PRÁCTICA 1

EJERCICIOS DE PRÁCTICAS - ESTRUCTURA DE COMPUTADORES

1. Dado N un valor entero almacenado en la dirección de memoria 0xf0, diseñar un programa ensamblador que almacene en memoria, a partir de la dirección 0x100, un vector de palabras formado por los N primeros números pares. Nota: Propón un valor de N.

2. Dados tres valores enteros almacenados a partir de la dirección de memoria 0x1000, diseñar un programa ensamblador que sume los tres valores y almacene el resultado en la siguiente palabra de memoria. El programa debe utilizar subrutinas:

   - SUMA_DOS: realiza la operación r2=r2+r3.
   - SUMA_TRES: realiza la operación r5=r2+r3+r4 llamando a SUMA_DOS.
   - Programa principal: realiza la operación pedida llamando a SUMA_TRES.

   Nota: Propón los tres valores a sumar.

3. Genera la codificación binaria de la instrucción: blt r7, r8, LOOP. Suponer que LOOP= 0x0014 y que la dirección donde está blt es 0x20.

4. Escribe un programa en lenguaje ensamblador de NIOS II que calcule la serie de Fibonacci de los 8 primeros números (0, 1, 1, 2, 3, 5, 8, 13). Observar que los dos primeros números son 0, 1.

5. Encuentra el número mayor de la siguiente lista de números enteros: 4, 5, 3, 6, 1, 8, 2. Almacena el resultado en una posición de memoria.

6. Calcular el producto escalar de 2 vectores con 6 elementos cada uno de ellos. Propón los seis valores de 32 bits de cada vector. El resultado debe almacenarse en la posición de memoria siguiente al último componente de uno de los vectores. Si se va a probar en DE2 con procesador NIOSII/e, también se debe incluir una subrutina que realice la multiplicación de enteros.

7. A continuación se muestra una lista de instrucciones ensamblador del procesador NIOS II utilizado en las prácticas de la asignatura. Realizar las siguientes actividades:

- Deducir lo que hace cada instrucción y agrupar las instrucciones similares.
- Identificar cuántos formatos de instrucción existen y qué campos necesitan.
- Clasificar las instrucciones de acuerdo con el formato de instrucción utilizado. Poner en un grupo separado las pseudoinstrucciones.
- Identificar cuántos modos de direccionamiento existen y cómo funcionan.

add rC, rA, rB
addi rB, rA, inmediato16
and rC, rA, rB
andhi rB, rA, inmediato16
andi rB, rA, inmediato16
beq rA, rB, etiqueta
bge rA, rB, etiqueta
bgeu rA, rB, etiqueta
bgt rA, rB, etiqueta
bgtu rA, rB, etiqueta
ble rA, rB, etiqueta
bleu rA, rB, etiqueta
blt rA, rB, etiqueta
bltu rA, rB, etiqueta
bne rA, rB, etiqueta
br etiqueta
call etiqueta
callr rA
cmpeq rC, rA, rB
cmpeqi rB, rA, inmediato16
cmpge rC, rA, rB
cmpgei rB, rA, inmediato16
cmpgeu rC, rA, rB
cmpgeui rB, rA, inmediato16
cmpgt rC, rA, rB
cmpgti rB, inmediato16
cmpgtu rC, rA, rB
cmpgtui rB, inmediato16
cmple rC, rA, rB
cmplei rB, inmediato16
cmpleu rC, rA, rB
cmpleui rB, inmediato16
cmplt rC, rA, rB
cmplti rB, rA, inmediato16
cmpltu rC, rA, rB
cmpltui rB, rA, inmediato16
cmpne rC, rA, rB
cmpnei rB, rA, inmediato16
div rC, rA, rB
divu rC, rA, rB
jmp rA
jmpi etiqueta
ldb rB, desplazamiento(rA)
ldbio rB, desplazamiento(rA)
ldbu rB, desplazamiento(rA)
ldbuio rB, desplazamiento(rA)
ldh rB, desplazamiento(rA)
ldhio rB, desplazamiento(rA)
ldhu rB, desplazamiento(rA)
ldhuio rB, desplazamiento(rA)
ldw rB, desplazamiento(rA)
ldwio rB, desplazamiento(rA)
mov rC, rA
movhi rB, inmediato16
movi rB, inmediato16
movia rB, etiqueta
movui rB, inmediato16
mul rC, rA, rB
muli rB, rA, inmediato16
nop
nor rC, rA, rB
or rC, rA, rB
orhi rB, rA, inmediato16
ori rB, rA, inmediato16
ret
rol rC, rA, rB
roli rC, rA, inmediato5
ror rC, rA, rB
sll rC, rA, rB
slli rC, rA, inmediato5
sra rC, rA, rB
srai rC, rA, inmediato5
srl rC, rA, rB
srli rC, rA, inmediato5
stb rB, desplazamiento(rA)
stbio rB, desplazamiento(rA)
sth rB, desplazamiento(rA)
sthio rB, desplazamiento(rA)
stw rB, desplazamiento(rA)
stwio rB, desplazamiento(rA)
sub rC, rA, rB
subi rB, rA, inmediato16
xor rC, rA, rB
xorhi rB, rA, inmediato16
xori rB, rA, inmediato16

8. Selecciona de la lista adjunta las instrucciones que se deben incluir en las posiciones 4, 6, y 10 del programa en ensamblador para que se pueda ejecutar correctamente en DE2. El programa realiza la siguiente funcionalidad: *N es un valor entero almacenado en la dirección de memoria 0xf0, y el programa almacena en memoria, a partir de la dirección 0x100, un vector de palabras formado por los N primeros números pares*.

Lista de instrucciones:

a) movia r4, V
b) stw r3,0(r4)
c) bne r2,r0,LOOP
d) subi r4,r4,1
e) addi r3,r3,2
f) br LOOP

Programa:

```
1.  _start:
2.  ldw r2,N(r0)
3.  addi r3,r0,2
4.
5.  LOOP:
6.
7.  subi r2,r2,1
8.  addi r3,r3,2
9.  addi r4,r4,4
10.
11. STOP:
12. br STOP
13. .org 0xf0
14. N:      .word 12
15. .org 0x100
16. V:      .skip 48
17. .end
```

9. Reordena la lista adjunta las instrucciones y directivas de ensamblador para que el programa resultado se pueda ejecutar correctamente en DE2 y realice la siguiente funcionalidad: *N es un valor entero almacenado en la dirección de memoria 0xf0, y el programa almacena en memoria, a partir de la dirección 0x100, un vector de palabras formado por los N primeros números pares*.

Lista de instrucciones y directivas de ensamblador:

a) bne r2,r0,LOOP
b) STOP: br STOP
c) V: .skip 48
d) _start:
e) addi r3,r0,2
f) addi r3,r3,2
g) addi r4,r4,4
h) ldw r2,N(r0)
i) LOOP: stw r3,0(r4)
j) movia r4, V
k) subi r2,r2,1
l) .end
m) .org 0x100
n) .org 0xf0
o) N: .word 12

Programa:

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____
11. _____
12. _____
13. _____
14. _____
15. _____