Cristian Robles
ECE425#16103
Project 2

Program Assembly Code:

```
        AREA Reset, CODE, Readonly
ENTRY
ADD0  EQU 0x40000000
ADDR   RN R0
COUNT  RN R1
NUM1P1 RN R2
NUM1P2 RN R3
NUM1P3 RN R4
NUM2P1 RN R5
NUM2P2 RN R6
NUM2P3 RN R7
HOLD1  RN R8
HOLD2  RN R9
HOLD3  RN R10

            ;load address1 into r1
            LDR ADDR,=ADD0
            MOV COUNT,#99
            ;initializing values
            MOV NUM1P1, #0
            MOV NUM1P2, #0
            MOV NUM1P3, #0
            MOV NUM2P1, #0
            MOV NUM2P2, #0
            MOV NUM2P3, #1

AGAIN
    ;add numbers
            ADDS HOLD3,  NUM1P3, NUM2P3
            ADCS HOLD2,  NUM1P2, NUM2P2
            ADC  HOLD1,  NUM1P1, NUM2P1

            ;store numbers
            STR HOLD3, [ADDR]
            ADD ADDR, ADDR, #4
            STR HOLD2, [ADDR]
            ADD ADDR, ADDR, #4
            STR HOLD1, [ADDR]
            ADD ADDR, ADDR, #4

            ;num2 = new num1
            MOV NUM1P1, NUM2P1
            MOV NUM1P2, NUM2P2
            MOV NUM1P3, NUM2P3

            ;hold = new num2
            MOV NUM2P1, HOLD1
            MOV NUM2P2, HOLD2
            MOV NUM2P3, HOLD3

            ;loop to count times
            SUBS COUNT,COUNT,#1

            ;branch to again if count is not yet done
            BNE AGAIN

stop B stop
        END
```

Snapshots of Code:

```
1       AREA Reset, CODE, Readonly
2   ENTRY
3   ADD0   EQU 0x40000000
4   ADDR    RN R0
5   COUNT   RN R1
6   NUM1P1  RN R2
7   NUM1P2  RN R3
8   NUM1P3  RN R4
9   NUM2P1  RN R5
10  NUM2P2  RN R6
11  NUM2P3  RN R7
12  HOLD1   RN R8
13  HOLD2   RN R9
14  HOLD3   RN R10
15
16          ;load address1 into r1
17          LDR ADDR,=ADD0
18          MOV COUNT,#99
19          ;initializing values
20          MOV NUM1P1, #0
21          MOV NUM1P2, #0
22          MOV NUM1P3, #0
23          MOV NUM2P1, #0
24          MOV NUM2P2, #0
25          MOV NUM2P3, #1
```

Figure 2.1 - Snapshot of Code in Keil

```
27  AGAIN
28          ;add numbers
29          ADDS HOLD3,  NUM1P3, NUM2P3
30          ADCS HOLD2,  NUM1P2, NUM2P2
31          ADC  HOLD1,  NUM1P1, NUM2P1
32
33          ;store numbers
34          STR HOLD3, [ADDR]
35          ADD ADDR, ADDR, #4
36          STR HOLD2, [ADDR]
37          ADD ADDR, ADDR, #4
38          STR HOLD1, [ADDR]
39          ADD ADDR, ADDR, #4
40
41          ;num2 = new num1
42          MOV NUM1P1, NUM2P1
43          MOV NUM1P2, NUM2P2
44          MOV NUM1P3, NUM2P3
45
46          ;hold = new num2
47          MOV NUM2P1, HOLD1
48          MOV NUM2P2, HOLD2
49          MOV NUM2P3, HOLD3
50
51          ;loop to count times
52          SUBS COUNT,COUNT,#1
53
54          ;branch to again if count is not yet done
55          BNE AGAIN
```

Figure 2.2 - Snapshot of Code in Keil

In Figure 2.1 which shows lines 1-25 I first declared all of the registers and constants I would use throughout the code. Second I initialized the first two values of the Fibonacci sequence by moving 0 into num1 and 1 into num2. Finally I split up the numbers into three registers so that I wouldn't run into any problems when I reached the higher numbers that would be too big to be represented by 32 bits. In Figure 2.2 which shows lines 27-55 I made a loop that would first add all three parts of the number and store them in a separate variable called hold. I then stored the numbers into their respective addresses and added 4 to the address so the next part of the number saved would not overwrite the last saved one. Next, I stored the hold into num2, num2 into num1 so the sequence could continue the next loop around. Finally I had a counter that would subtract once after every loop and a check to make sure the program did not run infinitely. Once the count reached 0 after 100 loops the program was done.

SRAM Memory Locations:

```
Memory 1
Address: 0x40000000
0x40000000: 01 00 00 00 00 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00 08
0x40000031: 00 00 00 00 00 00 00 00 00 00 00 0D 00 00 00 00 00 00 00 00 00 00 00 15 00 00 00 00 00 00 00 00 00 00 00 22 00 00 00 00 00 00 00 00 00 00 00 37 00
0x40000062: 00 00 00 00 00 00 00 00 00 00 59 00 00 00 00 00 00 00 00 00 00 00 90 00 00 00 00 00 00 00 00 00 00 00 E9 00 00 00 00 00 00 00 00 00 00 00 79 01 00
0x40000093: 00 00 00 00 00 00 00 00 00 62 02 00 00 00 00 00 00 00 00 00 00 DB 03 00 00 00 00 00 00 00 00 00 00 3D 06 00 00 00 00 00 00 00 00 00 00 18 0A 00 00
0x400000C4: 00 00 00 00 00 00 00 00 55 10 00 00 00 00 00 00 00 00 00 00 6D 1A 00 00 00 00 00 00 00 00 00 00 C2 2A 00 00 00 00 00 00 00 00 00 00 2F 45 00 00 00
0x400000F5: 00 00 00 00 00 00 00 F1 6F 00 00 00 00 00 00 00 00 00 00 20 B5 00 00 00 00 00 00 00 00 00 00 11 25 01 00 00 00 00 00 00 00 00 00 31 DA 01 00 00 00
0x40000126: 00 00 00 00 00 00 42 FF 02 00 00 00 00 00 00 00 00 00 00 73 D9 04 00 00 00 00 00 00 00 00 B5 D8 07 00 00 00 00 00 00 00 00 00 28 B2 0C 00 00 00 00
0x40000157: 00 00 00 00 00 DD 8A 14 00 00 00 00 00 00 00 00 00 00 05 3D 21 00 00 00 00 00 00 00 00 E2 C7 35 00 00 00 00 00 00 00 00 00 E7 04 57 00 00 00 00 00
0x40000188: 00 00 00 00 C9 CC 8C 00 00 00 00 00 00 00 00 00 00 B0 D1 E3 00 00 00 00 00 00 00 00 79 9E 70 01 00 00 00 00 00 00 00 00 29 70 54 02 00 00 00 00 00
0x400001B9: 00 00 00 A2 0E C5 03 00 00 00 00 00 00 00 00 CB 7E 19 06 00 00 00 00 00 00 00 00 6D 8D DE 09 00 00 00 00 00 00 00 00 38 0C F8 0F 00 00 00 00 00 00
0x400001EA: 00 00 A5 99 D6 19 00 00 00 00 00 00 00 00 DD A5 CE 29 00 00 00 00 00 00 00 00 82 3F A5 43 00 00 00 00 00 00 00 00 5F E5 73 6D 00 00 00 00 00 00 00
0x4000021B: 00 E1 24 19 B1 00 00 00 00 00 00 00 00 40 0A 8D 1E 01 00 00 00 00 00 00 00 21 2F A6 CF 01 00 00 00 00 00 00 00 61 39 33 EE 02 00 00 00 00 00 00 00
0x4000024C: 82 68 D9 BD 04 00 00 00 00 00 00 00 E3 A1 0C AC 07 00 00 00 00 00 00 00 65 0A E6 69 0C 00 00 00 00 00 00 00 48 AC F2 15 14 00 00 00 00 00 00 00 AD
0x4000027D: B6 D8 7F 20 00 00 00 00 00 00 00 F5 62 CB 95 34 00 00 00 00 00 00 00 A2 19 A4 15 55 00 00 00 00 00 00 00 97 7C 6F AB 89 00 00 00 00 00 00 00 39 96
0x400002AE: 13 C1 DE 00 00 00 00 00 00 00 D0 12 83 6C 68 01 00 00 00 00 00 00 09 A9 96 2D 47 02 00 00 00 00 00 00 D9 BB 19 9A AF 03 00 00 00 00 00 00 E2 64 B0
0x400002DF: C7 F6 05 00 00 00 00 00 00 BB 20 CA 61 A6 09 00 00 00 00 00 00 9D 85 7A 29 9D 0F 00 00 00 00 00 00 58 A6 44 8B 43 19 00 00 00 00 00 00 F5 2B BF B4
0x40000310: E0 28 00 00 00 00 00 00 4D D2 03 40 24 42 00 00 00 00 00 00 42 FE C2 F4 04 6B 00 00 00 00 00 00 8F D0 C6 34 29 AD 00 00 00 00 00 00 D1 CE 89 29 2E
