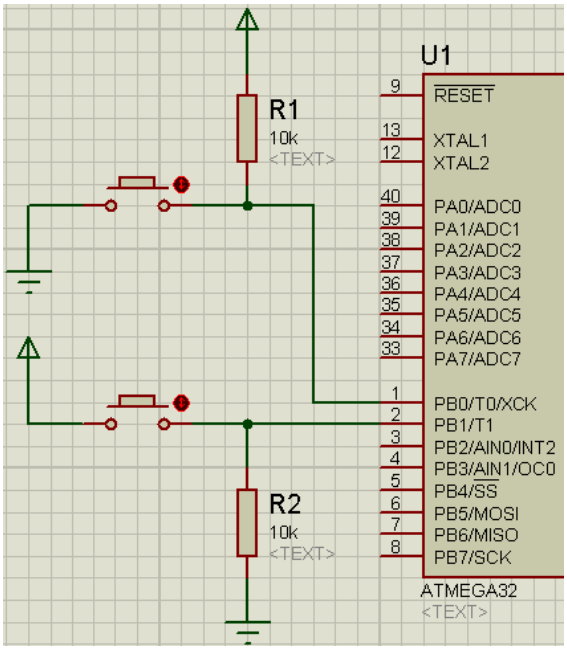
آزمایش دوم: شمارنده و شیفت‌رجیستر: اتصال کلید و LED به میکروکنترلر

الف) مداری طراحی و شبیه‌سازی کنید که یک کلید روی پورت PA0 و هشت عدد LED روی پورت B وجود داشته باشد و با هر بار زدن کلید، LEDهای موجود به صورت یک شیفت رجیستر عمل نماید.

ب) مداری طراحی، شبیه‌سازی و پیاده‌سازی کنید که یک کلید روی پورت PA0 و هشت عدد LED روی پورت B وجود داشته باشد و با هر بار زدن کلید، LEDهای موجود به صورت یک شمارنده بالا شمار عمل نماید.

توضیحات: توسط کلید می‌توان منطق صفر یا یک را به میکروکنترلر وارد کرد و بر اساس آن پردازش را انجام داد. برای خواندن منطق کلید از ثبات PIN استفاده می‌شود. نحوه استفاده از ثبات PIN به این صورت است که اگر کلید زده شد آنگاه کار مورد نظر انجام شود. اتصال کلید به دو صورت Pull-up و PullDown انجام می‌شود که در شکل زیر مشاهده می‌کنید. در کلید pull up در حالتی که کلید زده نشده منطق یک و در حالت فشردن کلید منطق صفر وارد میکروکنترلر می‌شود. معمولا از مقاومت 10k برای این کار استفاده می‌شود.

```
if(PINB.0==0) {...} //Pull-up
if(PINB.1==1) {...} //PulDown
```



اما مشکلی که در کلید وجود دارد، بوجود آمدن لرزش یا bounce در هنگام قطع و وصل کلید است. در زمانی که کاربر کلید را فشار می‌دهد تقریبا 20 میلی‌ثانیه طول می‌کشد تا منطق کلید از صفر به یک (یا بالعکس) ثابت شود تا قبل این به علت وجود جرقه کلید منطق ثابتی ندارد.

در واقع، این مشکل زمانی برای ما مسئله‌ساز می شود که زمانی که کاربر کلید را یکبار فشار می‌دهد و انتظار دارد تا میکروکنترلر متوجه یکبار فشار دادن آن شود اما به علت قرار گرفتن if در درون حلقه نامتناهی while چندین بار شرط PINB.0==0 برقرارشده و کار مورد نظر چندین بار انجام می‌شود. راه حل این مشکل ساده است و آن هم قرار دادن مقداری delay در حلقه if است.

برای حل این مشکل بسته به نوع برنامه یکی از سه روش زیر قابل استفاده است:

توضیح: به محض فشار دادن کلید توسط کاربر شرط if برقرارشده و دستورات مورد نظر اجرا می‌شود سپس به علت ایجاد تاخیر زیاد توسط تابع) delay_ms در اینجا 200 میلی‌ثانیه) با این کار احتمال اینکه زمانی که برنامه در حلقه while بهif می‌رسد و شرط برقرار باشد، کاهش می‌یابد. مزیت این کلید این است که در صورتی که کاربر کلید را فشار داده و نگه دارد تقریبا در هر 200 میلی‌ثانیه یکبار کار مورد نظر صورت می گیرد. عیب این روش نیز این است که هنوز احتمال دارد که زمانی که یکبار کلید زده شود دوبار کار مورد نظر انجام شود.	**if(PINB.0==0) {** **دستورات مربوط به بعد از زدن کلید** **delay_ms(200);** **}**
توضیح: به محض فشار دادن کلید توسط کاربر شرط if برقرارشده و برنامه به مدت 20 میلی‌ثانیه صبر می کند تا منطق کلید ثابت شود و از منطقه bounce عبور کند سپس توسط حلقه while با همان شرط برقراری کلید در این مرحله برنامه تا زمانی که کلید توسط کاربر فشرده‌شده است در حلقه گیر می‌کند و هیچ کاری انجام نمی‌دهد. به محض اینکه کاربر دست خود را بر می‌دارد، شرط برقرار نبوده و خط بعدی یعنی دستورات مربوطه اجرا می‌شود. مزیت این روش این است که در هر بار فشردن کلید برنامه تنها یکبار اجرا می شود. معایب این روش این است که تا زمانی که کاربر کلید را نگه داشته اتفاقی نمی‌افتد و به محض رها کردن کلید کار مورد نظر انجام می‌شود.	**if(PINB.0==0) {** **delay_ms(20);** **while(PINB.0==0);** **دستورات مربوط به بعد از زدن کلید** **}**
توضیح: این کلید به صورت start/stop عمل می‌کند یعنی بار اولی که کاربر کلید را فشار می‌دهد دستورات مربوط به بعد از زدن کلید دائما اجرا می‌شود تا زمانی که کاربر دست خود را از روی کلید رها کرده و دوباره کلید را فشار دهد، در این صورت دستورات دیگر اجرا نمی‌شود. دو متغیر از نوع bit با نام‌های flag و start با مقدار اولیه صفر برای این کلید باید تعریف شود. زمانی که کاربر برای اولین بار کلید را فشار می‌دهد شرط if برقرارشده و flag=1 وstart=1 می‌شود. در این صورت شرط if دوم برقرار بوده و دستورات مربوطه با هر بار چرخش برنامه درون حلقه نامتناهی while یکبار اجرا می‌شود. زمانی که کاربر دست خود را از روی کلید بر می‌دارد و منطق یک وارد میکروکنترلر می‌شودflag=0 شده و برنامه دوباره آماده این می‌شود که کاربر برای بار دوم کلید را فشار دهد. زمانی که کاربر بار دوم کلید را می‌فشارد start=0شده و دستورات مربوطه اجرا نخواهد شد سپس با برداشته شدن دست کاربر از روی کلید، همه چیز به حالت اول بر می‌گردد. این کلید طوری نوشته‌شده است که bounce در آن کمترین تأثیر مخرب ممکن را دارد.	**if((PINB.0==0) && (flag==0))** **{** **flag=1; start=!start; }** **else if (PINB.0 == 1) flag=0;** **if(start){** **دستورات مربوط به بعد از زدن کلید** **}**