

Taller de Arquitectura - Ciclo de Instrucción

Sistemas Digitales

Primer Cuatrimestre 2025

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 la *Planilla de Seguimiento* para describir los pasos. Link a la planilla: Será posible comparar, analizar y validar los seguimientos realizados en papel utilizando el *Simulador RIPES*.

En el link <https://campus.exactas.uba.ar/mod/page/view.php?id=37235> o directamente en el campus entrando a Guías de ejercicios/ Simulador RISC-V: Ripes se encuentran las indicaciones para bajar e instalar Ripes para Ubuntu, Windows y Mac. Además contiene dos videos, que recomendamos ver con mucha atención, ya que contiene lo necesario para el uso del simulador.

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 explícitamente, comienzan con valor 0.
- Los desplazamientos en el programa se expresan en decimal, pero en la planilla expreselos en complemento a 2 de 8 bits. **No pueden aparecer en decimal a la hora de codificar la solución.**
- 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.

Ejercicios

Para los próximos códigos realizar el seguimiento con la planilla

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

```

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
        j resta
fin:    beq zero, zero, fin

resta:  sub a0,a0,a1
        beq a0,zero, fin

sigo:   j resta

```

- Indicar en qué posiciones de memoria se encuentra cada etiqueta.
- Indicar el desplazamiento de las llamadas a etiquetas.
- 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?
- ¿Cómo resuelve los valores inmediatos que no son representables en 12 bits C2?
- ¿Cuál es el valor final de *a1*?
- ¿Cuál es el valor final de *PC* ?
- Listar la secuencia descrita por el *pc*
- Reemplazar la segunda instrucción *li a1,2114* de modo que *a1* sea *a0 dividido 2* con una única instrucción

Ejercicio 3

3.a Realizar el seguimiento del siguiente programa por al menos 12 ciclos 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>
20:      0080006f      jal x0 8 <guardar>

00000024 <fin_programa>:
24:      0000006f      jal x0 0 <fin_programa>

00000028 <guardar>:
28:      fffa6737      lui x14 0xfffa6
2c:      9fd70713      addi x14 x14 -1539

```

```
30:      00c70633      add x12 x14 x12
34:      02b62423      sw x11 40 x12
38:      fedff06f      jal x0 -20 <fin_programa>
```

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>
18:      0080006f      jal x0 8 <guardar>

0000001c <fin_programa>:
1c:      0000006f      jal x0 0 <fin_programa>

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
30:      fedff06f      jal x0 -20 <fin_programa>
```