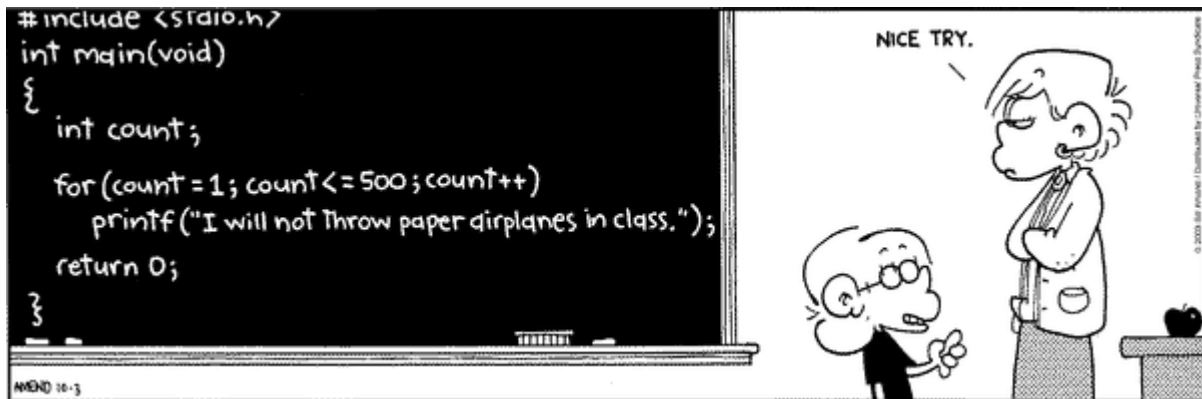


Fundamentos de programación

Ejercicios resueltos



Curso 2007/08

Ejercicios de Fundamentos de Programación - revisión 2007/08 v1.0

Copyright © Alejandro Castán Salinas

Se otorga el permiso para copiar, distribuir y/o modificar este documento bajo los términos de la licencia de documentación libre GNU, versión 1.2 o cualquier otra versión posterior publicada por la *Free Software Foundation*.

Puedes consultar dicha licencia en <http://www.gnu.org/copyleft/fdl.html>.

El contenido de este documento puede cambiar debido a ampliaciones y correcciones enviadas por los lectores. Encontrarás siempre la última versión del documento en <http://www.xtec.net/~acastan/textos/>.

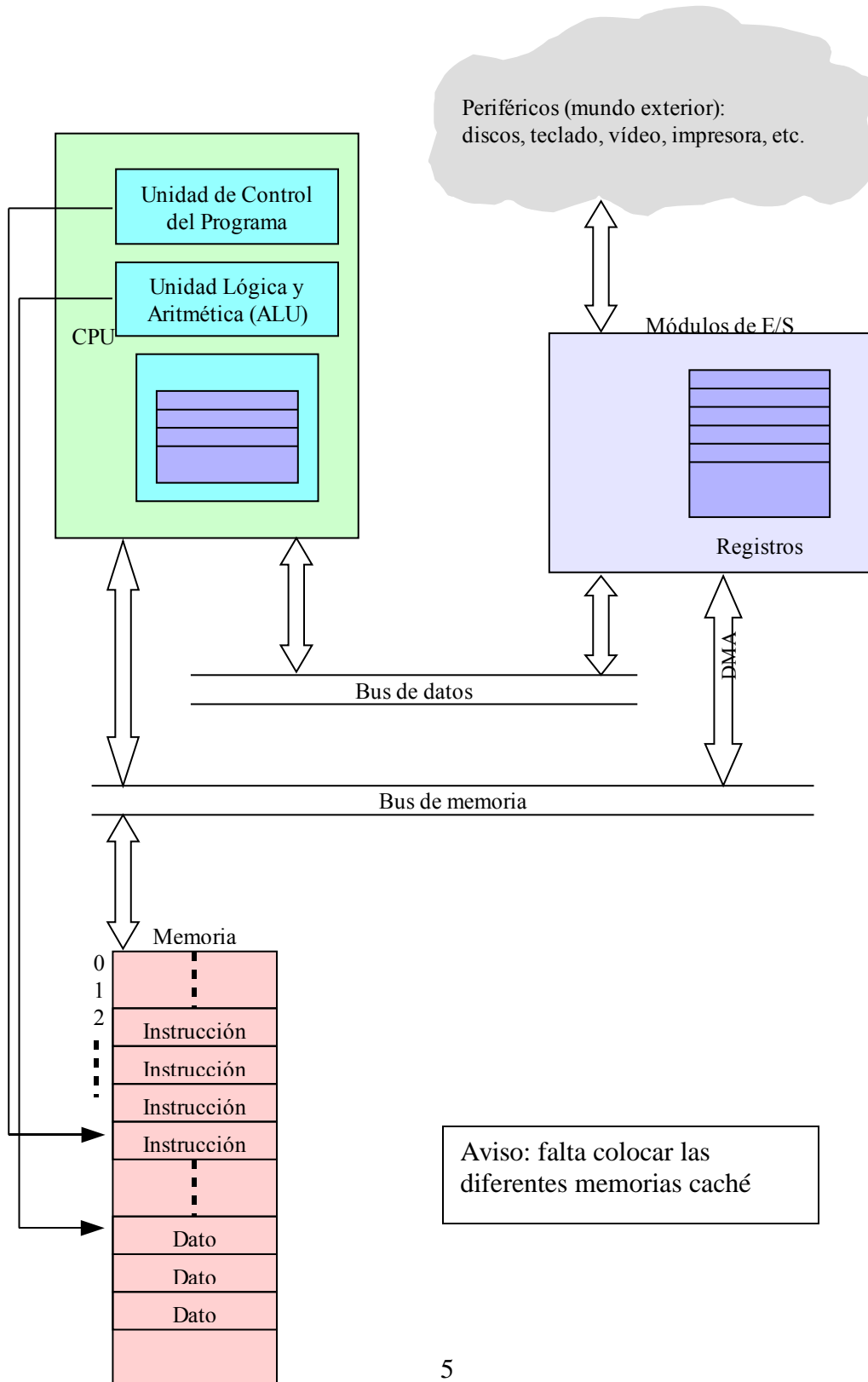
Índice de contenido

Practica 1: Arquitectura del computador.....	5
Practica 2: Codificación de la información.....	7
Practica 3: Lenguajes de programación, compiladores e intérpretes, y entornos de desarrollo.....	10

Practica 1: Arquitectura del computador

1.1 <http://es.wikipedia.org/zz>

1.2 Esquema de la arquitectura de un ordenador.



1.3 Imagina que eres vendedor/a de una tienda de informática. Aconseja a los siguientes tres clientes que llegan a la vuestra tienda sobre cual es el ordenador que se ajusta a sus necesidades, es decir, qué componentes y accesorios necesitaran, como deben ser éstos y el porqué. Como lista de componentes tenemos: microprocesador, memoria RAM, disco duro, módem, tarjeta de red, tarjeta de vídeo, tarjeta de sonido, caja y s.a.i.

- Cliente 1: Javi es un chico que ya tiene un ordenador. Lo utiliza sobretodo para jugar: es un apasionado de los videojuegos. El problema es que los últimos juegos que ha comprado ya van un poco lentos en su ordenador. ¿Qué componentes de su ordenador cambiaríais para mejorar el rendimiento en los juegos? Además Javi también utiliza el ordenador como a asistente a la hora de componer música.
- Cliente 2: Julia es una ingeniera. En el trabajo necesita un ordenador para hacer simulaciones de dinámicas de fluidos (cálculos muy costosos y muchísimos datos). El resto de ordenadores del trabajo accederán constantemente a este ordenador mediante su red local para consultar los resultados de les simulaciones.
- Cliente 3: Juan trabaja en una oficina. Quiere un ordenador para poder escribir en casa informes y llevárselo del trabajo a casa.
- ¿Y tú? ¿Cómo es el ordenador que necesitas?

Componente	Cliente 1	Cliente 2	Cliente 3
microprocesador	↑ rápido (cálculos y atender peticiones)	↑ rápido (cálculos)	-
memoria RAM	↑ rápida (cálculos) ↑ grande (datos)	↑ rápida (cálculos) ↑ grande (datos texturas)	-
disco duro	↑ grande (datos) ↑ rápido (atender peticiones)	-	-
tarjeta de red	↑ buena (atender peticiones)		
tarjeta de vídeo	-	↑ rápida (cálculos)	-
tarjeta de sonido		↑ (calidad sonido/pistes)	
altavoces		↑ (calidad sonido)	
caja	-	-	portátil
s.a.i.	↑ (cortes de luz)		

Practica 2: Codificación de la información

3.1 Completa la siguiente tabla:

Binario	Hexadecimal	Decimal
10000000.11	80.C	128.75
10011101.11001	9D.C8	157.78125
10111110.10100111	BE.A7	190.65234375

3.2 Representa los números 223 y -223 en binario en los diferentes códigos para la representación de enteros con signo:

		Binario
Binario natural	223	11011111
	-223	No representable
Signo y magnitud	223	011011111
	-223	111011111
Complemento a 2	223	011011111
	-223	100100001
Exceso (256)	223	111011111
	-223	000100001

3.3 Calcula el valor decimal del número binario 10100111 representado en los diferentes códigos.

		Decimal
Binario natural	10100111 ₍₂₎	167
Signo y magnitud	10100111 ₍₂₎	-39
Complemento a 2	10100111 ₍₂₎	-89
Exceso (128)	10100111 ₍₂₎	39

3.4 Realiza las siguientes sumas en 8 bits y complemento a 2, dando el valor del resultado en decimal.

$$26 + 24 = 50$$

$$-15 + 82 = 67$$

$$84 + 69 = 103$$

$$-46 + (-10) = -56$$

$$\begin{array}{r}
 + 26 \\
 + 24 \\
 + 50 \\
 \hline
 \end{array}
 \begin{array}{|c|c|c|c|c|c|c|c|}
 \hline
 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
 \hline
 + & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
 \hline
 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
 \hline
 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\
 \hline
 \end{array}$$

$$\begin{array}{r}
 - 15 \\
 + 82 \\
 \hline
 \end{array}
 \begin{array}{|c|c|c|c|c|c|c|c|}
 \hline
 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
 \hline
 + & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\
 \hline
 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
 \hline
 \end{array}$$

$$+ 67 \quad \boxed{01000011}$$

$$+ 84 \quad \boxed{01010100}$$

$$+ 69 \quad + \boxed{01000101}$$

$$\hline \boxed{10011001}$$

$$- 103 \quad \boxed{01100111}$$

OVERFLOW !

$$- 46 \quad \boxed{11010010}$$

$$- 10 \quad + \boxed{11110110}$$

$$\hline \boxed{11100100}$$

$$- 56 \quad \boxed{00111000}$$

3.5 Representa 123.12 según la norma IEEE 754 de simple precisión. Utilizar truncado o redondeo si fuera necesario.

$$\boxed{01000010111101100011110101110001}$$

- $123.12_{(10)} = 111011.000111101011100001_{(2)} = 1.111011000111101011100001_{(2)} \times 2^6$
- 123.12 es positivo, y por lo tanto el signo se representa con 0.
- En exceso 127, el exponente 6 se representa como $6+127 = 133_{(10)} = 10000101_{(2)}$
- La mantisa es 111011000111101011100001, que tiene 24 bits cuando debería tener 23 bits. Truncando la mantisa a 23 bits obtenemos la nueva mantisa 11101100011110101110000 y redondeando a 23 bits obtenemos la nueva mantisa 11101100011110101110001. Trabajaremos con el valor redondeado, ya que con él obtenemos una precisión más grande.

3.6 En un microprocesador de 16 bits deseamos ejecutar un programa que ocupa 20000 posiciones de memoria y que además necesita espacio para: 10000 números enteros (16 bits), 5000 números reales de simple precisión (32 bits), 2000 números reales de doble precisión (64 bits) y 1000 alarmas binarias (1 bit). Indica la capacidad mínima de la memoria necesaria en posiciones de memoria, bits, bytes y kilobytes.

Suponiendo que una posición de memoria tiene capacidad para almacenar 16 bits.

- 20000 posiciones de memoria ocupa el código del programa.
- $10000 \text{ números enteros} \times 16 \text{ bits/entero} \times 1 \text{ posición}/16 \text{ bits} = 10000 \text{ posiciones}$
- $5000 \text{ números reales de simple precisión} \times 32 \text{ bits/real} \times 1 \text{ posición}/16 \text{ bits} = 10000 \text{ posiciones}$
- $2000 \text{ números reales de doble precisión} \times 64 \text{ bits/real} \times 1 \text{ posición}/16 \text{ bits} = 8000 \text{ posiciones}$

- 1000 alarmes binarias \times 1 bits/alarma \times 1 posición/16 bits = 62.5 = 63 posiciones

La capacidad mínima de la memoria necesaria es 20000 + 10000 + 10000 + 8000 + 63 = **48063 posiciones de memoria.**

$$48063 \text{ posiciones de memoria} \times 16 \text{ bits/posición} = 769008 \text{ bits}$$

$$769008 \text{ bits} \times 1 \text{ byte}/8 \text{ bits} = 96126 \text{ bytes}$$

$$105654 \text{ bytes} \times 1 \text{ Kbyte}/1024 \text{ bytes} = 93.87 \text{ Kbytes}$$

3.7 Queremos enviar por una línea telefónica una información en paquetes de 8+1 bits. Aplica un código de paridad par y otro impar para enviar la cadena de caracteres Alumno (sin las comillas) en código ASCII de 8 bits. Discute la posibilidad de detectar transmisiones erróneas.

