

1 Fibonacci code

```

/***** CODE SECTION *****/
@Name Théo Denis
.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

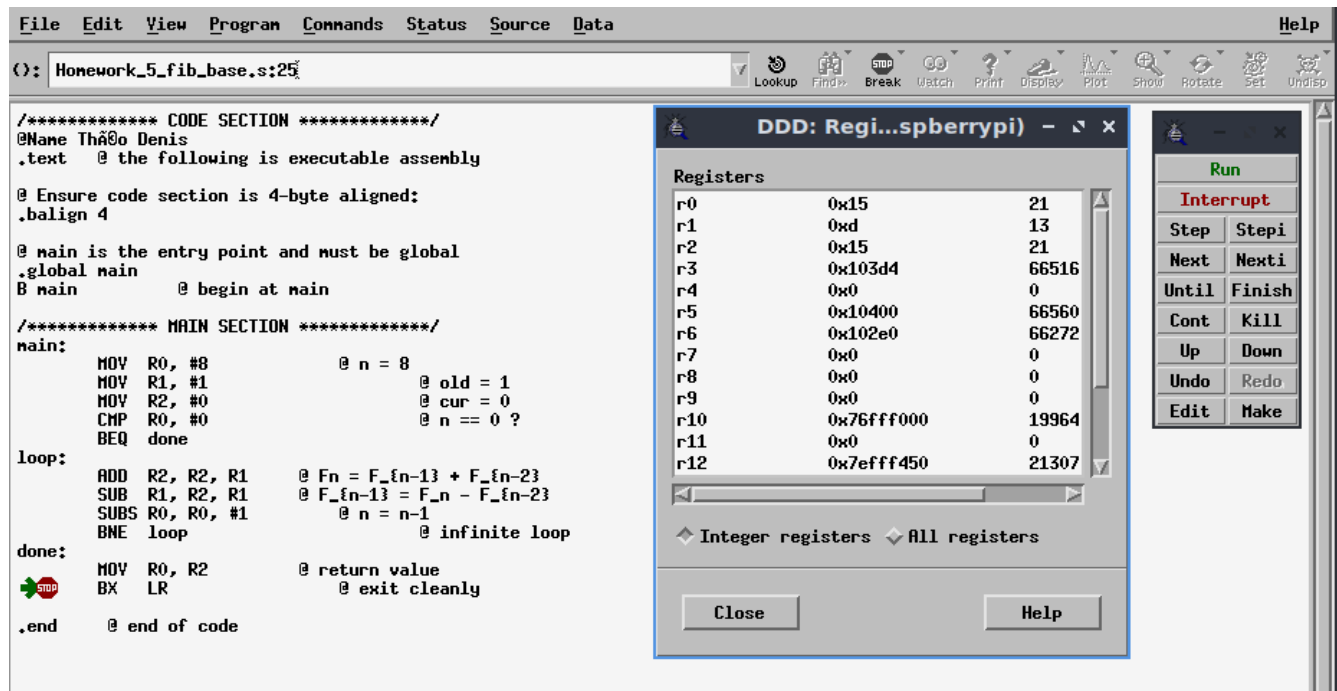
/***** MAIN SECTION *****/
main:
    MOV    R0, #8        @ n = 8
    MOV    R1, #1        @ old = 1
    MOV    R2, #0        @ cur = 0
    CMP    R0, #0        @ n == 0 ?
    BEQ    done
loop:
    ADD    R2, R2, R1     @ Fn = F_{n-1} + F_{n-2}
    SUB    R1, R2, R1     @ F_{n-1} = F_n - F_{n-2}
    SUBS   R0, R0, #1     @ n = n-1
    BNE    loop          @ infinite loop
done:
    MOV    R0, R2        @ return value
    BX     LR            @ exit cleanly

.end                @ end of code

```

The value for fib(8) obtained with the code was 21 or 0x15 in hexadecimal.

2 DDD screenshot fib



3 Floating point hand analysis

Representation in IEEE754 single point precision

Decimal	Binary	Hexadecimal
2.0	0 10000000 000000000000000000000000	0x40000000
3.5	0 10000000 110000000000000000000000	0x40600000
0.50390625	0 01111110 000000100000000000000000	0x3F010000
65535.6875	0 10001110 11111111111111110110000	0x477FFFB0
0.50390625 + 65535.6875	0 10001111 0000000000000000000011000	0x47800018

The last number was truncated at 65536.19.

