < 0, phép **xor $t1, $s1, $s2** sẽ có kết quả là một số âm, do đó chương trình sẽ chuyển đến nhãn EXIT để kết thúc chương trình do câu lệnh **bltz $t1, EXIT**, $t0 = 0 => Không xảy ra tràn số.

* **Tổng kết**:
* Lệnh addu giúp có thể lưu giá trị khi kết quả phép cộng nằm ngoài vùng giá trị của thanh ghi.
* Hai số cùng dấu thì phép xor của hai số đó sẽ luôn có giá trị là một số dương, và ngược lại sẽ có giá trị là số âm nếu hai số trái dấu.
* Tổng của hai số trái dấu với nhau không thể là một giá trị tràn số.
* Tổng của hai số dương sau lệnh addu có giá trị âm thì sẽ xảy ra tràn số, tổng của hai số âm sau lệnh addu có giá trị dương thì cũng sẽ xảy ra tràn số.

Assignment 2:

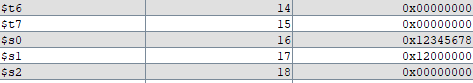
1. Extract MSB of $s0

* Code:

**.text**

**li $s0, 0x12345678**

**andi $s1, $s0, 0xff000000**

****

1. Clear LSB of $s0

* Code:

**.text**

**li $s0, 0x12345678**

**andi $s1, $s0, 0xffffff00**

****

1. Set LSB of $s0 (bits 7 to 0 are set to 1)

* Code:

**.text**

**li $s0, 0x12345678**

**ori $s1, $s0, 0x000000ff**

****

1. Clear $s0 ($s0=0, must use logical instructions)

* Code:

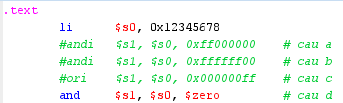
**.text**

**li $s0, 0x12345678**

**and $s1, $s0, $zero**

****

* **Tổng kết cả 4 câu:**

****

Assignment 3:

1. abs $s0,$s1

* Code:

**.text**

**li $s1, -10**

**slt $t0, $s1, $zero # $t0 = $s1 < 0 ? 1 : 0**

**beqz $t0, ABS # $t0 = 0 => ABS**

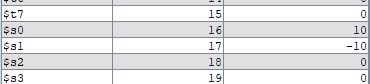
**sub $s0, $zero, $s1 # else $s1 = 0 - $s1**

**j END**

**ABS:**

**add $s0, $s1, $zero # $s0 = | $s1 |**

**END:**

****

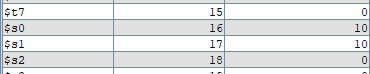
1. move $s0,$s1

* Code:

**.text**

**li $s1, 10**

**add $s0, $s1, $zero # $s0 = $s1**

****

1. not $s0, $s1

* Code:

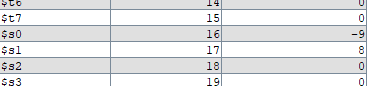
**.text**

**li $s1, 8**

**addi $s0, $s1, 1 # $s0 = $s1 + 1**

**sub $s0, $zero, $s0 # $s0 = 0 - $s0**

**# not $s0, $s1**

****

1. ble $s1,$s2,label

* Code:

**.text**

**li $s1, 3**

**li $s2, 5**

**sle $t0, $s1, $s2 # $t0 = $s1 <= $s2 ? 1 : 0**

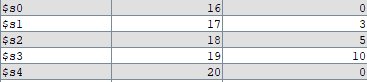
**bnez $t0, label**

**j exit**

**label:**

**li $s3, 10 # neu $s1 <= $s2 thi $s3 = 10**

**exit:**

****

Assignment 4:

* Code:

**.text**

**li $s1, 0x4fffffff**

**li $s2, 0x5fffffff**

**li $t0, 0**

**addu $s3, $s2, $s1 # $s3 = $s2 + $s1**

**xor $t1, $s1, $s2**

**# kiem tra xem $s1 va $s2 co cung dau khong, neu co $t1 > 0**

**bltz $t1, exit # $t1 < 0 thi khong tran dau**

**xor $t2, $s3, $s1**

**# kiem tra xem $s1 va $s3 co cung dau khong, neu trai dau => $t2 < 0 => tran so**

**bgtz $t2, exit**

**# neu $t2 > 0 thi $s3 va $s1 cung dau => khong tran so**

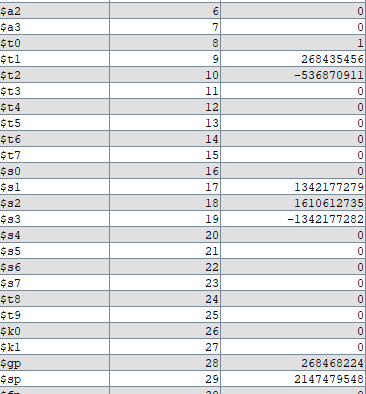
**overflow:**

**li $t0, 1**

**exit:**

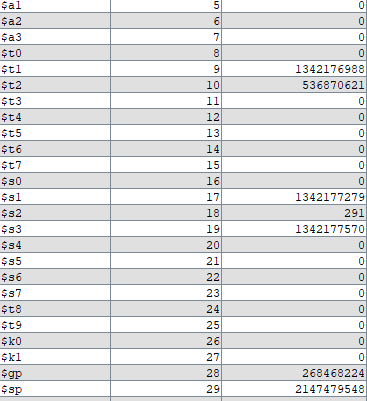
Ta xét hai trường hợp không tràn số và có tràn số:

* TH1: $s1 = 0x4fffffff và $s2 = 0x5fffffff => Có tràn số

****

Nhận xét: Theo lý thuyết, có hiện tượng tràn số xảy ra nếu ta đặt $s1 = 0x4fffffff và $s2 = 0x5fffffff. Mặt khác, giá trị của $t0 = 1 => thỏa mãn lý thuyết.

* TH2: $s1, 0x4fffffff và $s2, 0x123



Nhận xét: Theo lý thuyết sẽ không có hiện tượng tràn số xảy ra nếu ta đặt $s1 = 0x4fffffff và $s2 = 0x123. Mặt khác, giá trị của $t0 = 0 => thỏa mãn lý thuyết.

* TH3: $s1 = 0x4fffffff, $s2 = 0x9fffffff

A table with numbers and letters

Description automatically generated

Nhận xét: Theo lý thuyết sẽ không có hiện tượng tràn số xảy ra nếu ta đặt $s1 = 0x4fffffff và $s2 = 0x9fffffff. Mặt khác, giá trị của $t0 = 0 => thỏa mãn lý thuyết.

* **Kết luận:** Quy tắc đề bài đưa ra là hoàn toàn đúng.

Assignment 5:

**.text**

**li $s0, 3 # dat so can nhan la x = 3**

**li $t1, 1 # buoc nhay la 1**

**li $t2, 5**

**# se co 4 vong lap tu 1 den 4, tuong ung voi viec ta**

**# se nhan x voi 2, 4, 8, 16.**

**loop:**

**sllv $s0, $s0, $t1**

**# phep lui $t1 bit, dong nghia voi phep nhan voi luy thua cua 2**

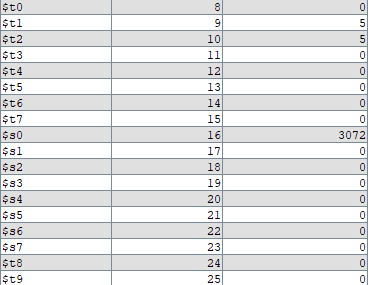
**addi $t1, $t1, 1 # $t1 = $t1 + 1**

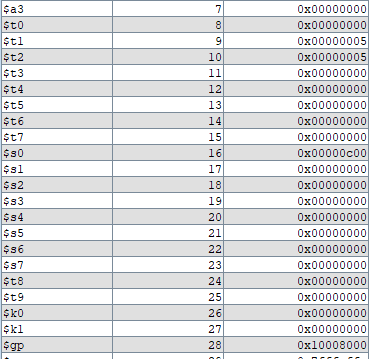
**sub $s1, $t1, $t2 # $s1 = $t1 - $t1**

**beqz $s1, endloop # neu $s1 = 0 thi dung vong lap**

**j loop**

**endloop:**





|  |  |  |  |
| --- | --- | --- | --- |
| Trạng thái | $t1 | $s0 | $s1 |
| Sau khi khởi tạo các giá trị | 0x00000001 | 0x00000003 | 0x00000000 |
| Sau khi chạy hết vòng for đầu tiên | 0x00000002 | 0x00000006 | 0xfffffffd |
| Sau khi chạy hết vòng for thứ hai | 0x00000003 | 0x00000018 | 0xfffffffe |
| Sau khi chạy hết vòng for thứ ba | 0x00000004 | 0x000000c0 | 0xffffffff |
| Sau khi chạy hết vòng for thứ ba | 0x00000005 | 0x00000c00 | 0x00000000 |

Nhận xét: Theo như tính toán, kết quả cuối cùng thu được trên thanh $s0 sẽ có giá trị là 3 \* 2 \* 4 \* 8 \* 16 = 3072. Mặt khác giá trị trên thanh $s0 sau khi chạy xong chương trình có giá trị là 0x00000c00 = 3072 => Kết quả thu được sau khi chạy chương trình hoàn toàn đúng với kết quả tính toán được trên lý thuyết.



* Code:

**# Chuong trinh nhan mot so x voi mot so luy thua cua 2:**

**.text**

**li $s1, 3 # dat x = 3**

**li $s2, 5**

**# $s2 = 5 dong nghia voi dich trai 5 bit,**

**# tuong duong phep nhan voi 32**

**sllv $s3, $s1, $s2**

**# dich bit cua $s1 sang trai 5 lan => nhan $s1 voi 32**

**A screenshot of a computer

Description automatically generated**

* Nhận xét: Giá trị trên thanh ghi $t3 = 96 đúng với phép nhân 3 \* 32 => Chương trình cho ra kết quả đúng với kết quả tính toán theo lý thuyết.

**P/s:** Ở bài 5 do em không chắc là mình đã hiểu đúng đề bài hay không nên em xin phép chia bài này thanh hai câu. Ở câu a) em làm là nhân một số x với các số lũy thừa của 2, ví dụ như x \* 2 \* 4 \* 8 \* 16 \* bằng phép dịch trái… Còn ở câu b) em làm là nhân một số x với một số lũy thừa của 2 bằng phép dịch trái.

**Tổng kết**:

* **SLLV**: Thực hiện phép dịch trái bit của giá trị trong một thanh ghi nguồn theo một số lượng bit được chỉ định bởi giá trị trong thanh ghi nguồn thứ hai.
* **SLL**: Thực hiện phép dịch trái bit của giá trị trong một thanh ghi theo một số lượng bit cố định được chỉ định trực tiếp trong lệnh.
* **SRLV:** Ngược lại với lệnh SLLV, lệnh này thực hiện phép dịch phải bit của giá trị trong một thanh ghi nguồn (source register) theo một số lượng bit được chỉ định bởi giá trị trong thanh ghi nguồn thứ hai (shift amount).
* **SRL:** Ngược lại với lệnh SLL, lệnh này thực hiện phép dịch phải bit của giá trị trong một thanh ghi theo một số lượng bit cố định được chỉ định trực tiếp trong lệnh.