



CENTRO DE CIENCIAS BÁSICAS
DEPARTAMENTO DE SISTEMAS ELECTRÓNICOS
LENGUAJE ENSAMBLADOR
7° "A"

OPERACIONES CON FLOTANTES

Profesor: Cristian Jael Mejía Aguirre

Alumno: Joel Alejandro Espinoza Sánchez

Fecha de Entrega: Aguascalientes, Ags., 1 de noviembre de 2021

Operaciones con Flotantes

Objetivo: Realizar operaciones con flotantes en lenguaje ASM.

Desarrollo: Lo más complicado en esta práctica fue la impresión del valor flotante. Este mismo elemento causó que para el caso de esta práctica realizara tres archivos:

El primero sobre la suma es el siguiente:

```
%include "io64.inc"
```

```
section .data
    val: dq 3.1416
    val2: dq 2.7128
    ten: dd 10
```

```
section .bss
    leftdigits: resd 1
    rightdigits: resb 100
    temp: resd 1
    control_word: resw 1
```

```
section .text
global CMAIN
CMAIN:
```

```
    mov ebp, esp ; Correct debugging
```

```
    fld qword[val]
    fadd val, val2
```

```
    ; Modificar modo de redondeo
    fstcw [control_word]
    mov ax, [control_word]
    or ah, 0b00001100 ; Modo de redondeo: Truncar
    mov [temp], ax
    fldcw [temp] ; Cargar nuevo modo de redondeo
```

```
    fist dword [leftdigits] ; Guardar parte entera
    fisub dword [leftdigits] ; Limpiar parte entera
```

```
    ; Cargar 10 y mover a st(1)
    fild dword [ten]
    fxch
```

```
    ; Isolatear dígitos de la parte fraccionaria y guardarlos ASCII
```

```

mov edi, rightdigits ; Apuntador al buffer ASCIIIZ
.get_fractional:
fmul st0, st1 ; On dígito decimal a entero
fist dword [temp] ; Guardar dígito
fisub dword [temp] ; Limpia parte entera
mov al, byte [temp] ; Carga dígito
or al, 0x30 ; Convierte a ASCII
mov byte [edi], al ; Guarda en rightdigits
add edi, 1 ; Incrementa el apuntador al string
fxam ; st0 == 0.0?
fstsw ax
fwait
sahf
jnz .get_fractional ; No: Itera
mov byte [edi], 0 ; Multi-termination para ASCIIIZ

; Limpia FPU
ffree st0 ; Vacía st(0)
ffree st1 ; Vacía st(1)
fldcw [control_word] ; Restaura el modo de redondeo anterior

PRINT_DEC 4, leftdigits
PRINT_CHAR '.' ; Punto decimal
PRINT_STRING rightdigits

xor eax, eax
ret

```

El segundo sobre la resta es el siguiente:

```
%include "io64.inc"
```

```
section .data
```

```

val: dq 3.1416
val2: dq 2.7128
ten: dd 10

```

```
section .bss
```

```

leftdigits: resd 1
rightdigits: resb 100
temp: resd 1
control_word: resw 1

```

```
section .text
```

```
global CMAIN
```

```
CMAIN:
```

```
mov ebp, esp ; Correct debugging
```

```

fld qword[val]
fsub val, val2

; Modificar modo de redondeo
fstcw [control_word]
mov ax, [control_word]
or ah, 0b00001100 ; Modo de redondeo: Truncar
mov [temp], ax
fldcw [temp] ; Cargar nuevo modo de redondeo

fist dword [leftdigits] ; Guardar parte entera
fisub dword [leftdigits] ; Limpiar parte entera

; Cargar 10 y mover a st(1)
fild dword [ten]
fxch

; Isolatear dígitos de la parte fraccionaria y guardarlos ASCII
mov edi, rightdigits ; Apuntador al buffer ASCIIZ
.get_fractional:
fmul st0, st1 ; On dígito decimal a entero
fist dword [temp] ; Guardar dígito
fisub dword [temp] ; Limpia parte entera
mov al, byte [temp] ; Carga dígito
or al, 0x30 ; Convierte a ASCII
mov byte [edi], al ; Guarda en rightdigits
add edi, 1 ; Incrementa el apuntador al string
fxam ; st0 == 0.0?
fstsw ax
fwait
sahf
jnz .get_fractional ; No: Itera
mov byte [edi], 0 ; Multi-termination para ASCIIZ

; Limpia FPU
ffree st0 ; Vacía st(0)
ffree st1 ; Vacía st(1)
fldcw [control_word] ; Restaura el modo de redondeo anterior

PRINT_DEC 4, leftdigits
PRINT_CHAR '.' ; Punto decimal
PRINT_STRING rightdigits

xor eax, eax
ret

```

El tercer sobre la multiplicación es el siguiente:

```
%include "io64.inc"
```

```
section .data
```

```
    val: dq 3.1416
```

```
    val2: dq 2.7128
```

```
    ten: dd 10
```

```
section .bss
```

```
    leftdigits: resd 1
```

```
    rightdigits: resb 100
```

```
    temp: resd 1
```

```
    control_word: resw 1
```

```
section .text
```

```
global CMAIN
```

```
CMAIN:
```

```
    mov ebp, esp ; Correct debugging
```

```
    fld qword[val]
```

```
    fmul val, val2
```

```
    ; Modificar modo de redondeo
```

```
    fstcw [control_word]
```

```
    mov ax, [control_word]
```

```
    or ah, 0b00001100 ; Modo de redondeo: Truncar
```

```
    mov [temp], ax
```

```
    fldcw [temp] ; Cargar nuevo modo de redondeo
```

```
    fist dword [leftdigits] ; Guardar parte entera
```

```
    fisub dword [leftdigits] ; Limpiar parte entera
```

```
    ; Cargar 10 y mover a st(1)
```

```
    fild dword [ten]
```

```
    fxch
```

```
    ; Isolatear dígitos de la parte fraccionaria y guardarlos ASCII
```

```
    mov edi, rightdigits ; Apuntador al buffer ASCIIIZ
```

```
    .get_fractional:
```

```
    fmul st0, st1 ; On dígito decimal a entero
```

```
    fist dword [temp] ; Guardar dígito
```

```
    fisub dword [temp] ; Limpia parte entera
```

```
    mov al, byte [temp] ; Carga dígito
```

```
    or al, 0x30 ; Convierte a ASCII
```

```
    mov byte [edi], al ; Guarda en rightdigits
```

```

add edi, 1 ; Incrementa el apuntador al string
fxam ; st0 == 0.0?
fstsw ax
fwait
sahf
jnz .get_fractional ; No: Itera
mov byte [edi], 0 ; Multi-termination para ASCIIIZ

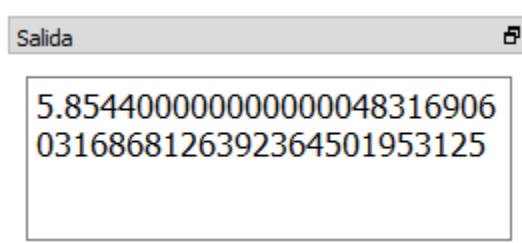
; Limpia FPU
ffree st0 ; Vacía st(0)
ffree st1 ; Vacía st(1)
fldcw [control_word] ; Restaura el modo de redondeo anterior

PRINT_DEC 4, lefthdigits
PRINT_CHAR '.' ; Punto decimal
PRINT_STRING righthdigits

xor eax, eax
ret

```

El resultado del primer programa es el siguiente:

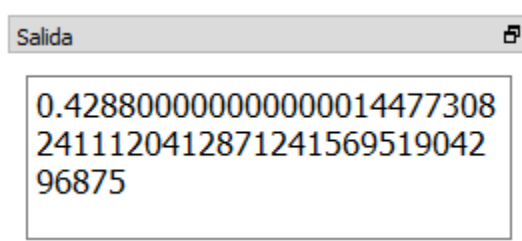


```

5.854400000000000048316906
0316868126392364501953125

```

El resultado del segundo programa es el siguiente:

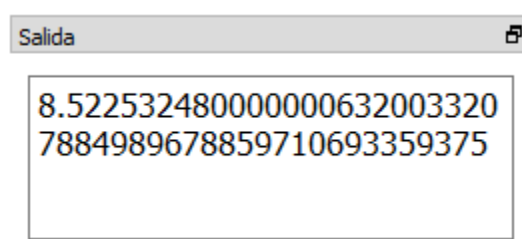


```

0.428800000000000014477308
2411120412871241569519042
96875

```

El resultado del tercer programa es el siguiente:



```

8.5225324800000000632003320
7884989678859710693359375

```

Conclusión: Ahora con este programa obtenemos la perspectiva de cómo se necesita realizar paso a paso algún procedimiento, pues simplemente la lectura del número en punto flotante es muy larga y que puede ser una parte muy amplia del código, sin embargo, nos muestra cómo pensar el algoritmo paso a paso para poder implementarlo.