Bài thực hành số 4

Lớp: 139365 – Học phần: Thực hành Kiến Trúc Máy Tính

Họ và tên: Đinh Thị Hồng Phúc **MSSV: 20215118**

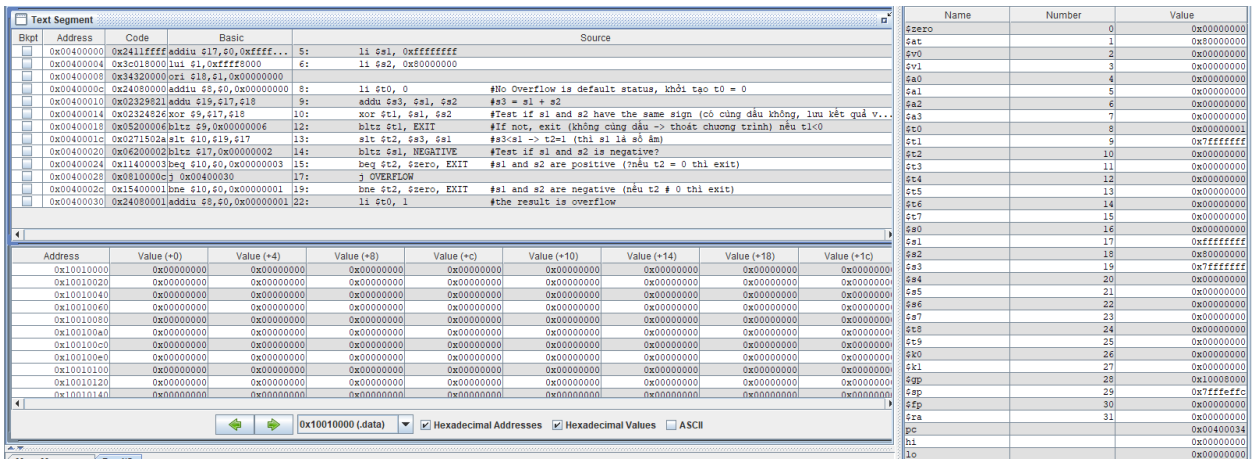
Bài 1.

```
#Assignment 1
#kiểm tra tràn số trong phép cộng s1 + s2
.text
start:
        li $s1, 0xffffffff
        li $s2, 0x80000000

        li $t0, 0               #No Overflow is default status, khởi tạo t0 = 0
        addu $s3, $s1, $s2      #s3 = s1 + s2
        xor $t1, $s1, $s2       #Test if s1 and s2 have the same sign (có cùng dấu không, lưu kết quả vào t1)

        bltz $t1, EXIT          #If not, exit (không cùng dấu -> thoát chương trình) nếu t1<0
        slt $t2, $s3, $s1       #s3<s1 -> t2=1 (thì s1 là số âm)
        bltz $s1, NEGATIVE      #Test if s1 and s2 is negative?
        beq $t2, $zero, EXIT    #s1 and s2 are positive (?nếu t2 = 0 thì exit)
                #if s3 > s1 then the result is not overflow
        j OVERFLOW
NEGATIVE:
        bne $t2, $zero, EXIT    #s1 and s2 are negative (nếu t2 # 0 thì exit)
                #if s3 < s1 then the result is not overflow
OVERFLOW:
        li $t0, 1               #the result is overflow
EXIT:
```

*Thực hiện gõ chương trình vào công cu **MARS***

- Thực hiện dòng lệnh 5 và 6: Các thanh ghi $s1 và $s2 được gán với các giá trị lần lượt là 0xffffffff và 0x80000000
- Thực hiện dòng lệnh số 8: Thanh ghi $t0 được gán giá trị 0 (tương ứng là không bị tràn số)
- Thực hiện dòng lệnh số 9: Dòng lệnh thực hiện ghi giá trị của: s3 = s1 + s2
- Thực hiện dòng lệnh số 10: Dòng lệnh thực hiện: s1^s2, lưu giá trị và t1, kiểm tra s1 có cùng dấu với s2 không
- Thực hiện dòng lệnh số 12: Nếu t1< 0 thì 2 số không cùng dấu, thoát chương trình (EXIT)
- Thực hiện dòng lệnh số 13, 15: So sánh s3 có nhỏ hơn s1 hay không (tổng nhỏ hơn số hạng đối với số dương)
- Thực hiện dòng lệnh số 14: Kiểm tra s1, s2 có là số âm hay không, nếu là số âm thì tổng nhỏ hơn số hạng thì EXIT không thì OVERFLOW
- Thực hiện dòng lệnh số 17: chuyển đến phần OVERFLOW

- Thực hiện dòng lệnh số 19: Nếu t2 ≠ 0 thì EXIT
- Thực hiện dòng lệnh số 22: gán t0 = 1 (tràn số)



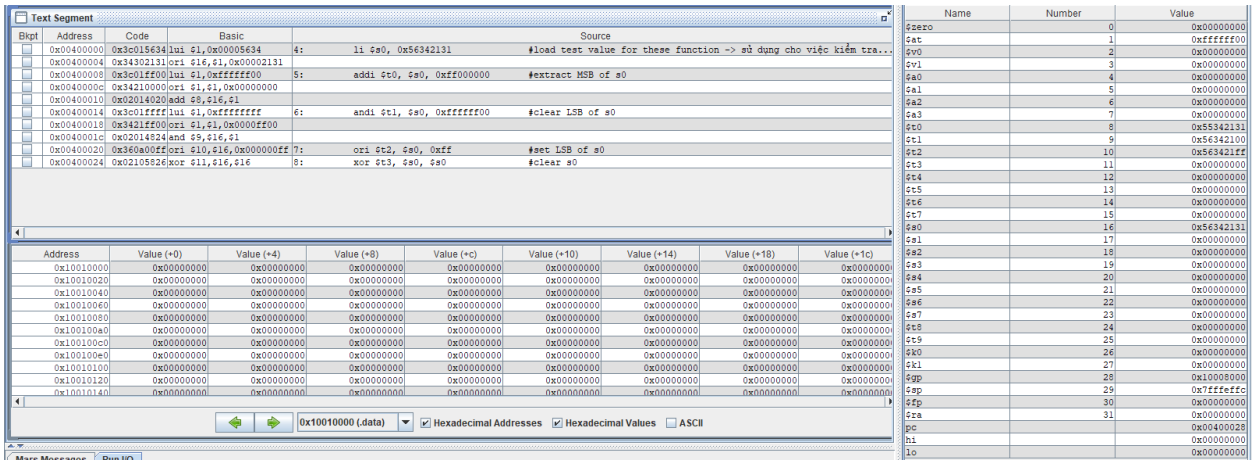
*Thực hiện chạy chương trình với **MARS***

Bài 2.

```
1  #Assignment 2
2  #trích xuất các bit cụ thể từ giá trị lưu trong thanh ghi s0
3  .text
4          li $s0, 0x56342131              #load test value for these function -> sử dụng cho việc kiểm tra các bit
5          addi $t0, $s0, 0xff000000       #extract MSB of s0
6          andi $t1, $s0, 0xffffff00       #clear LSB of s0
7          ori $t2, $s0, 0xff              #set LSB of s0
8          xor $t3, $s0, $s0               #clear s0
```

*Thực hiện gõ chương trình vào công cụ **MARS***

- Dòng 4: Thanh ghi s0 được gán giá trị 0x56342131
- Dòng 5: Lấy giá trị MSB của s0 bằng phép and với 0xff000000. Khi đó t0 = 0x55342131
- Dòng 6: Xóa giá trị LSB của s0 bằng phép and với 0xffffff00. Khi đó t1 = 0x56342100
- Dòng 7: Set LSB của s0 bằng phép or với 0xff. Khi đó t2 = 0x563421ff
- Dòng 8: Xóa giá trị s0 bằng cách xor s0 với chính nó. Khi đó t3 = 0x00000000



*Thực hiện chạy chương trình với **MARS***

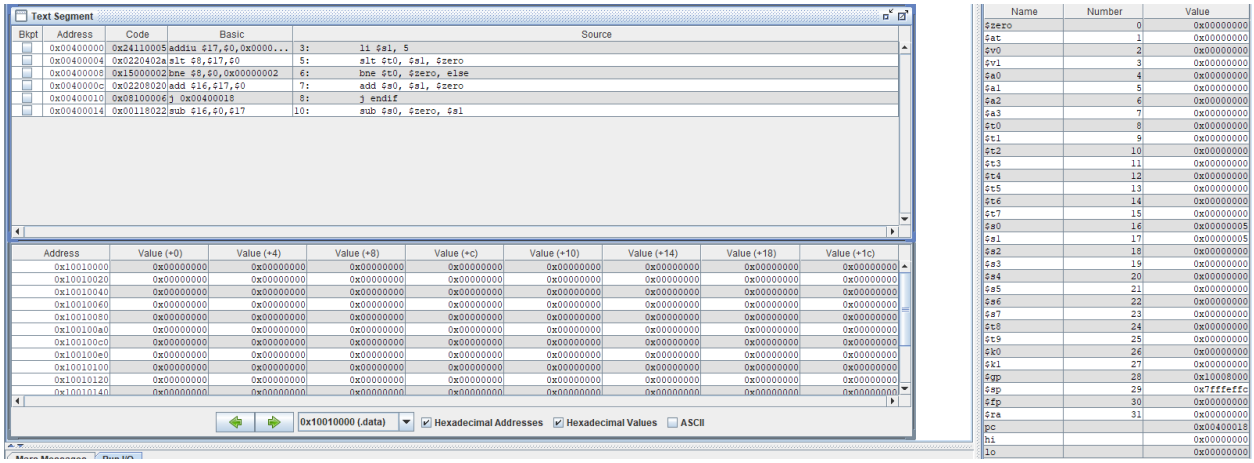
Bài 3.

a. abs

```
1   #Assignment 3a
2   .text
3           li $s1, 5
4   start:
5           slt $t0, $s1, $zero
6           bne $t0, $zero, else
7           add $s0, $s1, $zero
8           j endif
9   else:
10          sub $s0, $zero, $s1
11  endif:
```

*Thực hiện gõ chương trình vào công cụ **MARS***

- Dòng 3: Thanh ghi s1 được gán với giá trị 5
- Dòng 5: So sánh s1 với 0, gán kết quả vào t0 (kiểm tra s1 là số âm hay dương)
- Dòng 6: Nếu t0 ≠ 0 thì chuyển đến else
- Dòng 7: Thực hiện s0 = s1
- Dòng 8: Thực hiện chuyển đến endif
- Dòng 10: Thực hiện s0 = 0 – s1



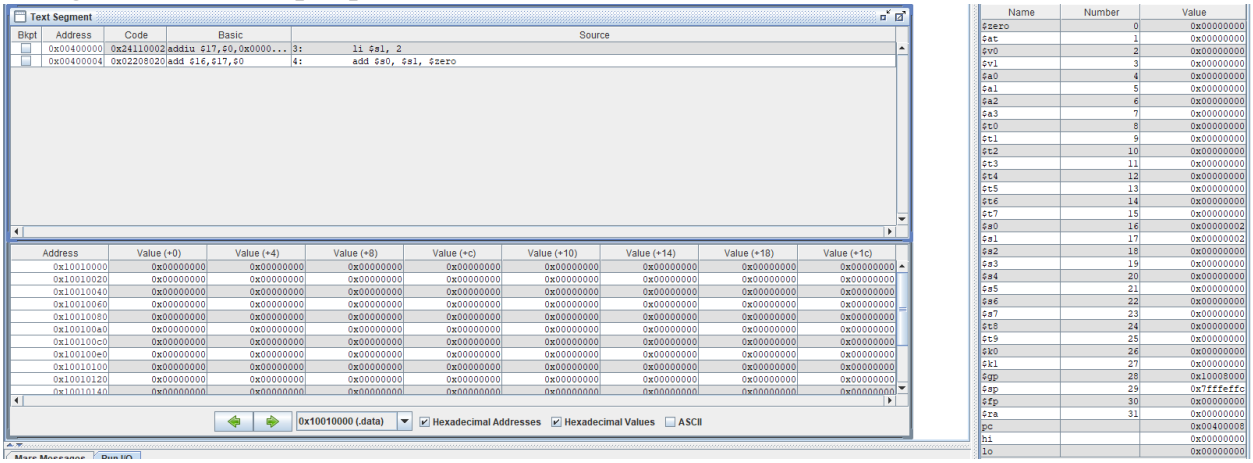
*Thực hiện chạy chương trình với **MARS***

b. move

```
1  #Assignment 3b
2  .text
3          li $s1, 2
4          add $s0, $s1, $zero
```

