

# PRÁCTICA

Estructura de Computadores  
Grado en Ingeniería Informática  
2014-02

Estudios de Informática, Multimedia y Telecomunicación

Cartagena99

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70

Universitat Oberta  
de Catalunya



## Presentación

La práctica que se describe a continuación es una práctica obligatoria. 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.

Durante el curso os proporcionaremos diversos ejemplos de programación en ensamblador. Entender estos ejemplos os puede ser de gran ayuda para hacer la práctica.

La **PRÁCTICA** es una **actividad evaluable individual**, podéis plantear dudas en el foro, pero **NO** podéis poner el código. Utilizar código publicado en el foro se puede considerar una copia. Para ayudaros entre vosotros en los temas de la PRÁCTICA, tenéis los ejemplos de ensamblador que os propondremos y que podéis discutir y exponer en el foro.

## 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 y el lenguaje C.

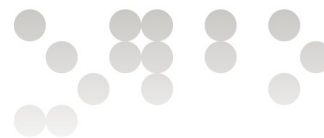
## Enunciado

El enunciado de la práctica se dividirá en tres partes: Práctica 1, Práctica 2 y Práctica 3. En cada una de las partes habrá que desarrollar las subrutinas en lenguaje ensamblador que se pidan. El código habrá que escribirlo a partir de

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

---

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70**



- Módulo 6: Programación en ensamblador (x86\_64)
- Documento “Entorno de trabajo”

## Criterios de valoración

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

- Eficacia: que haga lo que se ha pedido, es decir, que funcione correctamente según las especificaciones dadas. Si las subrutinas no tienen la funcionalidad especificada, 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 compacto pero claro). Utilizar modos de direccionamiento adecuados: los que sean necesarios. Evitar el uso excesivo de variables y utilizar registros para almacenar valores temporales.

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

La práctica es obligatoria y habrá diferentes fechas correspondientes a las tres partes que la componen. Cada parte tendrá un peso en la nota final de prácticas:

- Práctica 1: 30%
- Práctica 2: 30%
- Práctica 3: 40%

En función de las entregas que se hagan y la calificación obtenida en cada parte se obtendrá una determinada nota.

Las calificaciones que se podrán obtener en cada parte son las siguientes:

- A: funciona todo correctamente.
- B: alguna de las subrutinas no funciona correctamente, pero todo el código está bien planteado.
- C+: alguna de las subrutinas no funciona correctamente y el código está mal planteado, o no se ha implementado alguna subrutina.

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

---

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70**

Cartagena99



Para hacer el cálculo se aplicará la siguiente correspondencia entre las notas cualitativas y las notas cuantitativas:

A	= 9 – 10
B	= 7 – 8
C+	= 5 – 6
D	= 1 – 2
N	= 0

Si no se presenta ninguna de las 3 partes la nota final de la práctica será No Presentado (N).

No es obligatorio presentar las 3 partes para superar la práctica, no es necesario tener una nota mínima para calcular la nota final de prácticas.

Los alumnos que no superen la PRÁCTICA obtendrán un suspenso o si no se han presentado, N, en la nota final de prácticas y con estas nota no se puede aprobar la asignatura, por este motivo la PRÁCTICA es obligatoria.

### Formato y fecha de entrega

La entrega se tiene que realizar a través de la aplicación Entrega y registro de EC del aula. Hay que entregar sólo un archivo, el archivo con el código ensamblador y es necesario que el código esté bien comentado.

Es muy importante que el nombre del archivo tenga vuestro nombre y apellidos, por ejemplo:

apellido1\_apellido2\_nombre\_prac1.asm

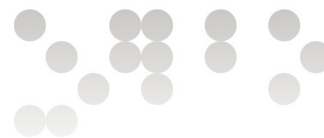
### Fecha límite Práctica 1:

**Viernes, 04 de abril de 2014 a las 23:59:59**

CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70

---

ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70



## Enunciado

La práctica consiste en la implementación de una calculadora, que tiene que tener las funcionalidades siguientes:

- Tiene que poder ejecutar diferentes operaciones: suma, “y” lógica, multiplicación, potencia, y negación.
- Se tienen que mostrar los operandos y resultados en diferentes bases: binaria: 2, octal: 8, decimal: 10 y hexadecimal: 16.
- Se podrá trabajar con operandos de diferentes tamaños.
- Los operandos y resultados se podrán leer y almacenar en memoria utilizando diferentes estructuras de datos (variables y vectores).
- Tiene que ser capaz de trabajar con una pila implementada sobre un vector.

En cada una de las tres partes que componen la práctica se irán implementado diferentes aspectos de la práctica. A continuación se enuncia el correspondiente a la primera parte (Práctica 1), más adelante durante el curso, se proporcionará el enunciado correspondiente a la segunda y tercera parte (Práctica 2 y Práctica 3).

### Implementación Práctica 1

El código del programa se dividirá en dos partes, una parte en lenguaje C y una parte en lenguaje ensamblador.

Utilizaremos C para:

- Definir variables utilizadas en el código C y en el código ensamblador.
- Escribir el programa principal y algunas funciones.
- Llamar a las funciones de C y d ensamblador que realizan las diferentes operaciones del programa.
- Leer valores por teclado y mostrar los resultados por pantalla.

