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

IIC2343 - Arquitectura de Computadores (I/2023)

Tarea 2

Respuestas sin desarrollo o justificación no tendrán puntaje.

Publicación: Lunes 10 de Abril 10:00
Entrega: Miércoles 26 de Abril 20:00

Instrucciones

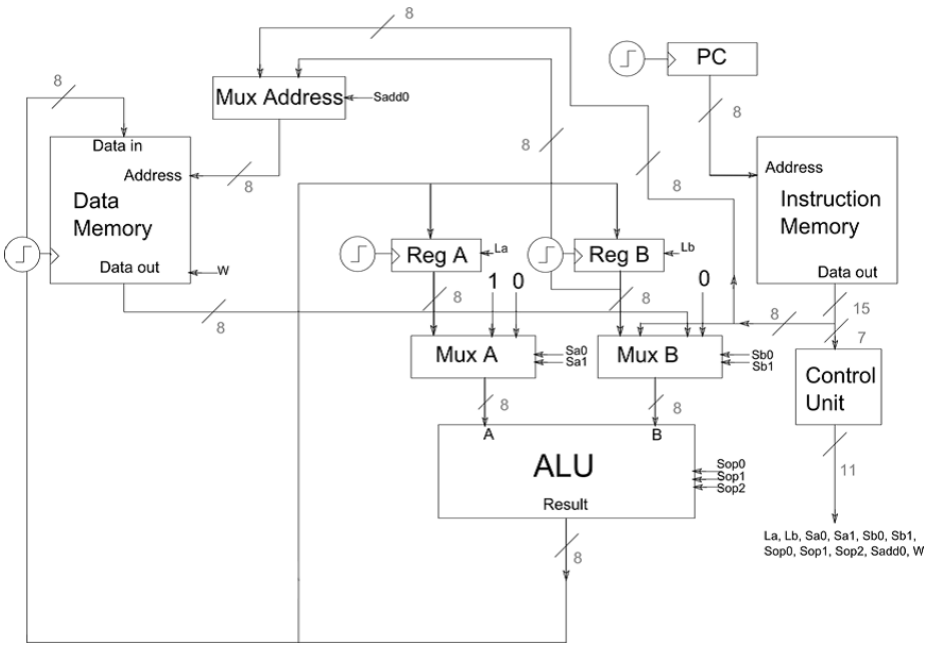
- Lea atentamente el enunciado. Responda cada pregunta en un PDF por separado. Ponga su nombre y número de alumno/a en cada pregunta.
- La entrega de la tarea será a través de **canvas**.
- La tarea debe ser en formato **digital** y no manuscrito, a excepción de los diagramas del computador básico. Estos pueden ser imágenes de su dibujo, pero debe ser **claro** y **ordenado**. 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: Programabilidad (16 ptos.)

Desarrolle las siguientes preguntas a partir del diagrama del computador básico **sin soporte de saltos** incluido a continuación.



(a) (8 ptos.) Modifique el diagrama del computador básico sin soporte de saltos para habilitar las siguientes instrucciones **en una sola iteración**:

- `Op1 B,(A)`; `Op1` $\in$ {`MOV, ADD, SUB, AND, OR, XOR`}
- `Op1 A,(A)`; `Op1` $\in$ {`MOV, ADD, SUB, AND, OR, XOR`}
- `INC A`
- `INC (A)`

En resumen, se le pide dar soporte para **direccionamiento indirecto a través del registro *A***. No es necesario asociar un *opcode* a las instrucciones anteriores, pero sí debe incluir la combinación de señales que ejecuta cada una de ellas. Además, debe indicar si alguna de las combinaciones de señales de instrucciones existentes se ve modificada por su diseño.

