			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
PC	0x00	**0x00000000**	0x00700293	**0x0000004**	0x00100313	**0x0000008**	0x0062f333	**0x0000000c**	0x00030463	**0x00000010**	0xfff28293	**0x00000014**	0x0012d293						
SP		0x00000010																	

	PC	SP	IR(hexa)	Instrucción	PC
1	0x00000000		00700293	addi x5, x0, 0x7	0x00000004
	Ejecución	Escribe en el registro x5 la suma entre x0 (0) y 0x7 (7). No modifica memoria			
2	0x00000004		00100313	addi x6, x0, 0x1	0x00000008
	Ejecución	Escribe en el registro x6 la suma entre x0 (0) y 0x1 (1). No modifica memoria			
3	0x00000008		0062f333	and x6, x5, x6	0x0000000c
	Ejecución	Realiza un AND entre los valores de x5 y x6. El resultado es escrito en x6. No modifica memoria			
4	0x0000000c		00030463	beq x6, x0, 0x8	0x00000010
	Ejecución	Si el valor de x6 es igual a 0, realiza un salto de 2 instrucciones (PC + 0x00000008). Como es falso, continua en 0x00000010			
5	0x00000010		fff28293	addi x5, x5, -1	0x00000014
	Ejecución	Escribe en el registro x5 la suma entre x5 (7) y -1. No modifica memoria			
6	0x00000014		0012d293	srli x5, x5, 1	0x00000018
	Ejecución	Realiza un Shift Right al valor guardado en x5 (Una sola vez). No modifica memoria			
7					
	Ejecución				
8					
	Ejecución				
9					
	Ejecución				
10					
	Ejecución				
11					
	Ejecución				
12					
	Ejecución				
13					
	Ejecución				
14					
	Ejecución				
15					
	Ejecución				

REGISTRO	VALOR
x0 / zero	
x1 / ra	
x2 / sp	
x3 / gp	
x4 / tp	
x5 / t0	~~0x7~~, ~~0x6~~, 0x3
x6 / t1	0x1
x7 / t2	
x8 / s0 / fp	
x9 / s1	
x10 / a0	
x11 / a1	
x12 / a2	
x13 / a3	
x14 / a4	
x15 / a5	
x16 / a6	
x17 / a7	

REGISTRO	VALOR
x18 / s2	
x19 / s3	
x20 / s4	
x21 / s5	
x22 / s6	
x23 / s7	
x24 / s8	
x25 / s9	
x26 / s10	
x27 / s11	
x28 / t3	
x29 / t4	
x30 / t5	
x31 / t6	

PC	0x00
SP	

Memoria			
0x0000000	0x00400593		
0x00000004	0x0005a603		
0x00000008	0x00400693		
0x0000000c	0x0006a683		
0x00000010	0x0006a683		
0x00000014	0xfed606e3		
0x00000018	0x0080006f		
0x0000001c	0x0000006f		
0x00000020	0xfffa6737		
0x00000024	0x9fd70713		
0x00000028	~~0x00c70633~~	~~0x00000004~~	0x00000000
0x0000002c	0x02b62423		
0x00000030	0xfedff06f		

	PC	SP	IR(hexa)	Instrucción	PC
1	0x00000008		00400593	addi x11, x0, 4	0x0000000c
	Ejecución	Escribe en el registro x11 la suma entre x0 (0) y 0x4 (4). No modifica memoria			
2	0x0000000c		0005a603	lw x12 0 x11	0x00000010
	Ejecución	Escribe en x12 lo que vale en la posicion de memoria 0(x11)			
3	0x00000010		00400593	addi x13, x0, 4	0x00000014
	Ejecución	Escribe en el registro x13 la suma entre x0 (0) y 0x4 (4). No modifica memoria			
4	0x00000014		0006a683	lw x13 0 x13	0x00000018
	Ejecución	Escribe en x13 lo que vale en la posicion de memoria 0(x13)			
5	0x00000018		fed606e3	beq x12 x13 -20	0x00000020
	Ejecución	Si el valor de x12 es igual al de x13, retrocede 5 instrucciones (PC - 0x00000020). Como es falso, continua en 0x00000020			
6	0x00000020		0080006f	jal x0 8	0x00000028
	Ejecución	Almacena el valor actual del PC en x0, pero al guardarlo en x0 este no se modifica y sigue valiendo 0 y actualiza el valor del PC, avanzando 2 instrucciones (PC + 0x00000008)			
7	0x00000028		fffa6737	lui x14 0xfffa6	0x0000002c
	Ejecución	Escribe en el registro x14 el valor 0xfffa6			
8	0x0000002c		9fd70713	addi x14 x14 -1539	0x00000030
	Ejecución	Escribe en el registro x14 la suma entre x14 (0xfffa6) y -1539. No modifica memoria			
9	0x00000030		00c70633	add x12 x14 x12	0x00000034
	Ejecución	Escribe en el registro x12 la suma entre el valor de la posiciones x14 y x12. No modifica memoria			
10	0x00000034		02b62423	sw x11 40(x12)	0x00000038
	Ejecución	Escribe en la posicion de memoria x11 el valor del registro x12			
11	0x00000038		fedff06f	jal x0 -20	0x00000018
	Ejecución	Almacena el valor actual del PC en x0, pero al guardarlo en x0 este no se modifica y sigue valiendo 0 y actualiza el valor del PC, retrocediendo 5 instrucciones (PC - 0x000000020)			
12	0x00000018		0000006f	jal x0 0	0x00000018
	Ejecución	Almacena el valor actual del PC en x0, pero al guardarlo en x0 este no se modifica y sigue valiendo 0 y actualiza el valor del PC, avanzando 0 instrucciones (PC + 0x00000000)			
13	0x00000018		0000006f	jal x0 0	0x00000018
	Ejecución	Almacena el valor actual del PC en x0, pero al guardarlo en x0 este no se modifica y sigue valiendo 0 y actualiza el valor del PC, avanzando 0 instrucciones (PC + 0x00000000)			
14	0x00000018		0000006f	jal x0 0	...
	Ejecución	Almacena el valor actual del PC en x0, pero al guardarlo en x0 este no se modifica y sigue valiendo 0 y actualiza el valor del PC, avanzando 0 instrucciones (PC + 0x00000000)			
15					
	Ejecución	.....			
15					
	Ejecución				

