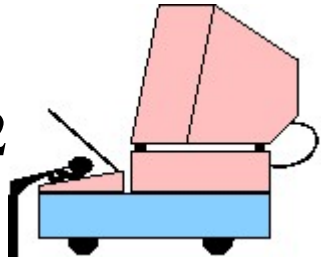
Fundamentos de Ordenadores 2



Práctica 2

Fecha de Entrega: 12 de Mayo de 2004

Entrega: A través del enlace http://www.it.uc3m.es/luis/fo2/p2/submit.html

Ejercicio 1: Cifra del César

Una de las formas más simples de cifrar mensajes es reemplazar cada letra por otra. Emisor y receptor disponen de la misma tabla de conversión, el primero para codificar el mensaje, y el segundo, aplicando exactamente el mismo procedimiento, obtiene el mensaje original. Este tipo de cifras se conocen como *cifras de sustitución*. La *cifra del César*, llamada así por ser utilizada por Julio César, es una cifra de sustitución que consiste en desplazar el alfabeto cifrado un cierto número de posiciones (indicado por la *clave*) respecto al alfabeto llano.

Por ejemplo, si la clave es `4`, la letra `'a'` se corresponde con la `'e'`, la `'b'` con la `'f'`, la `'v'` con la `'z'`, la `'w'` con la `'a'`, y la `'z'` con la `'d'`.

Utilizando el código ASCII de los caracteres, simplemente hay que sumar la clave al carácter original para cifrarlo (o restar para descifrarlo), excepto para las últimas letras, puesto que la suma desbordaría el rango de caracteres del alfabeto. Para estos casos hay que restar además el número de letras del alfabeto ASCII (26), para que continúe la correspondencia por el inicio del mismo.

Según el ejemplo anterior:

Carácter llano	*Carácter cifrado*	*Proceso*	*Observaciones*
`'a'` (0x61)	`'e'` (0x65)	`'e' = 'a' + 4`	
`'w'` (0x77)	`'a'` (0x61)	`'a' = 'w' + 4 - 26`	0x77 + 4 = 0x7B (`'{'`) > `'z'` (fuera del rango del alfabeto)

Dada la siguiente definición de datos:

```
.data
```

```
clave:  .byte 4
texto:  .asciz "veni, vidi, vinci"
```

Se pide: Completar el programa `cesar.s` que cifre la cadena de caracteres almacenada a partir de la dirección de memoria asociada a la etiqueta `texto`.

El programa debe sustituir cada carácter alfabético (se supone que ya están en minúsculas) por su correspondiente carácter cifrado, según la clave almacenada en la posición de memoria `clave`. Sólo se modifican los caracteres alfabéticos (los espacios, signos de puntuación, etc. no se ven afectados). El programa presenta por pantalla el mensaje antes y después de codificarlo.

Sugerencia: antes de sustituirlo, compruebar primero si el carácter está en el rango entre 'a' y 'z'.

Con los datos definidos tal y como se explica anteriormente, la salida del programa debe ser (nótese que únicamente las letras se cambian):

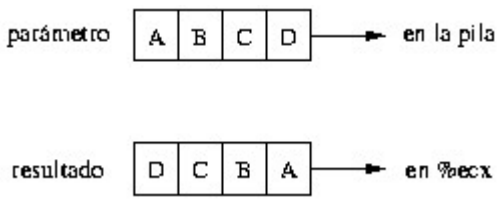
```
unix$ ./cesar
veni, vidi, vinci
zirm, zmhm, zmrgm
unix$:
```

Ejercicio 2: Reorganización de los caracteres de un string

En un único fichero con nombre `shuffle.s`, escribir los siguientes bloques de código:

1. Rutina con nombre `cambia` que recibe como parámetros 4 caracteres ASCII almacenados en cuatro posiciones consecutivas de memoria y devuelve esos cuatro bytes pero en orden diferente. Intercambia el primero con el cuarto y el segundo con el tercero. El parámetro se debe pasar a través de la pila, y el resultado se devuelve a través del registro %ecx.

   Por ejemplo, si se reciben como parámetro los bytes 'abcd', se deben devolver 32 bits que representen los códigos ascii en orden 'dcba' tal y como se ilustra en la siguiente figura.



   Si la rutina detecta el valor 0 en **alguno** de los cuatro bytes, devuelve como resultado cuatro bytes con valor 0.

2. Programa principal, que dado un string terminado en 0 y almacenado en la etiqueta `mensaje` realiza las 3 operaciones siguientes:

   1. Imprime el string tal y como está inicialmente seguido de un final de línea
   2. Modificar el string aplicando la operación de la rutina `cambia` en grupos de cuatro en cuatro bytes. Si el tamaño del string no es múltiplo de cuatro, no es preciso aplicar la transformación al último grupo. Imprimir el string resultante.

3. Repetir el paso anterior para verificar que al aplicar dos veces esta transformación se obtiene el string original.

Se debe utilizar la siguiente definición de datos en el programa:

```
      .data   /* Comienzo del segmento de datos. */
mensaje:.string "Computers are useless.  They can only give you answers."
```

Y con esta definición, la salida debe ser:

```
unix$ ./shuffle
Computers are useless.  They can only give you answers.
pmoCretura ssu esele  .syehTnac lno ig yy eva uoewsnrs.
Computers are useless.  They can only give you answers.
unix$
```

Ejercicio 3: Visualización de un Histograma por pantalla

En aplicaciones de proceso de datos, una forma típica de mostrar resultados es a través de un **histograma**. Un histograma es una colección de barras alineadas por su parte inferior y de altura proporcional a un conjunto de magnitudes dadas.

El objetivo de este problema es dibujar en pantalla un histograma a partir de un conjunto de números enteros dados. Para simplificar la implementación, se visualizará con barras horizontales de izquierda a derecha con tantos caracteres como indique la magnitud. Por ejemplo, dadas las magnitudes 3,5,7,4, en pantalla se visualiza:

```
AAA
BBBBB
CCCCCCC
DDDD
```

Se pide: Diseñar un programa ensamblador que **conste de dos ficheros** con nombres `histograma.s` y `linea.s`. En cada uno de estos ficheros se debe incluir lo siguiente:

Fichero linea.s

Este fichero contiene **únicamente** el código de la rutina **linea** que recibe como parámetros un entero que denota la longitud de la línea a imprimir, y un byte que contiene el código ascii de la letra a utilizar para dibujar la línea. Este parámetro se almacena en el byte menos significativo.

Los parámetros se depositan en la pila en este orden. La rutina devuelve como resultado a través de la pila el valor `1` si el entero dado como parámetro es negativo o mayor que 80, en cuyo caso no imprime nada, o `0` en caso contrario. El espacio para el resultado debe estar en la posición más lejana al puntero al bloque de activación. Esta rutina **únicamente imprime por pantalla una línea del histograma**.

Es posible que esta rutina necesite definir algún string. Se puede incluir un segmento de datos en su código.

Fichero histograma.s

Este fichero contiene el programa principal con la etiqueta `start` así como la definición de datos del histograma. El programa dibuja el histograma invocando repetidas veces la rutina del fichero `linea.s` con los parámetros pertinentes.

En caso de que se detecte un error, se detiene el dibujo del histograma y se imprime un mensaje de error. Las barras del histograma se deben imprimir cada una de ellas con letras mayúsculas correlativas, comenzando por la A. Se asume que nunca se darán más de 28 números.