4 Floating point addition code

```

/***** CODE SECTION *****/
@Name Théo Denis
@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    0x3f010000
fpNum1:    .word    0x477fffb0

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

@ Your Code goes here:
@ masking and shifting down exponents
    LSR r2, r0, #23      @ r2[7:0] = exponent of r0
    LSR r3, r1, #23      @ r3[7:0] = exponent of r1

@ masking and appending leading 1
/* can be perform with orr-ing with leading 1 and then shifting
left to delete exponent and sign bits */
    ORR r4, r0, #0x00800000
    ORR r5, r1, #0x00800000
    LSL r4, r4, #8        @ r4[31:8] = mantissa of r0
    LSL r5, r5, #8        @ r5[31:8] = mantissa of r1

@ comparing exponents and setting based on value
    SUBS r6, r2, r3        @ r6 = r2 - r3 if r2 > r3
    MOVGE r7, r2          @ r7 = r2 if r2 >= r3
    MOVL T r7, r3          @ r7 = r3 if r2 < r3
```

@ right shifting mantissa by difference of exponents

```
LSRGT r5, r5, r6      @ r5 = r5 >> r6
SUBLT r6, r3, r2      @ r6 = r3 - r2 if r3 > r2
LSRLT r4, r4, r6      @ r4 = r4 >> r6
```

@ summing mantissas

```
ADDS r8, r4, r5      @ r8 = r4 + r5
```

@ normalize the result in case of overflowing

```
LSRCS r8, r8, #1      @ r8 = r8 >> 1
ADDCS r7, r7, #1      @ r7 += 1
```

@ rounding result, do nothing because trunc is fine

@ strip leading 1 and merge everything

@ R8 = mantissa, R7 = exponent, sign = 0 because work with positive

```
LSL r8, r8, #1      @ r8 = r8 << 1 strip leading 1
LSR r8, r8, #9      @ r8[22:0] = mantissa
LSL r7, r7, #23      @ r7[31:23] = biased exponent
ORR r0, r7, r8      @ r0 = sum in IEEE754
```

```
bx lr
```

```
.end      @ end of code
```

5 Results of addition

Decimal	Binary	Hexadecimal
2.0	0 10000000 000000000000000000000000	0x40000000
3.5	0 10000000 110000000000000000000000	0x40600000
0.50390625	0 01111110 000000100000000000000000	0x3F010000
65535.6875	0 10001110 111111111111111110110000	0x477FFFB0
0.50390625 + 65535.6875	0 10001111 0000000000000000000011000	0x47800018

6 DDD screenshot fp

The screenshot shows the DDD (Digital Design Debugger) interface. The main window displays assembly code for a floating-point addition program. The code includes comments and instructions for loading constants, extracting exponents, normalizing the mantissas, and rounding the result. A right-hand panel shows a list of registers (r0 to r12) with their current values and addresses. A bottom panel shows the 'Registers' window with a table of register values.

Assembly Code:

```

File Edit View Program Commands Status Source Data Help
(): Homework_5_fp.s:62
fpNum0: .word 0x3f010000
fpNum1: .word 0x477ffffb0

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

@ Your Code goes here:
@ masking and shifting down exponents
    LSR r2, r0, #23    @ r2[7:0] = exponent of r0
    LSR r3, r1, #23    @ r3[7:0] = exponent of r1

@ masking and appending leading 1
/* can be perform with orr-ing with leading 1 and then shifting
   left to delete exponent and sign bits */
    ORR r4, r0, #0x00800000
    ORR r5, r1, #0x00800000
    LSL r4, r4, #8      @ r4[31:8] = mantissa of r0
    LSL r5, r5, #8      @ r5[31:8] = mantissa of r1

@ comparing exponents and setting based on value
    SUBS r6, r2, r3     @ r6 = r2 - r3 if r2 > r3
    MOVGE r7, r2        @ r7 = r2 if r2 >= r3
    MOVLTI r7, r3       @ r7 = r3 if r2 < r3

@ right shifting mantissa by difference of exponents
    LSRGT r5, r5, r6     @ r5 = r5 >> r6
    SUBLT r6, r3, r2     @ r6 = r3 - r2 if r3 > r2
    LSRLT r4, r4, r6     @ r4 = r4 >> r6

@ summing mantissas
    ADDS r8, r4, r5      @ r8 = r4 + r5

@ normalize the result in case of overflowing
    LSRCS r8, r8, #1     @ r8 = r8 >> 1
    ADDCS r7, r7, #1     @ r7 += 1

@ rounding result, do nothing because trunc is fine

@ strip leading 1 and merge everything
@ R8 = mantissa, R7 = exponent, sign = 0 because work with positive
    LSL r8, r8, #1       @ r8 = r8 << 1 strip leading 1
    LSR r8, r8, #9       @ r8[22:0] = mantissa
    LSL r7, r7, #23      @ r7[31:23] = biased exponent
    ORR r0, r7, r8       @ r0 = sum in IEEE754

bx lr
.end @ end of code

```

Registers Window:

Register	Value	Address
r0	0x47800018	11995
r1	0x477ffffb0	11995
r2	0x7e	126
r3	0x8e	142
r4	0x8100	33024
r5	0xfffffb000	42949
r6	0x10	16
r7	0x47800000	11995
r8	0x18	24
r9	0x0	0
r10	0x76fff000	19964
r11	0x0	0
r12	0x7efff450	21307

Registers Window Controls:

- Integer registers (selected)
- All registers
- Close
- Help