Proyecto N°2

Organización de Computadoras Departamento de Ciencias e Ingeniería de la Computation
Universidad Nacional del Sur
Segundo Cuatrimestre de 2016

Federico Esteche, LU 96477
Pablo Lencina, LU 82001
27/11/2016

Definiciones
y
Especificación de Requerimientos

Enunciado

El objetivo de este proyecto consiste en implementar en lenguaje ensamblador, un programa para volcar el contenido de un archivo en formato hexadecimal y ASCII. Esta salida será similar a la producida por comando hexdump -C.
La implementación debe realizarse utilizando el ensamblador Yasm, sobre arquitectura Intel x86, haciendo uso de las llamadas al sistema provistas por el sistema operativo Linux.

Opciones del programa

El programa, llamado volcar se debe ejecutar de la siguiente manera:
$ ./volcar [-h] <archivo>
<archivo>: La ruta a un archivo de cualquier formato (binario, imagen, texto u otro), de tamaño máximo 1MB.
-h: Imprime un mensaje de ayuda y tiene una terminación normal (0). Es **opcional**, y siempre aparece en primera posición en la lista de argumentos. El programa debe terminar su ejecución luego de imprimir el mensaje de ayuda, sin considerar otros argumentos que pudieran aparecer a continuación.

Objetivo y funcionalidades

El programa debe tomar el contenido del archivo de entrada y mostrarlo por pantalla, organizado de la siguiente forma:
[Dirección base] [Contenido hexadecimal] [Contenido ASCII]
La salida debe organizarse en filas de a 16 bytes. La primera columna muestra la dirección base de los siguientes 16 bytes, expresada en hexadecimal. Luego siguen 16 columnas que muestran el valor de los siguientes 16 bytes del archivo a partir de la dirección base, expresados en hexadecimal. La última columna (delimitada por caracteres '|') de cada fila muestra el valor de los mismos 16 bytes, pero expresados en formato ASCII, mostrando sólo los caracteres imprimibles, e indicando la presencia de caracteres no imprimibles con '.').
Es responsabilidad de la comisión investigar cuáles son los caracteres imprimibles, utilizando búsquedas web o bibliografía.
Por ejemplo, la ejecución de volcar, con el parámetro /bin/sh, debe presentar la siguiente salida:

```
$ ./volcar /bin/sh
00000000 7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00 |.ELF............|
00000010 03 00 3e 00 01 00 00 00 70 67 00 00 00 00 00 00 |..>.....pg......|
00000020 40 00 00 00 00 00 00 00 18 26 07 00 00 00 00 00 |@........&......|
00000030 00 00 00 00 40 00 38 00 09 00 40 00 17 00 16 00 |....@.8...@.....|
00000040 06 00 00 00 05 00 00 00 40 00 00 00 00 00 00 00 |........@.......|
00000050 40 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 |@.......@.......|
00000060 f8 01 00 00 00 00 00 00 f8 01 00 00 00 00 00 00 |................|
00000070 08 00 00 00 00 00 00 00 01 00 00 00 05 00 00 00 |................|
00000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................|
00000090 00 00 00 00 00 00 00 00 2e 0d 06 00 00 00 00 00 |................|
...
```

Códigos de retorno del programa

Cuando el programa finalize su ejecución, debe informar sobre la situación de terminación (*exit status*) a quien haya invocado al programa. Para ello se debe hacer uso de la llamada al sistema sys_exit, respetando la siguiente convención:

EBX	Detalle
0	Terminación normal.
1	Terminación anormal.
2	Terminación anormal por error en el archivo de entrada.

