

Romo García Miguel Ángel.

## EXAMEN PARCIAL 1.

**Tarea:** Elaborar un ejemplo para llamar una función en ensamblador desde C.

El ejemplo consiste en sumar tres números, compilado y corrido en Ubuntu. El código del archivo en C se muestra a continuación:

```
/* Programa en C */
extern int asmfun();

int kmain(int arg1, int arg2,int arg3){
int val=asmfun(arg1,arg2,arg3);
return val;
}
```

La función *asmfun* está declarada en el código en ensamblador mostrado a continuación:

```
;Programa en ensamblador
global asmfun
asmfun:
    push    ebp           ; Crea stack frame
    mov     ebp, esp
    mov     eax, [ebp+8]   ; Toma el primer argumento
    mov     ecx, [ebp+12] ; Toma el segundo argumento
    mov     ebx, [ebp+16] ; Toma el tercer argumento
    add     eax, ecx       ; Suma el primer argumento con el segundo
    add     eax, ebx       ; Al resultado se le suma el tercero
    pop     ebp           ; Restaurar el puntero base
    ret                     ;Regresa a donde fue llamado
```

Los nombres de los archivos son *main.c* y *asmfun.s* respectivamente. Para comprobar el funcionamiento de los códigos se utilizó como código base, el ejemplo de “*The little book about os development*” para llamar una función en de desde ensamblador. La liga para descargar el repositorio es la siguiente: <https://github.com/hurricanerix/littleos>.

Dentro del repositorio encontraremos archivos que se deben modificar para poder probar este ejemplo. Como primer paso se modificara el Makefile presente en la carpeta, modificando la primer linea de la siguiente manera:

```
OBJECTS=obj/loader.o obj/main.o obj/asmfun.o
```

El archivo *asmfun.s* creado previamente debe de copiarse a la carpeta src presente en el repositorio, al igual que el archivo *main.c*.

El otro archivo a modificar es el loader.s que se corre cuando se inicia el sistema con bochs. El código queda de la siguiente manera:

```
extern kmain
```

```
loader:                                ; the loader label (defined as entry point in the linker script)
    ; The assembly code
    push dword 1                        ; arg3
    push dword 2                        ; arg2
    push dword 3                        ; arg1
    call kmain
```

En esta parte del código se llama a la función *kmain*, presente en nuestro archivo main.c, por lo que es necesario declararla como *extern*, dando a entender que está declarada en algún otro archivo. Es importante notar como se acomodan los argumentos que entran a la función *kmain*.

De esta manera, *loader* llama a la función *kmain* escrita en C, que a su vez llama a la función *asmfun* escrita en ensamblador que suma los 3 argumentos que se pasan desde el loader, regresando el valor obtenido primero a la función *kmain* y después de regreso al loader, guardando el valor obtenido en el registro *eax* (*RAX* en caso de Ubuntu).

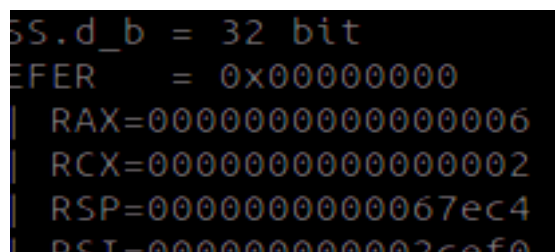
Para compilar, abrimos una terminal y nos dirigimos a la carpeta donde está el archivo Makefile, escribiendo *make*. Después se teclea lo siguiente:

```
bochs -f bochsrc.txt -q
```

Se abrirá entonces bochs y se correrá el programa loader, quedando después en el ciclo infinito loop del archivo. Se cierra el emulador y para comprobar el resultado se teclea:

```
cat bochslog.txt
```

Se busca en el archivo el registro *eax* (*RAX* en Ubuntu) y debe contener el resultado de la suma en hexadecimal, como se muestra a continuación:



```
SS.d_b = 32 bit
EFER    = 0x00000000
RAX=0000000000000006
RCX=0000000000000002
RSP=00000000000067ec4
RST=0000000000002cfe0
```

Imagen 1. Resultado de la suma en el RAX