بسمه تعالی

دانشگاه صنعتی شریف

واحد کنترل ریزبرنامه‌ریزی‌شده

درس:

آزمایشگاه معماری کامپیوتر

کیان بهادری (۹۹۱۰۵۳۱۲)
امیرحسین براتی (۹۹۱۰۱۳۰۸)

هدف آزمایش:

در این آزمایش مدار کنترل کامپیوتر ساخته شده در آزمایشهای پنجم ، ششم، و هفتم را به صورت ریزبرنامه‌پذیر طراحی و پیاده‌سازی می کنیم .

شرح آزمایش:

برای این بخش کافی است همان بخش کنترل‌کننده مدار را به صورت ذخیره شده در مموری قرار دهیم. برای اینکار کافی است سیگنال های کنترلی مدار را در مموری کنار دستور هم قرار دهیم. در واقع این کار باعث می‌شود که حجم دیکود کردن ما در مدار به شدت کاهش یابد. به طور مثال دیگر نیازی به دیکود کردن برای یافتن رجیستر مقصد نداریم. می‌توانیم برای اینکار سیگنال‌های Enable رجیسترها را به صورت یک تک بیت در کنار دستور قرار دهیم. سپس کارایی مدار را بررسی کنیم.

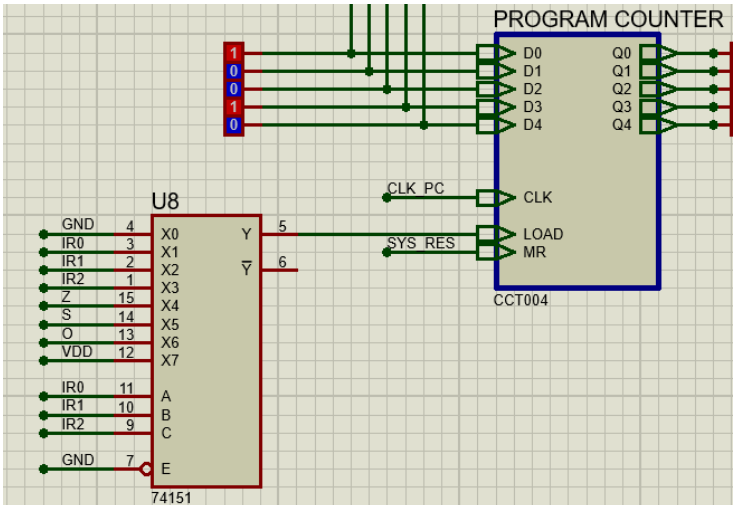
پیاده‌سازی آزمایش:

اکنون نوبت به پیاده سازی مدار مربوطه به این آزمایش می‌رسد. همان طور که گفته شد مدار را به سه بخش اساسی تقسیم می کنیم. بخش محاسبات مربوط به ثبات‌ها، بخش مربوط به دستورات درون حافظه و بخش مربوط به مدار کنترلی. برای این که بتوانیم قسمت کنترلی را پیاده‌سازی کنیم، باید قسمت دستورات حافظه را هم تغییر دهیم. کافی است طول دستورات مدار را به صورت دلخواه افزایش دهیم. پس از اینکه این کار را انجام دادیم، هر کدام از بیت‌های حافظه را به عنوان یک بیت کنترلی در نظر میگیریم و بر اساس آنها بیت‌های کنترلی مدار را به بخش‌های مورد نظر متصل می کنیم.

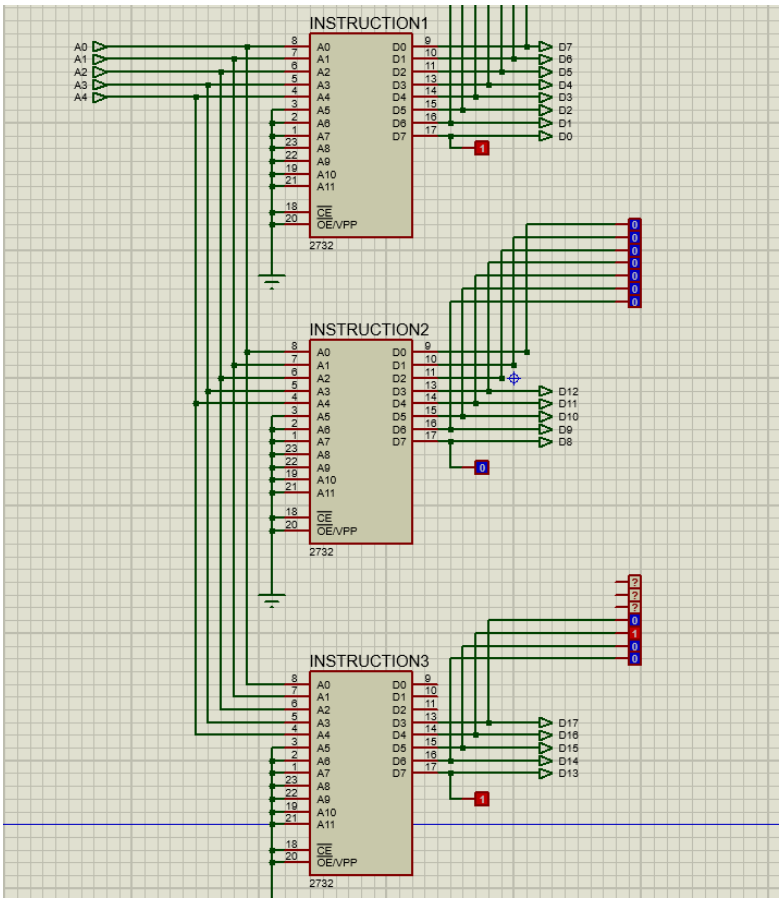
قسمت مربوط به محاسبات و alu نیازی به تغییر ندارد و تنها باید سیگنال‌های کنترلی آن را به صورت درستی متصل کنیم. بنابراین مستقیم به سراغ بخش پیاده سازی حافظه این مدار می‌رویم:

پیاده‌سازی بخش حافظه دستور:

ابتدا بخش گفته شده برای دستورات پرش را پیاده سازی می‌کنیم. کافی است که در این قسمت تنها load مربوط به PC را به بخش گفته شده در مدار متصل کتیم. بنابراین این قسمت به صورت زیر خواهد بود:



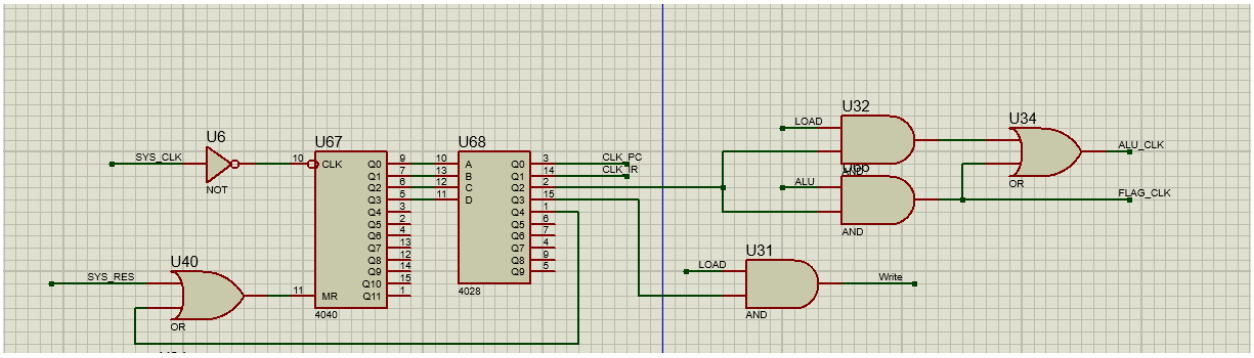
پس از این بخش به سراغ پیاده‌سازی بخش مربوط به حافظه می‌رسیم.



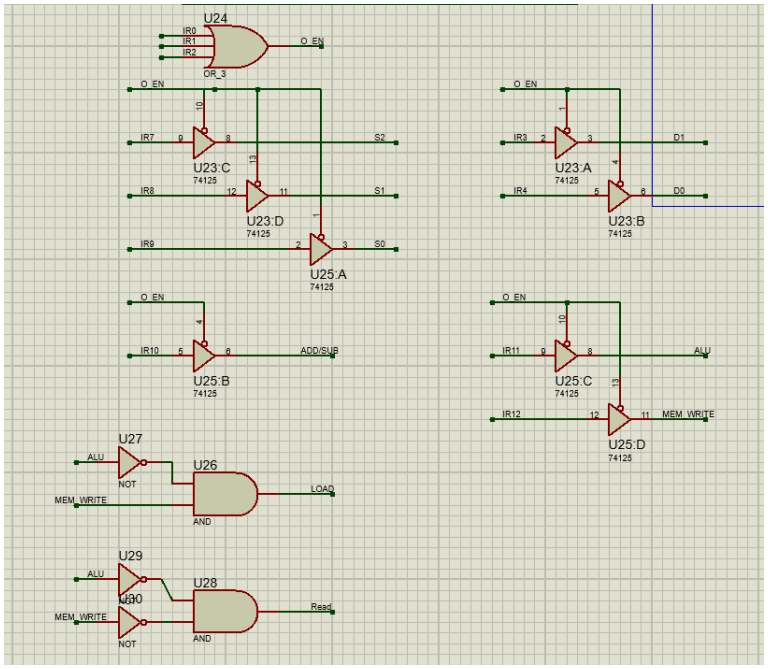
این بخش تنها شامل جندین حافظه است که به صورت موازی با ورود یک آدرس جدید مقدار موجود در این آدرس را به ما بار خواهد گرداند. این حافظه همانطور که نمایش داده شده است سه بخش دارد. بخش اول این مدار شامل ۸ بیت اول IR خواهد بود. سه بیت اول این مدار به قسمت مربوط به پرش متصل می‌شود. بیت چهارم تا هفتم این بخش به عنوان بیت‌های کنترلی برای load ثبات‌ها استفاده خواهند شد. بیت هشتم تا دهم برای سینگال‌های کنترلی ALU به کار خواهند رفت. بیت دهم به عنوان نشان‌دهنده عملیات مورد نظر برای ALU است که میتواند تنها جمع یا تفریق باشد. بیت یازدهم برای نشان دادن این است که آیا عملیات در حال انجام یک عملیات مربوط به حافظه است یا مربوط به ALU. بیت دوازدهم برای این به کار می‌رود که اگر عملیات مموری بود، تشخیص دهیم که این عملیات write است یا read. بیت سیزدهم تا هفدهم به عنوان آدرس یا مقدار مستقیم برای ثبات به کار برده می‌شود.

بخش کنترل‌کننده مدار:

اکنون به سراغ بخش کنترل‌کننده مدار می‌رویم. این بخش شامل یه sequencer است که به صورت زیر است:



برای اینکه سیگنال‌های مدار را پیاده‌سازی کنیم، از مدار زیر استفاده می‌کنیم:



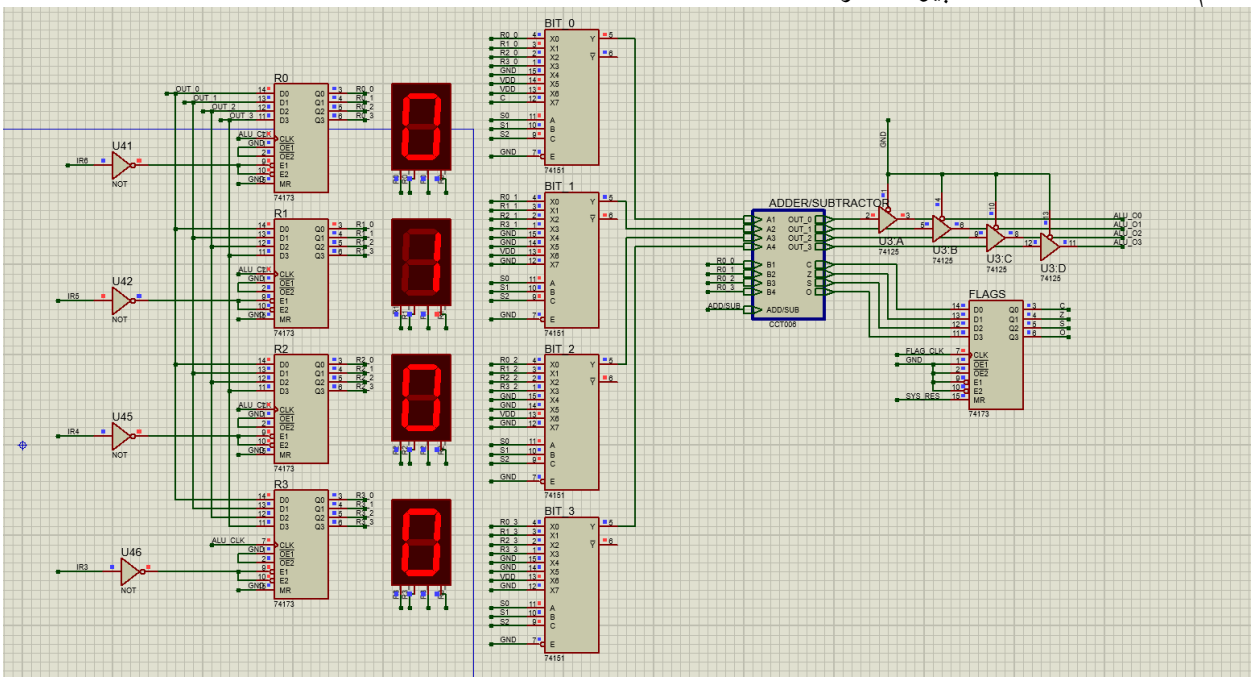
به این صورت بخش کنترلی مدار به این صورت پایان می‌یابد. تنها باید سیگنال‌های مربوطه را به عنوان ورودی به بخش ALU متصل کنیم. حال سراغ یک مثال از این مدار می‌رویم.

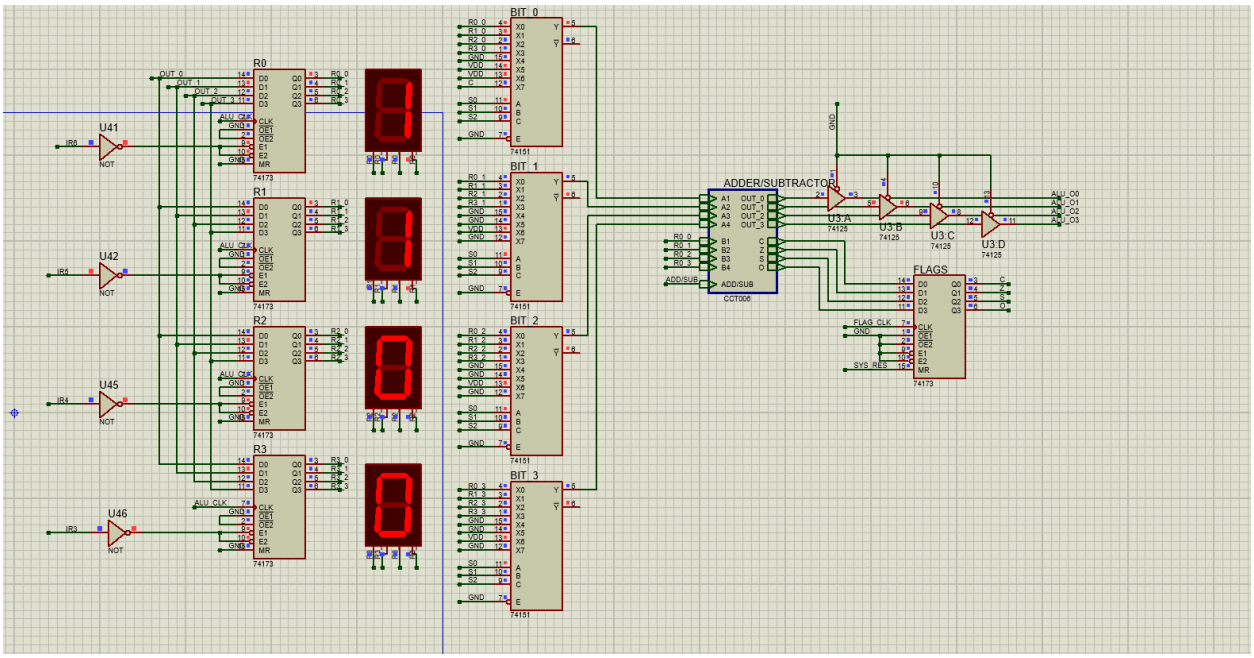
تست کردن مدار آزمایش:

برای تست این مدار، همان برنامه گفته شده، یعنی یافتن ۵امین عدد فیبوناچی استفاده میکنیم. برنامه به صورت زیر خواهد بود:

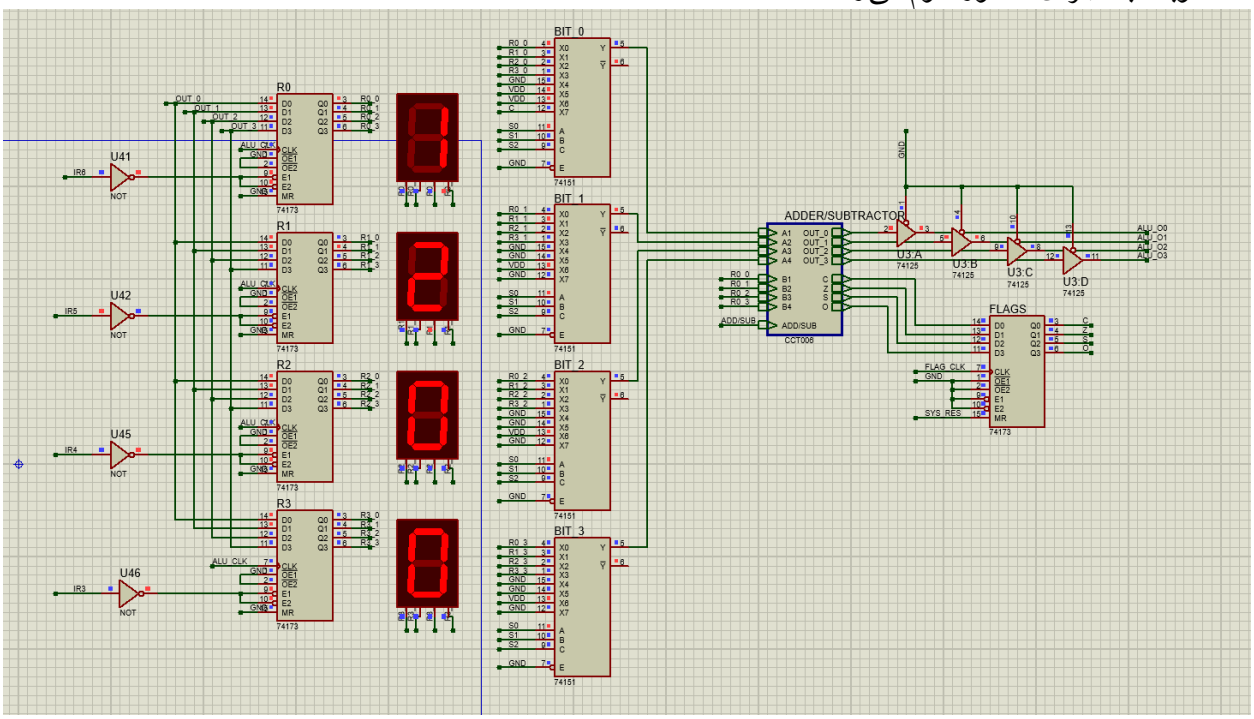
```
Sub R0, R0
Add R1, 1
Add R0, R1
Add R1, R0
Add R0, R1
Add R1, R0
Sub R0, R0
Add R0, R1
Store R0, #0000000
Jump #9
```

حال عکس‌های مداری که این دستورات را پیاده‌سازی کرده است را در زیر قرار می‌دهیم. فایل مربوط به مموری هر کدام از حافظه‌های مدار در پیوست قرار دارد.

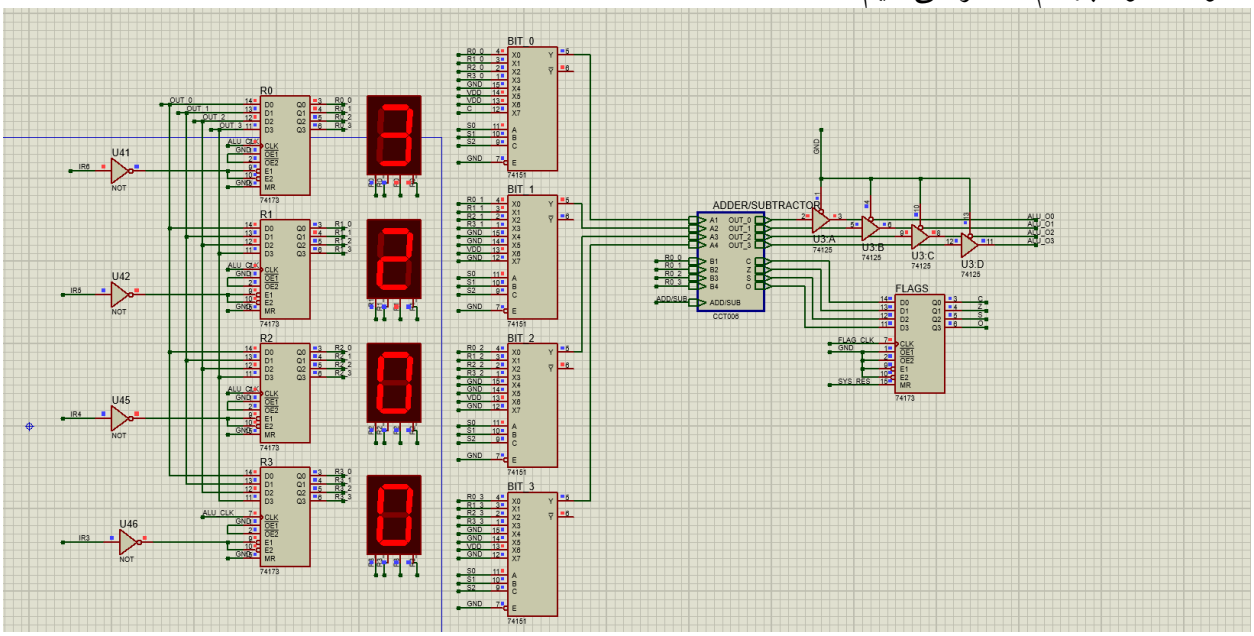




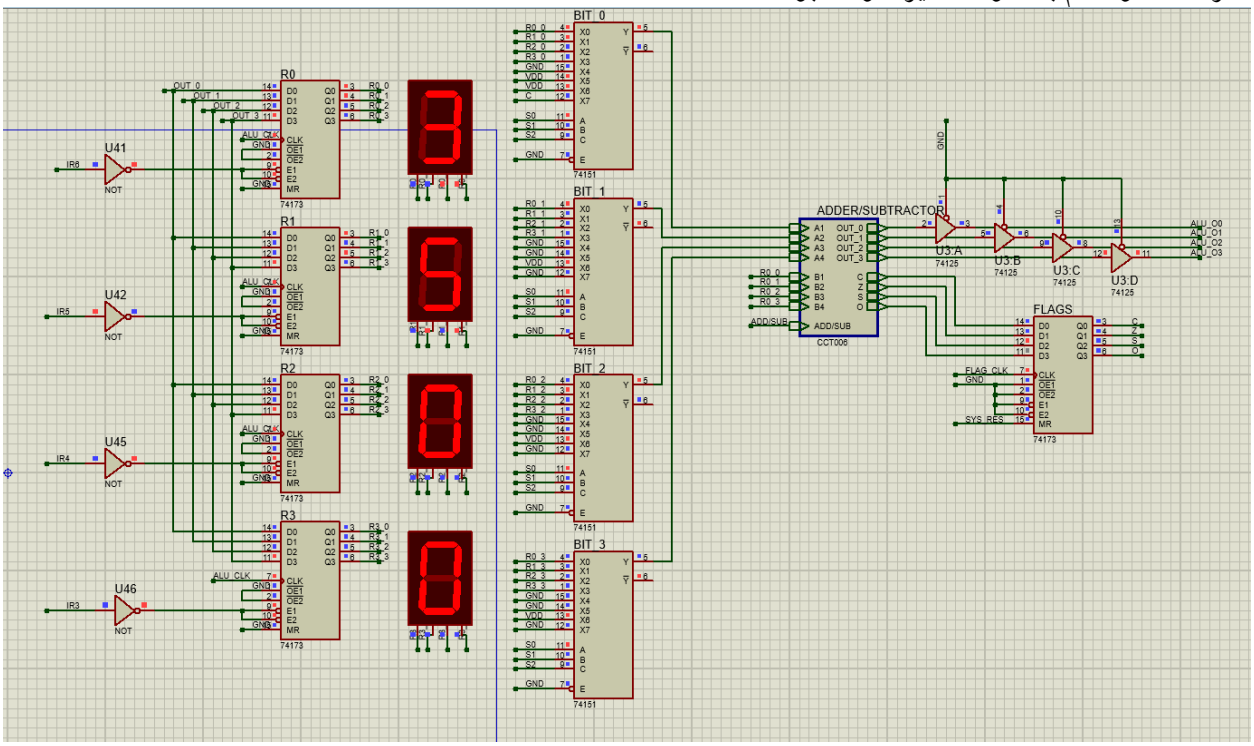
حال نوبت به اجرای دستور سوم می‌رسد:



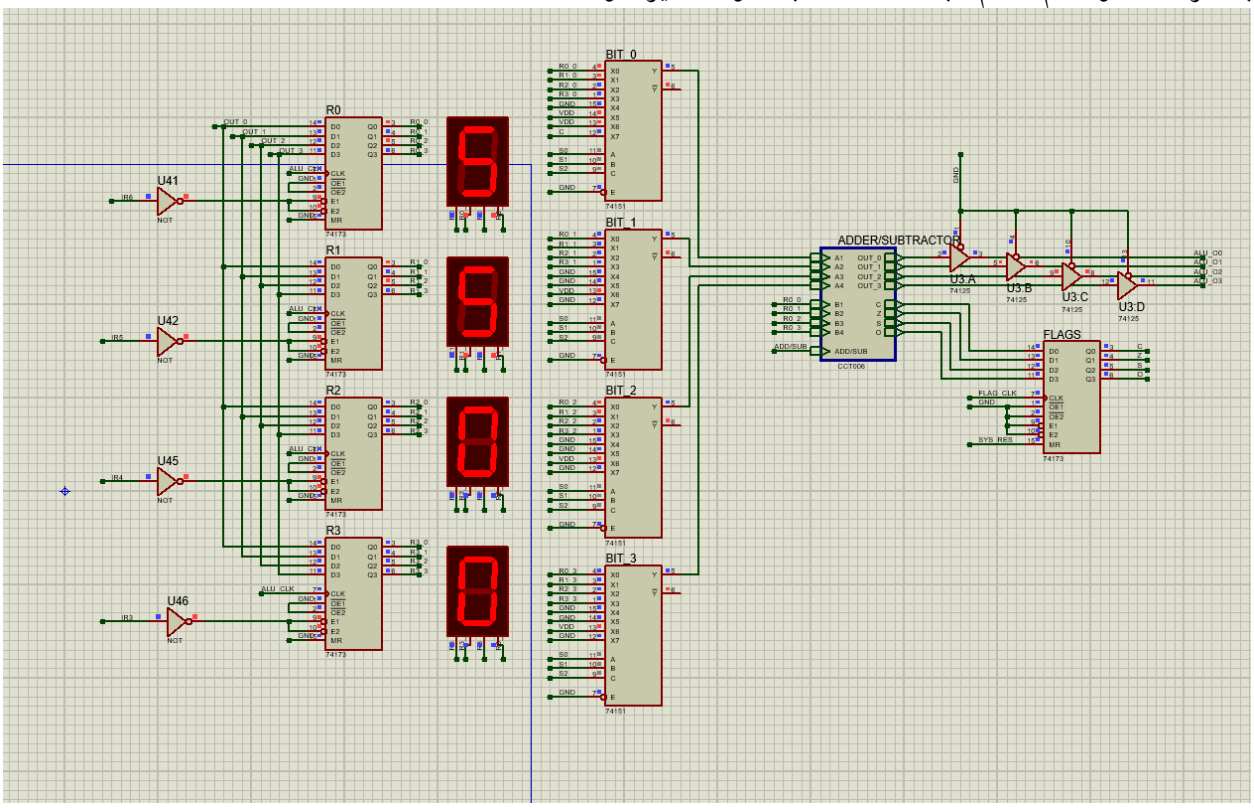
اکنون دستور چهارم را اجرا می کنیم:



اجرای دستور ۵ام به صورت زیر خواهد بود:



با اجرای دستور ۶ام و ۷ام، ثبات‌های مدار به صورت زیر خواهند شد:



با اجرای دستور ۸ام، تغییری در ثبات‌ها ایجاد نمی‌شود ولی مدار مقدار ثبات R0 را در آدرس صفرم حافظه قرار می‌دهد. دستور ۹ام، تنها یک پرش به همین خانه از حافظه است، یعنی برنامه تا بینهایت به همین آدرس از حافظه پرش خواهد کرد و تغییری در ثبات‌های مدار ایجاد نخواهد شد، تنها مقدار PC لود می‌شود و پس از اجرا شدن دستور دوباره به همین خانه پرش خواهد کرد.

پایان