Cuestiones de "Evaluación de Expresiones"

1. El valor que se carga en la posición res es de 1
2. Después de esto se carga 0
3. La comparación entre estos dos es (dato1 < dato2)
4. Se debería cambiar la instrucción *slt* por *seq*
5. Resolución utilizando la pseudoinstrucción *sge*

```
.data
dato1: .word 30
dato2: .word 30
res: .space 1
.text
main: lw $t0,dato1($0)  ;cargar dato1 en t0
lw $t1,dato2($0)        ;cargar dato2 en t1
sge $t2,$t0, $t1        ;poner a 1 $t2 si t0>=t1
sle $t3,$t0, $t1        ;poner a 1 $t3 si t0<=t1
and $t2, $t2, $t3       ;poner a 1 $t2 si t2 == 1 && t3 == 1
sb $t2,res($0)          ;almacenar $t2 en res
```

6. En la posición *res* se almacena un 1
7. Después de esto en *res* hay un 0
8. Ahora el resultado es 1
9. Realiza la comparación (dato1 <= dato2)
10. Evaluación utilizando pseudoinstrucciones *sle*

```
.data
dato1: .word 40
dato2: .word 40
res: .space 1
.text
main: lw $t0,dato1($0)  ;cargar dato1 en t0
lw $t1, dato2($0)       ;cargar dato2 en t1
sle $t2, $t0, $t1       ;poner a 1 $t2 si t0<=t1
sb $t2,res($0)          ;almacenar $t2 en res
```

11. Modificando el código anterior sin utilizar pseudoinstrucciones

```
.data
dato1: .word 30
dato2: .word 40
res: .space 1
.text
main: lw $t0,dato1($0)  ;cargar dato1 en t0
lw $t1, dato2($0)       ;cargar dato2 en t1
sgt $t2, $t0, $t1       ;poner a 1 $t2 si t0>t1
bne $t0,$t1,fineval     ;si t0<>t1 salta a fineval
ori $t2,$0,1            ;poner a 1 t3 si t0=t1
fineval: sb $t2,res($0) ;almacenar $t2 en res
```

12. Modificando el código anterior utilizando pseudoinstrucciones *sge*

```
.data
dato1: .word 40
dato2: .word 40
res: .space 1
.text
main: lw $t0,dato1($0)   ;cargar dato1 en t0
lw $t1, dato2($0)        ;cargar dato2 en t1
sge $t2, $t0, $t1        ;poner a 1 $t2 si t0<=t1
sb $t2,res($0)           ;almacenar $t2 en res
```

13. En *res* se carga un 1
14. Ahora se carga un 0
15. Acá también se carga un 0
16. Acá también se carga un 0
17. La comparación que se lleva a cabo es (dato1 <> 0 && dato2 <> 0)
18. Modificación del código con la condición nueva solicitada

```
.data
dato1: .word 40
dato2: .word -50
res: .space 1
.text
main: lw $t8,dato1($0)
lw $t9,dato2($0)
and $t0,$t0,$0
and $t1,$t1,$0
beq $t8,$0,igual
ori $t0,$0,1
igual: beq $t9,$t8,fineval
ori $t1,$0,1
fineval: and $t0,$t0,$t1
sb $t0,res($0)
```

19. Se carga un 1
20. Ahora se carga un 0
21. Acá también se carga un 0
22. La comparación que se ha evaluado es (dato1 <> 0 && dato2 < dato1)

23. Modificación del código con la condición nueva solicitada

```
main: lw $t8,dato1($0)
lw $t9,dato2($0)
and $t2,$t2,$0
and $t1,$t1,$0
and $t0,$t0,$0
beq $t8,$t9,igual
ori $t0,$0,1
igual: slt $t1,$t8,$t9
bne $t8,$t9,fineval
ori $t2,$0,1
fineval:    or $t1, $t1, $t2
            and $t0,$t0,$t1
sb $t0,res($0)
```

24. Modificando el código anterior utilizando pseudoinstrucciones *sle*

```
main: lw $t8,dato1($0)
lw $t9,dato2($0)
and $t1,$t1,$0
and $t0,$t0,$0
beq $t8,$t9,igual
ori $t0,$0,1
igual: sle $t1,$t8,$t9
fineval: and $t0,$t0,$t1
sb $t0,res($0)
```