Utilizaremos ensamblador para:

- Escribir subrutinas que llamaremos desde C o desde otras subrutinas de ensamblador.
- Llamar a determinadas funciones de C.

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

---

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70**

Cartagena99



(fichero `prac1.asm`) siguiendo las indicaciones de este documento y las especificaciones que encontraréis en la cabecera de cada una de las subrutinas. Las subrutinas utilizadas para hacer las llamadas a funciones de C también os las damos hechas. Se pueden añadir más variables y más subrutinas en el código ensamblador, pero hay que especificar qué funcionalidad tienen y explicar claramente porque se han añadido, y en ningún caso para cambiar o sustituir la funcionalidad de las que ya están definidas.

Sí podéis modificar, en el archivo `prac1.c`, el valor que se asigna a las variables en la función `main()` para hacer diferentes pruebas de ejecución del programa.

También os damos el fichero `prac1Completa.c` con todas las funcionalidades que se piden implementadas en C y siguiendo la misma estructura que la práctica, para que las podáis utilizar de guía para implementar las subrutinas de ensamblador que tienen una funcionalidad equivalente.

La función principal corresponde a la función `main()` y realiza las tareas siguientes:

- Muestra la interfaz de la calculadora.
- Inicializa las variables que guardan los operandos y los resultados.
- Muestra el valor de las variables.
- Ejecuta un bucle del cual se sale sólo en el caso de pulsar la tecla 'q', dentro del bucle se hace lo siguiente:
  - Se lee una tecla y se llama a la función correspondiente a la tecla leída.
  - Se actualiza la información en pantalla: valor de las variables, bits de resultado, base utilizada y tamaño de los operandos.

En esta primera parte (Práctica 1): trabajaremos sólo con variables de tamaño 1 byte (A1, B1, R1) y tamaño 4 bytes (A4, B4, R4).

Las variables A1, B1, A4 y B4 almacenarán los operandos fuente para las diferentes operaciones de la calculadora, las variables R1 y R4 almacenarán el resultado de cada operación.

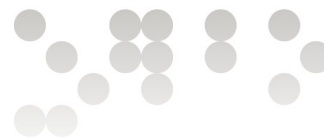
Las teclas válidas serán:

- Operaciones: 'a': suma, 'b': AND, 'c': multiplicación, 'd': potencia, 'e':

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

---

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70**



### Subrutinas que hay que implementar en ensamblador:

Las subrutinas que hay que implementar en ensamblador para la Práctica 1, se describen a continuación, consultad en el archivo `prac1.asm` los detalles de como se tienen que implementar. En algunas subrutinas os daremos parte del código hecho, en otras tendréis que hacer todo el código.

Os recomendamos que dejéis las subrutinas `showChar1` y `showInt1` para el final porque son las más difíciles de implementar, mientras utilizad las llamadas a las funciones de C, `showChar1_C` y `showInt1_C`.

#### showChar1

Mostrar un valor de tipo `char`, `BYTE` (`size=1`), recibido como parámetro según la base (`base`) con la que estemos trabajando a partir de la posición `[row,col]` de pantalla también recibida como parámetro.

#### showInt1

Mostrar un valor de tipo `int`, `DWORD` (`size=4`), recibido como parámetro según la base (`base`) con la que estemos trabajando a partir de la posición `[row,col]` de pantalla también recibida como parámetro.

#### executeOperation1

Llamar a la función para hacer la operación seleccionada (`opADD1`, `opAND1`, `opMUL1`, `opPOW1`, `opNEG1`). La opción seleccionada se recibe como parámetro.

#### opADD1

Si operamos con variables de tipo `BYTE` (`size=1`), sumar las variables `A1` y `B1` y guardar el resultado en la variable `R1`. Si operamos con variables de tipo `DWORD` (`size=4`) sumar las variables `A4` y `B4` y guardar el resultado en la variable `R4`.

#### opAND1

Si operamos con variables de tipo `BYTE` (`size=1`), hacer la operación `AND` de las variables `A1` y `B1` y guardar el resultado en la variable `R1`. Si operamos

**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

---

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70**

Cartagena99



tipo DWORD (size=4), multiplicar las variables A4 y B4 y guardar el resultado en la variable R4.

### **opPOW1**

Si operamos con variables de tipo BYTE (size=1), hacer la potencia de las variables A1 y B1 ( $A1 \wedge B1$ ) y guardar el resultado en la variable R1. Si operamos con variables de tipo DWORD (size=4), hacer la potencia de las variables A4 y B4 ( $A4 \wedge B4$ ) y guardar el resultado en la variable R4.

### **opNEG1**

Si operamos con variables de tipo BYTE (size=1), hacer la operación NEG de la variable A1 y guardar el resultado en la variable R1. Si operamos con variables de tipo DWORD (size=4), hacer la operación NEG de la variable A4 y guardar el resultado en la variable R4.



**CLASES PARTICULARES, TUTORÍAS TÉCNICAS ONLINE  
LLAMA O ENVÍA WHATSAPP: 689 45 44 70**

---

**ONLINE PRIVATE LESSONS FOR SCIENCE STUDENTS  
CALL OR WHATSAPP:689 45 44 70**