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

IIC2343 - Arquitectura de Computadores (II/2024)

Interrogación 2 - RISC-V

Instrucciones

Lea atentamente el enunciado. Tendrá hasta las 23:59 del lunes 14 de octubre para subirlo a Canvas. Cualquier duda que tenga, debe realizarla a través de las *issues* del repositorio del curso usando el *label* "`Actividad RISC-V`". Se responderán dudas hasta las 19 horas del día lunes 14 de octubre. Luego de ese horario, no se responderán más preguntas en ningún medio.

Pregunta 4: Concurso de Belleza Pokémon (6 ptos.)

La jueza Susana quiere evaluar la belleza criaturas de bolsillo llamadas Pokémon™ en un concurso de belleza. Para facilitar su objetivo, usted deberá simular el concurso de los Pokémon *starters* de Santiago haciendo uso del lenguaje de ensamblado RISC-V en el emulador RARS.

Cada Pokémon estará ubicado en la sección de `.data` y tendrá los siguientes valores almacenados de manera **consecutiva**.

Offset	Nombre	Descripción
0	**ID**	Número del Pokémon en la Pokédex.
4	**Tipo**	Puede ser tipo: planta (0), fuego (1) o agua (2).
8	**Listones**	Número de listones del Pokémon.
12	**Estado**	Estado de salud del Pokémon. Saludable (0), Envenenado (1), Paralizado (2) o Quemado (3).
16	**Estrellas doradas**	Cantidad de estrellas doradas del Pokémon.
20	**Ataque**	Puntos de ataque físico del Pokémon.
24	**Defensa**	Puntos de defensa físico del Pokémon.
28	**Ataque Especial**	Puntos de ataque especial del Pokémon.
32	**Defensa Especial**	Puntos de defensa especial del Pokémon.
36	**Velocidad**	Puntos de velocidad del Pokémon.
40	**Objeto equipado**	Objeto que tiene el Pokémon: Piedra (0), Pañuelo (1), Huevo (2), Gafas (3)
44	**Rol Batalla**	Como se usa en concurso: Físico (0), Especial (1), Mixto (2), Soporte (3)
48	**Estrellas plateadas**	Cantidad de estrellas plateadas del Pokémon.

El concurso va a consistir en la presentación entre dos Pokémon. Se enfrentarán hasta que alguno de ellos llegue a tener 10 **listones**. De esta manera, el flujo del concurso es el siguiente:

1. El Pokémon con mayor velocidad se presenta primero.
2. Luego, es el turno del Pokémon con menor velocidad para presentarse.
3. Se presenta una jueza a premiar o castigar a alguno de los dos participantes.
4. El Pokémon con mayor estrellas plateadas gana una estrella dorada. En caso de empate, ninguno recibe estrella.
5. Si alguno de los Pokémon llega a tener 5 estrellas doradas, la presentación finaliza y el ganador se lleva un **listón**. En caso de empate, el Pokémon con mayor **ID** gana.
6. Luego, se revisa lo siguiente:
   - Si un Pokémon se encuentra **envenenado**, pierde la mitad de sus estrellas plateadas (división entera). Luego tiene 30 % de probabilidad de volver a estado Saludable.
   - Si un Pokémon está **quemado**, pierde la mitad de sus estrellas doradas (división entera). Luego, tiene 40 % de probabilidad de volver a estado Saludable.
   - Si un Pokémon se encuentra **paralizado**, pierde una estrella dorada. Luego, tiene 70 % de probabilidad de volver a estado Saludable.
7. El flujo anterior se repite hasta que haya un ganador.

Este flujo **ya se encuentra implementado** en un archivo base adjunto a este enunciado, **¡no debe programarlo!** Su tarea consistirá en **completar** el código de un conjunto de subrutinas, las que son descritas en las siguientes secciones. Adicionalmente, en este mismo archivo contará con subrutinas complementarias **ya implementadas** para facilitar su desarrollo.

(2.5 ptos.) 1. Presentación: Subrutina `presentation`

La subrutina `presentation` tiene como fin darle a un Pokémon cierta cantidad de estrellas de plata al presentarse, actualizando su contador interno. Esta debe tomar como argumentos las direcciones del Pokémon que presenta y su contrincante a través de los registros `a0` y `a1` respectivamente.

Para ganar estrellas, el Pokémon usará sus puntos de belleza. Estos se computan de la siguiente forma:

- Si un Pokémon tiene el objeto Piedra, `Puntos belleza = Puntos Ataque`.
- Si un Pokémon tiene el objeto Pañuelo, `Puntos belleza = Puntos Ataque especial`.
- Si un Pokémon tiene el objeto Huevo, `Puntos belleza = Puntos Defensa`.
- Si un Pokémon tiene el objeto Gafas, `Puntos belleza = Puntos Defensa especial`.

Para computar la cantidad de estrellas que gana por presentación, debe usar la siguiente fórmula:

```
Estrellas plata = Puntos belleza (atacante)
                  -Puntos belleza (defensor)
                  +Ponderador Rol
```

`Ponderador rol` corresponde a un ponderador que indica si la presentación es más efectiva (o no) dado el tipo del Pokémon que presenta respecto al de su contrincante. Se calcula de la siguiente forma:

- Si el Pokémon tiene un rol físico y su contrincante un rol especial, `Ponderador rol = -3`.
- Si el Pokémon tiene un rol especial y su contrincante un rol físico, `Ponderador rol = +3`.
- Si el Pokémon tiene rol mixto, `Ponderador rol = +1`.
- Si ambos Pokémon tienen rol soporte, `Ponderador rol = -2`.
- En cualquier otro caso, `Ponderador rol = 0`.

