**Báo cáo Thực hành KTMT buổi 4**

**Họ và tên:** Nguyễn Đức Phú

**MSSV:** 20215116

**Assignment 1:**

* **Trường hợp 1: Cộng hai số trái dấu**

Graphical user interface, text

Description automatically generated

* **Trường hợp 2: Cộng hai số dương không tràn**

Graphical user interface, text, application

Description automatically generated

* **Trường hợp 3: Cộng hai số dương có tràn số**

Graphical user interface, text

Description automatically generated

* **Trường hợp 4: Cộng hai số âm không tràn**

Graphical user interface, text, application

Description automatically generated

* **Trường hợp 5: Cộng hai số âm có tràn số**

Text

Description automatically generated

**Assignment 2:**

* **Code:**

.text  
li $s0, 0x87654321 # example number  
andi $t1, $s0, 0xff000000

srl $t1, $t1, 24 # extract MSB  
andi $t2, $s0, 0xffffff00 # clear LSB  
or $t3, $s0, 0x000000ff # set LSB  
xor $s0, $s0, $s0 # clear $s0

* **Giải thích:**
* Logic AND từng bit có đặc điểm: and với bit 1 thì kết quả là chính nó, and bit 0 kết quả luôn là 0
* Lệnh andi với 0xff000000 sẽ giữ lại 8 bit cao (and với 0xff) và loại bỏ các bit còn lại (and với 0x000000) => Trích xuất MSB
* Lệnh andi với 0xffffff00 hoạt động tương tự và clear LSB
* Logic OR từng bit có đặc điểm: or với bit 1 kết quả luôn là 1, or với bit 0 kết quả là chính nó
* Lệnh OR với 0x000000ff sẽ giữ nguyên 24 bit cao (do or với toàn bộ là bit 0) và đưa tất cả các bit còn lại lên 1 (do or với toàn bộ là bit 1)
* Set LSB
* Logic XOR từng bit có đặc điểm: xor 2 bit trùng nhau luôn cho ra kết quả 0
* Lệnh xor $s0 với chính nó sẽ làm giá trị tất cả các bit trở về 0
* Clear $s0
* **Kết quả khi chạy toàn bộ code:**

Graphical user interface, text, application

Description automatically generated

* Các kết quả extract MSB, clear LSB, set LSB, lần lượt được lưu tại $t1, $t2, $t3. Giá trị $s0 đã clear và trở thành 0x00000000

**Assignment 3:**

1. abs $s0, $s1:

sra $t0, $s1, 31  
xor $s0, $t0, $s1  
subu $s0, $s0, $t0

* Bit của $t0 được điền toàn bộ bởi bit dấu của $s1
* Nếu bit dấu là 1 lệnh XOR đảo bit toàn bộ $s1
  + Giá trị của $s0 sẽ là số (- $s1 -1)
  + Lệnh subu $s0 với $t0 (0xfffffffff = -1) sẽ là (-$s1 - 1) -1
  + Kết quả $s0 thu được chính là giá trị tuyệt đối $s1
* Nếu bit dấu là 0 lệnh XOR giữ nguyên toàn bộ bit
  + Giá trị $s0 chính là $s1 và $t0 là (0x00000000 = 0)
  + Lệnh subu $s0 và giá trị 0 sẽ giữ nguyên $s0
  + Kết quả $s0 là $s1 ban đầu (là số dương nên trị tuyệt đối là chính nó)

1. move $s0, $s1:

andi $s0, $s1, 0xffffffff

* Lệnh And $s1 với toàn bộ các bit 1 (0xffffffff) sẽ giữ nguyên các bit của $s1 và lưu vào $s0

1. not $s0, $s1:

nor $s0, $s1, $zero

* Do lệnh Or với giá trị có tất cả các bit là 0 giữ nguyên giá trị
* Nor sẽ đảo ngược các bit

1. ble $s1, $s2, label:

slt $t0, $s2, $s1  
beq $t0, $zero, label

*label:*

* slt set giá trị $t0 là 1 nếu $s2 > $s1và lưu vào $t0 (bằng 0 trong trường hợp còn lại)
* Thực hiện nhảy tới *label* khi $s1<= $s2 nên sử dụng lệnh beq giữa $t0 và $zero (jump khi $t0 có giá trị 0)
* **Kết quả chạy thử:**

1. abs $s0, $s1:

Graphical user interface, application

Description automatically generated

b. move $s0, $s1:

Graphical user interface, application, Word

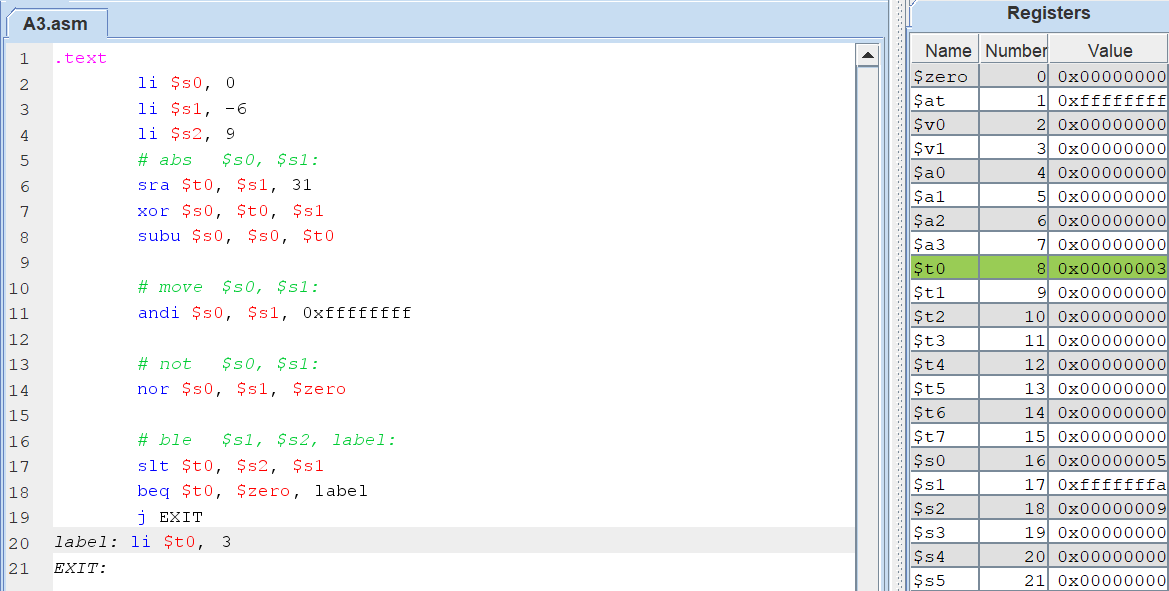
Description automatically generated

1. not $s0, $s1:

Graphical user interface, application, Word

Description automatically generated

1. ble $s1, $s2, label:



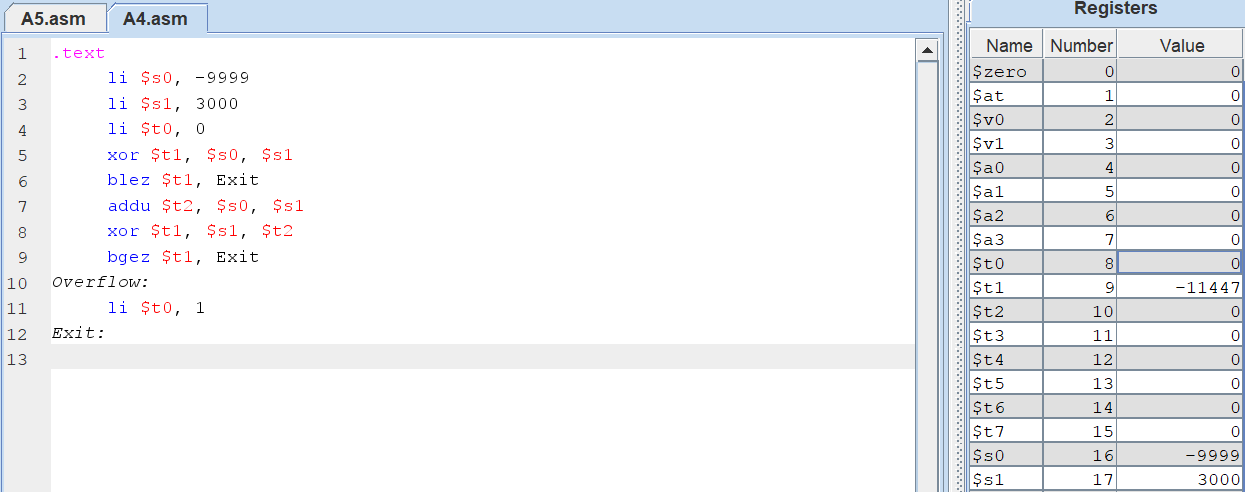
**Assignment 4:**

* **Code:**

.text  
 li $s0, -9999

li $s1, -2003  
 li $t0, 0  
 xor $t1, $s0, $s1  
 blez $t1, Exit  
 addu $t2, $s0, $s1  
 xor $t1, $s1, $t2  
 bgez $t1, Exit  
*Overflow:* li $t0, 1  
*Exit:*

* **Giải thích:**
* Sử dụng phép XOR để kiểm tra xem $s1 và $s2 có cùng dấu không
  + - Nếu $t0 là số dương tức là hai số trên cùng dấu
    - Nếu trong trường hợp còn lại tức là hai số khác dấu
* Nhảy tới EXIT
* Trong trường hợp khác dấu, cộng hai số rồi lưu vào $t2, sau đó tiếp tục sử dụng phép XOR để kiểm tra dấu của kết quả cộng đó với một trong hai số (ở đây là $s1) => Lưu kết quả XOR vào $t1
  + - Nếu $t1 lớn hơn hoặc bằng 0 tức là kết quả XOR là cùng dấu
* Không xảy ra tràn, nhảy tới EXIT
  + - Trong trường hợp còn lại tức là đã xảy ra tràn số, chương trình tiếp tục chạy tới lệnh li và gán cho $t0 giá trị 1
* **Kết quả chạy thử**
* Trường hợp 1: Hai số trái dấu

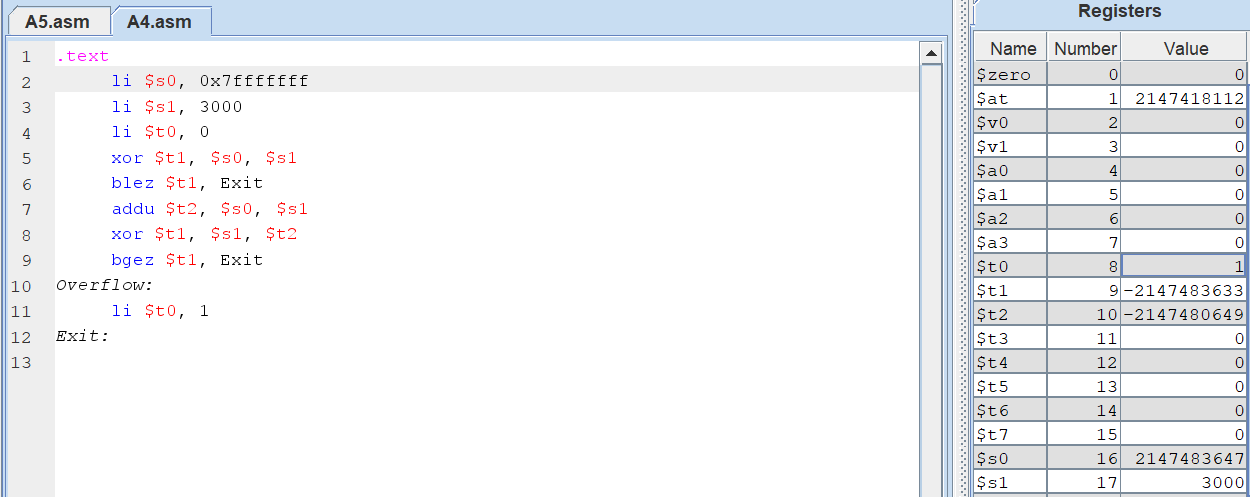


* Trường hợp 2: Hai số cùng dương không tràn

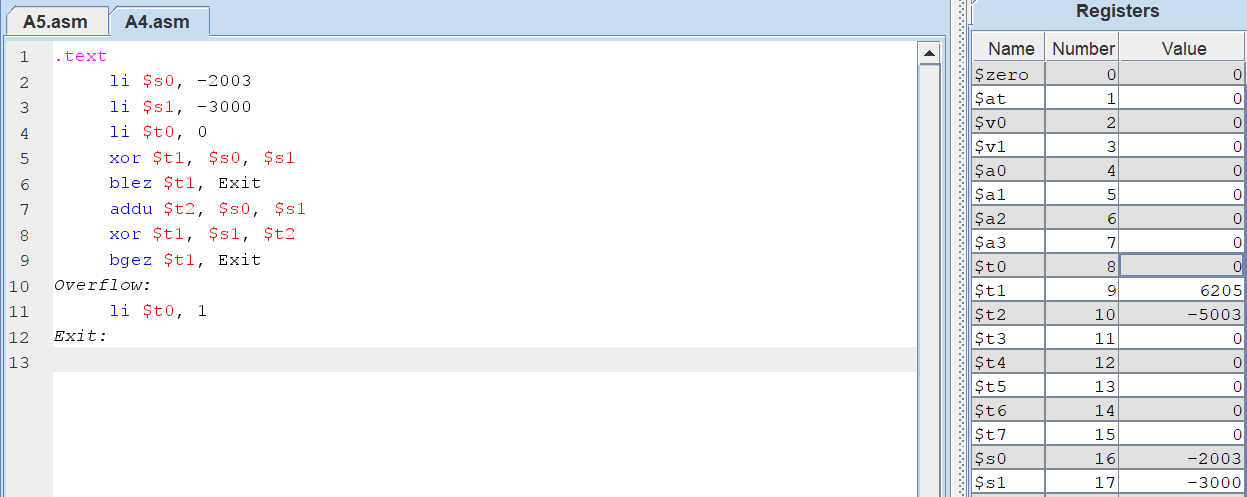
Graphical user interface, application, Word

Description automatically generated

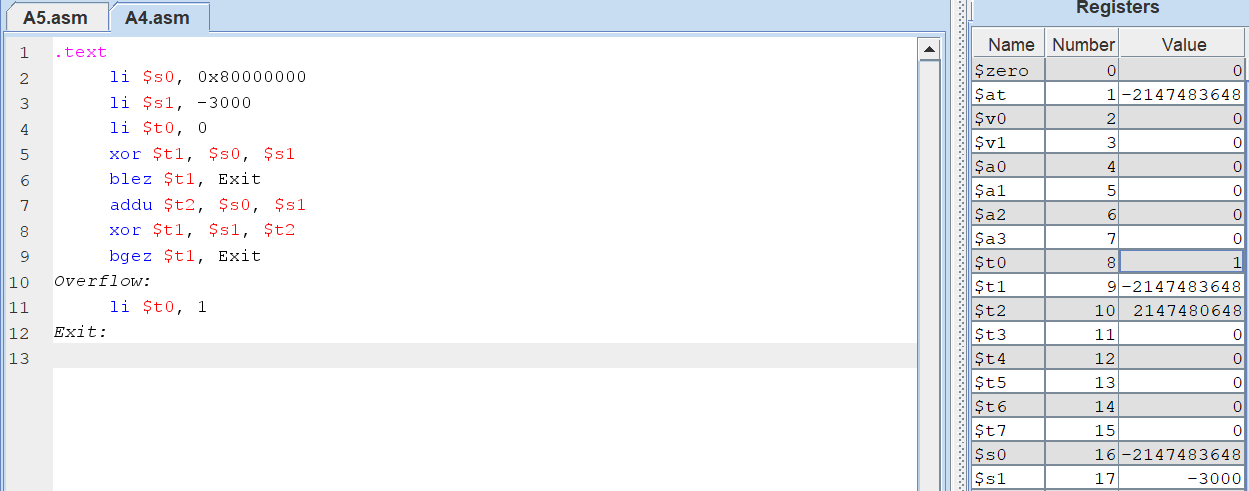
* Trường hợp 3: Hai số cùng dương có gây tràn



* Trường hợp 4: Hai số cùng âm không tràn



* Trường hợp 5: Hai số cùng âm có gây tràn



**Assignment 5:**

* **Code:**

.text  
 li $s0, 9   
 li $s1, 32  
 li $s2, 0  
 move $t1, $s1  
*loop:* beq $t1, 1, multiple  
 srl $t1, $t1, 1  
 addi $s2, $s2, 1  
 j loop  
*multiple:* sllv $t0, $s0, $s2

* **Giải thích:**
* Một số là bội của 2 khi đổi sang nhị phân sẽ có duy nhất một bit bằng 1 còn lại là 0. Khi liên tục dịch phải từng bit của số đó, dịch tới khi 1 nằm ở vị trí bit số 0 thì số lần dịch sẽ chính là số mũ của 2
* Nguyên tắc trên được áp dụng trong phần ***loop***, thoát khỏi vòng lặp khi giá trị $t1 chỉ còn là 1 và nhảy tới phần multiple
* Ở ***multiple***  ta dịch trái $s0 số bit tương ứng với số được lưu tại $s2 (chính là số mũ của 2 đã được tính tại vòng lặp)
  + - Dịch trái 1 bit là , 2 bit là , …, n bit là
* Kết quả được lưu tại $t0 chính là kết quả của phép nhân
* **Kết quả chạy thử:**

Graphical user interface, text, application

Description automatically generated

* Số mũ của 32 là 5 được lưu tại $s2
* Giá trị của phép nhân 9\*32=288 được lưu tại $t0

***Consolutions*1. What is the difference between SLLV and SLL instructions?**

* Lệnh sll $s1, $s2, imm:
* Dịch trái $s2 số bit được quy định ở phần immediate, sau đó lưu kết quả vào $s1.
* Lệnh sllv $s1, $s2, $s3:
* Dịch trái $s2 số bit được quy định bởi 5 bit trật tự thấp (low-order) của $s3, mang giá trị từ 0-31 và lưu kết quả vào $s1.

**2. What is the difference between SRLV and SRL instructions?**

* Lệnh sll $s1, $s2, imm:
* Dịch phải $s2 số bit được quy định ở phần immediate, sau đó lưu kết quả vào $s1.
* Lệnh sllv $s1, $s2, $s3:
* Dịch phải $s2 số bit được quy định bởi 5 bit trật tự thấp (low-order) của $s3, mang giá trị từ 0-31 và lưu kết quả vào $s1.