

12/25/2020



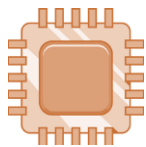
Homework 6

Lec 22-25



MICROPROCESSOR
AND
ASSEMBLY LANGUAGE

Fall 2021



1) فرض کنید وضعیت حافظه و رجیسترها به شکل زیر باشد:

آدرس حافظه

0x8010	0x00000001
0x800C	0xFEEDDEAF
0x8008	0x00008888
0x8004	0x12340000
0x8000	0xBABE0000

رجیستر

0x13	R0
0xFFFFFFFF	R1
0xEEEEEEEE	R2
0x8000	R3

پس از اجرای دستور زیر وضعیت و محتوای حافظه و رجیسترها بکشید و دلیل آن را توضیح دهید.

LDMIA R3!, {R0, R1, R2}

پاسخ:

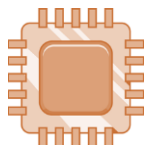
در اینجا دستور LDM به صورت Increasing After انجام می شود در نتیجه پس از هربار خواندن از مموری به اندازه ۴ واحد پوینتر افزایش می یابد و در انتها نیز مقدار نهایی پوینتر در رجیستر R3 ذخیره می شود. مقادیر خوانده شده از حافظه نیز به ترتیب در R2, R1, R0 ذخیره می شوند.

آدرس حافظه

0x8010	0x00000001
0x800C	0xFEEDDEAF
0x8008	0x00008888
0x8004	0x12340000
0x8000	0xBABE0000

رجیستر

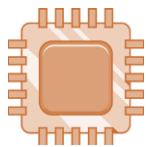
0xBABE0000	R0
0x12340000	R1
0x00008888	R2
0x800C	R3



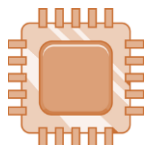
(2) برنامه ای به زبان اسمبلی بنویسید که نشان دهد یک عدد دلخواه اول است یا خیر. (عدد دلخواه را در رجیستر R0 قرار دهید. همچنین برای مشخص کردن اول نبودن عدد 0x00000000 و برای اول بودن عدد 0x11111111 را در رجیستر R3 بریزید)

پاسخ:

```
AREA Prime_or_Not,code,readonly
ENTRY
MOV R0,#15;           Number which you want to test
CMP R0,#01;           Comparing with 01
BEQ PRIME;             If equal declare directly as prime
CMP R0,#02;           Compare with 02
BEQ PRIME;             If equal declare directly as prime
MOV R1,R0;            Copy test number in R1
MOV R2,#02;           Initial divider
UP
BL DIVISION;          Call for division sub-function
CMP R8,#0;            Compare remainder with 0
BEQ NOTPRIME;         If equal then its not prime
ADD R2,R2,#01;        If not increment divider and check
CMP R2,R1;            Compare divider with test number
BEQ PRIME;            All possible numbers are done means It's
prime
B UP;                 If not repeat until end
NOTPRIME
LDR R3,#0x11111111;    Declaring test number is not prime
B STOP;               Jumping to infinite looping
PRIME
LDR R3,#0xFFFFFFFF;   Declaring test number is prime
number
STOP B STOP;          Infinite looping
```



DIVISION;	Function for division operation
MOV R8,R0;	Copy of data from main function
MOV R9,R2;	Copy of divider from main function
LOOP	
SUB R8,R8,R9;	Successive subtraction for division
ADD R10,R10,#01;	Counter for holding the result of
division	
CMP R8,R9;	Compares for non-zero result
BPL LOOP;	Repeats the loop if subtraction is still
needed	
MOV PC,LR;	Return back to main function
END	



3) عددی را در خانه 0x05000000 ثبت کنید. برنامه ای بنویسید که آن را تقسیم بر توان های 2، از یک تا ده کند و آن را در ده رجیستر اول بریزد

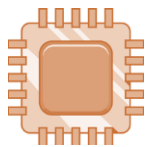
پاسخ:

```
LDR r0, =4096
LDR r1, =0x5000000
STR r0, [r1]

LDR r10, [r1]

ASR r0, r10, #1
ASR r1, r0, #1
ASR r2, r1, #1
ASR r3, r2, #1
ASR r4, r3, #1
ASR r5, r4, #1
ASR r6, r5, #1
ASR r7, r6, #1
ASR r8, r7, #1
ASR r9, r8, #1

here B here
```



4) قطعه کد اسمبلی معادل با کد C زیر را بنویسید

(الف)

```
for (R0 = 0 ; R0 < 10; R0++){  
    if (R1 == 0) {  
        R2++;  
    }  
}
```

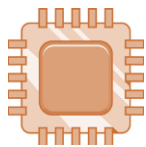
(ب)

```
int *ptr;  
int sum = 0;  
for (int i = 0; i < 20; i++)  
    sum += *(ptr++);
```

پاسخ:

(الف)

```
        MOV R0, #0  
AGAIN   CMP R0, #10  
        BEQ HERE  
        CMP R1, #0  
        ADDEQ R2, R2, #1  
        ADD R0, R0, #1  
        B AGAIN  
HERE    B HERE END
```



MICROPROCESSOR AND ASSEMBLY LANGUAGE

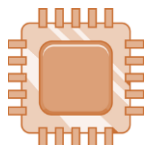
Dr. Farbeh

Homework 6



(ب)

```
ADR R0, 0X400
MOV R1, #0
MOV R2, #21
LOOP CMP R2, #1
    BEQ FINISH
    ADD R0, R0, #4
    LDR R3, [R0]
    ADD R1, R1, R3
    SUB R2, R2, #1
    B LOOP
FINISH END
```



5) در هر بخش آدرس خانه حافظه‌ای که به آن اشاره می‌شود را بدست آورید و رجیستری که با علامت سوال مشخص شده است را در هر مورد بنویسید. مراحل کار خود را توضیح دهید.
در همه موارد فرض کنید: $R3=0x4000$, $R4=0x20$
همچنین هر جا نیاز به خواندن خانه‌ای از حافظه را داشتید مقدار آن را $0xFF$ فرض کنید.

- a. `LDR R9, =0x11223344`
`STRH R9, [R3, R4]`
- b. `LDRB R8, [R3, R4, LSL #3] ; R8 = ?`
- c. `LDR R7, [R3], R4 ; R7 = ?, R3 = ?`
- d. `LDR R6, =0x11223344`
`STRB R6, [R3], R4, ASR #2, R3 = ?`

پاسخ:

*در صورت سوال اشتباهی به جای $R3$, $R5$ قرار داده شده است. در صورتی که $R3$ را به صورت فرضی در نظر گرفته باشید باز نیز مورد قبول است.

a) نوع آدرس دهی در این بخش: **Pre-indexed addressing mode with fixed offset**

در این حالت مقدار $R3$ با $R4$ جمع می‌شود و $R9$ در آدرس حاصل جمع ذخیره می‌شود:

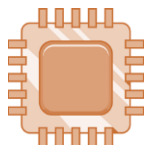
$$R3 + R4 = 0x4020$$

$$\text{Memory Address } 0x4020 = 0x44$$

$$\text{Memory Address } 0x4021 = 0x33$$

b) نوع آدرس دهی در این بخش: **Pre-indexed addressing mode with fixed offset of shifted register**

در این حالت مقدار $R4$ سه بار به چپ شیفت منطقی می‌یابد و سپس با $R3$ جمع می‌شود و از آدرس حاصل، یک بایت خوانده شده و در $R8$ ذخیره می‌شود:



LSL #3: $R4 = 0x100$
 $R3 + \text{Shifted } R4 = 0x4100$
Final Memory Address: $0x4100$
 $R8 = 0x000000FF$

(c) نوع آدرس دهی در این بخش: **Post-indexed addressing mode with fixed offset**

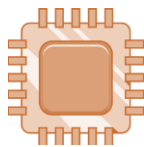
در این حالت ابتدا محتوای آدرس حافظه‌ای که $R3$ اشاره می‌کند خوانده می‌شود و در $R7$ ذخیره می‌شود. سپس مقدار $R3$ با $R4$ جمع می‌شود و در $R3$ ذخیره می‌شود:

Memory Address = $0x4000$
 $R7 = 0xFFFFFFFF$
 $R3 = R3 + R4 = 0x4020$

(d) نوع آدرس دهی در این بخش: **Scaled register post-indexed**

در این حالت ابتدا یک بایت از محتوای $R6$ در خانه‌ای که رجیستر $R3$ به آن اشاره می‌کند ذخیره می‌شود سپس مقدار $R4$ دو بار به راست شیفت حسابی می‌یابد و سپس با $R3$ جمع می‌شود و حاصل در $R3$ ذخیره می‌شود:

Memory Address = $0x4000$
Content of memory address = $0x44$
 $R4 \text{ ASR } \#2 = 0x08$
 $R3 = R3 + (R4 \text{ ASR } \#2) = 0x4008$



6) 4 مورد از قوانین استاندارد AAPCS برای پیاده‌سازی توابع را نام ببرید

پاسخ:

- آرگومان‌های تابع باید از طریق رجیسترهای R0 تا R3 فرستاده شوند.
- مقدار بازگشتی باید در R0 (و R1 اگر مقدار 64 بیتی است) قرار گیرد.
- توابع می‌توانند از رجیسترهای R4 تا R8 و R10 و R11 برای ذخیره اطلاعات موقت استفاده کنند. البته مقادیر این رجیسترها هنگام ورود باید ذخیره شود و قبل از بازگشت بازگردانی شوند.
- استک باید به صورت full ascending استفاده شود.