دانشگاه صنعتی شریف
دانشکده مهندسی کامپیوتر
گزارش کار هفتم درس آزمایشگاه معماری

عنوان:

کنترل توسط برنامه ذخیره شده در حافظه

Control by program stored in EPROM

نگارش

کیان قاسمی ۴۰۱۱۰۲۲۶۴
آیین پوست فروشان ۴۰۱۱۰۵۷۴۲
دیبا هادی اسفنگره ۴۰۱۱۱۰۲۴۵

استاد

دکتر حمید سربازی آزاد

دستیار آموزشی

مهندس عطیه غیبی فطرت

مرداد ۱۴۰۳

فهرست مطالب

فهرست تصاویر

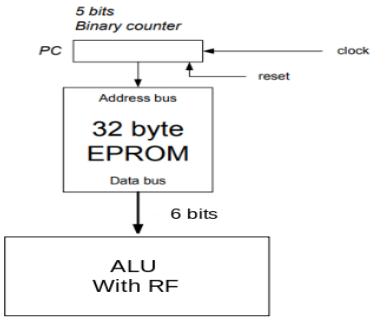
۱ مقدمه

در این آزمایش قرار است بر روی واحد محاسبه گر (*ALU*) طراحی شده در آزمایش قبل واحدی طراحی کنیم تا دستورات ورودی آن از یک حافظه قابل برنامه ریزی گرفته شود. همچنین شمارنده ای نیز به عنوان *program counter*(*PC*) نیز خواهیم داشت که دستورات پشت سر هم اجرا شوند و برای توقف آن نیز بسته به تعداد دستور های برنامه شرط اجرا شدن تمام دستورات چک می شود و خروجی برنامه نیز داخل سه ثبات مدار قرار میگیرند.
در نهایت بلوک دیاگرام نهایی برای شبیه سازی مدار و قطعات استفاده شده به صورت زیر می باشد:

۱.۱ قطعات لازم

- دیکودر ۴ به ۱۶ (قطعه 74154)
- گیت های منطقی *OR*، *AND*، *XOR* ۲ بیتی و گیت منطقی *NOT*
- مالتی‌پلکسر ۲ به ۱ چهاربیتی (قطعه 74157)
- مالتی‌پلکسر هشت‌بیتی (قطعه 74181)
- گیت های منطقی *NAND* با تعداد پایه های مختلف
- رجیستر هشت‌بیتی (قطعه 74198)
- جمع کننده چهاربیتی (قطعه 7483)

۲.۱ بلوک دیاگرام نهایی



شکل ۱: بلاک دیاگرام کلی

۲ شبیه سازی مدار

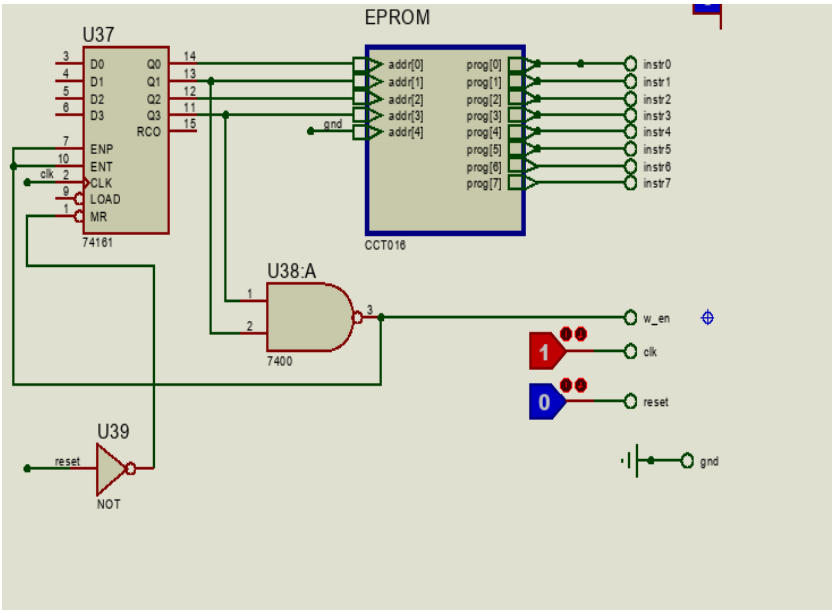
برای طراحی این مدار ابتدا واحد حافظه و کنترلی را ساختیم و سپس خروجی آن را به مدار آزمایش قبل متصل کردیم.

۱.۲ طراحی کلی واحد حافظه و کنترلی دستورات

در این قسمت یه واحد *EPROM* در نظر گرفتیم که ورودی آن چهاربیتی است که شماره کلاک فعلی را نشان می دهد. دقت کنید که *write – enable* واحد ثبات هایی که در آزمایش قبل طراحی کردیم با بیت w_en مشخص میشود و همچنین این بیت به *ENT* و *ENP* شمارنده نیز متصل است و درواقع زمانی خواندن دستورات و ذخیره نتیجه ادامه پیدا می کند که این بیت یک باشد و با توجه به شکل مدار نیز این بیت زمانی ۰ می شود که شماره کلاک و درواقع خروجی شمارنده برابر *instruction count* باشد و درواقع داریم:

$$W_{en} = (PC == IC)$$

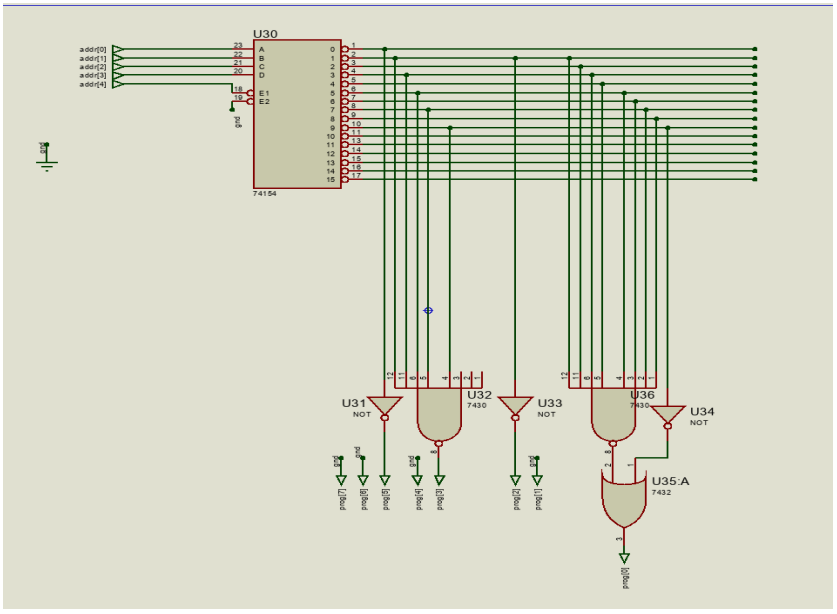
شکل کلی این قسمت نیز به صورت زیر است: حال شماره کلاک در حال اجرا و یا درواقع



شکل ۲: ماژول حافظه و *PC*

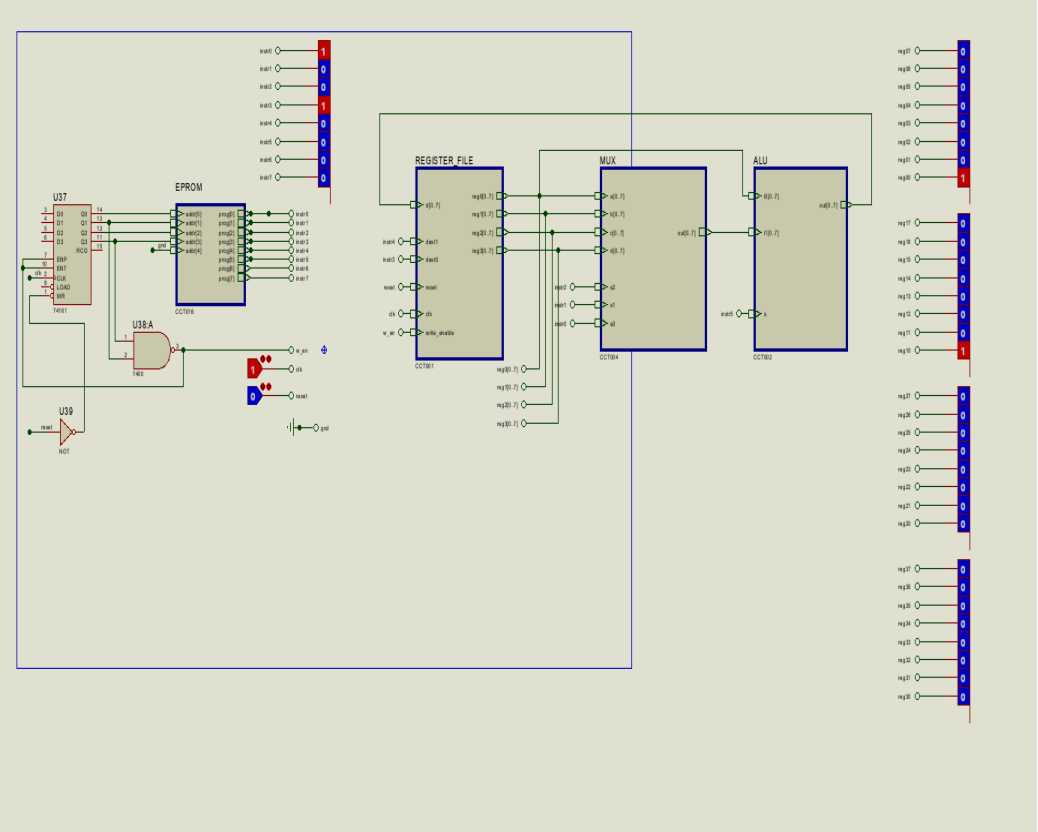
شماره دستور فعلی به صورت باینری به واحد *EPROM* ورودی داده می شود. در این ماژول یک دیکورد داریم که شماره دستور فعلی را مشخص می کند و خروجی 8 بیتی آن همان دستور فعلی است که چون در این برنامه به ۶ بیت نیاز داریم دو بیت دیگر آن بلااستفاده است.

دقت کنید که اگر دستور nام درحال اجرا باشد بیت nام دیکودور ۱ است. حال چون دیکودر استفاده شده *active low* است برای هر بیت دستور یک پایه *NAND* در نظر گرفتیم و شماره دستوراتی که در آن ها این بیت دستور یک است را به پایه های *NAND* وصل کردیم و به این صورت در شماره دستوراتی که این بیت ۱ است، این بیت ۱ خروجی داده می شود و بالعکس. همچنین بیت های *enable* دیکورد نیز چون *active low* است به زمین و بیت ۵ام ورودی وصل شده که همیشه ۰ است (دیکورد ۴ بیتی است و حافظه ۳۲ بیت دارد اما چون به بقیه دستورات نیازی نداریم صفر در نظر گرفته شده است.)



شکل ۳: مدار داخلی ماژول *EPROM*

نمای کلی مدار نیز به صورت زیر است:



شکل ۴: شکل کلی مدار

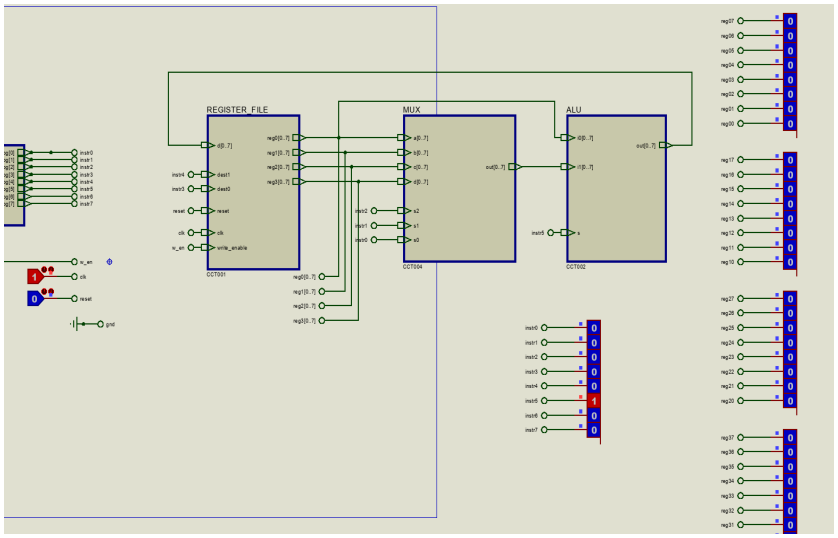
۲.۲ آزمایش درستی مدارد و بررسی درستی عملکرد آن

برای بررسی درستی مدار برنامه ای نوشتیم که تا جمله دهم دنباله فیبوناچی را حساب کند. دستورات این مدار با توجه به آزمایش قبل و معماری دستورات به صورت زیر است:

Address	Code	Instruction
00000	100000	$R0 \leftarrow R0 - R0$
00001	001101	$R1 \leftarrow 1$
00010	000001	$R0 \leftarrow R0 + R1$
00011	001001	$R1 \leftarrow R0 + R1$
00100	000001	$R0 \leftarrow R0 + R1$
00101	001001	$R1 \leftarrow R0 + R1$
00110	001001	$R1 \leftarrow R0 + R1$
00111	000001	$R0 \leftarrow R0 + R1$
01000	001001	$R1 \leftarrow R0 + R1$
01001	000001	$R0 \leftarrow R0 + R1$
01010	001001	$R1 \leftarrow R0 + R1$

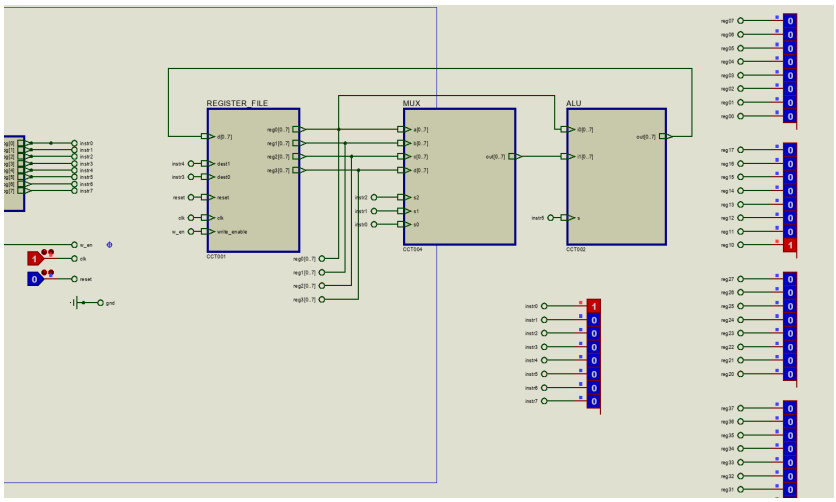
این دنباله دستورات را روی $EPROM$ ساختیم و دنباله خروجی ها در هر کلاک و بعد از انجام هر دستور به صورت زیر می باشد(ثبات ها به ترتیب از بالا به پایین ثبات های $R0$ تا $R3$ هستند):

۱) در ابتدا ثبات $R0$ را از خودش کم میکنیم، خروجی بعد از این دستور به صورت زیر است:



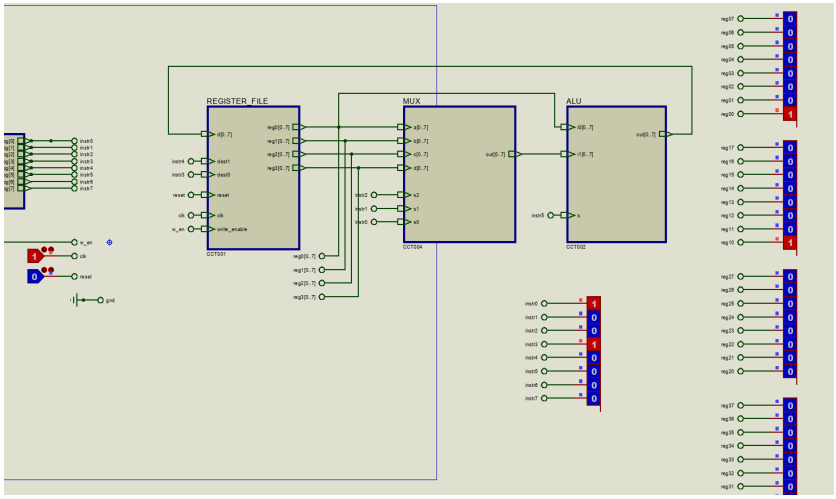
شکل ۵: خروجی بعد از دستور اول

۲) در دستور دوم عدد ۱ داخل ثبات $R1$ ریخته شده است. خروجی بعد از این دستور به صورت زیر است:



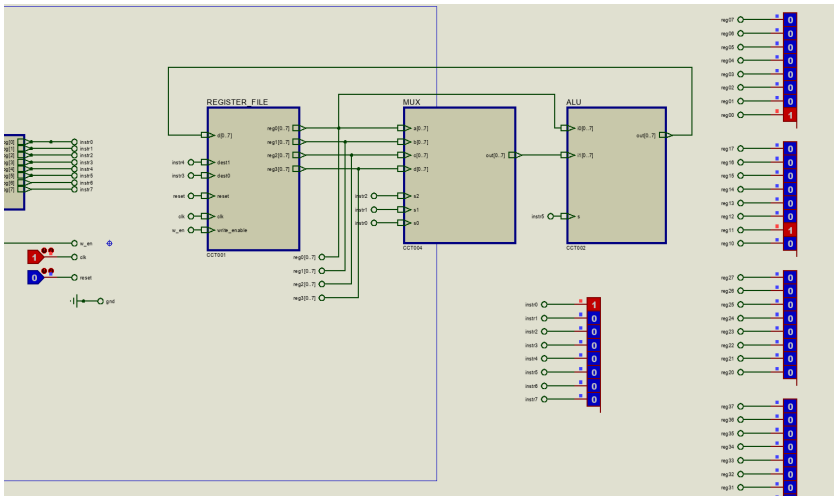
شکل ۶: خروجی بعد از دستور دوم

۳) در دستور سوم جمع دو ثبات داخل ثبات $R0$ ریخته شده است. خروجی بعد از این دستور به صورت زیر است:



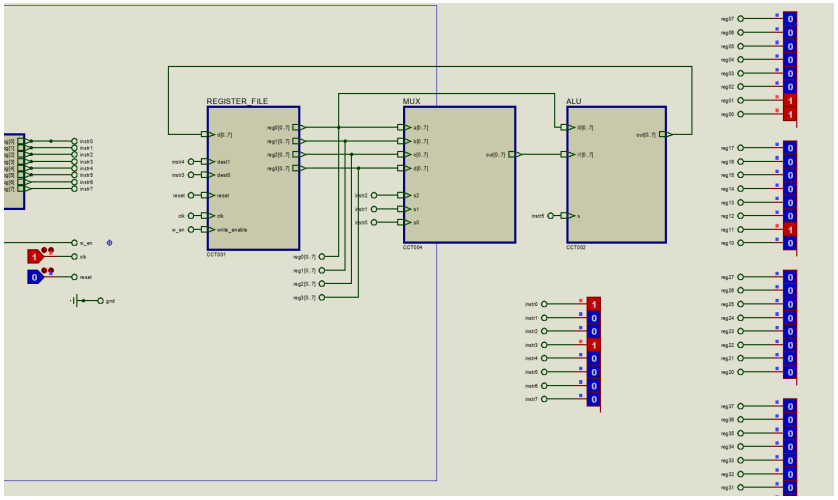
شکل ۷: خروجی بعد از دستور سوم

۴) از این دستور به بعد دنباله فیبوناچی داخل ثبات ها ریخته می شود، بعد از این دستور ثبات $R1$ برابر ۲ می شود. خروجی بعد از این دستور به صورت زیر است:



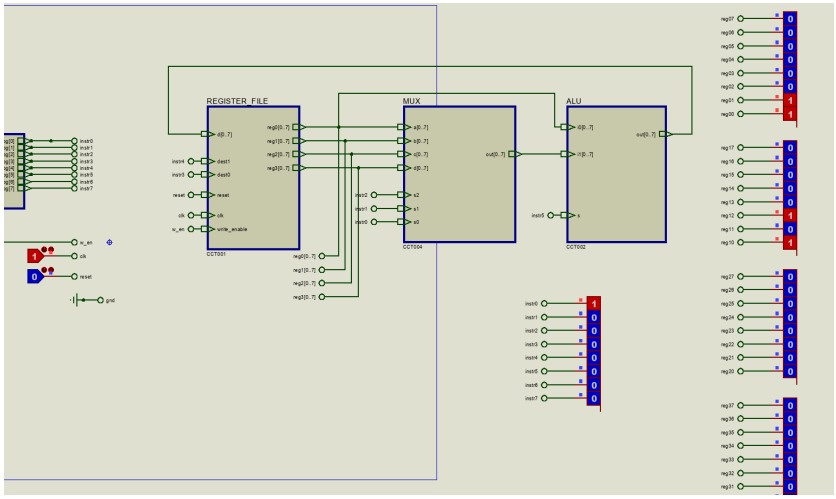
شکل ۸: خروجی بعد از دستور چهارم

۵) بعد از دستور پنجم ثبات $R0$ برابر ۳ می شود. خروجی بعد از این دستور به صورت زیر است:



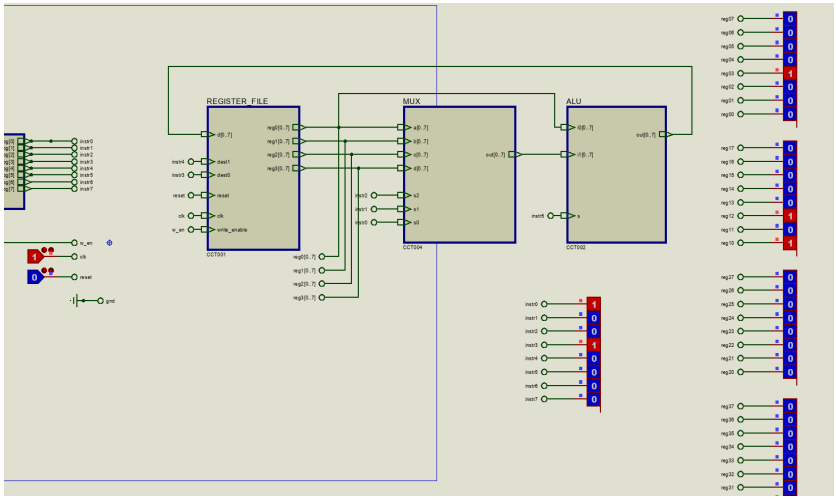
شکل ۹: خروجی بعد از دستور پنجم

۶) بعد از دستور ششم ثبات $R1$ برابر ۵ می شود. خروجی بعد از این دستور به صورت زیر است:



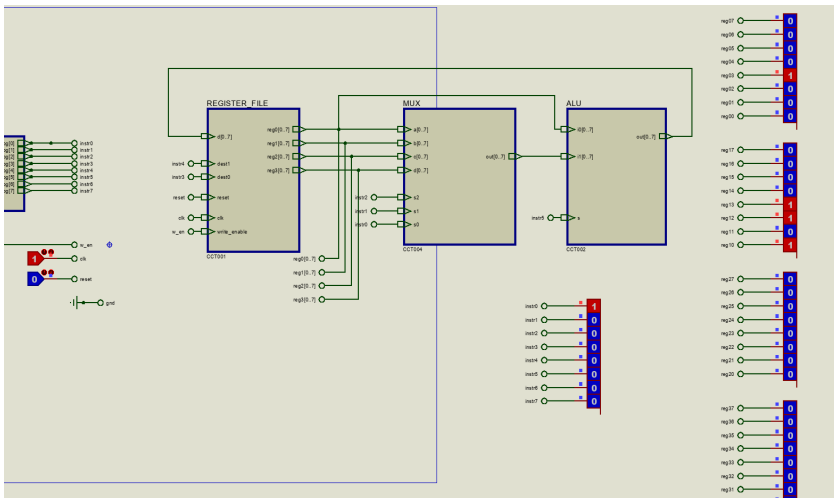
شکل ۱۰: خروجی بعد از دستور ششم

۷) بعد از دستور پنجم ثبات $R0$ برابر ۸ می شود. خروجی بعد از این دستور به صورت زیر است:



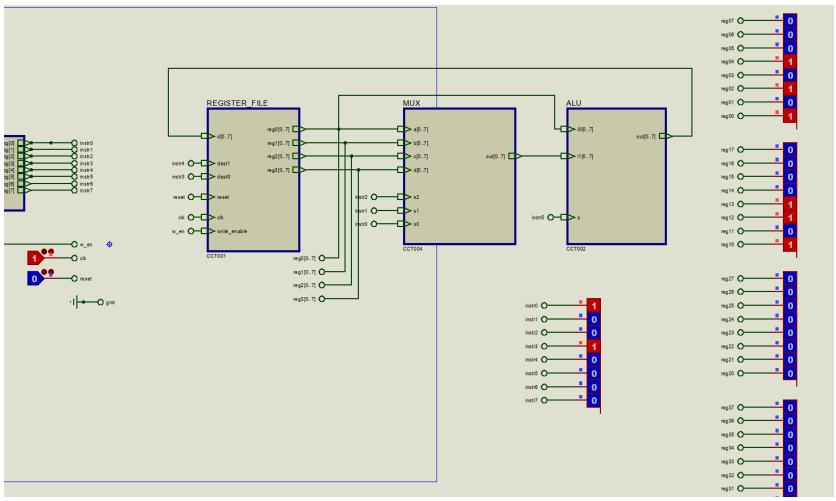
شکل ۱۱: خروجی بعد از دستور هفتم

۸) بعد از دستور هشتم ثبات $R1$ برابر ۱۳ می شود. خروجی بعد از این دستور به صورت زیر است:



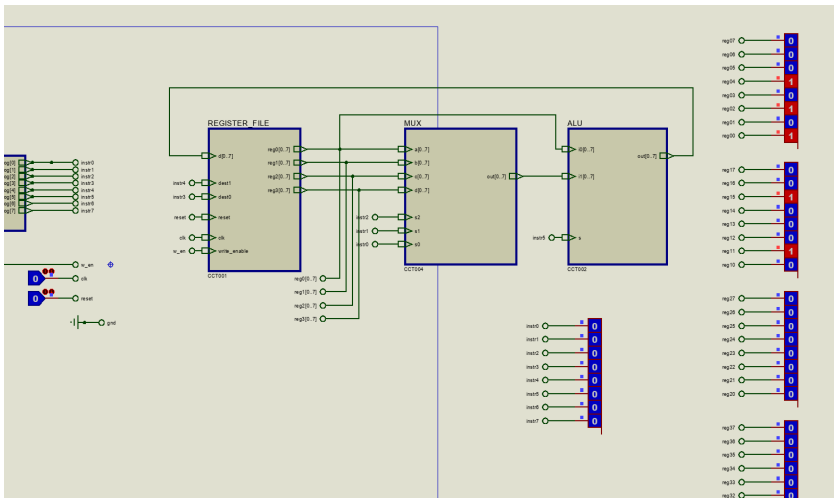
شکل ۱۲: خروجی بعد از دستور هشتم

۹) بعد از دستور پنجم ثبات $R0$ برابر ۲۱ می شود. خروجی بعد از این دستور به صورت زیر است:



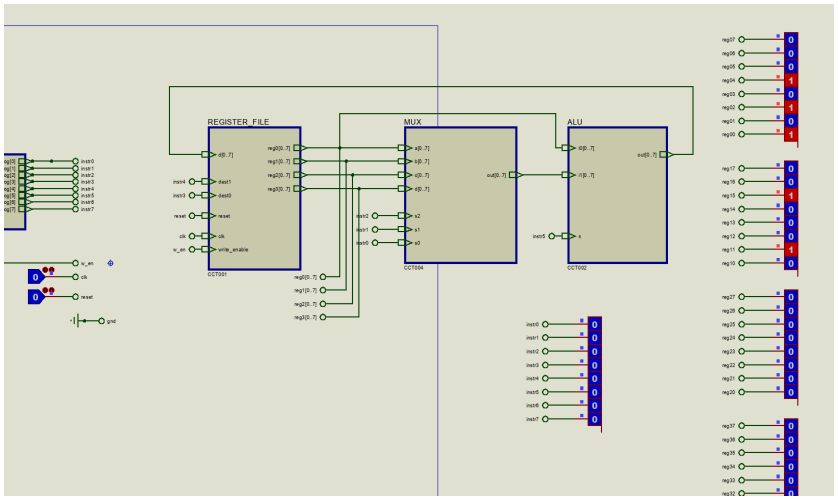
شکل ۱۳: خروجی بعد از دستور نهم

۱۰) بعد از دستور پنجم ثبات $R1$ برابر ۳۴ می شود. خروجی بعد از این دستور به صورت زیر است:



شکل ۱۴: خروجی بعد از دستور دهم

در نهایت نیز بیت های ENTو ENP صفر می شوند و خروجی با هر تعداد کلاک تغییر نمی کند و دستورات به انتها رسیده است:



شکل ۱۵: شرط پایان دستورات

۳ چالش ها

با توجه به این که این آزمایش پیاده سازی فیزیکی نداشت چالشی در این مدار نداشتیم.

۴ نتیجه و بحث

در این آزمایش موفق شدیم قابلیت طراحی برنامه به روی آزمایش قبلی اضافه کنیم و دستورات را از واحد *EPROM* بخوانیم. همچنین دنباله فیبوناچی را با این مدار تا جمله دهم محاسبه کردیم.