# Taller de Ciclo de Instrucción

Sistemas Digitales

## 1. Introducción

El presente taller consiste en una serie de ejercicios en los cuales se deberá realizar en papel el ensamblado de diversos códigos fuente y luego sus correspondientes seguimientos utilizando las *Planillas de Seguimiento*. Será posible comparar, analizar y validar los seguimientos realizados en papel utilizando el *Simulador RIPES*.

## 2. Planilla de seguimiento: Reglas de notación

- Cuando un registro pasa a tener un nuevo valor, la celda que muestra el valor viejo debe ser tachada.
- Todos los valores de los registros y las posiciones de memoria que no se definan explicitamente, comienzan con valor 0.
- Los desplazamientos se expresan decimal, pero expreselos en complemento a 2 de 8 . **No pueden aparecer en decimal**.
- Toda instrucción escrita en la columna de *Instrucción Decodificada* debe corresponderse a la decodificación de la palabra presente en el registro IR
- En la celda de *Ejecución* debe mostrarse cómo se produce la actualización de todos los registros afectados y los pasos intermedios que considere necesarios.
- Para toda instrucción de salto debe indicarse explícitamente en la celda de *Ejecución* si se produce o no, en caso de que sí, su justificación.

## 3. Ejercicios

Para los próximos códigos realizar el seguimiento con la planilla

## 3.1. Ejercicio 1

Realizar el seguimiento de una ejecución del siguiente código binario decodificandolo para instrucciones de Risc-V. El contenido de las posiciones de memoria desde 00 hasta 14 aparecen al lado de cada una.

```
00: 00700293
04: 00100313
08: 0062f333
0c: 00030463
10: fff28293
14: 4012d293
```

## 3.2. Ejercicio 2

2.a Dado el siguiente código en Risc-V, y asumiendo que el PC inicia en 0x00000000, comentar cada línea y responder:

```
          li a0,4228
          li a1,2114
          jal ra, resta
fin:      beq zero, zero, fin
resta:
prologo:  addi sp, sp,-4
          sw ra,0(sp)
          sub a0,a0,a1
          beq a0,zero,epilogo
 sigo:    jal ra, resta
epilogo:
          lw ra, 0(sp)
          addi sp, sp,4
          ret
```

*a*) Indicar en qué posiciones de memoria se encuentra cada etiqueta.

*b*) Indicar el desplazamiento de las llamadas a etiquetas.

*c*) Indique el rango de constantes, en decimal y binario que pueden utilizarse en la instrucción *li*. ¿Coinciden con el rango del imm de la instrucción ADDI?

*d*) ¿Cómo resuelve los valores inmediatos que no son representables en 12 bits C2?

*e*) ¿Cuál es el valor final de $a1$?

*f*) ¿Cuál es el valor final de $PC$ ?

*g*) Listar la secuencia descripta por el *pc*

*h*) Indique qué valores toman los registros *ra* y *sp*: al inicio, durante y al finalizar la ejecución.

*i*) Reemplazar la segunda instrucción *li a1,1023* de modo que a1 sea a0 *dividido* 2 con una única instrucción

### 3.3. Ejercicio 3

3.a Realizar el seguimiento del siguiente programa por al menos 12 cilos de instrucción, qué comportamiento presenta? Asumir que el PC arranca en 0x08 y que toda dirección de memoria con un valor de memoria no explicitado vale 0.

```
00000008 <main>:
    08:        00400593        addi x11 x0 4
    0c:        0005a603        lw x12 0 x11
    10:        00400693        addi x13 x0 4
    14:        0006a683        lw x13 0 x13
    18:        0006a683        lw x13 0 x13
    1c:        fed606e3        beq x12 x13 -20 <main>

00000020 <guardar>:
    20:        fffa6737        lui x14 0xfffa6
    24:        9fd70713        addi x14 x14 -1539
    28:        00c70633        add x12 x14 x12
    2c:        02b62423        sw x11 40 x12

00000030 <fin_programa>:
    30:        00000513        addi x10 x0 0
    34:        05d00893        addi x17 x0 93
    38:        00000073        ecall
```

3.b Suponga que el programa hubiese sido cargado en la posición 0x0000 y el PC comienza con ese valor. ¿Cambia la ejecución del programa? ¿De qué manera? ¿Por qué?

```
00000000 <main>:
    00:        00400593        addi x11 x0 4
    04:        0005a603        lw x12 0 x11
    08:        00400693        addi x13 x0 4
    0c:        0006a683        lw x13 0 x13
    10:        0006a683        lw x13 0 x13
    14:        fed606e3        beq x12 x13 -20 <main>

00000018 <guardar>:
    18:        fffa6737        lui x14 0xfffa6
    1c:        9fd70713        addi x14 x14 -1539
    20:        00c70633        add x12 x14 x12
    24:        02b62423        sw x11 40 x12

00000028 <fin_programa>:
    28:        00000513        addi x10 x0 0
    2c:        05d00893        addi x17 x0 93
    30:        00000073        ecall
```