باسمه تعالی

آزمایشگاه معماری کامپیوتر

A blue and black logo

Description automatically generated

دانشکده مهندسی کامپیوتر دانشگاه صنعتی شریف

تابستان 1403

**استاد:**

دکتر حمید سربازی آزاد

مهندس عطیه غیبی فطرت

**اعضای گروه:**

سعید فراتی کاشانی – 401106299

زهرا آذر – 99109744

امیرحسین صوری – 401106182

فهرست عناوین

[موضوع آزمایش: 3](#_Toc173267660)

[شرح کلی آزمایش: 3](#_Toc173267661)

[فرم کلی مدار 3](#_Toc173267662)

[پیاده‌سازی مدار در پروتئوس: 3](#_Toc173267663)

[محاسبه‌ی اختلاف توان‌ها **Error! Bookmark not defined.**](#_Toc173267664)

[یکسان کردن توان‌ها 6](#_Toc173267665)

[محاسبه‌ی مجموع یا اختلاف 8](#_Toc173267666)

[نرمال کردن حاصل مجموع یا اختلاف 9](#_Toc173267667)

[نمایش حاصل و سیگنال‌های خروجی 10](#_Toc173267668)

[تست عملکرد مدار: 11](#_Toc173267669)

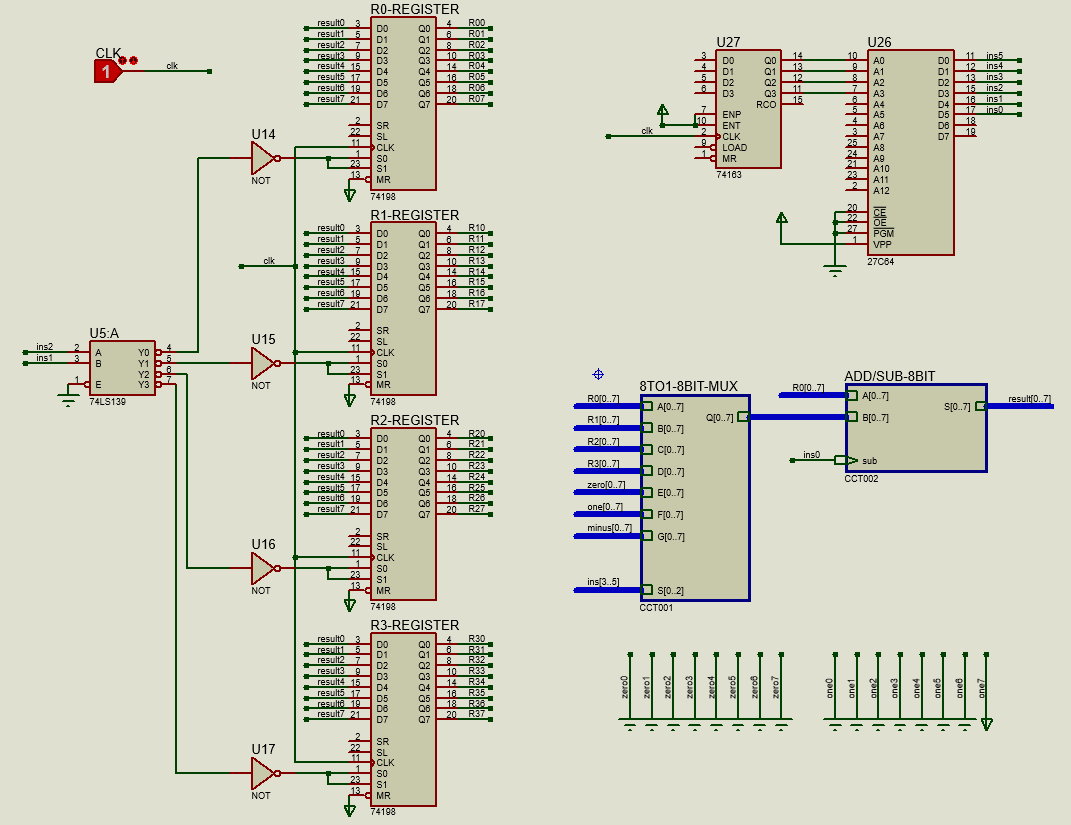
موضوع آزمایش:

کنترل توسط برنامه‌ی ذخيره شده در حافظه

شرح کلی آزمایش:

در اين آزمايش، فرمان‌های مورد نیاز برای کنترل مدار آزمايش ششم از برنامه‌ای که در حافظه EPROM ذخیره شده است، گرفته می‌شود. اين فرمان‌ها به ترتیب توسط يک شمارنده برنامه (PC) آدرس‌دهی شده و پس از واکشی از حافظه، اجرا می‌شوند. برای اين منظور، بايد مدارهای لازم به مدار آزمايش ششم افزوده شوند. در نهایت باید کدهای مربوط به محاسبه‌ دنباله‌ی فیبوناچی در آن نوشته شود و جمله‌ها به صورت یکی در میان در R0 و R1 قرار بگیرند.

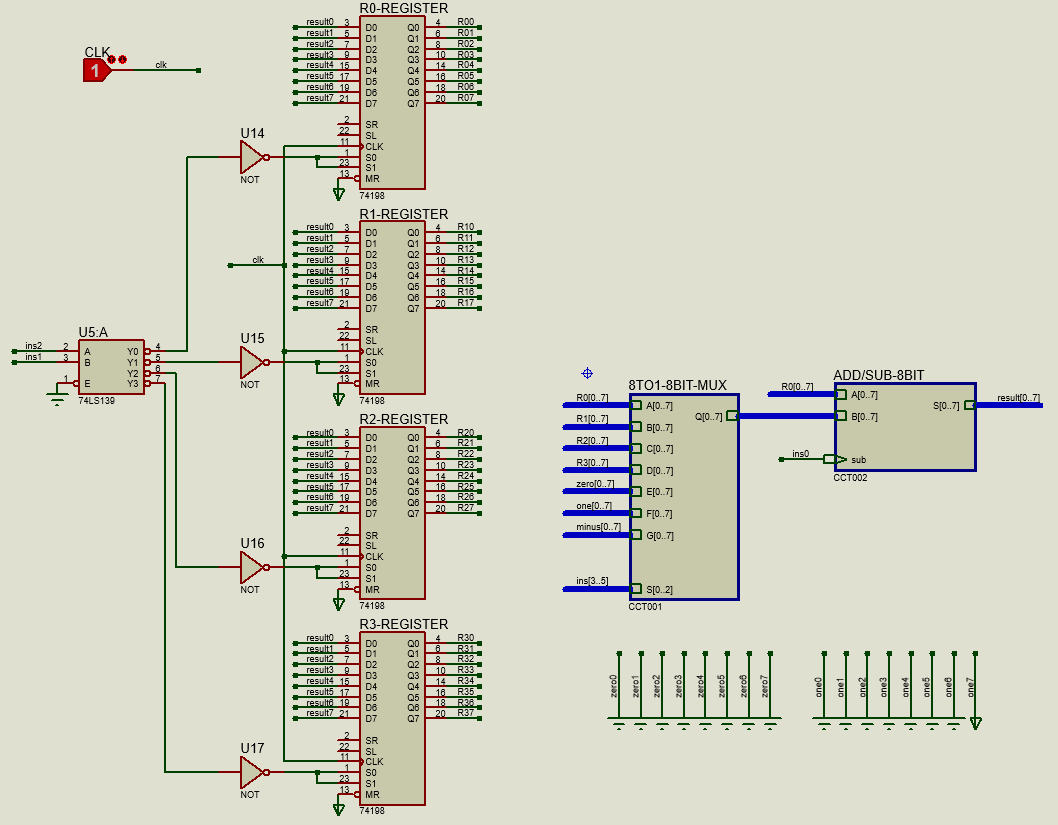
پیاده‌سازی مدار در پروتئوس:



شکل 1 - 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.

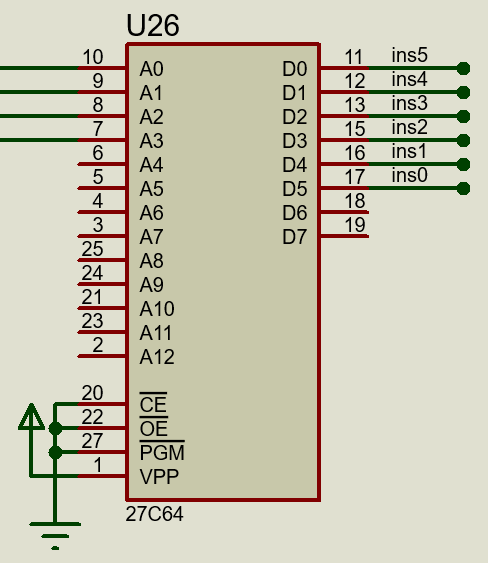


شکل 2 – بخش قبلی مدار

ذخیره‌ی دستورات در **EPROM**

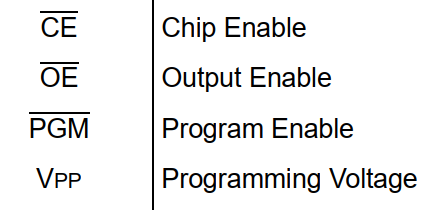
توضیحات تراشه

ما در این آزمایش از تراشه‌ی 27C64 به عنوان EPROM استفاده کردیم. این تراشه 8k×8 را می‌تواند در خود جای دهد. از آن‌جا که دستورات ما 6 بیتی است، صرفاً با 6 بیت کم‌ارزش خروجی کار داریم و دو بیت دیگر برایمان کاربردی ندارند. هنگام ذخیره کردن برنامه نیز آن‌ها را 0 در نظر می‌گیریم. همان‌طور که مشاهده می‌کنید آن 6 بیت را به عنوان 6 بیت دستور در نظر گرفتیم. بیت‌های An نیز مشخص می‌کنند که می‌خواهیم کدام سطر از حافظه را در خروجی مشاهده کنیم.



شکل 3 – تراشه‌ی 27C64

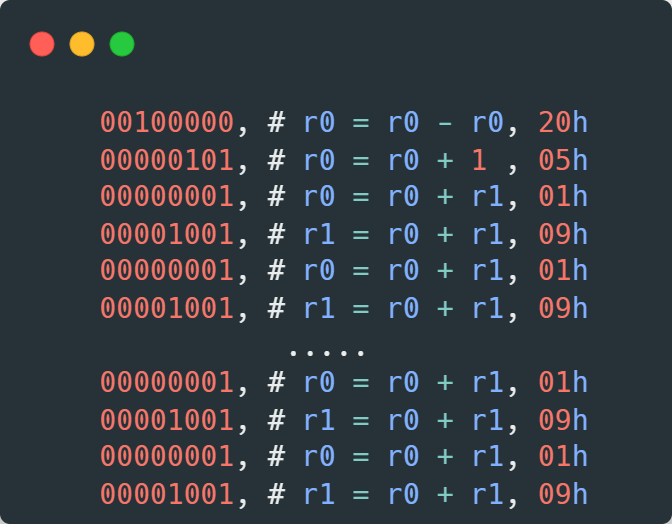
پین‌های باقی‌مانده نیز به صورت زیر هستند:



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.



شکل 4- کد فیبوناچی

قالب دستورات نیز جهت یادآوری به صورت زیر است. لازم به ذکر است که دو بیت سمت چپ را با 0 گسترش دادیم زیرا دستورات ما 6 بیتی هستند.