*Thực hiện gõ chương trình vào công cụ **MARS***

- Dòng 3: Thanh ghi s1 được gán giá trị 2
- Dòng 4: Thực hiện phép toán s0 = s1



*Thực hiện chạy chương trình với **MARS***

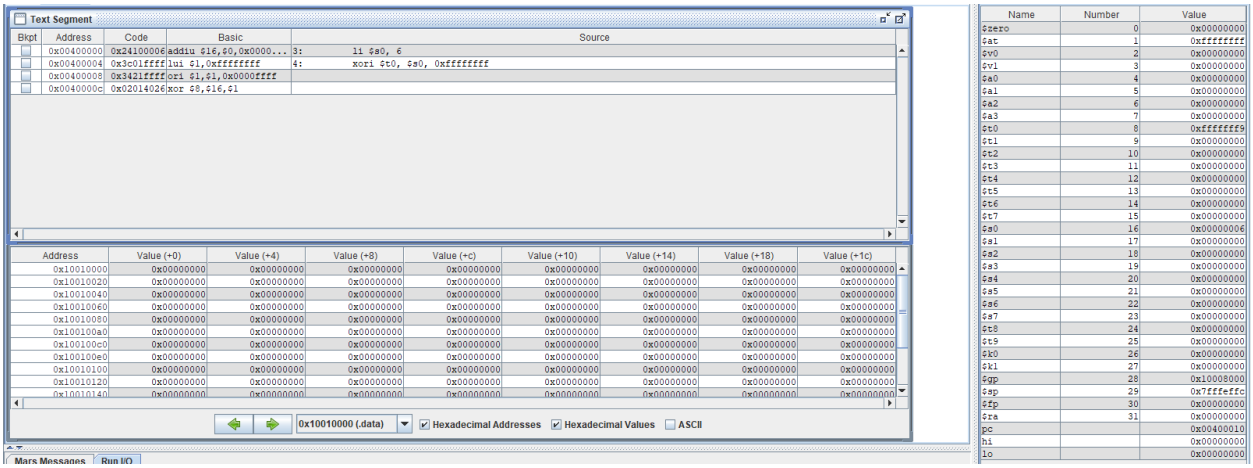
c. not

```
1  #Assignment 3c
2  .text
3          li $s0, 6
4          xori $t0, $s0, 0xffffffff
```

*Thực hiện gõ chương trình vào công cụ **MARS***

- Dòng 3: Thanh ghi s0 được gán giá trị 6

- Dòng 4: Thực hiện đảo dấu s0 bằng phép xor với 0xffffffff. Khi đó t0 = 0xfffffff9



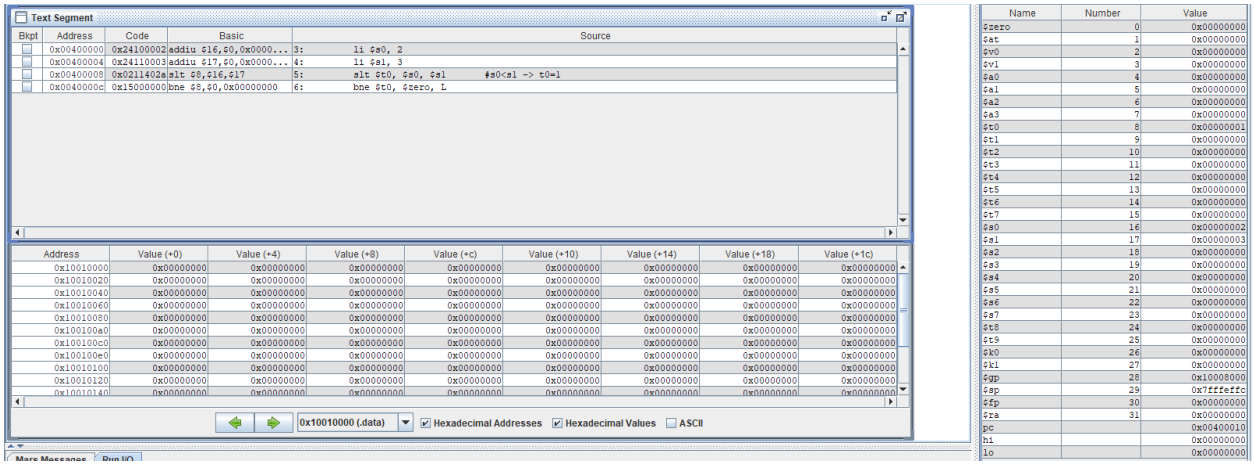
*Thực hiện chạy chương trình với **MARS***

