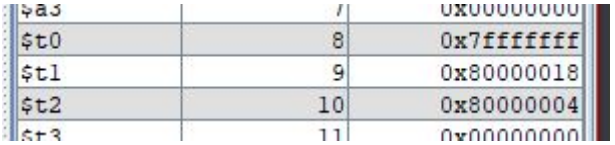
Practica 1

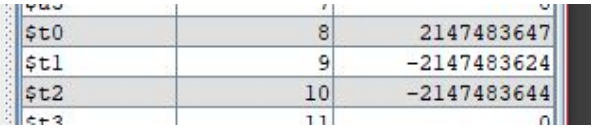
- Actividad 5

➢ Volved a escribir el programa cambiando addi por addiu y dad como valor inicial de $t0 el positivo más grande posible ($t0 = 0x7FFFFFFF) y ejecutadlo observando el contenido de $t1 en hexadecimal y en decimal.

```
1  # Actividad 5
2
3  # Especificamos desde que lugar en la memoria empezamos
4  .text 0x00400000
5
6  # Inicializamos $8 con el valor hexadecimal.
7  addiu $t0, $zero, 0x7FFFFFFF
8
9  # Le sumamos 25 y 5 al valor hexadecimal.
10 addiu $t1, $t0, 25
```



$t0	8	0x7fffffff
$t1	9	0x80000018
$t2	10	0x80000004



$t0	8	2147483647
$t1	9	-2147483624
$t2	10	-2147483644

➢ ¿Qué ha ocurrido?

Lo que sucede es que no se puede representar dicho número con la cantidad de bytes actuales, y ocurre un desbordamiento

➢ Si el programador considera que está operando con números naturales, el resultado que hay en $t1 sería correcto? ¿Cuál sería su valor en decimal?

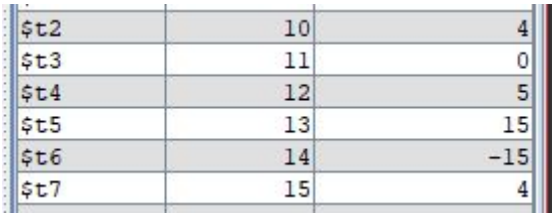
No, ya que los números naturales no comprenden los números negativos, y el valor en decimal de $t1 en decimal es -2147483624.

- Cuestión 6

➢ Escribe el código que haga las siguientes acciones utilizando el convenio de registros y utilizando la instrucción addi:
$12=5
$10= 8
$13=$12 + 10
$10=$10 - 4
$14=$13 – 30
$15=$10

```
# Cuestion 6

# Especificamos desde que lugar en la memoria empezamos
.text 0x00400000

# $12=5
addi $t4, $zero, 5

# $10=8
addi $t2, $zero, 8

# $13=$12 + 10
addi $t5, $t4, 10

# $10=$10 - 4
addi $t2, $t2, -4

# $14=$13 - 30
addi $t6, $t5, -30

# $15=$10
addi $t7, $t2, 0
```

➢ Ensamblad y ejecutad el programa y comprobad que el resultado final es
$t7 = $t2 = 4
$t6=-15
$t4=5
$t5=15



$t2	10	4
$t3	11	0
$t4	12	5
$t5	13	15
$t6	14	-15
$t7	15	4

Practica 2

- Cuestión 13

➢ Escribe el código que haga la operación lógica OR de $t1 y $t2 y lo guarde en $t3, la operación lógica AND de $t1 y $t2 y lo guarde en $t4, y la operación lógica XOR de $t1 y $t2 y lo guarde en $t5. Escribe en la ventana de registros, tras ensamblarlo, los siguientes valores para los registros $t1=0x55555555 y $t2= 0xAAAAAAAA. Ejecuta el código y estudia los resultados.

```
.text

or $t3, $t2, $t1
and $t4, $t1, $t2
xor $t5, $t1, $t2

# Salir del programa
addi $v0, $zero, 10
syscall
```

$t1	9	0x55555555
$t2	10	0xaaaaaaaa
$t3	11	0xffffffff
$t4	12	0x00000000
$t5	13	0xffffffff

El resultado de $t3 y $t5 se deben a que $t1 se compone de A (hex) = 1010 (bin) y $t2 se compone de 5 (hex) = 0101 (bin), entonces en bin, la operación sería:

1010 or 0101 = 1010 xor 0101 = 1111 = F

El resultado de $t4 se debe a que:

1010 and 0101 = 0000.

- Cuestión 14

➢ Supón que $t1=0x0000FACE, utilizando únicamente las instrucciones lógicas de la tabla anterior, escribe el código que reordene los bits de $t1 de manera que en $t2 aparezca el valor 0x0000CAFE. Ensambla y escribe en la ventana de registros $t1=0x0000FACE. Ejecuta y comprueba que el código es correcto.

```
.text

# Bueno.. mejor que suponer, lo guardamos para cuando lo usemos.
addi $t1, $zero, 0x0000FACE


# Primero vamos a ver que valores tienen en bin.
# 0000FACE = 0000 0000 0000 0000 1111 1010 1100 1110
# 0000CAFE = 0000 0000 0000 0000 1100 1010 1111 1110

# Parece que la opción más directa sería usando xor...

# 0000FACE xor 0000CAFE = 0000 0000 0000 0000 0011 0000 0011 0000 = 0003030
# Entoces 0000FACE xor 0003030 = 0000CAFE
xori $t2, $1, 0x0003030

# Salir del programa
addi $v0, $zero, 10
syscall
```

$t1	9	0x0000face
$t2	10	0x0000cafe

- Cuestión 16

➢ Escribe el código que lee un valor entero por teclado y escribe el mismo valor en binario por la consola.

```
.text 0x00400000

# Pide un entero
add $v0, $zero, 5
syscall

# Movemos el valor al registro de salido $a0
add $a0, $v0, $zero

# Muestra el numero guardado en $a0 en binario
add $v0, $zero, 35
syscall

# Salir del programa
addi $v0, $zero, 10
syscall
```

Práctica 3

- Cuestión 8.

➢ Escribe un programa que lea del teclado una letra en mayúscula y la escriba en minúscula en la consola.

```
# Cuestión 8.
#   Escribe un programa que lea del teclado una letra en mayúscula y la escríba en minúscula en la consola.


# Vale.. vamos a ver que diferencia hay entre las letras
#   mayusculas y las minusculas, y a ver si dicha diferencia es equivalente entre otras letras.

# A = 0x41 | a = 0x61 => 0100 0001 = A > 0110 0001 = a | a - A = 0010 0000 = 0x20 = 32 en decimal
# Z = 0x5A | z = 0x7A => 0101 1010 = Z > 0111 1010 = z | z - A = 0010 0000 = 0x20 = 32 en decimal

# La diferencia entre las minusculas y las mayusculas es de 32 y se mantiene a pesar de la letra,
#     entonces, para pasar de mayuscula a minuscula solo
#     tenemos que sumarle 32 al valor del caracter.

li $v0, 12          # Función 12. Read character
syscall             # Carácter leído en $v0

move $t0, $v0       # Carácter a escribir en $t0

li $a0, '\n'        # Preparamos un salto de linea para que quede mas guapo.
li $v0, 11          # Función 11. Print character
syscall             # Hacemos un salto de linea.

addi $a0, $t0, 32   # Le sumamos 32 para que se convierta en minuscula y lo dejamos en $a0.

li $v0, 11          # Función 11. Print character
syscall             # Mostramos el caracter en minuscula.

li $v0, 10          # Función 10. Acaba programa
syscall
```

```
A
a
-- program is finished running --


Reset: reset completed.

B
b
-- program is finished running --

Z
z
-- program is finished running --


Reset: reset completed.

U
u
-- program is finished running --
```

- Cuestión 10.

➢ Convierte caracteres numéricos. Escribe el código que lea del teclado un carácter numérico (del ‘0’ al ‘9’) y lo convierta en un valor numérico (del 0 al 9) y lo escriba por pantalla. Itera el código.

```
# Cuestión 10.
# Convierte caracteres numéricos. Escribe el código que lea del teclado un
#    carácter numérico (del ‘0’ al ‘9’) y lo convierta en un valor numérico (del 0 al 9)
#    y lo escriba por pantalla. Itera el código.


# Pues lo mismo que en la cuestion 8, pero cambiando la manera de recoger y mostrar datos y con un tag a primera vista...
#    vamos a ver que diferencia hay entre los caracter y los numeros y si dicha diferencia es consistente.

# 0 = 0x00 | '0' = 0x30 => '0' - 0 = 0011 0000 - 0000 0000 = 0011 0000 = 0x30 = 48.
# 9 = 0x09 | '9' = 0x39 => '9' - 9 = 0011 1001 - 0000 1001 = 0011 0000 = 0x30 = 48.

# La diferencia es de 48, y es consistente con todos los numeros.


empieza: li $v0, 12             # Creamos el tag.
syscall                         # Lee un caracter

subi $a0, $v0, 48               # le restamos 48, y lo preparamos para que el valor sea mostrado como numero

li $v0, 1                       # Muestra un entero
syscall

li $a0, '\n'                    # Ponemos un salto de linea para que quede bien
li $v0, 11
syscall

j empieza                       # Salta al principio, y vuelta a empezar
```