دانشگاه صنعتی شریف

دانشکده مهندسی برق

آزمایشگاه ساختار کامپیوتر و میکروپروسسر

پاییز ۹۹

گروه درس دکتر باقری شورکی

پروژه اول **طراحی سیستم میکروپروفسور – MPF**		
نام و نام خانوادگی همکاران	**صدرا صبوری**	
شماره دانشجویی	**۹۷۱۰۱۹۷۲**	
ارزشیابی	پیش گزارش (۲۰)	
	آمادگی (٤٠)	
	حضور و غیاب (۱۰)	
	گزارش (۳۰)	
	نمره کل (۱۰۰)	

نام دستیار تصحیح کننده:	**تاریخ:**
ملاحظات:	

۱. بخش اول

دستگاهی که در ابتدا طراحی آن هستیم دستگاهی است که وظیفه اجرای دستورات برنامه (به عنوان ورودی) به صورت Real-Time و ذخیره تاثیرات آن روی حافظه برای استفاده کننده از آن فراهم می سازد، با توجه به نظریه زبان ها و اتوماتا طراحی چنین ماشین محاسبی در حالت کلی یک مسئله تصمیم ناپذیر ولی تشخیص پذیر است به این منظور قطعه کدی که روح اصلی برنامه را اجرا خواهد کرد باید توانایی رفتار با خانه های حافظه به همان شکلی که سیستم اصلی با کد اصلی برنامه رفتار میکند را داشته باشد-یعنی ۳ مرحله Fetch, Decode, Execute ولی این بار به صورت نرم افزاری-.

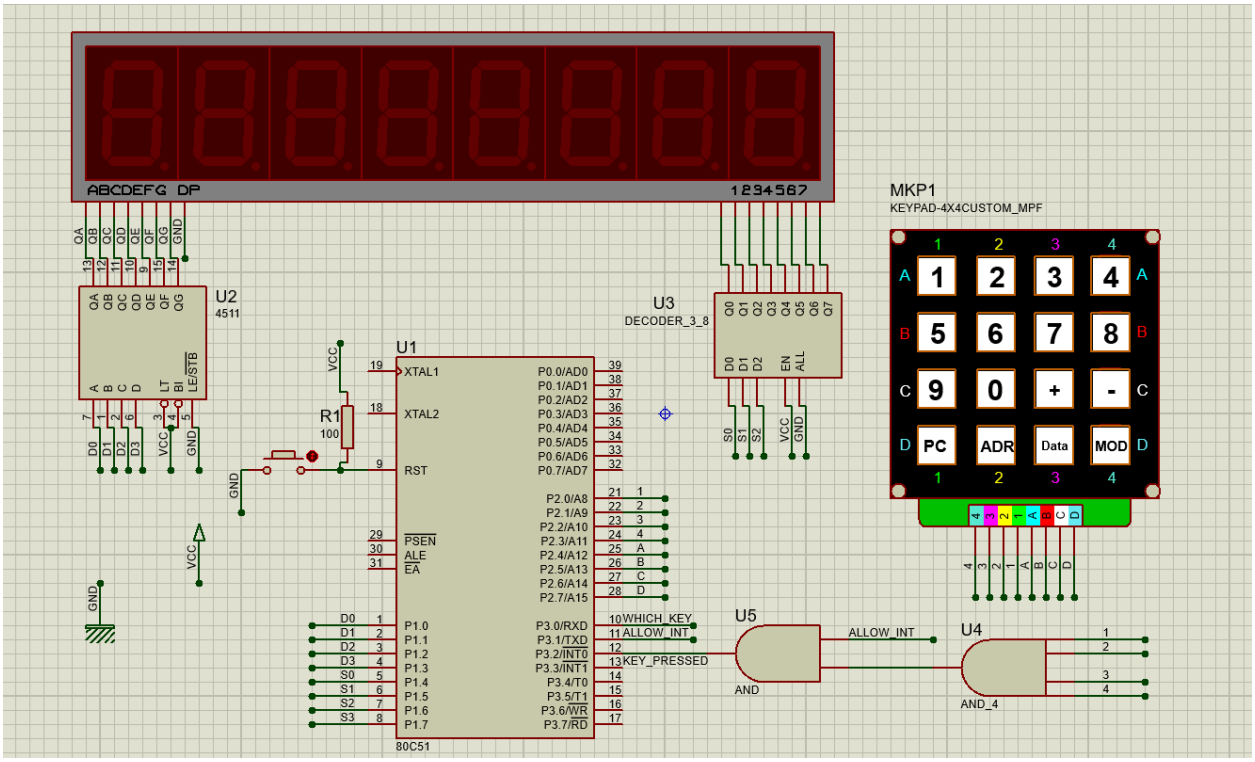
مطابق عملکرد نسخه اول همین محصول در دنیای واقعی، عملکرد کلی این دستگاه به این شکل است که ابتدا با فشردن دکمه ADDRESS توسط کاربر دستگاه در وضعیت نمایش اطلاعات خانه های حافظه را به خود بگیرد و با وارد کردن خانه مورد نظر حافظه توسط کاربر، اطلاعات خانه مشخص شده نمایش داده شود، ضمنا بهتر است زمانی که تعداد ارقام وارد شده توسط کاربر از تعداد مشخصی بیشتر شد، ارقام نشان دهنده آدرس از سمت دیگر خارج شوند، همچنین بهتر است برای جلو گیری از اختلاط کد اصلی و حافظه مورد نیاز برنامه اصلی با برنامه کاربر، **آدرس را از مکان مشخصی، مثلا 1000H آغاز کنیم**.