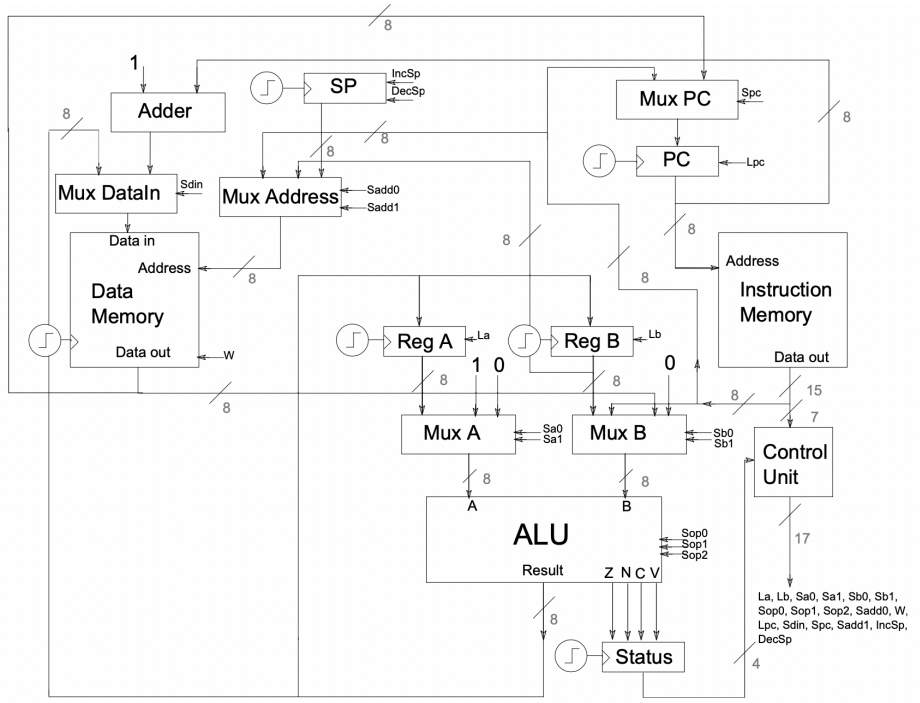
(b) (8 ptos.) Realice una modificación adicional sobre la arquitectura del computador básico para habilitar, al menos, **una nueva instrucción**. **No es válido**:

- Realizar cambios ya conocidos, tales como el soporte de saltos y subrutinas o diagramas explicitados en las diapositivas de clases.
- Hacer uso de la modificación de la pregunta anterior.
- Usar combinaciones de señales no utilizadas. Por ejemplo, cargar simultáneamente una operación en los registros A y B.

Debe agregar al menos un nuevo componente o conexión al diagrama y realizar las modificaciones pertinentes para que la instrucción nueva se pueda ejecutar.

Pregunta 2: Saltos y subrutinas (30 ptos.)

Desarrolle las siguientes preguntas a partir del diagrama del computador básico incluido a continuación.



(a) (6 ptos.) Modifique el diagrama del computador básico para que las instrucciones `RET` y `POP` tomen un solo ciclo. Debe cuidar que sus modificaciones **no alteren el funcionamiento del resto de las instrucciones**.

(b) (12 ptos.) Una máquina de *stack* es un computador que utiliza una memoria de *stack* en vez de registros para almacenar los resultados de las operaciones. Esto significa que cada instrucción aritmética o lógica de dos parámetros, toma los dos valores en el tope del *stack* y luego los elimina, sustituyéndolos por el valor de la operación recién realizada. Para el caso de las operaciones de un parámetro, por ejemplo `NOT`, el computador solo sustituye un valor en el tope del *stack* por el nuevo valor. Además, una máquina de *stack* es capaz de cargar valores literales en el tope del *stack* y también descartarlos. Al igual que en el computador básico, la memoria de *stack* se accede a través del contador `SP`.

A partir de la descripción anterior, elabore el diagrama de la microarquitectura de una máquina de *stack* de 8 bits. Esta máquina debe ser capaz de realizar las mismas operaciones aritméticas y lógicas que el computador básico. Puede incluir en su diagrama un **registro de memoria auxiliar** donde puede almacenar datos temporalmente para ejecutar operaciones de más de un parámetro, así como también el contador `PC` para la lectura de la memoria de instrucciones. Junto a su diagrama, debe incluir el *set* de instrucciones *Assembly* para su máquina, indicando *opcodes*, señales de control y cantidad de ciclos de ejecución. Recuerde que un *opcode* se asocia a un ciclo de ejecución, por lo que puede tener instrucciones asociadas a más de uno.

(c) (8 ptos.) Extienda la implementación de la máquina de *stack* para que incluya saltos condicionales e incondicionales, con la restricción que las *flags* provenientes de la `ALU` solo pueden ser almacenadas dentro del *stack*, no en un registro *Status*. Debe actualizar tanto el diagrama de la microarquitectura como el *set* de instrucciones.

(d) (4 ptos.) Extienda la implementación de la máquina de *stack* para dar soporte a la ejecución de subrutinas. Debe actualizar tanto el diagrama de la microarquitectura como el *set* de instrucciones.

Pregunta 3: Assembly (14 ptos.)

