Fibonacci

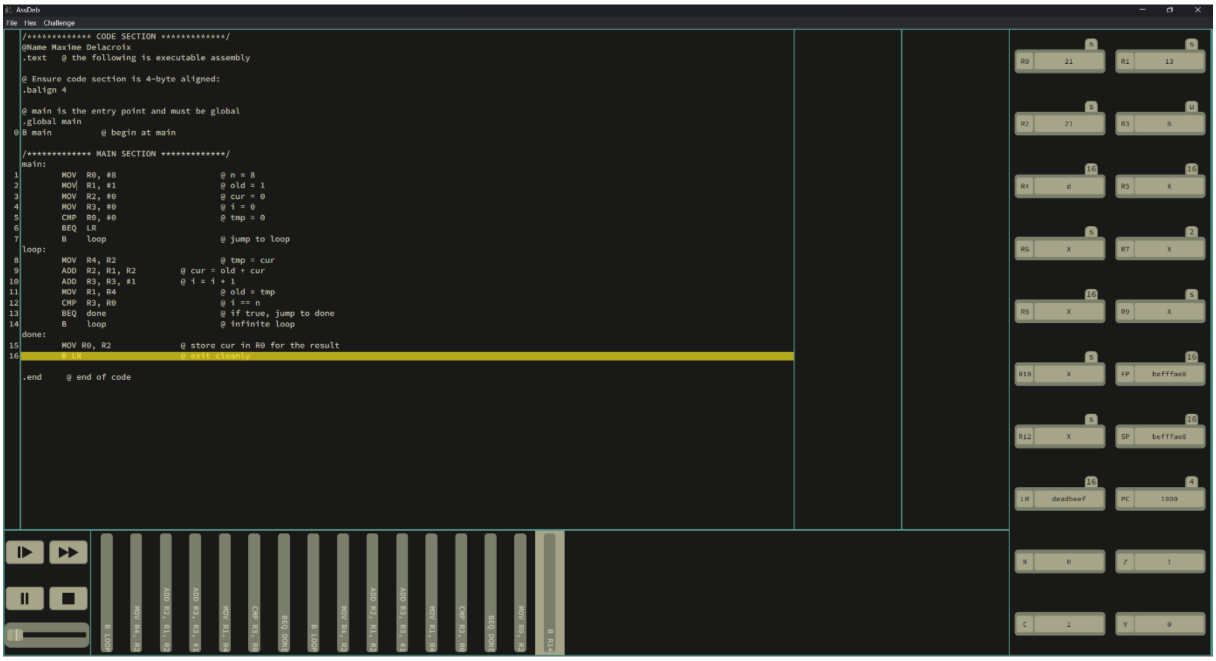
1. Homework_5_fib.s

```
/************* CODE SECTION *************/
@Name Maxime Delacroix
.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  R0, #8         @ n = 8
    MOV  R1, #1         @ old = 1
    MOV  R2, #0         @ cur = 0
    MOV  R3, #0         @ i = 0
    CMP  R0, #0         @ tmp = 0
    BEQ  LR
    B    loop           @ jump to loop
loop:
    MOV  R4, R2         @ tmp = cur
    ADD  R2, R1, R2     @ cur = old + cur
    ADD  R3, R3, #1     @ i = i + 1
    MOV  R1, R4         @ old = tmp
    CMP  R3, R0         @ i == n
    BEQ  done           @ if true, jump to done
    B    loop           @ infinite loop
done:
    MOV R0, R2          @ store cur in R0 for the result
    B LR            @ exit cleanly

.end     @ end of code
```

2. Result of fib(8)

The result of fib(8) = 21 = 0x15

3. AssDeb



Floating point

1. Hand analysis

a) 2 = 0 1000 0000 0000 0000 0000 0000 0000 000 = 0x4000 0000

b) 3,5 = 0 1000 0000 1100 0000 0000 0000 0000 000 = 0x4060 0000

c) 0,50390625 = 0 0111 1110 0000 0010 0000 0000 0000 000 = 0x3F01 0000

d) 65535.6875 = 0 1000 1110 1111 1111 1111 1111 0110 000 = 0x477F FFB0

e) 65536.19 = 0 1000 1111 0000 0000 0000 0000 0011 000 = 0x4780 0018

2. Homework_5_fp.s

```
/************* CODE SECTION *************/
@Name Maxime Delacroix
@Floating Point Addition
.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

/************* FPNUMS *****************/
@ These addresses contain the two fp numbers to be added

fpNum0:  .word   0x40600000
fpNum1:  .word   0x40400000


/************* MAIN SECTION *************/
main:
    ldr r0, fpNum0      @ r0 = fpNum0
    ldr r1, fpNum1      @ r1 = fpNum1

    @ Shift the numbers to the right by 23 bits to
    @ get the exponents
    lsr r2, r0, #23     @ r2 = r0 >> 23
    lsr r3, r1, #23     @ r3 = r1 >> 23

    @ Mask the exponent to get the fraction
    lsl r4, r0, #8      @ r4 = r0 << 8
    lsl r5, r1, #8      @ r5 = r1 << 8
    orr r4, r4, #0x80000000 @ Adding the leading 1
    orr r5, r5, #0x80000000 @ Adding the leading 1

    @ Substract and raise the flags
    subs r6, r2, r3     @ r6 = r2 - r3

    @ Compare the exponent and store the bigger one in r7
    movge r7, r2        @ r7 = r2 if r2 >= r3
    movlt r7, r3        @ r7 = r3 if r2 < r3

    @ Shift the fraction
    lsrgt r5, r5, r6    @ r5 = r5 >> r6 if r2 > r3
    sublt r6, r3, r2    @ r6 = r3 - r2 if r2 < r3
    lsrlt r4, r4, r6    @ r4 = r4 >> r6 if r2 < r3

    @ Add the fraction
    adds r4, r4, r5     @ r4 = r4 + r5
    @ r4 hold now the sum of the fraction

    @ Normalize the fraction
    lsrcs r4, r4, #1    @ r4 = r4 >> 1 if r4 overflowed
    addcs r7, r7, #1    @ r7 = r7 + 1 if r4 overflowed

    @ Rounding the result

    @ Strip the leading 1 off the resulting mantissa
    @ and merge the signe, exponent and fraction bits
    lsl r7, r7, #23     @ r7 = r7 << 23
    lsl r4, r4, #1      @ r4 = r4 << 1
    lsr r4, r4, #9      @ r4 = r4 >> 9
    orr r0, r7, r4      @ r0 = r7 | r4

    b lr
.end     @ end of code
```

3. Result of additions

1.0 + 1.0 = 2.0 (0x3F80 0000 + 0x3F80 0000 = 0x4000 0000)

2.0 + 1.0 = 3.0 (0x4000 0000 + 0x3F80 0000 = 0x4040 0000)

3.0 + 3.5 = 6.5 (0x4040 0000 + 0x4060 0000 = 0x40D0 0000)

0.50390625 + 65535.6875 = 65536.19 (0x3F01 0000 + 0x477F FFB0 = 0x4780 0018)

4. AssDeb