به نام خدا

دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر

آزمایشگاه ریزپردازنده

گزارش‌کار آزمایش اول اسمبلی

آشنایی با شیوه پیکربندی رجیستر ها و مدیریت سطح پایین با زبان اسمبلی

استاد درس: مهندس معصوم زاده

محمّد امین رضائی/ ۹۷۳۱۰۲۴

مهدی رحمانی/ ۹۷۳۱۷۰۱

«پیش گزارش»

پرسش اول: توضیحات مختصری درباره ی دستورات LDR ، MOV و STR بدهید:

:LDR R0, [R1]

دیتای مستقر در یک آدرس را درون رجیستر مقصد بارگذاری میکند. براکت های اطراف R1 حاکی از این است که رجیستر یک آدرس دارد. ما با استفاده از پرانتز ها دیتا را به جای خود آن آدرس در آدرس R0 قرار میدهیم. همچنین میتوانیم از این علامت گذاری برای قرار دادن دیتای افست از یک آدرس معین استفاده کنیم.

:MOV R0, R1

این دستور دیتا را از جایی به جای دیگر منتقل میکند. هم محتوای درون ثبات ها و هم مستقیم و بلافاصله خود دیتا را میتوان با کمک این دستور درون ثبات اول ریخت.

:STR R1, R[0]

ذخیره سازی یک عملیات مکمل را برای بارگیری انجام میدهد. ذخیره سازی، محتویات رجیستر را در محل حافظه قرار میدهد. کد زیر دیتای درون R1 را در آدرس R0 ذخیره میکند. پرانتز ها باز هم نشان دهنده ی این هستند که R0 یک آدرس دارد و ما میخواهیم دیتای آن آدرس را تغییر دهیم.

تمامی دستور بالا میتواند حالت های مختلف مانند شرطی داشته باشد.

پرسش دوم: ایده ای برای پیاده سازی تابع تاخیر در زبان اسمبلی ارایه دهید:

ایده پیاده سازی تاخیر به کمک یک counter میباشد. میدانیم که هر دستور مثل ADD یا SUB و... یک مدت زمانی برای اجرا و fetch شدن لازم دارند. حال اگر برای مثال از 0 تا یک مقدار زیادی به کمک یک counter بشماریم میتوانیم تا حدی تاخیر ایجاد کنیم. با کمک دستور ADD و مقایسه دو ثبات و بعد Branch کردن میتوان تابع delay را تا حدی پیاده سازی کرد. به عنوان مثال کد زیر:

```
delay_loop
    ADD R4, R4, #1
    CMP R4, R5
    BNE delay_loop
    BX LR
```

« گزارش کار»

هدف ازمایش:

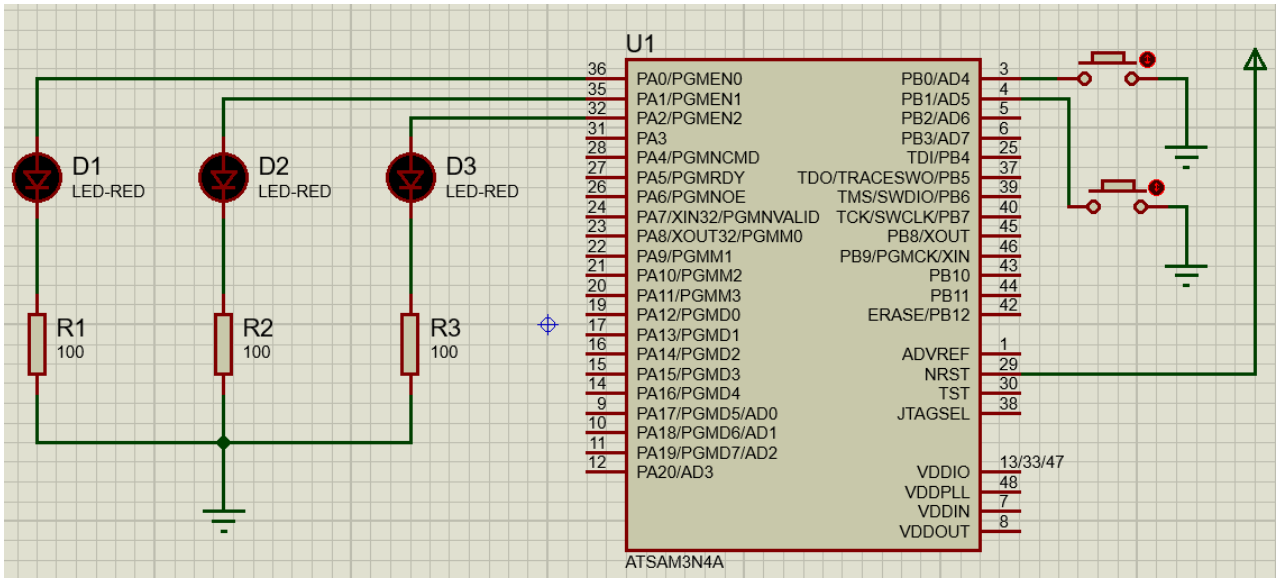
- آشنایی با شیوه پیکربندی رجیستر ها و مدیریت سطح پایین با زبان اسمبلی

قطعات آزمایش:

- میکروکنترلر ATSAM3N4A
- دیود نورانی LED
- کلید
- مقاومت 220Ω
- مقاومت 10KΩ

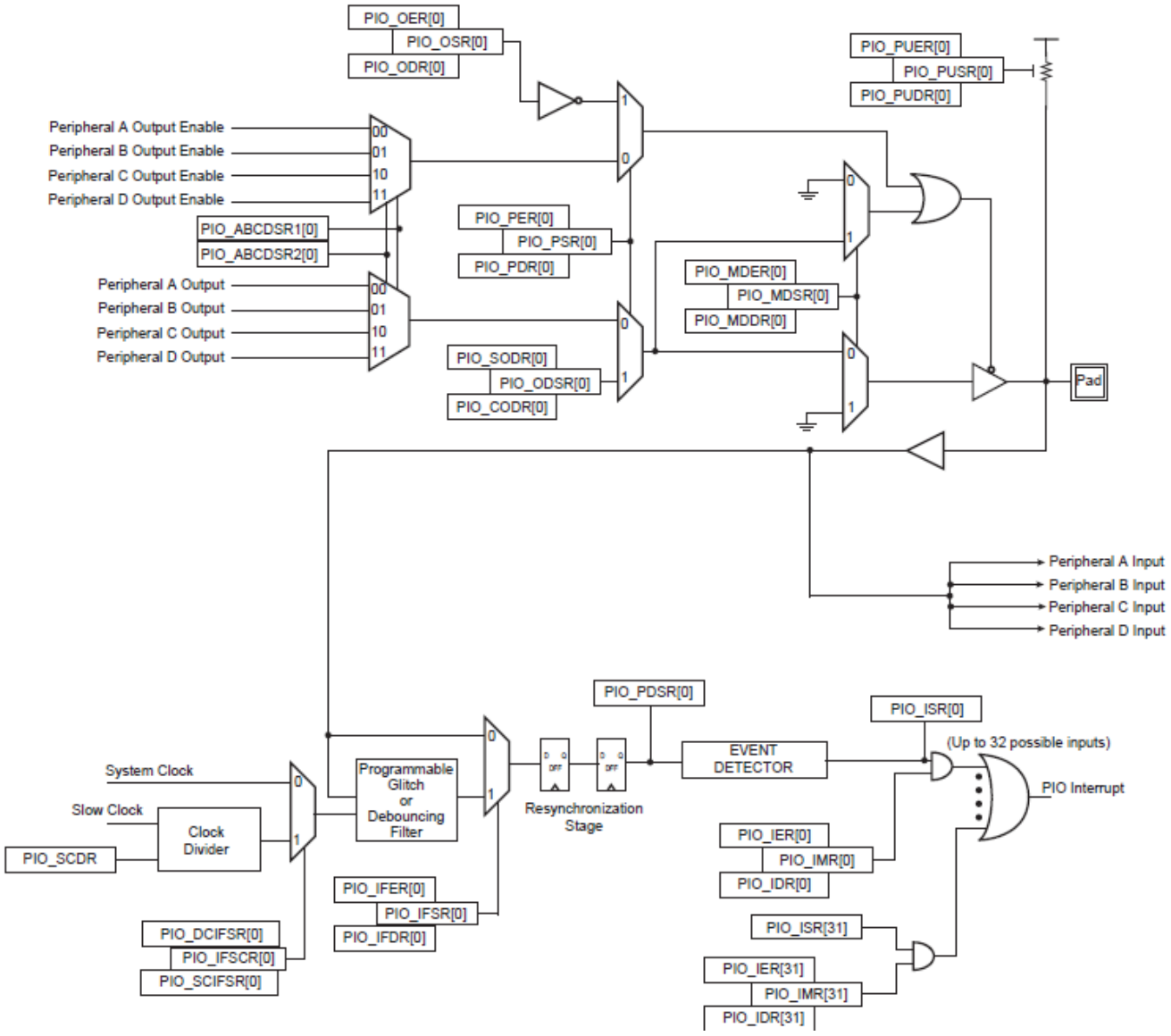
در این آزمایش ما میخواهیم به کمک بورد ATSAM3N4A، ۳ ال ای دی را با کمک زبان اسمبلی کنترل کنیم و با ۳ تاخیر مختلف آنها را چشمک زن کنیم و بتوانیم آنها را خاموش کنیم.

ابتدا مدار را به صورت زیر میبندیم.



همانطور که در شکل هم مشاهده میشود LEDهای 1 و 2 و 3 را به ترتیب به پین های 0 و 1 و2 از PIOA متصل میکنیم. همچنین pushButton1 را به پین 0 از PIOB متصل میکنیم. زمانی که فشرده شود باید LED ها شروع به چشمک زدن بکنند و همچنین pushButton2 را نیز به پین 1 از PIOB وصل کردیم که زمانی که فشرده شود باید LED ها خاموش شوند.

حال با کمک تصویر زیر از دیتاشیت بورد مورد استفاده کار را ادامه میدهیم:



در ادامه به دو روش کد زدیم و هرکدام را توضیح میدهیم.

روش اول)

ابتدا یک سری دیتا داریم و اسم گذاری آن‌ها را باید به کمک directive های EQU و RN انجام دهیم.در شکل زیر این قسمت را میتوانید مشاهده کنید که در کد با یک AREA به اسم myData مشخص شده است:

```
    AREA myData, DATA
PIO_PER EQU 0x400E0E00      ;(PIOA)-> PIO Enable Register
PIO_OER EQU 0x400E0E10      ;(PIOA)-> PIO Output Enable Register
PIO_SODR EQU 0x400E0E30     ;(PIOA)-> PIO Set Output Data Register
PIO_CODR EQU 0x400E0E34     ;(PIOA)-> PIO Clear Output Data Register
PIO_ISR EQU 0x400E104C      ;(PIOB)-> PIO Interrupt Status Register
;determin the delay value of LEDs
LED1_delay EQU 0x000C0000  ;set the delay value for LED1(left LED)
LED2_delay EQU 0x00030000  ;set the delay value for LED2(middle LED)
LED3_delay EQU 0x0000A000  ;set the delay value for LED2(right LED)
;assign a name to some registers
LED1_toggle RN R8          ;hold the toggle value of LED1 in R8(if R8==1 : LED1->on // if R8==0 : -> LED1->off)
LED2_toggle RN R9          ;hold the toggle value of LED1 in R8(if R9==1 : LED1->on // if R9==0 : -> LED2->off)
LED3_toggle RN R10         ;hold the toggle value of LED1 in R8(if R10==1 : LED1->on // if R10==0 : -> LED3->off)
```

در همین بخش دیتا در گام اول آدرس رجیسترهای مربوط به تنظیمات گرفتن ورودی و دادن خروجی را باید به یک اسم های مرتبط با آن‌ها assign کنیم تا در ادامه راحت تر با آن ها کار کنیم.

بخش دوم قسمت کد میباشد. این قسمت در کد اسمبلی ما با یک AREA به نام myCode مشخص شده است. در آن جا نیز مشخص کردیم تابع شروع و اصلی ما چیست و آن را export کردیم:

```
15
16      AREA myCode, code, READONLY
17      export __main
18      entry
```

همچنین لازم است یک MACRO تعریف کنیم تا به عنوان ورودی 4 تا پارامتر میگرد و تصمیم میگیرد که اکنون اصلا زمان toggle کردن LED هست یا خیر و اگر بود براساس ورودی ها کدام LED را خاموش یا روشن کند.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;define a MACRO for managing the on or off process of each LED
;$ARG1-> select wich pio must be on or off
;$ARG2-> delay of related LED
;$ARG3-> register that hold the delay value of related LED and we want to restor it
;$ARG4-> register that hold the toggle value(we understand the led would be on or off)
    MACRO
$lbl myhandler $ARG1, $ARG2, $ARG3, $ARG4
    BEQ $lbl.continue                   ;SUBS set the flags. if the result of sub was 0 we can go to continue label of macro
    B $lbl.finish                       ;if the sub result wasn't 0 we should exit from MACRO
$lbl.continue
    LDR $ARG3, =$ARG2                   ;restore the register that holds delay value
    EOR $ARG4, $ARG4, #1                ;xor the LED#_toggle with 1 for handling toggle
    CMP $ARG4, #1                       ;compare LED#_toggle with 1
    BEQ $lbl.led_on                     ;if LED#_toggle==1 : turn on the LED#
    B $lbl.led_off                      ;if LED#_toggle==0 : turn on the LED#
$lbl.led_on                             ;*** (this section set the SODR register to turn on the LED)***;
    MOV R4, $ARG1
    LDR R5, =PIO_SODR
    STR R4, [R5]
    B $lbl.finish
$lbl.led_off                            ;*** (this section set the CODR register to turn off the LED)***;
    MOV R4, $ARG1
    LDR R5, =PIO_CODR
    STR R4, [R5]
$lbl.finish
    MEND

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

در قسمت main__ یک سری مقدار دهی باید انجام شود. مثلا در رجیسترهای مربوط به toggle کردن در وهله اول مقدار 1 میریزیم به این معنی که در شروع کار LEDها روشن هستند و موقع toggle کردن میبینیم چون مقدار رجیستر مربوطه 1 هست پس اکنون که زمان toggle رسیده وچون مقدار آن رجیستر 1 هست باید LED خاموش شود . همچنین مقادیر تعیین شده برای delay هر LED را نیز باید در یک رجیستر نگه داریم که ما رجیسترهای R1 و R2 و R3 را برای این کار انتخاب کردیم:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
__main

    MOV LED1_toggle, #1                 ;initialize R8(LED1_toggle) with 1
    MOV LED2_toggle, #1                 ;initialize R9(LED2_toggle) with 1
    MOV LED3_toggle, #1                 ;initialize R10(LED3_toggle) with 1

    LDR R1, =LED1_delay                 ;initialize R1 with LED1_delay
    LDR R2, =LED2_delay                 ;initialize R2 with LED2_delay
    LDR R3, =LED3_delay                 ;initialize R3 with LED3_delay

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

سپس بعد از اینکه مقدار دهی ها انجام شد وارد section بعدی میشود. در این قسمت ما باید دائما چک کنیم تا درصورتی که کاربر pushButton1 را فشار داد ما به قسمت بعدی برویم و فرآیند چشمک زدن LEDها را اجرا کنیم. برای این کار لازم است تا مقداری که دربیت متناظر با پین متصل به آن دکمه در PIO_ISR هست را بخوانیم و براساس آن تصمیم بگیریم که از به section بعدی برویم یا خیر. کد مربوطه به صورت زیر است:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
pushButton1

    LDR R6, =PIO_ISR            ;load the address of PIO_ISR reg in R6
    LDR R4, [R6]                ;load the value of PIO_ISR reg in R4
    AND R5, R4, #2_1            ;check the value in first bit of PIO_ISR in PIOB
    CMP R5, #0                  ;compare that bit with 0
    BEQ pushButton1             ;if R5==0(if the button isn't pushed) we should wait and check interrupt again
    BL enable_pio               ;if the button is pushed we should enable the pio pins
    BL leds_on                  ;then we should turn on all of LEDs

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

پس از اینکه فرد دکمه اول را فشار داد باید وارد مرحله بعدی شویم که کد آن در قسمت mainLoop آمده است. از قبل ما تاخیر هر کدام از LEDها را در یک رجیستر نگه داشته بودیم. حالا برای ایجاد تاخیر باید یک واحد یک واحد از مقدار هرکدام از رجیسترها کم کنیم (و این فرآیند هربار زمانی میگیرد و موجب ایجاد تاخیر میشود) و زمانی که مقدار رجیستر مربوطه 0 شد باید اگر LED روشن است خاموش شود و بالعکس. همچنین باید چک کنیم تا هرجایی کاربر pushButton2 را فشار داد تمامی LEDها را خاموش کنیم و دوباره از اول شروع کنیم و منتظر فشردن دکمه1 بمانیم. کد مربوطه به صورت زیر است:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
mainLoop

    BL pushButton2                          ;we should check push button2 interrupt
    SUBS R1, R1, #1                         ;a downward counter for creating delay for LED1
LED1 myhandler #2_1, LED1_delay, R1, LED1_toggle  ;call macro and if the result of SUBS is 0 the flags are set and do a suitable job related to LED1

    BL pushButton2                          ;we should check push button2 interrupt
    SUBS R2, R2, #1                         ;a downward counter for creating delay for LED2
LED2 myhandler #2_10, LED2_delay, R2, LED2_toggle ;call macro and if the result of SUBS is 0 the flags are set and do a suitable job related to LED2

    BL pushButton2                          ;we should check push button2 interrupt
    SUBS R3, R3, #1                         ;a downward counter for creating delay for LED3
LED3 myhandler #2_100, LED3_delay, R3, LED3_toggle;call macro and if the result of SUBS is 0 the flags are set and do a suitable job related to LED3

    B mainLoop                              ;else we should continue to our work and return to mainLoop

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

برای اینکه چک کنیم آیا دکمه 2 فشار داده شده است یا نه باید بیت مربوط به پینی که این دکمه به آن متصل است را در رجیستر PIO_ISR بخوانیم و براساس آن اگر ورودی آمده بود باید تمامی LEDها را خاموش کنیم. این کد در زیر آمده است:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
pushButton2                        ;***(this section checks if an interrupt comes from pushButton2 we shoud turn LEDs off)***;

    LDR R6, =PIO_ISR               ;load the address of PIO_ISR reg in R6
    LDR R4, [R6]                   ;load the value of PIO_ISR reg in R4
    AND R5, R4, #2_10              ;check the value in second bit of PIO_ISR in PIOB
    CMP R5, #2_10                  ;compare that bit with 0
    BEQ leds_off                   ;if R5==2_10(if the button is pushed) we should turn off the LEDs
    BX LR

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

همچنین در قسمت های گذشته به یک سری label ها میپریدیم که کد آن ها در زیر آمده است.

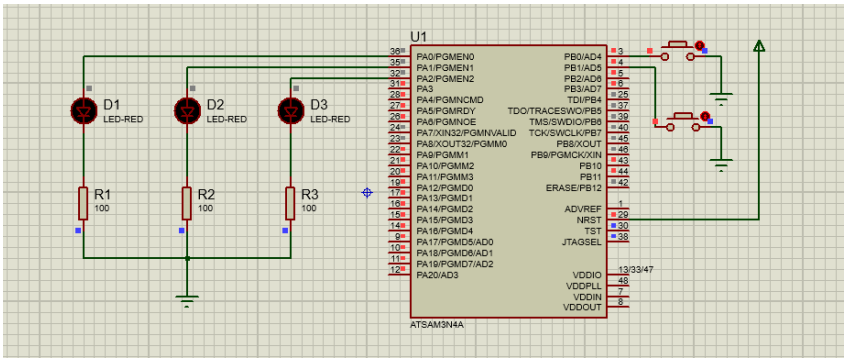
یک بخش مربوط به enable کردن پین های PIO ما میباشد که به LED ها متصل اند. باید زمانی که کاربر دکمه 1 را فشار داد پیش از هرکاری به این قسمت BL کنیم. برای اینکار چون 3 تا LED را به ترتیب به پین های 0 و 1 و 2 از PIOA متصل کردیم باید مقدار 2_111 را در رجیسترهای PIO_PER و PIO_OER بنویسم. در کد زیر آمده است:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
enable_pio                         ;***(this section enable the PIOs that we want to use)***;

    MOV R4, #2_111                 ;move the value 2_111 to R4

    LDR R5, =PIO_PER               ;load the address of PIO_PER reg in R5
    STR R4, [R5]                   ;store 2_111 in PIO_PER

    LDR R5, =PIO_OER               ;load the address of PIO_OER reg in R5
    STR R4, [R5]                   ;store 2_111 in PIO_OER

    BX LR                          ;retun to location that we were first(brach to where link reg is point to that)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

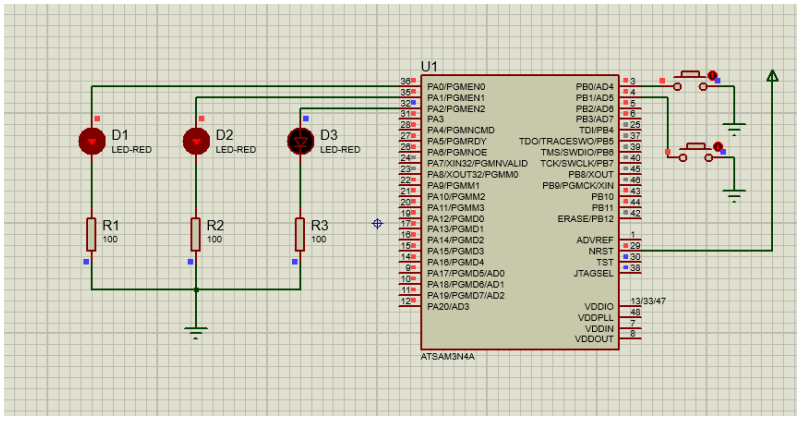
بخش دیگری هم که مانده است مربوط به روشن کردن همه LEDها باهم و خاموش کردن تمام LED ها باهم هست. زمانی که کاربر دکمه 1 را فشار داد پس از enable کردن PIO ها باید تمامی LED ها را همزمان شروع کنیم پس باید به leds_on ، BL کنیم. زمانی هم که کاربر دکمه 2 را فشار میدهد باید تمامی LEDها خاموش شود. برای این کار باید به leds_off برنچ کند. کد مربوط به این دو در زیر آمده است:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
leds_on                                 ;***(this section turn on all LEDs)***;
                                        ;(NOTE) we call this if the push button1 is pushed
    MOV R4, #2_111                      ;move the value 2_111 to R4
    LDR R5, =PIO_SODR                   ;load the address of PIO_SODR reg in R5
    STR R4, [R5]                        ;store 2_111 in PIO_SODR
    BX LR                               ;retun to location that we were first(brach to where link reg is point to that)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
leds_off                                ;***(this section turn off all LEDs)***;
                                        ;(NOTE) we call this if the push button2 is pushed
    MOV R4, #2_111                      ;move the value 2_111 to R4
    LDR R5, =PIO_CODR                   ;load the address of PIO_SODR reg in R5
    STR R4, [R5]                        ;store 2_111 in PIO_SODR
    B __main                            ;return to __main and wait for comming an interrupt from push button1

    END
```

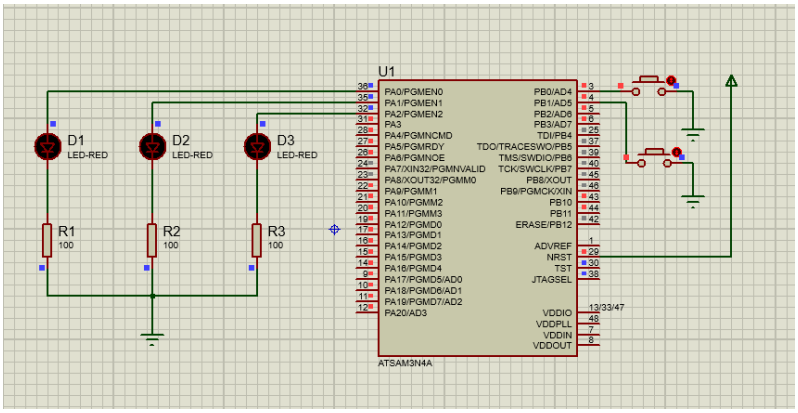
حال از این کد یک فایل hex. میسازیم و آن را روی برد آپلود میکنیم. اگر اجرا کنیم در شروع کار LEDها خاموش اند و منتظرند که دکمه 1 فشرده شود:



سپس اگر دکمه 1 را بزنیم با توجه به تاخیرهایی که در کد دادیم از چپ به راست سرعت چشمک زدن بیشتر میشود. در ابتدای کار هم هر3 باهم روشن میشوند: (البته بدیهی است که چشمک زدن را نمیتوان در تصویر نمایش داد. ☺)



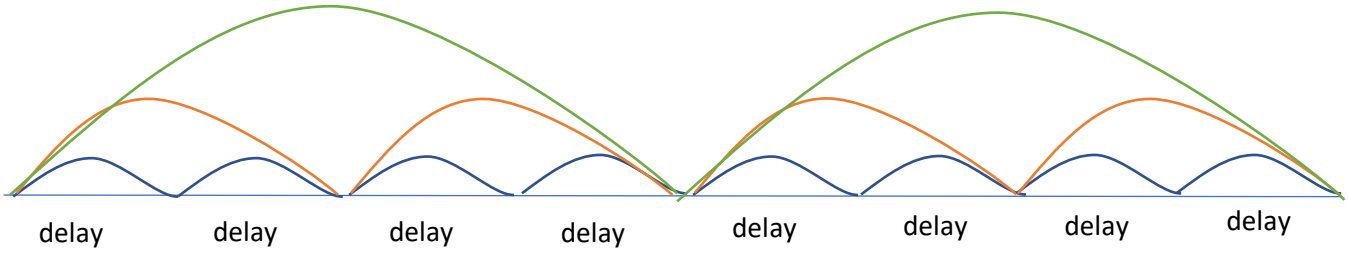
همچنین وقتی دکمه 2 را فشار میدهیم همه LEDها باهم خاموش میشوند و باز منتظر میمانند تا دکمه 1 فشار داده شود:



روش دوم)

در این روش ما یم delay ثابت تعریف کردیم و براساس اینکه 3 تا LED داریم 8 تا فاز خواهیم داشت تا LED ها در یک دوره خاموش و روشن شوند و دوباره این موضوع تکرار شود

LED3 و LED2 و LED1



در اینجا ما با چند PIO کار داریم که در کد باید آنها را تنظیم کنیم تا بتوانیم آزمایش را پیاده کنیم. آدرس آنها در حافظه به این صورت میباشد:

```
PIO_PER EQU 0x400E0E00  ;(PIOA)
PIO_OER EQU 0x400E0E10  ;(PIOA)
PIO_SODR EQU 0x400E0E30 ;(PIOA)
PIO_CODR EQU 0x400E0E34 ;(PIOA)
PIO_ISR EQU 0x400E104C  ;(PIOB)
```

با کمک دستور EQU آدرس های مربوطه را درون یک متغیر میریزیم که هم نام PIO مورد نظر میباشد.

حال کد را بررسی میکنیم:

```
 7      AREA myCode, code, READONLY
 8      export __main
 9      entry
10
11  __main
12      LDR R6, =PIO_ISR
13      LDR R2, [R6]
14      AND R3, R2, #2_1
15      CMP R3, #0
16      BEQ __main
17      BL enable_pio
18
```

در ان بخش ورودی را گرفته و چک میکنیم که اگر نداشتیم به خودش برنچ بزند و همینطور ادامه دهد تا یک ورودی برسد. اینکار با مقایسه ورودی با یک انجام میشود.

```
45  enable_pio
46      MOV R4, #2_111
47
48      LDR R5, =PIO_PER
49      STR R4, [R5]
50
51      LDR R5, =PIO_OER
52      STR R4, [R5]
53
54      BX LR
55
```

در بخش بالا پس از ورودی گرفتن به اینجا برنچ میشود و در اینجا مقادیر اولیه PIO برای راه اندازی داده میشود و سپس به link register برنچ میزنیم.

```
18
19  mainLoop
20      BL led1_on
21      BL led2_on
22      BL led3_on
23      BL delay
24      BL led3_off
25      BL delay
26      BL led3_on
27      BL led2_off
28      BL delay
29      BL led3_off
30      BL delay
31      BL led3_on
32      BL led2_on
33      BL led1_off
34      BL delay
35      BL led3_off
36      BL delay
37      BL led3_on
38      BL led2_off
39      BL delay
40      BL led3_off
41      BL delay
42
43      B mainLoop
```

حال وارد خط ۱۸ میشود و در این mainloop ما با کمک برنچ زدن به خطوطی که توابع مورد نظر نوشته شدند، کار تاخیر و روشن شدن را انجام میدهیم و بعد اجرای یک دور خط ۲۰ تا ۴۲، دوباره از اول اینکار را تکرار میکنیم و کار چشمک زدن را ادامه میدهیم.

```
56  led1_on
57      MOV R4, #2_1
58      LDR R5, =PIO_SODR
59      STR R4, [R5]
60      BX LR
61
62  led1_off
63      MOV R4, #2_1
64      LDR R5, =PIO_CODR
65      STR R4, [R5]
66      BX LR
67
68  led2_on
69      MOV R4, #2_10
70      LDR R5, =PIO_SODR
71      STR R4, [R5]
72      BX LR
73
74  led2_off
75      MOV R4, #2_10
76      LDR R5, =PIO_CODR
77      STR R4, [R5]
78      BX LR
79
80  led3_on
81      MOV R4, #2_100
82      LDR R5, =PIO_SODR
83      STR R4, [R5]
84      BX LR
85
```

```
86  led3_off
87      MOV R4, #2_100
88      LDR R5, =PIO_CODR
89      STR R4, [R5]
90      BX LR
91
92  leds_off
93      MOV R4, #2_111
94      LDR R5, =PIO_CODR
95      STR R4, [R5]
96      B __main
97
98  delay
99      ; first we should check push button 2 interrupt
100     LDR R2, [R6]
101     AND R3, R2, #2_10
102     CMP R3, #2_10
103     BEQ leds_off
104     ; set the registers for create delay
105     MOV R4, #0
106     LDR R5, =0x00060000
107
108 delay_loop
109     ADD R4, R4, #1
110     CMP R4, R5
111     BNE delay_loop
112     BX LR
113
114     END
```

در بخش بالا هم ما با مقداری دهی مناسب PIO ها در حافظه، ال ای دی ها را متناسب با ورودی یا چشمک زن میکنیم با سرعت های مختلف یا به کل خاموش میکنیم.

یاعلی