



دانشگاه صنعتی شریف
دانشکده مهندسی برق

پروژه درس ساختار کامپیوتر میکروپروسسور و آز

استاد : دکتر باقری

سید محمد عرفان باطنی – 400100792

قبل از شروع ، ابتدا پروژه را یک بررسی کلی می کنیم. این پروژه شامل 5 بخش می باشد که عبارت اند از :

تعریف پروژه – چالش احتمالی – پیاده سازی – تست سیستم – نتیجه گیری.

1. تعریف پروژه : شامل توصیف رسمی از طراحی سیستم ، بررسی ابعاد فنی و چالش های احتمالی و ذکر راه حل های چالش ها

2. چالش احتمالی : بررسی دستورات مرسوم که دچار مشکل میشوند یا خیر و سپس توجه به تغییرات آن ها البته در صورت لزوم

3. پیاده سازی : شبیه سازی سیستم ذکر شده در نرم افزار پروتئوس

4. تست سیستم : نوشتن کد برنامه ای به زبان اسمبلی که عدد n را دریافت کند و از 1 تا n را در خانه های حافظه با شروع از n بنویسد.

5. نتیجه گیری : در این بخش چالش هایی که با آن ها رو به رو بودیم را شرح می دهیم و سپس جمع بندی نهایی را داریم.

بخش 1 (

میکروپروفیسور چیست؟

The Micro-Professor MPF-1 was a microcomputer designed to teach the fundamentals of machine code and assembly language. It was contained within a plastic case that could be placed on a bookshelf like any other training manual or book. It was manufactured by Multitech, known since 1987 as Acer. The item is still being sold as of 2018 by Flite Electronics International, a former international distributor for Acer who purchased the copyright for the device in 1993.

Advert text:

"Here in one attractive package is a Z80 based microcomputer to lead you step by step to a thorough knowledge of the world of microprocessors. The Micro-Professor is a complete hardware and software system whose extensive teaching manual gives you detailed schematics and examples of program code. A superb learning tool for students, hobbyist and microprocessor enthusiasts, as well as an excellent teaching aid for instructors of electrical engineering and computer science courses. But the Micro-Professor is much more than a teaching device. With it you can do bread-boarding and prototyping, designing your own custom hardware and software application with Z80, 8080 and 8085 compatible code. The standard 2K bytes of RAM is expandable to 4K, and the standard 2K bytes of ROM can be increased to 8K. All this plus a built-in speaker, a cassette interface, and sockets to accept optional CTC/PIO. Bus is extendable. As well as being an

exciting learning tool, the Micro-Professor is a great low-cost board for OEM's. MPF-Basic software is included in the ROM."

Hardware Specifications

CPU: Zilog Z-80 CPU with 158 instructions and 2.5 Mhz maximum clock rate. The MPF-I system clock is 1.79 Mhz.

ROM: Single +5V EPROM 2516 (2532), total 2K (4K) bytes. Monitor EPROM Address: 000-07FF (0FFF).

RAM: Static RAM: 6116, total 2K bytes. Basic RAM Address: 1800-1FFF.

Memory Expansion Area: Single +5V EPROM 2516/2716/2532/2732 EPROM or 6116 static RAM on-Board Expansion Address: 2000-2FFF.

I/O Port: Programmable I/O Port 8255, a total 24 parallel I/O lines are used for keyboard scanning and seven segment LED display control. I/O addresses: 00-03. Programmable PIO, a total of 16 parallel I/O lines, I/O address: 80-83H. Programmable CTC, a total of 4 independent counter timers channels, I/O address: 40-43H.

Display: 6-digit, 0.5", 7-Segment red LED display

Keyboard: 36 keys including 19 function keys, 16 hexadecimal keys and 1 user-defined key.

Speaker and Speaker Driver Circuits: A 2.25" - diameter speaker is provided for user's expansion.

با توجه به متن بالا و خواسته ی سوال در واقع ما میخواهیم یک مینیم سیستم جامع جهت بررسی کد ذکر شده در بخش چهار بر روی میکروکنترلر 8051 اجرا کنیم. باتوجه به ماهیت این میکروپروسسور و کارایی آن باید شامل اجزایی باشد من جمله میکروکنترلر 8051 ، تعدادی 7-Segment ، یک صفحه کلید (که می توان به مانند عکس داخل توضیحات پروژه علاوه بر اعداد شامل حروف نیز باشد که باتوجه به عدم استفاده از حروف از کیبوردی استفاده میکنیم که حروف نداشته باشد و برای کامل تر شدن آن میتوان از کیبورد 8*8 استفاده کرد که در ادامه به تشریح آن میپردازیم.) ، PPI 8255 ، دیکودر 3 به 8 برای مموری مپینگ ، حافظه خارجی ، لچ 8 بیتی و ...

حال که به شمای کلی سیستم آشنا شدیم میتوان به چالش های احتمالی پیش روی خود بپردازیم :

چالش اول : وقتی کد اجرا می شود، خروجی دارد ولی همه ی خروجی ها به مانند هم نیستند که در هر صورت باید بر روی حافظه ذخیره شوند که این احتمال متفاوت بودن آن ها میتواند مشکل ساز و چالش ساز باشد مانند به هم خوردن توالی عملیات.

راه حل 1 : می توان تعدادی رجیستر انتخاب کرد و داخل آن ها تعداد ورودی ها و خروجی ها را ذخیره کنیم تا این مشکل حل شود.

راه حل 2 : می توان تعدادی رجیستر انتخاب کرد ولی این بار داخل آن ها تعداد تعداد عملیات انجام شده را ذخیره کنیم تا این مشکل نیز حل شود.

چالش دوم : یکی از مهم ترین چالش های پیش روی ما این است که این سیستم از کجا بفهمد و تشخیص بدهد که مقادیر وارد شده دقیقاً قرار است چه کاری را انجام دهند؟ به طور کلی تر opcode هستند یا operand ؟

راه حل : راه حل این چالش تقسیم بندی و دسته بندی کلید هاست. به طوری که تعدادی از کلید ها را برای اعداد در نظر میگیریم همینطور تعدادی نیز به حروف انگلیسی و به همین نحو بعضی از کلید ها را میتوان برای استفاده های دیگر

اختصاص داد به عنوان مثال کلیدی برای روشن و خاموش کردن ، کلیدی برای run کردن برنامه ، کلیدی برای delete کردن دیتایی ، کلیدی برای نمایش دادن اطلاعات روی 7Seg به نام SHOW و ...

چالش سوم : باتوجه به اینکه سیستم ما باینری است و برای دریافت آدرس 16 بیتی دچار مشکل می شویم چون که سیستم ما 8 بیتی است.

راه حل : برای حل این مشکل می توان دستوراتمان را در چند مرحله به سیستم بدهیم. به عنوان مثال میتوان برای دریافت آدرسی که 16 بیتی است دو بار آن را دریافت کنیم. یکبار 8 بیت اول و بار دیگر 8 بیت دوم.

بخش 2)

مشکل اساسی این بخش ، مشکل در آدرس دهی است. یعنی ممکن است از خانه ای از حافظه شروع به نوشتن کد کنیم که آن بخش حافظه از قبل استفاده شده باشد و کدی داخل آن نوشته شده باشد. پس این کار ما باعث از بین رفتن آن ها می شود که این می تواند بسیار آسیب زا باشد.

از مهم ترین دستورات ذکر شده در کلاس میتوان به `RET` , `JMP` , `ADD` , `CALL` , `SUB` ، دستورات منطقی و ... هستند که اگر بخواهیم طبق مشکل بالا آن ها را دسته بندی کرد، هر دستوری که در آن پرش داشته باشد بیشتر در معرض این خطر هست به مانند : `RET` , `JMP` , `CALL` و ... زیرا توابع را در خانه هایی از حافظه ذخیره کردیم که ممکن به دلیل مشکل آدرس دهی محتوای آن ها حذف شده باشد.

اما در دستوراتی از قبیل `ADD` , `SUB` و دستورات منطقی احتمال خطای شان خیلی کمتر است زیرا پرش در فضای حافظه ندارند و بیشتر عملیات آن ها در فضای `ALU` انجام می شود.

راه حل : استفاده از `PPI 8255`

سوال : `PPI 8255` چیست؟

The **Intel 8255** (or **i8255**) Programmable **Peripheral Interface** (PPI) chip was developed and manufactured by Intel in the first half of the 1970s for the **Intel 8080** microprocessor. The 8255 provides 24 parallel input/output lines with a variety of programmable operating modes.

The 8255 is a member of the **MCS-85 Family** of chips, designed by Intel for use with their **8085** and **8086** microprocessors and their descendants. It was first available in a 40-pin **DIP** and later a 44-pin **PLCC** packages. It found wide applicability in digital processing systems and was later cloned by other manufacturers. The 82C55 is a **CMOS** version for higher speed and lower current consumption.

The functionality of the 8255 is now mostly embedded in larger **VLSI** processing chips as a sub-function. A **CMOS** version of the

8255 is still being made by [Renesas](#) but mostly used to expand the I/O of [microcontrollers](#).

The 8255 gives a CPU or digital system access to programmable parallel I/O. The 8255 has 24 input/output pins. These are divided into three 8-bit ports (A, B, C). Port A and port B can be used as 8-bit input/output ports. Port C can be used as an 8-bit input/output port or as two 4-bit input/output ports or to produce handshake signals for ports A and B.

The three ports are further grouped as follows:

1. Group A consisting of port A and upper part of port C.
2. Group B consisting of port B and lower part of port C.

Eight data lines (D0–D7) are available (with an 8-bit data buffer) to read/write data into the ports or control register under the status of the

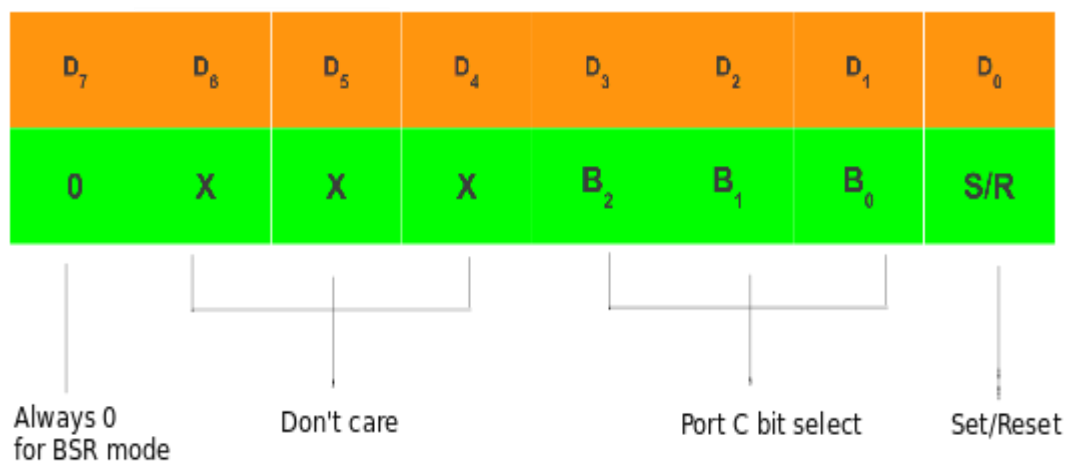
RD (pin 5) and **WR** (pin 36), which are active-low signals for read and write operations respectively. [Address lines](#) A_1 and A_0 allow to access a data register for each port or a control register, as listed below:

A_1	A_0	Port selected
0	0	port A
0	1	port B
1	0	port C
1	1	control register

The control signal chip select **CS** (pin 6) is used to enable the 8255 chip. It is an active-low signal, i.e., when $CS = 0$, the 8255 is enabled. The **RESET** input (pin 35) is connected to the RESET line of system like

8085, 8086, etc., so that when the system is reset, all the ports are initialized as input lines. This is done to prevent 8255 and/or any peripheral connected to it from being destroyed due to mismatch of port direction settings. As an example, consider an input device connected to 8255 at port A. If from the previous operation, port A is initialized as an output port and if 8255 is not reset before using the current configuration, then there is a possibility of damage of either the input device connected or 8255 or both, since both 8255 and the device connected will be sending out data.

The control register (or the control logic, or the command word register) is an 8-bit register used to select the modes of operation and input/output designation of the ports.



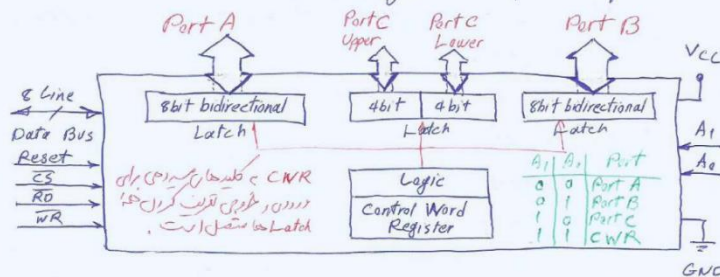
8255 Control Register format for BSR Mode

دستگاه‌های جانبی Peripherals

در معرفی دروس دخی به ریل جانبی CPU لازم داشت آنرا به شکل حافظه بسند. در تبدیل فیزیک بسیاری از دستگاه‌ها به حالت حافظه Latch های دوطرفه نگه داشته هستند. علاوه بر این در بسیاری از دروس دخی ها CPU کانتر از گیت‌های زمان‌بندی گوناگون در طولانی است (همیشه نایب، همیشه گیت، ...). که این مورد بسیاری از دروس CPU را می‌گیرد. همچنین گاهی فیزیک برخی دستگاه‌ها در ذات هر فضای جانبی برپا (ستونی) اطلاعات کار می‌کنند (MODM، FSM، FSM، ...) که تبدیل این رفتار به رفتار مولاری جانبی دارد. در CPU نیز مستقیم استفاده از سخت افزارها به نایب است. همچنین دلیل درج‌های مجتمع خاصی برای هرولت انجام فعالیت‌های فزونی در کنار یک CPU خاص طراحی کرده‌اند.

1. Parallel Peripheral Interface (P.P.I.)

این مدل به جمع دو نوع Latch دوطرفه برآید به در داخل یک مدل به جمع هستند 8255 (Programmable Peripheral Interface)

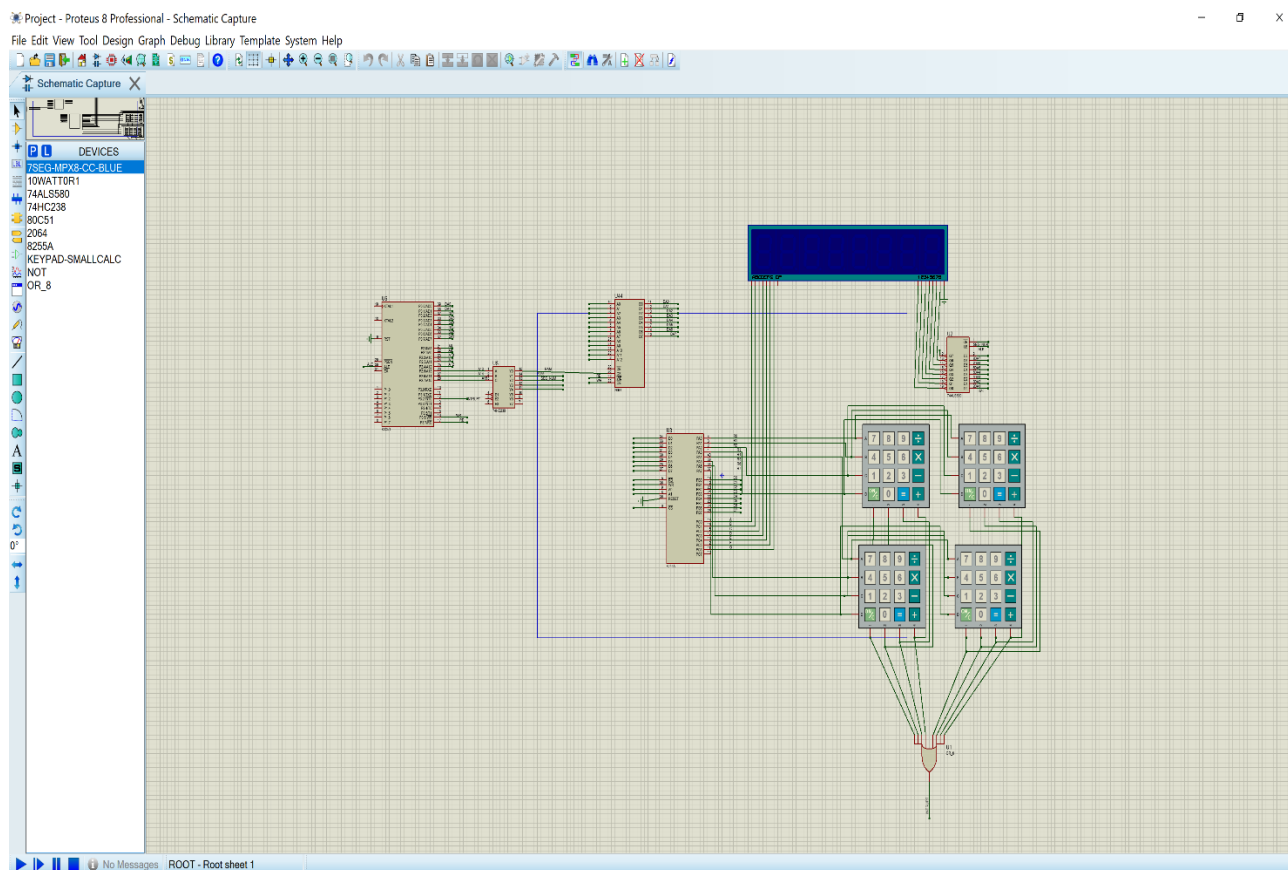


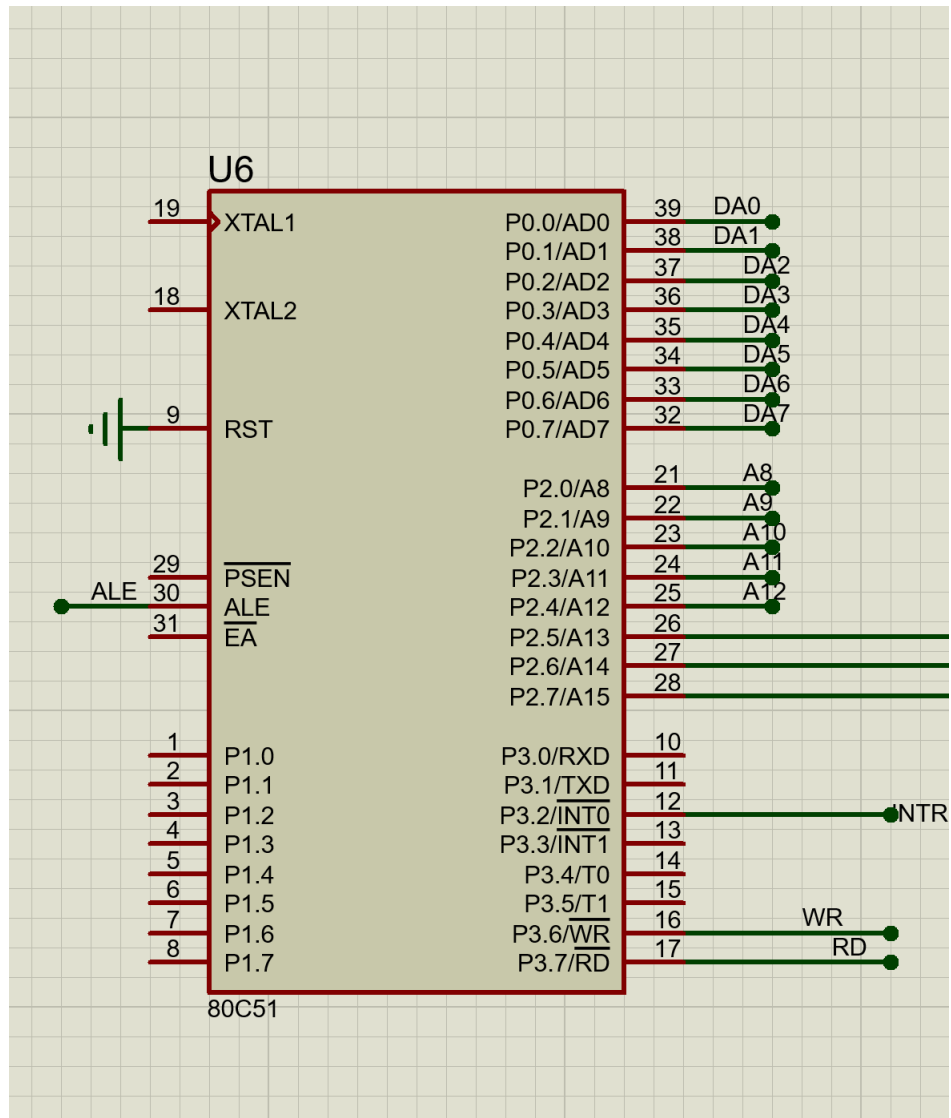
۳۴

با استفاده از PPI 8255 و معموری مینینگ مناسب می‌توان از این چالش نیز عبور کرد.

بخش 3 (

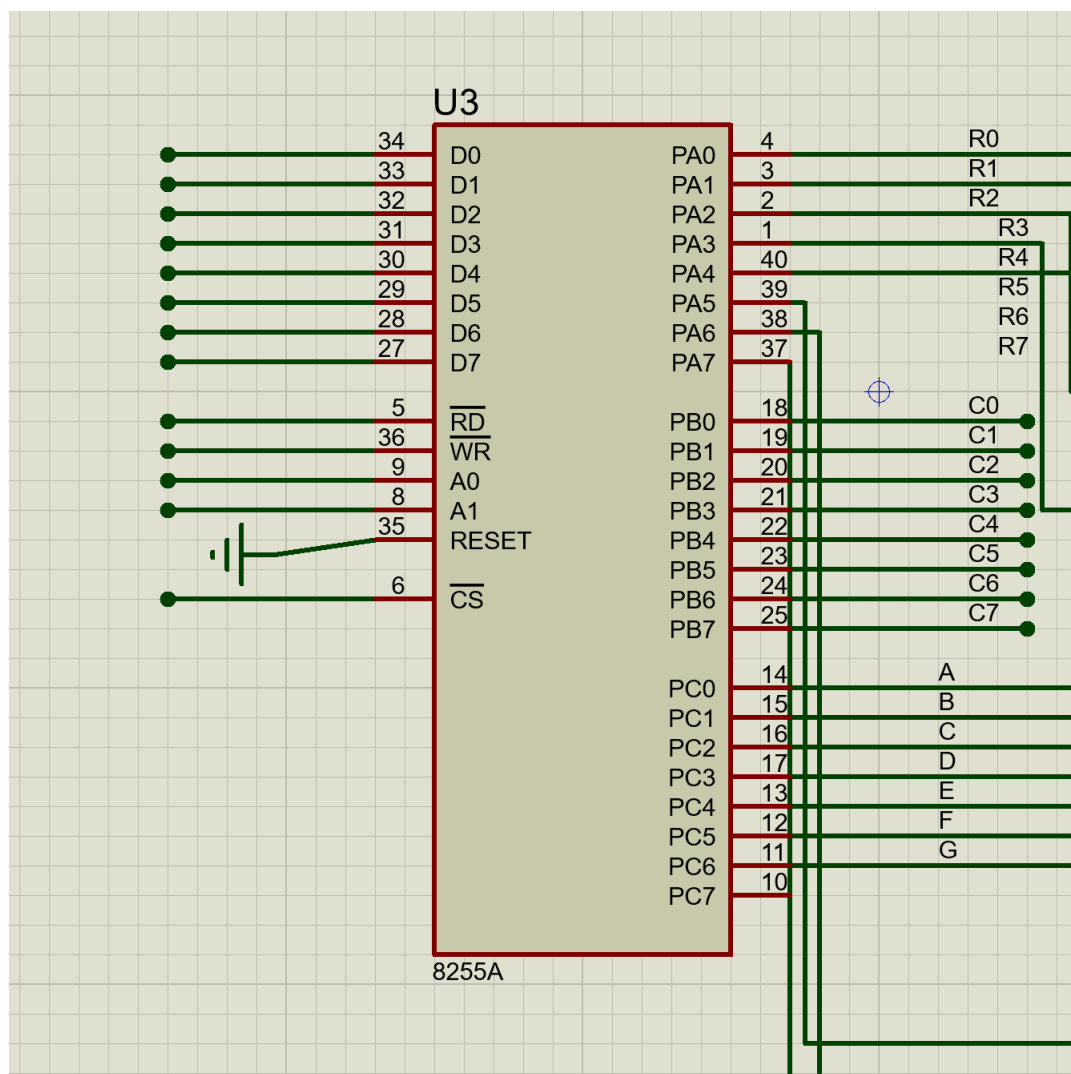
نمای کلی مدار به صورت زیر است همچنین فایل کامل این طراحی در پوشه ی simulations قرار گرفته است.



