Nro. ord.	Apellido y nombre	L.U.
	Micaela Yaque	808/19

ORGANIZACIÓN DEL COMPUTADOR I - **Parcial**

Segundo Cuatrimestre 2022

Ej.1	Ej.2	Ej.3	Ej.4	Ej.5	Nota

Corrector:

Aclaraciones

- Anote apellido, nombre, LU en *todos* los archivos entregados.
- El parcial es domiciliario y todos los ejercicios deben estar aprobados para que el parcial se considere aprobado. Hay dos fechas de entrega, en ambos casos el conjunto de ejercicios a entregar es el mismo. En la primera instancia deberán defender su trabajo frente a su tutorx, quien les ayudará también a encaminar el trabajo de los ejercicios pendientes, si los hubiera.
- El link de entrega es: `https://forms.gle/QKFkwLu2PtjXhesf9`. Ante cualquier problema pueden comunicarse con la lista docente o preferentemente con el/la corrector/a.
- La fecha límite de entrega es el domingo 13 de Noviembre a las 21:00. El coloquio será el jueves 24 de Noviembre en el horario de cursada de los jueves (TM: 9 a 13hs - TT: 17 a 21hs) de **forma presencial** en un aula que figura en el calendario.
- Todas las respuestas deben estar correctamente justificadas.

Introducción

Este parcial está dividido en cinco preguntas, las primeras cuatro se relacionan con la arquitectura RISC-V. Se trata de una especificación y no una implementación de un procesador concreto. Si bien la arquitectura es diferente a la que utilizamos en clase, encontrarán los mismos conceptos que estudiamos y trabajamos aplicados al diseño de esta ISA. Toda la información necesaria está disponible en la `Guía Práctica de RISC-V` que se puede acceder libremente en: `http://riscvbook.com/spanish/guia-practica-de-risc-v-1.0.5.pdf`. Les recomendamos que hagan primero una lectura completa de los primeros tres capítulos de la guía y luego intenten responder las preguntas. Los ejercicios varían en temática y complejidad así que también les recomendamos ordenar la resolución de los mismos como les resulte más sencillo. Las últimas dos preguntas se relacionan con extensiones que introdujimos a nuestra implementación de `Orga1SmallI`

Ejercicios

Ejercicio 1 ¿Cuál es el problema de tener algunas instrucciones en nuestro ISA donde el orden de destino y fuente se invierten?

Ejercicio 2 Para este ejercicio utilizar un simulador de RISC-V en el cual ejecutar sus programas, les recomendamos que primero tengan un boceto de solución ya sea en papel o archivo de texto sobre el cuál ordenar el trabajo a realizar y ordenar las preguntas que guíen una nueva lectura de la referencia que presentamos al comienzo del parcial. Algunos de los simuladores disponibles son:

- Ripes `https://github.com/mortbopet/Ripes`
- Rars `https://github.com/TheThirdOne/rars`

Para utilizar Ripes les recomendamos descargar la imagen de Linux de este link:
`https://github.com/mortbopet/Ripes/releases/download/v2.2.5/Ripes-v2.2.5-linux-x86_64.AppImage`
Recuerden habilitar el permiso de ejecución antes de lanzar el programa.

Complete el cuerpo del siguiente programa de assembler de `RV32I` para conseguir una implementación que vaya desplazando el valor que se encuentre en el registro `x10` de un bit por vez y vaya sumando cada valor que toma en el registro `x11`. El programa debe finalizar una vez que el valor del registro `x10` sea igual a `0x0`. Por ejemplo, si el valor inicial de `x10` es `0x8` el valor final de `x11` debe ser `0xF`

```
#este es un comentario
.text #comenzamos con esta directiva

#completar

#finalizamos con estas lineas
fin: addi x17, x0, 0x11
ecall
```

desplazando el valor en x10 de un bit por vez y sume lo que se desplaza en x11
x10 = 0000 0000 0000 1000
x11 = 8 + 4 + 2 + 1 = 15 = 0xF

x11 = x10
x10= desplazamiento>>
x11 = x11+x10
x10=desplazamiento>>...

Utilice las instrucciones básicas presentadas en la referencia y, de ser posible, una única etiqueta a´dicional (aparte de `fin`). ¿En qué posiciones de memoria, omitiendo la finalización, quedarán codificadas las instrucciones? Justifique el uso de los registros utilizados y presente una decodificación de las instrucciones agregadas.

en realidad, codificación

Cada programa debería tener el programa en assembler de RV32I y su equivalente en formato binario (según especificado en el manual) con las etiquetas resueltas y reemplazadas por sus posiciones de memoria.

Ejercicio 3 ¿Cuáles son los objetivos no funcionales que guían el diseño del set de instrucciones de `RISC-V`? ¿Cómo impactan en su diseño?

Ejercicio 4 ¿~~Qué problema puede introducir deshabilitar interrupciones cuando se atiende una interrupción en Orga1SmallI~~?

En realidad debería decir: "¿Qué problema puede introducir habilitar interrupciones cuando se atiende una interrupción en Orga1Small?"

La necesidad de realizar numerosas operaciones sobre datos estructurados en memoria parece seguir creciendo mientras que el costo y la escala de las operaciones realizadas por el procesador se vuelven más costosas en tiempo e infraestructura. Debido a esto se están evaluando alternativas para realizar diversas operaciones en memoria sin tener la obligación de mover datos entre los registros y la memoria principal.

Proponemos el uso de un controlador de memoria con las siguientes señales de control:

- `mem_op` es una señal de tres bits que indican la operación a realizar
  - `000` no realiza ninguna operación
  - `001` permite guardar el valor del registro de destino (`wrAddr`)
  - `010` permite guardar el valor del registro de fuente (`rdAddr`)
  - `011` realiza una escritura en la dirección de destino desde el valor encontrado en inData (equivale al `load` de nuestra arquitectura)
  - `100` suma el valor que se encuentra en la dirección de fuente al valor que se encuentra en la direcicón de destino y lo guarda en la dirección de destino
  - `101` multiplica el valor que se encuentra en la dirección de fuente por el valor que se encuentra en la direcicón de destino y lo guarda en la dirección de destino
  - `110` limpia el valor que se encuentra en la dirección de destino
  - `111` mueve el valor que se encuentra en la dirección de fuente (`rdAddr`) y lo copia en la dirección de destino (`wrAddr`)
- `enOut` permite volcar el valor encontrado en la direción de fuente en el registro outData

Y las siguientes señales de datos:

- `wrAddr` indica la dirección de destino (la dirección de la palabra donde realizaremos una escritura)
- `rdAddr` indica la dirección de fuente (la dirección de la palabra donde realizaremos una lectura)
- `inData` indica el valor de la palabra a guardar en la dirección de destino
- `outData` indica el valor de la palabra a leer de la dirección de fuente

Ejercicio 5 Suponiendo que las señales de control están conectadas a la unidad de control (`mem_op`, `enOut`) y los registros de datos al bus (`wrAddr, rdAddr, inData, outData`):
Escriba un microprograma que permita sumar el contenido del valor que se encuentra en la dirección referida por el registro indicado en el **operando X del decoder** consigo mismo.