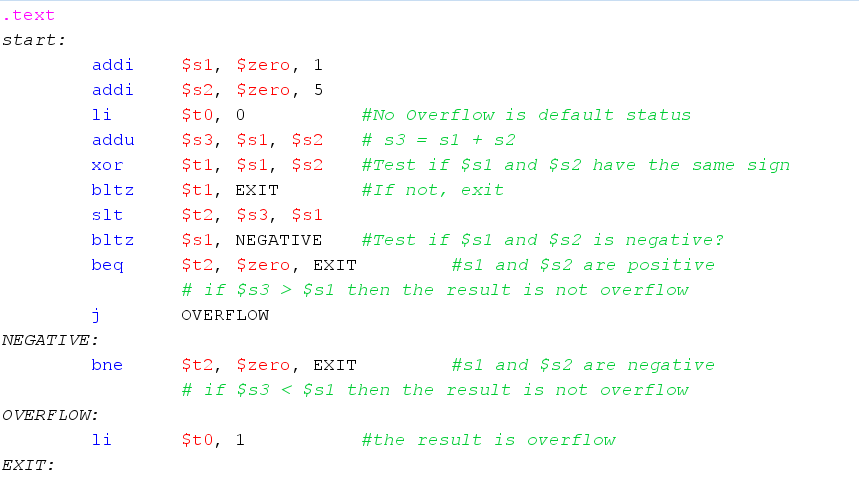
**Báo cáo thực hành tuần 4**

Họ tên: Kiều Đăng Nam

MSSV: 20176830

**Asignment 1**

****

1. Trường hợp 1, s1, s2 đều là số nguyên dương, không overlow

s1 = 1, s2 = 5, s3 = s1 + s2 = 6



T1 = xor s1, s2 = 4 -> cùng dấu



T2 = slt s3, s1 = 0 (s3 > s1), s1, s2 dương

1. Trường hợp 1, s1, s2 đều là số nguyên dương, overlow



s1 = 0x7fffffff, s2 = 0x7fffffff, s3 vượt quá 32bit -> overflow.



T1 = xor s1, s2 =000…000-> cùng dấu



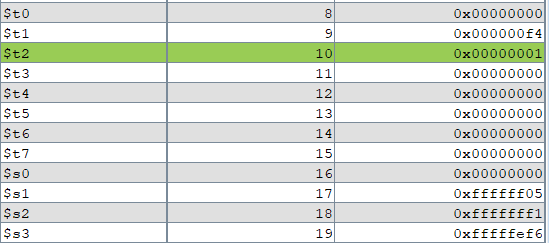
T2 = slt s3, s1 = 1 (s3 > s1), do bị overflow

T0 = 1 -> overflow.

1. Trường hợp 1, s1, s2 đều là số nguyên âm, không overlow



s1 = 0xffffff05, s2 = 0xfffffff1 = -1, s3 không quá 32 bit



T1 = 000…f4 -> s1, s2 cùng dấu

T2 = 1 (s3 < s1)

T0 = 0 do không bị overflow

1. Trường hợp 1, s1, s2 đều là số nguyên âm, overlow

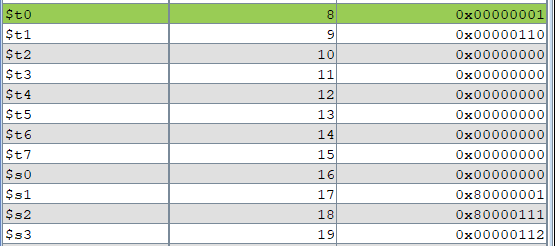


s1 = 0x80000001, s2 = 0x80000111, s3 vượt quá 32bit -> overflow.

T1 = xor s1, s2 = 00..0110 -> cùng dấu

T2 = slt s3, s1 = 0 (s3 >= s1), do bị overflow

T0 = 1 -> overflow.



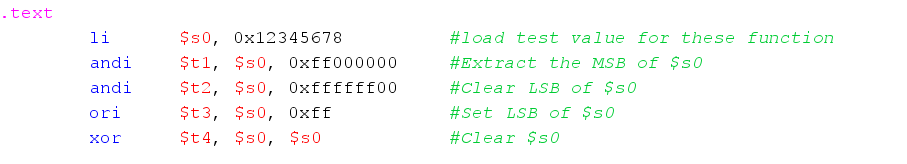
1. Trường hợp s1, s2 trái dấu



T1 = 0xff… =111… -> s1, s2 trái dấu -> EXIT



**Asignment 2:**

****

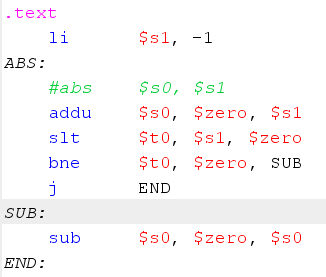
Kết quả:

1. Extract MSB of $s0: Để extract MSB, ta cần andi $0 với hằng số có giá trị 2 byte đầu là 1. Tức: 0xff000000
2. Clear LSB of $s0: Clear LSB tức ta đưa tất cả giá trị LSB của $s0 về 0, vì vậy ta sẽ andi LSB của $s0 với 0, tức AND $s0 với 0xffffff00
3. Set LSB of $s0: Set LSB tức ta đưa tất cả giá trị LSB của $s0 về 1, vì vậy ta sẽ ori LSB của $0 với 1, tức OR $s0 với 0xff
4. Clear $0: xor $s0 với chính nó



**Asignment 3**

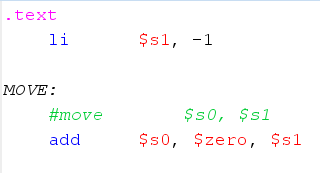
* 1. abs:



Kết quả:



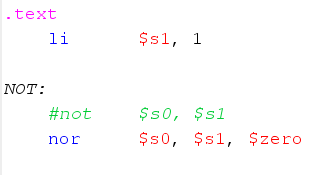
* 1. move:



Kết quả:



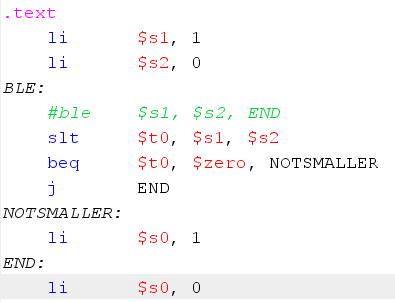
* 1. not:



Kết quả:



* 1. ble:

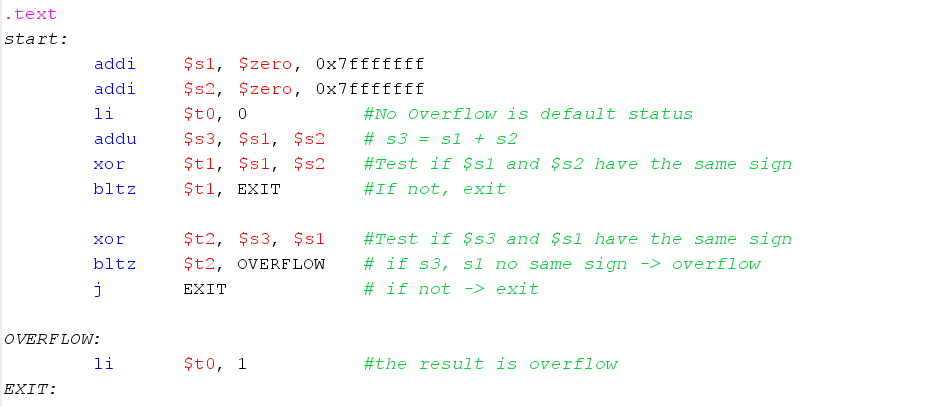


Kết quả:



**Assignment 4:**

Chương trình phát hiện overflow:

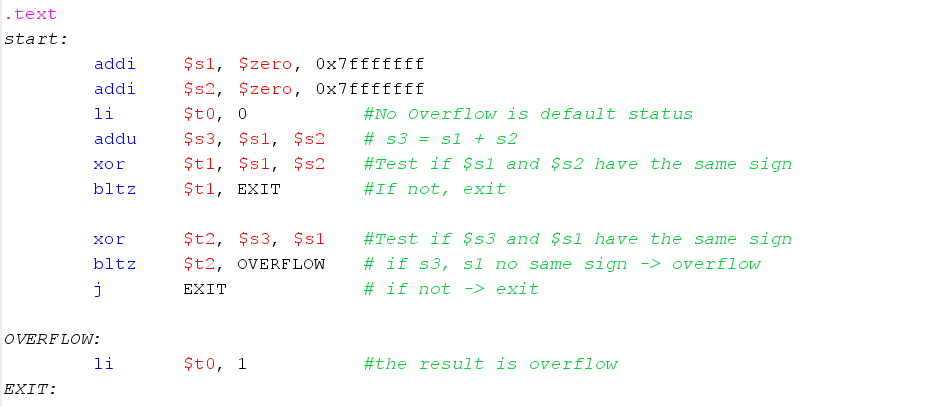
****

Thay đổi dectect overflow bằng cách:

* Khi mà dấu s1, s2 giống nhau, Tiến hành so sánh dấu của s3 ( =s2 + s1) với s1 bằng phép xor . Nếu s1, s3 cùng dấu -> không overflow, ngược lại sẽ overflow

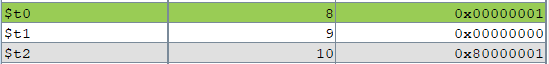
Kiểm tra:

* Trường hợp cộng 2 số dương overflow:



Khi đó, kết quả thu được:



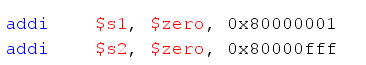


T1 = 0x000… - > s1, s2 cùng dấu

T2 = xor s3, s1 = 0x8…. = 0x 1000 ….. -> s3, s1 ngược dấu nhau

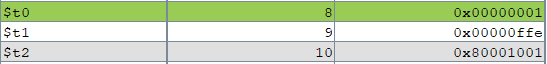
T0 = 1 -> overflow.

* Trường hợp 2 số âm, overflow



Kết quả:



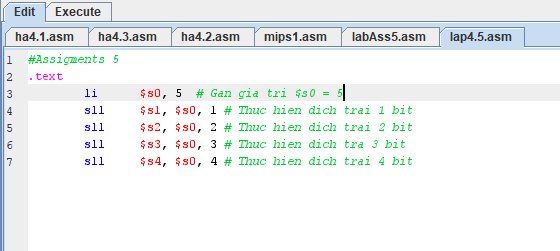


T1 = 0x000… - > s1, s2 cùng dấu

T2 = xor s3, s1 = 0x8…. = 0x 1000 ….. -> s3, s1 ngược dấu nhau

T0 = 1 -> overflow

**Asignment 5**

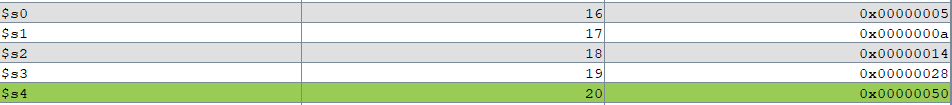


Nguyên tắc dịch trái: Giá trị thanh ghi mới = Giá trị thanh ghi hiện tại x 2^(số bit dịch) ( hệ 10 )

* Sau khi dịch trái 1 bít $s1 = 5(Giá trị $s0) x 2^1 = 10
* Sau khi dịch trái 1 bít $s2 = 5(Giá trị $s0) x 2^2 = 20
* Sau khi dịch trái 1 bít $s3 = 5(Giá trị $s0) x 2^3 = 40
* Sau khi dịch trái 1 bít $s4 = 5(Giá trị $s0) x 2^4 = 80

Hoặc có thể chuyển sang nhị phân rồi chèn thêm 0 vào bên phải

* $s0 = 00000101
* $s1 = 00001010
* $s2 = 00010100
* $s3 = 00101000
* $s4 = 01010000



**Consolution**

1. **Khác nhau giữa sll và sllv**

- Lệnh sll $t1, $t2, 10: dịch trái $t2 số bit được quy định ở phần intermediate, ở đây là 10, rồi lưu kết quả vào $t1.

- Với lệnh sllv $t1, $t2, $t3: dịch trái $t2 số bit được quy định bởi 5 bit trật tự thấp (low-order) của $t3, mang giá trị từ 0-31 và lưu kết quả vào $t1.

1. **Khác nhau giữa srl và srlv**

- Ví dụ với lệnh srl $t1, $t2, 1: dịch phải $t2 số bit được quy định ở phần intermediate, ở đây là 1 rồi lưu kết quả vào $t1.

- Với lệnh srlv $t1, $t2, $t3: Dịch phải $t2 số bit được quy định bởi 5 bit trật tự thấp (low-order) của $t3, mang giá trị từ 0-31 và lưu kết quả vào $t1.