FUNDAMENTOS DE MICROPROCESADORES
PROBLEMAS DE LA UNIDAD-3.- EL PROCESADOR I: EL LENGUAJE MAQUINA

3.1. Un programa escrito para MIPS invoca a la subrutina func, a la que transfiere tres argumentos a, b y c. La subrutina tiene dos variables locales m y n. Mostrar la posición del puntero de pila y los contenidos de los registros relevantes de la pila, inicialmente vacía, cuando se invoca la subrutina en las líneas del programa que se indican:
a) Justo antes de la llamada en la línea rotulada como # Apartado a), antes de la instrucción **jal.**
b) Cuando se completa la zona de pila para **func.**
c) Justo antes de ejecutar la instrucción **lw** en la línea comentad con # Apartado c).

```
# Programa para llevar a la pila los argumentos a, b y c
        jal     func            # Apartado a)
        lw      $s1, 0($t2)     # Apartado c)
        …       …
func:                           # Comienzo de la subrutina
        …       …
        jr      $ra
```

3.2. En el proceso de enlace, un determinado código simbólico genera una tabla de símbolos que asocia a cada símbolo la posición de memoria en donde se ubicará tras el proceso de carga. Se pide escribir la tabla de símbolos que corresponde al código siguiente. Utilizar “U” para los símbolos indefinidos.

```
        .text   0x0100
        j       main
        .text   0x0200
main:   or      $s2, $s1, 0x1000
        srl     $s2, $s2, 10
lab_4:  sw      $s2, K($0)
        addi    $s1, $s1,  -1
foo:    sw      $s1, K ($0)
        and     $s1, $s1, $0
        beq     $s1, $s2, lab_5
        .data    0x3000
cons:   .space  12
K:              0x4000
```

3.3. Un desensamblador es un programa que lee un modulo objeto y genera un lenguaje simbólico. Dado el siguiente código objeto se pide el correspondiente lenguaje desensamblado para MIPS. Dado que el código objeto no contiene información para poder determinar los nombres de los símbolos, se les asignará ordenadamente las letras del abecedario a medida que sean necesarias.

```
0010 0010 0011 0001 0000 0000 0000 0001
0000 0010 1100 0000 1010 1000 0010 0101
0001 0010 0011 0101 0000 0000 0000 0010
0000 0000 0001 0110 1011 0010 1000 0010
0000 1100 0000 0000 0000 0000 0000 0000
0000 0011 1110 0000 0000 0000 0000 1000
```

3.4. Escribir dos funciones o subrutinas para MIPS llamadas **push** y **pop** que sirvan, la primera a partir de la posición 0x0100 de memoria para escribir dos registros en la pila del sistema y la segunda a partir de la posición 0x0200 para leer dos registros que estén guardados en la pila del sistema. Utilice como argumentos los registros de MIPS definidos para el paso de parámetros.

FUNDAMENTOS DE MICROPROCESADORES
PROBLEMAS DE LA UNIDAD-3.- EL PROCESADOR I: EL LENGUAJE MAQUINA

3.5. El programa escrito para MIPS que se muestra invoca a una subrutina **sub1**, a la que transfieren dos argumentos A y B. La subrutina **sub1**, utiliza dos variables locales M y N y devuelve una variable resultado Z. Por su parte **sub1** invoca a la función **func1** a la que transfiere un único argumento P y de la que recibe también una variable S. La función **func1** necesita utilizar una variable local. Si el paso de variables se realiza por medio de la pila y el valor del puntero de pila antes del comienzo del programa es 0x0004000, se pide, utilizando la estructura facilitada para justificar cada respuesta, el contenido del registro $sp y los valores guardados en la pila en los siguientes hitos de la ejecución.

a) Justo antes de ejecutar la instrucción rotulada a.
b) Justo antes de ejecutar la instrucción rotulada d.
c) Justo antes de ejecutar la instrucción rotulada func1.
d) Justo antes de ejecutar la instrucción rotulada f.
e) Justo antes de ejecutar la instrucción rotulada e.
f) Escribir las instrucciones que se deben escribir en la posición rotulada con las letras b y c.

```
        # Programa para resolver

a:      jal     sub1
b:      # Instrucción nueva 1
c:      # Instrucción nueva 2
        # ---------------------------------------
sub1: # Comienza la subrutina sub1
        # ----
d:      jal     func1
        # ----
e:      jr      $ra     # retorno de sub1
        # ---------------------------------------
func1:  # Comienza la función func1
        # ----
        # ----
f:      jr      $ra    # retorno de func 1
```

3.6. Dado el siguiente programa escrito para MIPS, en donde se mezclan instrucciones en ensamblador y en código máquina (hexadecimal). Se pide, **a)** completar el código ensamblador y **b)** Indicar la función que ejecuta el código y escribir el valor de la posición de memoria señalada por la etiqueta F.