Los datos utilizados por el programa principal se definen de la siguiente forma:

```
        .data   /* Comienzo del segmento de datos. */
datos:  .int 1, 3, 2, 10, 11, 15, 20, 32, 40, 45, 39, 26, 19, 21, 17, 13, 15
        .int 13, 6, 9, 10, 6, 5, 3, 1, 0
longitud: .int 26 /* Longitud del array de datos */
```

Se puede extender esta definición con los datos que hagan falta. Con esta definición, el programa debe imprimir por pantalla el siguiente texto:

```
unix$ ./histograma
A
BBB
CC
DDDDDDDDD
EEEEEEEEEE
FFFFFFFFFFFFFF
GGGGGGGGGGGGGGGGGGG
HHHHHHHHHHHHHHHHHHHHHHHHHHHHHH
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
JJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJJ
KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK
LLLLLLLLLLLLLLLLLLLLLLLL
MMMMMMMMMMMMMMMMMM
NNNNNNNNNNNNNNNNNNN
OOOOOOOOOOOOOOOO
PPPPPPPPPPPP
QQQQQQQQQQQQQQ
RRRRRRRRRRRR
SSSSSS
TTTTTTTTT
UUUUUUUUUU
VVVVVV
WWWW
XXX
Y

unix$
```

Punto Extra: Máximo Común Divisor con el algoritmo de Euclides Recursivo

El máximo común divisor de dos números enteros `a` y `b` puede calcularse mediante el algoritmo de Euclides, que su implementación recursiva es:

```
    int mcd(int a, int b) {
       int resto;
       resto = a % b      // a módulo b
       if (resto == 0) {
```

```
        return b;
     }
     return mcd(b, resto);
  }
```

Se pide: Escribir el programa `mcd.s` que contenga la rutina `mcd` que implemente el algoritmo recursivo de Euclides, teniendo en cuenta lo siguiente:

1. La intrucción `div` opera con registros implícitos para realizar la división de enteros. Si el divisor es un entero de 32 bits, los operandos de la instrucción son:

   - dividendo: **implícito** de 64 bits en la concatenación de los registros `EDX:EAX`.
   - divisor: operando explícito proporcionado en la instrucción.

   Además, esta instrucción produce **dos resultados**. El cociente se almacena en el registro `eax` y el resto en `edx`.

   La instrucción `div` también puede procesar datos de otros tamaños. Consulta el manual de instrucciones para comprobar qué registros se utilizan en otros casos.

2. El algoritmo se debe implementar de forma recursiva, por tanto deberá llamarse a sí mismo. Tanto parámetros como resultado **deben pasarse a través de la pila**.

3. La rutina sólo debe realizar los cálculos, **sin imprimir nada por pantalla**. Es el programa principal el que debe presentar el resultado. Para ello utilizar el siguiente esqueleto:

```
        .data
numeros:.int 2, 3, 5, 10, 12, 90
texto:  .asciz "mcd(%d, %d) = %d\n"

        .text
        .global start

start:  push %eax               /* salvar registros */
        push %ebx
        push %ecx
        push %edx
        push %esi

        mov $numeros, %ebx      /* direccion de los elementos */
        mov $3, %esi            /* contador del bucle */

calcular:
        /* calcular mcd */
        sub $4, %esp            /* reservar espacio para el resultado */
        push 4(%ebx)
        push (%ebx)
        call mcd                /* llamada a subrutina */
        add $8, %esp            /* restaurar pila */
        pop %eax

        /* imprimir resultado */
        push %eax
        push 4(%ebx)
        push (%ebx)
        push $texto
        call printf
```

```
        add $16, %esp

        add $8, %ebx            /* siguiente par de numeros */
        dec %esi
        jnz calcular

        pop %esi                /* restaurar registros */
        pop %edx
        pop %ecx
        pop %ebx
        pop %eax
        ret

mcd:                            /* Comienzo de subrutina */
      /* Inserta to código aquí */
        ret
```

El programa debe mostrar el siguiente resultado por pantalla:

```
unix$ mcd
mcd(2, 3) = 1
mcd(5, 10) = 5
mcd(12, 90) = 6
```

Last Revision: 04/25/2004 16:21:46