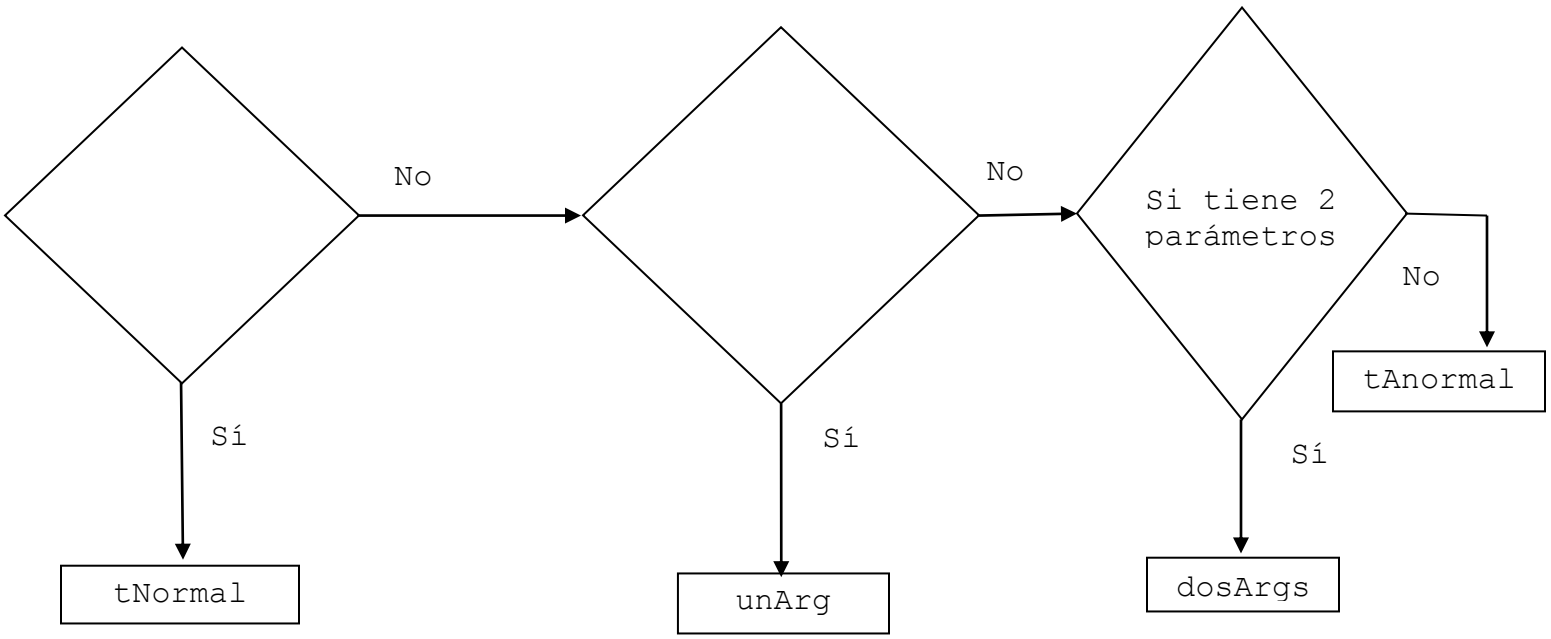
Arquitectura del Programa

_start

1. Descripción

Este es el procedimiento principal del programa.

2. Modelo de implementación

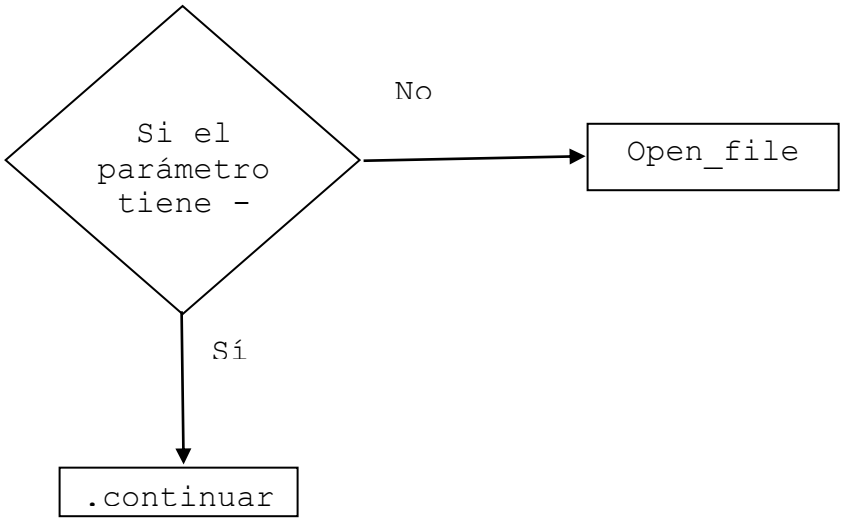


unArg

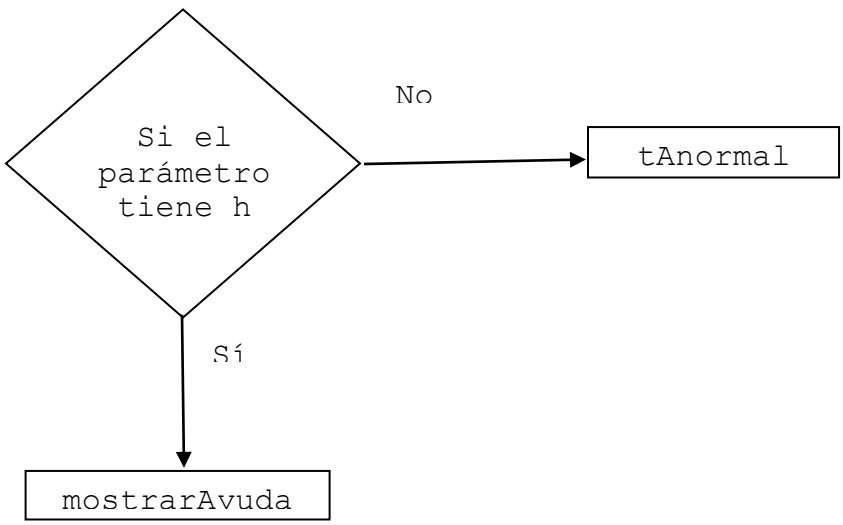
1. Descripción

Devuelve la ayuda en el caso de que tenga el parámetro –h o realizo el vuelco del archivo en el caso de que el parámetro sea un archivo.

2. Modelo de Implementación



.continuar

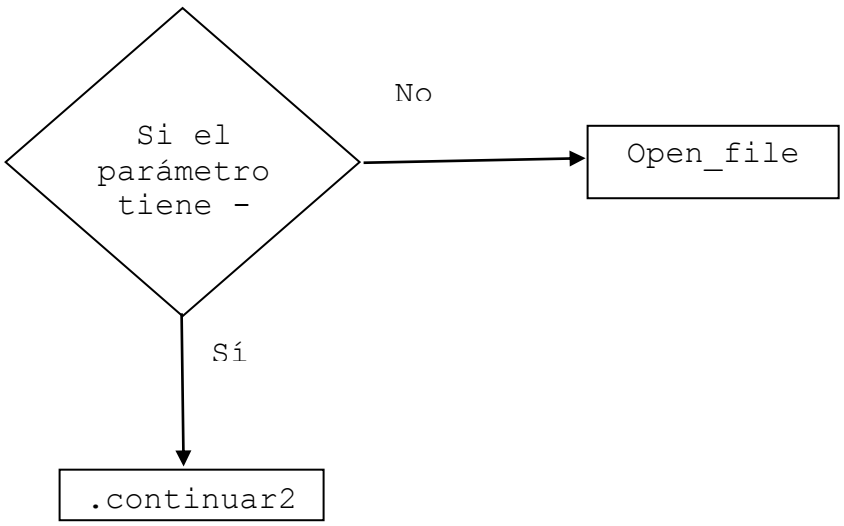


dosArgs

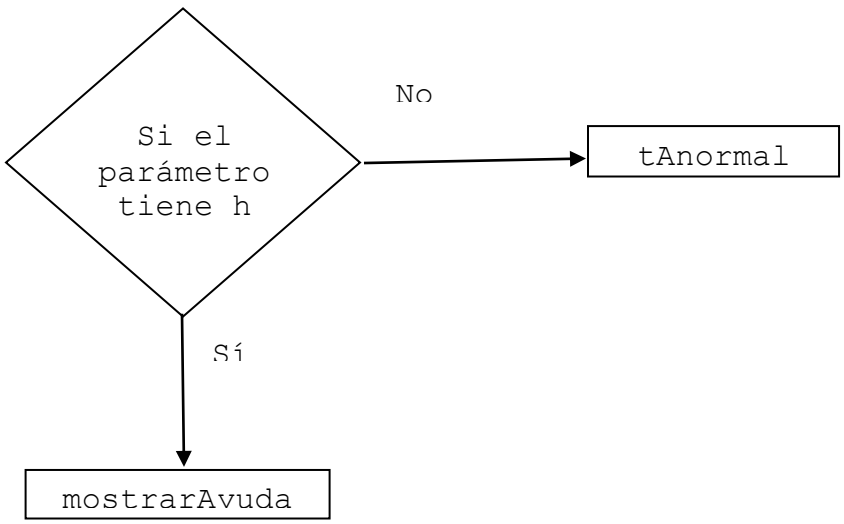
1. Descripción

Devuelve la ayuda en el caso de que tenga el parámetro –h, desestimo el otro argumento.

