Escuela Politécnica Superior

UAM

Laboratorio de Sistemas Basados en Microprocesadores

Práctica 3a: Diseño de programas utilizando C y ensamblador del 80x86.

En esta práctica combinaremos ficheros fuente en código de alto nivel en lenguaje C linkados con funcionalidades implementadas a bajo nivel en lenguaje ensamblador. En particular se pide al alumno crear unas funciones en ensamblador que serán llamadas desde un código C proporcionado. Será crucial entender el uso de la pila en las llamadas a funciones para el paso de parámetros y recordar el uso standard de AX o DX:AX para retornar valores de una función ensamblador.

Para el desarrollo de la práctica se proporciona **un programa principal escrito en C** (pract3a.c) que hace llamadas a las funciones que deberán ser **desarrolladas por los alumnos en ensamblador en dos ficheros** (módulos) diferentes (pract3a.asm y pract3b.asm). El alumno deberá realizar el correspondiente análisis de los requisitos expuestos y realizar su implementación.

Ejercicio 1: Programa pract3a.asm (6 ptos)

En el módulo pract3a.asm se deben desarrollar en ensamblador un conjunto de funciones, F1 y F2 de acuerdo a los prototipos de cabecera especificados abajo. Estas funciones manejarán interrupciones de video para dibujar formas simples por pantalla. Dichas funciones serán llamadas desde el programa principal **pract3a.c** que se proporciona con el material de la práctica. Se proporciona además una serie de ejemplo de uso de interrupciones 10h y 15h para trabajar en modo video y para implementar esperas activas (respectivamente).

F1: char pintaPixel(unsigned char color, int x,int y); **(2 ptos)**

La función realizará la configuración a modo video 640x480 16 color graphics (VGA) y procederá a llamar a la interrupción 10h para pintar del color especificado en la entrada el pixel de pantalla indicado por los parámetros x e y también pasados por parámetro. Antes de retornar un valor de OK o ERR se deberá restaurar el modo video/texto.

```
/*********************************************************************
* Nombre: pintaPixel
*
* funcionalidad: pasa a modo video, pinta un pixel de color COLOR en la posicion
*                                de la pantalla dado por X e Y
* Parámetros de entrada:
*                               - unsigned char COLOR
*                               - int X ; posicion fila en pantalla
*                               - int Y ; posicion columna en pantalla
* Valores retorno:
*                               -  char: -1 si posicion fuera de rango permitido; 0 si ok
*********************************************************************/
```

F2: unsigned char pintaCuadrado (unsigned char color, int tam,int x,int y); **(4 ptos)**

La función realizará la configuración a modo video 640x480 16 color graphics (VGA) y procederá a llamar a la interrupción 10h para pintar del color especificado en la entrada los pixeles correspondientes a un cuadrado de tamaño (en pixeles) dado en el párametro de entrada "tam" y comenzando en la posición de pantalla indicado por los parámetros x e y también pasados por parámetro. Antes de retornar un valor de OK o ERR se deberá restaurar el modo video/texto.

```
/**************************************************************************
* Nombre: pintaCuadrado
*
* funcionalidad: pasa a modo video, pinta un pixel de color "COLOR" en la posicion
*                                de la pantalla dado por "X" e "Y" además de continuar
*                               pintando todos  los pixeles vecinos hasta completar un
*                               cuadrado de dimensiones "tam"
* Parámetros de entrada:
*                               - unsigned char: COLOR
*                               - int tam; dimensión en pixeles del lado del cuadrado
*                               - int X ; posicion fila en pantalla donde comienza el cuadrado
*                               - int Y ; posicion columna en pantalla comienzo de cuadrado
* Valores retorno:
*                               -  char: -1 si posicion fuera de rango permitido; 0 si ok
**************************************************************************/
```

Ejercicio 2: Programa pract3b.asm (4 ptos)

En el módulo pract3b.asm se debe desarrollar en ensamblador una función F3 que será llamada también desde el programa principal **prac3a.c** de acuerdo al siguiente prototipo:

F3: void pintaListaPixeles(unsigned int numPixeles, unsigned char bgcolor, long int tiempoEspera, int* pixelList_x, int* pixelList_y, unsigned char* pixelList_color); **(4 ptos)**

La función F3 deberá pintar en pantalla todos los pixeles dados por las posiciones X e Y de los vectores de entrada (lista) nombrados pixelList_x y pixelList_y. El número de elementos de las listas de pixeles será indicado por parámetro en la variable numPixeles. Como ejemplo, se pintará el pixel en la posición del primer elemento de pixelList_x y primer elemento de pixelList_y. Además cada pixel podrá tener un color diferente, que será dado en la lista pixelList_color. El fonde de la pantalla se pintará con el color bgcolor pasado por parámetro. También se pide configurar el tiempo de espera o visualización entre llamadas a la función de pintado, usando para ello la interrupción 15H para el tiempo introducido en la variable long int "tiempoEspera".

Notas adicionales:

- El compilador precede todas las referencias externas con el carácter "_", por lo que es necesario que todas las funciones desarrolladas en ensamblador comiencen con el mismo (ejemplo: _pintaPixel)
- Todas las funciones desarrolladas en ensamblador deben ser declaradas como PUBLIC para poder ser llamadas desde el programa principal
- No se permite la declaración de variables globales en ensamblador. De hecho, cada módulo desarrollado en ensamblador contendrá **exclusivamente** un solo segmento de código, que debe comenzar con las siguientes líneas:

```
<nombre módulo> SEGMENT BYTE PUBLIC 'CODE'
ASSUME CS: <nombre módulo>
```

- El programa a desarrollar en C (pract3a.c) debe estar compilado en modelo LARGE. Las funciones desarrolladas en ensamblador serán FAR.
- El programa pract3a.c pedirá al usuario diferentes valores necesarios para probar cada una de las funciones desarrolladas en ensamblador en los diferentes módulos, mostrando el resultado de forma gráfica pintando la pantalla.
- Se adjunta un fichero vgaddga.asm de ejemplo de uso de las interrupciones 10H y 15H, además de la información proporcionada en el anexo. Importante recordar que vgaddga.asm es un programa "stand-alone" que no está desarrollado para ser llamado desde un programa en C. Sólo sirve de referencia por tanto para el uso de las interrupciones mencionadas.

ENTREGA DE LA PRÁCTICA: Fechas límite y contenido

Se deberá subir a Moodle un fichero **zip** que contenga los ficheros fuentes de los programas y el fichero makefile. Sólo podrá ser subido por uno de los miembros de la pareja. Los ficheros a entregar deberán contener en la cabecera los nombres de los autores y el identificador de la pareja. Así mismo, el código de los ficheros entregados deberá estar correctamente tabulado y comentado. La falta de comentarios o la baja calidad de éstos, será calificada negativamente.

Ejercicio 1:

- Contenido: zip con el fichero fuente **pract3a.asm** y el makefile utilizado
- Grupo del Jueves: 15 de Abril a las 20.15
- Grupos del Viernes: 16 de Abril a las 19:15h

Ejercicio 2:

- Contenido: un zip con el fichero fuente pract3b.asm y el makefile utilizado
- Grupos del Jueves: 21 de Abril a las 23:55
- Grupos del Viernes: 22 de Abril a las 23:55h

Anexo I: Compilación de un proyecto desarrollado en C y ensamblador

En esta práctica disponemos de 3 ficheros, el programa principal escrito en C (pract3a.c) y los 2 módulos que contienen las funciones escritos en ensamblador (pract3a.asm y pract3b.asm).

Para realizar la compilación del programa en C se utilizará el compilador del TurboC (tcc). Para conocer todas las opciones de compilación que ofrece podemos ejecutar tcc sin parámetros dentro del DosBox. Los módulos en ensamblador se compilarán como en las prácticas anteriores, utilizando el tasm.

El programa en C debe ser compilado con la opción *–ml* (memory model large), mientras que los módulos en ensamblador deben ser ensamblados con la opción */ml* (case sensitivity on all symbols).

Para generar el ejecutable final "pract3.exe" podemos utilizar el siguiente makefile (respetando los tabuladores):

```
all: pract3.exe

pract3.exe: pract3.obj pract3a.obj pract3b.obj

	tcc -v -ml -Lc:\compila\tc\lib pract3.obj pract3a.obj pract3b.obj

pract3.obj: pract3a.c

	tcc -c -v -ml -Ic:\compila\tc\include pract3a.c

pract3a.obj: pract3a.asm

	tasm /zi /ml pract3a,,pract3a

pract3b.obj: pract3b.asm

	tasm /zi /ml pract3b,,pract3b
```

Anexo II: Interrupción 10H – Set Video Mode

Function	Function code	Parameters	Return
Set video mode	AH=00h	AL = video mode Where possible video modes are: AL = 00 40x25 B/W text (CGA,EGA,MCGA,VGA) = 01 40x25 16 color text (CGA,EGA,MCGA,VGA) = 02 80x25 16 shades of gray text (CGA,EGA,MCGA,VGA) = 03 80x25 16 color text (CGA,EGA,MCGA,VGA) = 04 320x200 4 color graphics (CGA,EGA,MCGA,VGA) = 05 320x200 4 color graphics (CGA,EGA,MCGA,VGA) = 06 640x200 B/W graphics (CGA,EGA,MCGA,VGA) = 07 80x25 Monochrome text (MDA,HERC,EGA,VGA) = 08 160x200 16 color graphics (PCjr) = 09 320x200 16 color graphics (PCjr) = 0A 640x200 4 color graphics (PCjr) = 0B Reserved (EGA BIOS function 11) = 0C Reserved (EGA BIOS function 11) = 0D 320x200 16 color graphics (EGA,VGA) = 0E 640x200 16 color graphics (EGA,VGA) = 0F 640x350 Monochrome graphics (EGA,VGA) = 10 640x350 16 color graphics (EGA or VGA with 128K) 640x350 4 color graphics (64K EGA) = 11 640x480 B/W graphics (MCGA,VGA) = 12 640x480 16 color graphics (VGA) = 13 320x200 256 color graphics (MCGA,VGA) = 8x EGA, MCGA or VGA ignore bit 7, see below = 9x EGA, MCGA or VGA ignore bit 7, see below	AL = video mode flag / CRT controller mode byte
Set text-mode cursor shape	AH=01h	CH = Scan Row Start, CL = Scan Row End Normally a character cell has 8 scan lines, 0-7. So, CX=0607h is a normal underline cursor, CX=0007h is a full-block cursor. If bit 5 of CH is set, that often means "Hide cursor". So CX=2607h is an invisible cursor.	

		Some video cards have 16 scan lines, 00h-0Fh. Some video cards don't use bit 5 of CH. With these, make Start>End (e.g. CX=0706h)	
Set cursor position	AH=02h	BH = Page Number, DH = Row, DL = Column	
Get cursor position and shape	AH=03h	BH = Page Number	AX = 0, CH = Start scan line, CL = End scan line, DH = Row, DL = Column
Read light pen position (Does not work on VGA systems)	AH=04h		AH = Status (0=not triggered, 1=triggered), BX = Pixel X, CH = Pixel Y, CX = Pixel line number for modes 0Fh-10h, DH = Character Y, DL = Character X
Select active display page	AH=05h	AL = Page Number	
Scroll up window	AH=06h	AL = lines to scroll (0 = clear, CH, CL, DH, DL are used), BH = Background Color and Foreground color. BH = 43h, means that background color is red and foreground color is cyan. Refer the BIOS color attributes CH = Upper row number, CL = Left column number, DH = Lower row number, DL = Right column number	
Scroll down window	AH=07h	like above	
Read character and attribute at cursor position	AH=08h	BH = Page Number	AH = Color, AL = Character

