# گزارش آزمایش هفتم

# آزمایشگاه معماری کامپیوتر


**عنوان آزمایش: کنترل توسط برنامه ذخیره شده در حافظه**

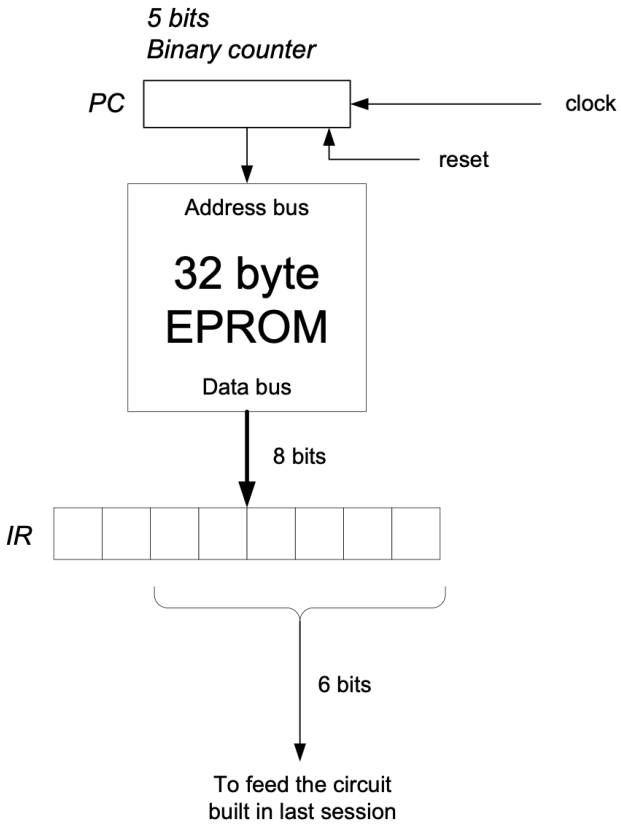
**دستیار آموزشی:**
**عطیه غیبی فطرت**

**اعضای تیم:**
**امیر اردلان دهقان پور ۴۰۱۱۰۵۹۰۱**
**رادین شاه دائی ۴۰۱۱۰۶۰۹۶**
**باربد شهر آبادی ۴۰۱۱۰۶۱۲۵**

## فهرست مطالب:

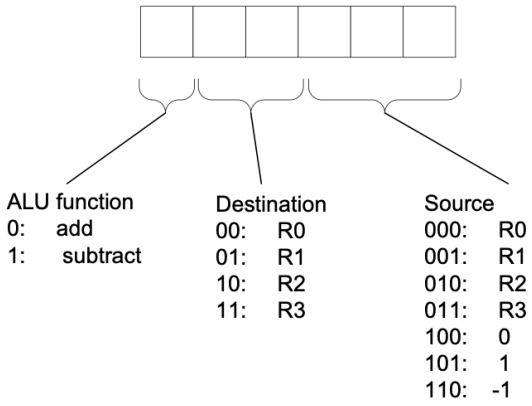
## شرح آزمایش

در این آزمایش، در ادامه آزمایش ششم که به طراحی واحد محاسبات و همچنین بانک ثبات یک کامپیوتر ساده پرداختیم، می‌خواهیم یک واحد حافظه و ثبات Program Counter اضافه کنیم، به طوری که بتوانیم از حافظه دستورات، دستور بخوانیم و طبق مدل فون‌نیومن، اعداد سری فیبوناچی را تولید کنیم. به منظور پیاده‌سازی حافظه دستور در ابزار شبیه‌سازی پروتئوس، از EPROM IC که 27C64 است استفاده می‌کنیم. این حافظه EPROM، قابلیت خواندن از یک initialize memory file را دارد که می‌توانیم قبل از اجرای برنامه، به آن دستورات مورد نیاز برای تولید اعداد سری فیبوناچی را بدهیم. شمای کلی بخشی که به مدار اضافه خواهیم کرد را در شکل زیر مشاهده می‌کنید:




شکل ۱

همانطور که در آزمایش قبل مشخص شده بود، دستورات این کامپیوتر ساده هر کدام ۶ بیتی هستند که هر بیت دستور به صورت زیر عمل می‌کند. دقت کنید که در این کامپیوتر ساده، مقدار مشخص شده در Source همواره با R0 جمع یا تفریق می‌شود.




شکل ۲

همانطور که در دستور کار آزمایش آمده است، نیاز است برای پیاده سازی و نمایش ۱۰ جمله اول فیبوناچی از سری دستورات زیر استفاده کنیم. در ادامه نشان می‌دهیم که کد ماشین ۶ بیتی هر کدام از این دستورات به چه صورت است و جدول شکل ۳ را پر می‌کنیم.

| Address | Code | Instruction | Comment | |
|---|---|---|---|---|
| 00000 | | Sub R0،R0 | R0← 0 | جمله اول در R0 |
| | | Add R1، 1 | R1← 1 | جمله دوم در R1 |
| | | Add R0،R1 | R0←1 | جمله سوم در R0 |
| | | Add R1،R0 | R1←2 | جمله چهارم در R1 |
| | | Add R0،R1 | R0←3 | جمله پنجم در R0 |
| | | Add R1،R0 | R1←5 | جمله ششم در R1 |
| | | Add R0،R1 | R0←8 | جمله هفتم در R0 |
| | | Add R1،R0 | R1←13 | جمله هشتم در R1 |
| | | Add R0،R1 | R0←21 | جمله نهم در R0 |
| | | Add R1،R0 | R1←34 | جمله دهم در R1 |

شکل ۳

| Address | Machine Code | Instruction |
|---|---|---|
| 00000 | 100000 | R0 <- 0 |
| 00001 | 001101 | R1 <- 1 |
| 00010 | 000001 | R0 <- R0+R1 |
| 00011 | 001001 | R1 <- R0+R1 |
| 00100 | 000001 | R0 <- R0+R1 |
| 00101 | 001001 | R1 <- R0+R1 |
| 00110 | 000001 | R0 <- R0+R1 |
| 00111 | 001001 | R1 <- R0+R1 |
| 01000 | 000001 | R0 <- R0+R1 |
| 01001 | 001001 | R1 <- R0+R1 |

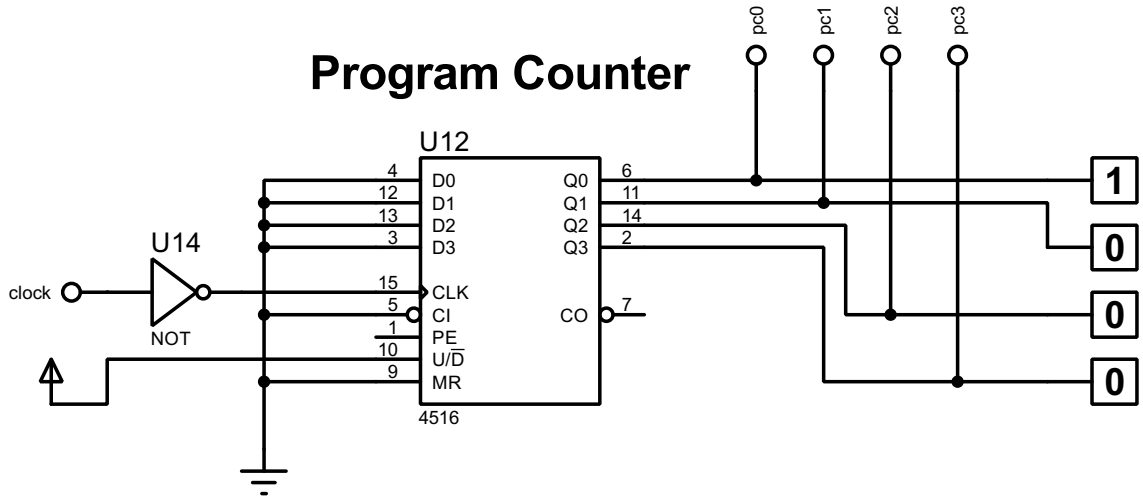
شکل ۴

بدین ترتیب، در ثبات R0 جملات با اندیس فرد و در ثبات R1، جملات با اندیس زوج به ترتیب ذخیره خواهند شد.

## شرح مدار شبیه‌ساز پروتئوس

در ادامه، به شرح مدار پروتئوس می‌پردازیم. دقت کنید که این مدار، عملا همان مدار آزمایش ششم می‌باشد با این تفاوت که بخش‌های EPROM و همچنین Program Counter بدان اضافه شده است. به همین منظور، تنها بخش‌های تغییر یافته از آزمایش ششم در این گزارش آورده شده‌است.

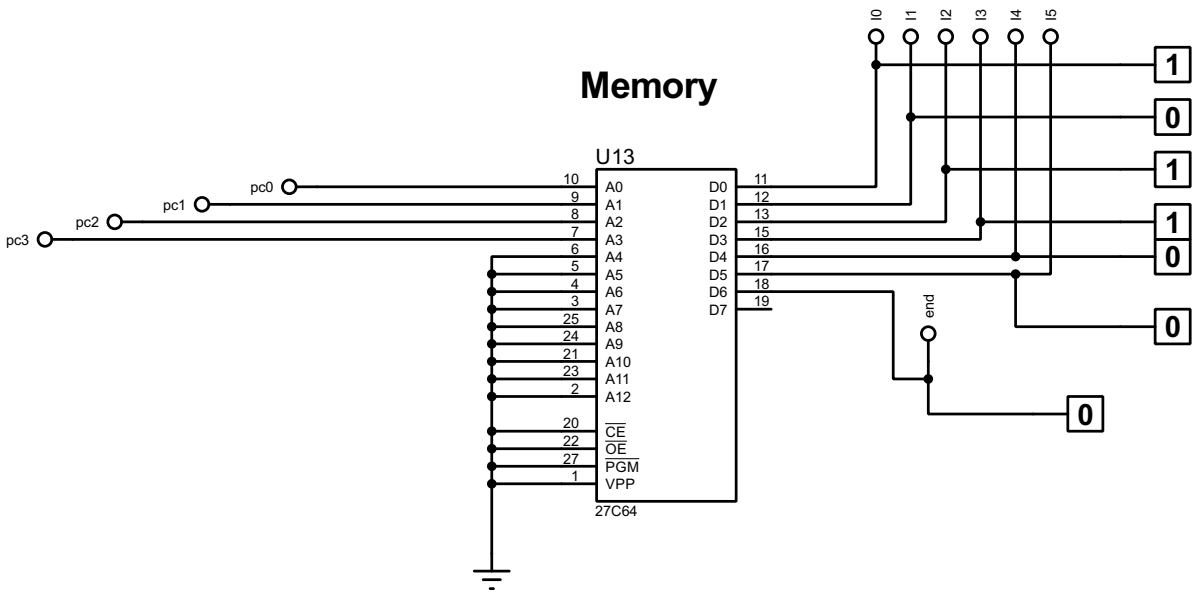
### ۱) شمارنده دستور یا Program Counter:




شکل ۵

این بخش شامل تنها یک IC شمارنده ۴ بیتی است که دارای شماره 4516 می‌باشد. ورودی Clock این شمارنده، از معکوس یا Not خود Clock ورودی برنامه می‌آید. همچنین مد U/D این شمارنده نیز به صورت UP می‌باشد.

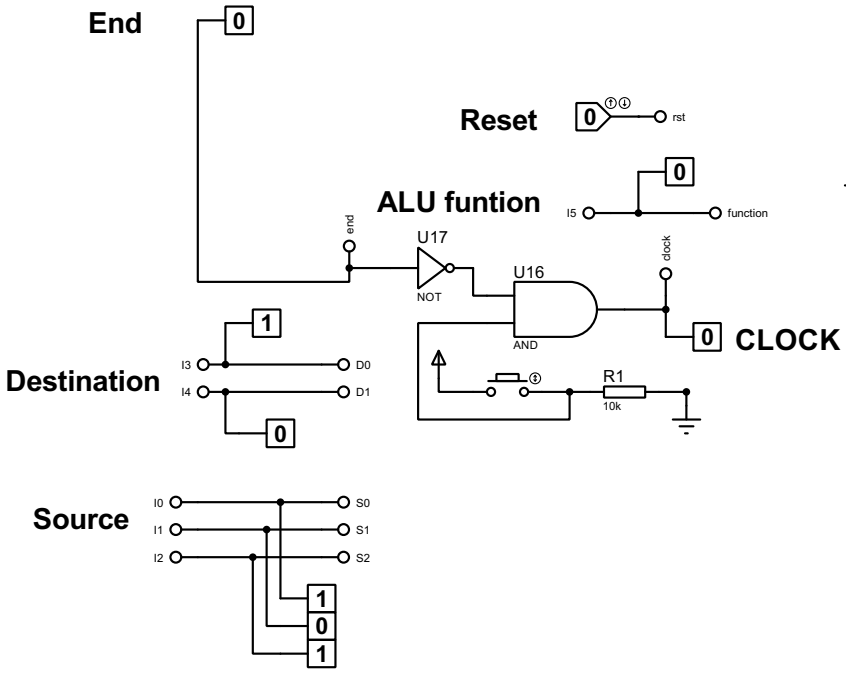
### ۲) حافظه دستور یا EPROM:




شکل ۶

این بخش نیز شامل 27C64 IC می‌باشد که EPROM استاندارد موجود در پروتئوس می‌باشد. ورودی آدرس این حافظه، با توجه به اینکه دستورات در این حافظه ریخته شده‌اند، همان Program Counter است. همچنین خروجی data این حافظه نیز، instruction های کامپیوتر محسوب می‌شوند که به ترتیب خوانده می‌شوند. دقت کنید که وقتی بیت هفتم data مقدار 1 را به خودش بگیرد، سیگنال end برنامه فعال می‌شود و دیگر از حافظه نمیخوانیم. دقت کنید که دستورات 6 بیتی هستند و از بیت هفتم data می‌توانیم به عنوان راهی برای اتمام برنامه استفاده کنیم. دقت کنید که این IC شامل آدرس 13 بیتی است که از بیت چهارم به بعد، همگی به صفر وصل شده‌اند. در نتیجه همواره ۱۶ دستور اول می‌تواند خوانده شود.

## ۳) باقی بخش‌های افزوده شده:

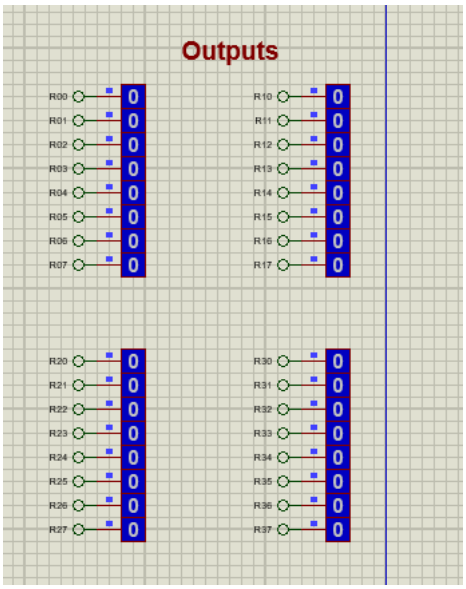



شکل ۷

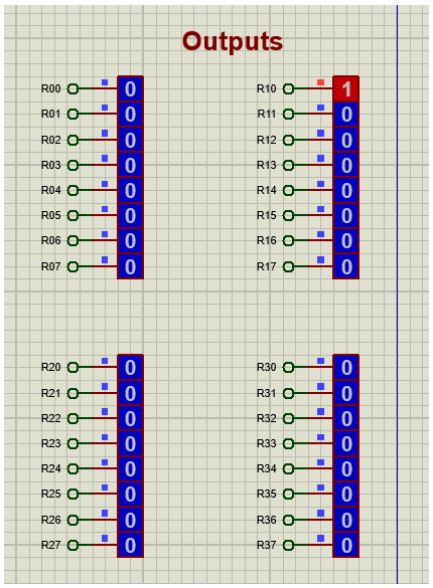
سیگنال Clock از AND کردن end~ و همچنین Push Botton ورودی می‌باشد. بدین ترتیب اگر دستورات به اتمام رسیده بودند و سیگنال end فعال شده بود، دیگه Clock جلو نمی‌رود و PC جلو نمی‌رود. سیگنال ALU Function به سادگی از بیت ششم دستور آورده می‌شود. همچنین مقادیر D[1..0] و S[2..0] که در واقع destination و source دستورات این کامپیوتر ساده می‌باشند به ترتیب از I[4..3] و I[2..0] خوانده می‌شوند.

## تست فیبوناچی

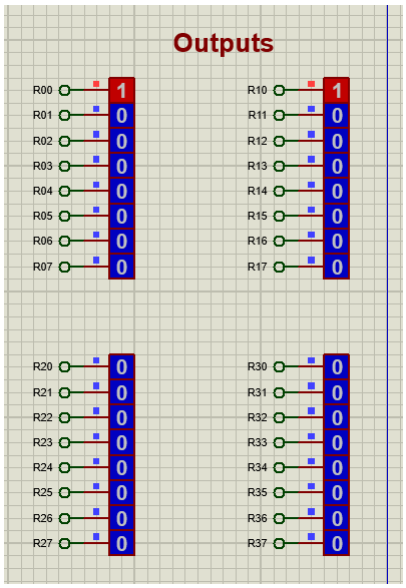
در ادامه، به تست حافظه و مدار جدید می‌پردازیم. به منظور این کار، ابتدا از بخش edit properties در EPROM استفاده شده، فایل fib.hex را بعنوان ورودی memory initialization file می‌دهیم. حال شبیه‌ساز را اجرا می‌کنیم و خروجی‌های زیر را به ترتیب می‌بینیم.




شکل ۸

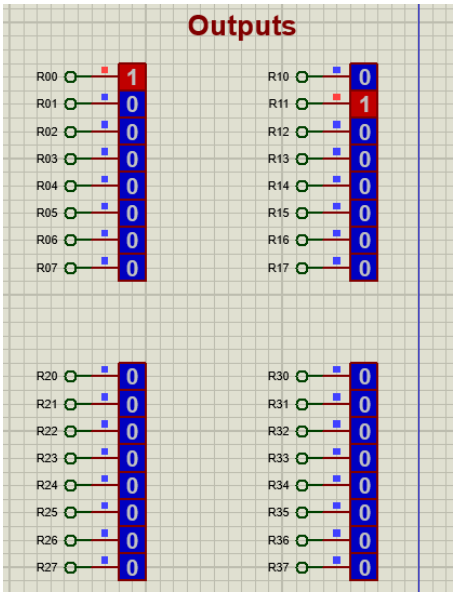



شکل ۹

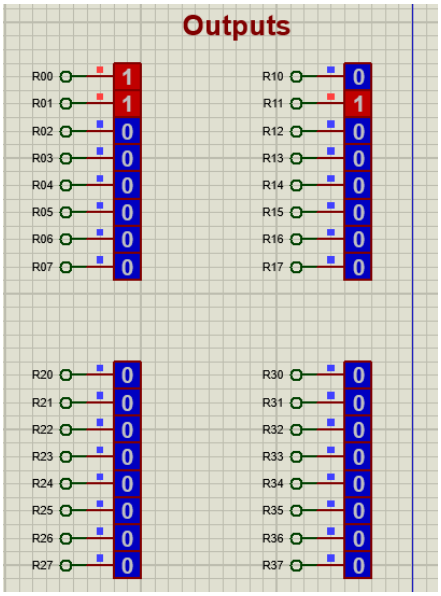



شکل ۱۰

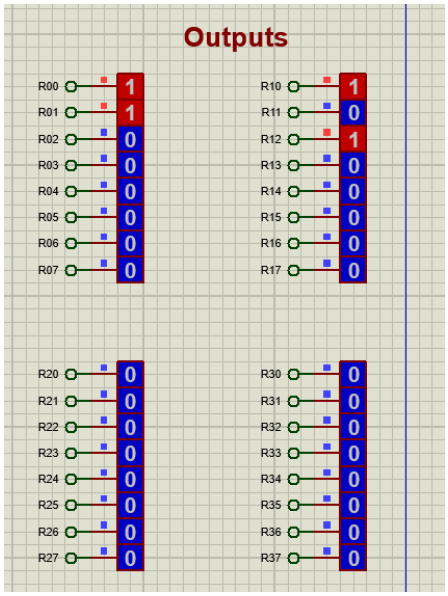
در شکل‌های ۸، ۹ و ۱۰، اجرای ۳ دستور اول برنامه را مشاهده می‌کنید. تا این‌جای برنامه، جملات دوم و سوم فیبوناچی در ثبات‌های R0 و R1 قرار دارند. در شکل‌های بعدی این بخش از گزارشکار نیز به همین ترتیب مقدار هر یک از ثبات‌های بعد از اجرای دستورات را مشاهده خواهید کرد.




شکل ۱۱

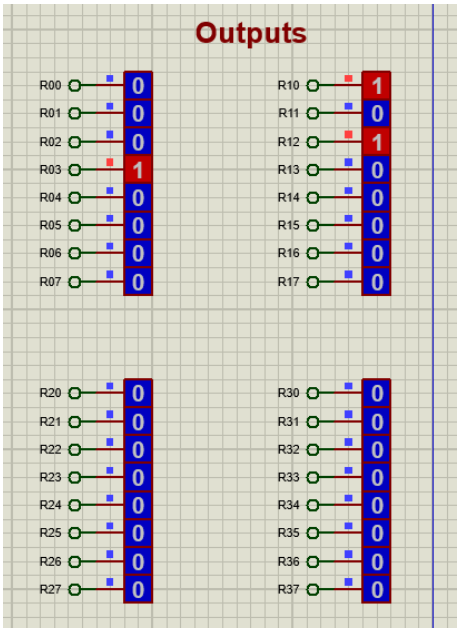



شکل ۱۲

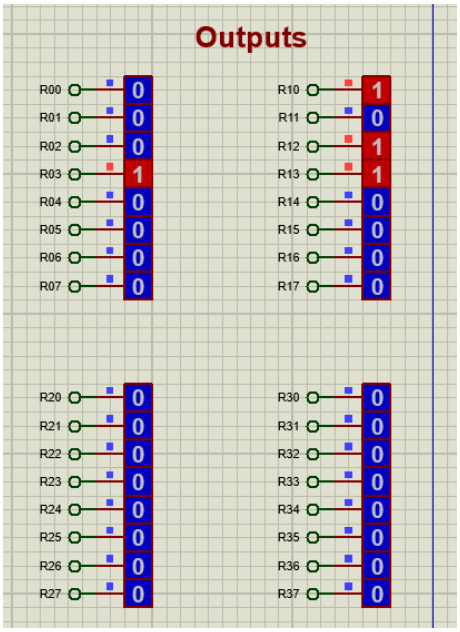



شکل ۱۳

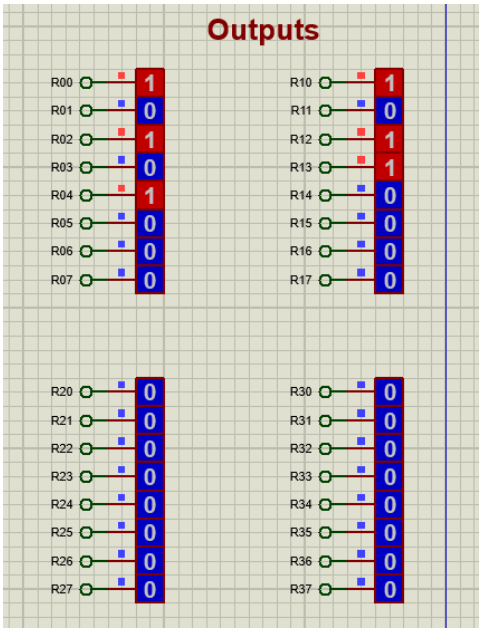
در ادامه نیز، دستورات بعدی برنامه اجرا شدند. در انتهای شکل ۱۳، مقدار ۳ در ثبات R0 و مقدار ۵ در ثبات R1 قرار دارند.




شکل ۱۴

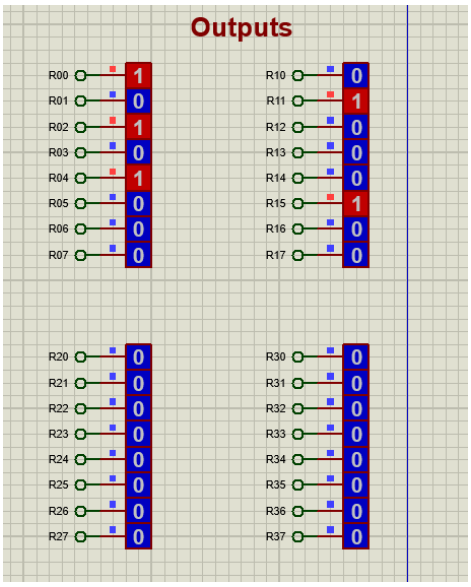



شکل ۱۵

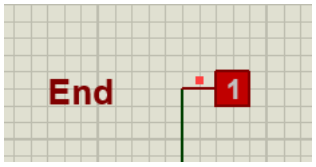



شکل ۱۶

در شکل‌های ۱۴ تا ۱۶، دستورات هفتم، هشتم و نهم برنامه به ترتیب اجرا می‌شوند که در نهایت در شکل ۱۶ مشاده می‌کنیم که در ثبات R0 مقدار 21 و در ثبات R1 مقدار 13 قرار دارد.




شکل ۱۷




شکل ۱۸

در نهایت مشاهده می‌کنید که دستور آخر برنامه اجرا شده‌است. همزمان با اجرای این دستور نیز سیگنال END فعال می‌شود و PC دیگر تغییر نمی‌کند. دقت کنید که در نهایت در ثبات R1 مقدار 34 قرار داده می‌شود که جمله دهم دنباله فیبوناچی می‌باشد. بدین ترتیب این مدار و برنامه فیبوناچی به درستی کار می‌کنند.