Práctica 13

Acceso a subrutinas de ensamblador por programas de alto nivel

---

Luis Eduardo Galindo Amaya (1274895)

Asignatura	Organización de Computadoras (331)
Docente	Arturo Arreola Alvarez
Fecha	25-11-2022

Acceso a subrutinas de ensamblador por programas de alto nivel

Luis Eduardo Galindo Amaya (1274895)

25-11-2022

Objetivo

Desarrollar subrutinas de lenguaje ensamblador para ser llamadas desde un lenguaje de alto nivel para comprender las convenciones que los compiladores trabajan en sistemas basados en microprocesador.

Desarrollo

1. Cree una carpeta llamada P13_nombre_apellido. Dentro de esta carpeta crear los archivos `P13_c.c` y `P13_a.asm`.

2. En el archivo P13_c.c definir una función externa llamada sumaMatrices, de la siguiente manera:

```
extern void sumaMatrices(int[WIDTH][HEIGHT], int[WIDTH][HEIGHT],
                         int[WIDTH][HEIGHT], int, int);
```

- a. Recibe 5 parámetros, 3 matrices, y otros 2 datos que representan las dimensiones de dichas matrices (Las matrices serán del mismo tamaño).
- b. Los primeros 2 parámetros son las matrices a ser sumadas. El tercer parámetro es la matriz resultante.
- c. Crear dos matrices de enteros con las dimensiones WIDTH y HEIGHT. Definir WIDTH y HEIGHT como Macros de C.
- d. Inicializar ambas matrices.

3. En el archivo P13_a.asm crear la subrutina sumaMatrices que realice el proceso de la suma de todos los elementos de las matrices pasadas como parámetros y los retorne.

4. Llamar la subrutina sumaMatrices definida en ensamblador desde C, pasando como parámetros las matrices declaradas anteriormente.

5. Validar su algoritmo en ensamblador imprimiendo las matrices desde C.

Capturas

```
make -k
nasm -f elf P13_ASM.asm
gcc -m32 -c P13_C.c
gcc -m32 P13_ASM.o P13_C.o -o P13.out
if [ -f P13.out ]; then ./P13.out; fi;
Matriz A:
1 2 3
4 5 6
7 8 8
1 2 3
Matriz B:
1 1 1
1 1 1
1 1 1
1 1 1
Matriz C:
2 3 4
5 6 7
8 9 9
2 3 4
if [ -f P13.out ]; then rm P13.out; fi;

Compilation finished at Fri Nov 25 11:55:30
```

Figura 1: Los resultados de la suma estan en la matriz C

Conclusiones y comentarios

A pesar de que los lenguajes de alto nivel son muy practicos para implementar algoritmos más complejos hay algunas situaciones en las que la velocidad de ensamblador es muy util, podriamos pensar que tener uno nos hace perder todas las ventajas del otro pero en los lenguajes de bajo nivel es posible integrarlos, solo tenemos que tener cudado con como el programa maneja las variables a basjo nivel.

Código

Makefile

```
create: asmcode ccode
        gcc -m32 P13_ASM.o P13_C.o -o P13.out
        if [ -f P13.out ]; then ./P13.out; fi;
        if [ -f P13.out ]; then rm P13.out; fi;

asmcode:
        nasm -f elf P13_ASM.asm

ccode:
        gcc -m32 -c P13_C.c
```

P13_ASM.asm

```
;; AUTHOR: Luis Eduardo Galindo Amaya
;;   DATE: 25-11-2022

section .data

section .bss

section .text
global sumaMatrices:

sumaMatrices:
    ;guardar el stack pointer y el stack frame
    push ebp
    mov ebp, esp

    ;guardar los valores de los punteros de memoria
        push ebx
        push edi
        push esi

    ;variables respecto EBP
    ;  +8:  a
    ; +12:  b
    ; +16:  c
    ; +20:  Filas
    ; +24:  Columnas

    mov eax, [ebp+24]           ;columnas
    mov ecx, [ebp+20]           ;filas

    ;calcular el numero de elemntos en la matriz

    mul ecx                     ;EDX:EAX = EAX*ECX
    mov edx, eax                ;edx = bits menos significativos

    mov ecx, edx
    mov esi, [ebp+16]           ;matriz c

    mov eax, [ebp+8]            ;matriz a
    mov edi, 0                  ;iterador

.sumar_A:                       ;-------------------------------
    mov ebx, [eax+edi*4]
    add [esi+edi*4], ebx
    inc edi
    loop .sumar_A

    mov ecx, edx                ;reinicia el loop
    mov eax, [ebp+12]           ;matriz b
```

```
50      mov edi, 0                  ;iterador
51
52  .sumar_B:                       ;-------------------------------
53      mov ebx, [eax+edi*4]
54      add [esi+edi*4], ebx
55      inc edi
56      loop .sumar_B
57
58      ;restaurar los valores de los registros
59      pop ESI
60          pop EDI
61          pop EBX
62
63      ;restaurar el stack pointer y el stack frame
64          mov esp, ebp
65          pop ebp
66          ret
```

P13_C.c

```
1   // AUTHOR: Luis Eduardo Galindo Amaya
2   //   DATE: 25-11-2022
3
4   #include <stdio.h>
5
6   #define FILAS 4
7   #define COLUMNAS 3
8
9   /**
10   * Suma
11   * @param
12   * @return
13   */
14  extern void sumaMatrices(int A[FILAS][COLUMNAS],
15                           int B[FILAS][COLUMNAS],
16                           int C[FILAS][COLUMNAS],
17                           int filas, int columnas);
18
19  /**
20   * Inicializa los valores de una matriz en ceros
21   * @param matriz
22   */
23  void ceros(int[FILAS][COLUMNAS]);
24
25  /**
26   * Imprime los valores de una matriz
27   * @param matriz
28   */
29  void mostrarMatriz(int[FILAS][COLUMNAS]);
30
31  int main(int argc, char *argv[]) {
32
33    int C[FILAS][COLUMNAS];
34    ceros(C);
35
36    int A[FILAS][COLUMNAS] = {
37         {1, 2, 3},
38         {4, 5, 6},
39         {7, 8, 8},
40         {1, 2, 3}
41    };
42
43    int B[FILAS][COLUMNAS] = {
44         {1, 1, 1},
45         {1, 1, 1},
46         {1, 1, 1},
47         {1, 1, 1}
48    };
49
```

```
50    sumaMatrices(A,B,C,FILAS,COLUMNAS);
51
52    puts("Matriz A:");
53    mostrarMatriz(A);
54
55    puts("Matriz B:");
56    mostrarMatriz(B);
57
58    puts("Matriz C:");
59    mostrarMatriz(C);
60
61    return 0;
62  }
63
64  void ceros(int A[FILAS][COLUMNAS]) {
65    for (int i = 0; i < FILAS; i++) {
66      for (int j = 0; j < COLUMNAS; j++)
67        A[i][j] = 0;
68    }
69  }
70
71  void mostrarMatriz(int A[FILAS][COLUMNAS]) {
72    for (int i = 0; i < FILAS; i++) {
73      for (int j = 0; j < COLUMNAS; j++) {
74        printf("%d ", A[i][j]);
75      }
76      printf("\n");
77    }
78  }
```