d. ble

```
1  #Assignment 3d
2  .text
3          li $s0, 2
4          li $s1, 3
5          slt $t0, $s0, $s1        #s0<s1 -> t0=1
6          bne $t0, $zero, L
7  L:
```

*Thực hiện gõ chương trình vào công cu **MARS***

- Dòng 3, 4: Thanh ghi s0, s1 được gán giá trị lần lượt là 2 và 3
- Dòng 5: So sánh s0 có nhỏ hơn s1 hay không cho kết quả vào t0
- Dòng 6: Nếu t0 ≠ 0 (s0<=s1) thì chuyển sang L



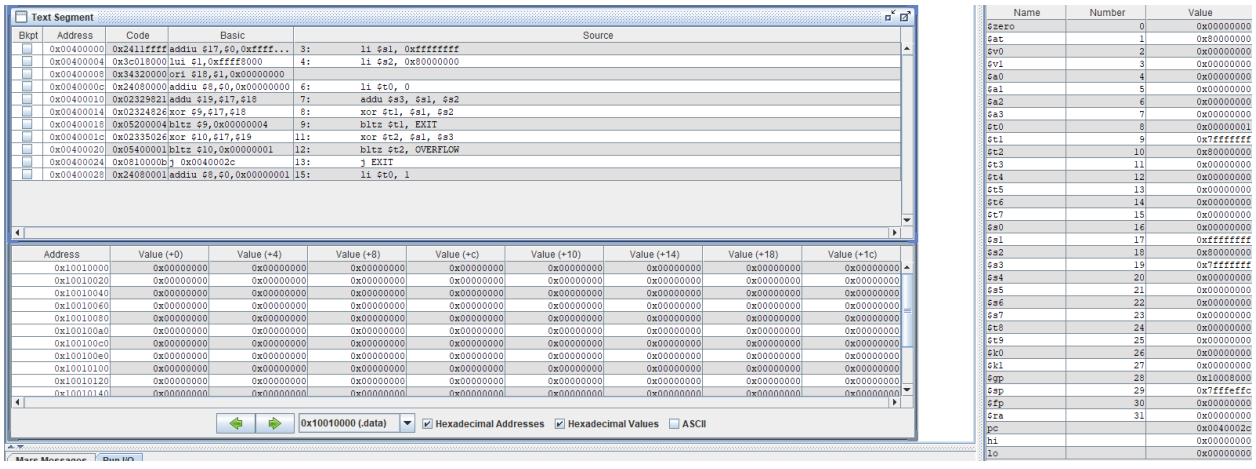
*Thực hiện chạy chương trình với **MARS***

Bài 4.

```
1   #Assignment 4
2   .text
3           li $s1, 0xffffffff
4           li $s2, 0x80000000
5   start:
6           li $t0, 0
7           addu $s3, $s1, $s2
8           xor $t1, $s1, $s2
9           bltz $t1, EXIT
10
11          xor $t2, $s1, $s3
12          bltz $t2, OVERFLOW
13          j EXIT
14  OVERFLOW:
15          li $t0, 1
16  EXIT:
```