2. Modelo de Implementación



.continuar2

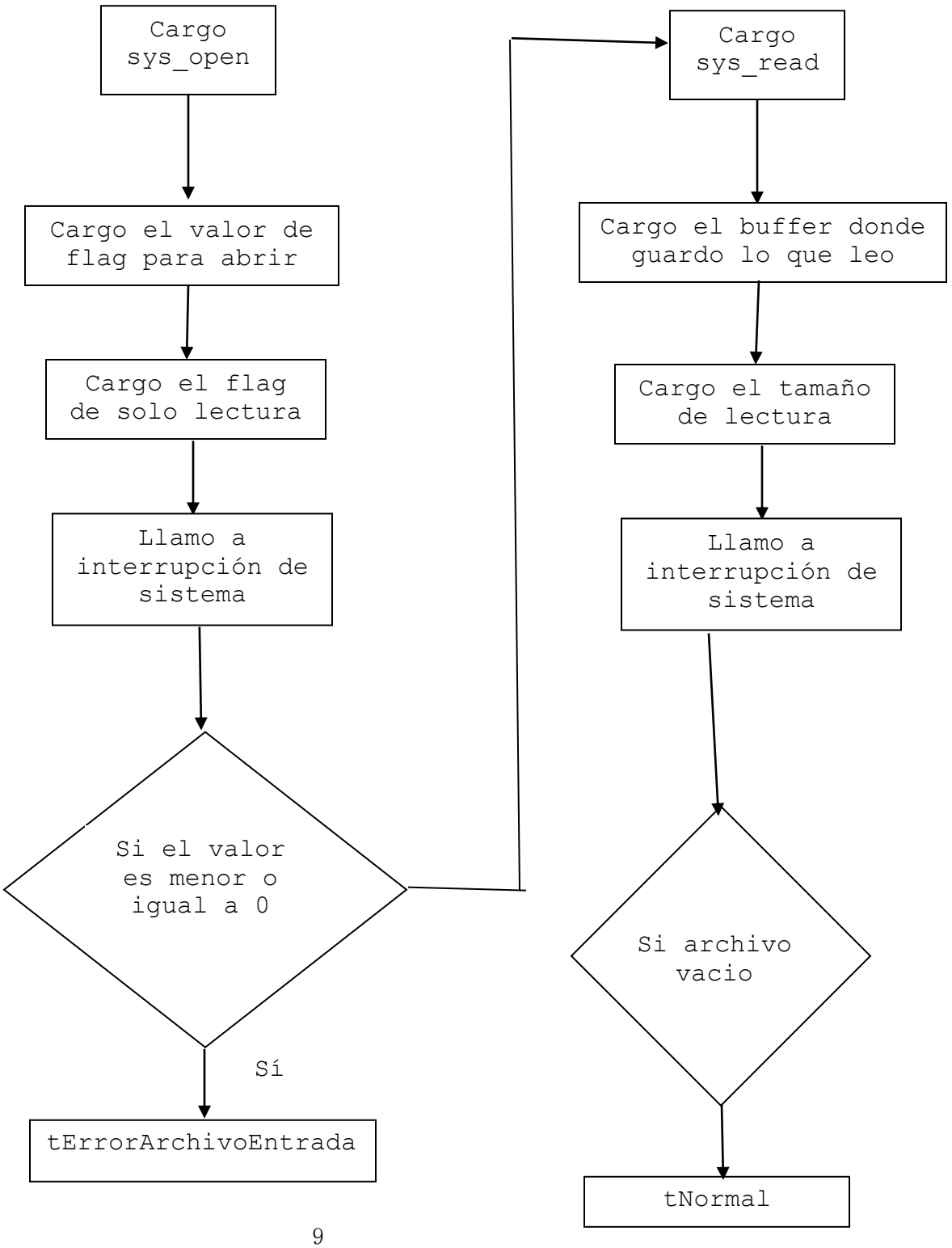


open_file

1. Descripción

Devuelve la ayuda en el caso de que tenga el parámetro –h o realizo el vuelco del archivo en el caso de que el parámetro sea un archivo.

2. Modelo de Implementación

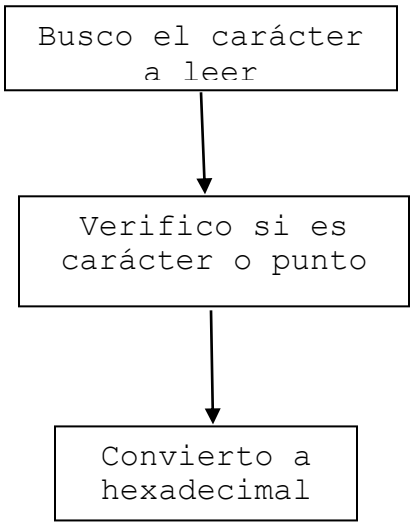


read_line

1. Descripción

Lee las líneas del buffer y lo guardo en la pila del programa, si no es imprimible pongo un “.”.

2. Modelo de Implementación

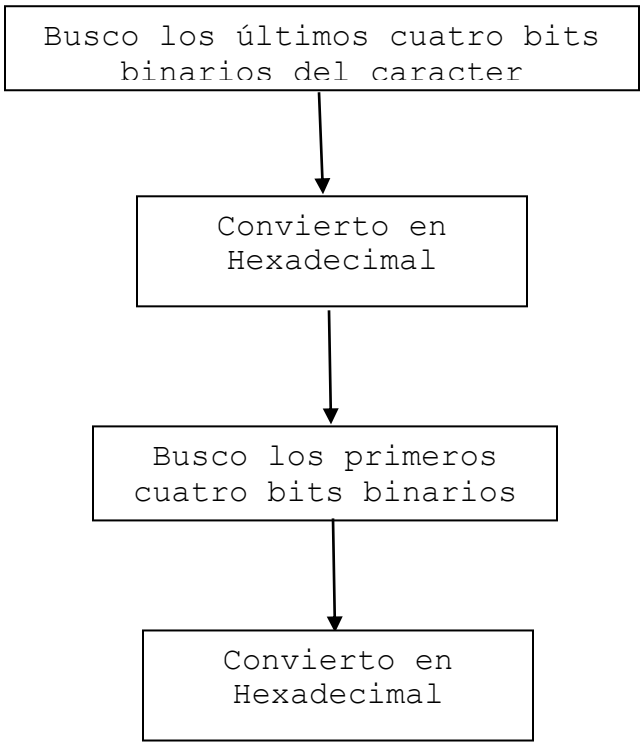


convertir_a_Hexa

1. Descripción

Recibe un carácter y lo convierte a hexadecimal.

2. Modelo de Implementación



imprimir_linea

1. Descripción

Imprime una linea de 16 bytes con el formato dado en el enunciado.

Eof

1. Descripción

Si el archivo termino imprimo la cantidad de líneas, si la última linea no tiene 16 bits la relleno

rellenar

1. Descripción

Rellena los caracteres faltantes en la última línea con espacios.

imprimir_faltante

1. Descripción

Imprime la línea que falta imprimir.

imprimir_contador

1. Descripción

Imprime la primera dirección de la linea.

tNormal

1. Descripción

   Termina el programa con valor de retorno 0 (Normal).

tAnormal

1. Descripción

   Termina el programa con valor de retorno 1 (Anormal).

tErrorArchivoEntrada

1. Descripción

   Termina el programa con valor de retorno 2 (Error del archivo).

mostrarAyuda

1. Descripción

   Imprime la ayuda.

contador

1. Descripción

   Convierto el valor de la dirección del carácter a hexadecimal.

caracter_o_punto

2. Descripción

   Verifica si es imprimible y si no lo es le pone ".".

Impr_salto_linea

1. Descripción

Imprime el salto de linea.

Codigo

volcar.asm

```
;Programa "volcar", Vuelca el contenido de un archivo segun un
formato.
;Autor:
;                Esteche Federico - Lencina Pablo.
;compilar:
;                $ yasm -f elf volcar.asm
;enlazar:
;                $ ld -o volcar volcar.o
;ejecutar:
;                $ ./volcar [parametro] <archivo>
;-------------------------------------------------------------------
----

%define hex_offset 8
%define char_offset 58

section .data
  ;formato de la linea de salida.
  linea db "000000  hh hh hh hh hh hh hh hh hh hh hh hh hh hh hh hh
|................|"
  long_Linea equ $ - linea ;                Tamaño de la linea
  ultima_Linea_Buffer db "000000"           ;Buffer para escribir la
ultima vez la

                ;cantidad de elementos leidos
  long_buffer_ultLinea equ $ - ultima_Linea_Buffer
  cont_lineas dd 0                          ;contador de lineas

  cant_caract_Leidos dd 0                   ;cantidad de caracteres
leidos hasta el momento.
  pos_caracter_en_linea dd char_offset ;offset a la posicion de la
linea donde insertar el char
  hex_pos dd hex_offset                     ;offset a la posicion de
la linea para insertar
                                                        ;la
representacion hexadecimal
  salto db 10                               ;"\n"
  barra db 7ch                              ;"|"
  resto dd 0                                ;Diferencia entre el
cont_lineas y la cantidad de lineas

  ;Mensaje de ayuda que se imprimira por pantalla con el argumento -
h.
  help db "Programa para volcar el contenido de un archivo en
formato hexadecimal y ASCII.", 10,
      db "                                                ",10,
                  db "Se detallan formato y opciones de invocacion:
", 10,
```

```
      db "Sintaxis: $ volcar [ -h ] < archivo_entrada >", 10,
      db "Los parametros entre corchetes denotan parametros
opcionales.",10,
                    db "Las opciones separadas por < > denotan
parámetros obligatorios.", 10,
      db "La opcion '-h' muestra un mensaje de ayuda (este
mensaje).", 10,
      db "El programa toma el contenido del <archivo_entrada> y lo
muestra por pantalla",10,
      db "organizado de la siguiente forma:",10,
      db "
",10,
                    db "    [Dirección base] [Contenido hexadecimal]
[Contenido ASCII]    ", 10,
                    db "
",10,
                    db "La salida se organiza en filas de a 16
bytes.",10
                    db "La primera columna muestra la dirección base
de los siguientes 16 bytes",10,
                    db "expresada en hexadecimal.", 10,
                    db "Luego siguen 16 columnas que muestran el
valor de los siguientes 16 bytes", 10,
                    db "del archivo a partir de la dirección base,
expresados en hexadecimal.", 10,
      db "La última columna (delimitada por caracteres '|') de cada
fila muestra el", 10,
                    db "valor de los mismos 16 bytes, pero expresados
en formato ASCII, mostrando ",10,
                    db "sólo los caracteres imprimibles, e indicando
la presencia de caracteres",10
      db ",no imprimibles con '.')."
                    db "Si no se especifica archivo alguno, la
terminación será anormal, mostrando un 3", 10,
      db "Para mas informacion, consulte la documentacion del
programa.", 10,
  longMensajeAyuda equ $ - help


section .bss

  buffer_ArchivoInput: resb 1048576         ;Se reserba 1mb en el
buffer donde se guardará el archivo.

section .text

global _start

_start:

;Controlo la cantidad de parametros
```

```
                pop eax          ; eax=cantidad de argumentos del
programa.
                pop ebx          ; ebx=nombre del programa, se
descarta.
                dec eax          ; Se descuenta 1 para no tener en
cuenta el nombre del programa.
                                 ; Solo se consideran aquellos
parametros dados por el usuario.
                cmp eax, 0       ; 0 parametros?
