**"به نام یزدان پاک"**



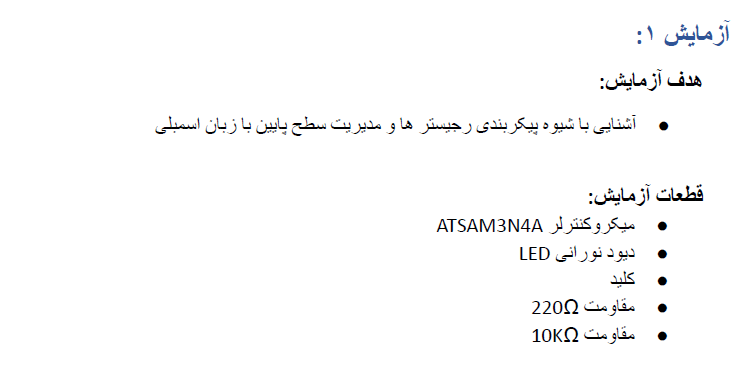
**گزارش آزمایش اول اسمبلی**

**اعضای گروه:**

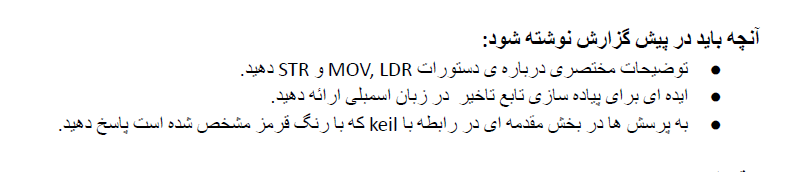
**محمد چوپان9831125**

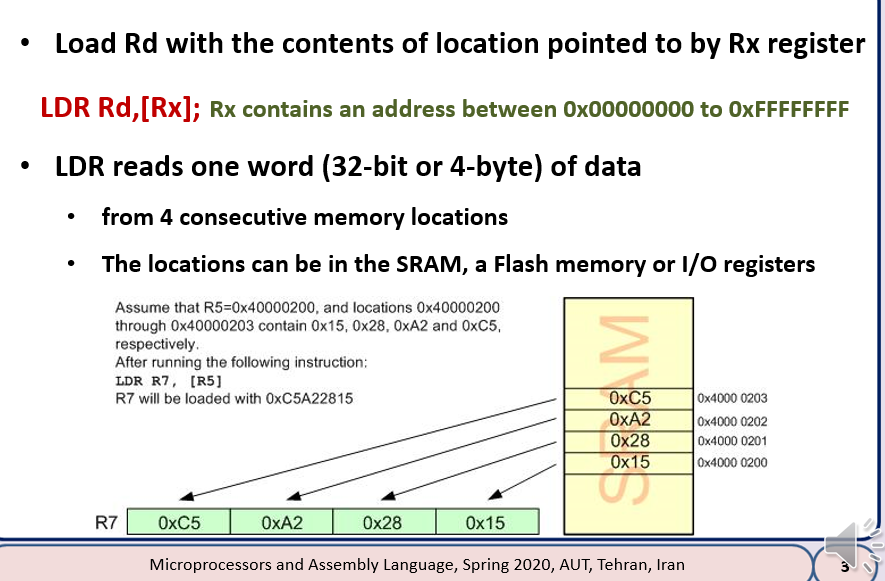
**محمد سپهر توکلی کرمانی 9831111**

**تاریخ آزمایش : 09/10/1400**

****

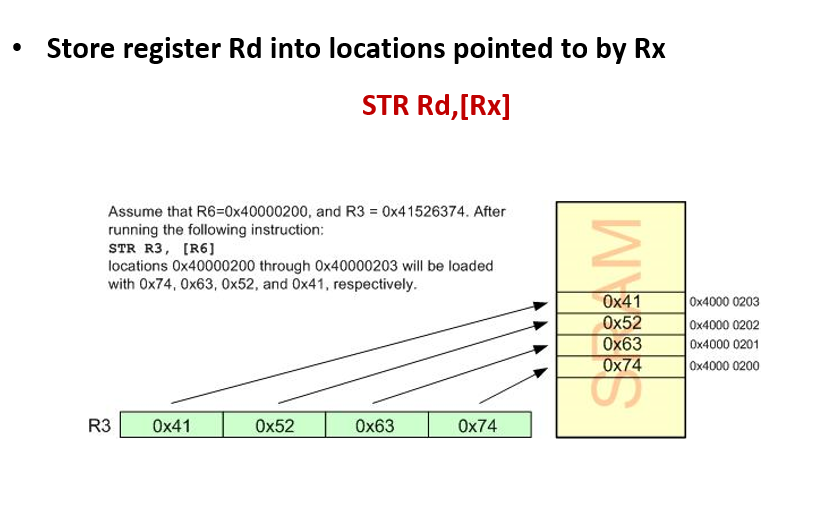
در این آزمایش باید توسط اسمبلی برنامه ای بنویسیم که بتواند چراغ هایی که ب پردازنده ما متصل است را روشن کند و برای راه انداختن هر کدام باید وقفه مرتبط با آن را اجرا کنیم.

  
**دستور ldr :** این دستور به معنای load است که مقدار موجود در آن ادرس را در رجیستر مورد نظر ما ذخیره میکند همانند عکس زیر:



**دستور mov :** مقادیر موجود در یک رجیستر را به رجیستر دیگر انقال می دهد که حال عملوند دوم میتواند یک عدد نیز باشد که با علامت # میتوان یک عدد را مشخص کرد و یا \_ مبنای آن را

**دستور str :** این دستور دستور store که عکس دستور ldr است و مقادیر موجود در یک رجیستر را درون حافظه می گذارد.

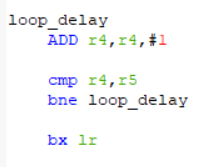


حال با دستوراتی مانند ldrb &strb میتوان تنها یک بایت را بار گذاری کرد.

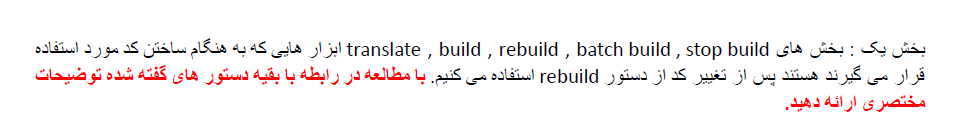
و نحوه قرار گرفتن در حافظه نیز طوری است که بیت کم ارزش تر در حافظه کمتر قرار میگیرد.



برای پیاده سازی این تابع از یک حلقه می توانیم استفاده کنیم که با استفاده از آن برای مثال 100 با یک حلقه را پیمایش کند .







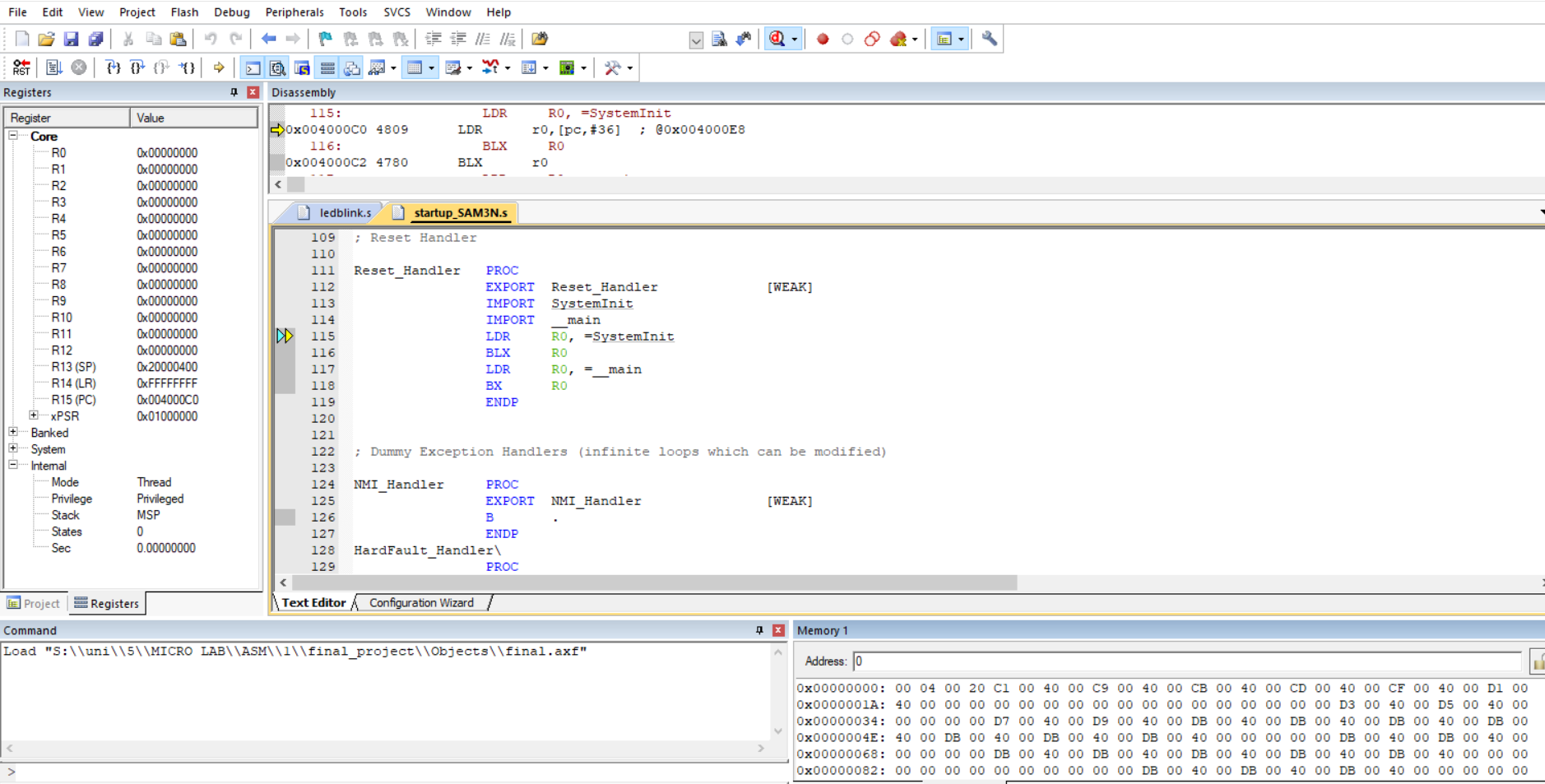
**دستور stop build :** این دستور تنها در هنگام build فعال است و build را متوقف میکند.

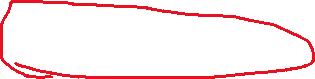
**دستور batch build :** این دستور compile & assemble میکند فایل هایی را که عوض شدند و object file آن ها را می سازد.

**Rebuild** : تمامی فایل های یک پروژه را از ابتدا build میکند.

**Build :** یک فایل را build کرده و object مورد نظر آن را می سازد.

**Translate** : برای مثال وقتی از متغیر ها و یا از زبان c یا زبان های دیگر در میان کد اسمبلی استفاده می کنم این ها باید ترجمه شوند و به deassmebler که بتواند دستور ها را اجرا کند.



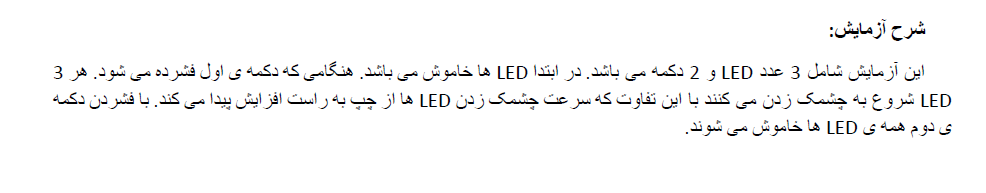


همانطور که میبینید علاوه بر این ها می توان ساختار deassmbler و یا حافظه را نیز دید و debug کرد.

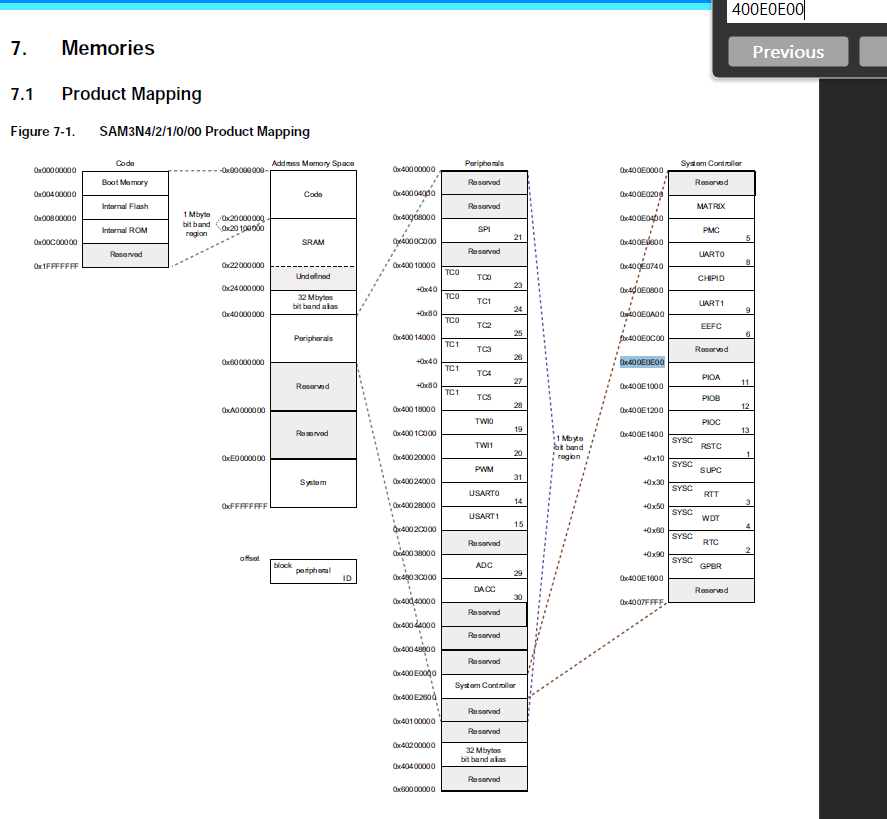


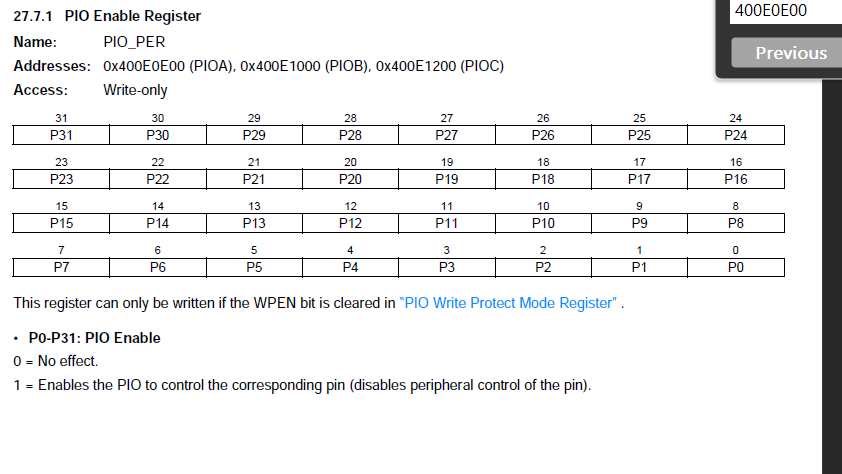
بخش اول که reset-handler می باشد در ابتدای اجرا هر باره میکرو اجرا می شود و isr و پایه های pio را از ابتدا اجرا کرده و شمارنده برنامه را برای init کردن سیستم و خواندن برنامه که نوشته شده روی آن آماده میکند.

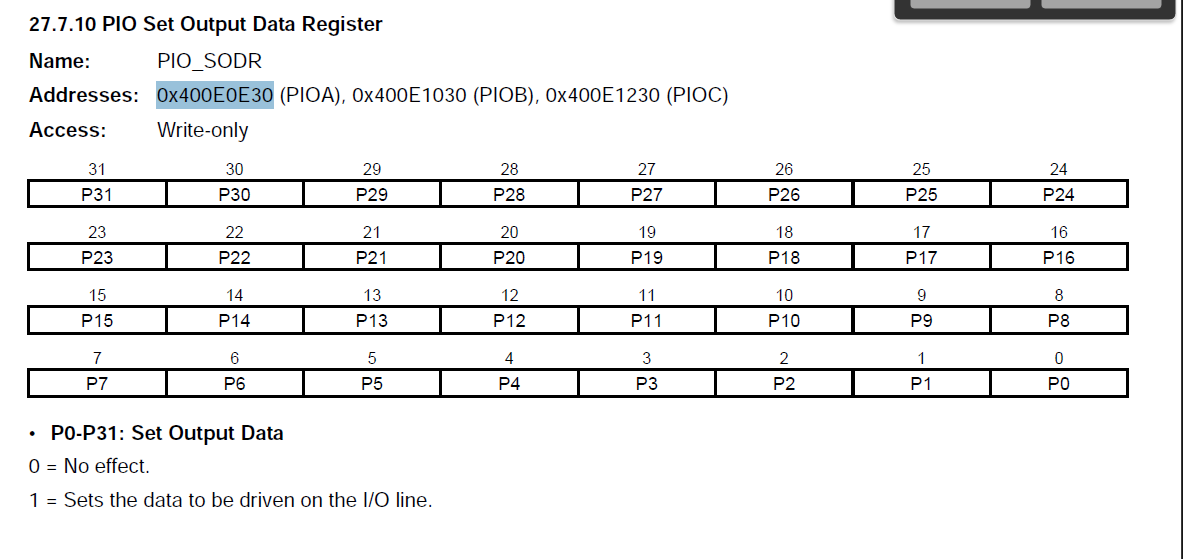
Interrupt vector: جدول وقفه ها است که وقفه مرتبط به هر قسمت در کجا اجرا می شود. کدام وقفه مرتبط با چی است.

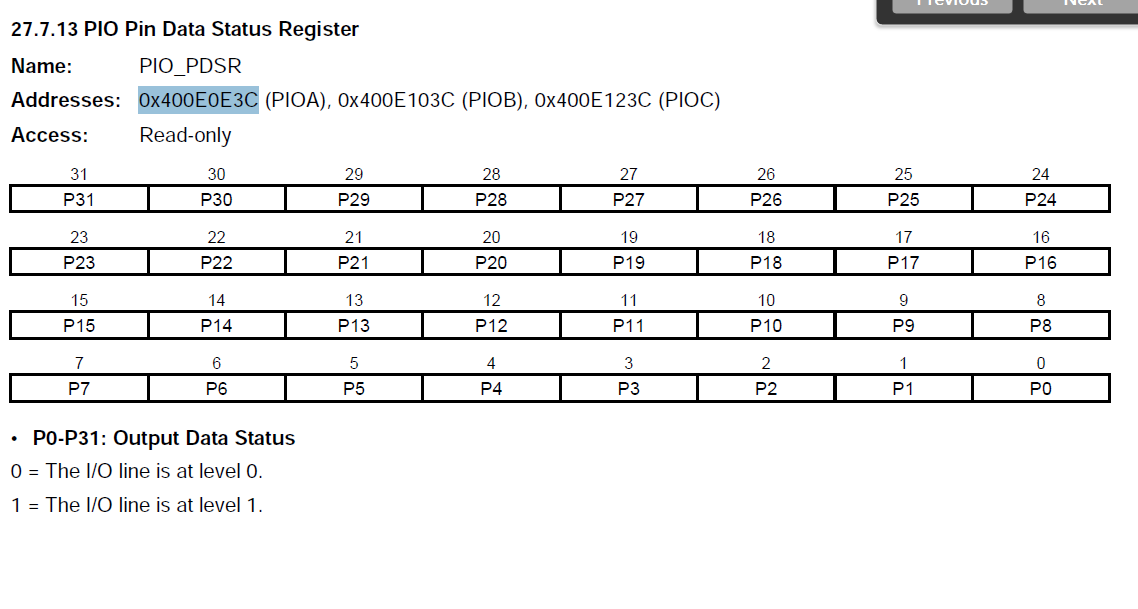


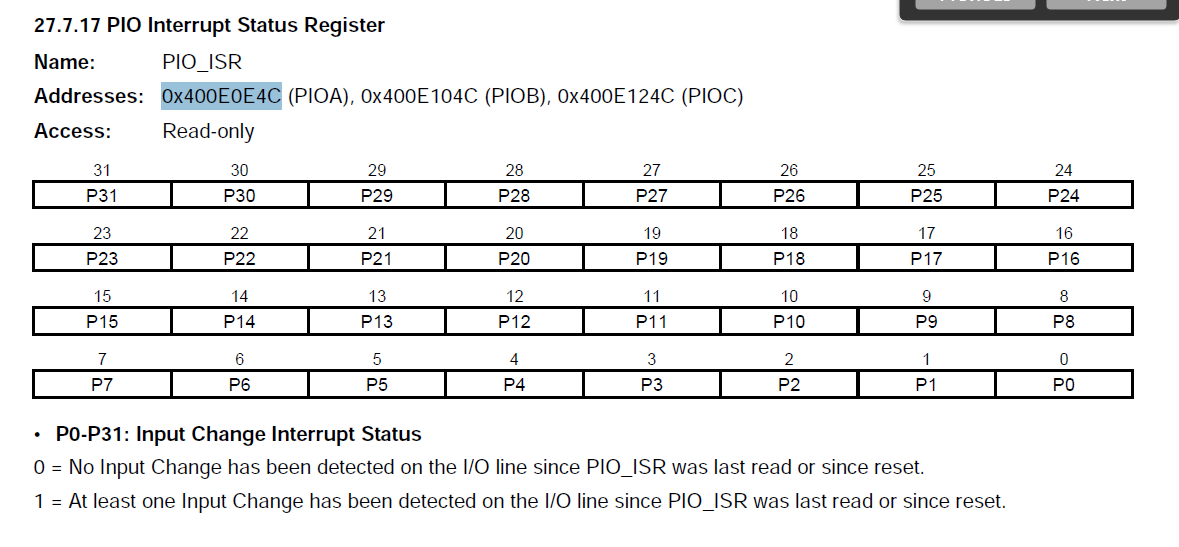
حال ابتدا باید مقدار پایه های pio را از دیتا شیت پردازنده خوانده و تعریف کنیم.



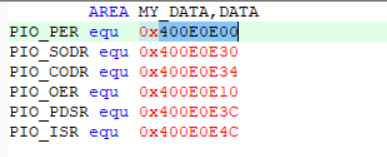




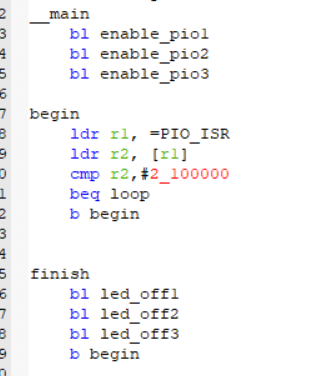


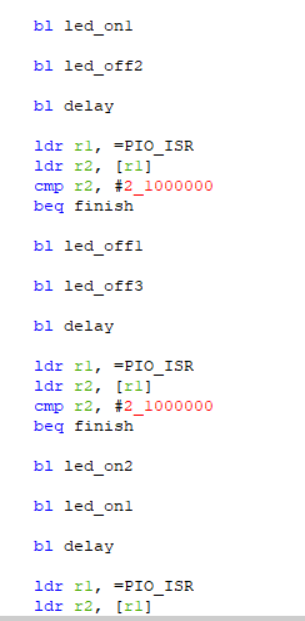
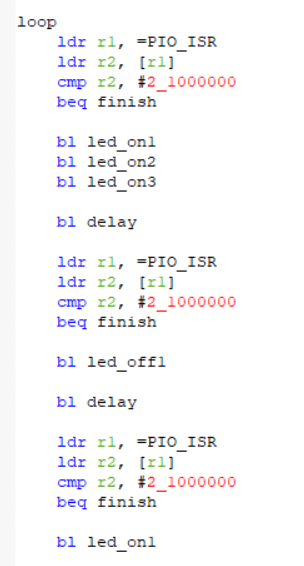


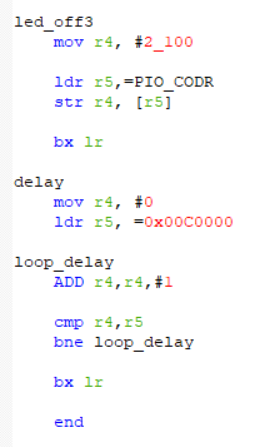
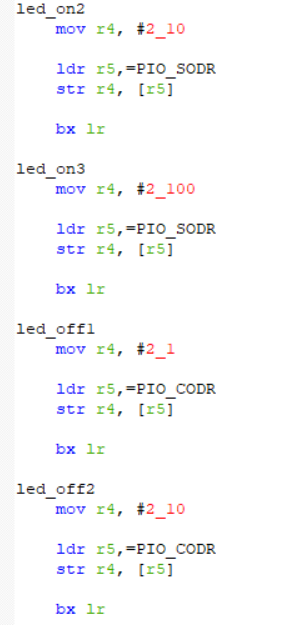
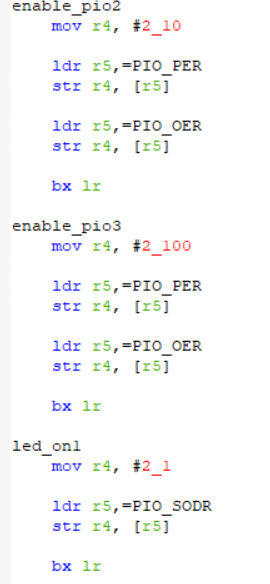
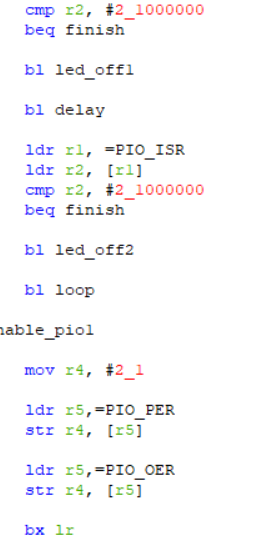
کد را مطابق این داده ها تنظیم میکنیم.



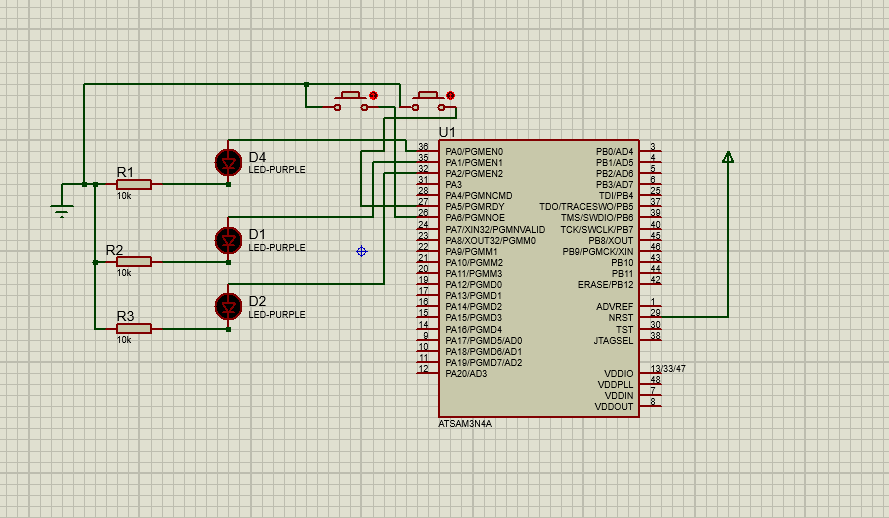
باقی کد را پیاده سازی میکنیم.



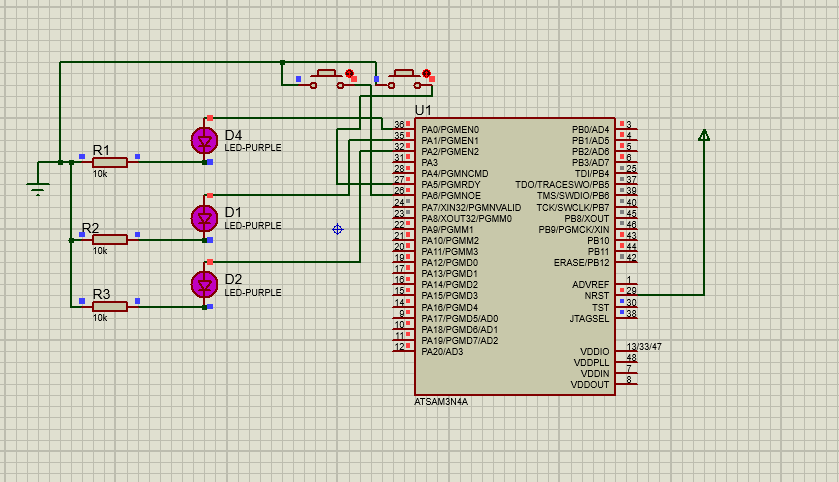




فایل hex را خروجی میگیریم . سپس مدار را پیاده سازی میکنیم.



وقتی کلید 1 را فشار می دهیم:



و کلید 2:

