



**Universidad
Nacional de
General
Sarmiento**

Organización del Computador II

TP N°1: FPU - Resolvente

- Ejercicios obligatorios :
 - Gestión de Memoria (4 , 6 y 7)
 - FPU (7)

Docentes:

- Alles, Sergio Gastón
- Dominguez, Ariel

Alumno:

- Diaz, Jonatan. - 35208154/2008 - je.d.10@hotmail.com

Luego de Compilar con el Makefile, los archivos “resolvente.asm” y “resolvente.c” obtendremos el programa “tp1”.

Para utilizarlo, **debemos pasarle los parámetros A, B y C, en el mismo comando de invocación del programa** (separando cada parámetro por un carácter espacio).

ejemplo “./tp1 -1 2 3”

desde C controlamos las restricciones que tiene el programa, como ingresar el valor 0 para el parámetro A o que el cálculo del valor discriminante sea invalido (osea no pertenezca al conjunto de números reales).

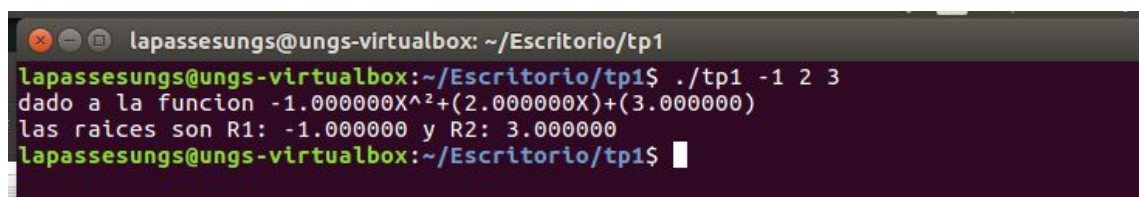
luego tendremos la función:

```
void resolvente(float a, float b, float c, float* ra, float* rb);
```

donde los punteros ra y rb, serán utilizados para que la función almacene los resultados desde assembler.

A continuación mostramos capturas de algunos ejemplos de ejecución.

A) utilizando números enteros



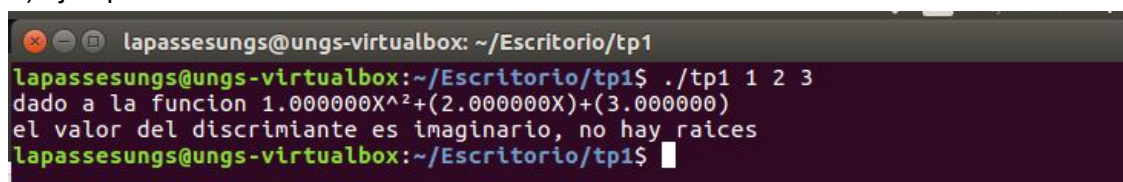
```
lapassesungs@ungs-virtualbox: ~/Escritorio/tp1
lapassesungs@ungs-virtualbox:~/Escritorio/tp1$ ./tp1 -1 2 3
dado a la funcion -1.000000X^2+(2.000000X)+(3.000000)
las raices son R1: -1.000000 y R2: 3.000000
lapassesungs@ungs-virtualbox:~/Escritorio/tp1$
```

B) utilizando números reales (punto flotante de precisión simple).



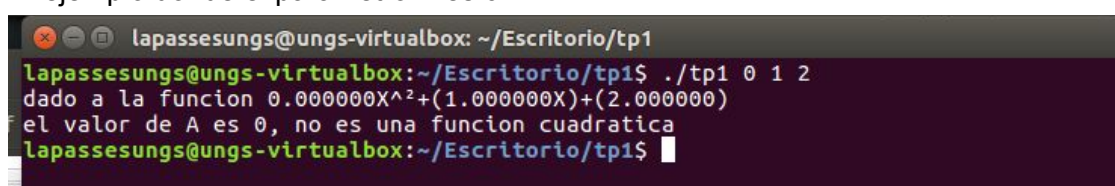
```
lapassesungs@ungs-virtualbox: ~/Escritorio/tp1
lapassesungs@ungs-virtualbox:~/Escritorio/tp1$ ./tp1 1.1 4.2 3.3
dado a la funcion 1.100000X^2+(4.200000X)+(3.300000)
las raices son R1: -1.106204 y R2: -2.711978
lapassesungs@ungs-virtualbox:~/Escritorio/tp1$
```

C) ejemplo donde el discriminante no es un número real



```
lapassesungs@ungs-virtualbox: ~/Escritorio/tp1
lapassesungs@ungs-virtualbox:~/Escritorio/tp1$ ./tp1 1 2 3
dado a la funcion 1.000000X^2+(2.000000X)+(3.000000)
el valor del discriminante es imaginario, no hay raices
lapassesungs@ungs-virtualbox:~/Escritorio/tp1$
```

D- ejemplo donde el parámetro A es 0.



```
lapassesungs@ungs-virtualbox: ~/Escritorio/tp1
lapassesungs@ungs-virtualbox:~/Escritorio/tp1$ ./tp1 0 1 2
dado a la funcion 0.000000X^2+(1.000000X)+(2.000000)
el valor de A es 0, no es una funcion cuadratica
lapassesungs@ungs-virtualbox:~/Escritorio/tp1$
```

Jonatan Diaz

Organización del Computador 2 - 2do semestre 2020

Trabajo Práctico 1: **Resolución de Ejercicios Obligatorios**

Práctica Gestión de memoria

4). Usted dispone de un dispositivo que utiliza un sistema de paginación con direcciones virtuales de 32 bits , 1 GB de memoria física y frames de 4 KB. ¿Cuántas entradas posee la tabla de páginas en cada uno de estos esquemas? (Obligatorio)

C. Si se utiliza un sistema de paginación de un solo nivel.

D. Si se utiliza un sistema de tabla de paginación invertido

RTA: 4

Sistema de direcciones virtuales: 32 bits, 2^{32}

Memoria Física: 1 GB , 2^{30}

Tamaño de Frames de 4 KB , 2^{12}

C) Cantidad de entradas que posee la tabla de páginas de un sistema de paginación de un solo nivel:

Sistema de dirección virtuales / Tamaño de Frames = $2^{32} / 2^{12} = 2^{20}$

D) Cantidad de entradas que posee la tabla de páginas de un sistema de paginación invertida:

Cantidad de memoria física/ Tamaño de Frames = $2^{30} / 2^{12} = 2^{18}$

6). (Obligatorio)

Se encuentran cargados los siguientes registros de segmento para el proceso P1:

CS -> base address: 1000 , limit: 800

DS -> base address: 500 , limit: 250

SS -> base address: 4000 , limit: 200

Por otro lado, el proceso lee las siguientes direcciones lógicas:

A. La dirección 0 para el segmento de datos.

B. La dirección 550 para el segmento de código.

C. La dirección 100 para el segmento de stack.

D. La dirección 4000 para el segmento de stack.

Calcular la dirección física asociada a cada uno de estos.

RTA 6:

A) la dirección física 0 para el DS es 500

B) la dirección física 550 para el CS es 1550

C) la dirección física 100 para el SS es 4100

D) la dirección física 4000 para el SS no existe, xq el límite del SS es de 200.

7. (Obligatorio) Dado el siguiente esquema, indicar el estado final de la cache TLB y tabla de páginas. También indicar la cantidad de rafagas utilizadas en cada secuencia.

Las páginas requeridas son las siguientes:

A). Pagina 1, Pagina 2, Pagina 6, Pagina 3, Pagina 2, Pagina 1, Pagina 4, Pagina 5

B). Pagina 6, Pagina 1, Pagina 3, Pagina 2, Pagina 4, Pagina 5, Pagina 4, Pagina 6

RTA 7:

A)

TLB

Página	Frame
5	1
4	3

Tabla de Páginas

Página	Frame	Valid
1	2	v
2	4	v
3		i
4	3	v
5	1	v
6		i

Memoria Principal

frame 1	frame 2	frame 3	frame 4
5	1	4	2

B)

TLB

Página	Frame
4	3
6	1

Tabla de Páginas

Página	Frame	Valid
1		i
2	4	v
3		i
4	3	v
5	2	v
6	1	v

Memoria Principal

frame 1	frame 2	frame 3	frame 4
6	5	4	2

Práctica FPU:

7)

(Obligatorio) Escriba una función en assembler IA-32 que reciba un puntero a un Vector de números de punto flotante de precisión simple y calcule la suma. El prototipo de la función es:

```
float suma_vf(float *vector, int cantidad);
```

archivo “suma_vf.asm”

```
global suma_vf
```

```
section .data
```

```
p0 dd 0.0
```

```
recicler dd 0.0
```

```
section .text
```

```
suma_vf:
```

```
    push ebp
```

```
    mov ebp, esp
```

```
    mov ebx, [ebp+8] ; puntero al vector
```

```
    mov ecx, [ebp+12] ; cantidad de numeros en el vector
```

```
    fld dword [p0]
```

```
comparar:
```

```
    cmp ecx, 0
```

```
    jle fin
```

```
    fld dword [ebx]
```

```
    fadd st1, st0
```

```
    fstp dword [recicler] ; Limpiar registro st0
```

```
    sub ecx, 1
```

```
    add ebx, 4
```

```
    jmp comparar
```

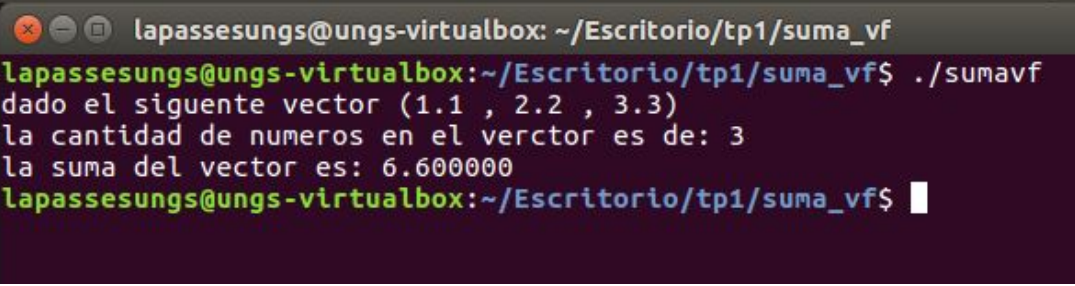
```
fin:
```

```
    fld dword [eax]
```

```
    mov esp, ebp
```

```
    pop ebp
```

```
    ret
```



```
lapassesungs@ungs-virtualbox: ~/Escritorio/tp1/suma_vf
lapassesungs@ungs-virtualbox:~/Escritorio/tp1/suma_vf$ ./sumavf
dado el siguiente vector (1.1 , 2.2 , 3.3)
la cantidad de numeros en el vector es de: 3
la suma del vector es: 6.600000
lapassesungs@ungs-virtualbox:~/Escritorio/tp1/suma_vf$
```

si bien el vector está hardcodedo en C, la función programada en ASM funciona correctamente.