*Thực hiện gõ chương trình vào công cụ **MARS***

- Dòng 3, 4: Thanh ghi s1, s2 được gán giá trị lần lượt là 0xffffffff và 0x80000000
- Dòng 6: Thanh ghi t0 được gán giá trị 0 (không tràn số)
- Dòng 7: Thực hiện phép toán s3 = s1 + s2
- Dòng 8: Thực hiện phép toán s1^s2 gán kết quả vào t1
- Dòng 9: Nếu t1<0 (s1, s2 trái dấu) thì EXIT (không tràn số)
- Dòng 11: Thực hiện phép toán s1^s3 gán kết quả vào t2
- Dòng 12: Nếu t2<0 (s1, s3 trái dấu) thì OVERFLOW
- Dòng 13: chuyển đến phần EXIT
- Dòng 15: Thanh ghi t0 được gán giá trị 1 (tràn số)



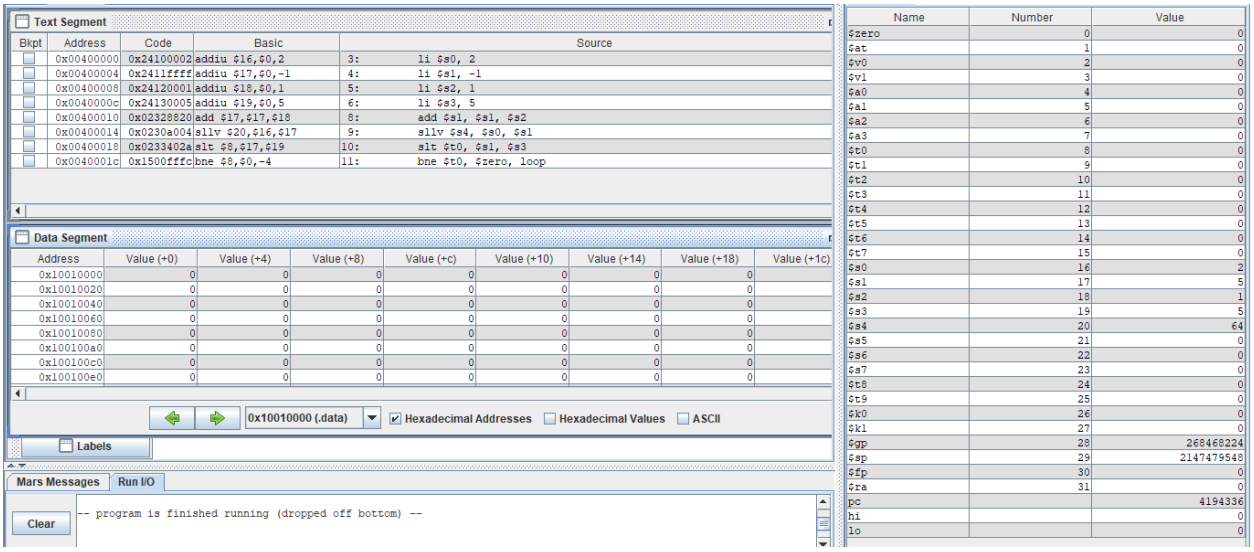
*Thực hiện chạy chương trình với **MARS***

Bài 5.

```
1   #Assignment 5
2   .text
3           li $s0, 2
4           li $s1, -1
5           li $s2, 1
6           li $s3, 5
7   loop:
8           add $s1, $s1, $s2
9           sllv $s4, $s0, $s1
10          slt $t0, $s1, $s3
11          bne $t0, $zero, loop
```

*Thực hiện gõ chương trình vào công cụ **MARS***

- Dòng 3, 4, 5, 6: Thanh ghi s0, s1, s2, s3 được gán giá trị lần lượt 2, -1, 1, 10 (s0: giá trị ban đầu, s1 = i = -1, s2 = step = 1, s3 = n = 10)
- Dòng 8: Thực hiện phép toán s1 = s1 + s2 (i = i + step)
- Dòng 9: Thực hiện phép dịch bit sang trái: s4 = s0 * 2^s1
- Dòng 10, 11: Kiểm tra s1 (i) đã đếm đến s3 (n = 5) hay chưa. Nếu chưa thì tiếp tục vòng lặp



*Thực hiện chạy chương trình với **MARS***