(a) (14 ptos.) Implemente en *Assembly* del computador básico visto en clases, el algoritmo para ordenar arreglos de enteros *min-max sort*. Este algoritmo opera buscando el máximo y el mínimo de un arreglo, a continuación los coloca en los extremos correspondientes, para luego repetir el proceso con el sub-arreglo que no contiene a los extremos. Cualquier cosa que asuma con respecto al ejercicio debe quedar claramente especificada.

Diagramas

Para facilitar la elaboración de sus diagramas, puede descargar el siguiente archivo, cargarlo en Google Drive y abrirlo con `draw.io` para ver y editar los *templates* incluidos. Estos son:

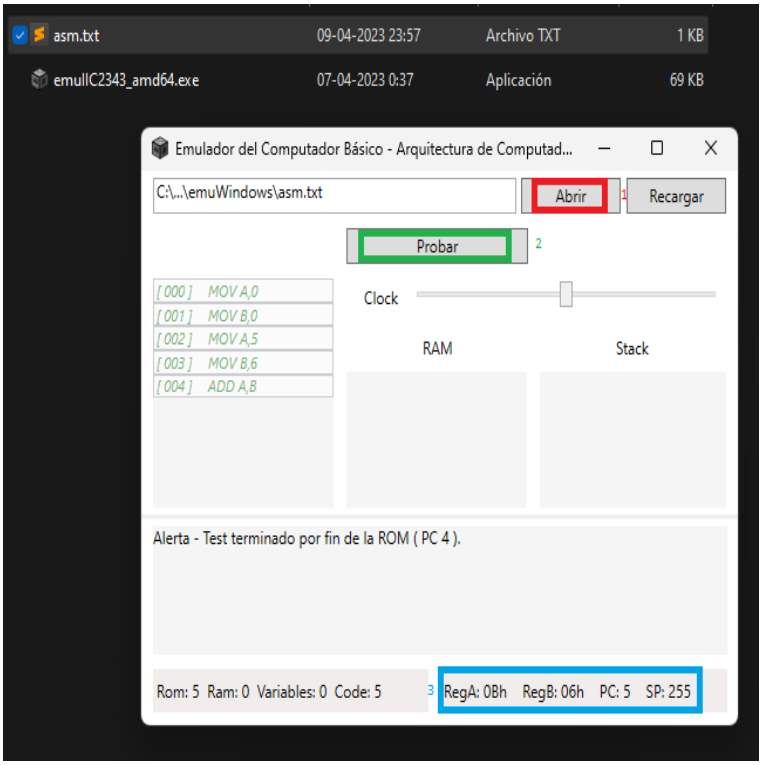
- `CPU sin saltos ni subrutinas`: Diagrama equivalente al incluido en la pregunta 1.
- `CPU completa`: Diagrama equivalente al incluido en la pregunta 2.
- `Elementos máquina stack`: Diagrama que solo tiene los componentes mínimos que requerirá su máquina de *stack*.

Ejecución de Assembly

Para probar la ejecución de su programa en *Assembly*, puede utilizar uno de los siguientes simuladores del computador básico.

Ejecución de Assembly: Windows

Es necesario descomprimir el archivo disponible en Canvas con el nombre `EmuWindows.zip`. Solo debe apretar el ejecutable, dar permisos, luego apretar el botón de `Abrir`, seleccionar un archivo con terminación `.txt` donde este el código y, finalmente, apretar el botón `Probar`. En la esquina inferior derecha se mostrara el valor de los registros, como se muestra en la figura:



Ejecución de Assembly: Linux

Es necesario descomprimir el archivo disponible en Canvas con el nombre `EmuLinux.zip`. Luego, abrir el terminal donde se descomprime el emulador y ejecutar:

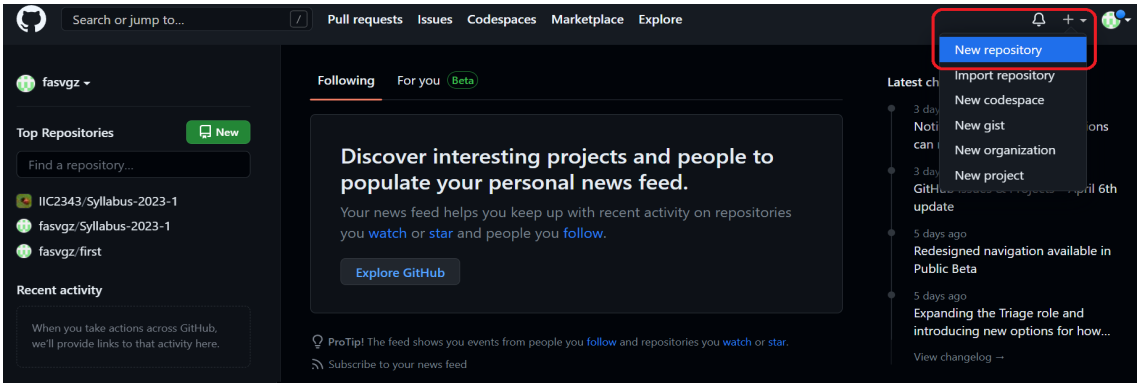
```
iic2343 $> chmod 777 emu-cli-linux
```