Importante: Si se computa un total **negativo** en `Estrellas plata`, estas se **restan** del total del Pokémon. No obstante, la cantidad de `Estrellas plata` de este debe ser mayor o igual a 0. Es decir, no puede quedar con una cantidad negativa de `Estrellas plata`.

En resumen, la subrutina debe hacer lo siguiente:

1. Computar `Puntos belleza` del Pokémon que presenta según su Objeto equipado.
2. Computar `Puntos belleza` del Pokémon contrincante según su Objeto equipado.
3. Computar `Ponderador Rol` según el rol de ambos Pokémon.
4. Computar `Estrellas plata` según la fórmula entregada.
5. Actualizar el valor `Estrellas plateadas` del Pokémon que presenta según lo computado, asegurando que el total sea mayor o igual a 0.
6. Retornar a donde fue llamada.

(1 pto.) 2. Turno presentación: Subrutina `presentation_turn`

Para llevar acabo la presentación por turnos, se implementará la subrutina `presentation_turn`. Esta debe tomar como argumentos las direcciones del primer y segundo Pokémon a través de los registros `a0` y `a1` respectivamente, y hacer lo siguiente:

1. Se debe decidir el orden en que deben presentar. El Pokémon con mayor **velocidad** es el que debe ir primero. Si un Pokémon está con estado Paralizado, su velocidad será reducida a un cuarto del valor original (división entera). Si hay empate de velocidad, el Pokémon de mayor **ID** presenta primero.

2. Presentar al primer Pokémon a partir del llamado de la subrutina `presentation`. Se debe guardar la dirección base del primer Pokémon en presentar en el registro `a0`, y la dirección base del segundo Pokémon en presentar se debe guardar en el registro `a1`.
3. Presentar al segundo Pokémon a partir del llamado de la subrutina `presentation`. Se debe guardar la dirección base del segundo Pokémon en presentar en el registro `a0`, y la dirección base del primer Pokémon en presentar se debe guardar en el registro `a1`.
4. Retornar a donde fue llamada.

(2.5 ptos.) 3. Fase jurado: Subrutina `judge`

Luego de finalizar un turno de presentación, existirá una jueza que premiará con estrellas de plata o castigará a los Pokémon. La intensidad del juicio dependerá de la cantidad de listones totales que se hayan entregado en la competencia.

Para simular este comportamiento, se debe implementar la subrutina `judge`. Esta debe tomar como argumentos las direcciones del primer y segundo Pokémon a través de los registros `a0` y `a1` respectivamente, y hacer lo siguiente:

1. Si ninguno ha ganado listones, la jueza dará 5 estrellas plateadas al Pokémon que tenga la menor cantidad de estrellas plateadas.
2. Si se han ganado entre 1 y 3 listones entre los dos Pokémon, la jueza envenenará al Pokémon con más listones.
3. Si se han ganado entre 4 y 7 listones entre los dos Pokémon, la jueza quemará al Pokémon con más listones.
4. Si se han ganado más de 7 listones entre los dos Pokémon, la jueza paralizará al Pokémon con más listones.
5. Retorna a donde fue llamada.

En cualquier caso de empate, el premio o castigo aplicará sobre el Pokémon con **mayor ID**. Por otra parte, si la jueza castiga a un Pokémon que ya posee su valor de Estado distinto de Saludable, se queda con el estado de valor mayor. Por ejemplo, si el Pokémon está Envenenado (1) y la jueza lo quema, entonces termina en estado Quemado (3).

Importante a considerar

Sobre el flujo completo

Como pudo dar cuenta, no se le piden subrutinas para actualizar estrellas doradas, listones, Estado de salud de los Pokémon ni para terminar la simulación. Esta lógica **ya está programada** en el archivo base. Su deber es solo completar las subrutinas solicitadas en base a lo pedido.

Convención RISC-V

En esta actividad, no se evaluará que respete la convención RISC-V. No obstante, si no se asegura del respaldo de los registros en las subrutinas a completar, **podría sobreescribir valores que impliquen una implementación incorrecta**. Por lo tanto, trate de respetar la convención revisando el código base entregado y asegurando el respaldo de los registros cuando corresponda.

Testing

Para probar sus subrutinas una a una, se le sugiere seguir este formato:

```
_start:
  la a0, bulbasaur
  la a1, charmander
  # Llamado de prueba: presentation(bulbasaur, charmander)
  jal ra, presentation

  # Impresión de cada pokemon para ver su estado.
  la a0, bulbasaur
  jal ra, _print_pokemon
  la a0, charmander
  jal ra, _print_pokemon
```

Como se observa, se usa el contenido bajo el *label* `_start` para probar la ejecución de la subrutina `presentation` y ver los resultados obtenidos. Basta con que sobreescriba el contenido de `_start` en su archivo base para probar. Usando este formato, puede probar las tres subrutinas solicitadas. También tiene la opción de modificar los datos de los Pokémon en `.data` para evaluar el cómputo correcto de `Puntos belleza`, `Estrellas plata`, `Ponderador rol` o el castigo de la jueza.

Adicionalmente, se le recomienda hacer uso de las siguientes subrutinas a modo de *debugging*:

- `_print_number`: Imprime un número entero almacenado en `a0`. Útil para corrobar que su cálculo de puntos, ponderador y estrellas es correcto.
- `_print_pokemon`: Imprime el estado completo del Pokémon cuya dirección está almacenada en `a0`. Útil para verificar que actualiza correctamente la cantidad de estrellas plateadas y el estado de los Pokémon.