DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
ESCUELA DE INGENIERÍA
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE

IIC2343 - Arquitectura de Computadores (I/2023)

Tarea 3

Respuestas sin desarrollo o justificación no tendrán puntaje.

Publicación: Lunes 15 de Mayo 10:00
Entrega: Viernes 2 de Junio 20:00

Instrucciones

- Lea atentamente el enunciado. Responda cada pregunta en un PDF por separado. Ponga su nombre y número de alumno/a en cada pregunta. Adicionalmente, debe incluir archivos de código con extensión `.asm` o `.py`, asegurando que el nombre de los archivos incluya el número y letra de la pregunta.
- La entrega de la tarea será a través de **canvas**.
- La tarea debe ser en formato **digital** y no manuscrito. De no cumplir esto, **no** se revisará dicha pregunta.
- Cualquier duda que tengan respecto a la tarea, preguntar en el foro de clases.
- **Cualquier uso de material externo al curso debe ser citado**.
- Para el uso de cupones de atraso, debe responder el siguiente formulario.
- Siga el código de honor.

Código de Honor de la UC

"Como miembro de la comunidad de la Pontificia Universidad Católica de Chile me comprometo a respetar los principios y normativas que la rigen. Asimismo, prometo actuar con rectitud y honestidad en las relaciones con los demás integrantes de la comunidad y en la realización de todo trabajo, particularmente en aquellas actividades vinculadas a la docencia, el aprendizaje y la creación, difusión y transferencia del conocimiento. Además, velaré por la integridad de las personas y cuidaré los bienes de la Universidad."

Pregunta 1: RISC-V (30 ptos.)

RISC-V es un *Instruction Set Architecture* (ISA) del tipo *Reduced Instruction Set Architecture* (RISC) que sigue un esquema *load-store* y, además, es *open source*, siendo mantenida por RISC-V International y propuesta originalmente por Krste Asanović y David Patterson en la Universidad de California, Berkeley.

Para poder trabajar con el assembly descrito por la especificación de RISC-V, usaremos el emulador RARS, que sólo requiere tener Java 8 o superior instalado para correr en sus computadores. Este emulador entrega acceso al set de instrucciones básico más las extensiones para punto flotante, palabras de largo variable, entre otras y algunas pseudo-instrucciones. Son libres de usar el emulador o plataforma que quieran, pero tengan en consideración que las correcciones se harán usando RARS 1.6.

(a) (4 ptos.) La especificación de RISC-V incluye una convención de llamada, que a partir de la versión V20191213 se encuentra definida como parte del psABI. Investigue y escriba brevemente qué indica la convención de llamada para la extensión base RV32I, los aspectos más relevantes de esta, cómo se utiliza y por qué es necesaria. Para las siguientes secciones de la tarea deberá escribir programas en *assembly* RISC-V, por lo que es importante que preste atención a cómo se utiliza la convención de llamada.

(b) (4 ptos.) Para esta pregunta, deberá desarrollar un programa que detecte si un número es primo gemelo. Un número primo p es gemelo si existe otro número primo q tal que $|p-q| = 2$. En este programa, recibirá en la sección `.data` una etiqueta `N`, que corresponde a un entero positivo. Al finalizar la ejecución, deberá usar el `ecall 1` de RARS para imprimir en consola un 1 si N es primo gemelo y un 0 en otro caso. La primera parte de su archivo se debiese ver de la siguiente forma:

```
.globl start
.data
    N: .word 11
.text
    start:
        # do stuff
```

Se evaluará la correctitud del código, que este respete la convención de llamada y que pueda aceptar un número N arbitrario como *input*.

(c) (8 ptos.) Para esta pregunta, deberá programar una pequeña versión de juego FizzBuzz en RISC-V ASM. Para esto, recibirá en la sección `.data` una etiqueta `N`, que corresponde a un entero positivo y deberá escribir, a partir de la etiqueta `out`, todos los números entre 0 y `N` como `.word`, con la salvedad de que si el número es divisible por 3, este se reemplaza por 70 (ASCII "F"); en caso de ser divisible por 5, se reemplaza por 66 (ASCII "B"); y si es divisible por ambos, se reemplaza por 7066. Además, si un número es primo gemelo, deberá reemplazarlo por 80 (ASCII "P"). A modo de ejemplo, si $N = 15$ la secuencia que se guardará en `out` será: `7066, 1, 2, 80, 4, 80, 70, 80, 8, 70, 66, 80, 70, 80, 14, 7066`
La primera parte de su archivo se debiese ver de la siguiente forma:

```
.globl start
.data
    N: .word 10
    out: .word
.text
    start:
        # do stuff
```

Se evaluará la correctitud del código, que este respete la convención de llamada y que pueda aceptar un número N arbitrario como *input*.

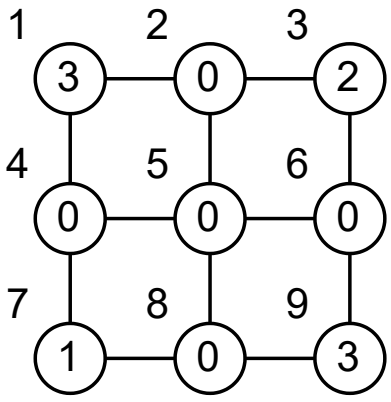
(d) (14 ptos.) Para esta pregunta, deberá resolver un puzzle utilizando RISC-V ASM. Recibirá como *input* un entero `N` y dos arreglos, `M` y `T`. `N` representa la cantidad de nodos de un grafo; `M` representa su matriz de adyacencia; y `T` representa el tipo de cada uno de sus nodos. Existen cuatro tipos de nodo:

- `Inicio` = `1`
- `Final` = `2`
- `Punto` = `3`
- `Vacío` = `0`

El objetivo del puzzle es trazar un camino desde `Inicio` hasta `Final`, conectado a todos los nodos `Punto`. El camino no puede pasar dos veces por el mismo vértice ni borde.

Como resultado, su programa deberá usar el `ecall 1` de RARS para imprimir, en orden, los vértices que recorre el camino de la solución.

A modo de ejemplo, un posible puzzle junto con su matriz de adyacencia, lista de tipo y solución, podría ser el siguiente:



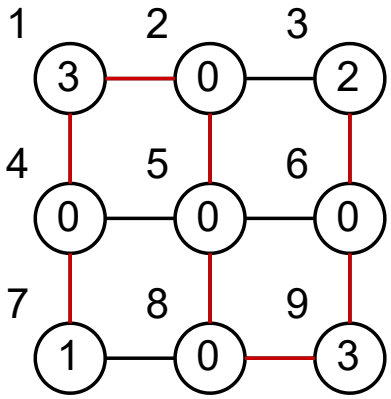
Su matriz de adyacencia es:

$$\begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Y su lista de tipos corresponde a:

$$3, 0, 2, 0, 0, 0, 1, 0, 3$$

La solución sería:



$$7, 4, 1, 2, 5, 8, 9, 6, 3$$

El *input* asociado podría verse de la siguiente manera:

```
.globl start
.data
    # --- No modificar labels ---
    N: .word 9
    M: .word 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
     0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0,
      0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0
    T: .word 3, 0, 2, 0, 0, 0, 1, 0, 3
    # --- End no modificar labels---
    # de aca para abajo van sus variables en memoria
.text
    start:
        # do stuff
```

Se evaluará que se respete la convención de llamada y que se entregue un *output* correcto a partir de una serie de casos de prueba.

Si lo desea, puede optar a un máximo de **6 ptos.** si realiza un programa que solo entregue un camino válido entre el nodo `Inicio` y el nodo `Final` del grafo entregado, sin necesidad de pasar por todos los nodos `Punto`.

Pregunta 2: Input/Output (30 ptos.)

