

## EJERCICIOS DE PRÁCTICAS - ARQUITECTURA DE COMPUTADORES

### PRÁCTICA 1

#### Arquitectura del repertorio de instrucciones y programación del procesador Nios2/e

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 DE0-Nano con procesador Nios II/e, también se debe incluir una subrutina que realice la multiplicación de enteros.

Algoritmo de multiplicación de dos datos: a, b

```
# Vamos a obtener el producto de a x b.
# Presuponemos que tanto a como b son positivos.
R=0
Si b !=0 entonces
  mientras b>0 hacer
    si (bit0 de b)=1 entonces
      R=R+a
    fin si
    b=b>>1
    a=a<<1
  fin mientras
fin si
```

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

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 DE0-Nano. 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 DE0-Nano 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. \_\_\_\_\_