Final 2024-02-29 - ejs parc. 2

ACLARACIÓN: Solo están los ejercicios que corresponden a los temas del parcial 2, ya que lo use como practica para ese parcial.

Ejercicio 4:

Realizar de tres formas distintas un delay loop con legv8.

1. Bucle con comparación y saltos:

   loop:

   ```
   cmp x0, #0     // Compara x0 con 0
   b.gt loop      // Si x0 > 0, vuelve a loop
   ```

2. Bucle con subs

   loop:

   ```
   subi x0, x0, #1   // Disminuimos el valor de x0 en 1
   b.ge loop         // si x0 ≥ 0, saltamos a loop
   ```

3. No se me ocurre otra, uso la de sub devuelta jajsja con algunas modificaciones tranquis

   loop:

   ```
   subi x1, x2, #1   //Disminumos el valor de x1 en 1
   b.NE loop         // Si x1 ≠ x2 volvemos a loop
   ```

   (este funciona si el valor de x2 > x1)

Ejercicio 5:

Ensamblar para .org 0x2000 y para .org 0xC00000.

```
MOVZ X0, #0xFFFF, LSL #0
LOOP: CBZ X0, EXIT
            SUBI X0, X0, #1
            B LOOP
EXIT:
```

Primero notemos que lo que ensamblemos va a ser indiferente para .org0x2000 y .org0xC00000, ya que el .org solo nos indica la dirección de memoria donde se cargara el código ensamblado.

Entonces ensamblemos el bloque de código:

MOVZ x0, #0xFFFF, LSL #0

Sabemos que MOVZ es de tipo “IM”, entonces:

opcode	LSL	MOV_immediate	Rd
110100101	00	1111111111111111	00000
Esto porque esta en la green card, y si la green card lo dice así es.	Esto ya que en tenemos 4 opciones: LSL #0 → 00 LSL #16 → 01 LSL #32 → 10 LSL #48 → 11	Esto ya que el immediate en este caso es: #0xFFFF y pasándolo a binario nos queda eso.	Ya que donde lo vamos a guardar es en x0.

Entonces pasando en limpio tenemos: 11010010100111111111111111100000

Ahora ensamblemos esta: CBZ X0, EXIT

vemos que CBZ es de tipo “cb” entonces:

opcode	COND_BR_Address	Rt
10110100	0000000000000000011	00000
Esto porque esta en la green card, y si la green card lo dice así es.	Esto ya que en tenemos que saltar 3 instrucciones si se cumple, y el COND_BR_address ocupa 19 bits por eso hay tantos 0.	Ya que donde lo vamos a guardar es en x0.

Entonces tenemos: 10110100000000000000000001100000

La otra: SUBI X0, X0, #1

vemos que SUBI es de tipo “i” entonces:

opcode	ALU_immediate	Rn	Rd
10101000100	00000000001	00000	00000
Esto porque esta en la green card, y si la green card lo dice así es.	Esto ya que en tenemos que el immediate es #1 ocupa 11 bits.	Ya que lo que vamos a restar es de x0.	Ya que donde lo vamos a guardar es en x0.

Entonces: 10101000100000000000010000000000

Por ultimo: B LOOP

vemos que B es de tipo “b” entonces:

opcode	BR_address
000101	**11111111111111111111101**
Esto porque esta en la green card, y si la green card lo dice así es.	Saltamos dos instrucciones para arriba o para atras ajj, y el campo es de 26 bits.

Entonces: 000101**11111111111111111111101**

Por lo tanto tenemos lo siguiente:

11010010100111111111111111100000 = D29FFFE0

10110100000000000000000001100000 = B4000060

10101000100000000000010000000000 = A8800400

00010111111111111111111111111101 = 17FFFFFD