# Problema 3.6	
	.text 0x0000
	0x8C112000
	add $s3, $s1, $0
	0x2232FFFF
	beq $s1, $0, fin
L1:	**0x0C00000A**
	addi $s2, $s2, -1
	beq $s2, $0, fin
	j L1
fin:	sw $s3, F($0)
parar:	**0x08000009**
func:	and $s6, $s6, $0
	add $s7, $s2, $0
L2:	add $s6, $s3, $s6
	addi $s7, $s7, -1
	beq $s7, $0, L3
	0x0800000C
L3:	add $s3, $s6, $s0
	0x03E00008
	.data 0x2000
N:	0x0003
F:	0x0000

3.7. Dado el siguiente programa escrito para MIPS, en donde se mezclan instrucciones en ensamblador y en código máquina (hexadecimal). Se pide, **a)** completar el código ensamblador y **b)** Indicar la función que ejecuta el código y escribir el valor del registro $s7 al final del proceso completo.

# Problema 3.7	
	.text 0x0000
	lw $s1, X($0)
	0x8C132004
	nor $s2, $s1, $s1
	nor $s4, $s3, $s3
	and $s5, $s4, $s1
	and $s6, $s3, $s2
	or $s7, $s6, $s5
	0x08000007
	.data 0x2000
X:	0x0055
Y:	0x00AA

FUNDAMENTOS DE MICROPROCESADORES
PROBLEMAS DE LA UNIDAD-3.- EL PROCESADOR I: EL LENGUAJE MAQUINA

3.8. Escribir en código ensamblador para MIPS, una función llamada **mover**, que sirva para trasladar una tabla de datos con tamaño de celda de 4 bytes desde una ubicación inicial a otra final. Los argumentos que se le pasarán a la función por medio de los registros internos $a0, $a1y $a2 serán, longitud de la tabla (en bytes), dirección origen (primera posición de memoria origen) y dirección destino (primera posición de memoria destino) respectivamente. La longitud, deberá ser siempre un múltiplo de 4 y mayor que cero.
Nota: escribir junto a cada instrucción un comentario explicativo.

3.9. Se pide completar el programa dado para MIPS que calcula el producto de dos números P = M*N, donde N se puede escribir como $N = 2^X - 2^Y$. En memoria figuran inicialmente M, X e Y, siendo X e Y positivos entre 0 y 31.
Nota: Señalar un breve comentario de cada una de las instrucciones que se escriban.

(*) Es equivalente poner lw $t1, M($0) que lw $t1, M. En ambos casos el ensamblador tendrá que traducir la etiqueta X a un dato inmediato que sumará al registro $0 para componer la dirección.

# Programa 3.9	**Comentario**
.text 0x0000	
lw $t1, M($0)	#Lee M en $t1 (*)
....	.....
....	.....
sw $s1, P($0)	#escribe el resultado
fin: j fin	
.data 0x2000	
M:	
X:	
Y:	
P:	

3.10. Utilizando el lenguaje ensamblador de MIPS, escribir una función denominada **swap** que sirva para intercambiar el contenido de dos registros cualesquiera del banco de registros (por ejemplo entre $s0 y $s1), sin modificar ningún otro registro de propósito general.
a) Utilice la pila para la transferencia de parámetros entre el programa principal y la rutina.
b) Realice la misma función, pero en este utilice los registros específicos de MIPS para el paso de parámetros a procedimientos ($a0-$a3) y para el retorno de resultados ($v0-$v1). Además de la función **swap**, complete el código de llamada a la función y el de retorno después de la misma.
Nota: Se valorará que el programa funcione y que utilice el menor número de instrucciones posible.

3.11. El tamaño de todas las instrucciones de un determinado procesador es de 24 bits. El procesador dispone de un banco de registros de propósito general con 64 registros. En el set de instrucciones se distinguen tres tipos diferentes. El número de instrucciones para cada uno de los tres tipos es: 14 con un formato que utiliza tres registros operando (dos fuentes y uno destino), 47 que utilizan dos (uno fuente y uno destino) y 4 que utilizan un solo registro operando (destino).
Nota: Considerar que el código de operación (OP) no tiene que ser del mismo tamaño para todas las instrucciones.
Indicar, justificando la respuesta:

- **a)** Los campos necesarios para el formato de cada uno de los tipos de instrucciones señaladas. Es decir cuántos bits se utilizan para indicar el tipo de instrucción, cuántos para el código de operación, cuántos para la dirección de registros y cuántos para los otros campos considerados.
- **b)** ¿A cuántas direcciones de memoria diferentes se puede acceder en instrucciones cuyo formato utiliza un único registro operando? Señalar un posible ejemplo para este tipo de instrucciones y cómo sería la operación que realiza.

FUNDAMENTOS DE MICROPROCESADORES
PROBLEMAS DE LA UNIDAD-3.- EL PROCESADOR I: EL LENGUAJE MAQUINA

3.12. Sea una arquitectura de 16 bits (tamaño de palabra y de registro) con 6 registros y un espacio de direccionamiento de 64 kBytes. La ALU es capaz de realizar 20 operaciones distintas (sumar, restar, and, or, etc…). En la resolución del problema se debe tener en cuenta que esta arquitectura no es MIPS y que:

1. Los códigos de operación tienen todos la misma longitud.
2. No todas las instrucciones tienen necesariamente el mismo tamaño.
3. Se puede leer/escribir de memoria un solo byte.

Se pide, **justificando necesariamente la respuesta**, diseñar el formato de instrucción que permita construir un juego de instrucciones con los siguientes tipos (un formato para cada tipo):

Tipo1: Instrucciones entre registros en la ALU, al menos 40 instrucciones.

Tipo2: Instrucciones para leer/escribir en memoria con direccionamiento absoluto a cualquier posición de memoria (la dirección está contenida en la instrucción). Al menos 5 instrucciones.

Tipo3: Instrucciones con memoria con direccionamiento relativo a un registro, con un desplazamiento máximo de 128 bytes (la dirección se obtiene sumando un dato inmediato al contenido de un registro). Al menos 5 instrucciones.

Tipo4: Instrucciones de salto condicional, con la posibilidad de saltar a cualquier otra instrucción guardada en memoria. Al menos 4 instrucciones.

3.13. Se quiere escribir un programa para guardar en la posición de memoria señalada por el literal **resul**, la suma de los números en hexadecimal 0xFF00 y 0x00FF. Se facilita parte del programa, señalando con una línea discontinua la posición que deben ocupar las instrucciones o valores que faltan y que el estudiante debe completar.

```
    .text 0x0800
main:  ①-------------------------------------# Escribir la instrucción que falta
       ②-------------------------------------# Escribir la instrucción que falta
       add $s1, $s2, $s3
       ③-------------------------------------# Escribir la instrucción que falta

    .data 0x2000
val1:          0xFF00
resul: ④-------------------------------------# Escribir el resultado del programa
```

a) Se pide construir la tabla de variables del programa dado.

b) Con la ayuda de la tabla de códigos adjunta, escribir en hexadecimal el código máquina para la instrucción: add $s1, $s2, $s3

c) Completar las líneas de código y los valores señalados en el programa.

FUNDAMENTOS DE MICROPROCESADORES
PROBLEMAS DE LA UNIDAD-3.- EL PROCESADOR I: EL LENGUAJE MAQUINA

3.14. Sea un sistema procesador basado en MIPS como el estudiado en clase. Con la ayuda de las tablas de código facilitadas, se pide:

a) Codificar en lenguaje máquina la instrucción: **beq $s1,$t1, fin.** Indique el resultado en binario y en hexadecimal.

b) Encontrar la codificación de la instrucción escrita en lenguaje máquina del código dado. Con los operandos obtenidos en la codificación, señale la función que ejecuta esta instrucción.

3.15. Se muestra el código MIPS para un cierto programa principal, main y dos rutinas sub1 y sub2 a las que la rutina principal llama durante su ejecución. Para el paso de parámetros entre rutinas, se utiliza la pila del sistema controlada por el puntero de la pila que utiliza el registro sp, que siempre señala a la última posición ocupada de la pila. Las tres rutinas están desarrolladas por personas diferentes por lo que hay que respetar los protocolos para la protección de registros que afectan a las rutinas escritas para MIPS.

```
        .text 0x0000
main:   addi $s3, $0, 0x000A
        lw $s5, A($0)
        addi $sp, $sp, -4
        sw $s5, 0($sp)
        beq $s3, $s5, salto1
salto2: jal sub1
        lw $s3, 0($sp)
        addi $sp, $sp, 4
        ori $s1, $s3, 0xFF00
        sw $s1, R($0)
        j fin
salto1: jal sub2
        lw $s4, 0($sp)
        addi $sp, $sp, 4
        andi $s1, $s4, 0xFF00
        sw $s1, R($0)
fin :   j fin

.data 0x2000
A: 0x0001
R:
```

```
        .text 0x0100
sub1:   lw $s1, 0($sp)
        addi $t2, $0, 0x0004
        sllv $t3, $s1, $t2
        sw $t3, 0($sp)
        jr $ra
```

```
        .text 0x0200
sub2:   lw $t3, 0($sp)
        addi $t5, $0, 0x0002
        sllv $t1, $t3, $t5
        sw $t1, 0($sp)
        jr $ra
```

Se pide,

a) Analizar el código dado y señalar:
   a1. En hexadecimal el valor del símbolo R.
   a2. En hexadecimal el valor que contiene la posición de memoria R.

b) Justificando brevemente la respuesta, señalar el número de parámetros de entrada y salida entre la rutina llamante y cada una de las rutinas llamadas.

c) Justificando brevemente la respuesta, aunque no afecte necesariamente al resultado final, señalar el error o errores cometidos por el programador de la rutina sub1.

FUNDAMENTOS DE MICROPROCESADORES
PROBLEMAS DE LA UNIDAD-3.- EL PROCESADOR I: EL LENGUAJE MAQUINA

3.16. Se sabe que el siguiente código realiza la media aritmética (redondeando hacia abajo) de los valores de las posiciones de memoria *A, B, C* y *D*, dejando el resultado en *R*. Para ello la función principal, *main*, llama a otra función, *med4*, que es la que realiza la media aritmética de sus cuatro operandos de entrada. El intercambio de información con dicha función se realiza mediante los registros reservados para tal fin en MIPS.
Se pide completar la función *med4* con las tres instrucciones que faltan (cada hueco corresponde a una y sólo una instrucción), así como dar el valor final del registro *$ra,* de la posición de memoria *R* y de las etiquetas *A*, *med4* y *fin*.

.text 0
lw $a0, A
lw $a1, B
lw $a2, C
lw $a3, D
jal med4
sw $v0, R
fin: j fin
	.text 0x0100
	med4: add $t0, $a0, $a1
	_____
	add $t0, $t0, $t1
	_____
	_____
		.data 0x2000
		A: 5
		B: 6
		C: 7
		D: 8
		R: 0

3.17. Se adjunta un programa en C para MIPS que implementa varias funciones y utiliza la pila para el paso de argumentos y el uso de variables locales. Se pide definir el estado de la pila en determinados instantes de ejecución del código. Considerar que el puntero a pila ($sp) empieza en la posición x4000 y que el compilador no realiza ninguna optimización de código. Indique, para cada caso, el valor real escrito en pila cuando sea posible saberlo. Todas las rutinas, la principal y las secundarias, utilizan la pila para guardar sus variables locales.

void main(){ *int aux;* *aux=fun1(3, 7);* **(a)** *fun2(aux);* **(d)** *}* **(f)**	*int fun1(int a, int b){* *int k;* **(b)** *k = a+b;* *return k+1;* **(c)** *}*	*void fun2(a){* *printf("El número es\n",a);* *return;* **(e)** *}*

Se pide el valro del registro $sp y el contenido de la pila en los siguientes hitos del programa:
a) En el "jal" en MIPS, **b)** en donde señala la etiqueta b, **c)** en el "jr" en MIPS, **d)** en el "jal" en MIPS, **e)** en el "jr" en MIPS y **f)** una vez terminado el programa *main*.

3.18. Se tienen dos números con signo guardados en dos posiciones de memoria cualesquiera etiquetadas como A y B y se desea escribir en las mismas posiciones la suma y el valor medio de ambos números respectivamente. Complete el programa para MIPS adjunto, que permita ejecutar esta tarea, y en el que existe una función denominada **SumMed** que recibe como parámetros de entrada los números y devuelve la suma y el valor medio de sus contenidos.

a. Utilice la pila para la transferencia de parámetros entre el programa principal y la rutina llamada. Complete el código de llamada a la función y el de retorno después de la misma.
b. Utilice los registros específicos de MIPS para el paso de parámetros a procedimientos ($a0-$a3) y para el retorno de resultados ($v0-$v1). Complete el código de llamada a la función y el de retorno después de la misma.

Nota: Se valorará que el programa funcione y que utilice el menor número de instrucciones posible, así como la adecuada inclusión de comentarios en la respuesta. Deje sin completar en la tabla las filas que no necesite.

3.19. Un programa escrito para MIPS, llama a una subrutina utilizando los registros específicos para el paso de parámetros.

a) Sabiendo que la subrutina llamada recibe dos parámetros y devuelve uno, señale la instrucción o instrucciones que debe escribir en la rutina principal para guardar el parámetro recibido en la posición de memoria 0x2008, así como su código máquina en hexadecimal.
b) Escriba un programa para MIPS para escribir en el registro $t4 la constante 0x12345678. Se valora la corrección del código y el uso del menor número de instrucciones posible. Comente brevemente la solución propuesta.
c) Escriba las instrucciones necesarias dentro de un programa para MIPS para leer de memoria el valor de un número guardado en la posición de memoria identificada por la etiqueta o símbolo A y escribir su complemento a dos en la posición siguiente. Se valora la corrección del código y el uso del menor número de instrucciones posible. Comente brevemente la solución propuesta.