REGISTRO	VALOR
x0 / zero	
x1 / ra	
x2 / sp	
x3 / gp	
x4 / tp	
x5 / t0	
x6 / t1	
x7 / t2	
x8 / s0 / fp	
x9 / s1	
x10 / a0	
x11 / a1	0x00000004
x12 / a2	~~0x0005a603~~, 0x00000000
x13 / a3	~~0x0005a603~~, 0x00000000
x14 / a4	~~0xfffa6000~~, 0xfffa59fd
x15 / a5	
x16 / a6	
x17 / a7	

REGISTRO	VALOR
x18 / s2	
x19 / s3	
x20 / s4	
x21 / s5	
x22 / s6	
x23 / s7	
x24 / s8	
x25 / s9	
x26 / s10	
x27 / s11	
x28 / t3	
x29 / t4	
x30 / t5	
x31 / t6	

2. a) Indicar en qué posiciones de memoria se encuentra cada etiqueta.

- fin: 0x00000014
- resta: 0x00000018
- sigo: 0x00000020

b) Indicar el desplazamiento de las llamadas a etiquetas.
Tanto *j resta* en la línea 3 como en la 7 retroceden dos instrucciones (PC - 0x00000008)

c) Indique el rango de constantes, en decimal y binario que pueden utilizarse en la instrucción *li*. ¿Coinciden con el rango del imm de la instrucción ADDI?
El rango de constantes de li en decimal es de [-(2^31); (2^31)-1] y en binario [10 ... 00; 01…11]. Y no coinciden con el rango del imm de ADDI ya que son de 12 bits.

d) ¿Cómo resuelve los valores inmediatos que no son representables en 12 bits C2?
Cuando li recibe un valor imm no representable en 12 bits c2 se dividen los 32 bits en los primeros 20 para un lui y los 12 restantes para un addi (Si los 12 bits del addi son 0, no se hace el addi).

e) ¿Cuál es el valor final de *a1*?

0x2114

f) ¿Cuál es el valor final de PC?

0x00000014 <fin> por el loop

g) Listar la secuencia descripta por el pc

1. 0x00000000
2. 0x00000004
3. 0x00000008
4. 0x0000000c
5. 0x00000010
6. 0x00000014
7. 0x00000018
8. 0x0000001c

9. 0x00000020 (a0 = 2114)
10. 0x00000018
11. 0x00000014
12. 0x0000001c
13. 0x00000020 (a0 = 0)
14. 0x00000018
15. 0x00000014
16. 0x00000014
17. …. (loop)

h) Reemplazar la segunda instrucción *li a1,2114* de modo que *a1* sea a0 dividido 2 con una única instrucción.
srli a1, a0, 1

3. b) Suponga que el programa hubiese sido cargado en la posición 0x0000 y el PC comienza con ese valor. ¿Cambia la ejecución del programa? ¿De qué manera? ¿Por qué?

La ejecución del programa cambia. Esto es porque en la instrucción lw x12 0 x11, con PC 0x0000 podemos acceder a la posición que corresponde (PC 0x00004) y guardar nuestro valor, en cambio inicializando el PC en 0x00008 no existe la posición 0x00004 y como toda dirección de memoria con un valor de memoria no explicitado vale 0, esta pequeña modificación altera significativamente el lo que nuestro código hace.