

Estructura de los computadores

Practica 4 5 6

Francisco Joaquín Murcia Gómez 48734281H

Grupo 3

Índice

Practica 4.....3-10

Aritmética de enteros (3) y funciones

Ejemplos3-8

Entregas9-10

Practica 5.....11-14

Estructuras de control

Ejemplos11-12

Entregas13-14

Practica 6.....15-18

Variables

Ejemplos15

Entregas16-18

Practica 4

Ejemplos

1. Instrucciones de desplazamiento

1.1. ¿Con qué instrucciones traducirá el ensamblador las pseudoinstrucciones rol y ror?.

"ori", "sll", "srl" y "or"

1.2. Las instrucciones de desplazamiento siguen el formato tipo R. ¿Cuál es la codificación en hexadecimal de la instrucción ssl \$t2,\$t1,3? Y de la instrucción srl \$t2,\$t1,7. ¿Cuál es el valor de cada campo? Ayúdate ensamblado un código de prueba.

0x000950c0, 0x000951c2

1.3. Descubre la palabra escondida. Dado el siguiente código, complétalo de tal manera que mediante instrucciones lógicas y de desplazamientos puedas escribir en la consola cada uno de los caracteres que se encuentran almacenados en cada byte del registre \$t1.

```
.text

li $t1, 1215261793

move $t2,$t1
srl $t2,$t2,24
# Desplazamiento de 24 bits a la derecha.
andi $t2,$t2,0x000000FF
move $a0,$t2
li $v0,11
syscall

move $t2,$t1
srl $t2,$t2,16
# Desplazamiento de 16 bits a la derecha.
andi $t2,$t2,0x000000FF
move $a0,$t2
li $v0,11
syscall

move $t2,$t1
srl $t2,$t2,8
# Desplazamiento de 8 bits a la derecha.
andi $t2,$t2,0x000000FF
move $a0,$t2
li $v0,11
syscall

move $t2,$t1
andi $t2,$t2,0x000000FF
move $a0,$t2
li $v0,11
syscall
```

```
-- program is finished :
Hola
-- program is finished :
Hola
-- program is finished :
```

2. Funciones del programa

2.1. Escribe una función con instrucciones suma que devuelve el cuádruplo del número entero que se le pasa. Escribid el programa principal que lea el número del teclado y escriba el cuádruplo en la consola aprovechando la función imprim.

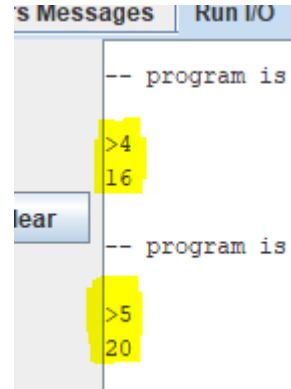
```

.text
li $a0, '>' #Comienza programa principal
li $v0,11
syscall
li $v0,5
syscall #Leer un enter
move $a0,$v0
jal cuadruplo
move $a0, $v0 #argumento a pasar en $a0
jal imprim #llamamos a la función
li $v0,10 #Acaba el programa
syscall

#-----Funcions-----
imprim: addi $v0,$0,1 #comienza la función
        syscall #Escribe un valor
        li $a0, '\n' #Salto de línea
        li $v0,11
        syscall
        jr $ra #Vuelta al programa principal

cuadruplo: #le suma tres veces a si mismo
        add $v0,$a0,$a0
        add $v0,$v0,$a0
        add $v0,$v0,$a0

        jr $ra
  
```



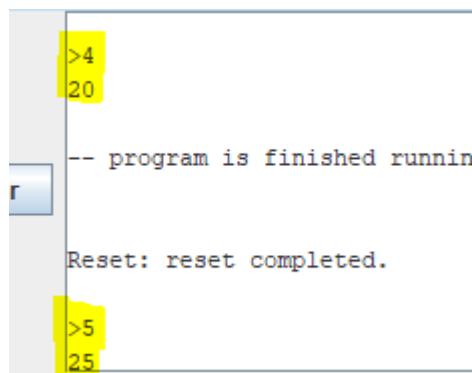
3. Convenio de los registros del MIPS

3.1. Observad el último código escrito, ¿se ajusta al convenio de MIPS de utilización de registros? De ahora en adelante haced servir el convenio.

