Práctica 2 - Arquitectura de Computadores - Curso 2023/2024

SIMULANDO EL COMPORTAMIENTO DE UN COMPUTADOR EN PYTHON

DEADLINE: Viernes 17 de mayo de 2024 a las 23:59

El propósito de esta última actividad entregable de la asignatura es poner en práctica todo lo aprendido durante el curso de *Arquitectura de Computadores*, así como todos los conocimientos de Python de los estudiantes.

A lo largo de esta práctica se va a simular el comportamiento interno de un computador utilizando Python. La nota máxima en caso de implementar los siguientes tres puntos es de un 8, mientras que será de 9 si se hace en Java y de 10 si se hace en C:

- Deberá existir una clase Procesador, que implementará la funcionalidad relacionada con la ejecución de instrucciones y que mantendrá algunos registros (como mínimo, el contador de programa, el registro de instrucción y varios registros de propósito general).

- Deberá existir también una clase MemoriaPrincipal, que implementará la funcionalidad de la memoria principal de nuestro computador. Trabajaremos con un espacio de direcciones de 16 bits y un tamaño de datos e instrucciones de 32 bits.

- Por último, deberá existir también una clase Computador, que tendrá un procesador y una memoria principal, y que será el encargado de iniciar y terminar la ejecución de programas.

Adicionalmente, el alumno podrá añadir algo extra a la práctica, que le permitirá optar a dos puntos adicionales sobre la nota final de la práctica. Como máximo, podrá sacar un 12, que se le guardará para hacer media con otras notas.

La práctica se podrá hacer por parejas o, excepcionalmente, en tríos en caso de que la mejora propuesta sea muy considerable.

Junto con el código desarrollado en la práctica no será necesario adjuntar una memoria, pero sí un pequeño texto (no más de una cara) justificando brevemente la mejora propuesta.

CLASE PROCESADOR

Hemos comentado que la clase Procesador se encargará de la ejecución de instrucciones. El conjunto de instrucciones a utilizar es el descrito para la arquitectura UNIE-asm y que se encuentra reflejado en la siguiente página. El número de registros de propósito general es

un parámetro configurable. En caso de que una instrucción haga referencia a un registro que no existe, la ejecución debe finalizar e informar del error crítico al usuario.

JUEGO DE INSTRUCCIONES DE UNIE-ASM

#	Modo de Op.	Sintaxis	Descripción
0	0000	`LOAD target, source`	Carga en el registro `target` el contenido de la dirección de memoria indicada por `source`.
1	0001	`LOADI target, source`	Carga en el registro `target` el valor indicado por `source`.
2	0010	`SWAP source1, source2`	Intercambia el contenido de los registros `source1` y `source2`.
3	0011	`STORE target, source`	Almacena en la posición de memoria `target` el contenido del registro `source`.
4	0100	`STOREI target, source`	Almacena en la posición de memoria `target` el valor indicado por `source`.
5	0101	`ADD target, source`	Suma el valor de los registros `source` y `target`, y lo almacena en el registro `target`.
6	0110	`ADDI target, source`	Suma el valor del registro `target` y el valor inmediato `source` y lo almacena en el registro `target`.
7	0111	`SUB target, source`	Resta el valor del registro `source` al valor del registro `target` y lo almacena en el registro `target`.
8	1000	`SUBI target, source`	Resta el valor indicado por `source` al del registro `target` y lo almacena en el registro `target`.
9	1001	`INC target`	Aumenta en una unidad el valor almacenado en el registro `target`.
10	1010	`DEC target`	Decrementa en una unidad el valor almacenado en el registro `target`.
11	1011	`JMP target`	Carga en el CP la instrucción ubicada en la dirección de memoria indicada por `target`.
12	1100	`JMPC target, source1, source2`	Carga en el CP la instrucción ubicada en la dirección de memoria indicada por `target`, en caso de que el valor de los registros `source1` y `source2` sea distinto.

En las instrucciones de la 0 a la 8 (ambas incluídas), destinamos 12 bits para codificar el valor de `target` y 16 bits para codificar el valor de `source`. En las instrucciones de la 9 a la 11, destinamos 16 bits para codificar el `target`, y los 12 bits restantes no los utilizaremos. En el caso de la instrucción 12, destinamos 12 bits para codificar el valor de `target` y 8 bits para codificar cada uno de los `source`. Para codificar el valor de los registros RA, RB, RC y RD utilizaremos los números 0, 1, 2 y 3 en binario.

CLASE MEMORIA PRINCIPAL

La clase para gestionar la memoria principal se puede crear de dos formas:

- Todas las direcciones de memoria se inicializan a cero.
- Se carga el contenido de una memoria en concreto.

Con el propósito de implementar la segunda forma, esta clase tendrá dos métodos, para guardar y cargar la memoria completa desde un fichero de texto. La implementación concreta del formato de escritura en texto debe ser igual a la de la memoria de ejemplo que ha sido adjuntada junto con este enunciado.

CLASE COMPUTADOR

Esta clase se encarga del funcionamiento del computador. Para ello, mediante un bucle infinito repite los siguientes pasos:

- Mira el PC.
- Carga en el IR la instrucción indicada por el PC.
- Pide al procesador que ejecute la instrucción.
- Actualiza el PC (en caso de que no se haya ejecutado una instrucción de salto).

La única forma de detener el bucle anterior es si llega una instrucción con código de operación “1111”, que debe abortar inmediatamente la ejecución de instrucciones y el programa finalizará.

De forma adicional, al acabar, el programa deberá imprimir el número de instrucciones ejecutadas, así como el número de operaciones de lectura y escritura a memoria.

Queda terminantemente prohibida la copia y el uso de herramientas de IA generativa para la realización de esta práctica. En caso de detectarse, conllevará la suspensión inmediata de la evaluación continua para los estudiantes implicados.

Únicamente uno de los miembros de la pareja/trio tendrá que subir a BlackBoard el código desarrollado, pero es **IMPRESCINDIBLE** incluir un fichero de texto con los nombres de los alumnos involucrados. Todo el código y ficheros a entregar durante la práctica tendrá que ser entregado en **UN FICHERO COMPRIMIDO .ZIP**.