0x40000341: 18 01 00 00 00 00 00 60 9F 50 5E 57 C5 01 00 00 00 00 00 31 6E DA 87 85 DD 02 00 00 00 00 00 91 0D 2B E6 DC A2 04 00 00 00 00 00 C2 7B 05 6E 62 80
0x40000372: 07 00 00 00 00 00 53 89 30 54 3F 23 0C 00 00 00 00 00 15 05 36 C2 A1 A3 13 00 00 00 00 00 68 8E 66 16 E1 C6 1F 00 00 00 00 00 7D 93 9C D8 82 6A 33
0x400003A3: 00 00 00 00 00 E5 21 03 EF 63 31 53 00 00 00 00 00 62 B5 9F C7 E6 9B 86 00 00 00 00 00 47 D7 A2 B6 4A CD D9 00 00 00 00 00 A9 8C 42 7E 31 69 60 01
0x400003D4: 00 00 00 00 F0 63 E5 34 7C 36 3A 02 00 00 00 00 99 F0 27 B3 AD 9F 9A 03 00 00 00 00 89 54 0D E8 29 D6 D4 05 00 00 00 00 22 45 35 9B D7 75 6F 09 00
0x40000405: 00 00 00 AB 99 42 83 01 4C 44 0F 00 00 00 00 CD DE 77 1E D9 C1 B3 18 00 00 00 00 78 78 BA A1 DA 0D F8 27 00 00 00 00 45 57 32 C0 B3 CF AB 40 00 00
0x40000436: 00 00 BD CF EC 61 8E DD A3 68 00 00 00 00 02 27 1F 22 42 AD 4F A9 00 00 00 00 BF F6 0B 84 D0 8A F3 11 01 00 00 00 C1 1D 2B A6 12 38 43 BB 01 00 00
0x40000467: 00 80 14 37 2A E3 C2 36 CD 02 00 00 00 41 32 62 D0 F5 FA 79 88 04 00 00 00 C1 46 99 FA D8 BD B0 55 07 00 00 00 02 79 FB CA CE B8 2A DE 0B 00 00 00
0x40000498: C3 BF 94 C5 A7 76 DB 33 13 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0x400004C9: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Figure 2.3 - SRAM Memory Location starting at 0x40000000 after program execution

CPSR Register:

CPSR	0x600000D3
N	0
Z	1
C	1
V	0
I	1
F	1
T	0
M	0x13

Figure 2.4 - CPSR Register after program execution

Used Registers after Code Execution:

Register	Value
Current	
R0	0x400004A4
R1	0x00000000
R2	0x0000000B
R3	0xDE2AB8CE
R4	0xCAFB7902
R5	0x00000013
R6	0x33DB76A7
R7	0xC594BFC3
R8	0x00000013
R9	0x33DB76A7
R10	0xC594BFC3
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000064

Figure 2.5 - After 100 loops(full program)

Register	Value
Current	
R0	0x4000024C
R1	0x00000000
R2	0x00000000
R3	0x00000001
R4	0xCFA62F21
R5	0x00000000
R6	0x00000002
R7	0xEE333961
R8	0x00000000
R9	0x00000002
R10	0xEE333961
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x00000064

Figure 2.6 - After 50 loops (half program)

To confirm that my program worked I was able to check through the registers tab. As I explained I stored the numbers added into a temporary variable named hold broken into 3 parts in registers R8-R10. In Figure 2.5 you can see R8 holds 0x13, R9 holds 0x33DB76A7, and R10 holds 0xC594BFC3. When you store those values next to each other in memory as I did as shown in Figure 2.3 you get 0x1333DB76A7C594BFC3 which equals the 100th Fibonacci number when converted to decimal. Just to be safe I also checked the 50th number in Figure 2.6 where R9 and R10 combined to hold 0x2EE333961 which is the 50th number in the Fibonacci sequence in hexadecimal.