***بنام خدا***

**شبیه­سازی کامپیوتر مبنا**

بر ­اساس کتاب­معماری­کامپیوتر، نوشته­ی موریس­مانو

**استاد راهنما:**

**دکتر گلی**

**طراحان :**

**معین سلیمی سرتختی**

**محمدرضا رهنما فلاورجانی**

**محمدجواد ملکی کهکی**

****

**بهار 1394**

**فهرست مطالب**

مقدمه -------------------------------------------------------------------------------------------------1

گذرگاه مشترک برای تبادل داده (باس) ----------------------------------------------------------------------2

واحد محاسباتی و منطقی ALU به همراه واحد شیفت دهنده -----------------------------------------------------3

واحد کنترل --------------------------------------------------------------------------------------------6

حافظه -------------------------------------------------------------------------------------------------9

ثبات­ها -----------------------------------------------------------------------------------------------11

واحد نمایش ------------------------------------------------------------------------------------------12

معرفی نرم افزار 010 Editor ----------------------------------------------------------------------------13

نحوه­ی کار با کامپیوتر ----------------------------------------------------------------------------------14

**مقدمه**

هدف از این پروژه ، شبیه­سازی کامپیوتری است که بر اساس کامپیوتر مبنای فصل پنجم کتاب "معماری کامپیوتر" (نوشته­ی موریس مانو) عمل کند.این پروژه با استفاده از نرم افزار **پروتئوس** **(Proteus)** طراحی شده است

بخش­های اصلی تشکیل دهنده­ی کامپیوتر عبارت اند از:

1. گذرگاه مشترک برای تبادل داده (باس)
2. واحد محاسباتی و منطقی ALU به همراه واحد شیفت دهنده
3. واحد کنترل
4. حافظه
5. ثبات­ها
6. واحد نمایش

این کامپیوتر قادر به انجام همه­ی اعمالی است که کامپیوتر فصل پنجم کتاب مذکور پشتیبانی می­کند

[Moeinsalimi.sartakhti@gmail.com](mailto:Moeinsalimi.sartakhti@gmail.com)

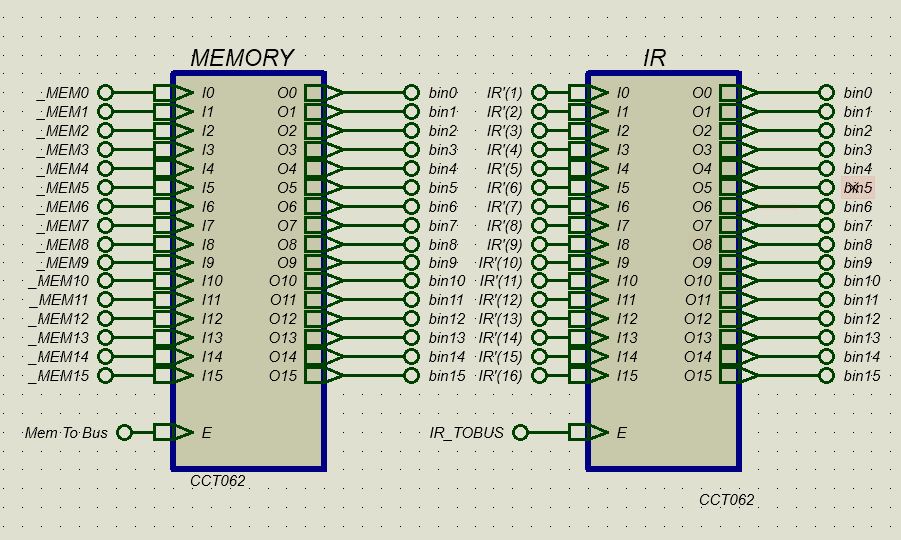
[Mohammadreza.rahnamaa@gmail.com](mailto:Mohammadreza.rahnamaa@gmail.com)

[Mjmaleki94@gmail.com](mailto:Mjmaleki94@gmail.com)

**گذرگاه مشترک برای تبادل داده (باس)**

در واقع در این پروژه باس یک چیز فیزیکی نیست بلکه یک سری ترمینال هایی است که به ورودی همه­ی ثبات­ها وصل شده­اند و از آن سو نیز باید کنترل کنیم که چه موقع ثبات ها و حافظه می­توانند اطلاعات خود را روی این ترمینال­ها بگذارند. این کنترل کردن ، بوسیله­ی تعدادی بافر و یک سری تابع های کنترلی (که در واحد کنترل ایجاد شده­اند) انجام شده است.

بعنوان نمونه دوتا از بافر­ها که کنترل می­کنند چه موقع باید اطلاعات از حافظه و IR به باس منتقل شود را در شکل زیر می بینید. لازم به ذکر است که تابع­های کنترلی و ورودی فعالساز این بافر­ها در واحد کنترل تولید می­شوند.



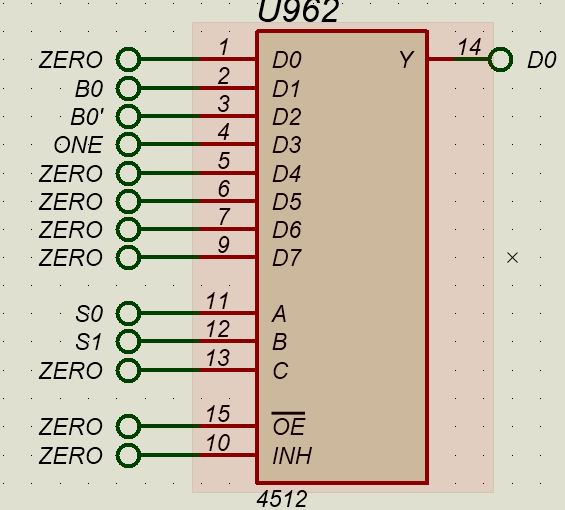
برای بقیه­ی ثبات­ها ( البته بجز ثبات­های Input,Output) نیز به همین ترتیب یک بافر کترلی وجود دارد

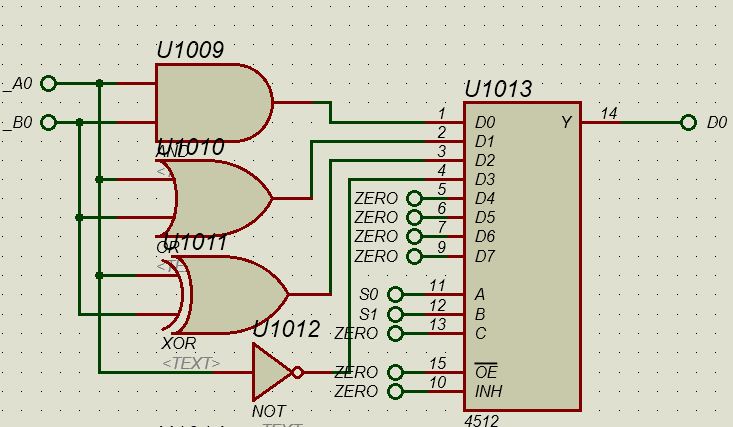
**واحد محاسباتی و منطقی ALU به همراه واحد شیفت دهنده**

سعی کردیم کل کامپیوتر بر اساس فصل پنجم طراحی شود اما ALU یک استثناء است. ALU این کامپیوتر بر اساس فصل چهارم طراحی شده . تنها تفاوت میان ALU فصل چهارم و پنجم در تعداد دستوراتی است که می­توانند انجام دهند بگونه­ای که اعمالی که ALU فصل چهارم قادر به انجامشان است بیشتر از اعمالی است که ALU فصل پنجم قادر به انجام دادن است. بنابراین ALU این کامپیوتر بر اساس فصل چهارم طراحی شد تا بتوان دستورات بیشتری را داشت.

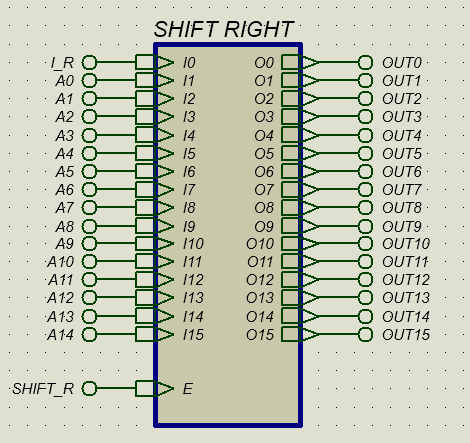
توجه شود که کامپیوتر فعلی فقط می­تواند دستورات فصل پنجم را پیاده سازی کند . برای مثال گرچه در ALU فصل چهارم اعمالی مثل OR کردن یا XOR کردن وجود دارد ، اما در واحد کنترل این دستورات پشتیبانی نشده و اصلا پیاده نشده­اند. درواقع اگر در واحد کنترل دستورات OR و XOR و مشابه آن­ها را اضافه کنیم در آن صورت می توان در این کامپیوتر اینگونه دستورات را نیز شبیه سازی و پیاده کرد

در شکل زیر یک طبقه از واحد محاسباتی ALU را مشاهده می­کنید که با خطوط انتخاب A,B میتوان 4 حالت مختلف ایجاد کرد



در شکل زیر یک طبقه از واحد منطق ALU را مشاهده می­کنید

**بخش شیفت دهنده**

برای انجام عمل شیفت از بافر استفاده کرده­ایم . در شکل زیر بافری را که برای انجام عمل شیفت به راست طراحی شده مشاهده می­کنید.لازم به ذکر است که سیگنال فعالساز این بافر از واحد کنترل صادر می­شود.

**ورودی­های ALU**

می­دانیم که ورودی­های ALU باید از بین AC,DR,Input انتخاب شوند در حالیکه فقط دو ورودی برای ALU داریم. برای این کار ما یکی از ورودی های ALU را بصورت دائمی به AC وصل کرده­ایم و برای ورودی دومی ALU از سه بافر کنترلی استفاده کردیم. این بافر­ها بررسی می­کنند که آیا ورودی دوم ALU ثبات DR باید باشد یا Input باشد یا مانند ورودی اولی ALU ، بازهم خود AC باشد؟

توجه کنید که هرگاه یکی از این بافرها فعال شود محتوای ثبات مربوطه به ورودی دومی ALU منتقل می شود.

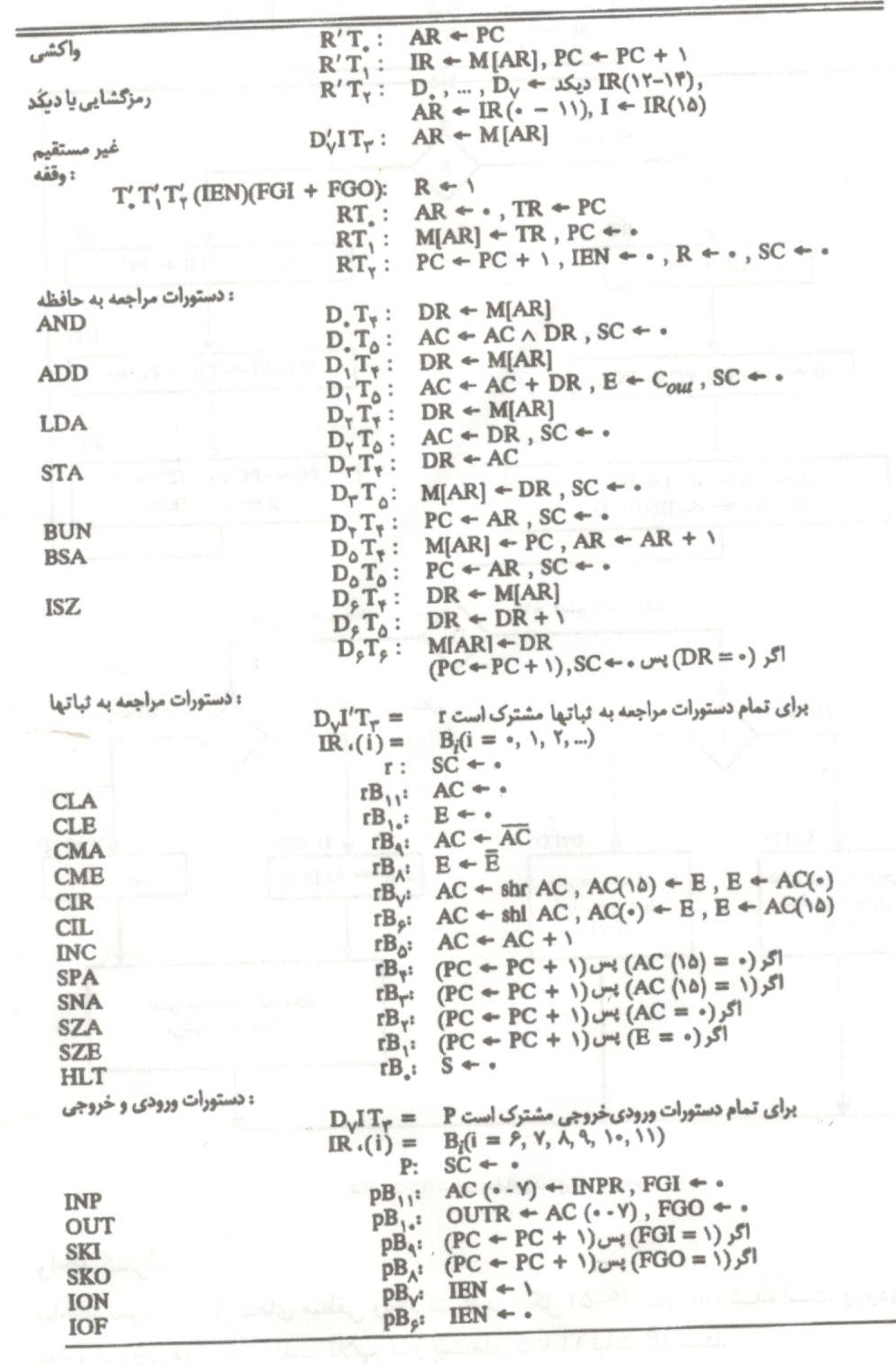
\*خطوط فعالساز بافرها ، از واحد کنترلی گرفته شدند.

**خروجی­های ALU**

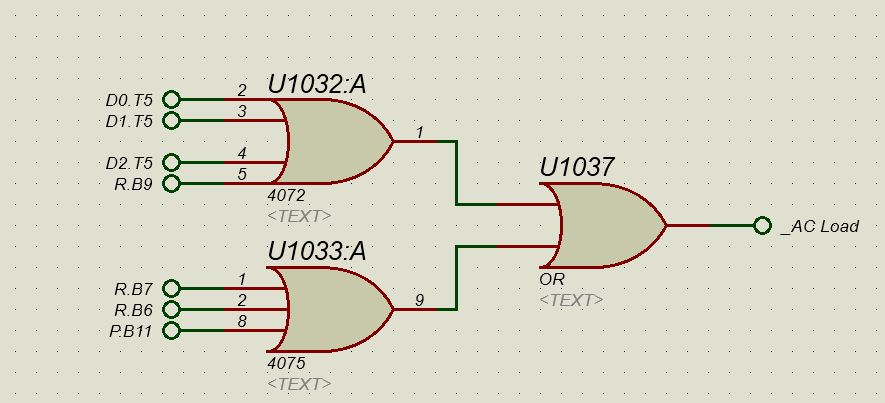
بطور کلی ما در واحد ALU چهار بافر مختلف داریم. دوتا برای شیفت به راست و چپ ، یکی برای انجام اعمال محاسباتی و یکی برای اعمال منطقی. اگر بخواهیم بطور واقعی به این 3 بخش (واحد محاسبات ، منطق ، شیفت) نگاه کنیم باید ذکر کنیم که همه­ی این 3 بخش در هر لحظه ، دارای جواب و خروجی هستند اما با استفاده از این بافر­ها کنترل می­کنیم که در هر لحظه کدام جواب بر روی خروجی ALU قرار بگیرد.

**واحد کنترل**

فلسفه­ی بوجود آمدن واحد کنترل این بود که می­باید همه­ی قسمت­ها اعم از واحد حافظه ، ثبات­ها و واحد ALU در زمان مقرر وظایف مربوط به خود را انجام دهند در واقع اجازه­ی انجام وظایف قسمت­های مختلف کامپیوتر ، توسط سیگنال­های کنترلی در واحد کنترل صادر می­شود.

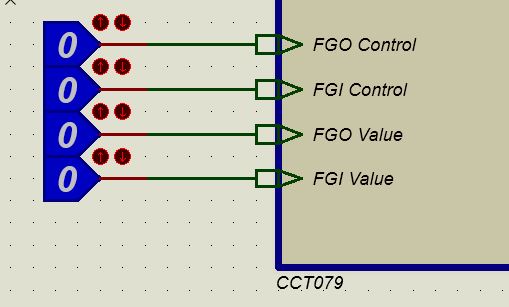
بعنوان مثال یکی از وظایف این واحد ، تولید سیگنال­های کنترلی مربوط به بار شدن ثباتی مثل AC است . برای تولید اینگونه سیگنال­ها ابتدا باید جدول تابع­های کنترلی و ریز عملیات کامپیوتر مبنا را تهیه کرد که این جدول در زیر آمده است:

اکنون با مراجعه به این جدول می­توان زمان­ها و پیش­شرط­هایی را که در آن­ ثبات AC آماده­ی بار شدن می­شود پیدا کرد و سپس همه­ی این حالت­ها را در بار شدن AC تاثیر داد که ما در واحد کنترل ، بگونه­ی زیر عمل کرده­ایم



برای بقیه­ی ثبات­ها نیز به همین ترتیب عمل کردیم .

شکل زیر مربوط به بخشی از ورودی های واحد کنترل است که توضیحات مربوط به آن در صفحه­ی بعد آمده است

****

بعضی اوقات باید کاربر قادر باشد تا مقدار دلخواهی از ثبات ورودی را به ثبات اکومولیتور منتقل کند. برای این کار باید بتواند بصورت دستی مقدار FGO را تغییر دهد. اگر FGO Control را برابر با یک کنیم ، در آن صورت کنترل FGO از حالت خودکار خارج شده و مقداردهی آن دستی می شود و اکنون می توان به FGO Value مقدار داد. منظور از FGO Value، خود مقدار FGO است که بصورت دستی قابل تعیین می باشد

برای ثبات Output نیز همین قضایا برقرار است

علاوه بر این ، در واحد کنترل :

* رمزگشایی و دیکد شدن دستورات انجام می گیرد
* زمانهای T0 تا T15 تولید می شود
* سیگنال­های ورودی فلیپ فلاپ­ها تولید می­شود
* سیگنال­های کنترلی جهت خواندن از حافظه و نوشتن در آن ایجاد می­شود
* و ....

**حافظه**

واحد حافظه از دو RAM ، دو ROM ، یک شمارنده و دو بافر کنترل کننده تشکیل شده است. حافظه­ی RAM و ROM هردو دارای 1024 کلمه ی 8 بیتی هستند که از کنار هم گذاشتن دوتای آنها ، می­توان حافظه­ای با 1024 کلمه­ به طول 16 بیت ایجاد کرد.

استفاده از ROM برای راحتی کار بود .به اینصورت که می­توان در ROM یک فایل باینری آپلود کرد و سپس اطلاعات ROM را به RAM منتقل کرد . با این کار دیگر لازم نیست که با هر بار روشن کردن کامپیوتر ،یک سری اطلاعات طولانی را بصورت دستی وارد RAM کرد

**دلیل وجود شمارنده :** برای انتقال اطلاعات از ROM به RAM ابتدا خطوط داده­ی ROM را به خطوط داده­ی RAM متصل کردیم سپس باید در هر مرحله ، یک آدرس یکسان به ROM و RAM داد تا اطلاعات ROM از طریق خطوط داده ، به همان آدرس در RAM منتقل شود. ما نیاز داشتیم تا بطور خودکار آدرس ROM و RAM از صفر تا 1023 افزایش یابد و همزمان که این آدرس­ها یکی یکی زیاد می­شود داده­های ROM به RAM منتقل شود.وظیفه­ی شمارنده تولید این آدرس هاست که این شمارنده می­تواند با کلاک کار کند یعنی با هر بار کلاک دادن به شمارنده می­توان یک واحد در مقدار آدرس ROM و RAM شاهد افزایش بود.

البته لازم به ذکر است که این شمارنده به جای کار با کلاک با یک مولد(Generator) کار می­کند . این مولد در هر ثانیه ، 30 کلاک تولید می­کند و در نتیجه در زمان خیلی کمی می­توان کل اطلاعات ROM را به RAM منتقل کرد !

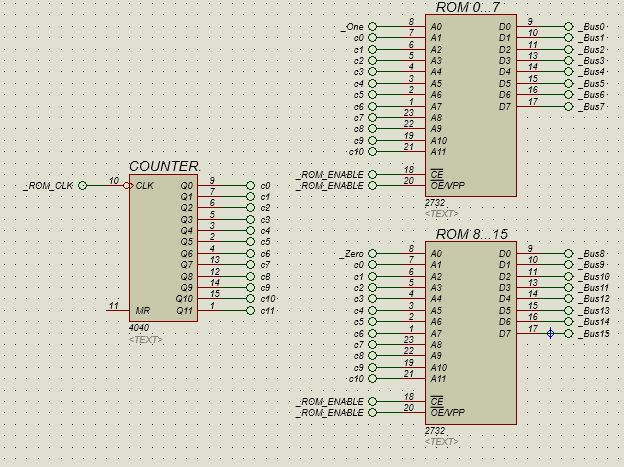
در مورد آدرس­های ROM بگونه­ای عمل کرده­ایم که همیشه یکی از ROM ها زوج آدرس­ها و دیگری فرد آدرس­ها را نگهداری کنند تا بدین ترتیب همواره دو کلمه­ی پشت سر هم را بتوانیم به RAM منتقل کنیم.

\*خروجی ROM به RAM متصل شده است که البته در شکل صفحه­ی RAM ها وجود ندارند

\*برای نوشتن در RAM ، باید ابتدا یک مقدار باینری بر روی خطوط آدرس RAM قرار دهیم سپس داده­ی متناظر را نیز آماده کنیم و آن را بر روی خطوط داده­ی RAM بگذاریم. اکنون برای نوشتن در RAM کافیست سیگنال Write را از **یک به صفر** تغییرداد. **با انجام این کار** ، داده­ی مورد نظر در آدرس مربوطه ذخیره می­شود

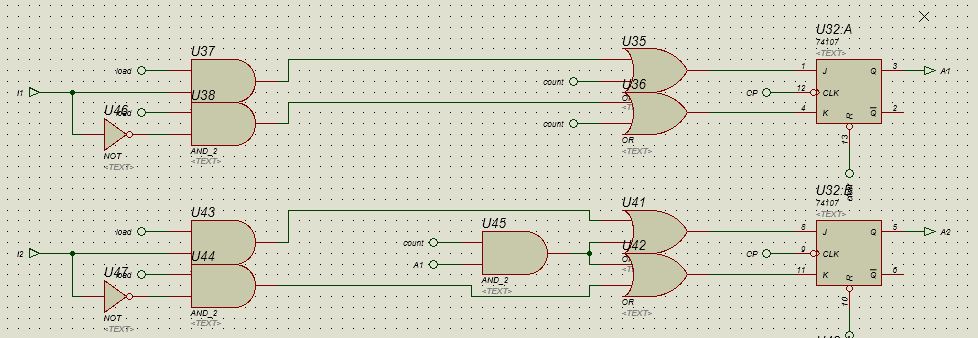
\*برای خواندن از RAM ، ابتدا آدرس مورد نظر را باید بر روی خطوط آدرس قرار دهیم سپس با **یک نمودن** خط Read ، داده­ی مربوطه بر روی خطوط داده­ی RAM ظاهر می­گردد

در شکل صفحه­ی بعد هر دو ROM و شمارنده را در یک تصویر می­توان مشاهده کرد

نمایی از ROM ها به همراه شمارنده­ی آدرس

**ثبات­ها**

در این پروژه 8 ثبات داریم. 4 تا از آن ها 16 بیتی هستند که عبارت اند از : AC,DR,IR,TR . ثباتهای AR,PC نیز 12 بیتی هستند و در آخر ثباتهای Input , Output هم 8بیتی می باشند. ساختار داخلی همه­ی ثباتها یکسان است با این تفاوت که بعضی از آنها 16 بیت ظرفیت دارند بعضی از آنها 12 بیت و برخی نیز 8 بیت. همه­ی ثبات­ها دارای قابلیت بارگذاری ، افزایش و پاک کردن هستند البته برای بعضی ثبات­ها مانند Input , Output ,IR این قابلیت غیر فعال شده ، زیرا هیچ وقت به آن نیاز ندارند.

در شکل زیر بخش کوچکی از ساختار داخلی ثبات­ها قابل مشاهده است

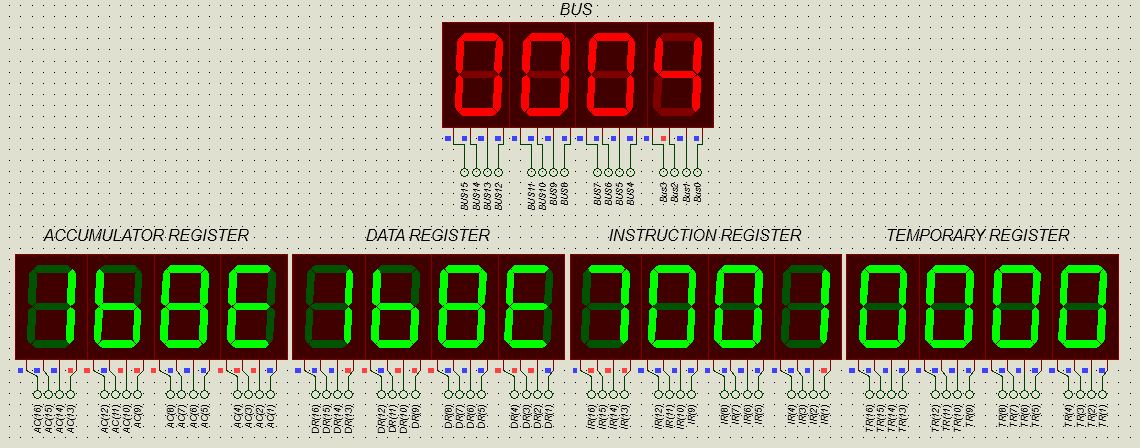
در واقع ما در ساختار داخلی ثبات­ها ، یک **شمارنده** پیاده سازی کرده­ایم که این شمارنده با کلاک ساعت کار می­کند بطوریکه هرگاه لازم باشد محتوای خودش را پاک می کند هر وقت نیاز باشد شمارش آن متوقف می­شود و همچنین می­تواند از هر عدد دلخواه شروع به شمارش کند و نیز می­تواند یک واحد به خودش اضافه کند. این مجموعه از قابلیت­ها همان چیزی را ایجاد می­کند که ما از ثبات انتظار داریم

**نحوه­ی پیاده سازی داخلی ثبات­ها**

همه­ی ثبات ها بر اساس

**واحد نمایش**

واحد نمایش همانگونه که از نامش پیداست برای نمایش خروجی قسمت­های مختلف کامپیوتر به کار می­رود .

 در شکل زیر بخشی از واحد نمایشِ اصلی را مشاهده می­کنید

هر رقم در واحد نمایش بالا ، در مبنای 16 می باشد

به غیر از این واحد نمایش که ما آن را واحد نمایش اصلی نامیده­ایم خروجی و ورودی ثبات­ها ، واحد کنترل ، واحد ALU و واحد حافظه نیز همگی قادر به نمایش خروجی خود می­باشند که نمایش آن­ها در مبنای 2 هستند

**معرفی نرم افزار 010 Editor**

واضح است فایلی که ما به ROM می­دهیم تبدیل به کد اسکی و در نهایت تبدیل به کد باینری می­شود. از طرف دیگر ما می­باید ابتدا دستورات را به باینری بنویسیم و سپس کاراکتر متناسب با آن را پیدا کنیم و آن را در یک فایل باینری ذخیره کنیم که پیدا کردن کاراکتر­های مربوط به کدهای باینری مختلف بسیار مشکل است.

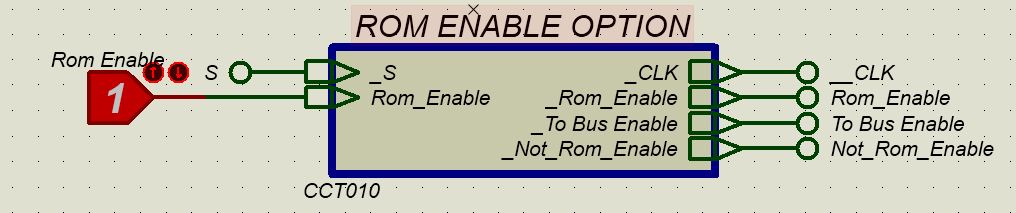
نرم افزار 010 Editor به ما این امکان را می­دهد که کدهای خود را در مبنای 16 بنویسیم و سپس این نرم افزار بطور خودکار آن را به کاراکترهای مربوطه تبدیل می کند و آن را در یک فایل ذخیره می کند. پس از اینکه دستورات مورد نظر را با استفاده از این برنامه نوشتیم و فایل مورد نظر نیز تولید شد ، حالا باید پسوند این فایل ایجاد شده را Bin بگذاریم . حالا فایل مورد نظر آماده­ی Load شدن در ROM است که نحوه­ی انجام این کار را در صفحه­ی بعد ملاحظه می­کنید.

**نحوه­ی کار با کامپیوتر**

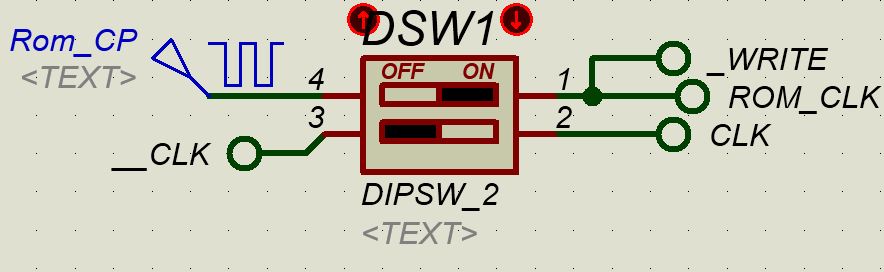
برای استفاده از کامپیوتر و کار با آن ابتدا باید سراغ کار با ROM رفت .

همانطور که در بحث های مربوط به حافظه ذکر کردیم ROM های مورد استفاده در این کامپیوتر قابلیت بارگذاری کردن و خواندن فایل را نیز دارند. برای این کار ابتدا باید به داخل واحد حافظه رفت. سپس بر روی ROM کلیک راست کرده و گزینه­ی Edit Properties را انتخاب کرد اکنون می­توان با دادن آدرس یک فایل (با پسوند Bin) ، ROM را مقدار اولیه داد.

برای گام بعد باید ROM را فعال نمود تا ROM وارد مدار گردد. مطابق شکل زیر باید فعالساز ROM را یک نمود

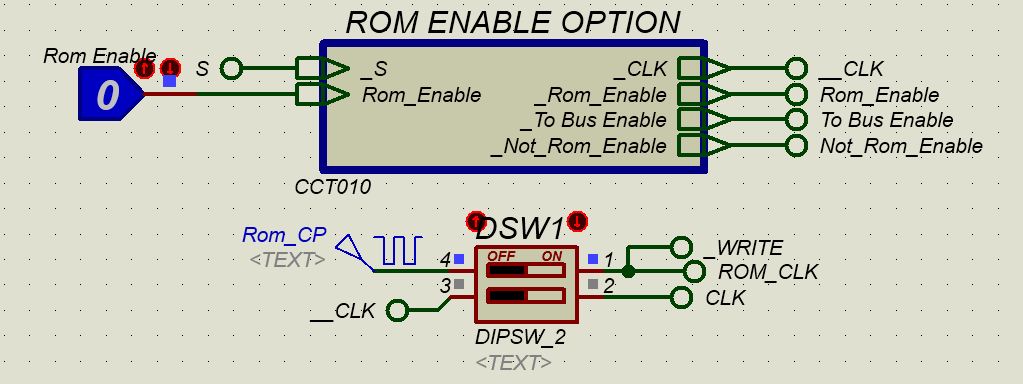


سپس کلید زیر را (منظور از کلید زیر ، کلید اولی است) از حالت Off به On تغییر می دهیم.



با این کار مولد ، صفر و یک هایی را که به سرعت دارد تولید می کند به شمارنده­ی موجود در واحد حافظه تحویل می دهد و اطلاعات ROM بسرعت به RAM منتقل می­شود .به همین ترتیب این صفر و یک های تولید شده ، بسرعت در اختیار خطوط Write نیز قرار گرفته و در نتیجه اطلاعات در RAM ذخیره می شوند

پس از چند ثانیه باید ROM را از مدار خارج کرد برای این کار بایستی کلید بالایی را از حالت On به Off تغییر داد و ضمنا فعالساز مربوط به ROM را هم صفر کرد که این تغییرات در شکل صفحه­ی بعد قابل مشاهده است



حالا می­توان با On کردن کلید کلاک کامپیوتر و در عرض چند ثانیه ، مشاهده کرد که کامپیوتر وظیفه­ی خود را به درستی انجام می­دهد

\*با انجام بررسی­های عملی مشاهده شد که فرکانس مولد را می­توان تا حدود 1M در ثانیه افزایش داد اما از این حدود به بعد ROM درست عمل نخواهد کرد و نخواهد توانست اطلاعات خودش را به درستی به RAM منتقل کند

\*مولد مربوط به پالس ساعت کامپیوتر نیز تقریبا به همین گونه عمل می­کند و از حوالی 1M کلاک در ثانیه به بعد ، درست عمل نمی­کند.

می توان فرکانس این مولد ها را نیز تغییر داد ، برای این کار باید بر روی قطعه­ی مربوطه کلیک راست کرده و گزینه­ی Edit Properties را انتخاب نمود.اکنون در این صفحه می­توانیم هر عددی را که مناسب می­دانیم برای مولد مورد نظر انتخاب کنیم.