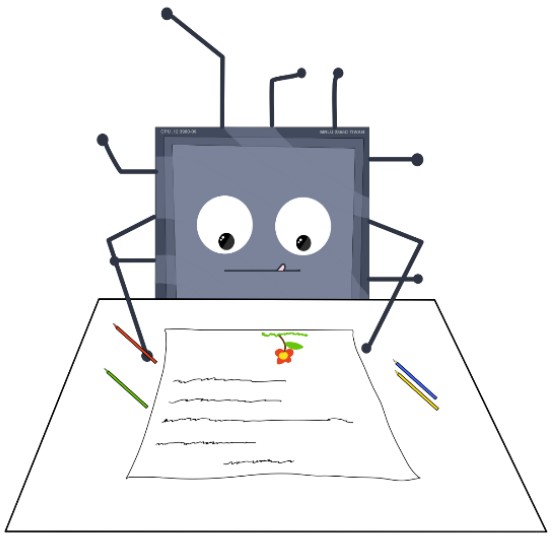
Department of Computer Engineering

Microprocessors and Assembly Language, Spring 2023, Dr. Farbeh

Homework 3 – Solutions

Lec 11-22



سوال ۱:

مبدلی در اختیار داریم که دما را به ولتاژی در بازه [۱۹, ۱۹-] تبدیل می‌کند. اگر خطای کمی سازی(Quantization Error) حدود ۰.۴۹ درصد باشد، تعداد گام‌ها و تعداد بیت‌هایی که برای نمایش مقادیر آنالوگ به صورت دیجیتال به کار رفته است را به دست آورید.

پاسخ:

تعداد گام‌ها:

$$Steps = (0.49 \div 100) \times 2 = 0.0098$$

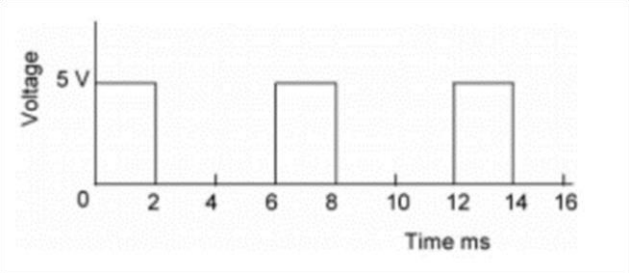
تعداد بیت‌ها:

$$Steps = Range \div 2^n \rightarrow 0.0098 = \big(19 - (-19)\big) \div 2^n \rightarrow 2^n = 38 \div 0.0098 = 3877.55$$
$$\rightarrow n = 12$$

سوال ۲:

در رابطه با PWM به سوالات زیر پاسخ دهید:

الف) شکل زیر نمودار ولتاژ اعمال شده به یک PWM بر حسب زمان را نشان می‌دهد. duty cycle را محاسبه کنید.



ب) مزایای تکنیک PWM بر استفاده از مبدل دیجیتال به آنالوگ چیست؟

پاسخ:

الف)

$$Duty\ Cycle: \left(\frac{2}{6}\right) \times 100 = 33.33\%$$

ب)

۱. محدودیت تعداد مبدل‌های دیجیتال به آنالوگ در میکروکنترلر(به عنوان مثال، برای مدیریت تعداد زیادی موتور توسط میکروکنترلر، به خروجی‌های متعدد آنالوگ نیازمندیم.)

۲. مبدل‌های دیجیتال به آنالوگ تاخیر دارند و در برخی کاربردها نیازمند عمل موتور (مانند تغییر و تنظیم سرعت، جهت و ...) در لحظه می‌باشیم و استفاده از مبدل به علت تاخیر ممکن نیست.

۳. یک مبدل دیجیتال به آنالوگ قادر به تولید هر مقدار آنالوگی نمی‌باشد (خطایQuantization) و به تعداد مقادیر دیجیتال موجود می‌تواند مقدار آنالوگ تولید کند.

سوال ۳:

بعد از اجرای قطعه کد زیر، ثبات R0 در چه مقداری ضرب شده‌است؟

```
ADD  R0, R0, R0, LSL #3
RSB   R0, R0, R0, LSL #2
```

پاسخ:

ابتدا مقدار R0 به دلیل ۳ واحد شیفت به چپ، در ۸ ضرب می‌شود. سپس با خودش جمع می‌شود که تا این‌جا می‌شود ۹ تا R0.
سپس باز ۲ واحد به چپ(۴ برابر) شیفت می‌دهیم . که برابر با ۳۶ = ۹*۴ تا R0 می‌شود. سپس ۹ تا مقدار اولی از آن کم می‌شود.
در نهایت مقدار آن در ۲۷ ضرب شده‌است.

سوال ۴:

جدول زیر مقادیر شماری از واژه‌های حافظه (word memory) را نشان می‌دهد.

Memory Word Address	Memory Content (Word)
0x3000201C	0x00000000
0x30002018	0xFFFFFFFF
0x30002014	0x3000201C
0x30002010	0x00000000
0x3000200C	0x30002014
0x30002008	0x00000080
0x30002004	0x70605040
0x30002000	0x30201000

فرض کنید که رجیسترهای آورده‌شده در زیر نیز مقدارهای نمایش داده شده را دارند.

r0 = 0x30002000
r1 = 0x00000004
r2 = 0x0000FFFF
r9 = 0x3000200C
r13(sp) = 0x30002008

در این صورت بگویید با اجرای هر یک از دستورهای زیر نتیجه چیست؟ توجه کنید که دو قسمت سوال کاملا از هم جدا هستند؛ یعنی تغییرات در مقدار رجیسترها یا محتوای حافظه مستقل است؛ مگر در رجیستر PC.

الف) `LDRSB r9, [r0, #4]!`

ب) `STRH r2, [r0, r1, LSL #2]`

پاسخ:

الف) با دستور اول از آدرس (r0 + 4 = 0x30002004) یک بایت به صورت علامت‌دار در رجیستر r9 لود می‌شود و سپس مقدار خود رجیستر r0، به 0x30002004 تغییر داده می‌شود.
مقدار عدد در آدرس حافظه 0x30002004 ، ۰۱۱۱۰۰۰۰۱۱۰۰۰۰۰۰۱۰۱۰۰۰۰۰۱۰۰۰۰۰۰ است. پس ۸ بیت سمت راست به صورت علامت دار لود میشود:

R9 = 00000000 00000000 00000000 01000000

ب) در ابتدا مقدار r1 دوبار به سمت چپ شیفت داده می‌شود (معادل با ضرب در ۴). پس برابر با 0x10 می‌شود. سپس این مقدار با r0 جمع می‌گردد و نتیجه برابر است با 0x30002010 ، سپس ۲ بایت سمت راست r2 باید در این آدرس ذخیره شود. پس مقدار 0x0000FFFF در این آدرس ذخیره می‌شود.

سوال ۵:

کد اسمبلی بنویسید که R1 را به توان R0 برساند و حاصل را در R2 ذخیره کند.

(می‌دانیم حاصل از ۳۲ بیت بیش‌تر نمی‌شود و در نظر بگیرید که در عملیات توان، پایه مثبت و توان غیرمنفی است.)

پاسخ:

```
power:
        PUSH    {R2, LR}        ; save R2 and LR on the stack
        MOV     R2, #1          ; initialize the result to 1
        CMP     R0, #0          ; check if exponent is 0
        BEQ     end             ; if exponent = 0, exit loop
        MOV     R4, R0          ; copy exponent to R4
loop:
        CMP     R4, #0          ; check if exponent is 0
        BEQ     end             ; if exponent = 0, exit loop
        MUL     R4, R4, R1      ; multiply result by base
        SUBS    R4, R4, #1      ; decrement exponent
        B       loop            ; repeat loop

end:
        POP     {R4, PC}        ; restore R4 and PC from the stack
```

سوال ۶:

پانزده عدد از خانه‌ی 0x04000000 شروع می‌شوند. با فرض اینکه جمع آن‌ها از یک Word بیشتر نمی‌شود، برنامه‌ای به زبان اسمبلی بنویسید که تمام آن‌ها را با هم جمع کند و در رجیستر R2 بریزد.

پاسخ:

```
1           LDR         R0, =0x04000000
2           MOV         R2, #0
3           MOV         R3, #15 ; R3 is loop counter
4           STR         R3, [R0] ; storing a number in memory
5           MOV         R4, #3 ; preparing another number
6           STR         #3, [R0, #4] ; stored on the next word
7
8 HERE
9           LDR         R1, [R0]
10          ADD         R2, R2, R1
11          ADD         R0, R0, #4 ; going to next word
12          SUBS        R3, R3, #1 ; decrease counter
13          BNE         HERE ;branch if not equal
14
15          END
```

سوال ۷(امتیازی):

تابع فاکتوریل را بصورت بازگشتی بنویسید. مقدار عدد داده شده در رجیستر R0 ذخیره شده است و مثبت می باشد.

پاسخ:

```
;R0 parameter
factorial
        PUSH {R1, R2, LR}
        CMP R0, #1
        BEQ ret ; return 1
        MOV R1, R0 ; kopiera R0 till R1
        SUB R0, R0, #1 ; dec R0
        BL factorial ; anropa factorial
        MUL R2, R1, R0 ; int result = fact(n-1)*n
        MOV R0, R2 ; R0 = result
        POP {R1, R2, LR} ; return
ret
        MOV R0, #1 ;return 1
        POP {R1, R2, LR} ; return
```