Carácter	Código ASCII	Binario 8 bits	Bit paridad par	Bit paridad impar
A	065 ₍₁₀₎	01000001 ₍₂₎	0	1
l	108 ₍₁₀₎	01101100 ₍₂₎	0	1
u	117 ₍₁₀₎	01110101 ₍₂₎	1	0
m	109 ₍₁₀₎	01101101 ₍₂₎	1	0
n	110 ₍₁₀₎	01101110 ₍₂₎	1	0
o	111 ₍₁₀₎	01101111 ₍₂₎	0	1

Con bit de paridad par la cadena enviada es:

01000001	0	01101100	0	01110101	1	01101101	1	01101110	1	01101111	0
----------	---	----------	---	----------	---	----------	---	----------	---	----------	---

Con bit de paridad impar la cadena enviada es:

01000001	1	01101100	1	01110101	0	01101101	0	01101110	0	01101111	1
----------	---	----------	---	----------	---	----------	---	----------	---	----------	---

Practica 3: Lenguajes de programación, compiladores e intérpretes, y entornos de desarrollo

3.1 Suma de números enteros:

El resultado de las sumas debería ser:

$$\begin{array}{r r r r r} 200 & + & 350 & = & \mathbf{550} \\ 250 & + & -300 & = & \mathbf{-50} \\ 20000 & + & 30000 & = & \mathbf{-15536} \\ 40000 & + & -10 & = & \mathbf{-25546} \\ 2000000000 & + & 2000000000 & = & \mathbf{-294967296} \end{array}$$

Lo que ha pasado en las tres últimas sumas es que se ha producido un error de desbordamiento (overflow). Los números de tipo entero en C se codifican en binario siguiendo un sistema de representación de complemento a dos y 16 bits (o 32 bits, dependiendo de la arquitectura del ordenador). Esto hace que el rango de representación de los enteros vaya del -32768 al 32767 (o -2147483648 a 2147483647 en 32 bits). Para arquitecturas de 16 bits los resultados correctos de la tercera suma, 50000 , y de la cuarta suma, 39990 , están por encima del número entero más grande que podamos representar: 32767 . Para arquitecturas de 32 bits el resultado correcto de la quinta suma, 4000000000 , está por encima del número entero más grande que podamos representar: 2147483647 .

Si queremos un rango de representación más grande a la hora de trabajar con números enteros, utilizaremos el tipo `long`, que utiliza el doble de bits que el tipo `int`.

3.2 Suma de números reales:

El resultado de las sumas debería ser:

$$\begin{array}{r r r r r} 250.30 & + & 300.50 & = & \mathbf{550.799988} \\ 1E12 & + & 1.0 & = & \mathbf{999999995904.000000} \\ 3333333333333333.333 & + & 6.667 & = & \mathbf{333333317812224.000000} \\ 987654321004 & + & -987654321002 & = & \mathbf{0.000000} \\ 123456789876543 & + & 0 & = & \mathbf{123456788103168.000000} \end{array}$$

El tipo `float` de C (estándar IEEE754 de 32 bits) tiene un rango de representación de $\pm 1.4e-45$ a $\pm 3.4e38$, con 11-12 cifras significativas. Por lo tanto se produce una pérdida de precisión en las sumas anteriores por superar el número de cifras significativas. En la última suma podemos observar claramente como C redondea el número real.

Si queremos más precisión trabajando con números reales, utilizaremos el tipo `double` de C (estándar IEEE754 de 64 bits), que utiliza el doble de bits que el tipo `float` y tiene un rango de representación de $\pm 4.9e-324$ a $\pm 1.8e308$, con 16-17 cifras significativas.

3.3 Formato de impresión:

```
Pruebas de formatos de impresión
-----
Entero 205 sin formato 2 veces : 205 205
Entero 205 con formato (6)      :    205
Real 205.5 sin formato          : 205.500000
Real 205.5 con formato (exp)    : 2.055000e+02
Real 205.5 con formato (12)     :   205.500000
Real 205.5 con formato (12.0)   :      206
Real 205.5 con formato (12.2)   :    205.50

Char 'a' y 'b' sin formato      : a b
Char 'a' y 'b' con formato (6) :   a   b
Literal 'mesa' y 'silla' sin formato : mesa silla
Literal 'mesa' y 'silla' con formato (6) : mesa silla

Linea completa con entero 205(6), real 205.5(8.2) y 'mesa'(8)
 205 205.50 mesa
```