                je     tNormal        ; No se imprime nada,
terminación normal.

                ; Se pasa a determinar si tiene 1 o mas argumentos.
                cmp eax, 1
                je unArg
    cmp eax, 2
                jedosArgs
                jg tAnormal ; El maximo n de argumentos que se
pueden ingresar a la vez es 1,si supera
                                 ; esta cantidad se provoca una
salida con error.

;-----------------------------------------------------------------
----
unArg:
                pop ebx; Se extrae el argumento ingresado.
                mov ecx, [ebx]
                cmp cl, '-'              ;Controlo si es un "-"
                je .continuar            ;Si es un "-" , sigo
leyendo el argumento.

                jne    open_file         ; Si no hay un '-' solo
puede haber un path
                                         ; de archivo.
                .continuar
                inc ebx          ; Se saltea el '-'.
                mov ecx, [ebx]
                cmp cl, 'h'      ; Compruebo si es un "-h"
                je mostrarAyuda ; Caso positivo, muestro ayuda

                jne tAnormal; Caso contrario, muestro error.

;-----------------------------------------------------------------
-------
dosArgs:
                pop ebx          ; Se extrae 1er argumento
ingresado.
                mov ecx, [ebx]
                cmp cl, '-'     ; Controlo si es un "-"
                je .continuar2
```

```
                jne open_file    ; Sino hay un '-' tienen que ser
el path de un archivo.

                .continuar2
                inc ebx          ; Se saltea el '-'.
                mov ecx, [ebx]
                cmp cl, 'h'
                je mostrarAyuda  ;Si es un "-h" muestro ayuda y se
provoca una salida normal.
                jne tAnormal;Si no es "-h" significa que el
parametro es incorrecto.
;-----------------------------------------------------------------
-----------

open_file:

;Abro el archivo que tiene el texto a imprimir
                mov EAX,5        ;SYS_OPEN
                mov ECX,0        ;Sin flags al abrir.
                mov EDX,0        ;RDONLY (abro el archivo en modo
lectura)
                int 80h

    add EAX,2
                cmp EAX, 0  ;Si al abrir el archivo se produjo un
error, el File Descriptor sera -1
                jle tErrorArchivoEntrada ;salida con error.
                sub EAX,2

                push EAX             ;Se guarda el File Descriptor
para cerrar el archivo al terminar.

                mov EBX,EAX          ;Se mueve el FD a EBX
                mov EAX,3            ;SYS_READ
                mov ECX,buffer_ArchivoInput          ;Buffer donde
se guarda el archivo.
                mov EDX,1048576      ;Tamaño maximo del buffer (1
mb)
                int 80h              ; Interrupcion del sistema

                cmp EAX, 0           ;Si el archivo de entrada
está vacío.
                je  tNormal          ;terminacion Normal

                mov [cant_caract_Leidos],EAX    ;Se guarda la
cantidad de caracteres leidos.

;-----------------------------------------------------------------
----------------------
read_line:
```

```
; Busco el caracter que se quiere leer. La idea principal es manejar
el
; buffer como si fuese un arreglo, por lo cual para obtener un
caracter en
; la posicion n se le suma "n" a la posicion inicial un cont_lineas
de caracteres.

                mov EBX,buffer_ArchivoInput          ;Pongo en EbX
la direccion inicial del buffer
                add EBX,[cont_lineas]           ;Le sumo el
cont_lineas
                mov CL,[EBX]                    ;Copio el caracter
almacenado en EBX.
                push ECX                             ;Guardo el
caracter

;Se escribe el caracter leido en la posicion correspondiente.
                mov EAX,linea                        ;Se carga en
EAX la direccion inicial de la linea.
                add EAX,[pos_caracter_en_linea] ;Luego se le suma
el Offset.
                call caracter_o_punto                ;Esta llamada
a funcion retorna un caracter imprimible,
                                                     ;lo que
significa que dado un caracter, si este es imprimible
                                                     ;lo retorna y
si no es imprimible retorna un punto '.'.

                mov [EAX],CL                    ;Se crea una copia
del caracter leido

                pop ECX                              ;Se elimina el
caracter leido.

                mov EAX,linea                        ;Guardamos la
posicion inicial de la linea actual.
                add EAX,[hex_pos]               ;se le suma el
offset.
                call convertir_a_Hexa           ;Esta funcion
retorna la representacion hexadecimal del
                                                     ;caracter que
recibe como parametro.

                mov [EAX],CX                    ;Finalmente se lo
escribe (agrega) en la linea.


                inc DWORD [pos_caracter_en_linea]    ;Se incrementa
la posicion en la linea, donde se va a escribir
                                                          ;el
```

```
proximo caracter.
                add [hex_pos],DWORD 3               ;Incremento la
posicion donde escribir el hexa en la linea

;Se suma 1 al cont_lineas, luego se controla si es EOF, en este caso
se deja de leer.

                inc DWORD [cont_lineas]
                mov EAX,[cont_lineas]               ;Aca se lo
mueve a EAX con el objetivo de poder comparar luego.
                cmp [cant_caract_Leidos],EAX        ;Si el
cont_lineas es igual a la cantidad de caracteres leidos.
                je Eof                                  ;salto a
Eof

;Si el cont_lineas es 16 o multiplo de 16, se imprime la linea y se
vuelve al formato inicial.

                mov EAX,[cont_lineas] ;Se pone el cont_lineas en el
registro EAX.
                mov EBX,16            ;cantidad de bit por linea.
                mov EDX,0             ;Se pone EDX en cero, para
comparar el resto de la division.
                idiv EBX
                cmp EDX,0             ;Si el resto es 0 Se imprime
la linea por pantalla y se resetea la linea.
                je imprimir_linea

                jmp read_line         ;Se vuelve a la lectura de un
caracter.

;-----------------------------------------------------------------
-----------------------------
imprimir_linea:
                ;Se imprime la linea por pantalla
                mov EAX,4   ; SYS_WRITE
                mov EBX, 1  ; STDOUT
                mov ECX,linea
                mov EDX,long_Linea
                int 80h
;-----------------------------------------------------------------
-----------------------------
;Reseteo las posiciones donde voy a escribir los caracteres
mov [pos_caracter_en_linea],DWORD char_offset
                ;pos_caracter_en_linea=57
mov [hex_pos],DWORD hex_offset
                ;hex_pos=8


;Escribo en la linea el cont_lineas, exceptuando la primera que ya
esta en 000000
```

```
mov EAX,[cont_lineas]                   ;Cargo el cont_lineas para
imprimir la cantidad actual

mov EBX,linea
call contador                 ;Llamo a la funcion que me escribe el
cont_lineas en la linea

call impr_salto_linea

jmp read_line                       ;Vuelvo a imprimir una linea
;-------------------------------------------------------------------
-------------------------

Eof:

                ;Si es el fin del archivo se imprime el
cont_lineas.
                mov EAX,[cont_lineas]      ;Pongo el cont_lineas en
EAX
                mov EDX,0                  ;Pongo EDX en cero.
                mov EBX,16                 ;Se pone en EBX la
cantidad maxima de bits por linea.
                idiv EBX
                cmp EDX,0                  ;Si el resto es cero, se
acabaron los caracteres
                je imprimir_contador  ;por lo tanto imprimo el
cont_lineas.

                ;Si el resto no es cero, se debe guardar para saber
con cuantos caracteres tengo que llenar la ultima
                ;linea.

                sub EBX,EDX                ;16-resto
                mov [resto], EBX      ;Se guarda el resto en EBX.

                ;Se agrega una barra vertical al final de los
caracteres
                mov EAX,linea              ;"|"
                add EAX,[pos_caracter_en_linea]
                mov BL,BYTE [barra]
                mov [EAX], BL
                inc BYTE [pos_caracter_en_linea]
;-------------------------------------------------------------------
-------------------------
; se rellenan todos los caracteres faltantes de la ultima linea con
espacios en blanco.
rellenar:

                  ;Reemplazo el caracter en la linea por un espacio
                  mov EAX,linea
                  add EAX,[pos_caracter_en_linea]    ;Posicion
```

```
donde se deberia seguir escribiendo.
                    mov BL,' '
                    mov [EAX],BL
                    inc BYTE [pos_caracter_en_linea]   ;Se incrementa
para seguir rellenando.

                   ;Los hexadecimales se deben llenar con dos
espacios. (por el formato de salida).
                    mov EAX,linea
                    add EAX,[hex_pos]
                    mov BL,' '
                    mov BH,' '
                    mov [EAX],BX
                    add BYTE [hex_pos],3

                    dec BYTE [resto]
                    cmp [resto], WORD 0
                    je imprimir_faltante


                    jmp rellenar
;-------------------------------------------------------------------
----------------------------

imprimir_faltante:
                  ;Imprimo la linea que falta
                  mov EAX,4         ; SYS_WRITE
                  mov EBX, 1        ;STDOUT
                  mov ECX,linea
                  mov EDX,long_Linea
                  int 80h

                  call impr_salto_linea
;-------------------------------------------------------------------
----------------------------
;Imprimo el cont_lineas con el valor final de caracteres leidos
imprimir_contador:

                  mov EAX,[cont_lineas]
