باسمه تعالی

آزمایشگاه معماری کامپیوتر

دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف

تابستان ۱۴۰۳

استاد:

دکتر حمید سربازی آزاد

مهندس عطیه غیبی فطرت

اعضای گروه:

سعید فراتی کاشانی – ۴۰۱۱۰۶۲۹۹

زهرا آذر – ۹۹۱۰۹۷۴۴

امیرحسین صوری – ۴۰۱۱۰۶۱۸۲

فهرست عناوین

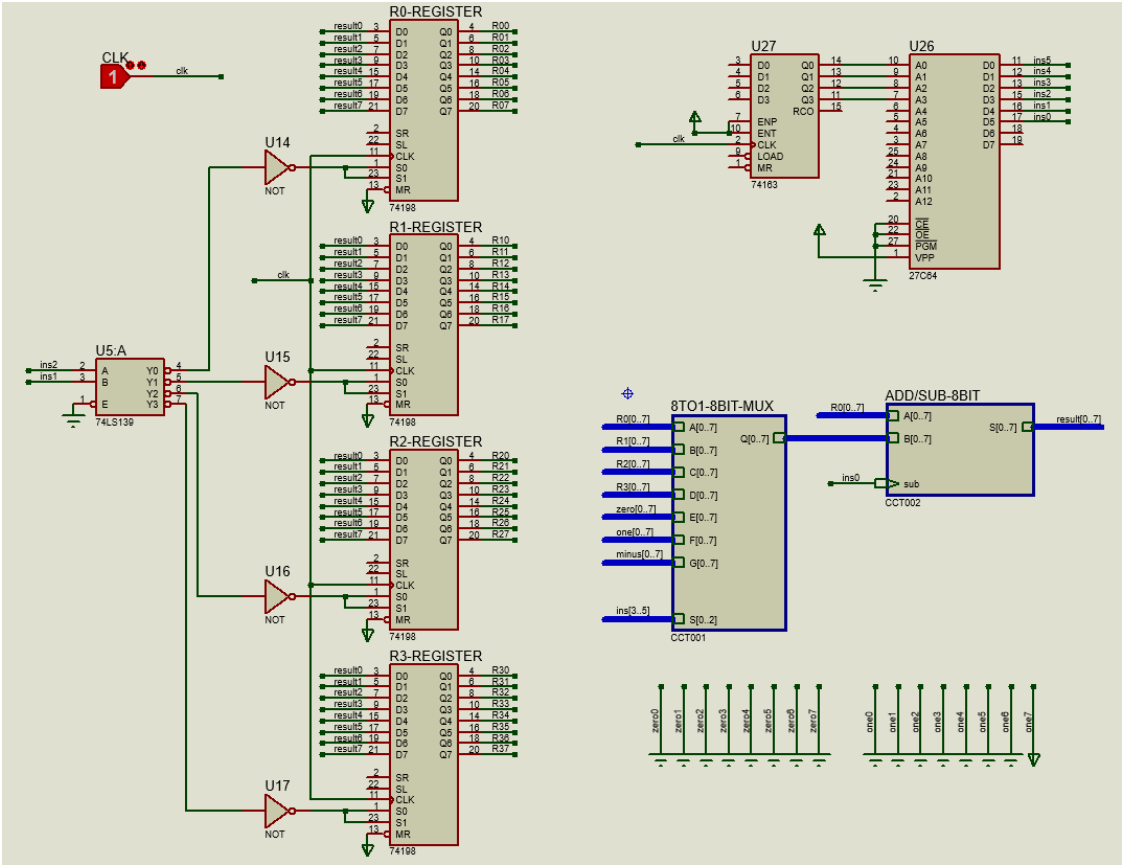
موضوع آزمایش:

کنترل توسط برنامه‌ی ذخیره شده در حافظه

شرح کلی آزمایش:

در این آزمایش، فرمان‌های مورد نیاز برای کنترل مدار آزمایش ششم از برنامه‌ای که در حافظه EPROM ذخیره شده است، گرفته می‌شود. این فرمان‌ها به ترتیب توسط یک شمارنده برنامه (PC) آدرس‌دهی شده و پس از واکشی از حافظه، اجرا می‌شوند. برای این منظور، باید مدارهای لازم به مدار آزمایش ششم افزوده شوند. در نهایت باید کدهای مربوط به محاسبه دنباله‌ی فیبوناچی در آن نوشته شود و جمله‌ها به صورت یکی در میان در R۰ و R۱ قرار بگیرند.

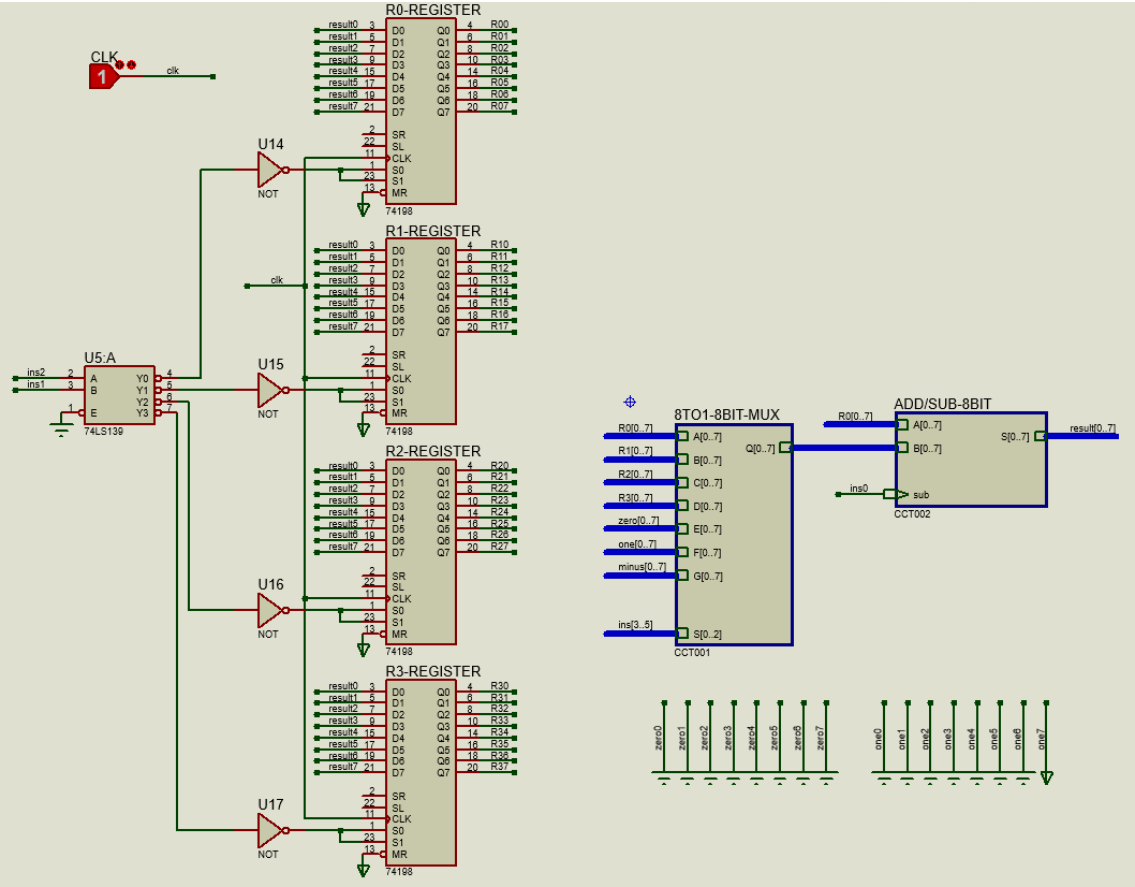
پیاده‌سازی مدار در پروتئوس:



شکل ۱ - تصویر کلی مدار

بخش قبلی مدار

بخش‌هایی که در تصویر زیر آمده‌اند مربوط به آزمایش ششم هستند و جزئیات مربوط به پیاده‌سازی آن در گزارش آزمایش ششم آمده است.

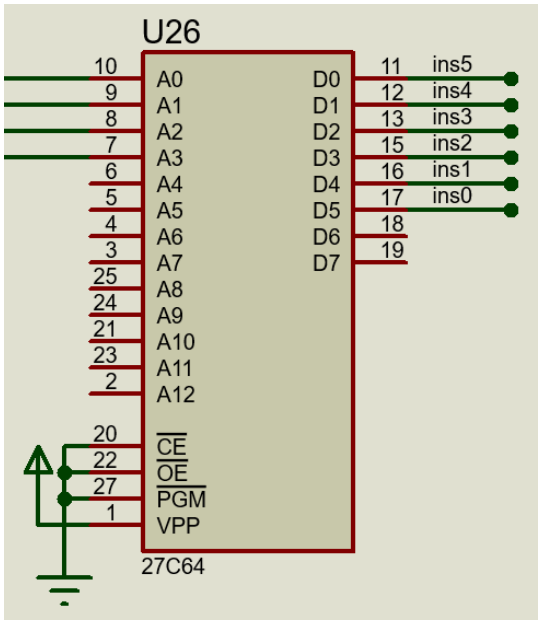


شکل۲ – بخش قبلی مدار

ذخیره‌ی دستورات در EPROM

توضیحات تراشه

ما در این آزمایش از تراشه‌ی ۲۷C۶۴ به عنوان EPROM استفاده کردیم. این تراشه ۸k×۸ را می‌تواند در خود جای دهد. از آن‌جا که دستورات ما ۶ بیتی است، صرفاً با ۶ بیت کم‌ارزش خروجی کار داریم و دو بیت دیگر برایمان کاربردی ندارند. هنگام ذخیره کردن برنامه نیز آن‌ها را ۰ در نظر می‌گیریم. همان‌طور که مشاهده می‌کنید آن ۶ بیت را به عنوان ۶ بیت دستور در نظر گرفتیم. بیت‌های An نیز مشخص می‌کنند که می‌خواهیم کدام سطر از حافظه را در خروجی مشاهده کنیم.



شکل۳ – تراشه‌ی ۲۷C۶۴

پین‌های باقی‌مانده نیز به صورت زیر هستند:

$\overline{CE}$	Chip Enable
$\overline{OE}$	Output Enable
$\overline{PGM}$	Program Enable
VPP	Programming Voltage

همان‌طور که مشاهده کردید، سه پین ابتدایی باید به GND و پین VPP باید هنگام خواندن به VCC متصل باشد.

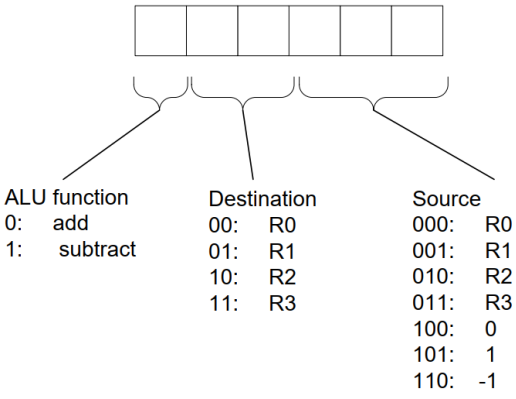
کد دنباله‌ی فیبوناچی

گام بعدی مربوط به نوشتن کد مدار در فایل HEX و قرار دادن آن در تراشه است. کد برنامه‌ی فیبوناچی را به صورت زیر نوشتیم. لازم به ذکر است که این قالب نوشتار به عنوان چرک‌نویس است و برای نوشتن فایل HEX صرفاً به ستون آخر که کد hex مربوط به هر دستور است نیاز داریم.

```
00100000, # r0 = r0 - r0, 20h
00000101, # r0 = r0 + 1 , 05h
00000001, # r0 = r0 + r1, 01h
00001001, # r1 = r0 + r1, 09h
00000001, # r0 = r0 + r1, 01h
00001001, # r1 = r0 + r1, 09h
          .....
00000001, # r0 = r0 + r1, 01h
00001001, # r1 = r0 + r1, 09h
00000001, # r0 = r0 + r1, 01h
00001001, # r1 = r0 + r1, 09h
```

شکل۴ - کد فیبوناچی

قالب دستورات نیز جهت یادآوری به صورت زیر است. لازم به ذکر است که دو بیت سمت چپ را با ۰ گسترش دادیم زیرا دستورات ما ۶ بیتی هستند.



شکل ۵ – قالب دستورات

ذخیره در قالب فایل HEX

حال باید کد آماده شده را در قالب فایل HEX ذخیره کنیم تا در برنامه قابل استفاده باشد.

قالب هر خط از فایل HEX به صورت زیر است:

:llaaaatt[dd...]cc

ll: تعداد بایت‌های داده

aaaa: آدرس شروع داده‌ها

tt: نوع ورودی

dd...: داده‌ها

cc: checksum تمام موارد قبلی

در این آزمایش داده‌ها (dd...) شامل تمام دستوراتی می‌شود که می‌خواهیم اجرا شوند. از آن‌جایی که هر دستور ۶ بیت است، با صفر قرار دادن دو بیت پرارزش می‌توانیم هر بایت از داده را معادل یک دستور در نظر بگیریم. بنابراین ll = تعداد دستورات ما خواهد بود. آدرس‌دهی ما از صفر شروع می‌شود بنابراین aaaa = ۰۰۰۰ است. نوع ورودی ما "رکورد داده" است. برای این نوع ورودی از tt = ۰۰ استفاده می‌شود. در نهایت برای محاسبه‌ی cc ابتدا تمام بایت‌های قبل از cc (با شروع از ll و تا انتهای داده‌ها) را جمع می‌زنیم و سپس مکمل دوم آن را محاسبه کرده و دو بایت کم‌ارزش خروجی را به عنوان cc به فایل اضافه می‌کنیم.

همچنین برای مشخص کردن اینکه فایل به اتمام رسیده است و داده‌ی دیگری در آن وجود ندارد، در انتهای فایل .hex عبارت زیر را اضافه می‌کنیم:

:۰۰۰۰۰۰۰۱FF

این مقدار نیز از فرمت قبلی پیروی می‌کند با این تفاوت که مقدار tt برای "رکورد EOF" معادل ۰۱ است. و چون داده‌ای نداریم ll = ۰۰ است. FF نیز checksum این مقادیر است.

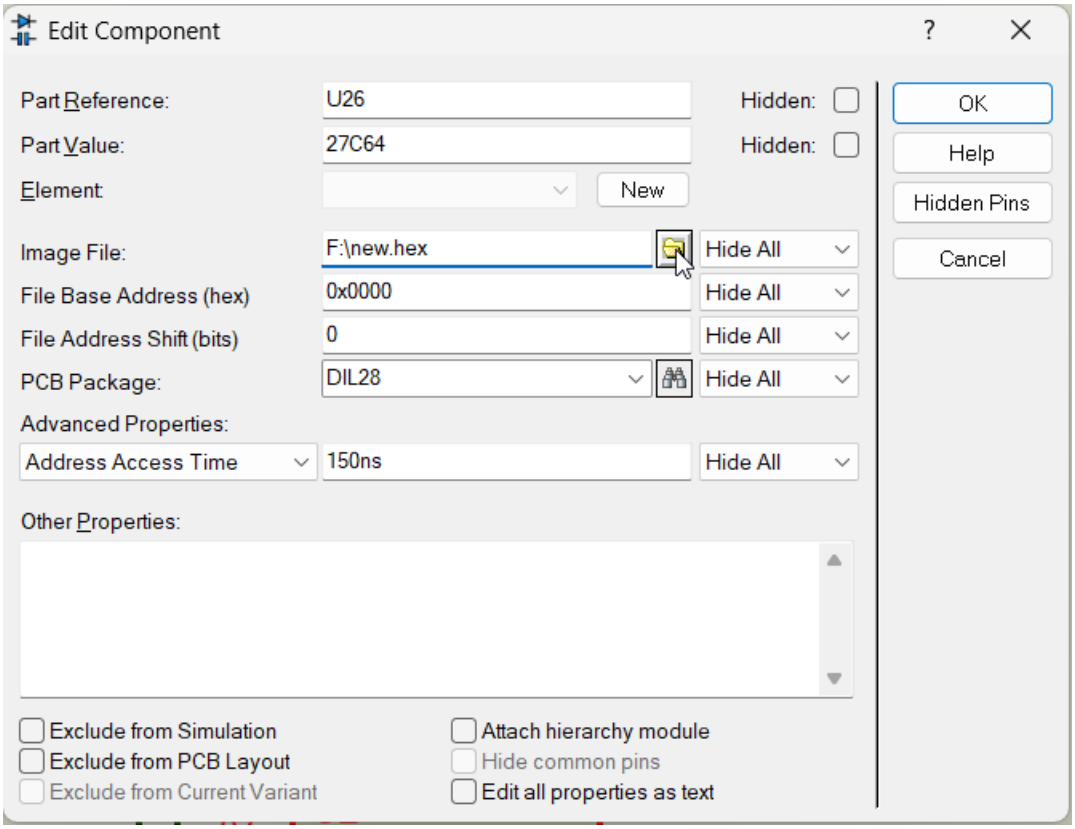
در نهایت نیز با توجه به توضیحات داده شده، محتوای فایل hex ما به صورت زیر شد:

```
:10000000200D0109010901090109010901097D
:00000001FF
```

شکل۶ – محتوای فایل برنامه

وارد کردن فایل HEX

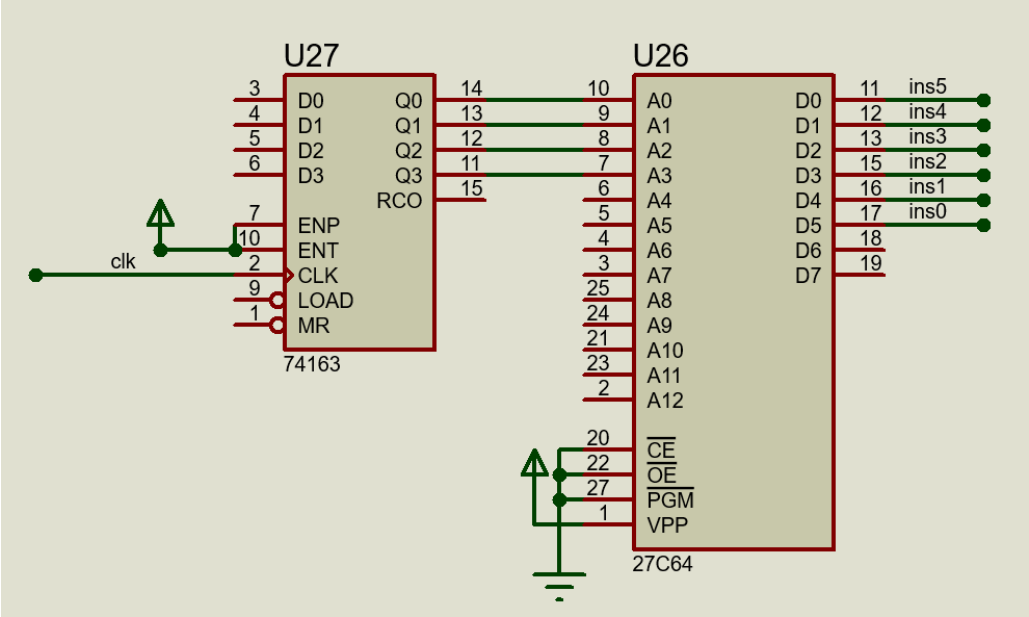
با کلیک کردن بر تراشه‌ی EPROM، با صفحه‌ی زیر مواجه می‌شوید. با استفاده از محل نشان داده شده، باید مسیر فایل HEX را مشخص کنید.



شکل ۷- وارد کردن فایل

شمارنده

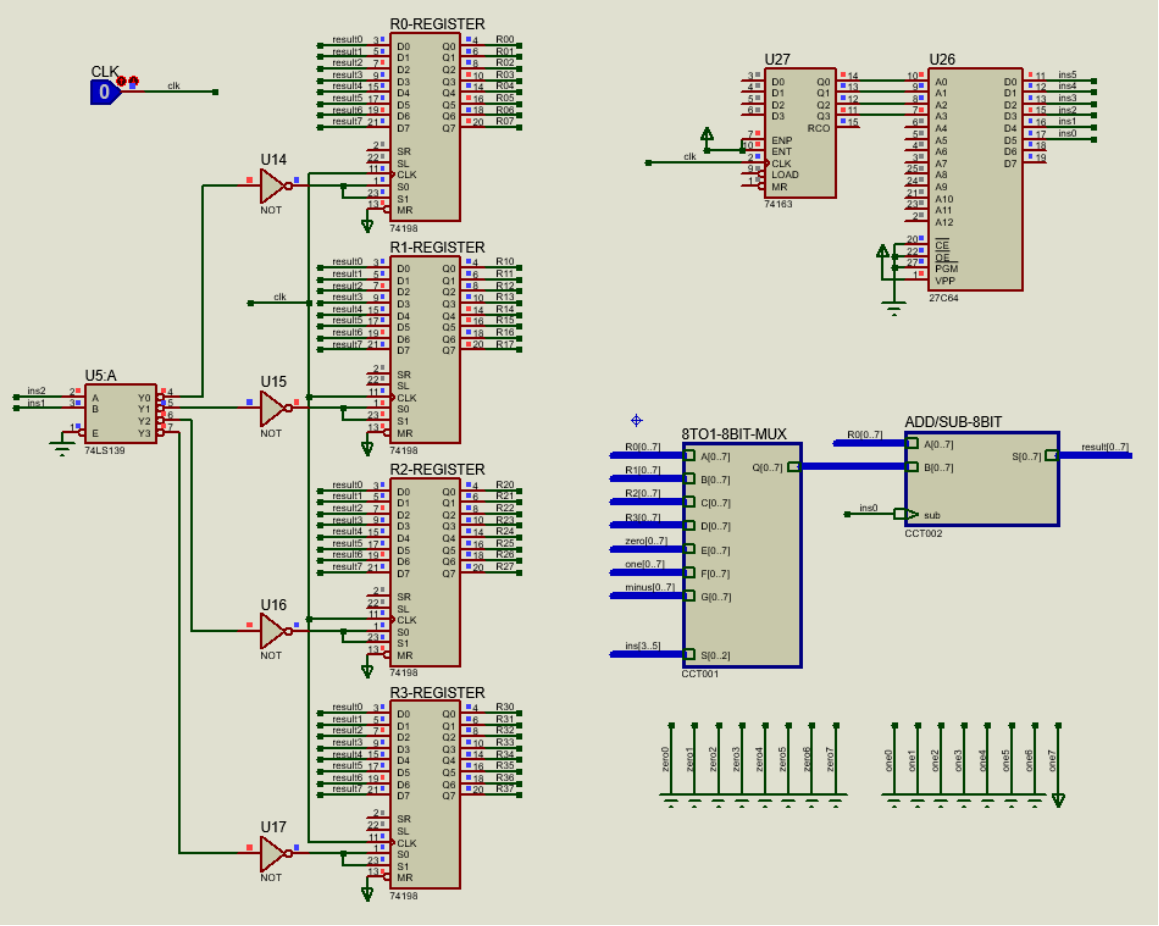
به عنوان گام نهایی، از یک شمارنده ۴ بیتی استفاده کردیم تا خانه‌های مورد نیازمان از حافظه را به ترتیب بخوانیم.



شکل ۸ – اتصال شمارنده به eprom

تست عملکرد مدار

وضعیت مدار پس از گذشتن چندین clock را در تصویر زیر می‌توانید مشاهده کنید. در این مرحله، اعداد ۱۳ و ۲۱ از دنباله‌ی فیبوناچی در R۰ و R۱ قرار دارند.



شکل ۹ - عملکرد مدار