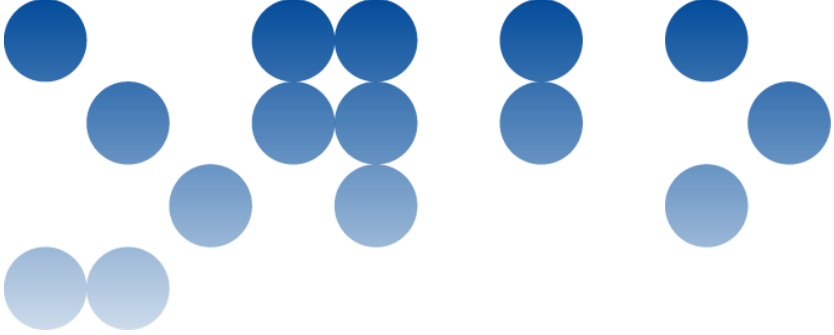

PRÁCTICA 2ª parte: 2048

Estructura de Computadores

Grado en Ingeniería Informática

feb20-jun20

Estudios de Informática, Multimedia y Telecomunicación

EIMT.UOC.EDU

Universitat Oberta de Catalunya

Presentación

La práctica que se describe a continuación consiste en la programación en lenguaje ensamblador x86_64 de un conjunto de subrutinas, que se tienen que poder llamar desde un programa en C. El objetivo es hacer un juego del 2048.

La PRÁCTICA es una **actividad evaluable individual**, por lo tanto no se pueden hacer comentarios muy amplios en el foro de la asignatura. Se puede hacer una consulta sobre un error que tengáis en la hora de ensamblar el programa o de algún detalle concreto, pero no se puede poner el código de una subrutina o bucles enteros.

Competencias

Las competencias específicas que persigue la PRÁCTICA son:

- [13] Capacidad para identificar los elementos de la estructura y los principios de funcionamiento de un ordenador.
- [14] Capacidad para analizar la arquitectura y organización de los sistemas y aplicaciones informáticos en red.
- [15] Conocer las tecnologías de comunicaciones actuales y emergentes y saberlas aplicar convenientemente para diseñar y desarrollar soluciones basadas en sistemas y tecnologías de la información.

Objetivos

Introducir al estudiante en la programación de bajo nivel de un computador, utilizando el lenguaje ensamblador de la arquitectura Intel x86-64 y el lenguaje C.

Recursos

Podéis consultar los recursos del aula pero no podéis hacer uso intensivo del foro.

El material básico que podéis consultar es:

- Módulo 6: Programación en ensamblador (x86_64)
- Documento “Entorno de trabajo”

La Práctica: 2048

La práctica consiste en implementar el juego del “2048” que consiste en ir moviendo los valores que se encuentran en un tablero de 4 x 4 de forma que cuando dos valores iguales entran en contacto se suman, hasta llegar a que en una casilla el valor sea igual a 2048. Se pueden desplazar los valores del tablero a la izquierda, derecha, arriba o abajo, al hacer un desplazamiento se arrastran todos los valores del tablero en la dirección seleccionada, eliminando los espacios en blanco que haya entre las casillas. Si al hacer el desplazamiento quedan dos casillas consecutivas con el mismo valor, se suman los valores, dejando una casilla con el valor sumado y la otra casilla en blanco.

Al finalizar un movimiento el programa genera un número 2 o un 4 que se añade al tablero en una casilla vacía seleccionada aleatoriamente.

SEGUNDA PARTE OPCIONAL: En la segunda parte se tienen que implementar las funcionalidades adicionales necesarias para completar todas las funcionalidades del juego del 2048.

Además hay que trabajar con el paso de parámetros entre las diferentes subrutinas, modificando la implementación hecha en la primera parte.

Os proporcionamos también dos ficheros para esta segunda parte: 2Kp2c-es.c y 2Kp2-es.asm. De forma parecida a la primera parte, el código C no lo tenéis que modificar, solo tenéis que implementar en ensamblador nuevas funcionalidades y habrá que modificar las subrutinas que habréis hecho en la primera parte para trabajar el paso de parámetros entre las subrutinas de ensamblador y también con las funciones de C.

Formato y fechas de entrega

Esta segunda parte se puede entregar antes de las **23:59 del 15 de mayo de 2020**. Si en la primera entrega se superó la primera parte se puede llegar a una puntuación de A en las prácticas.

En cambio, si no se hizo la primera entrega o si no fue satisfactoria se puede entregar la primera parte juntamente con esta segunda parte para obtener una calificación máxima de B. Si solamente se hace la primera parte, se podrá obtener una calificación máxima de C+.

Este esquema de entregas y calificación se puede resumir en la siguiente tabla.

Primera Entrega 10-04-2020	Primera parte Superada	Primera parte NO Superada NO Presentado	Primera parte Superada	Primera parte NO Superada NO Presentado	Primera parte NO Superada NO Presentado
Segunda Entrega 15-05-2020	Segunda parte NO Superada NO Presentado	Primera parte Superada	Segunda parte Superada	Primera parte Superada Segunda parte Superada	Primera parte NO Superada NO Presentado
Nota Final Práctica	**B**	**C+**	**A**	**B**	**D/N**

Los alumnos que no superen la PRÁCTICA tendrán un suspenso (calificación 0-2) o N si no se ha presentado nada, en la nota final de prácticas y con esta nota no se puede aprobar la asignatura, por este motivo la PRÁCTICA es obligatoria.

La entrega se tiene que hacer a través de l 'aplicación **Entrega y registro de EC** del aula. Se tiene que entregar solo un fichero con el código ensamblador muy comentado.

Es muy importante que el nombre del fichero tenga vuestros apellidos y nombre con el formato:

apellido1_apellido2_nombre_2Kp2.asm

Fecha límite Segunda Entrega:

Viernes, 15 de mayo de 2020 a las 23:59:59

Criterios de valoración

La práctica tiene que funcionar completamente para considerarse superada, y hay que implementar todas las funcionalidad pedidas en el enunciado, **la opción del menú correspondiente al juego completo en ensamblador tiene que funcionar correctamente.**

Las otras opciones del menú son solo para comprobar individualmente cada una de las subrutinas que se tienen que implementar.

No es suficiente para aprobar la práctica que las opciones correspondientes a las subrutinas individuales funcionen.

Dado que se trata de hacer un programa, sería bueno recordar que las dos virtudes a exigir, por este orden son:

- Eficacia: que haga el que se ha pedido y tal como se ha pedido, es decir , que todo funcione correctamente según las especificaciones dadas. Si las subrutinas no tienen la funcionalidad pedida, aunque la práctica funcione correctamente, se podrá considerar suspendida.

- Eficiencia: que lo haga de la mejor forma. Evitar código redundante (que no haga nada) y demasiado código repetido (que el código sea compacte pero claro). Usar modos de direccionamiento adecuados: los que hagan falta. Evitar el uso excesivo de variables y usar registros para almacenar valores temporales.

IMPORTANTE: Para acceder a los vectores en ensamblador se ha de utilizar direccionamiento relativo o direccionamiento indexado: [m+rsi], [rbx+rdi]. No se pueden utilizar índices con valores fijos para acceder a los vectores.

Ejemplo del que **NO** se puede hacer:

```
mov WORD [m+0], 0
mov WORD [m+4], 0
mov WORD [m+8], 0
...
```

Y repetir este código muchas veces.

Otro aspecto importante es la documentación del código: que clarifique, dé orden y estructura, que ayude a entenderlo mejor. No se tiene que explicar qué hace la instrucción (se da por supuesto que quién la lee sabe ensamblador) si no que se tiene que explicar porque se usa una instrucción o grupo de instrucciones (para hacer qué tarea de más alto nivel, relacionada con el problema que queremos resolver).

Implementación

Como ya hemos dicho, la práctica consta de una parte de código en C que os damos hecho y NO **PODÉIS MODIFICAR** y un programa en ensamblador que contiene algunas subrutinas ya implementadas y las subrutinas que tenéis que implementar. Cada subrutina de ensamblador tiene una cabecera que explica que hace esta subrutina y como se tiene que implementar, indicando las variables, funciones de C y subrutinas de ensamblador que se tienen que llamar.

No podéis añadir otras variables o subrutinas.

Para ayudaros en el desarrollo de la práctica, en el fichero de código C encontraréis implementado en este lenguaje las subrutinas que tenéis que hacer en ensamblador para que os sirvan de guía durante la codificación.

En el código C se hacen llamadas a las subrutinas de ensamblador que tenéis que implementar, pero también encontraréis comentadas las llamadas a las funciones de C equivalentes. Si queréis probar las funcionalidades hechas en C lo podéis hacer sacando el comentario de la llamada en C y poniéndolo en la llamada a la subrutina de ensamblador.

Por ejemplo en la opción 1 del menú hecho en C hay el código siguiente:

```
//=======================================================
showNumberP2(18, 26, score);
//showNumberP2_C(18, 26, score);
//=======================================================
```

El código hace un llamada a la subrutina de ensamblador showNumberP2(), podemos cambiar el comentario y llamar a la función de C.

```
//=======================================================
//showNumberP2(18, 26, score);
showNumberP2_C(18, 26, score);
//=======================================================
```

Recordad volver a dejar el código como estaba para probar vuestras subrutinas.

Subrutinas que ya están implementadas y que no tenéis que modificar para la Segunda Parte:

```
gotoxyP2
printchP2
getchP2
readKeyP2
printMessageP2
insertTileP2
playP2
```

Subrutinas que hay que implementar en ensamblador para la Segunda Parte:

```
showNumberP2
updateBoardP2
rotateMatrixRP2
copyMatrixP2
shiftNumbersRP2
addPairsRP2
checkEndP2
```