-

دانشکده مهندسی
کامپیوتر و فناوری اطلاعات

6/13/2021

Homework 6

Lec 19-21

MICROPROCESSOR
AND
ASSEMBLY LANGUAGE

Spring 2021

1) کد اسمبلی‌ای بنویسید که بدون استفاده از MUL بتواند R1 را به توان R0 برساند.

ابتدا توضیح مختصری در رابطه با کد میدهیم. در اینجا فرض شده است که توان R0 همواره مقداری بزرگ تر مساوی 0 میتواند داشته باشد ولی R1 میتواند منفی هم باشد. در ضمن حاصل نهایی در R3 نگهداری میشود.

در اینجا ابتدا ما مقدار موجود در R1 را ازش قدر مطلق میگیریم و مقدار توان آن را حساب میکنیم و در آخر علامت R1 را در حاصل اعمال میکنیم. به این صورت است که در هنگامی که از R1 داریم قدر مطلق میگیریم اگر مقدارش منفی بود مقدار R7 را برابر با 1 میکنیم در غیر اینصورت مقدار R7 برابر با 0 میماند. در پایان نیز که حاصل توان را توسط حلقه حساب کردیم به سراغ چک کردن توان میرویم که اگر زوج باشد قطعا خروجی مثبت است و حاصل در همان R3 میباشد ولی اگر توان فرد باشد و مقدار R1 نیز منفی بوده باشد باید علامت حاصلی که در R3 است منفی شود.

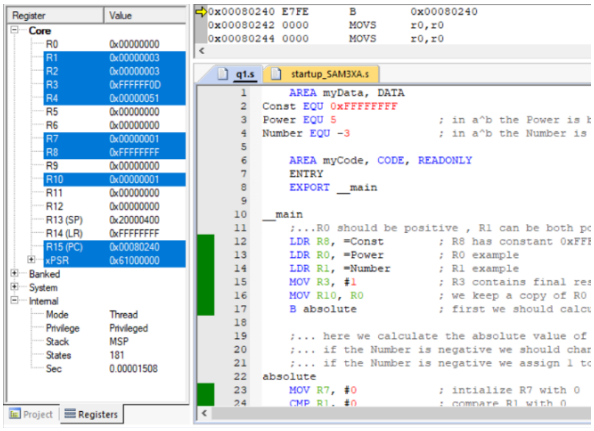
همچنین چون حق استفاده از ضرب نداریم به کمک حلقه‌ی تو در تو و استفاده از جمع و اعمال ساده دیگر حاصل توان را حساب کردیم. در زیر اسکرین شات هایی از کدی که در keil زده شده است آمده است:

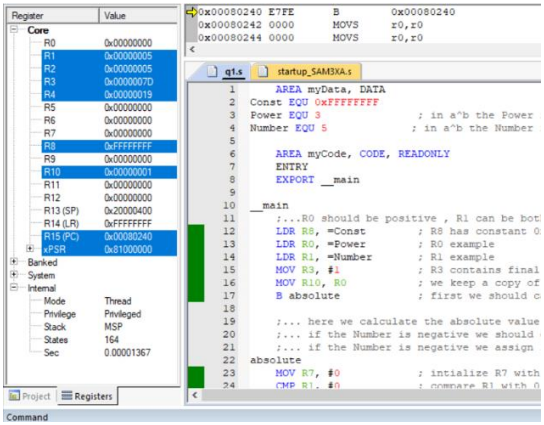
q1.s

```
1     AREA myData, DATA
2  Const EQU 0xFFFFFFFF
3  Power EQU 5                ; in a^b the Power is b
4  Number EQU -3              ; in a^b the Number is a
5
6     AREA myCode, CODE, READONLY
7     ENTRY
8     EXPORT __main
9
10 __main
11    ;...R0 should be positive , R1 can be both positive and negative
12    LDR R8, =Const          ; R8 has constant 0xFFFFFFFF
13    LDR R0, =Power          ; R0 example
14    LDR R1, =Number         ; R1 example
15    MOV R3, #1              ; R3 contains final result
16    MOV R10, R0             ; we keep a copy of R0 or Power to check is it even or not
17    B absolute              ; first we should calculate the absolute value of Number
18
19    ;... here we calculate the absolute value of Number
20    ;... if the Number is negative we should change it to positive
21    ;... if the Number is negative we assign 1 to R7 to check if the power is Odd then we impact it
22 absolute
23    MOV R7, #0              ; intialize R7 with 0
24    CMP R1, #0              ; compare R1 with 0
25    BGE here                ; if (R0 >= 0) then go to here
26    EOR R1, R1, R8          ; R1 = R1 xor R8
27    ADD R1, R1, #1          ; R1 = R1 + 1
28    ADD R7, R7, #1          ; R7++
29 here B external_loop  ;  go to external_loop
```

```
    ;... in this two loop we calculate |R1|^R0
external_loop
    CMP R0, #0              ; compare R0 with zero
    BLS impact_sign         ; if (R0 <= 0) then finish the process of calculation of |R1|^R0 and we should impact the sign in result
    MOV R4, R3              ; R4 = R3
    SUB R0, R0, #1          ; R0--
    MOV R2, #1              ; initial R2 with 1 (R2 = 1)

internal_loop
    CMP R2, R1              ; compare R1 with R2
    BEQ external_loop       ; if (R2 == R1) then go to external_loop
    ADD R3, R3, R4          ; R3 += R4
    ADD R2, R2, #1          ; R2++
    B internal_loop         ; again on internal_loop

    ;...here we check :
    ;......if the Power is odd and the Number is negative --> result is negative
    ;......if the Power is even  --> result is positive
impact_sign
    MOV R9, #0              ; initialize R9 with 0

    ;...if the power is even the value of R9 is going to 1
loop
    CMP R10, #0             ; compare R10 with 0
    BNE here_2              ; if (R10 != 0) then go to here_2
    ADD R9, R9, #1          ; R9++
    B here_4                ; go to here_4

here_2
    CMP R10, #1             ; compare R10 with 1
    BNE here_3              ; if (R10 != 1) then go to here_3
    B here_4                ; go to here_4

here_3
    SUB R10, R10, #2        ; R10--
    B loop                  ; go to third_looop

here_4
    CMP R9, #1              ; compare R9 with 1
    BEQ done                ; if (R9 == 1) the power is even and the result doesn't need to be changed
    CMP R7, #1              ; compare R7 with 1
    BNE done                ; if (R9 == 0) the power is odd and if (R7 != 1) the Number was positive and the result doesn't need to be changed
    EOR R3, R3, R8          ; R1 = R1 xor R8
    ADD R3, R3, #1          ; R1 = R1 + 1

done B done                 ; done , note that final result is in R3
    END
```

برای مثال در عکس سمت راست حاصل 5 به توان 3 سمت چپ حاصل 3- به توان 5 را حساب کرده ایم و در R3 ذخیره کردیم:





همچنین توجه شود که چون به مقدار R0 تا آخر برنامه نیاز نبود و یک کپی از آن در R10 برای چک کردن زوج و فرد بودن توان داشتیم ، مقدارش در جریان برنامه تغییر کرده است.

2) برای کد اسمبلی زیر شبه کدی بنویسید و توضیح دهید چه کاری انجام می دهد.

من در این سوال با توجه به اینکه در CMP اول رجیستر می آید وOp2 میتواند یک مقدار immediate داشته باشد ، برداشت کردم که شاید اشتابه تایپی باشد که این مقادیر جابه جا شده و با توجه به آن پاسخ داده ام.

```
CMP #0,R1
BLS I1
CMP R2,15
BEQ E1
B HERE
I1    CMP #0, R0
BEQ I2
CMP R2, 15
BEQ E1
B HERE
I2    ADD R2,R2,#1
B HERE
E1 MOV R2, #0
ADD R3, R3, 1
HERE B HERE
```

شبه کد زیر را میتوان برای این کد اسمبلی نوشت و توضیحات مربوطه در صفحه بعد آمده است:

```
if ( R1 <= 0){
    if( R0 == 0){
        R2 = R2 +1
    }else if( R2 == 15){
        R2 = 0
        R3 = R3 +1
    }
}else if ( R2 == 15) {
    R2 = 0
    R3 = R3 +1
}
```

در این کد براساس مقادیر برخی متغیرها یا همان رجسترها تصمیم گرفته میشود که چه اتفاقی بیوفتد.

اگر مقدار R1 کوچک تر مساوی 0 باشد باید دو حالت را بررسی کنیم اینکه اگر مقدارR0 برابر با 0 بود مقدار R2 را یک واحد زیاد کند. درغیر اینصورت اگر مقدار R2 برابر با 15 بود باید مقدار آن را 0 کند و یک واحد با R3 زیاد کند. اگر هم شرط اول یعنی R1 کوچکتر مساوی 0 برقرار نبود در صورتی که R2 برابر با 15 بود باید مقدارش را 0 کند و به R3 یک واحد اضافه کند.

3) کد اسمبلی‌ای بنویسید که 4 را به توان مقدار رجیستر R1 برساند.

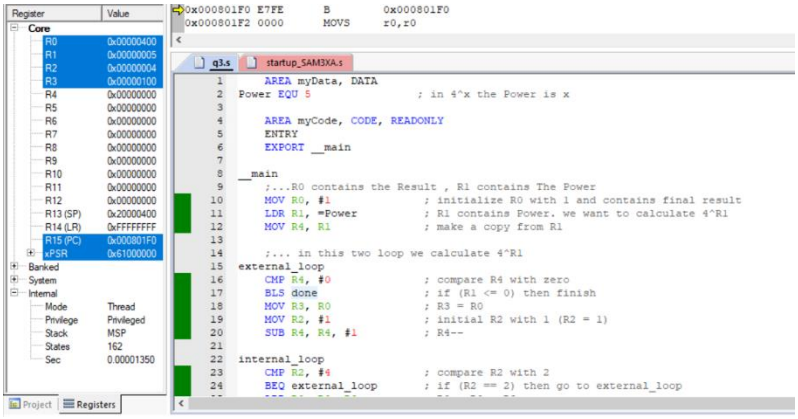
با توجه به کدی که برای سوال 1 نوشتیم، این سوال نیز به همان شیوه قابل حل میباشد و چون اینجا میدانیم مقدار عددی که به توان میرسد مثبت است دیگر نیازی به چک کردن علامت نیست. پس بدون ضرب و به همان شیوه قبلی و با لوپ تو در تو میتوان و اعمال ساده جمع و ... میتوان حاصل 4 به توان مقدار موجود در R1 را حساب کرد.

همچنین حاصل را در R0 ذخیره میکنیم. کد آن به صورت زیر میباشد:

q3.s | startup_SAM3XA.s

```
    AREA myData, DATA
Power EQU 5                 ; in 4^x the Power is x

    AREA myCode, CODE, READONLY
    ENTRY
    EXPORT __main

__main
    ;...R0 contains the Result , R1 contains The Power
    MOV R0, #1              ; initialize R0 with 1 and contains final result
    LDR R1, =Power          ; R1 contains Power. we want to calculate 4^R1
    MOV R4, R1              ; make a copy from R1

    ;... in this two loop we calculate 4^R1
external_loop
    CMP R4, #0              ; compare R4 with zero
    BLS done                ; if (R1 <= 0) then finish
    MOV R3, R0              ; R3 = R0
    MOV R2, #1              ; initial R2 with 1 (R2 = 1)
    SUB R4, R4, #1          ; R4--

internal_loop
    CMP R2, #4              ; compare R2 with 2
    BEQ external_loop       ; if (R2 == 2) then go to external_loop
    ADD R0, R0, R3          ; R0 = R0 + R3
    ADD R2, R2, #1          ; R2++
    B internal_loop         ; again on internal_loop

done B done                 ; done , note that final result is in R0
    END
```

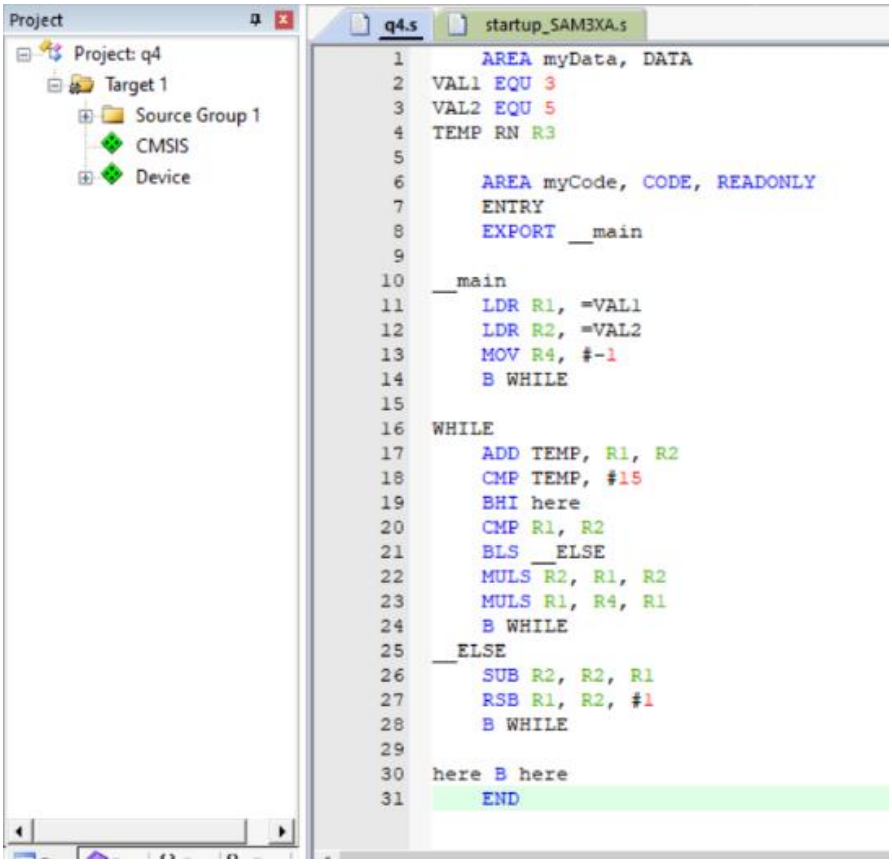
من مقدار اولیه برای R1 را برابر با 5 گرفتم و برای مثال اگر ران کنیم حاصل 4 به توان 5 را میتوانیم در R0 مشاهده کنیم.



4) کد زیر را به زبان اسمبلی ARM تبدیل کنید.

```
while R1 + R2 <= 15:
    if R1 > R2:
        R2 = R1 * R2
        R1 *= -1
    else:
        R2 = R2 - R1
        R1 = 1 - R2
```

در ادامه بنده یک سری مقادیر فرضی هم برای رجیسترها در نظر گرفتم ولی درواقع کد اصلی که اسمبلی همین عکس بالاس، از خط 16 شروع میشود. کد آن به صورت زیر میشود:



5) با توجه به مقادیر اولیه‌ی R0 و R1 مقادیر نهایی آن‌ها را در هر قسمت مشخص کنید.

Mov R1, #12

Mov R2, #3

Mov R3, #9

الف) MLA R1, R2, R3, R1

ب) MOV R1, R3, LSR, R2

ج) MOV R3, R1, ROR, R2

الف)

مقدار R2 برابر 3 و مقدار R3 برابر 9 میماند ولی مقدار R1 برابر با 39 میشود. چراکه R2 را در R3 ضرب میکند و حاصل با R1 جمع میکند و حاصل نهایی آن را در R1 میریزد.

ب)

مقدار R2 برابر 3 و مقدار R3 برابر 9 میماند ولی مقدار R1 برابر با 1 میشود. چرا که مقدار R3 را به اندازه R2 که برابر 3 است یعنی 3 بیت به راست شیفت میدهد و آن را در R1 ذخیره میکند.

ج)

مقدار R2 برابر 3 و مقدار R1 برابر 12 میماند ولی مقدار R3 برابر با 0x80000001 میشود. چراکه مقدار رجیستر R1 را به اندازه R2 یعنی 3 بیت Rotate به راست میدهد.

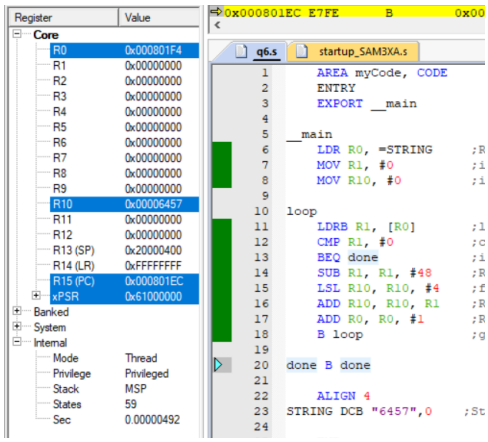
6) فرض کنید String شامل 4 کاراکتر عددی در اختیار داریم که به طور کامل در یک Word ذخیره شده باشد. (هر کاراکتر را یک Byte) فرض کنید. رجیستر R0 به اولین Byte این رشته اشاره می‌کند. برنامه اسمبلی‌ای بنویسید که این رشته را به BCD Packed تبدیل کند و در رجیستر R10 ذخیره کند. (فرض کنید رشته به صورت Little Endian ذخیره شده باشد.)

برای مثال بنده یک استرینگ به صورت "6457" در نظر گرفتم و به صورت گفته شده ذخیره شده است. در نهایت به کمک کد زیر آن را بایت به بایت لود میکنیم در R1 و 48 واحد از کد اسکی آن کم میکنیم و حاصل را در 4 بیت از R10 ذخیره میکنیم که ه صورت BCD Packed باشد. همچنین لازم است تا هربار 4 بیت رجیستر R10 را شیفت به چپ دهیم تا عدد جدید به فرم BCD را در آن 4 بیت خالی ایجاد شده قرار دهیم.

در آخر نیز حاصل در R10 ذخیره شده است. کد به صورت زیر میباشد:

q6.s | startup_SAM3XA.s

```
1   AREA myCode, CODE
2   ENTRY
3   EXPORT __main
4
5   __main
6   LDR R0, =STRING     ;R0 points to first Byte of STRING
7   MOV R1, #0          ;initialize R1 with 0
8   MOV R10, #0         ;initialize R10 with 0 ( it contains the result at the end)
9
10  loop
11  LDRB R1, [R0]       ;load first byte (or first character)
12  CMP R1, #0          ;compare R1 with 0
13  BEQ done            ;if (R1==0)it means that we arrive to the end of the string and we should finish the process
14  SUB R1, R1, #48     ;R1 = R1 - 48 (this is for chnaging the ASCII code of numbers to the right value of them)
15  LSL R10, R10, #4    ;for creating packed BCD we should save each number in BCD in 4 bits so we shift 4 bits to left each time
16  ADD R10, R10, R1    ;R10 = R10 + R1
17  ADD R0, R0, #1      ;R0 = R0 + 1 ( the address increments one unit and points to next byte or character in memory)
18  B loop              ;go back to loop again
19
20  done B done
21
22  ALIGN 4
23  STRING DCB "6457",0    ;String that contains for character and save in one word of memory
24
25  END
```



برای رشته ورودی ما که "6457" میباشد حاصل آن به صورت BCD Packed در نمایش هگزادسیمال برابر با 0x00006457 و در نمایش باینری (در 16 بیت اول رجیستر) برابر با : 0110 0100 0101 0111 میباشد و در تصویر روبرو نیز خروجی در R10 در نمایش هگزا دسیمال میباشد:

- مهلت ارسال تمرین ساعت 23.55 روز جمعه 4 ام تیر می‌باشد.
- سوالات خود را می‌توانید تنها از طریق ایمیل زیر بپرسید.
  - AUTMicroTA@gmail.com

- ارائه پاسخ تمرین بهتر است به دو روش زیر باشد:
  1) استفاده از فایل docx. تایپ پاسخ‌ها و ارائه فایل Pdf
  2) چاپ تمرین و پاسخ دهی به صورت دستنویس خوانا
- فایل پاسخ تمرین را تنها با قالب **HW5-9731***.pdf** در مودل بارگزاری کنید.
  - نمونه: HW5 -9731063