فربد فولادی-98243045

عرفان رفیعی اسکوئی-98243027

در ابتدا یک پروژه می سازیم برای هر سوال بورد STM32 را انتخاب کردیم.

برای سوال اول از ما خواسته بود تا حاصل SUM برای اعداد 1 تا 8 را حساب کنیم.

برای حل این سوال ابتدا یک AREA تعریف می کنیم که دو مقدار count و sum که به ترتیب برابربا 8 و 0 هستند را با دستور EQU ایجاد کردیم.

```
    AREA myData, DATA, READONLY

COUNT   EQU 8
SUM     EQU 0
```

سپس کد اصلی را زدیم به این صورت که در ابتدا مقادیر count و sum را در رجیستر های R0 و R1 قرار دادیم و رجیستر R2 را نیر برابر با 1 قرار دادیم که حکم i برای حلقه را دارد.

```
AREA RESET, CODE, READONLY
ENTRY
    LDR R0, =COUNT
    LDR R1, =SUM
    LDR R2, =1   ; r2 stores the initial value of i
```

در ادامه یک برنچ به نام myLoop درست کردیم که یک حلقه است تا با شمارنده i (R2) از 1 تا 8 را جمع کند و در sum قرار دهد.

پس ابتدا مقدار sum + i را حساب می کنیم و در sum قرار می دهیم سپس مقدار i را یکی اضافه می کنیم و با کم کردن R0 از R2 چک میکنیم که با مقدار count برابر است یا خیر.(با ست کردن فلگ Z)

- اگر برابر باشد اخرین مقدار i را نیز به sum اضافه میکنیم و وارد یک حلقه بی نهایت می شویم و برنامه به پایان می رسد.

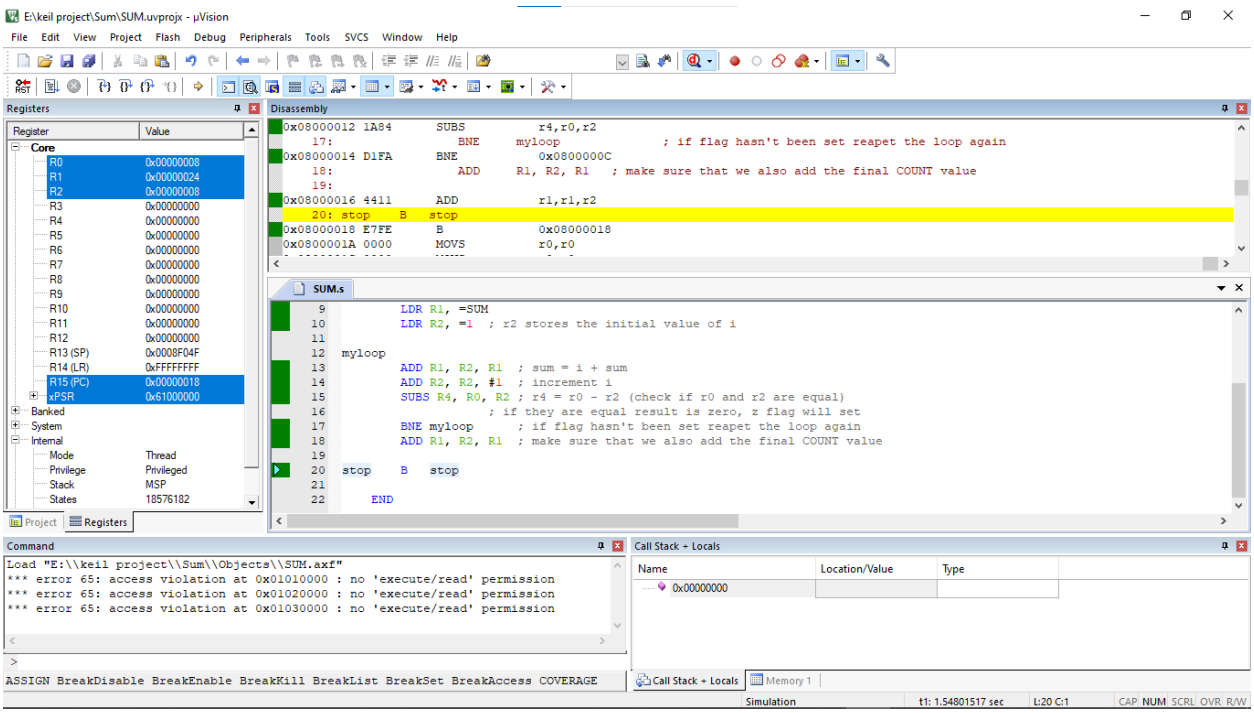
- اگر برابر نباشد دوباره به اول حلقه برنچ می زنیم و همان کار هارو تکرار می کنیم تا i برابر با count شود (شرط حلقه).

```
myloop
        ADD R1, R2, R1  ; sum = i + sum
        ADD R2, R2, #1  ; increment i
        SUBS R4, R0, R2 ; r4 = r0 - r2 (check if r0 and r2 are equal)
                        ; if they are equal result is zero, z flag will set
        BNE myloop      ; if flag hasn't been set reapet the loop again
        ADD R1, R2, R1  ; make sure that we also add the final COUNT value

stop    B   stop

    END
```

عکس از خروجی:



برای سوال دوم از ما خواسته بود تا یک مقدار دلخواه را تعداد 0 و 1 ان را حساب کنیم و یک سابروتین هم صدا بزنیم تا عملیات مد نظرش را انجام دهد.

برا ی حل ابتدا یک کد زدیم که تعداد 0 و 1 را حساب کند به این صورت که در R0 مقدار 0x1234 را قرار دادیم و در R4 مقدار 16 که تعداد بیت های عدد ما است و همین طور شمارنده حلقه ما.

همچنین یک AREA جدید تعیین می کنیم که DATA مون در ان قرار دارد و یک اسم نیز به عنوان برچسب قرار می دهیم برای دسترسی به این بخش و فضای 6 وردی تعیین می کنیم.

حال در AREA اصلی چون descending است اول باید sp را ببریم بالا ترین جایی که هست که میشه همون 0x18 یا 24.

```
 AREA RESET, DATA, READONLY
  DCD           0     ;initial_sp
  DCD           MAIN;reset_vector

 AREA SA,DATA,READWRITE
SS SPACE 0x18

 AREA MyCode,CODE,READONLY
      ENTRY
MAIN
        LDR R11, =SS
        ADD R11, R11,#0x00000018
        MOV SP, R11

        MOV R0, #0x1234      ; Register Rd.
        MOV R4, #0x10        ; Register's length (counter (16 bit)).
```

حال یک برنچ START می سازیم که main کد ما است.

در این جا ابتدا عدد را یک بیت به راست شیفت می دهیم و flag ها را نیز ست میکنیم.

سپس چک می کنیم که فلگ C ست شده است یا نه:

- اگر 1 باشد ست می شود و به COUNTING_ONE برنچ می شویم و مقدار R1 یکی اضافه می شود.
- اگر 0 باشد ست نمی شود و به COUNTING_ZERO برنچ می شویم و مقدار R2 یکی اضافه می شود.

```
START
        MOVS R0, R0, LSR #1 ; Logical shift right of number.
        BCS COUNTING_ONE    ; Branch to COUNTING_ONE if crray flag was set .
        BCC COUNTING_ZERO   ; Branch to COUNTING_zero if crray flag was NOT set.
```

```
COUNTING_ONE
       ADD R1, R1, #1       ; R3 holds number of 1s.
       B   CONTINUE         ; Back to CONTINUE to decrement the counter and check the end of the loop.


COUNTING_ZERO
       ADD R2, R2, #1       ; R4 holds number of 0s.
       B   CONTINUE         ; Back to CONTINUE to decrement the counter and check the end of the loop.
```

در مرحله بعد به COUNTINUE که ادامه START است برنچ می شویم که شرط حلقه را بررسی می کند که حلقه را تکرار کند یا به سابروتین برنچ شود.

```
CONTINUE
        SUB R4, R4, #1      ; decrementing counter value.
        CMP R4, #0x0        ; Check for end of the loop.
        BNE START           ; If there was still any bits we run the loop again.
        BL  SUBR
```

در سابروتین هم مقادیر R1 و R2 و LR را در استک میریزیم و یک سری دستورات MUL و SUB انجام می دهیم و دوباره اون مقادیر را از استک بر می گردونیم و به START بر می گردیم و وارد STOP می شویم و برنامه به اتمام می رسد.

```
SUBR
    PUSH {R1, R2, LR}

    MOV R5, #3
    MUL R2, R1, R5       ; R2 = R1 * 3
    MOV R6, #100
    SUB R3, R2, R6       ; R3 = R2 - 100

    POP {R1, R2, LR}

    BX  LR

      END
```

عکس از خروجی: