Taller de Arquitectura

Organización del Computador 1

Grupo 24: nullPointerException

Primer cuatrimestre 2025

Ejercicio 1

PC	IR (hexa)	Instrucción y ejecución	PC sig.
0x0	0x00700293	`addi x5, zero, 7` x5 ← 7	0x04
0x4	0x00100313	`addi x6, zero, 1` x6 ← 1	0x08
0x8	0x0062f333	`and x6, x5, x6` Se preserva la parte baja entre x6 y x5	0x0c
0x0c	0x00030463	`beq x6, x0, 8` No salta, porque x6 ≠ 0	0x10
0x10	0xfff28293	`addi x5, x5, -1` x5 ← x5 - 1	0x14
0x14	0x4012d293	`srai x5, x5, 1` Desplaza aritméticamente x5 una posición	0x18

Registro	Valor
x5 / t0	~~7~~ → ~~6~~ → 3
x6 / t1	~~1~~ → 1

Contenido de registros

Los registros que no figuran en el segundo cuadro no sufrieron modificaciones a lo largo de la ejecución del código

Ejercicio 2

a

- Etiqueta **fin**: 0x14
- Etiqueta **resta**: 0x18
- Etiqueta **sigo**: 0x20

b

PC	Llamado	Desplazamiento	Siguiente PC
0x10	Resta	PC + 0000 0100	0x18
0x20	Resta	PC + 1111 1000	0x18
0x1c	Fin	PC + 0000 0100	0x14
0x14	Fin	PC + 1111 1000	0x14

c

li es una *pseudoinstrucción* que, **si el valor inmediato entra en 12 bits, es lo mismo que hacer addi**. Entonces, li se comporta como:

```
addi x5, x0, 42
```

Rango de li cuando usa solamente addi: 12 bits en complemento a 2, por lo tanto el rango en decimal es [**-2048, 2047**].

Si el valor inmediato *no* entra en 12 bits, li se traduce en lui y luego addi. Entonces li se comporta como:

```
lui x5, 0x1
addi x5, x5, 132
```

Rango de li cuando se usa lui+addi: Se puede cargar cualquier valor entero de 32 bits con signo, ya que lui carga los 20bits más altos y addi los 12 bits bajos con un valor con signo. Por lo tanto, el rango en decimal es [**-2^{31}, $2^{31}-1$**].

d

Los imm de 12 bits se manejan con li, que cuando imm > 12 bits combina *lui* y *addi* para cargar el valor. lui carga los 20 bits altos y addi los 12 bits bajos con un valor con signo. Ejemplo:

```
li a0,  4228:
  lui a0, 1         -> a0 = 0x1000 = 4096
  addi a0, a0, 132 -> a0 = 4096 + 132 = 4228
```

e

El valor final de `a1` es **2114** ya que el mismo nunca se modifica a lo largo del ciclo.

f

El valor final de PC es **0x14** ya que a partir de ahí comienza el bucle de fin. El ciclo se ejecuta hasta que `a0` == 0, al ser `a0` = 4228 y `a1` = 2114 la resta se ejecuta dos veces y termina cuando se salta a fin.

g

PC	Instrucción	Desplazamiento	Siguiente PC
0x00	li1	No hay salto en PC (+8)	0x08
0x08	li2	No hay salto en PC (+8)	0x10
0x10	jump	PC + 0000 0100	0x18
0x18	resta	No hay salto en PC (+4)	0x1c
0x1c	beq	No cumple condición (+4)	0x20
0x20	jump (sigo)	PC + 1111 1000	0x18
0x18	resta	No hay salto en PC (+4)	0x1c
0x1c	beq	PC + 0000 0100	0x14
0x14	beq (fin)	PC + 1111 1000	0x14

h

Instrucción modificada:

```
srai a1, a0, 1
```

srai realiza un shift a la derecha aritmético, es decir que divide por 2 si `a0` es positivo

Ejercicio 3

Ciclo	PC	IR (hexa)	Instrucción y ejecución	PC sig.
1	0x08	0x00400593	`addi x11, x0, 4` $x11 \leftarrow 4$	0x0c
2	0x0c	0x0005a603	`lw x12, 0, x11` $x12 \leftarrow \text{MEM}[4] =$ `0x0005a603`	0x10
3	0x10	0x00400693	`addi x13, x0, 4` $x13 \leftarrow 4$	0x14
4	0x14	0x0006a683	`lw x13, 0, x13` $x13 \leftarrow \text{MEM}[4] =$ `0x0005a603`	0x18
5	0x18	0x0006a683	`lw x13, 0, x13` $x13 \leftarrow \text{MEM}[0x005a6023] = 0$	0x1c
6	0x1c	0xfed606e3	`beq x12, x13, -20` $x12 \neq x13 \Rightarrow$ no salta	0x20
7	0x20	0x0080006f	`jal x0, 8` Salta a 0x28 (`guardar`)	0x28
8	0x28	0xfffa6737	`lui x14, 0xfffa6` $x14 \leftarrow$ `0xfffa6000`	0x2c
9	0x2c	0x9fd70713	`addi x14, x14, -1539` $x14 \leftarrow x14 + (-1539) =$ `0xfffa59fd`	0x30
10	0x30	0x00c70633	`add x12, x14, x12` $x12 \leftarrow x14 + x12 =$ `0xfffa59fd + 0x0005a603`	0x34
11	0x34	0x02b62423	`sw x11, 40, x12` $\text{MEM}[x12 + 40] \leftarrow x11 = 4$	0x38
12	0x38	0xfedff06f	`jal x0, -20` `0x38 - 0x14 = 0x24` $\Rightarrow$ salta a 0x24 (`fin_programa`)	0x24

En el ciclo 10 se hace la suma `0xfffa59fd + 0x0005a603 = 0x100000000`. Lo anterior es un número de 33 bits, pero el valor almacenado por RISC-V en `x12` es `0x00000000` ya que son registros de 32bits.

Los casos donde se hace `jal x0, val` son saltos: no se guarda ningún valor (x0 contiene siempre 0) y se salta directamente de PC en PC.