No, porque no utiliza el \$s

4. Multiplicación por desplazamientos y sumas

4.2. Modifica el código en el que ahora haya una función multi5 que multiplique por 5 y muestre el resultado por consola. Comprobar que el resultado es correcto.



```

.text
li $a0, '>'
li $v0,11
syscall
li $v0,5
syscall #Leer un entero

move $a0, $v0 #parámetro a pasar en $a0
jal mult4 #llamamos a la función mult4
move $a0, $v0
jal imprim #Llamamos a la función imprim
li $v0,10 #Acaba el programa
syscall

#-----Funcions-----#

imprim: addi $v0,$0,1 #función imprim
syscall #Escribe el valor en $a0
li $a0, '\n' #Salto de línea
li $v0,11
syscall
jr $ra #Vuelve al programa principal

mult4:
sll $v0, $a0, 2 #Función multiplica entero por 4
add $v0,$v0,$a0 # le suma a si mismo para llegar al 5
jr $ra

```

4.3. Modifícalo ahora para tener una nueva función mult10 que multiplique por 10. Comprobar que el resultado es correcto.

```

.text
li $a0, '>'
li $v0,11
syscall
li $v0,5
syscall #Leer un entero

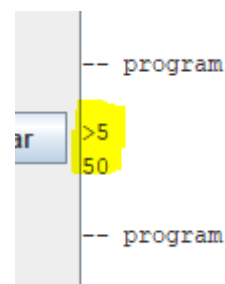
move $a0, $v0 #parámetro a pasar en $a0
jal mult10 #llamamos a la función mult4
move $a0, $v0
jal imprim #Llamamos a la función imprim
li $v0,10 #Acaba el programa
syscall

#-----Funcions-----#

imprim: addi $v0,$0,1 #función imprim
syscall #Escribe el valor en $a0
li $a0, '\n' #Salto de línea
li $v0,11
syscall
jr $ra #Vuelve al programa principal

mult10:
sll $v0, $a0, 3 #Función multiplica entero por 8
add $v0,$v0,$a0
add $v0,$v0,$a0
#se suma dos veces si mismo
jr $ra

```



4.4. Escribe una función que multiplique por 60. Escribe el programa principal que lea una cantidad de minutos y devuelve por consola la cantidad en segundos.

```
.text
li $a0, '>'
li $v0, 11
syscall
li $v0, 5
syscall #Leer un entero

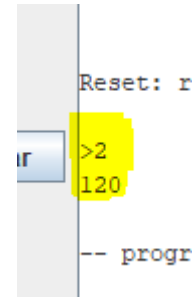
move $a0, $v0 #parámetro a pasar en $a0
jal mult60 #llamamos a la función mult4
move $a0, $v0
jal imprim #Llamamos a la función imprim
li $v0, 10 #Acaba el programa
syscall

#-----Funcions-----#

imprim: addi $v0, $0, 1 #función imprim
syscall #Escribe el valor en $a0
li $a0, '\n' #Salto de línea
li $v0, 11
syscall
jr $ra #Vuelve al programa principal

mult60:
sll $v0, $a0, 5 #Función multiplica entero por 60
sll $t0, $a0, 4 #Función multiplica entero por 60
add $v0, $v0, $t0
sll $t0, $a0, 3
add $v0, $v0, $t0
sll $t0, $a0, 2
add $v0, $v0, $t0

jr $ra
```



4.5. Modifica el código de tal manera que ahora lo que lea sea una cantidad de hora y muestre por consola la cantidad de segundos.

```
Reset: res
>
7200
-- program
```

```

.text
li $a0, '>'
li $v0,11
syscall
li $v0,5
syscall #Leer un entero

move $a0, $v0 #parámetro a pasar en $a0
jal mult60 #llamamos a la función mult4
move $a0, $v0
    move $a0, $v0 #parámetro a pasar en $a0
jal mult60 #llamamos a la función mult4
move $a0, $v0
jal imprim #Llamamos a la función imprim
li $v0,10 #Acaba el programa
syscall

#-----Funcions-----#
|
imprim: addi $v0,$0,1 #función imprim
syscall #Escribe el valor en $a0
li $a0, '\n' #Salto de línea
li $v0,11
syscall
jr $ra #Vuelve al programa principal

mult60:
sll $v0, $a0, 5 #Función multiplica entero por 60
sll $t0, $a0, 4 #Función multiplica entero por 60
add $v0,$v0,$t0
sll $t0, $a0, 3
add $v0,$v0,$t0
sll $t0, $a0, 2
add $v0,$v0,$t0

jr $ra

```

5. Multiplicación por desplazamientos, sumas y restas

5.1. Escribe la función que multiplique por la constante 11 según el algoritmo de Booth y comprueba que el resultado es correcto.

```

-- program is
ar    >2
      22
-- program is

```

```
.text
li $a0, '>'
li $v0, 11
syscall
li $v0, 5
syscall #Leer un entero

move $a0, $v0 #parámetro a pasar en $a0
jal mult11 #llamamos a la función mult4
move $a0, $v0
jal imprim #Llamamos a la función imprim
li $v0, 10 #Acaba el programa
syscall

#-----Funcions-----#

imprim: addi $v0, $0, 1 #función imprim
syscall #Escribe el valor en $a0
li $a0, '\n' #Salto de línea
li $v0, 11
syscall
jr $ra #Vuelve al programa principal

mult11:
sll $v0, $a0, 4 #Función multiplica entero por 11
sll $t0, $a0, 3
sub $v0, $v0, $t0
sll $t0, $a0, 2
add $v0, $v0, $t0
sub $v0, $v0, $a0
jr $ra
```


Entregas

Escribe el código que lee el valor x y escribe por pantalla la solución de la ecuación: $5x^2+2x+3$.

Pruebas:

Si $x=2 \rightarrow$ resultado 27

```
-- program i:
>2
27
```

Si $x=5 \rightarrow$ resultado 138

```
-- progr
>5
138
```

Si $x=10 \rightarrow$ resultado 523

```
reset: re:
>10
523
-- program
```

Main:

```
syscall
li $v0,5
syscall #Leer un entero

move $a0, $v0 #parámetro a pasar en $a0

jal cuadrado
# hace el cuadrado de x cuadrado guardandolo en $a1
jal multiplicar5
# coge el resultado de $a1 y lo multiplica por 5, guardandolo en $a2
jal multiplicar2
# coge el valor de x de $a0 lo multiplica por dos y lo guarda en $a3
jal sumatodo
# suma $a2 + $a3 + 3 ( 5x^2 + 2x + 3)guardandolo en $a0
jal imprim
# función imprim
li $v0,10 #Acaba el programa
syscall
```

Funciones:

```
#-----Funciones-----#  
imprim:  
addi $v0,$0,1  
syscall  
li $a0, '\n'  
li $v0,11  
syscall  
jr $ra #Vuelve al programa principal  
  
cuadrado:  
mult $v0,$v0  
mflo $a1  
jr $ra #Vuelve al programa principal  
  
multiplicar5:  
sll $a2, $a1, 2 #desplazamiento a la izquierda  
add $a2, $a1,$a2  
jr $ra #Vuelve al programa principal  
  
multiplicar2:  
sll $a3, $a0, 1  
jr $ra #Vuelve al programa principal  
  
sumatodo:  
add $a0,$a2,$a3  
addi $a0,$a0,3  
jr $ra #Vuelve al programa principal
```

Practica 5

Ejemplos

2. Como hacer uso de las instrucciones de salto condicionales

2.3. La instrucción bltz (branch if less than zero) salta si el valor es menor que cero y es similar a bgez ya que también compara un registro con 0 pero siendo contraria la condición de salto. Cambiad la instrucción bgez por bltz, ¿Qué modificaciones tendríais que hacer en el código?

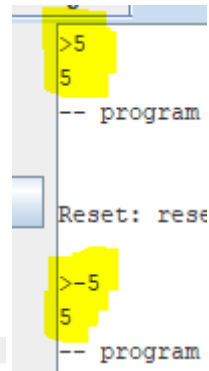
```
.text

li $a0, '>'
li $v0, 11 #Indicación de escribir un valor
syscall
li $v0, 5
syscall #Leer el entero A

bltz $v0, else # Si (A > 0) salta a else
move $a0, $v0 # En $a0 el valor A
j exit #Acaba parte if-then

else:
sub $a0, $zero, $v0 # En $a0 el negativo de A
exit:
li $v0, 1 #Imprimir lo que hay en $a0
syscall

li $v0, 10 #Acaba el programa
syscall
```

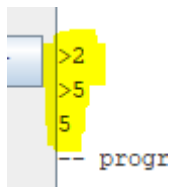


```
>5
5
-- program

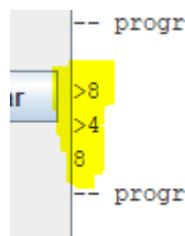
Reset: reset

>-5
5
-- program
```

2.4. Escribid el programa que lea dos enteros del teclado y escriba en la consola el mayor.



```
>2
5
-- progr
```



```
>8
4
8
-- progr
```

```
.text

li $a0, '>'
li $v0, 11 #Indicación de escribir un valor
syscall
li $v0, 5
syscall #Leer el entero A

move $s1, $v0

li $a0, '>'
li $v0, 11 #Indicación de escribir un valor
syscall
li $v0, 5
syscall #Leer el entero b

move $s0, $v0

slt $t0, $s0, $s1 # $t0=1 si a<b
beq $t0, $zero, et1 # mira si son iguales a 0

move $a0, $s1
li $v0, 1 #Imprimir lo que hay en $a0
syscall

li $v0, 10 #Acaba el programa
syscall

et1:
move $a0, $s0
li $v0, 1 #Imprimir lo que hay en $a0
syscall
```

4. Bucles

```
.text
li $s0, 1 #Iniciamos contador
li $s1, 11 #Condición de finalización
li $s2, 0 #Contador
inicio_for:
sle $t1, $s0, $s1
beqz $t1, final_for
add $s2, $s2, $s0 #cuerpo del bucle
addi $s0, $s0, 1 #incremento del contador

move $a0, $s2
li $v0, 1 #Imprimir lo que hay en $a2
syscall
li $a0, '\n'
li $v0, 11
syscall

j inicio_for
final_for:
```

1
3
6
10
15
21
28
36
45
55
66

Añade al código anterior la posibilidad de leer un valor n por teclado y escribe el resultado de la suma por consola. Haz que haya un bucle infinito que lea por teclado excepto en el caso de que se escriba 0, en el cual se saldrá del programa.

```
.text
li $s0, 1 #Iniciamos contador

li $v0, 5
syscall #Leer el entero A
move $s1, $v0

li $s2, 0 #Contador

inicio_for:
add $s2, $s2, $s0 #cuerpo del bucle
addi $s0, $s0, 1 #incremento del contador
move $a0, $s2
li $v0, 1 #muestra a0
syscall

li $a0, '\n' #salto de linea
li $v0, 11
syscall

sle $t1, $s0, $s1
beqz $t1, final_for

j inicio_for
final_for:

li $v0, 10
syscall
```

9
1
3
6
10
15
21
28
36
45

-- progra:
2
1
3

-- progra:

Reset: re

5
1
3
6
10
15

Entregas

1. Haz el código que lea dos enteros de la consola y escriba la suma y vuelva a comenzar si el resultado es distinto de 0. Es pseudocódigo sería:

```
.text

seguí:
li $a0, '>'
li $v0, 11 #Indicación de escribir un valor
syscall

li $v0, 5
syscall #Leer el entero A

move $s1, $v0

li $v0, 11 #Indicación de escribir un valor
syscall

li $v0, 5
syscall #Leer el entero b

move $s0, $v0

add $s2, $s1, $s0 #suma entero a y b

move $a0, $s2
li $v0, 1 #Imprimir lo que hay en $a2
syscall

li $a0, '\n'
li $v0, 11
syscall

li $a0, '\n'
li $v0, 11
syscall

#dobles salto para que quede mas visual
beq $s2, $zero, acabar # compara si es 0
# en ese caso acaba

j sigui
acabar:
li $v0, 10
syscall

-- program is finished running --
```

2. Haz el código que lee de teclado dos valores positivos A y B en los que $A < B$. El programa tiene que escribir por consola los valores comprendidos entre ambos, incluyéndolos a ellos mismos. Es decir, si $A=3$ y $B=6$, escribe en la consola 3 4 5 6 (puedes escribir, por ejemplo, un salto de línea después de cada uno de los valores a mostrar).

```
.text

li $a0, '>'
li $v0, 11
syscall

li $v0, 5 #Leemos el primer numero(A)
syscall

move $s0, $v0 #Guardamos en $s0.

li $a0, '>'
li $v0, 11
syscall

li $v0, 5 #Leemos el segundo numero(B)
syscall

move $s1, $v0 # Guardamos en $s1.

#inicio del bucle
sumar1:
move $a0, $s0
li $v0, 1 # muestra a0.
syscall

li $a0, '\n'
li $v0, 11
syscall

beq $s0, $s1, fin
# Si el numero correspondiente en esa iteracion
#es igual que B, el programa finaliza saltando a fin.

#si no, sigue sumando 1
add $s0, $s0, 1 # suma 1

j sumar1

fin:
li $v0, 10
syscall
```

```
>4
>7
4
5
6
7

-- program is finished

Reset: reset completed

>5
>18
5
6
7
8
9
10
11
12
13
14
15
16
17
18

-- program is finished
```

Practica 6

Ejemplos

4. Acceso a la memoria

4.2. ¿Cuántos bytes de la memoria principal están ocupados por datos del programa?

12 bytes

4.3. ¿Cuántas instrucciones de acceso a la memoria contiene el programa?

3

4.4. ¿Qué valor tiene el registro \$t1 cuando se ejecuta la instrucción lw \$s1,0(\$t1)?

La direccion de b

4.5. ¿En qué dirección se almacena el resultado?

En c

4.6. Sustituid la instrucción sw \$s2,0(\$t2) por sw \$s2,2(\$t2) ¿Qué ocurre cuando se intenta ejecutar el programa? Razonad la respuesta

Error, cogemos una palabra que no es multiplo de 4

4.7. ¿Cuál es la codificación en lenguaje máquina de la instrucción lw \$s1,0(\$t1)? Desglosa la instrucción en los distintos campos del formato.

0x8d310000 → 8d=10001101 → 1001

4. Uso de la memoria

4.3. ¿Cuál es la dirección del byte donde está el carácter '1'?

0x10010020 + 4=0x10010024

5. Los punteros

5.2. Ensambla el código y ejecútalo. ¿Qué hace la función 4 para la instrucción syscall?

Imprime cadenas de caracteres

5.3. ¿Cuál es la codificación máquina de la instrucción syscall?

0x000000c

5.4. ¿En qué dirección se guarda el resultado de la suma?

0x10010008

Entregas

Escribe el código que lee dos enteros del teclado mostrando sendos mensajes por consola: uno que pida al usuario que introduzca el primer valor y tras haberlo leído que muestre otro solicitando el segundo valor. Los datos se almacenarán en la memoria, para lo cual debes haber reservado previamente espacio en el segmento de datos. Una vez almacenados los datos tienes que llamar a una función, que denominaremos SWAP, que intercambie el contenido de las dos posiciones de memoria. Para finalizar se leerán los valores guardados en la memoria y se mostrarán ordenados de menor a mayor en la pantalla.

MAIN:

```
.data
A: .word 0
B: .word 0
VAE1: .asciiz "Primer entero: "
VAE2: .asciiz "Segundo entero: "

.text
# impresion del mensaje 1
la $a0,VAE1
li $v0,4
syscall

li $v0,5
syscall #Leer un entero A

la $t0,A
sw $v0,A #lo guarda en A
# impresion del mensaje 2
la $a0,VAE2
li $v0,4
syscall

li $v0,5
syscall #Leer un entero B

la $t1,B
sw $v0,B #lo guarda en A

jal swap # intercanvia los valores
jal comparar # compara cual es menor
#e imprime

li $v0,10
syscall
```


FUNCIONES:

```
#-----FUNCIONES-----#

swap:
lw $s0,0($t1) # b es ahora a
lw $s1,0($t0) # a es ahora b

sw $s0,A #lo guarda en A
sw $s1,B #lo guarda en b

jr $ra

comparar:
blt $s1,$s0,mayorA # salta si A es mayor
#si menor a
    move $a0,$s0 #muestra A
li $v0,1
syscall

li $a0, '\n'
li $v0,11
syscall

move $a0,$s1 #muestra b
li $v0,1
syscall

jr $ra

mayorA:
#si menor b
move $a0,$s1 #muestra B
li $v0,1
syscall

li $a0, '\n'
li $v0,11
syscall

move $a0,$s0 #muestra A
li $v0,1
syscall

jr $ra
```

Ejemplo de ejecucion:

se introducen los datos

```
Primer entero: 5
Segundo entero: 8
```

se guardan en memoria

Address	Value (+0)	Value (+4)
268500992	0x00000005	0x00000008
268501024	0x00000005	0x00000008

los intercambian

\$s0	16	0x00000008
\$s1	17	0x00000005

finalmente los ordena e imprime

```
Primer entero: 5
Segundo entero: 8
5
8
```

```
Primer entero: 5
Segundo entero: 3
3
5
-- program is finishe

Reset: reset complete

Primer entero: 7
Segundo entero: 9
7
9
-- program is finishe

Reset: reset complete

Primer entero: 7
Segundo entero: 5
5
7
-- program is finishe
```