PRÁCTICO 8 - Ensamblado y desensamblado de LEGv8, límites de la ISA

Instruction	Rt [4:0]	Instruction	Rt [4:0]
B.EQ	00000	B.VC	00111
B.NE	00001	B.HI	01000
B.HS	00010	B.LS	01001
B.LO	00011	B.GE	01010
B.MI	00100	B.LT	01011
B.PL	00101	B.GT	01100
B.VS	00110	B.LE	01101

Ejercicio 1:

Extender los siguientes números de 26 bits en complemento a dos a 64 bits. Si el número es negativo verificar que la extensión a 64 bits codifica el mismo número original de 26 bits.

- 1.1) 00 0000 0000 0000 0000 0000 0001
- 1.2) 10 0000 0000 0000 0000 0000 0000

Ejercicio 2:

Tenemos las siguientes instrucciones en assembler LEGv8:

ADDI X9, X9, #0 STUR X10, [X11,#32]

- 2.1) ¿Qué formato (R, I, D, B, CB, IM) de instrucciones son?
- **2.2) Ensamblar** a código de máquina LEGv8, mostrando sus representaciones en binario y luego en hexadecimal.

Ejercicio 3:

Dar el tipo de instrucción, la instrucción en assembler y la representación binaria de los siguientes campos de LEGv8:

- **3.1)** op=0x658, Rm=13, Rn=15, Rd=17, shamt=0
- 3.2) op=0x7c2, Rn=12, Rt=3, const=0x4

Ejercicio 4:

Transformar de binario a hexadecimal. ¿Qué instrucciones LEGv8 representan en memoria?

- **4.2)** 1101 0010 1011 1111 1111 1111 1110 0010

Ejercicio 5:

Ejecutar el siguiente código assembler que está en memoria para dar el valor final del registro X1. El contenido de la memoria se dá como una lista de pares, dirección de memoria: contenido, suponiendo alineamiento de memoria del tipo big endian. Describa sintéticamente que hace el programa.

0x10010000: 0x8B010029 0x10010004: 0x8B010121

Ejercicio 6:

Decidir cuáles de las siguientes instrucciones en assembler se pueden codificar en código de máquina LEGv8. Explique qué falla en las que no puedan ser ensambladas.

1.	LSL XZR, XZR, 0	6. STUR X9, [XZR,#-129]
2.	ADDI X1, X2, -1	7. LDURB XZR, [XZR,#-1]
3.	ADDI X1, X2, 4096	8. LSR X16, X17, #68
4.	EOR X32, X31, X30	9. MOVZ X0, 0x1010, LSL #12
5.	ORRI X24, X24, 0x1FFF	10. MOVZ XZR, ØxFFFF, LSL #48

Ejercicio 7:

Ensamblar estos delay loops.

MOVZ X0, 0x1, LSL #48 L1: SUBI X0,X0,#1 CBNZ X0, L1	MOVZ X0, 0xFFFF, LSL #32 L1: SUBIS X0,X0,#1 B.NE L1	MOVZ X0, 0x2, LSL #16 L1: SUBIS XZR,X0,#0 B.EQ EXIT SUBI X0,X0,#1 B L1 EXIT:
---	---	--

Ejercicio 8:

Dadas las siguientes direcciones de memoria:

0x00014000 0x00114524 0x0F000200

- **8.1)** Si el valor del PC es 0x00000000, ¿es posible llegar con **una** sola instrucción **conditional branch** a las direcciones de memoria arriba listadas?
- **8.2)** Si el valor del PC es 0x00000600, ¿es posible llegar con **una** sola instrucción **branch** a las direcciones de memoria arriba listadas?
- **8.3)** Si el valor del PC es 0x00000000 y quiero saltar al primer GiB de memoria 0x40000000 . Escribir exactamente 2 instrucciones contiguas que posibilitan el **salto lejano** (*far jump*).

Ejercicio 9:

Suponiendo que el PC está en la primera palabra de memoria 0x00000000 y se desea saltar a la última instrucción de los primeros 4 GiB o sea a 0xFFFFFFFC, ¿Cuántas instrucciones B son necesarias? (no se puede usar BR).

Ejercicio 10:

¿Qué valor devuelve en X0 este programa?

.org 0x0000 MOVZ X0, 0x0400, LSL #0 MOVK X0, 0x9100, LSL #16 STURW X0, [XZR,#12] STURW X0, [XZR,#12]