Write character and attribute at cursor position	AH=09h	AL = Character, BH = Page Number, BL = Color, CX = Number of times to print character	
Write character only at cursor position	AH=0Ah	AL = Character, BH = Page Number, CX = Number of times to print character	
Set background/border color	AH=0Bh, BH = 00h	BL = Background/Border color (border only in text modes)	
Set palette	AH=0Bh, BH = 01h	BL = Palette ID (was only valid in CGA, but newer cards support it in many or all graphics modes)	
Write graphics pixel	AH=0Ch	AL = Color, BH = Page Number, CX = x, DX = y	
Read graphics pixel	AH=0Dh	BH = Page Number, CX = x, DX = y	AL = Color
Teletype output	AH=0Eh	AL = Character, BH = Page Number, BL = Color (only in graphic mode)	
Get current video mode	AH=0Fh		AL = Video Mode, AH = number of character columns, BH = active page
Change text mode character set[3]	AH=11h	BH = Number of bytes per character, CX = Number of characters to change, DX = Starting character to change, ES:BP = Offset of character data	
Write string (EGA+, meaning PC AT minimum)	AH=13h	AL = Write mode, BH = Page Number, BL = Color, CX = Number of characters in string, DH = Row, DL = Column, ES:BP = Offset of string	
set VESA-Compliant video modes, beginning at 640 by 480 and reaching 1280 by 1024 with 256 colors	AX=4f02h	BX = video mode, if Sign bit (bit 15) set, video memory will not be refreshed	
Other VESA VBE commands	AX=4F00h to 4F15h	See spec	See spec

Anexo III: Paleta de colores

Dec	Hex	Binary	Color
0	0	0000	Black
1	1	0001	Blue
2	2	0010	Green
3	3	0011	Cyan
4	4	0100	Red
5	5	0101	Magenta
6	6	0110	Brown
7	7	0111	Light Gray
8	8	1000	Dark Gray
9	9	1001	Light Blue
10	A	1010	Light Green
11	B	1011	Light Cyan
12	C	1100	Light Red
13	D	1101	Light Magenta
14	E	1110	Yellow
15	F	1111	White

Anexo IV: Ejemplo de Código para Interrupciones 10H y 15H

```
        ; usamos interrupción 10h para entrar en modo video
MOV AH,0Fh ; Peticion de obtencion de modo de video
INT 10h ; Llamada al BIOS
MOV _VID,AL ; SALVO EL MODO VIDEO Y LO ALMACENO EN AL


mov ah, 00h ; configuramos entrada a modo video
;mov al, 13H ; a 320x200 256 color graphics (MCGA,VGA)
mov al, 12h ; 640x480 16 color graphics (VGA)
int 10h


; configuramos color fondo verde 0010b
mov ah, 0bh
mov bh, 00h
mov bl, 02h; 0 color negro 2 ; color verde
int 10h


;Int pintar punto en pantalla: AH=0Ch        AL = Color, BH = NumPag, CX = x, DX = y
mov ah, 0Ch
; lee de la pila el valor apropiado para el color
```

```
mov al,4  ; color rojo 4, color blanco 15
mov bh, 00h
mov cx,50      ;Posición X donde pintar el punto
mov dx,45      ; Posición Y donde pintar el punto
int 10h


; llamada a la interrupción 15H para espera activa
MOV     CX, 0FH ;1 second = F:4240H --> 3 seconds 2D:C6C0
MOV     DX, 4240H
MOV     AH, 86H ;int 15h con AH=86h para espera de microsegundos en cx:dx
INT     15H


mov ah, 00h ; restaurar configuracion entrada a modo video
mov al, _VID ;
int 10h
```