25. Se carga 0 en *res*
26. Ahora se carga un 1
27. Acá también se carga un 1
28. Acá se carga un 0
29. La comparación que se realiza es (dato1 < dato2 || dato2 == 0)
30. Modificación del código con la condición nueva solicitada

```
.data
dato1: .word 30
dato2: .word -20
res: .space 1
.text
main: lw $t8,dato1($0)
lw $t9,dato2($0)
and $t0,$t0,$0
and $t1,$t1,$0
sle $t0,$t8,$t9
ble $t9,$0,fineval
ori $t1,$0,1
fineval: or $t0,$t0,$t1
sb $t0,res($0)
```

31. Modificando el código anterior utilizando pseudoinstrucciones *sle*

```
.data
dato1: .word 30
dato2: .word -20
res: .space 1
.text
main: lw $t8,dato1($0)
lw $t9,dato2($0)
and $t0,$t0,$0
and $t1,$t1,$0
sle $t0,$t8,$t9
sle $t1, $t9, $0
fineval: or $t0,$t0,$t1
sb $t0,res($0)
```

Cuestiones de "Control de flujo condicional"

1. La instrucción que evalúa la condición es beq
2. El beq sería el if, el then sería lo que se encuentra en la etiqueta "entonces" la cual sólo se ejecuta si beq no se cumple
3. En res se carga el 71
4. Si dato1 es 0, se carga 30 y si dato2 es 0 se carga 40
5. Implementación del programa descrito en pseudocódigo solicitado

```
.data
dato1: .word 40
dato2: .word 0
res: .space 4
.text
main: lw $t0,dato1($0)      ;cargar dato1 en $t0
lw $t1,dato2($0)            ;cargar dato2 en $t1
and $t2,$t2,$0 #t2=0
Si: ble $t1,$0,finsi        ;si $t1 = 0 finsi
entonces: div $t0,$t1       ;t0/$t1
mflo $t2                    ;#almacenar LO en $t2
finsi: add $t3,$t0,$t1      ;$t3=$t0+$t1
add $t2,$t3,$t2             ;$t2=$t3+$t2
sw $t2,res($0)              ;almacenar en res $t2
```

6. Pseudocógido sin mencionar registros

```
VARIABLES
ENTERO: dato1=40; dato2=30; res;
INICIO
if (dato1!=0 && dato2!=0) res=dato1/dato2;

res=res+dato1+dato2;
FIN
```

7. Las instrucciones que evalúan la condición en el programa son los 2 *beq*, estas son la condición contraria a la condición *if* en el pseudocódigo
8. La estructura condicional del *if* está formada por 2 *beq* que forman la condición y el *then* está formada por lo que sigue en la etiqueta “entonces”
9. Se almacena el 71
10. Si dato1 = 0, se almacena 30 y si dato2 = 0, se almacena 40
11. Implementación del programa descrito en pseudocódigo

```
.data
dato1: .word 40
dato2: .word 0
res: .space 4
.text
main: lw $t0,dato1($0)  ;cargar dato1 en t0
lw $t1,dato2($0)        ;cargar dato2 en $t1
and $t2,$t2,$0          ;pone a 0 $t2
Si: blt $t1,$0,finsi    ;si $t1=0 saltar finsi
ble $t0,$0,finsi        ;si $t0 =0 saltar finsi
entonces: div $t0,$t1   ;t0/$t1
mflo $t2                ;almacenar LO en t2
finsi: add $t3,$t0,$t1  ;t3=t0+$t1
```

12. Descripción en lenguaje algorítmico equivalente al programa

```
VARIABLES
ENTERO: dato1=30; dato2=40; res;
INICIO
Si (dato1 >= dato2):
        res = dato2
Sino:
        res = dato1
FIN
```

13. Se almacena 30 en res, si dato1 = 35 se almacena 30 en res
14. El conjunto de instrucciones que implementan la pseudoinstrucción *bge* son *slt* y *beq*

15. Implementación en ensamblador del programa descrito en lenguaje algorítmico

```
.data
dato1: .word 30
dato2: .word 40
res: .space 1

.text
main:
lw $t0,dato1($0)        ;cargar dato1 en t0
lw $t1,dato2($0)        ;cargar dato2 en t1
and $t2,$t2,$0

blt $t0,$t1, else
sub $t2, $t0, $t1
j skip

else:
    sub $t2, $t1, $t0

skip:
    sb $t2,res($0)      ;almacenar $t2 en res
```

16. Descripción en lenguaje algorítmico equivalente al programa

```
VARIABLES
ENTERO: dato1=30; dato2=40, dato3=-1; res;
INICIO
        Si (dato3 < dato1):
                res = 1
Sino:
        Si (dato3 <= dato2):
                res = 0
FIN
```