در گام بعدی، بعد از مشخص شدن مکانی که باید در آن داده را بنویسیم، با فشردن Data میتوانیم داده مورد نظر را در مکان آدرس مورد نظر بنویسیم (این داده میتواند Op Code مربوط به دستورت یا Operand مربوط به دستورات باشد) و نیز برای ذخیره شدن آن از دکمه های + یا – استفاده بکنیم که یک خانه به بالا یا پایین برویم و محتویات آن خانه ها را عوض کنیم.

نمای اولیه طراحی شده برای این پروژه به صورت زیر است:

یکی از معایب پروتئوس این است که keypad مناسب به اندازه ۸ در ۸ ندارد و به جای آن از **keypad ٤ در ٤ استفاده کردیم و با decompose کردن مدل و عوض کردن نام کلیدها آنان را به دلخواه خود در آوردیم، اما مشکل اینجاست که برای وارد کردن ورودی ها که در مبنای ١٦ است تعداد ارقام است این مشکل در قسمت آدرس با دکمه های + و – قابل حل است ولی برای قسمت داده میتوان هم محدودیت ورودی اعداد ایجاد کرد و هم میتوان عدد ورودی توسط کاربر را به صورت در مبنای ۱۰ تعریف کرد و بدین صورت میتوان از ۰ تا ۲۵۵ را در ٤**

سون سگمنت باقی مانده نمایش داد و در آخر باقی مانده تقسیم عدد ورودی توسط کاربر را بر ۲۵۶ به عنوان عدد ورودی مورد استفاده قرار می گیرد.



اجزای اصلی مدار مطابق خواست سوال طراحی شده اند ۲ دیکودر یکی برای تبدیل اعداد به معادل **سگمنت های ۷سگمنت و دیگری برای مشخص کردن اینکه الآن کدام یک از ۷ سگمنت ها می بایست روشن شود مورد استفاده قرار میگیرد، این کار باعث میشود که بتوان با تنها استفاده از یکی از پورت های آی سی 8051 کل فرآیند نمایش را کنترل کرد که میدانیم بسیار ارزشمند است**، زیرا احتمالا به زودی با مشکل کمبود پورت مواجه خواهیم شد.

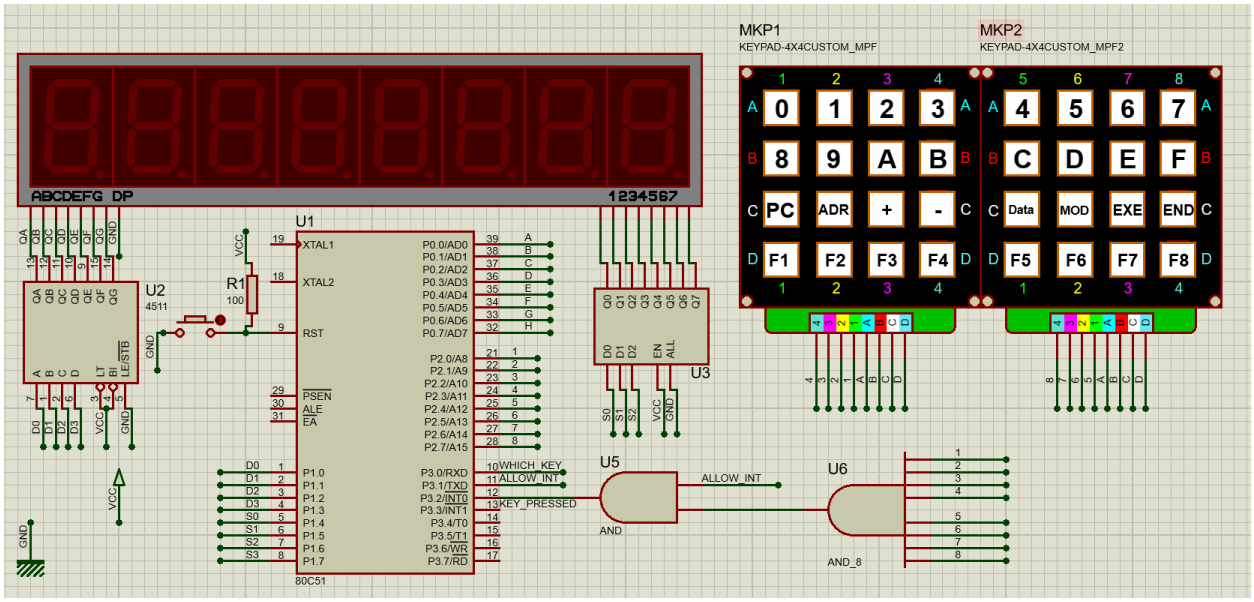
فرآیند تشخیص کلید زده شده هم مانند آزمایش ۷ ام آزمایشگاه این درس با کلیک شدن حداقل یکی از کلید ها Interrupt مربوط به کلید فعال می شود و CPU مشغول دریافت ورودی از کاربر می شود.

بعد از اتمام این مراحل بهتر است فضایی را برای دیتایی که باید برای نمایش به روی ۷ سگمنت ها استفاده بشوند را در محلی ذخیره کنیم (هر رقم معادل یک بایت است و با در نظر گرفتن ۸ بایت برای نمایش این اعداد می توان پیوسته این اعداد را نمایش داد و از طریق برنامه تنها محتویات این خانه ها را عوض کنیم که این موضوع باعث عوض شدن مقادیری نمایشی توسط ۷سگمنت ها نیز می شود)

دکمه Mod روی این دستگاه برای تعویض مد کاری احتمالی دستگاه است و نیز دکمه PC برای زمانی است که در حالت ADDRESS به جایDATA کلیک شود و در نتیجه آن، برنامه کابر از همان آدرس شروع به اجرا شود.

مشکل دیگری که در این طراحی وجود دارد مشکل بعضی دستورات خاص که روند خطی برنامه را مختل میکنند است، دستوراتی مثل CALL و یا دستورات شرطی که در آن ها مقدار PC به گونه ای غیرقابل بازگشت عوض می شود، **در این حالات می توانیم از بانک های رجیستری و یا Stack برای ذخیره کردن PC قبل از اجرای دستورات برنامه کاربر استفاده کنیم و بعد از اجرای آن ها به روند اصلی برنامه بازگردیم.**

یکی از راه حل های احتمالی برای حل مشکل ورودی ها (چون ممکن است دردسر ساز شوند) استفاده از ۲ ماژول کلید و متصل کردن آن ها به یکدیگر است:



مشکلی که در این حالت به وجود می آید این است که در صورت نیاز به استفاده از حافظه خارجی با کمبود پورت روبه رو هستیم که برای **رفع آن میتوان ساختار کلید ها را به استفاده از ۲ BDL کنترل کرد که در نتیجه تنها نیاز به استفاده از ۸ پورت بود.**

۲. بخش دوم

همانطور که گفته شد برای حل این مشکل هم میتوان از رم بیرونی برای ذخیره برنامه کاربر استفاده کرد و یا یک محدوده برای کد اصلی برنامه تعیین کرد که کاربر توانایی نوشتن برنامه خود از آن آدرس به بعد را داشته باشد برای اجرای برنامه کاربر میتوانیم کل آن را به کل یک subroutine نگاه کنیم و با ردن دکمه اجرا برنامه را به آدرس مشخص شروع برنامه کاربر Call کنیم و قبل از این کار آخرین خانه برنامه کاربر را نیز با Opcode مربوط به RET پر بکنیم تا پس از اتمام برنامه کابر برنامه به روند خود بازگردد، مشخصا در هنگام اجرای برنامه کاربر روند برنامه اصلی مختل خواهد بود که قابل پیش بینی بود.

با پیاده سازی این روش (استفاده از حافظه استک برای به یاد داشتن نقطه قبلی اجرای برنامه) هیچ کدام از دستورات شرطی دچار مشکل نمی شود (تا زمانی که آدرس های شرطی وارد کد اصلی نشوند، که طبیعتا کاربر می بایست به آدرس دستورات برنامه خود branch بزند و هزینه branching به محدوده کد اصلی بر عهده کابر است (کاری که سیستم عامل میکند این است که در صورتی که برنامه های کاربردی اجازه دسترسی به حافظه مربوط به سیستم اصلی میکند یک Exception تولید میکند ولی در اینجا برای این کار می بایست نحوه اجرای شرط ها را تغییر دهیم که به صرفه نیست) همچنین برای استفاده از دستور Call نیز مشکلی به وجود نمی آید، تنها تفاوت این است که به جای شروع از خانه اول استک از خانه دوم استک برنامه آغاز خواهد شد.