(a) (15 ptos.) Luego del accidente de East Plestine, Ohio, se decide actualizar los sistemas de monitoreo de borde de vía para trenes (WTMS), que consisten en un pequeño computador instalado al borde de una vía con una serie de sensores conectados que miden diferentes parámetros de operación de un tren (temperatura de los ejes, sobredimensión o protuberancias, derrame de líquidos, etc.) y transmiten esta información por radio a la locomotora, junto con almacenar un reporte. Adicionalmente, el sistema se puede actualizar agregando nuevos sensores o componentes a lo largo de su vida útil.

Asuma que existe un componente de control que implementa la ISA de RISC-V. Este componente, correspondiente a un microcontrolador, tiene puertos de entrada y salida digitales llamados GPIO (*General Purpose Input/Ouput*), cuyos registros de estado y comando son de 32 bits y se encuentran mapeados en memoria desde la dirección 0x10030000. Estos son:

Offset	Nombre	Descripción
0x00	`dir_ise`	Contiene la dirección ISR del manejo de alerta del tren
0x04	`reg_tr`	Registro de comandos del tren
0x08	`sensor_te`	Registro de estado de sensor de temperatura
0x0C	`sensor_pr`	Registro de estado de sensor de protuberancia
0x18	`sensor_dr`	Registro de estado de sensor de derrame
0x20	`rad_cl`	Registro de comandos de la radio

A su vez, las especificaciones de estado o comando por registro se definen en la siguiente tabla:

Nombre	Comando o Estado	Valor
`reg_tr`	Notificar locomotora	255
`reg_tr`	Freno de emergencia	127
`sensor_te`	Temperatura alta	255
`sensor_te`	Temperatura baja	1
`sensor_te`	Temperatura normal	127
`sensor_pr`	Sobredimensión del tren o protuberancias	4
`sensor_pr`	Rieles libres de obstáculos	2
`sensor_dr`	Existencia de derrame	3
`sensor_dr`	Sin derrame de fluidos en el tren	1
`rad_cl`	Mandar reporte de sensores	1

Escriba la ISR que se encargará de apagar activar el freno de emergencia, notificar a la locomotora y mandar un reporte cuando un estado sea no deseado. Un estado no deseado se define cuando tiene la combinación de uno o más de los siguientes estados: derrame; temperatura distinta a la normal; existencia de elementos sobresalientes. Por políticas del seguro de calidad del sistema, es requerido que la interacción con cada registro de estado de cualquier sensor se haga a través de una subrutina dentro de la ISR, respetando el estándar de llamadas de RISC-V. Asuma que esta ISR ya está referenciada en el vector de interrupciones correspondiente y utilice la instrucción `mret` de RISC-V para finalizarla.

(b) (15 ptos.) La librería `fakeio`, escrita en Python 3.10, entrega una clase que simula un controlador de dispositivos de I/O *Port-Mapped*. La librería viene acompañada de una pequeña documentación de uso, pero a grandes rasgos expone una interfaz `OUT` que toma como argumentos un puerto, un valor y no retorna nada; e `IN`, que toma como argumento solo un número de puerto y retorna un *string*. Los dispositivos disponibles son:

- Una cinta magnética, `tape`.
- Un generador de enteros aleatorios, `entropy`.
- Una impresora, `printer`.
- Un coprocesador de números de punto flotante, `coprocessor`.

Deberá elaborar un pequeño programa en Python que haga uso **exclusivo** de esta librería y que realice las siguientes interacciones:

1. Leer un par de caracteres de la cinta magnética a partir de una posición `data` obtenida del dispositivo `entropy`.
2. Realizar tres operaciones diferentes en el coprocesador usando como operandos los caracteres obtenidos de la cinta magnética, interpretados como su valor ASCII.
3. Escribir el resultado de la operación al inicio de la cinta magnética.
4. Escribir tres caracteres de la cinta magnética en la impresora y terminar con un salto de línea.

La librería `fakeio` se encuentra en el siguiente enlace, junto con su documentación. Esta librerería necestia Python 3.10 o superior para funcionar.

Si encuentra algún *bug*, puede reportarlo en las *issues* del repositorio para arreglar su funcionamiento.