Práctica 1 Avance 4

Alejandro Cavadavid
Universidad EAFIT
Medellín, Colombia
Email: acadavid@eafit.edu.co

Adolfo Builes
Universidad EAFIT
Medellín, Colombia
Email: abuilesj@eafit.edu.co

Sebastian Castillo
Universidad EAFIT
Medellín, Colombia
Email: scatil1@eafit.edu.co

***Abstract*—Durante esta 4 semanas de trabajo, hemos venido haciendo refinaciones sobre los problemas propuestos, para este informe y gracias a la ayuda brindada por los anteriores, ya tenemos implementados en lenguaje ensamblador operaciones de suma, resta, transpuesta de una matriz y multiplicacin por escalar.**

I. Suma de matrices

A. De la refinación al código

Luego de examinar cada uno de los diagramas de flujo, y llegar al de nivel más bajo, y teniendo en cuenta además lo de mas alto nivel, fue muy fácil pasar a implementar la solución en lenguaje ensamblador, tal vez, en lo que gastamos algo de tiempo fue en la compresión del coprocesador para las operaciones con dobles. Es esta versión estamos trabajando con matrices quemadas en el código, esta son de un tamao de 2 filas y 2 columnas.

B. Implementación

Partiendo del diagrada de la Figura 1 (Fig.1), llegar a la implentacion en lenguaje ensamblador se hizo de una manera muy intuitiva, el siguiente código corresponde a la suma de matrices de tamao flotante.

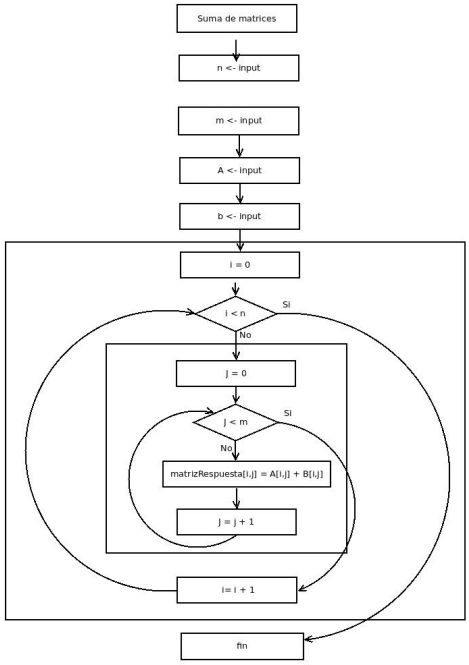

Fig. 1. Diagrama de flujo para la suma de matrices

```
segment .data
matriz1: dq 4.0,1.0,4.0,1.0
matriz2: dq 3.0,1.0,4.0,1.0
 msg2: db ''%1f ,'',10,0

segment .bss
matriz3: resq 10
aux: resd 1
aux2: resd 1
aux3: resq 1

segment .text
    global main
    extern printf
main:
    mov    eax,matriz1
    mov    ebx,matriz2
    mov    edx,matriz3
    mov    ecx,10
    jmp    lp

lp:
    fld    qword [eax]
    fld    qword [ebx]
    faddp  st1
    fstp   qword [edx]
    add    eax,8
    add    ebx,8
    add    edx,8
    loop   lp
    mov    ecx,10
    mov    eax,matriz3
    call   printm
    jmp    exit

printm:
    mov    [aux],eax
    mov    [aux2],ecx
    push   dword [eax+4]
    push   dword [eax]
    push   dword msg2
```

```
    call    printf
    add     esp,12
    mov     eax,[aux]
    mov     ecx,[aux2]
    add     eax,8
    loop    printm

exit:
    mov     eax,1
    mov     ebx,0
    int     0x80
```

II. Resta de matrices

A. Implementación

Al igual que para la suma, teniendo el diagrama (Fig.2),

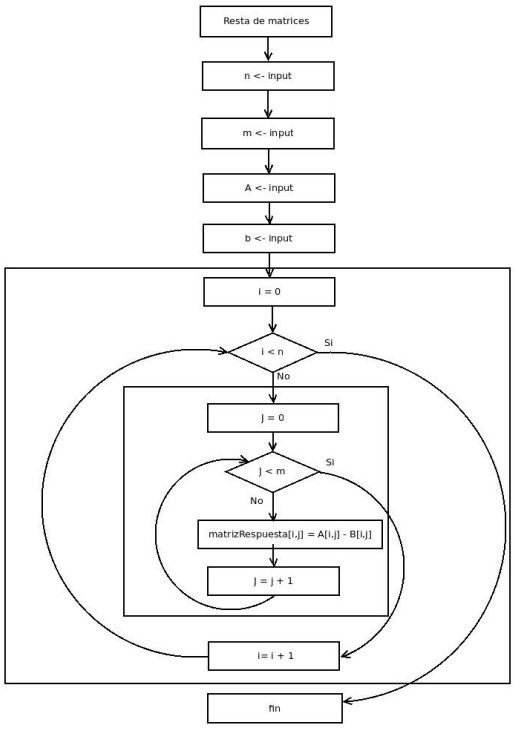

Fig. 2. Diagrama de flujo para la resta de matrices

es inmediato el paso al código ensamblador, además es muy parecido a la suma.

```
segment .data
matriz1: dq 4.0,1.0,4.0,1.0
matriz2: dq 3.0,0.0,3.0,0.0
 msg2: db ''%lf  ,'',10,0

segment .bss
matriz3: resq 10
aux: resd 1
aux2: resd 1
aux3: resq 1

segment .text
    global main
    extern printf
main:
    mov     eax,matriz1
    mov     ebx,matriz2
    mov     edx,matriz3
    mov     ecx,10
    jmp     lp

lp:
    fld     qword [eax]
    fld     qword [ebx]
    fsubp   st1
    fstp    qword [edx]
    add     eax,8
    add     ebx,8
    add     edx,8
    loop    lp
    mov     ecx,10
    mov     eax,matriz3
    call    printm
    jmp     exit

printm:
    mov     [aux],eax
    mov     [aux2],ecx
    push    dword [eax+4]
    push    dword [eax]
    push    dword msg2
    call    printf
    add     esp,12
    mov     eax,[aux]
    mov     ecx,[aux2]
    add     eax,8
    loop    printm

exit:
    mov     eax,1
    mov     ebx,0
    int     0x80
```

III. Multiplicación de Matriz por escalar

A. Implementación

El diagrama (Fig.3), nos muestra el diagrama de flujo de ultimo nivel para esta operación, el siguiente código corresponde a la multiplicación de una matriz por un escalar, en este ejemplo estamos usando un escalar de tipo doble.

```
segment .data
matriz1: dq 4.0,1.0,4.0,1.0
escalar: dq 2.0
 msg2: db ''%lf  ,'',10,0
```

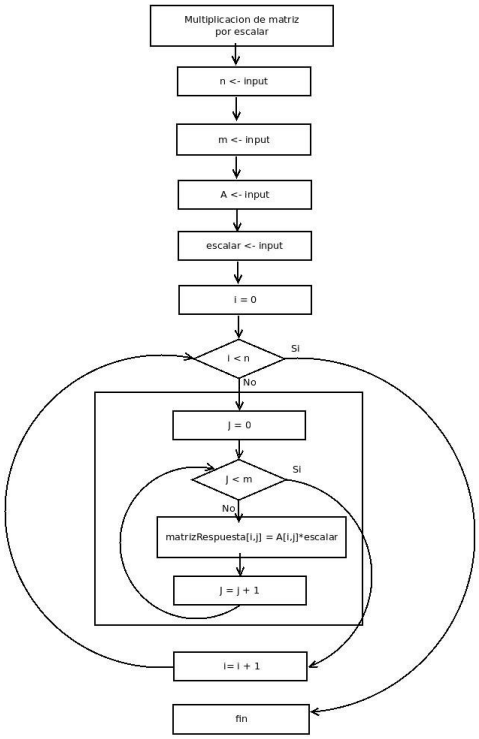

Fig. 3. Diagrama de flujo para la multiplicacion de matriz por escalar

```
segment .bss
matriz3: resq 10
aux: resd 1
aux2: resd 1
aux3: resq 1

segment .text
    global main
    extern printf
main:
    mov    eax, matriz1
    mov    edx, matriz3
    mov    ecx,10
    jmp    lp

lp:
    fld    qword [eax]
    fld    qword [escalar]
    fmulp  st1
    fstp   qword [edx]
    add    eax,8
    add    edx,8
    loop   lp
    mov    ecx,10
    mov    eax, matriz3
    call   printm
    jmp    exit

printm:
    mov    [aux], eax
    mov    [aux2], ecx
    push   dword [eax+4]
    push   dword [eax]
    push   dword msg2
    call   printf
    add    esp,12
    mov    eax,[aux]
    mov    ecx,[aux2]
    add    eax,8
    loop   printm

exit:
    mov    eax,1
    mov    ebx,0
    int    0x80
```

IV. Matriz transpuesta

A. Diagramas de flujo

Incluimos los diagramas de flujo a tres niveles, para el proceso de obtener la matriz transpuesta.

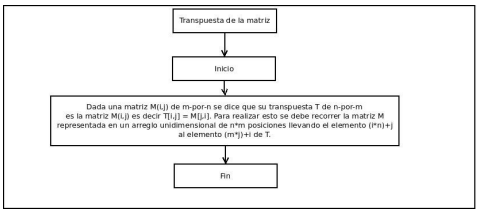

Fig. 4. Diagrama de flujo a nivel general

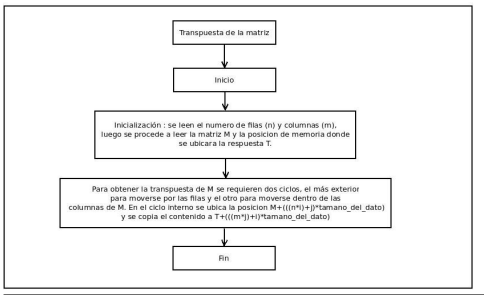

Fig. 5. Diagrama de flujo a nivel 1

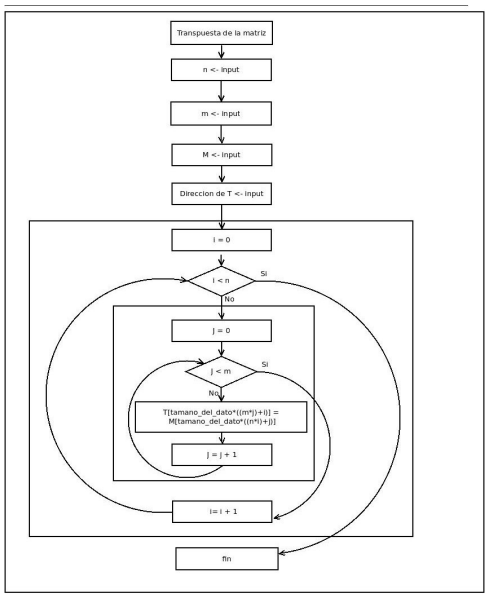

Fig. 6. Diagrama de flujo a nivel 2

V. Conclusiones

Gracias a las abstracciones hechas en las entregas anteriores, llegar a la impletación en lenguaje ensamblador fue mucho más fácil, aun nos queda faltando la multiplicación de matrices.... hasta la próxima edición ! .

VI. Bibliografía

[1] http://www.drpaulcarter.com/pcasm/pcasm-book-pdf.zip, Visitado Viernes 19 de Marzo.