Luego, ejecutar lo siguiente:

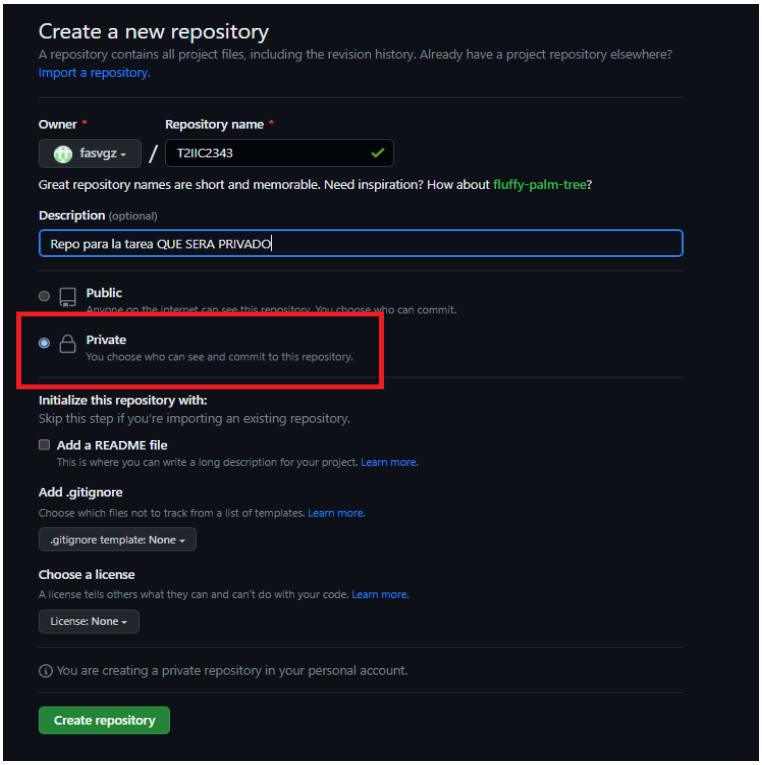
```
iic2343 $> ./emu-cli-linux asm.txt
```

Ejecución de Assembly: Github Action

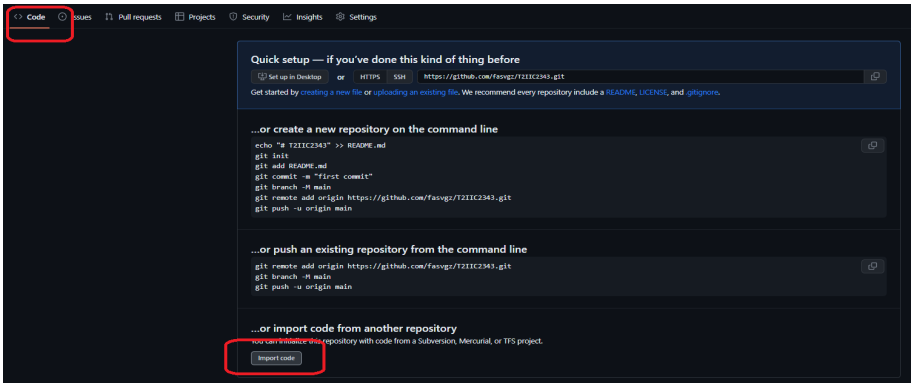
En caso de que ninguna de las opciones anteriores sea compatible con su sistema operativo, deberá **crear un repositorio privado personal** en Github. **Debe ser un repositorio privado, repositorios públicos se considerarán una falta a la integridad académica.**
Para crear un repositorio, apretar en el botón que se muestra en la figura:



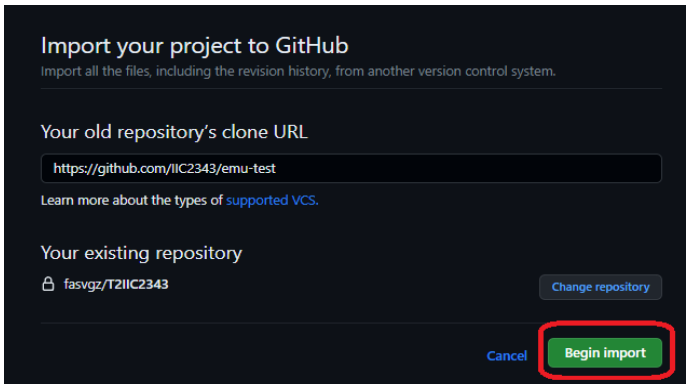
Luego, seleccionar la opción *Private* y crear el repositorio:



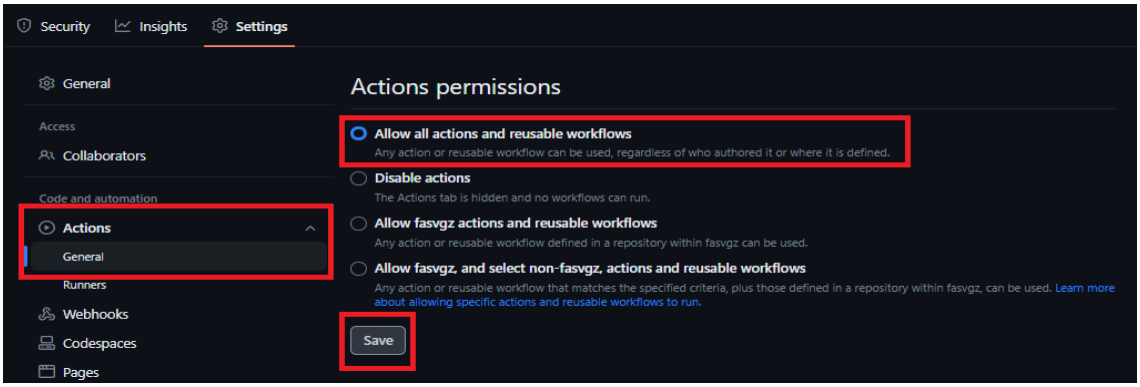
Seleccionar la pestaña *Code* y luego apretar *Import Code*, como muestra la figura a continuación.



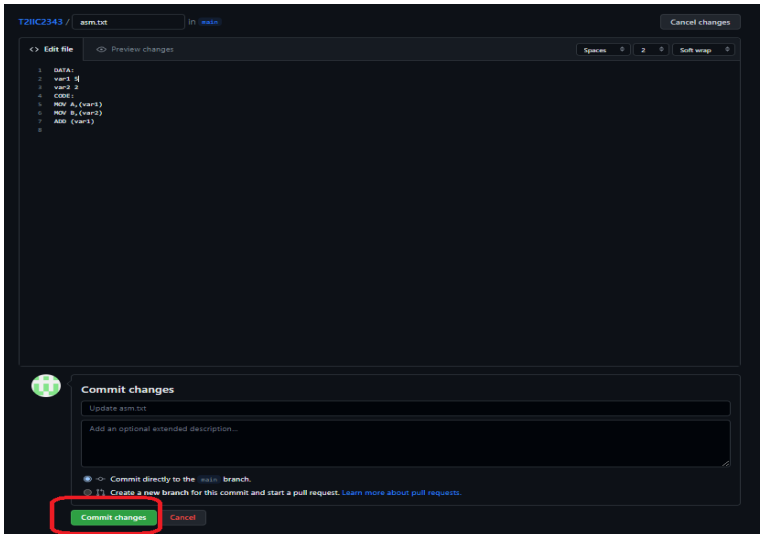
Usar el enlace `https://github.com/IIC2343/emu-test` y apretar en el botón verde.



Luego de crear el repositorio, ir a la pestaña de ajustes o *Settings* y seleccionar *Action*. Ir a *General*, seleccionar la opción mostrada en la imagen y luego apretar *Save*.



Luego, dentro del mismo repositorio, modificar el archivo `asm.txt`, y hacer un *Commit Change*. **Nota:** Aquí en la imagen se muestra con interfaz gráfica pero es posible hacer esto mismo en consola haciendo `git commit` y `git push`.



Finalmente, después de esperar unos minutos y volver a la pestaña de *Actions*, al seleccionar el *commit* podrá ver el resultado de su ejecución en la pestaña *Correr el emulador*, como se muestra en la imagen: