فصل ۵

حلقه‌های تکرار[1]

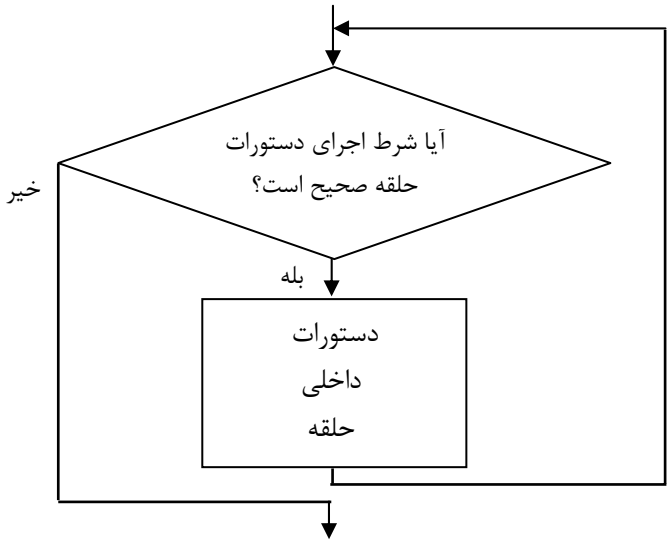
در فصل قبل با برنامه‌نویسی شرطی و نحوه تصمیم‌گیری در برنامه‌های اسمبلی آشنا شدیم. یکی از مواردی که در نوشتن برنامه‌ها بسیار اتفاق می‌افتد، تکرار دستورات مشابه است که به اصطلاح حلقه یا لوپ نامیده می‌شود. مثلاً در سیستم خودپرداز بانک، چنانچه رمز کاربر سه مرتبه به اشتباه وارد شود، کارت ضبط می‌شود؛ پس عمل دریافت رمز و کنترل صحت آن باید سه مرتبه اجرا شود. برای انجام این کار، دستورات فوق را در یک حلقه که سه مرتبه اجرا شود قرار می‌دهیم. در این فصل با حلقه‌ها و مورد خاص و مهم کاربرد آن در ایجاد تأخیر آشنا خواهیم شد.

انواع حلقه‌ها

شکل ۵-۱ ساختار کلی حلقه‌ها را نشان می‌دهد.

حلقه‌ها از دیدگاه تعداد تکرار دستورات به دو دسته تقسیم می‌شوند:

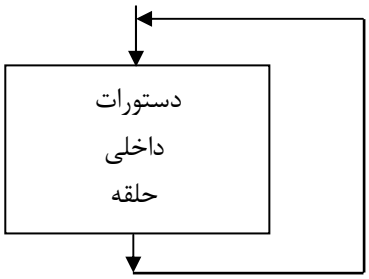
حلقه‌های **تکرار نامعین**، حلقه‌هایی هستند که تعداد تکرار دستورات داخلی آنها مشخص نیست؛ شکل.... را ببینید.



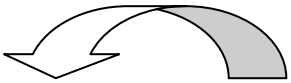
هربار در ابتدای حلقه، شرطی کنترل می‌شود و در صورت صحت آن، دستورات داخل حلقه تکرار می‌شوند.

حالت خاصی از حلقه‌های تکرار نامعین به حلقه‌های تکرار بینهایت[1] معروفند که دستورات داخل آنها بینهایت بار (تا قبل از خاموش شدن میکروکنترلر) اجرا می‌شوند.

[1] Loops

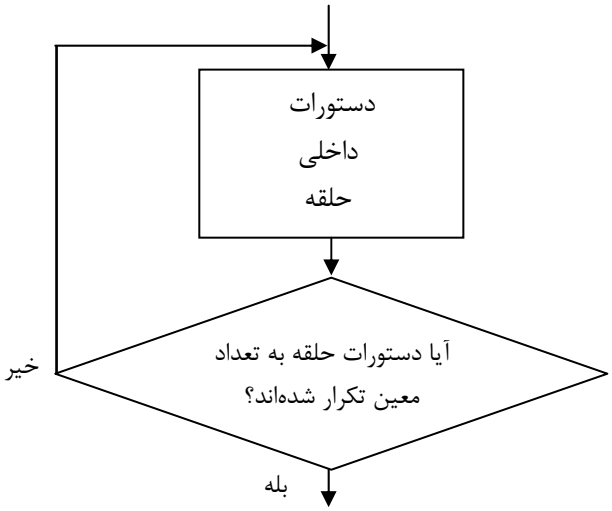


حلقه‌های تکرار نامعین با استفاده از دستورات پرش شرطی پیاده می‌شوند؛ مثلاً دستورات زیر تا زمانی که پین P1.0 «یک» باشد، این پین را می‌خواند:



```
AGAIN:
        JB      P1.0, AGAIN
```

حلقه‌های **تکرار معین**، حلقه‌هایی هستند که دستورات داخلی آنها به تعداد معین تکرار می‌شوند. ساختار کلی حلقه‌های تکرار معین به شکل زیر است:



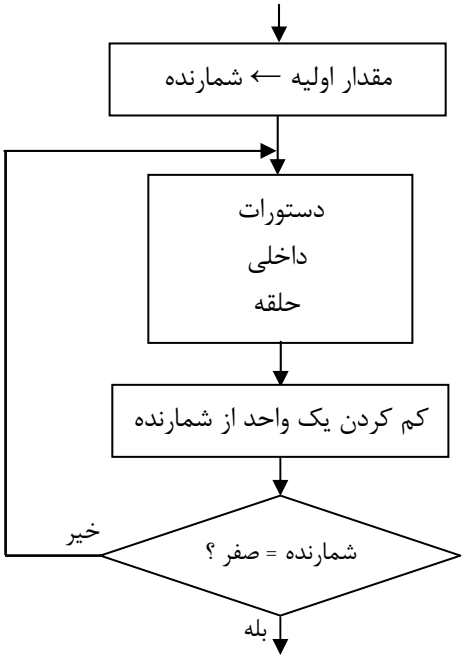
همانطور که می‌بینید، با هربار اجرای دستورات حلقه، میکروکنترلر بررسی می‌کند آیا حلقه به تعداد معین تکرار شده یا خیر و در صورت لزوم دستورات را تکرار می‌کند.

در این فصل به حلقه‌های تکرار معین و کاربرد مهم آنها در ایجاد تأخیر می‌پردازیم.

[1] Infinite Loops

حلقه‌های تکرار معین

ساختار کلی حلقه‌های تکرار معین به شکل زیر است:



شمارنده[1] یکی از ثباتهای میکروکنترلر است که عمل شمارش تعداد تکرار دستورات حلقه را بر عهده دارد و قبل از ورود به حلقه مقداردهی می‌شود. با هربار اجرای حلقه یک واحد از شمارنده کم می‌شود و اگر مقدار آن صفر نشده بود، دوباره دستورات حلقه اجرا می‌شوند.

با مثالی نحوه پیاده‌سازی حلقه‌های تکرار معین را بررسی می‌کنیم.

مثال) با اجرای برنامه ... LED متصل به پین P1.0 مرتباً خاموش و روشن می‌شود (این برنامه مثالی از حلقه‌های بینهایت است).

اکنون می‌خواهیم این برنامه را طوری بنویسیم که LED متصل به پین P1.0 را ۱۰ بار خاموش و روشن کند و سپس برنامه به پایان برسد.

همانطور که گفته شد باید از یک ثبات به‌عنوان شمارنده برای شمارش تعداد خاموش و روشن شدن LED استفاده کنیم. در این مثال از ثبات A برای این کار استفاده می‌کنیم.

```
$MOD51
ORG     0
        MOV   A, #10
AGAIN:
        SETB  P1.0
```

دستورات ایجاد تأخیر

```
        CLR   P1.0
```

دستورات ایجاد تأخیر

[1] Counter

```
        DEC     A
        JNZ     AGAIN

        SJMP    $
END
```

در ابتدای اجرای برنامه، عدد ۱۰ در شمارنده ذخیره می‌شود و با هربار خاموش و روشن شدن LED یک واحد از آن کم می‌شود. اگر شمارنده صفر نشده بود، عمل چشمک زدن LED ادامه می‌یابد. وقتی شمارنده صفر شود، یعنی ۱۰ بار چشمک زدن LED به پایان رسیده و برنامه خاتمه می‌یابد.

دستور ایجاد حلقه تکرار معین

همانطور که در شکل ... می‌بینید، در انتهای حلقه باید:

- یک واحد از ثبات شمارنده کم شود.
- مقدار ثبات شمارنده با صفر مقایسه شود.
- اگر مقدار شمارنده غیرصفر بود با پرش به ابتدای حلقه، دستورات حلقه مجدداً تکرار شود.

دستور زیر این سه عمل را انجام می‌دهد:

DJNZ مقصد پرش، ثبات شمارنده

این دستور یک واحد از ثبات شمارنده کم می‌کند و اگر صفر نشده بود، به «مقصد پرش» می‌پرد (کلمه DJNZ[1] مخفف همین عمل است). «مقصد پرش» نام نقطه‌ای از برنامه است.

مثال) برنامه قبلی را با دستور DJNZ بازنویسی کنید.

```
$MOD51
ORG     0
        MOV     A, #10
AGAIN:
        SETB    P1.0
        دستورات ایجاد تأخیر
        CLR     P1.0
        دستورات ایجاد تأخیر
        DJNZ    A, AGAIN

        SJMP    $
END
```

به جای ثبات A می‌توان از هرکدام از ثباتهای دیگر میکروکنترلر هم استفاده کرد.

[1] Decrease counter and Jump if counter is Not Zero.

پرسش) الف) نشان دهید برنامه زیر نیز مانند مثال گفته شده عمل می‌کند.

ب) چرا مقدار اولیه شمارنده ۲۰ است؟

```
$MOD51
ORG     0
        MOV   R0, #20
NEXT:
        CPL   P1.0
```

دستورات ایجاد تأخیر

```
        DJNZ  R0, NEXT

        SJMP  $
END
```

مثال) برنامه‌ای بنویسید که اعداد ۱۵ تا صفر را مرتباً روی LEDهای متصل به چهار بیت کم‌ارزش پورت P1 نشان دهد.

برای این کار باید از یک حلقه ۱۵تایی برای ارسال اعداد به پورت P1 استفاده کنیم. چون قرار است «این حلقه ۱۵تایی» مرتباً تکرار شود، آن را در یک حلقه بینهایت قرار می‌دهیم.

```
$MOD51
ORG     0
AGAIN:
        MOV   R1, #15
NEXT:
        MOV   P1, R1
```

دستورات ایجاد تأخیر

```
        DJNZ  R1, NEXT
        SJMP  AGAIN
END
```

پرسش) چرا مقداردهی اولیه شمارنده درون حلقه بینهایت انجام شده است؟

مثال) برنامه‌ای بنویسید که اگر کلید فشاری متصل به P2.0 فشرده شده باشد، ۵ بار و اگر کلید فشاری متصل به P2.1 فشرده شود، ۱۰ بار LED متصل به P1.0 چشمک بزند. کلیدها را به گونه‌ای بسته‌ایم که در صورت فشرده شدن خروجی آن «یک» می‌شود.

```
$MOD51
ORG     0
        SETB  P2.0                  ;Define P2.0 as input
        SETB  P2.1                  ;Define P2.1 as input
AGAIN:
        JB    P2.0, TURN_10_TIMES ; If "KEY->P2.0" is pressed

        ; If "KEY->P2.0" is not pressed then
        JNB   P2.1, AGAIN           ; If "KEY->P2.1" is not pressed
```

```
            ; If "KEY->P2.1" is pressed
            MOV   R0, #5
            JMP   ON_OFF
TURN_10_TIMES:
            MOV   R0, #10
ON_OFF:
            CPL   P1.0
```

دستورات ایجاد تأخیر

```
            DJNZ  R0, ON_OFF

            JMP   AGAIN
END
```

پرسش) در مثال.... اگر هر دو کلید با هم فشرده شوند چه اتفاقی می‌افتد؟

پرسش) برنامه مثال.... را با فرض استفاده از کلیدهایی که هنگام فشرده شدن خروجی آنها «صفر» می‌شود بازنویسی کنید.

در مثال زیر با نوع دیگری از ایجاد حلقه آشنا می‌شویم:

مثال) برنامه‌ای بنویسید که مرتباً اعداد صفر تا ۱۵ را روی LEDهای متصل به پورت P1 نمایش دهد.

```
$MOD51
ORG         0
AGAIN:
            MOV   R2, #0
NEXT:
            MOV   P1, R2
```

دستورات ایجاد تأخیر

```
            INC   R2
            CJNE  R2, #16, NEXT
            JMP   AGAIN
END
```

حلقه‌های تأخیر

گاهی در برنامه‌نویسی لازم است عمداً میکروکنترلر را در نقطه‌ای از برنامه معطل کنیم؛ یکی از کاربردهای این موضوع را در ایجاد فاصله بین خاموش و روشن شدن LED برای دیده شدن چشمک زدن آن قبلاً دیدیم.

از آنجا که اجرای هر دستور توسط میکروکنترلر مدت زمانی به طول می‌انجامد، برای ایجاد تأخیر در برنامه باید دستوراتی که تأثیری در نتیجه برنامه ندارند اجرا شوند. دستور NOP[1]، دستوری است که هیچ کاری جز معطل کردن میکروکنترلر انجام نمی‌دهد. روش کلی ایجاد تأخیر، تکرار دستور NOP (یا هر دستور بی‌نتیجه دیگر) به تعداد معین است. این تعداد معین به مدت زمان تأخیری که می‌خواهیم ایجاد کنیم بستگی دارد. به بیان دیگر باید از یک حلقه تکرار معین شامل یک یا چند دستور NOP برای ایجاد تأخیر استفاده کنیم.

[1] No Operation

برای اینکه بتوانیم یک تأخیر به میزان دلخواه ایجاد کنیم، باید ابتدا شیوه اندازه‌گیری زمان اجرای دستورات یک قطعه برنامه (مثلاً یک حلقه تکرار معین) را بیاموزیم.

زمان اجرای یک دستور

برای آشنایی با نحوه اندازه‌گیری زمان اجرای دستورات باید با مفاهیم سیکل دستور و سیکل ماشین آشنا شویم.

سیکل ساعت[1]: کوچکترین واحد زمانی در میکروکنترلر سیکل ساعت نامیده می‌شود که آن را با T نشان می‌دهیم. همانطور که در فصل اول دیدیم، CPU یک مدار منطقی ترتیبی است که برای هماهنگی عملکرد اجزای داخلی به یک موج مربعی همزمان کننده که اصطلاحاً پالس ساعت نامیده می‌شود، نیاز دارد. فرکانس پالس ساعت در میکروکنترلر، برابر فرکانس کریستالی است که به آن متصل کرده‌ایم. سیکل ساعت T دوره تناوب پالس ساعت است که مقدار آن برای یک کریستال با فرکانس f برابر $\frac{1}{f}$ است. شکل.... پالس ساعت و سیکل ساعت نظیر آن را نشان می‌دهد.

مثال) سیکل ساعت میکروکنترلر ۸۰۵۱ را با دو کریستال مختلف محاسبه می‌کنیم:

الف) کریستال 11.0592 MHz

$$\text{f} = 11.0592\ \text{MHz}\ ,\ T = \frac{1}{f} = \frac{1}{11.0592MHz} = \frac{1}{11.0592\times10^{6}}S = \frac{1}{11.0592}\mu s$$

ب) کریستال 12 MHz

$$\text{f} = 12\ \text{MHz}\ ,\ T = \frac{1}{f} = \frac{1}{12MHz} = \frac{1}{12\times10^{6}}S = \frac{1}{12}\mu s$$

سیکل ماشین[2]: واحد اندازه‌گیری زمان اجرای دستورات میکروکنترلر ۸۰۵۱، سیکل ماشین است. یک سیکل ماشین حداقل زمانی است که انجام یک دستور به طول می‌انجامد. مثلاً اجرای دستورات MOV A, #10 و ADD R0, R1 یک سیکل ماشین، اجرای دستورات DJNZ R0, NEXT و JMP AGAIN دو سیکل ماشین و اجرای دستورات ضرب و تقسیم چهار سیکل ماشین طول می‌کشد. به بیان دیگر زمان اجرای دستورات میکروکنترلر ۸۰۵۱ یک یا دو یا چهار سیکل ماشین است.

هر سیکل ماشین در میکروکنترلر ۸۰۵۱ پایه، ۱۲ سیکل ساعت به طول می‌انجامد. به بیان دیگر زمان یک سیکل ماشین برابر است با:

$$T_{M.C} = 12T = \frac{12}{f}$$

مثال) زمان یک سیکل ماشین را با چند کریستال مختلف محاسبه می‌کنیم:

الف) کریستال 11.0592 MHz

$$T_{M.C} = 12T = \frac{12}{f} = \frac{12}{11.0592MHz} = \frac{12}{11.0592\times10^{6}}S = 1.085\mu s$$

ب) کریستال 12 MHz

---

[1] Clock Cycle
[2] Machine Cycle

$$T_{M.C} = 12T = \frac{12}{f} = \frac{12}{12MHz} = \frac{12}{12\times10^6}S = 1\mu s$$

ج) کریستال 16 MHz

$$T_{M.C} = 12T = \frac{12}{f} = \frac{12}{16MHz} = \frac{12}{16\times10^6}S = 0.75\mu s$$

د) کریستال 24 MHz

$$T_{M.C} = 12T = \frac{12}{f} = \frac{12}{24MHz} = \frac{12}{24\times10^6}S = 0.5\mu s$$

توجه کنید که با افزایش فرکانس کریستال، زمان یک سیکل ماشین کوتاهتر می‌شود؛ به عبارت دیگر دستورات سریعتر اجرا می‌شوند.

مثال) در میکروکنترلر ۸۰۵۱ دستور DJNZ دو سیکل ساعت به طول می‌انجامد. زمان اجرای آن با فرض استفاده از چند کریستال مختلف را محاسبه می‌کنیم:

الف) کریستال 11.0592 MHz

$$T_{Total} = 2T_{M.C} = 2\frac{12}{f} = 2\frac{12}{11.0592MHz} = 2\times1.085\mu s = 2.17\mu s$$

الف) کریستال 12 MHz

$$T_{Total} = 2T_{M.C} = 2\frac{12}{f} = 2\frac{12}{12MHz} = 2\times1\mu s = 2\mu s$$

الف) کریستال 16 MHz

$$T_{Total} = 2T_{M.C} = 2\frac{12}{f} = 2\frac{12}{16MHz} = 2\times0.75\mu s = 1.5\mu s$$

الف) کریستال 24 MHz

$$T_{Total} = 2T_{M.C} = 2\frac{12}{f} = 2\frac{12}{24MHz} = 2\times0.5\mu s = 1\mu s$$

زمان اجرای دستور DJNZ با کریستالهای مختلف را مقایسه کنید.

نحوه محاسبه زمان اجرای یک قطعه برنامه

برای محاسبه زمان اجرای یک قطعه برنامه، مراحل زیر را طی می‌کنیم:

۱) **محاسبه زمان یک سیکل ماشین:** با توجه به فرکانس کریستال متصل شده به میکروکنترلر، زمان یک سیکل ساعت را حساب می‌کنیم ($T = \frac{1}{f}$). زمان یک سیکل ماشین در میکروکنترلر پایه ۸۰۵۱، دوازده برابر زمان یک سیکل ساعت است ($T_{M.C} = 12T = \frac{12}{f}$). بعدها خواهیم دید این نسبت در خانواده‌های مختلف ۸۰۵۱ متفاوت است.

۲) **محاسبه تعداد کل سیکلهای ماشین برنامه:** با مراجعه به برگه اطلاعاتی میکروکنترلر ۸۰۵۱ (که در انتهای این کتاب هم آمده) می‌توان دریافت که اجرای هر دستور چند سیکل ماشین به طول می‌انجامد. گام بعدی به دست آوردن مجموع تعداد سیکلهای ماشین دستورات قطعه برنامه مورد نظر است.

۳) محاسبه زمان اجرای قطعه برنامه: از ضرب مجموع تعداد سیکلهای ماشین قطعه برنامه مورد نظر (مرحله ۲) در زمان یک سیکل ماشین (مرحله ۱)، زمان اجرای قطعه برنامه به دست می‌آید.

مثال) زمان اجرای برنامه زیر را با فرض استفاده از کریستال 12MHz به دست آورید. تعداد سیکل ماشین مربوط به هر دستور مقابل آن نوشته شده است.

```
ORG     0
AGAIN:
        MOV   A, P0      ; 1 Machine Cycle
        ADD   A, P1      ; 1 Machine Cycle
        MOV   P2, A      ; 1 Machine Cycle
        JMP   AGAIN      ; 2 Machine Cycle
END
```

توجه کنید که ORG و END راهنما هستند؛ یعنی دستور نیستند که مدت زمانی به اجرای آنها اختصاص یابد.

ابتدا زمان یک سیکل ماشین را به دست می‌آوریم:

$$T_{M.C} = 12T = \frac{12}{f} = \frac{12}{12\times10^6}S = 1\mu s$$

مجموع سیکلهای ماشین دستورات عبارتست از :

۱ + ۱ + ۱ + ۲ = ۵

زمان اجرای این برنامه عبارتست از:

۵ × ۱ μs = ۵ μs

توجه کنید که این برنامه بینهایت بار اجرا می‌شود؛ اما زمان اجرای یک مرتبه برابر ۵ میکروثانیه است. به بیان دیگر هر ۵ میکروثانیه یک بار مجموع P0+P1 روی P2 ظاهر می‌شود.

پرسش) نشان دهید با استفاده از کریستال 24MHz، یک دور اجرای این برنامه ۲/۵ میکروثانیه طول می‌کشد. چرا با استفاده از این کریستال زمان اجرای برنامه نصف شده است؟

پرسش) زمان اجرای یک دور برنامه زیر را در دو حالت P0 = 0 و P0 ≠ 0 با استفاده از کریستالهای 11.0592MHz، 12MHz و 16MHz به دست آورید.

```
$MOD51
ORG     0
        MOV   P0, #255 ; 1 Machine cycle
AGAIN:
        MOV   A, P0    ; 1 Machine cycle
        JZ    ON       ; 1 Machine cycle
        CLR   P1.0     ; 1 Machine cycle
        JMP   AGAIN    ; 2 Machine cycle
ON:
        SETB  P1.0     ; 1 Machine cycle
        JMP   AGAIN    ; 2 Machine cycle
END
```

توجه کنید که درهرکدام از دو حالت گفته شده، فقط بعضی از دستورات اجرا می‌شوند که باید زمان اجرای آنها به حساب آید.

نکته مهم دیگر این است که در میکروکنترلر ۸۰۵۱، زمان اجرای دستورات پرش شرطی، چه شرط آنها درست باشد و پرش انجام شود و چه شرط آنها نادرست باشد و پرش انجام نشود یکسان است. بنابراین دستور JZ در هر دوحالت ۲ سیکل ماشین به طول می‌انجامد[1].

ایجاد تأخیر

اکنون که با روش اندازه‌گیری زمان اجرای یک قطعه برنامه آشنا شدیم، می‌خواهیم نحوه ایجاد تأخیر با استفاده از حلقه‌های تکرار معین شامل دستورات NOP را ببینیم.

مثال) زمان اجرای قطعه برنامه زیر را فرض استفاده از کریستال 24MHz به دست آورید.

```
        MOV   R0, #10       ; 1 Machine Cycle
DELAY:
        NOP                 ; 1 Machine Cycle
        DJNZ  R0, DELAY     ; 2 Machine Cycle
```

زمان یک سیکل ماشین برابر با ۰/۵ میکروثانیه است (چرا؟)

این قطعه برنامه شامل چند سیکل ماشین است؟ اگر با جمع ۲ + ۱ + ۱ به ۴ سیکل ماشین برسید، کمی بی‌دقتی کرده‌اید!

توجه کنید که دستورات NOP و DJNZ در یک حلقه تکرار معین قرار دارند که ۱۰ بار اجرا می‌شود؛ یعنی دسورات فوق هرکدام ۱۰ بار اجرا می‌شوند. پس تعداد کل سیکلهای ماشین عبارتست از:

$$\underset{\text{MOV}}{۱} + \underset{\text{NOP}}{۱ \times ۱۰} + \underset{\text{DJNZ}}{۲ \times ۱۰} = ۳۱$$

زمان اجرای این قطعه برنامه عبارتست از:

$$۳۱ \times ۰/۵\ \mu s = ۱۵/۵\ \mu s$$

این قطعه برنامه کاری جز ایجاد ۱۵/۵ میکروثانیه تأخیر انجام نمی‌دهد.

مثال) حداکثر زمان تأخیری که می‌توان با کریستال 24MHz و حلقه مثال قبل ایجاد کرد، چقدر است؟

حداکثر تأخیر وقتی اتفاق می‌افتد که شمارنده حلقه حداکثر مقدار ممکن یعنی ۲۵۵ باشد:

```
        MOV   R0, #255      ; 1 Machine Cycle
DELAY:
        NOP                 ; 1 Machine Cycle
        DJNZ  R0, DELAY     ; 2 Machine Cycle
```

[1] در بعضی پردازنده‌ها (مانند پردازنده کامپیوتر)، اجرای دستور پرش شرطی در حالتی که پرش انجام شود بیش از زمانی که پرش انجام نشود، به طول خواهد انجامید.

تعداد کل سیکلهای ماشین عبارتست از:

$$\underset{\text{MOV}}{۱} + \underset{\text{NOP}}{۱ \times ۲۵۵} + \underset{\text{DJNZ}}{۲ \times ۲۵۵} = ۷۶۶$$

زمان اجرای این قطعه برنامه با کریستال 24MHz برابر است با:

$$۷۶۶ \times ۰/۵\ \mu s = ۳۸۳\ \mu s$$

پرسش) الف) با استفاده از مثالهای .... و .....، برنامه‌ای بنویسید که پین P1.0 را هر ۳۸۳ میکروثانیه خاموش و روشن کند.

ب) با استفاده از کریستال 12MHz زمان خاموش و روشن شدن پین P1.0 در برنامه قسمت الف چقدر می‌شود؟

ج) چگونه می‌توان با استفاده از کریستال 24MHz و تغییر قطعه برنامه تأخیر مثال....، زمان تأخیر را بیشتر کرد؟

ایجاد تأخیر بیشتر

دیدیم که روش ایجاد تأخیر، تکرار دستورات بی‌نتیجه (مانند NOP) در یک حلقه تکرار معین است. زمان اجرای یک قطعه برنامه تأخیر برابر حاصلضرب تعداد سیکلهای ماشین دستورات قطعه برنامه در زمان اجرای یک سیکل ماشین است؛ بنابراین برای زیاد کردن زمان تأخیر دو راه داریم:

۱) زیاد کردن زمان یک سیکل ماشین که با استفاده از کریستال با فرکانس کمتر (سیکل ساعت بزرگتر) حاصل می‌شود.

۲) زیاد کردن تعداد سیکلهای ماشین قطعه برنامه تأخیر

معمولاً از روش اول زیاد استفاده نمی‌شود؛ چون بالا رفتن زمان یک سیکل ماشین باعث بالا رفتن زمان اجرای کل برنامه و کندی سیستم می‌شود.

برای زیاد کردن تعداد سیکلهای ماشین قطعه برنامه تأخیر، باید حلقه تکرار معین ایجاد تأخیر بزرگتر شود؛ برای این کار می‌توان تعداد دستورات NOP داخل حلقه را بیشتر کرد یا از شمارنده‌ای با مقدار اولیه بزرگتر استفاده نمود.

مثال) قطعه برنامه زیر شامل چند سیکل ماشین است؟

```
        MOV   R2, #150
AGAIN:
        NOP
        NOP
        DJNZ  R2, AGAIN
```

$$\underset{\text{MOV}}{۱} + \underset{\text{NOP}}{۱ \times ۱۵۰} + \underset{\text{NOP}}{۱ \times ۱۵۰} + \underset{\text{DJNZ}}{۲ \times ۱۵۰} = ۶۰۱$$

اگر به جای دو دستور NOP، از چهار دستور NOP استفاده کنیم، تعداد سیکلهای ماشین این قطعه برنامه چقدر می‌شود؟

$$\underset{\text{MOV}}{۱} + \underset{\text{NOP}}{۱ \times ۱۵۰} + \underset{\text{NOP}}{۱ \times ۱۵۰} + \underset{\text{NOP}}{۱ \times ۱۵۰} + \underset{\text{NOP}}{۱ \times ۱۵۰} + \underset{\text{DJNZ}}{۲ \times ۱۵۰} = ۹۰۱$$

ایجاد تأخیر بزرگتر به این روش کمی آزاردهنده است.

حال بیایید به جای اضافه کردن دستور NOP، شمارنده حلقه قبلی را به حداکثر مقدارش یعنی ۲۵۵ افزایش دهیم:

$$\underset{\text{MOV}}{۱} + \underset{\text{NOP}}{۱ \times ۲۵۵} + \underset{\text{NOP}}{۱ \times ۲۵۵} + \underset{\text{DJNZ}}{۲ \times ۲۵۵} = ۱۰۲۱$$

برای ایجاد تأخیرهای خیلی بزرگ (مثلاً چند صد میلی‌ثانیه) ناچاریم هم تعداد دستورات NOP و هم مقدار اولیه شمارنده حلقه را بزرگ کنیم. برای امتحان، سعی کنید حلقه تکرار معینی بنویسید که شامل ۴۰۰ سیکل ماشین باشد.

روش دیگر برای ایجاد تأخیرهای طولانی، استفاده از حلقه‌های تودرتو است.

مثال) تعداد سیکلهای ماشین این قطعه برنامه را به دست آورید:

```
        MOV   R1, #100
LOOP1:
        MOV   R2, #255
LOOP2:
        NOP
        DJNZ  R2, LOOP2
        DJNZ  R1, LOOP1
```

همانطور که می‌بینید، حلقه LOOP2 (که با حروف پررنگتر نشان داده شده است) داخل حلقه LOOP1 قرار دارد؛ به این ساختار حلقه تودرتو[1] گفته می‌شود.

برای به‌دست آوردن تعداد سیکلهای ماشین حلقه تودرتو، باید ببینیم که هر دستور چند بار اجرا می‌شود؟

دستور MOV R2, #255 را در نظر بگیرید. این دستور در حلقه LOOP1 قرار دارد و ۱۰۰ مرتبه اجرا می‌شود (چون مقدار اولیه R1 که شمارنده حلقه LOOP1 است برابر ۱۰۰ می‌باشد).

اکنون دستور NOP را در نظر بگیرید. این دستور در حلقه LOOP2 قرار دارد و ۲۵۵ مرتبه اجرا می‌شود (مقدار اولیه شمارنده آن برابر ۲۵۵ است). اما حلقه LOOP2 خود داخل حلقه LOOP1 قرار دارد؛ بنابراین دستورات حلقه LOOP2 (از جمله NOP) ۱۰۰ مرتبه اجرا می‌شوند. بنابراین دستور NOP، ۲۵۵×۱۰۰ بار اجرا می‌شود.

محاسبه تأخیر حلقه‌های تودرتو

روش کلی به دست آوردن تعداد سیکلهای ماشین یک حلقه تودرتو به صورت زیر است:

۱) حدود هر حلقه را معلوم کنید؛ محدوده هر حلقه از دستور DJNZ آن حلقه تا مقصد پرش آن (نام نقطه ابتدایی حلقه) است.

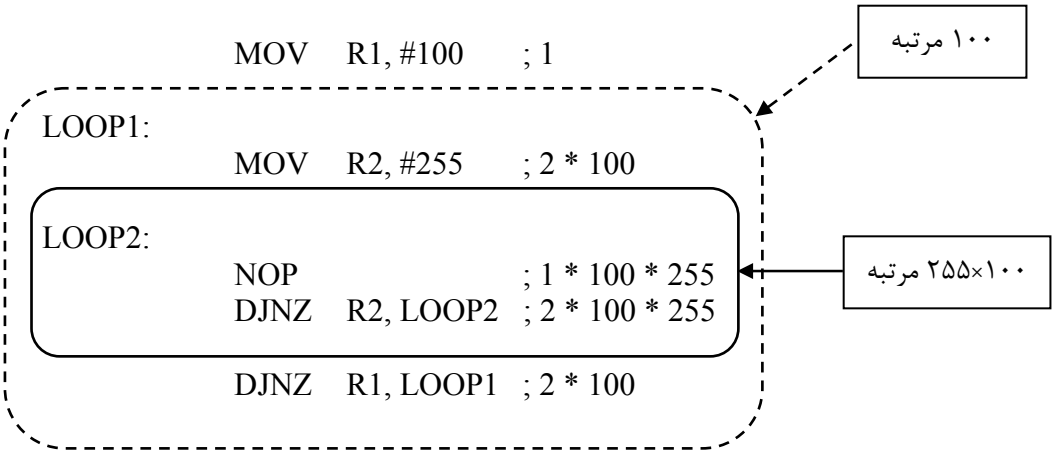
۲) از خارجی‌ترین حلقه شروع کنید و با توجه به مقدار اولیه شمارنده هر حلقه، مشخص کنید دستورات هر حلقه چند بار اجرا می‌شوند. به تعداد تکرار دستورات حلقه‌های تودرتو که خود، داخل حلقه‌ای دیگر قرار دارند توجه کنید.

[1] Nested Loop

۳) تعداد سیکلهای ماشین هر دستور برنامه را در تعداد تکرار آن دستور ضرب کنید و روبروی آن یادداشت کنید.

۴) تعداد کل سیکلهای ماشین را با هم جمع کنید.

مثال) تعداد کل سیکلهای ماشین قطعه برنامه زیر را به دست آورید.



۱ + ۱ × ۱۰۰ + ۱ × ۱۰۰ × ۲۵۵ + ۲ × ۱۰۰ × ۲۵۵ + ۲ × ۱۰۰ = ۷۶۸۰۱

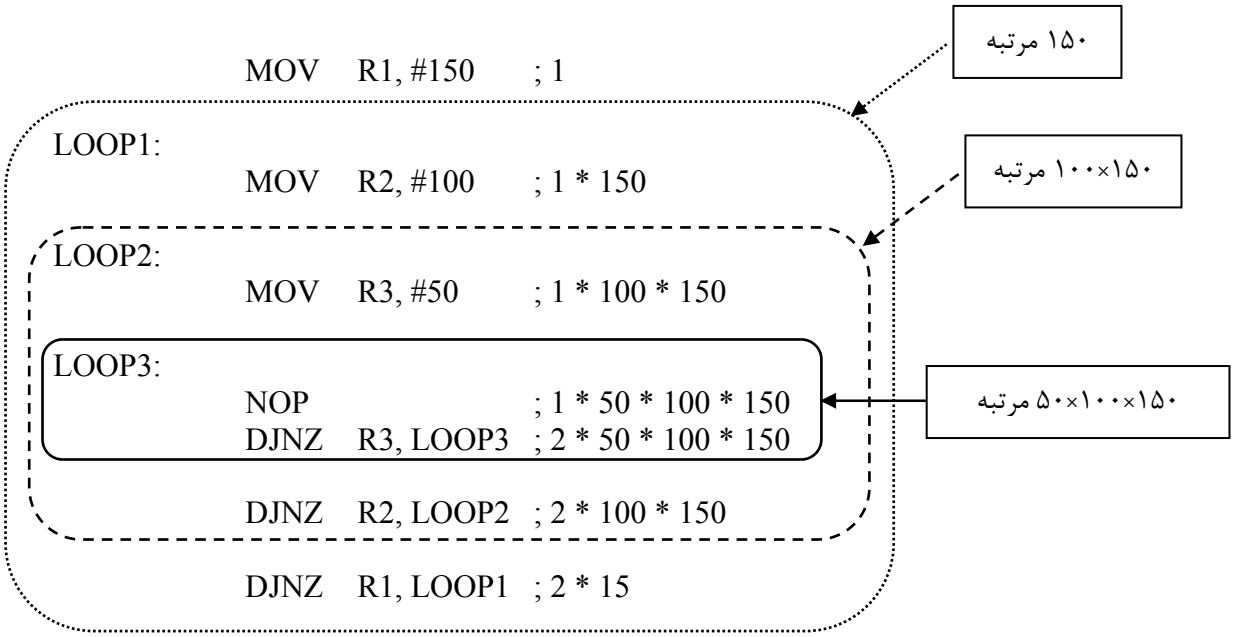
با استفاده از کریستال 12 MHz ($T_{M.C}=1\mu s$) زمان اجرای این قطعه برنامه برابر ۷۶/۸۰۱ میلی‌ثانیه می‌شود.

پرسش) الف) زمان اجرای این قطعه برنامه با کریستالهای 24 MHz و 16 MHz چقدر است؟

ب) اگر شمارنده حلقه خارجی برابر ۱۲۰ شود، زمان اجرای این قطعه برنامه با کریستال 12MHz چقدر خواهد بود؟

ج) اگر به جای یک دستور NOP داخل حلقه داخلی از دو دستور NOP استفاده کنیم، زمان اجرای این قطعه برنامه با کریستال 12MHz چقدر خواهد بود؟

مثال) تعداد سیکل ماشین هر دستور قطعه برنامه زیر را به دست آورید.

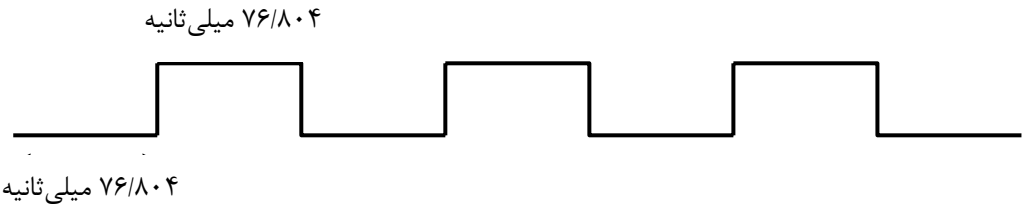


پرسش) زمان اجرای این قطعه برنامه را با استفاده از کریستال 24 MHz را به دست آورید.

مثال) برنامه زیر را در نظر بگیرید که در میکروکنترلر با کریستال 12MHz اجرا می‌شود.

```
$MOD51
ORG     0
AGAIN:
        CPL     P1.0

        MOV     R1, #100
DELAY:
        MOV     R2, #255
LOOP:
        NOP
        DJNZ    R2, LOOP
        DJNZ    R1, DELAY

        JMP     AGAIN
END
```

دستوراتی که پررنگتر نشان داده شده‌اند به هدف ایجاد تأخیر نوشته شده‌اند؛ در واقع در این برنامه، پین P1.0 در فواصل زمانی که دستورات ویژه تأخیر ایجاد می‌کنند، خاموش و روشن می‌شود.
زمان اجرای دستورات ویژه تأخیر با کریستال 12MHz برابر ۷۶/۸۰۱ میلی‌ثانیه است (نشان دهید). بنابراین پین P1.0 در فواصل زمانی ۷۶/۸۰۱ میلی‌ثانیه خاموش و روشن می‌شود. البته برای اینکه دقیقتر سخن بگوییم، باید زمان اجرای دستورات CPL (یک سیکل ماشین) و JMP (دو سیکل ماشین) را هم به زمان فوق اضافه کنیم که زمان دقیق مابین خاموش و روشن شدن برابر ۷۶/۸۰۴ میلی‌ثانیه خواهد شد.
اگر موج مربعی ایجاد شده روی پین P1.0 را با اسیلوسکوپ مشاهده کنیم، شکل زیر را خواهیم دید:



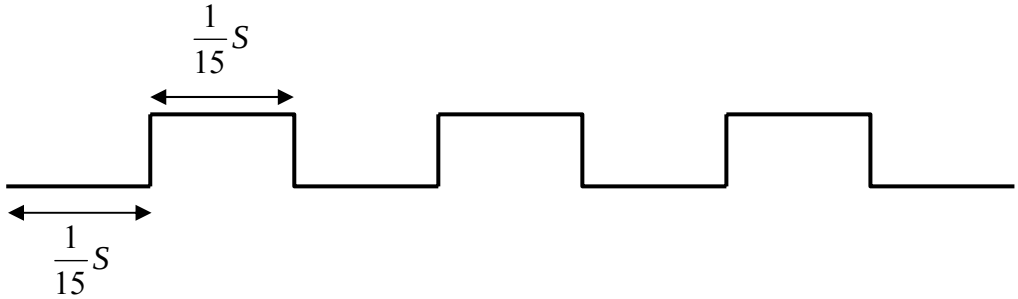
فرکانس این موج مربعی عبارتست از:

$$f = \frac{1}{2 \times 76.804ms} = 6.51Hz$$

می‌خواهیم به یک پرسش مهم پاسخ دهیم:
اگر یک LED را به پین P1.0 متصل کنیم، آیا خاموش و روشن شدن آن دیده می‌شود؟
قبلاً گفتیم اثر یک نقطه روشن تا $\frac{1}{15}$ ثانیه (حدود ۶۷ میلی‌ثانیه) در چشم باقی می‌ماند. یعنی وقتی LED روشن می‌شود، تا ۶۷ میلی‌ثانیه چشم همچنان آن را روشن می‌بیند (حتی اگر خاموش شده باشد). بنابراین اگر بخواهیم روشن و خاموش شدن LED دیده شود، باید زمان خاموش بودن آن از ۶۷ میلی‌ثانیه بیشتر باشد. اگر زمان روشن و

خاموش بودن LED را مساوی در نظر بگیریم، فرکانس چشمک زدن آن به صورتی که بـه راحتـی دیـده شـود، بـه صورت زیر به دست می‌آید.

$$f = \frac{1}{2 \times \frac{1}{15}S} = 7.5Hz$$

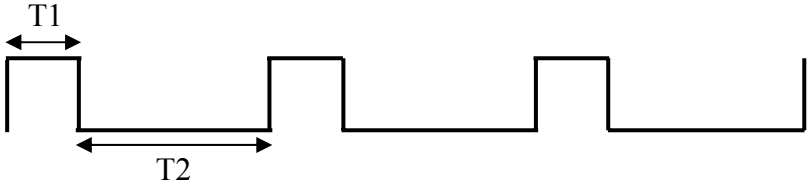


این نشان می‌دهد که اگر فرکانس خاموش و روشن شدن یک LED کمتر از ۷/۵ هرتز باشـد، چـشمک زدن آن بـه راحتی دیده می‌شود. بنابراین اگر در مثال .... یک LED به پین P1.0 متصل شود، چشمک زدن آن دیده می‌شود.

اگر فرکانس خاموش و روشن شدن یک LED بیش از ۷/۵ هرتز باشد چه می‌شود؟
تا حدود فرکانس ۲۵ هرتز (یعنی وقتی زمان خاموش یا روشن ماندن LED حدود ۴۰ میلی‌ثانیه باشد) باز هم چشم متوجه کم و زیاد شدن نور می‌شود. اما هرچه فرکانس زیادتر شود، چشم بیشتر متقاعد می‌شـود کـه LED همیـشه روشن است و متوجه خاموش شدن گاه به گاه آن نمی‌شود. از این موضوع به طوری که در بخش بعـد مـی‌بینـیم در نمایشگرهای نوری استفاده می‌شود.

اثر سیکل وظیفه

در یک موج مربعی نسبت زمان «یک» بودن موج به زمان یـک دوره تنـاوب مـوج (در شـکل ... نـسبت $\frac{T1}{T1+T2}$) سیکل وظیفه[1] نامیده می‌شود که معمولاً برحسب درصد بیان می‌شود.



در مثال... چون زمان روشن و خاموش بودن موج مربعی یکسان است، سیکل وظیفه آن ۵۰٪ است.
دانستیم اگر موج مربعی با فرکانس ۷/۵ هرتز و زمان روشن و خاموش بودن یکسان (سـیکل وظیفـه ۵۰٪) بـه یـک LED اعمال شود، خاموش و روشن شدن LED دیده می‌شود.
پرسش بعدی این است که اگر فرکانس از ۷/۵ هرتز بیشتر شود، اما زمـان خـاموش شـدن LED را از زمـان روشـن بودن آن بیشتر کنیم (سیکل وظیفه کمتر از ۵۰٪ مانند شکل...)، می‌توان خاموش و روشن شدن آن را دید؟

[1] Duty Cycle

پاسخ مثبت است. در واقع هم فرکانس و هم سیکل وظیفه در دیده شدن چشمک زدن یک LED اثر دارند. هرچه فرکانس و سیکل وظیفه کمتر باشد، روشن و خاموش شدن LED بهتر دیده می‌شود. اثر کم شدن فرکانس از کم شدن سیکل وظیفه، بیشتر است.

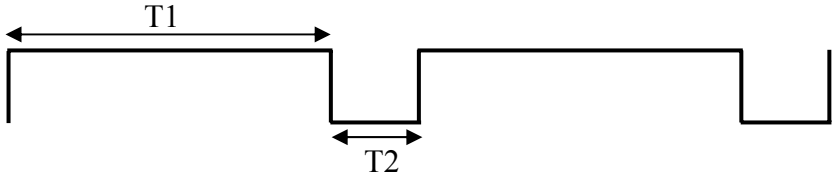
نمایشگرهای نوری

در مثالهای قبلی به خاطر اینکه روشن و خاموش شدن یک نقطه **دیده شود**، عمداً بین روشن و خاموش شدن آن تأخیر ایجاد کردیم تا به چشم فرصت درک این تغییر را بدهیم.

در بعضی از سیستمهای نوری مانند تلویزیون، نمایشگر کامپیوتر، تابلوهای تبلیغاتی و ... عکس این موضوع اتفاق می‌افتد؛ یعنی در این سیستمها در هر لحظه فقط یک نقطه روشن است! اما از لحظه‌ای که یک نقطه روشن و بعد خاموش می‌شود تا لحظه ای که دوباره روشن می‌شود، این قدر کم به طول می‌انجامد که چشم متوجه خاموش شدن آن نمی‌شود و تصور می‌کند که دائماً روشن است؛ مثلاً وقتی تصویر یک گل را در تلویزیون می‌بینید، تصور نکنید تمام نقاط این تصویر (که پیکسل نامیده می شوند) همیشه روشن هستند؛ بلکه در هر لحظه فقط یک نقطه روشن است و کنترل کننده نمایشگر با شروع از یک گوشه تصویر، هر نقطه را متناسب با تصویر مورد نظر روشن می‌کند و به سراغ نقطه بعدی می‌رود. این کار بسیار سریع انجام می‌شود؛ آن قدر سریع که وقتی کنترل‌کننده به نقطه شروع باز می‌گردد و دوباره آن را روشن می‌کند، چشم هنوز متوجه خاموش شدن آن نشده است؛ به این ترتیب همه نقاط تصویر همیشه روشن دیده می‌شوند.

برای اینکه یک نقطه در تلویزیون همواره روشن دیده شود، حدود ۵۰ بار در ثانیه روشن شود. این میزان در نمایشگر کامپیوتر حدود دو برابر است. اما برای اینکه یک LED همیشه روشن دیده شود (مثلاً در تابلوهای تبلیغاتی که متشکل از تعداد زیادی LED است یا در نمایشگرهای هفت‌قسمتی تازه‌شونده[1]) ۲۵ بار روشن شدن در ثانیه نیز کافی است.

اگر فرکانس را نتوانیم خیلی بالا ببریم، اضافه کردن سیکل وظیفه (در یک دوره تناوب نقطه نورانی بیشتر روشن باشد تا خاموش) به روشن دیده شدن نقطه کمک می‌کند. شکل .... را ببینید.



مثال) برنامه زیر را که در میکروکنترلری با کریستال 12MHz اجرا می‌شود، در نظر بگیرید و فرکانس موج مربعی خروجی و سیکل وظیفه آن را تعیین کنید.

```
$MOD51
```

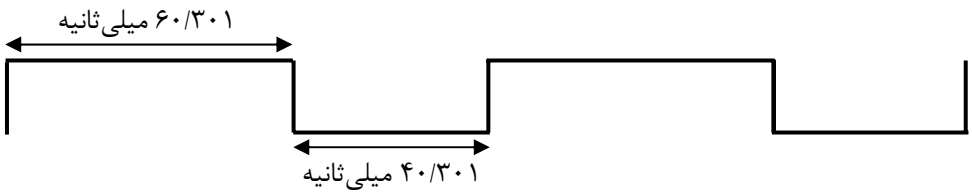
[1] Refreshing 7-Segments

```
ORG     0
AGAIN:
        SETB    P1.0

        MOV     R0, #100
LOOP0:                          ┐
        MOV     R1, #200        │
LOOP1:                          │ T1
        NOP                     │
        DJNZ    R1, LOOP1       │
        DJNZ    R0, LOOP0       ┘

        CLR     P1.0

        MOV     R2, #100
LOOP2:                          ┐
        MOV     R3, #200        │ T2
LOOP3:                          │
        DJNZ    R3, LOOP3       │
        DJNZ    R2, LOOP2       ┘

        JMP     AGAIN
END
```

توجه کنید که حلقه داخلی دستورات ایجاد تأخیر T2، دستور NOP ندارد و فقط دستور ایجاد حلقه DJNZ R1, LOOP3 در آن تکرار می‌شود. چون نقطه پرش در همان خط است، به جای این دستور می‌توان از دستور DJNZ R3, $ هم استفاده کرد (چرا؟).

برای تمرین نشان دهید T1 = ۶۰/۳۰۱ ms و T2 = ۴۰/۳۰۱ ms.

اگر موج مربعی ایجاد شده روی P1.0 را با اسیلوسکوپ ببینید، شکل زیر را مشاهده خواهید کرد.



فرکانس این موج مربعی برابر $f = \frac{1}{T1+T2} = \frac{1}{60.301ms + 40.301ms} = 9.94Hz$ و سیکل وظیفه آن برابر با $\frac{60.301}{60.301+40.301}$ (٪۶۰) است.

فرض کنید یک LED را به پین P1.0 متصل کرده‌ایم. فرکانس موج مربعی بیش از ۷/۵ هرتز است؛ اما چون سیکل وظیفه آن بیش از ٪۵۰ است (در یک دوره تناوب، LED بیشتر روشن است تا خاموش) بازهم خاموش و روشن شدن آن دیده می‌شود.

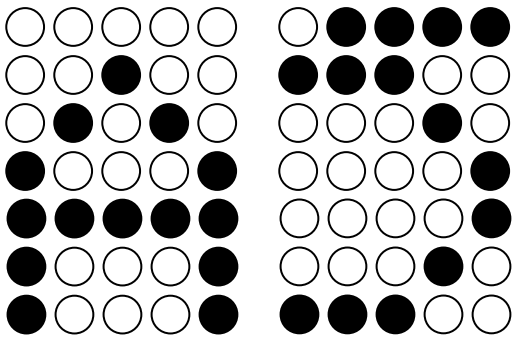
پرسش) شکل موج خروجی، فرکانس و سیکل وظیفه موج مربعی ایجاد شده توسط قطعه برنامه زیر که روی میکروکنترلر ی با کریستال 16MHz اجرا می‌شود، را به دست آورید.

```
$MOD51
```

```
ORG     0
AGAIN:
        SETB    P1.0

        MOV     R0, #250
LOOP1:
        MOV     R1, #200
        DJNZ    R1, $
        DJNZ    R0, LOOP1

        CLR     P1.0

        MOV     R0, #120
LOOP2:
        MOV     R1, #250
        DJNZ    R1, $
        DJNZ    R0, LOOP2

        JMP     AGAIN
END
```

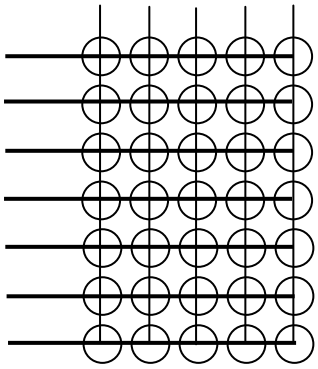
نمایشگرهای ماتریس ــ نقطه‌ای[1]

نمایشگر ماتریس ـ نقطه‌ای، از یک مجموعه LED تشکیل می‌شود که برای نمایش یک نشانه یا حرف به کار می‌رود. شکل زیر نمایش کاراکترهای A و «ح» روی یک نمایشگر ماتریس ـ نقطه‌ای ۵×۷ را نشان می‌دهد.



روش نمایش یک کاراکتر در نمایشگر ماتریس ـ نقطه‌ای به این صورت است که برای آن کاراکتر یک الگوی ماتریسی (مانند آنچه در شکل بالا برای حرف A یا «ح» می‌بینید) در نظر می‌گیریم. برای نمایش یک کاراکتر روی نمایشگر ماتریس ـ نقطه‌ای، می‌توان هر LED نمایشگر را به یک پین میکروکنترلر متصل کرد. اما این روش پینهای زیادی از میکروکنترلر را اشغال می‌کند. به همین دلیل، معمولاً نمایشگرهای ماتریس ــ نقطه‌ای را به صورت ماتریسی (شکل....) می‌سازند.

[1] Dot-Matrix



برای روشن شدن یک LED باید سطر و ستون متصل به آن LED فعال شود. هر الگویی که به سطرهای نمایشگر ماتریس ـ نقطه‌ای ارسال شود، روی ستونی که فعال است نمایش داده می‌شود. مثلاً برای نمایش ستون اول کاراکتر A، الگوی ۰۰۰۱۱۱۱ را به سطرهای نمایشگر ماتریس ـ نقطه‌ای ارسال و ستون اول آن را فعال می‌کنیم. با این روش در هر لحظه می‌توان LEDهای یک ستون را روشن کرد. اما با استفاده از خاصیت چشم، می‌توان به نحوی عمل کرد که تمام LEDهای لازم، روشن نشان داده شوند. برای این کار، ابتدا الگوی مربوط به ستون اول را به سطرها ارسال می‌کنیم و ستون اول را برای مدت کوتاهی روشن می‌کنیم. سپس الگوی مربوط به ستون دوم را به سطرها می‌فرستیم و ستون دوم را فعال می‌کنیم. این کار برای هر ۵ ستون انجام می‌شود. سپس بلافاصله به سراغ ستون اول برمی‌گردیم و LEDهای آن را روشن می‌کنیم. اگر این کار با سرعت کافی انجام شود، چشم متوجه اینکه در هر لحظه یکی از ستونها روشن و بقیه خاموش است نمی‌شود و همه ستونها را روشن می‌پندارد.

از این روش در تابلوهای تبلیغاتی که از تعداد زیادی LED تشکیل می‌شود هم به کار می‌رود که در فصل..... با نمونه‌ای از آن آشنا خواهیم شد.

توجه کنید اگر سرعت روشن کردن LEDها کافی نباشد، نور LEDها ضعیف می‌شود و پرشهایی در نمایش آنها دیده می‌شود.

ایجاد تأخیرهایی به میزان مشخص

تاکنون آموخته‌ایم چگونه با توجه به کریستال استفاده شده در مدار و تعداد سیکلهای ماشین دستورات برنامه، زمان اجرای آن را به دست آوریم.

اکنون می‌خواهیم ببینیم چگونه دستوراتی بنویسیم که تأخیری به میزان مشخص و دقیق ایجاد کنند. این موضوع در مواردی که نیاز به انجام کاری در فواصل زمانی دقیق داریم (مثلاً وقتی باید مقدار یک پورت میکروکنترلر را هر یک ثانیه یک بار خوانده و پردازش کنیم) کاربرد دارد.

روش زیر برای طراحی تأخیرهایی به میزان مشخص به کار می‌رود:

۱) ابتدا با توجه به مقدار تأخیر مورد نیاز، حلقه‌ای دلخواه از دستورات NOP بنویسید که زمان اجرای آن کمتر از زمان تأخیر مورد نیاز باشد.

۲) زمان اجرای حلقه فوق را اندازه بگیرید.

۳) زمان تأخیر مورد نیاز را بر زمان اجرای این حلقه تقسیم کنید. خارج قسمت صحیح این تقسیم نشان می‌دهد حلقه‌ای که در مرحله اول نوشته‌ایم چند بار باید تکرار شود تا تأخیرمورد نظر ایجاد شود.
۴) با توجه به نتیجه مرحله ۳، یک حلقه تودرتو می‌نویسیم که حلقه داخلی آن همان حلقه مرحله ۱ است.
۵) چون شمارنده حلقه خارجی دقیق نیست (مقدار آن در مرحله ۳ گرد می‌شود)، زمان اجرای حلقه تودرتو کمتر از اندازه مورد نیاز ماست. بنابراین زمان اجرای آن را اندازه گرفته و اختلاف زمانی آن با زمان تأخیر مورد نظر را با نوشتن حلقه کوچکتری جبران می‌کنیم.

مثال) می‌خواهیم قطعه برنامه‌ای بنویسیم که با استفاده از یک کریستال 12MHz، ۴۰۰ میلی‌ثانیه تأخیر ایجاد کند.

مراحل گفته شده را گام به گام طی می‌کنیم:

۱) یک حلقه دلخواه از دستورات NOP می‌نویسیم. مثلاً از حلقه زیر استفاده می‌کنیم:

```
        MOV   R1, #255
LOOP1:
        NOP
        NOP
        NOP
        NOP
        NOP
        DJNZ  R1, LOOP1
```

زمان اجرای این حلقه برابر ۱/۷۸۶ میلی‌ثانیه است.

۲) می‌خواهیم ۴۰۰ میلی‌ثانیه تأخیر ایجاد کنیم؛ در ضمن حلقه‌ای داریم که ۱/۷۸۶ میلی‌ثانیه به طول می‌انجامد. بنابراین این حلقه باید $\frac{400}{1.786} = 223.96 \approx 223$ مرتبه اجرا شود.

۳) بنابراین اگر حلقه‌ای که در مرحله ۱ نوشته شد، ۲۲۳ مرتبه تکرار شود، تقریباً به تأخیر ۴۰۰ میلی‌ثانیه رسیده‌ایم. پس یک حلقه تودرتو می‌نویسیم که حلقه قبلی را ۲۲۳ اجرا کند:

```
        MOV   R2, #223
LOOP2:

        MOV   R1, #255
LOOP1:
        NOP
        NOP
        NOP
        NOP
        NOP
        DJNZ  R1, LOOP1

        DJNZ  R2, LOOP2
```

۴) زمان اجرای این قطعه برنامه کمتر از ۴۰۰ میلی‌ثانیه است؛ چون در واقع حلقه‌ای که در مرحله ۱ نوشته شده باید ۲۲۳/۹۶ مرتبه تکرار شود تا به تأخیر ۴۰۰ میلی‌ثانیه برسیم که در قطعه برنامه فوق، ۲۲۳ مرتبه تکرار شده است. زمان دقیق اجرای دستورات این حلقه برابر ۳۹۸/۷۲۵ میلی‌ثانیه است که ۱/۲۷۵ میلی‌ثانیه تا ۴۰۰ میلی‌ثانیه

تأخیر مورد نظر فاصله دارد. با یک حلقه کوچکتر باید این فاصله را جبران کنیم. حلقه زیر دقیقاً ۱/۲۷۵ میلی‌ثانیه تأخیر ایجاد می‌کند.

```
        MOV   R3, #255
LOOP3:
        NOP
        NOP
        NOP
        DJNZ  R3, LOOP3
```

بنابراین با اجرای دستورات زیر، به اندازه ۴۰۰ میلی‌ثانیه تأخیر ایجاد می‌کنیم:

```
        MOV   R2, #223
LOOP2:
        MOV   R1, #255
LOOP1:
        NOP
        NOP
        NOP
        NOP
        NOP
        DJNZ  R1, LOOP1
        DJNZ  R2, LOOP2

        MOV   R3, #255
LOOP3:
        NOP
        NOP
        NOP
        DJNZ  R3, LOOP3
```

توجه کنید این مدت زمان تأخیر با استفاده از کریستال 12MHz ایجاد می‌شود. اگر از کریستال 24MHz استفاده کنیم، میزان تأخیر برابر ۲۰۰ میلی‌ثانیه می‌شود (نشان دهید).

پرسش) نشان دهید اگر شمارنده حلقه خارجی برابر ۲۲۴ انتخاب شود، ۴۰۰/۲۸۹ میلی‌ثانیه تأخیر ایجاد می‌شود.

مثال) قطعه برنامه‌ای بنویسید که با استفاده از کریستال 16MHz، ۱/۵ میلی‌ثانیه تأخیر ایجاد کند.

مراحل را گام به گام دنبال می‌کنیم:

۱) نوشتن یک حلقه دلخواه

```
        MOV   R1, #100
LOOP1:
        NOP
        DJNZ  R1, LOOP1
```

۲) اندازه‌گیری زمان اجرای حلقه دلخواه که در این مثال برابر ۲۲۵/۷۵ میکروثانیه می‌شود.

۳) تقسیم زمان تأخیر مورد نیاز بر زمان اجرای حلقه دلخواه برای به دست آوردن شمارنده حلقه خارجی تودرتو

$$\frac{1.5ms}{225.75\mu s} = 6.64 \approx 6$$

۴) نوشتن حلقه تودرتو

```
        MOV   R2, #6
LOOP2:
        MOV   R1, #100
LOOP1:
        NOP
        DJNZ  R1, LOOP1
        DJNZ  R2, LOOP2
```

۵) اندازه‌گیری مجدد زمان اجرای حلقه تودرتو که در این مثال برابر ۱/۳۶۴ میلی‌ثانیه می‌شود.

۶) جبران زمان اختلاف (۱۳۵ μs = ۱/۳۶۴ms - ۱/۵ms) با یک حلقه کوچکتر

```
        MOV   R3, #60
LOOP3:
        NOP
        DJNZ  R3, LOOP3
```

۷) نوشتن دستورات کامل تأخیر

```
MOV   R2, #6
LOOP2:
        MOV   R1, #100
LOOP1:
        NOP
        DJNZ  R1, LOOP1
        DJNZ  R2, LOOP2

        MOV   R3, #60
LOOP3:
        NOP
        DJNZ  R3, LOOP3
```

توجه کنید که حلقه زیر هم حدود ۱/۵ میلی‌ثانیه تأخیر ایجاد می‌کند:

```
        MOV   R0, #255
DELAY:
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        DJNZ  R0, DELAY
```

اما نوشتن حلقه‌های تکرار معین با دستورات NOP زیاد از نظر برنامه‌نویسی جالب نیست و بهتر است از حلقه‌های تودرتو استفاده شود.

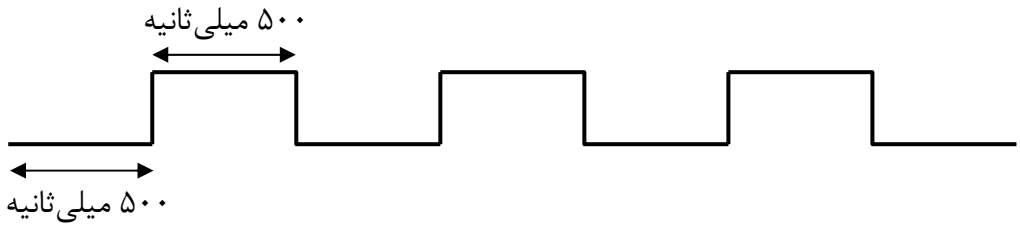
پرسش) با استفاده از کریستالهای 11.0592MHz، 12MHz، 16MHz و 24MHz قطعه برنامه‌هایی بنویسید که الف) ۴۵ میلی‌ثانیه ب) ۶۵۰ میلی‌ثانیه ج) ۱ ثانیه تأخیر ایجاد کند.

توجه کنید برای ایجاد یک تأخیر طولانی (مثلاً ۱ ثانیه) می‌توان دستورات ایجاد تأخیر ۵۰۰ میلی‌ثانیه را دوبار تکرار کرد یا حلقه دلخواه اولیه را بزرگتر در نظر گرفت.

مثال) برنامه‌ای بنویسید که با کریستال 12MHz، یک موج مربعی با فرکانس یک هرتز و سیکل وظیفه ۵۰٪ روی پین P2.0 تولید کند.
ابتدا دوره تناوب موج مربعی را به دست می‌آوریم:

$$T = \frac{1}{f} = \frac{1}{1Hz} = 1s$$

چون سیکل وظیفه ۵۰٪ است، زمان روشن بودن پین برابر نیمی از زمان تناوب (یعنی ۵۰۰ میلی‌ثانیه) است؛ یعنی زمان روشن و خاموش بودن پین یکسان است.



بنابراین باید برنامه‌ای بنویسیم که ۵۰۰ میلی‌ثانیه پین P2.0 را روشن و ۵۰۰ میلی‌ثانیه آن را خاموش کند. برای تمرین این برنامه را بنویسید.

بلندگوهایی به نام PZ[1] وجود دارند که با ارسال یک موج مربعی به آنها، صوتی تولید می‌شود. بلندگوی داخلی کامپیوتر نمونه‌ای از این بلندگوهاست. در فصل.... خواهیم دید چگونه با استفاده از فرکانسهای مخصوص نتهای مختلف موسیقی و تولید موجهای مربعی با این فرکانسها و ارسال آنها به یک بلندگو، آهنگهای مختلف و متنوعی ساخت.

پرسش) سیستمی طراحی کنید که با استفاده از یک کریستال دلخواه، یک موج مربعی با فرکانس ۴۰۰ هرتز و سیکل وظیفه ۷۵٪ (زمان روشن بودن ۳ برابر زمان خاموش بودن) روی پین P1.1 تولید کند.

عوامل مؤثر بر سرعت اجرای برنامه

سه عامل مهم بر زمان اجرای برنامه تأثیر می‌گذارند.

۱) **مجموع تعداد سیکلهای ماشین دستورات**

۲) **فرکانس کریستال:** اگر برنامه‌ای ۲۰۰ سیکل ماشین به طول بیانجامد، زمان اجرای آن با کریستالهای 12MHz و 16MHz و 24MHz به ترتیب برابر ۲۰۰، ۱۵۰ و ۱۰۰ میکروثانیه خواهد بود. حداکثر فرکانس کریستالی که می‌تواند به یک میکروکنترلر متصل شود، یکی از اعدادی است که روی تراشه میکروکنترلر نوشته می‌شود (فصل.... را ببینید).

[1] Piezo Sounder

۳) **نوع میکروکنترلر و زمان یک سیکل ماشین:** همانطور که گفته شد، زمان یک سیکل ماشین در میکروکنترلر پایه ۸۰۵۱، دوازده برابر زمان یک سیکل ساعت است. این موضوع در میکروکنترلر های مختلف خانواده ۸۰۵۱ متفاوت است؛ مثلاً زمان یک سیکل ماشین در میکروکنترلر P89C54X که ساخته شرکت فیلیپس است شش برابر زمان یک سیکل ساعت، در میکروکنترلر DS5000T شرکت دالاس چهار برابر زمان یک سیکل ساعت و در میکروکنترلر DS89C4XX دالاس برابر زمان یک سیکل ساعت است. اگر زمان یک سیکل ساعت را با T نشان دهیم، جدول... زمان یک سیکل ساعت میکروکنترلر های مختلف را نشان می‌دهد.

نوع میکروکنترلر	زمان یک سیکل ماشین
۸۰۵۱ پایه	12T
P89C54X	6T
DS5000T	4T
DS89C4XX	T

واضح است هرچه زمان یک سیکل ماشین کمتر باشد، برنامه روی میکروکنترلر سریعتر اجرا می‌شود. مثال زیر را ببینید.

مثال) برنامه زیر را در نظر بگیرید.

```
ORG     0
AGAIN:
        MOV   A, P0     ; 1 Machine Cycle
        ADD   A, P1     ; 1 Machine Cycle
        MOV   P2, A     ; 1 Machine Cycle
        JMP   AGAIN     ; 2 Machine Cycle
END
```

یک دور اجرای این برنامه ۵ سیکل ماشین به طول می‌انجامد. اگر از یک کریستال 12MHz استفاده کنیم، زمان اجرای این برنامه با استفاده از میکروکنترلر های جدول... به شکل زیر است:

نوع میکروکنترلر	زمان یک سیکل ماشین	زمان اجرای برنامه
۸۰۵۱ پایه	12T = 1µs	5µs
P89C54X	6T = 0.5µs	3µs
DS5000T	4T = 0.33µs	1.66µs
DS89C4XX	T = 0.083µs	0.42µs

همانطور که می‌بینید استفاده از میکروکنترلرهای پیشرفته‌تر باعث سرعت بخشیدن به اجرای برنامه می‌شود.

پرسش) با استفاده از کریستالهای 12MHz، 16MHz و 24MHz و میکروکنترلر های جدول...، قطعه برنامه‌هایی بنویسید که الف) ۴۵ میلی‌ثانیه ب) ۶۵۰ میلی‌ثانیه ج) ۱ ثانیه تأخیر ایجاد کند.