بسمه تعالی

گزارش آزمایش ٤

آزمایشگاه ریزپردازنده

سید محمدرضا حسینی

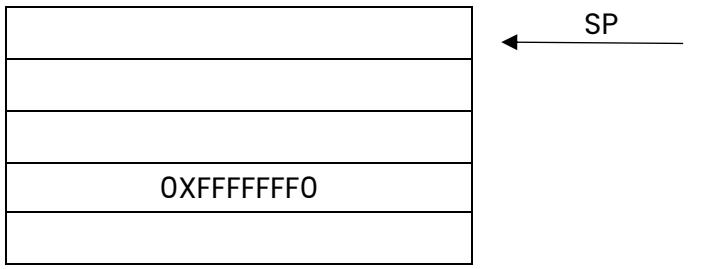
حسین حاجی رومنان

سوالات تحلیلی:

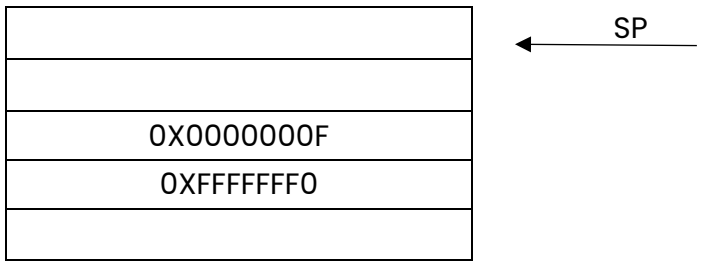
الف)

در ابتدا استک به صورت بالا میباشد .

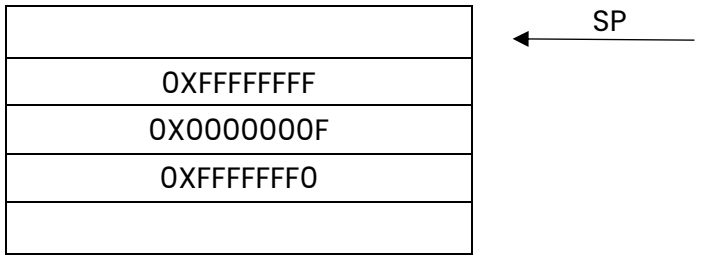
با اجرای [SP,#-۱۲],R۰ STR محتویات استک به صورت روبرو میشود.



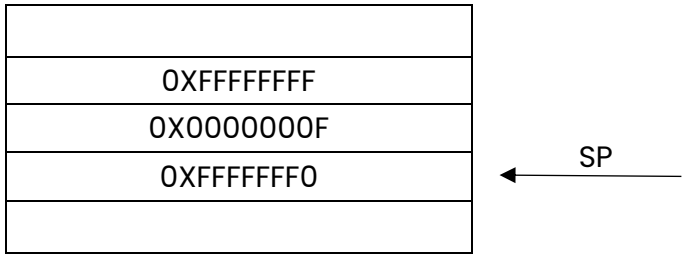
با اجرای [SP,#-۸],R۱ STR محتویات استک به صورت روبرو میشود.



با اجرای [SP,#-٤],R۲ STR محتویات استک به صورت روبرو میشود



با اجرای #۱۲ , SP , SUB SP مقدار رجیستر SP به اندازه ۱۲ واحد کاهش میابد.



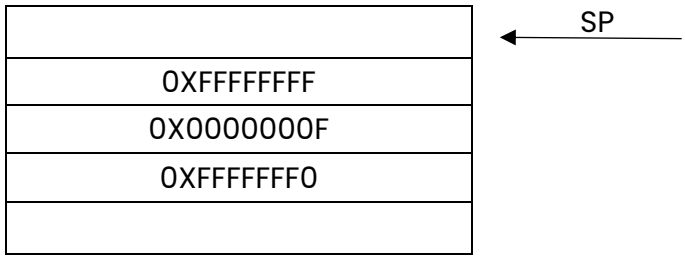
اجرای ۳ دستور زیر تغییری در پشته به وجود نمی آورد .

LDR R۰, [SP, #۸]

LDR R۱, [SP, #٤]

LDR R۲, [SP, #۰]

با اجرای دستور ADD SP,SP,#۱۲ مجددا مقدار SP برابر مقدار اولیه می شود.



مقدار رجیستر ها به صورت زیر خواهد بود:

R۰: ۰XFFFFFFFF

R۱ : ۰X۰۰۰۰۰۰۰F

R۲: ۰XFFFFFFF۰

ب)

```
LDR R٠,=٠XFFFFFFF٠
LDR R١,=٠X٠٠٠٠٠٠F
LDR R٢,=٠XFFFFFFFF
MOV R٣,SP
SUB R٣,#١٢
STM R٣,{R٢,R١,R٠}
SUB SP, SP, #١٢
MOV R٣,SP
LDM R٣,{R٢,R١,R٠}
ADD SP, SP, #١٢
```

ج)

```
LDR R0,=0XFFFFFFF0
LDR R1,=0X0000000F
LDR R2,=0XFFFFFFFF
PUSH {R2,R1,R0}
POP  {R2,R1,R0}
```

دستور کار:

۱)

```
13  ; main logic of code
14  __main FUNCTION
15         ENTRY
16  START
17      MOV R0,#2 ; fitst element of lucas
18      MOV R1,#1 ; second element of lucas
19      MOV R2,#15  ; N
20      PUSH {R2}
21      MOV R3,#0
22      MOV R2,#0
23  LUCAS
24      POP {R2}
25      PUSH {R0} ; pushing first
26      ADD R3,R3,#1
27      CMP R3,R2
28      BEQ loop
29      PUSH {R1} ; pushing second
30      ADD R3,R3,#1
31      CMP R3,R2
32      BEQ loop
33      ADD R0,R0,R1
34      PUSH {R0} ; pushing third
35      ADD R3,R3,#1
36      CMP R3,R2
37      BEQ loop
38  LUCAS_CON ; computing lucas for n>=4
39      CMP R0,R1
40      ADDLT R0,R0,R1
41      ADDGT R1,R0,R1
42      PUSHLT {R0}
43      PUSHGT {R1}
44      ADD R3,#1
45      CMP R3,R2
46      BEQ loop
47      B LUCAS_CON
48  loop
49      b loop
50   ENDFUNC
51   END
52
53
```

مقدار N را از استک پاپ کرده و در R۲ ذخیره میکنیم.

ابتدا مقدار ۰ و ۱ را در R۰ , R۱ میریزیم سپس هر دو را در استک ذخیره میکنیم. چون مقدار R۰>R۱ بزرگتر است لازم است مقدار R۲ را به صورت دستی حساب کنیم و بقیه الگوریتم را میتوان به صورت نوشته شده در LOCAS_CON ادامه داد.

در حلقه LOCAS_CON مقدار جدید در رجیستری ذخیره میشود که مقدار کمتر را دارا باشد پس آن یک واحد به مقدار R۳ اضافه میشود و چک میکنیم آیا به تعداد N های دلخواه رسیده ایم یا خیر . در صورت رسیدن برنامه به اتمام میرسد و در غیر اینصورت حلقه تکرار میشود.

(۲

```
; main logic of code
__main FUNCTION
       ENTRY
START
    MOV R0,#2 ; fitst element of lucas
    MOV R1,#1 ; second element of lucas
    MOV R3,#0

LUCAS
    ADD R0,R0,R1
    ADD R3,R3,#3

LUCAS_CON ; FINDING BIGGEST LUCAS NUMBER WHICH IS LESS THAN 0XFFFFFFFF
    CMP R0,R1
    ADDSLS R0,R0,R1
    ADDSHI R1,R0,R1
    ADD R3,#1
    BCS FINAL ; IF CARRY FLAG SET => 33BIT
    B LUCAS_CON
FINAL
    CMP R0,R1 ; BIGGER NUMBER IN THE REGISTERS
    PUSHHI {R0}
    PUSHLS {R1}
    SUB R3,#1
    PUSH {R3}
loop
    b loop
 ENDFUNC
 END
```

در این برنامه تا منتظر میمانیم تا کری فلگ بر اثر جمع دو مقدار یک شود در آنصورت به جمله ای از سری لوکاس رسیده ایم که به ۳۳ بیت برای ذخیره سازی نیاز دارد .

پس میتوانیم مقدار دیگر که بزرگترین ۳۲ بیتی است را ذخیره کرده و مقدار R۳ را به عنوان تعداد در استک ذخیره کنیم.

(۳

```
; main logic of code
__main FUNCTION
       ENTRY
        LDR R2,=0X1
        STR R2,[SP]
        MOV R5,#0; NUM OF SHIFTS

        LDR R4,[SP]
SET     LSR R4,R4,#1  ; FINDING OUT NUM OF BIT IN THE NUMBER
        ADD R5,R5,#1
        CMP R4,#0
        BNE SET
        EOR R0,R0,R0
        EOR R1,R1,R1
COUNT
        LSRS R2,#1  ; RIGHT SHIFT AND SETTING FLAGS
        ADDCS R1,#1  ; INC NUM OF ONE IF C=1
        ADDCC R0,#1 ; INC NUM IF ZEROS IF C=0
        SUB R5,#1
        CMP R5,#0
        BEQ loop
        B COUNT
loop
    b loop
 ENDFUNC
 END
```

ابتدا تعداد بیت های عدد را محاسبه میکنیم.

سپس با استفاده از شیفت منطقی راست مقدار را بیت به بیت بررسی میکنیم.

اگر ۱ بوده باشد کری فلگ برابر ۱

اگر ۰ بوده کری فلگ ۰ میشود

٤)

```
; main logic of code
__main FUNCTION
       ENTRY
        LDR R0,=0XFFFFFF ; N
        MOV R2,#1
FACT
        CMP R0, #1
        BLE loop
        MUL R2, R0, R2 ; CALCULATING DOUBLE FACTORIEL
        SUB R0, R0, #2 ; DECREASING BY 2
        B FACT

loop
    b loop
 ENDFUNC
 END
```

N را در R۰ ذخیره میکنیم و مقدار R۲ را برابر یک قرار میدهیم . سپس تا زمانی که مقدار R۰ کمتر مساوی ۱ شود ، مقدار آن را در R۲ ضرب میکنیم و حاصل را در R۲ ذخیره میکنیم. از R۲ نیز ۲ واحد کم میکنیم .