Assignment 1:

1. Register observation:

* After running the first command:

+ The value of $s0 changes from 0 to 0x00003007 = 12295.

+ The value of pc changes from 0x00400000 to 0x00400004.

* After running the second command:

+ The value of $s0 changes from 0x00003007 back to 0.

+ The value of pc changes from 0x00400004 to 0x00400008.

1. Changing the command to addi $s0, $zero, 0x2110003d:

It is equivalent to 3 commands:

+ lui $1, 0x00002110, which loads the upper immediate value. $at is now 0x00002110

+ ori $1, $1, 0x0000003d, which makes $at becomes 0x2110003d

+ add $16, $0, $1, which makes $s0 = $zero + $at. $s0 is now 0x2110003d

Explain: addi can only add constant with no more than 16 bits, but 0x2110003d needs more than 16 bits to represent it.

Assignment 2:

1. Register observation:

* After running the first command:

+ The value of $s0 changes from 0 to 0x21100000 = 554696704.

+ The value of pc changes from 0x00400000 to 0x00400004.

* After running the second command:

+ The value of $s0 changes from 0x21100000 to 0x2110003d.

+ The value of pc changes from 0x00400004 to 0x00400008.

1. In Data Segment, we can see that this byte is similar to the first and second column of Text Segment.

Assignment 3:

* The command li $s0, 0x2110003d is equivalent to 2 commands:

+ lui $1, 0x00002110

+ ori $16, $1, 0x0000003d

* Explain: li is I format and 0x2110003d needs more than 16 bits to represent it.

Assignment 4:

1. Register observation:

* After running the first command:

+ The value of $t1 changes from 0 to 0x0000005 = 5.

+ The value of pc changes from 0x00400000 to 0x00400004.

* After running the second command:

+ The value of $t2 changes from 0 to 0xffffffff = -1.

+ The value of pc changes from 0x00400004 to 0x00400008.

* After running the third command:

+ The value of $s0 changes from 0 to 0x0000000a = 10, which is the result of the equation 2\*5-1.

+ The value of pc changes from 0x00400008 to 0x0040000c.

1. Comparison between assembly language and machine language of addi command,

* The syntax of assembly language is: addi $rt, $rs, imm, in which rt, rs is the name and imm is a decimal number.
* The syntax of machine language is: addi $rt, $rs, imm, in which rt, rs is the number and imm is a hexa number.

Assignment 5:

The anomaly is 'mul $s0, $s0, 3' is divided to two instructions: 'addi $1,$0,3' and 'mul $16,$16,$1' because mul work with two register, so MIPS store the value '3' into one register first then call mul.

Assignment6: