Arquitectura de Computadoras
LAB 4

Docente: Jorge Gonzales Reaño

Integrantes:

Nombre y apellidos	Código	Correo
Jose Leandro Machaca	202110202	jose.machaca.s@utec.edu,pe
Diego Pacheco Ferrell	202010159	diego.pacheco@utec.edu.pe
Dimael Rivas Chavez	202110307	dimael.rivas@utec.edu.pe

Exercise 1: Fibonacci Numbers

Write an assembly language program to calculate the 13th Fibonacci number. To get you started, here is a template for the body of the assembly program that you should write:

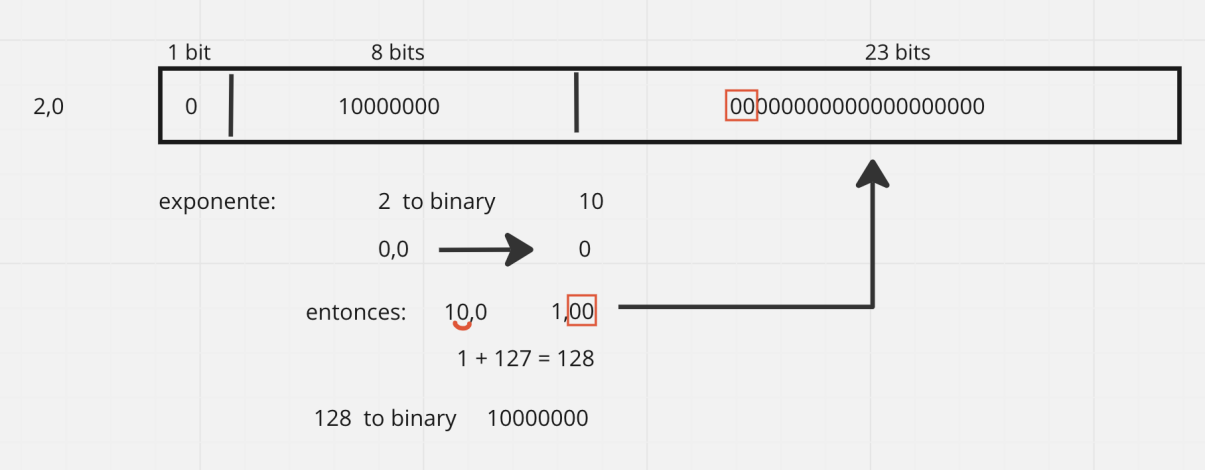
```
/************* CODE SECTION *************/
.text @ the following is executable assembly
@ Ensure code section is 4-byte aligned:
.balign 4
@ main is the entry point and must be global
.global main
B main @ begin at main
/************* MAIN SECTION *************/
main:
    MOV R1, #0
    MOV R2, #1     @ Cargar los primeros dos números de Fibonacci

    BL loop

    B done

loop:
   PUSH {R2, R1, LR}
   CMP R0, #1
   BGT else
   ADD SP, SP, #12
   MOV R1, R3
   MOV PC, LR

else:
   SUB R0, R0, #1
   POP {R1, R2, LR}
   ADD R3, R2, R1
   MOV R1,
R2
   MOV R2, R3
   B LOOP

done:
```

Exercise 2: Floating Point Addition

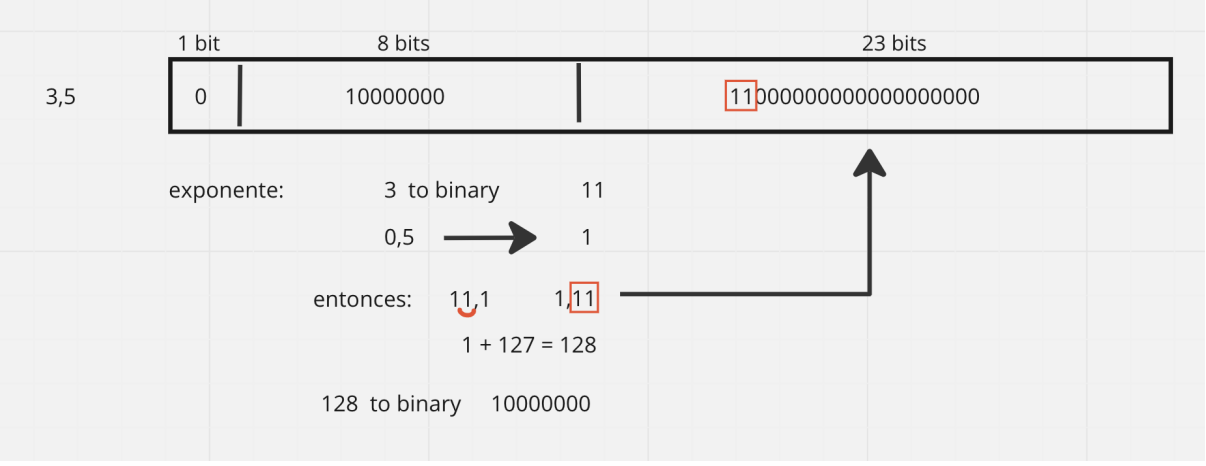
Hand Analysis

1.0 = 0 01111111 00000000000000000000000 = $3F800000_{16}$

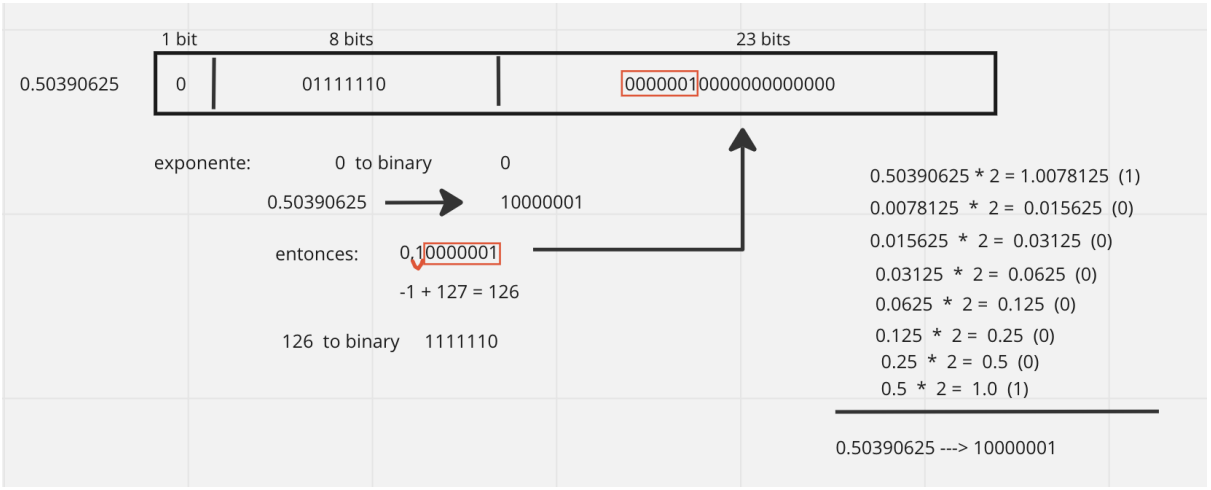
2.0 = 0 1000000 00000000000000000000000 = 20000000_{16}



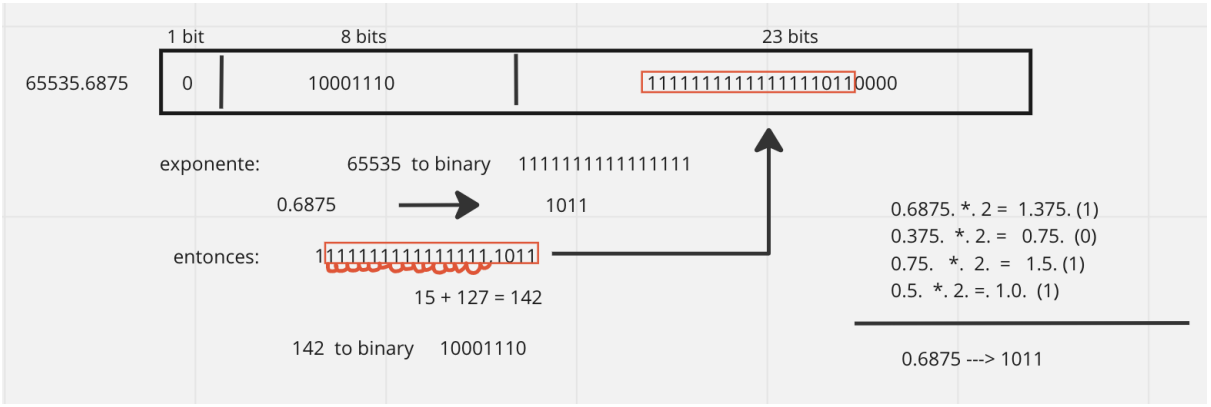
3.5 = 0 1000000 11000000000000000000000 = 20600000_{16}



0.50390625 = 0 01111110 00000010000000000000000 = $3F010000_{16}$



65535.6875 = 0 10001110 11111111111111110110000= $477FFFB0_{16}$



P2:

Suma de 2 numeros flotantes

```
.global _start
_start:

/*Input:
r0 = address del primer numero
```

```
r1 = address del segundo numero
Output:
r2 = La suma
*/
    LDR    r3, [r0]      //r3 = Primer numero
    LDR    r4, [r1]      //r4 = Segundo numero
       MOV r9 , #0 //Direccion a guardar el resultado
//Compare exponent values
       LSR r5,r3, #23 // Exponente del primer numero
       LSR r6,r4, #23 // Exponente del segundo numero

//Mantisa
       MOV r7,#1
       LSL r7,r7,#24 //añadir mas 1
//Extrayendo la primera mantisa
       ROR r3,r3,#23
       LSR r3,r3,#9
       ORR r3, r3, r7
//Extrayendo la segunda mantisa
       ROR r4,r4,#23
       LSR r4,r4,#9
       ORR r4, r4, r7

//Alinear Mantisas
       SUBS r7, r5,r6 //Diferencia entre exp1-exp2

       BLE primercaso // Exponente del primer numero <= exponente del segundo numero
       //Caso en que Exp1 > Exp2
       FOR:
              LSR r4,r4,#1
              SUB r7,r7,#1
              CMP r7,#0
              BNE FOR
       ADD r8, r3,r4 //Sumar las mantisas
       //Ocultando el 1 de la suma de las mantisas
       ROR r8, r8, #23
       LSR r8,r8,#9

       //Numero a guardar
       //exponente
       LSL r5, r5, #23
       ORR r2,r5,r8
       STR r2, [r9]
       primercaso:
       //Caso en que Exp1 <= Exp2
       SUB r7, r6,r5
       FOR:
              LSR r4,r4,#1
              SUB r7,r7,#1
              CMP r7,#0
              BNE FOR
       ADD r8, r3,r4 //Sumar las mantisas
       //Ocultando el 1 de la suma de las mantisas
       ROR r8, r8, #23
       LSR r8,r8,#9
```

```
        //Numero a guardar
        //exponente
        LSL r6, r6, #23
        ORR r2,r6,r8
        STR r2, [r9]
```