17. En ambos casos se almacena en *res* el valor 1

18. El siguiente código traducido desde el pseudocódigo sería

```
data
dato1: .word 30
dato2: .word 40
dato3: .word -1
res: .space 4
.text
main: lw $t1,dato1($0)          ;cargar dato1 en $t1
lw $t2,dato2($0)                ;cargar dato2 en $t2
lw $t3,dato3($0)                ;cargar dato3 en $t3
Si: blt $t3,$t1, sino           ;si $t3<$t1 ir sino
bgt $t3,$t2, sino               ;si $t3<=$t2 ir a sino
entonces: addi $t4,$0,1         ;$t4=1
j finsi                         ;ir a finsi
sino: and $t4,$0,$0             ;$t4=0
finsi: sw $t4,res($0)           ;almacenar res
```

19. El programa que implementa la función de bucle *while* funciona de la siguiente manera: Primero, carga la cadena en el registro temporal *$t0* e inicializa el contador de letras en el registro *$t2* en 0. Luego, empieza el bucle con la etiqueta *mientras*, su primer instrucción es almacenar en *$t1* el primer carácter de la palabra. Una vez almacenado, llega la condición de salida. Esta llega cuando ya no quedan palabra por recorrer en memoria, almacenando sucesivamente espacios con valor 0. La ejecución de la condición se repetirá hasta que *$t1* sea distinto a 0. El bucle continúa sumando los contadores *$t2* (nuestra solución) y *$t0* (logrando así un “corrimiento” de caracteres en la palabra). Por último se utiliza la instrucción j mientras, la cual nos devuelve a donde empezó el loop. Una vez se haya conseguido la condición de finalización del bucle, entonces guardamos el contenido de *$t2* en nuestra solución, con el nombre de n.
20. Se almacena un 4, ya que la palabra es “hola”

21. Implementación del programa descrito en pseudocódigo

```
.data
tira1: .asciiz "hola"
tira2: .asciiz "adios"
.align 2
n: .space 4
.text
main: la $t0,tira1                  ;carga dir. tira1 en $t0
la $t1, tira2                       ;carga dir. tira2 en $t1
andi $t3,$t3, 0 #$t2=0
mientras: lb $s0,0($t0)             ;almacenar byte en $s0
lb $s1, 0($t1)                      ;almacenar byte en $s1
beq $s0,$0,finmientras              ;si $s0=0 saltar a finmientras
beq $s1, $0, finmientras            ;si $s1=0 saltar a finmientras
addi $t2,$t2, 1                     ;$t2=$t2+1
addi $t0,$t0, 1                     ;$t0=$t0+1
addi $t1, $t1, 1                    ;$t1 = $t1+1
j mientras                          ;saltar a mientras
finmientras: sw $t2,n($0)           ;almacenar $t2 en n
```

22. Carga los vectores enteros (6,7,8,9,10,1), guarda espacio para la respuesta (*res*). Dentro del *main* se guarda la dirección del vector en *$t2*, se limpia *$t3*, se carga en *$t0* un 0 y en *$t1* se carga un 6. Luego entramos dentro del for que repetirá su ciclo siempre y cuando *$t0* < *$t1* (osea *$t0* < 6), después se carga en *$t4* con la dirección del vector un elemento de este mismo, de esta forma se va aumentando *$t2* para ir corriendo por todos los elementos del vector. Todo esto se va sumando en *$t3*, sumando así todos los valores del array y aumentando a $t0 en 1 por cada valor. Cuando termina el *for* ya que la pseudoinstrucción *bgt* nos hace saltar a finpara, acá se guarda en *res* el resultado de sumar todos los elementos del array.
23. El resultado almacenado es 41

24. Implementación del programa descrito en pseudocódigo

```
.data
v: .word 6,7,8,9,10,-1,34,23
v2: .space 32
.text
main:
la $t2,v
la $t3,v2

li $t0,0                    ;$t0=0
li $t1,7                    ;$t1=7
para: bgt $t0,$t1,finpara
lw $t4,0($t2)               ;cargar primer elemento de v
lw $t5,0($t3)               ;cargar primer elemento de v2

addi $t5, $t4, 1
sw $t5, 0($t3)              ;guardar contenidos modificados del array

addi $t0, $t0, 1
addi $t2, $t2, 4
addi $t3, $t3, 4

j para                      ;saltar a bucle
finpara:
```