Laboratory Exercise 1

Yêu cầu 1
- Sự thay đổi của $s0: 0x00000000 -> 0x00003007 -> 0x00000000
- Sự thay đổi của $pc: 0x00400000 -> 0x00400004 -> 0x00400008

Yêu cầu 2
1)
addi $s0, $zero, 0x3007
+ opcode: 8
+ rs: $zero => $0
+ rt: $s0 => $16
+ imm: 0x3007 => 12295

Ghép lại: 001000 00000 10000 0001100000000111
           0010 0000 0001 0000 0001 1000 0000 0111
           0x20103007
2)
add $s0, $zero, $0
+ opcode: 0
+ rs: $zero => $0
+ rt: $0
+ rd: $s0 => $16
+ shamt: 0
+ funct: 32

Ghép lại: 000000 00000 00000 10000 00000 100000
           0000 0000 0000 0000 1000 0000 0010 0000
           0x00008020

Yêu cầu 3
addi $s0, $zero, 0x2110003d
- Lệch addi được chuyển thành 3 lệch:
    + lui $1, 0x00002110
    + ori $1, $1, 0x0000003d
    + add $16, $0, $1
- Lệch lui cho nửa upper vào $1 = cách dịch bits.
- Lệch ori cho nửa lower vào $1 = cách OR nửa lower với $1.
- Lệch add dùng phép cộng để gán hoàn chỉnh 32 bits vào $16.

==========
Laboratory Exercise 2

Yêu cầu 1
- Sự thay đổi của $s0: 0x00000000 -> 0x21100000 -> 0x2110003d
- Sự thay đổi của $pc: 0x00400000 -> 0x00400004 -> 0x00400008

Yêu cầu 2
- Các byte đầu tiên ở dòng "0x00400000", cột "Value (+0)".

```
====================
Laboratory Exercise 3
- Lệch li là lệch pseudo. Có 2 cách chuyển đổi về lệch basic tùy vào immediate:
    + Ở lệch li 1, kích thước immediate lớn. Lệch li được chuyển thành 2 lệch:
        > lui cho nửa upper vào $1 = cách dịch bits.
        > ori cho 0x2110003d vào $16 = cách OR nửa lower với $1.
    + Ở lệch li 2, kích thước immediate nhỏ. Lệch li được chuyển thành 1 lệch:
        > addiu cho 0x2 vào $17 bằng cách cộng unsigned.

====================
Laboratory Exercise 4

Yêu cầu 1
- Sau lệch 1:
    + $t1 = 0x00000005
    + $t2 = 0xffffffff
- Sau lệch 2:
    + $s0 = 0x0000000a -> 0x00000009
- Sau khi kết thúc chương trình, kết quả đúng.

Yêu cầu 2
1)
addi $t1, $zero, 5
+ opcode: 8 = 001000
+ $t1: 9 = 01001
+ $zero: 0 = 00000
+ 5 = 0000 0000 0000 0101
Ghép lại:   001000 00000 01001 0000 0000 0000 0101
            001000 00000 01001 0000 0000 0000 0101
            0x20090005
2)
addi $t2, $zero, -1
+ opcode: 8 = 001000
+ $t2: 10 = 01010
+ $zero: 0 = 00000
+ -1 = ffff ffff ffff ffff
Ghép lại:   001000 00000 01010 ffff ffff ffff ffff
            0010 0000 0000 1010 ffff ffff ffff ffff
            0x200affff

Yêu cầu 3
1)
+ opcode: 0 = 000000
+ $s0: 16 = 10000
+ $t1: 9 = 01001
+ $t1: 9 = 01001
+ shamt: 0 = 00000
+ funct: 32 = 100000
Ghép lại:   000000 01001 01001 10000 00000 100000
            0000 0001 0010 1001 1000 0000 0010 0000
```

```
            0x01298020
2)
+ opcode: 0 = 000000
+ $s0: 16 = 10000
+ $s0: 16 = 10000
+ $t2: 10 = 01010
+ shamt: 0 = 00000
+ funct: 32 = 100000
Ghép lại:   000000 10000 01010 10000 00000 100000
            0000 0010 0000 1010 1000 0000 0010 0000
            0x020a8020

=====================
Laboratory Exercise 5
Yêu cầu 2.1
1) Sau addi $t1, $zero, 4
    + $t1 = 0x00000004
2) Sau addi $t2, $zero, 5
    + $t2 = 0x00000005
3) Sau mul $s0, $t1, $t2
    + HI-LO = $t1 * $t2 = X * Y = 4*5 = 20 = 0x00000014
    + $s0 = LO = 0x00000014
4) Lệch mul $s0, $s0, 3 được chuyển làm các lệch addi và mul:
    + Sau lệch cộng: $at = $zero + 3 = 0x00000003
    + Sau lệch nhân: $s0 = LO = 0x0000003c
5) Truy xuất LO bằng lệch mflo vào $s1:
    + $s1 = LO = 0x0000003c
    + HI = 0x00000000

Yêu cầu 2.2
- Sau khi kết thúc chương trình, kết quả đúng.

Laboratory Exercise 6
Yêu cầu 1
- Lệch la được chuyển thành các lệch lui và ori để lấy giá trị của địa chỉ X:
    + Cơ chế giống li khi làm việc với immediate lớn như trên.
    + Điểm khác biệt là la có đầu vào địa chỉ và kết quả được lưu là địa chỉ biến.

Yêu cầu 2
1) la $t8, X
Sau khi ghép lại thì mã máy tính và mã hợp ngữ tương đồng.
2) la $t9, Y
Sau khi ghép lại thì mã máy tính và mã hợp ngữ tương đồng.

Yêu cầu 3.1
- Sau lệch la $t8, X:
    ...

Yêu cầu 3.2
    + Vai trò của lệch lw: Lấy số từ bộ nhớ
```

+ Vai trờ của lệch sw: Lưu số từ bộ nhớ