                  mov EBX,ultima_Linea_Buffer          ;Buffer
especial que contiene "000000"
                  call contador

                  mov EAX, 4        ;SYS_WRITE
                  mov EBX, 1        ;STDOUT
                  mov ECX,ultima_Linea_Buffer
                  mov EDX,long_buffer_ultLinea
                  int 80h
                  call impr_salto_linea

                  ;Cierro el archivo
```

```
                pop EBX
                mov EAX, 6       ;sys_close
                int 80h
;-------------------------------------------------------------------
-----------------------------
; Terminaciones, en ebx se guarda el modo (indicadas en el
enunciado).

tNormal:
                ; Salgo sin error
                mov EAX,1 ; SYS_EXIT
                mov EBX,0 ; 0 Significa terminacion normal.
                int 80h

tAnormal:
                mov     eax, 1   ;sys_exit
    mov     ebx, 1  ;1 significa terminacion anormal
    int     80h

tErrorArchivoEntrada:
    mov     eax, 1  ;sys_exit
    mov     ebx, 2  ;2 significa error por archivo de entrada.
    int     80h


;-------------------------------------------------------------------
-----------------------------
mostrarAyuda:
                mov eax, 4       ; sys_write
                mov ebx, 0       ; La consola.
                mov ecx, help    ; Puntero a el texto a mostrar
                mov edx, longMensajeAyuda  ; su tamaño.
                int 80h          ; llamada al servicio.
                jmp tNormal
;-------------------------------------------------------------------
-----------------------------

convertir_a_Hexa:
;Convierte el caracter en CL a hexa . Los caracteres hexa se
almacenan
;en los registros CH y CL.
;Los caracteres se guardan en orden invertido para que la impresion
por pantallas sea correcta.
;RETORNO:
;  CL - Primer hexa.
;  CH - Segundo hexa.

  mov DL,CL         ; Se copia el caracter
  and DL,00001111b  ;se recuperan los 4 bits menos significativos.
  call en_hexa    ;Convierto 4 bits a hexa (0..9A..F)
```

```
  mov CH,DL         ;Copio segundo hexa en CH. (ORDEN INVERTIDO)
  mov DL,CL         ;Hago copia de caracter
  shr DL,4          ;Obtengo primeros 4 bits

  call en_hexa    ;Convierto 4 bits a hexa (0..9A..F)

  mov CL,DL         ;Copio primer hexa en CL. (ORDEN INVERTIDO)
  ret                    ;Fin

;--------------------------------- - -----------------------------

;Convierte (el numero almacenado en el registro EAX) ascii hexa,
para guardarlo
; en los primeros 5 lugares del registro EBX (recordar que en este
registro esta
; el buffer)
contador:
  add EBX, 5        ;Sumo 5 a buffer, se empieza desde la posicion
menos significativa
  mov ECX,16        ;ECX=16 divisor


bucle_contador:

  mov EDX,0         ;Se resetea EDX para utilizar la division entera
  idiv ECX          ;Se divide por 16,en el registro EAX se guarda
el
                           ;resultado y en el registro EDX se
guarda el resto.

  call en_hexa    ;se convierte el resto de la division en hexa.
  mov [EBX], DL   ;se mueve el resultado al buffer.
  dec EBX           ;Decremento buffer. Muevo una posicion a la
izquierda

  cmp EAX,0         ;Comparo Cociente con 0.
  jne bucle_contador     ;Si (Cociente!=0): sigo dividiendo

  ret                    ;Si (Cociente=0): Fin

;-----------------------------------------------------------------
--------
;Convierte un caracter almacenado en DL a hexa (0..9A..F)
en_hexa:
  cmp DL,9                    ;
  jg esAlfabetico   ;si el caracter es mayor a 9, entonces es A-B-C-
D-E o F

esnumero:           ;Si el caracter es menor o igual a 9 entonces es
0-1-2-3-4-5-6-7-8-9
  add DL,48         ;Si el caracter es un numero entre el 0 y el 9,
```

```
le sumo 48

  jmp fin_en_hexa ;saltar al fin del bucle

esAlfabetico:
  add DL,55         ;Caracter es letra le sumo 55 para convertir a
(A..F)

fin_en_hexa:
  ret                   ;Fin del bucle.


;-----------------------------------------------------------------
--------
caracter_o_punto:
;Si el caracter guardado en el registro CL es imprimible, lo
retorno.Si no es imprimible
;lo transforma en un punto ".".
;Cabe aclarar que los caracteres imprimibles van del 0 al 31 y
ademas el 127.

  cmp CL, 31
  jg imprimible     ;si el caracter es mayor que 31 entonces es
imprimible.

no_imprimible:      ;si el caracter es menor que 31 entonces no es
imprimible.

  mov CL, 46        ;Se convierte el caracter no imprimible en un
punto.
  jmp fin_caracter_o_punto ;Salto al fin del bucle.

imprimible:         ;Si el caracter es mayor a 31, controlo que no
sea 127

  cmp CL, 127       ;
  je no_imprimible  ;Si es 127, entonces no es imprimible

fin_caracter_o_punto:

  ret                   ;Fin cel bucle.


;-----------------------------------------------------------------
-----------
; imprimo un salto de linea por pantalla
impr_salto_linea:
                mov EAX,4; SYS_WRITE
                mov EBX, 1 ; STDOUT
                mov ECX,salto
                mov EDX,1
```

```
int 80h
ret
```