A diagram of a function

Description automatically generated

شکل 5 – 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

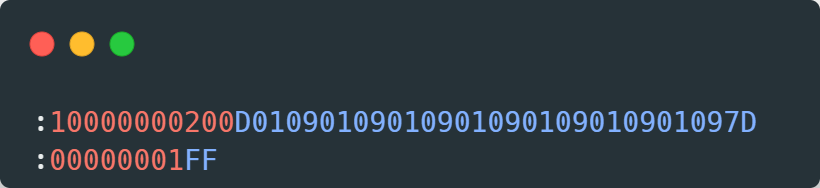
در این آزمایش داده‌ها (dd…) شامل تمام دستوراتی می‌شود که می‌خواهیم اجرا شوند. از آن‌جایی که هر دستور ۶ بیت است، با صفر قرار دادن دو بیت پرارزش می‌توانیم هر بایت از داده را معادل یک دستور در نظر بگیریم. بنابراین ll = تعداد دستورات ما خواهد بود. آدرس‌دهی ما از صفر شروع می‌شود بنابراین aaaa = 0000 است. نوع ورودی ما "رکورد داده" است. برای این نوع ورودی از tt = 00 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:

:00000001FF

این مقدار نیز از فرمت قبلی پیروی می‌کند با این تفاوت که مقدار tt برای "رکورد EOF" معادل 01 است. و چون داده‌ای نداریم ll = 00 است. FF نیز checksum این مقادیر است.

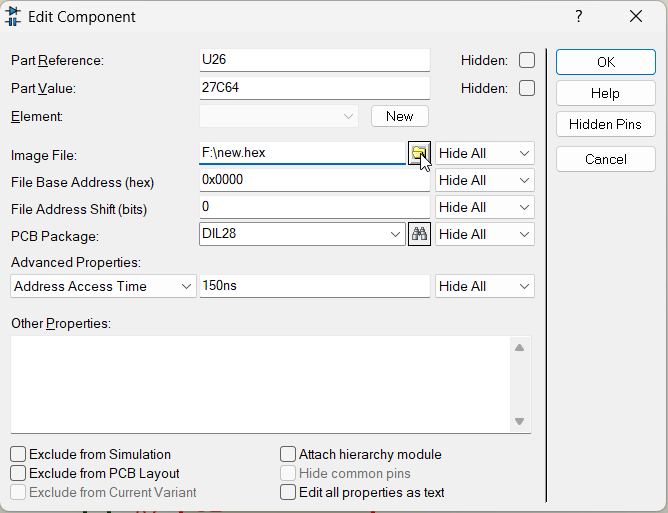
در نهایت نیز با توجه به توضیحات داده شده، محتوای فایل hex ما به صورت زیر شد:



شکل 6 – محتوای فایل برنامه

وارد کردن فایل HEX

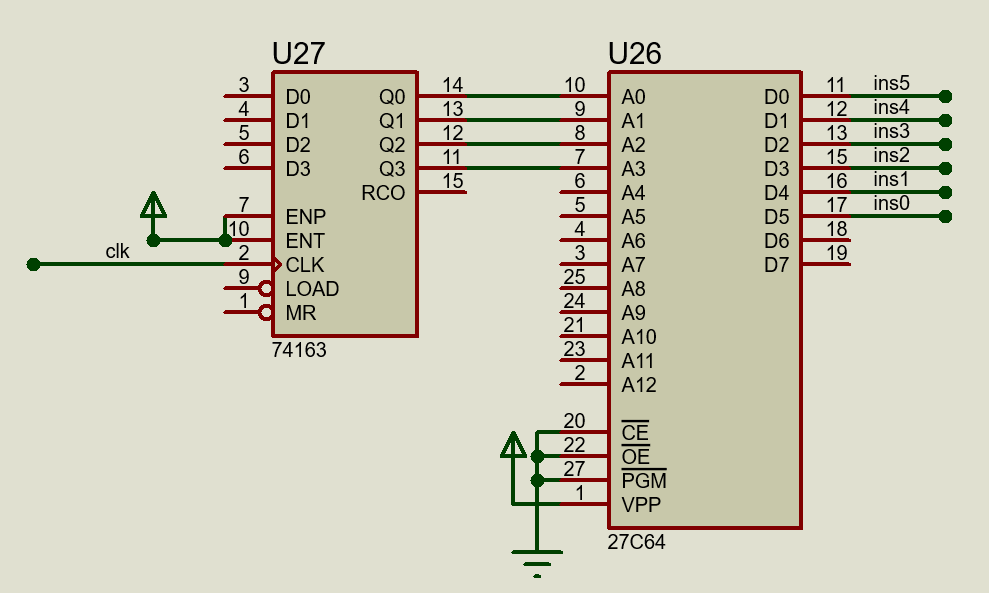
با کلیک کردن بر تراشه‌ی EPROM، با صفحه‌ی زیر مواجه می‌شوید. با استفاده از محل نشان داده شده، باید مسیر فایل HEX را مشخص کنید.



شکل 7- وارد کردن فایل

شمارنده

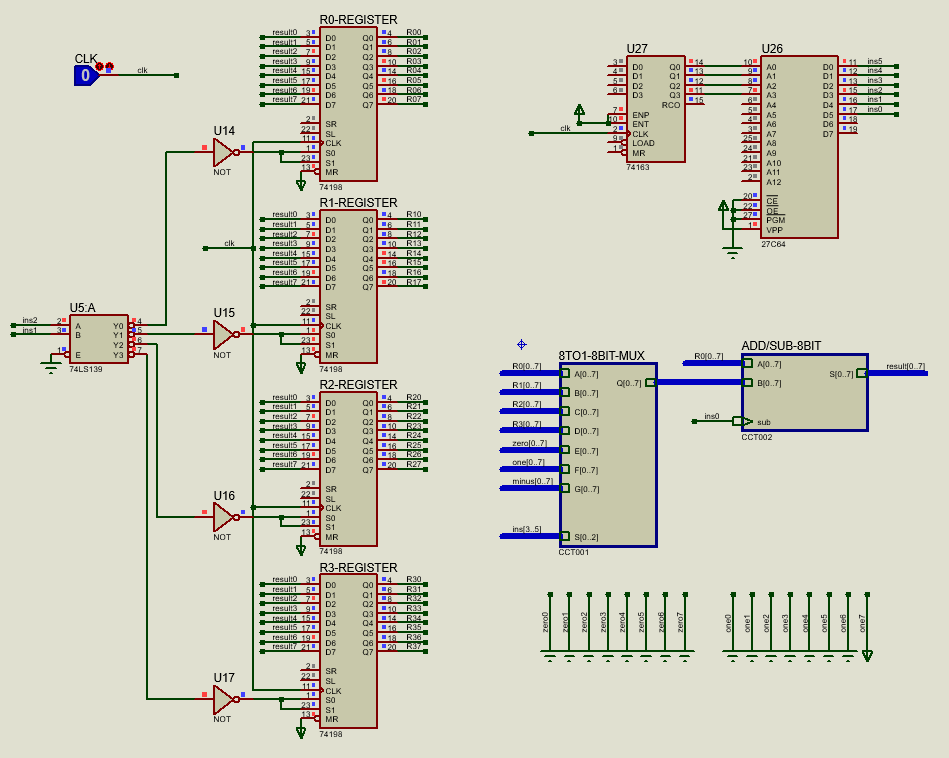
به عنوان گام نهایی، از یک شمارنده 4 بیتی استفاده کردیم تا خانه‌های مورد نیازمان از حافظه را به ترتیب بخوانیم.



شکل 8 – 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 13 و 21 از دنباله‌ی فیبوناچی در R0 و R1 قرار دارند.



شکل 9- عملکرد مدار