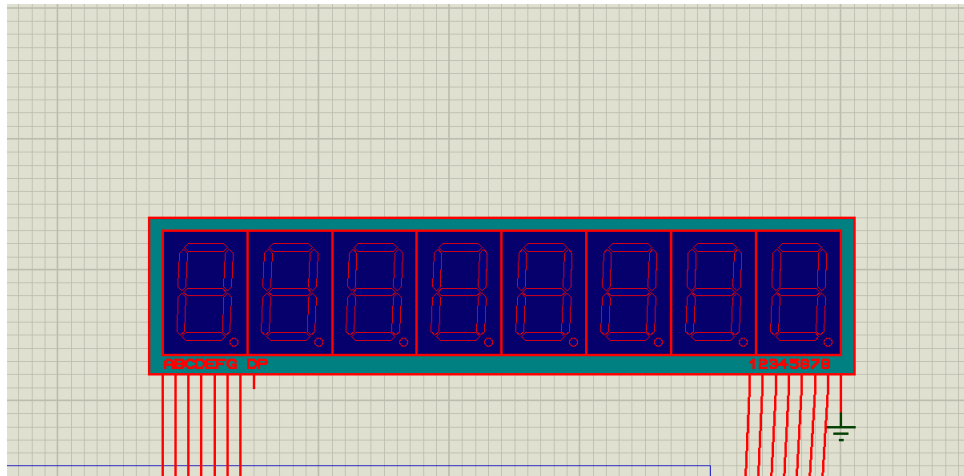


8051

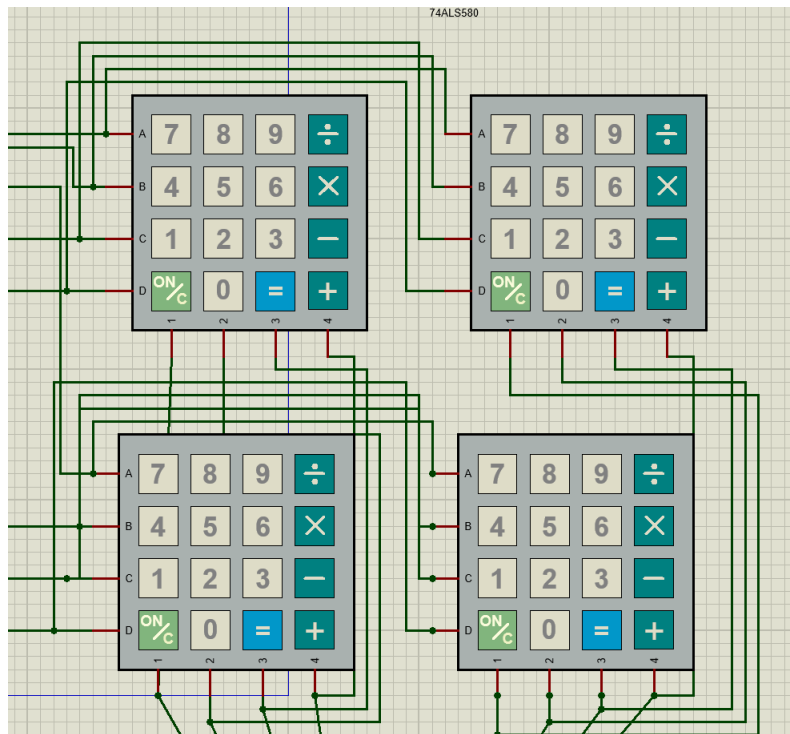
همانطور که بالاتر گفتیم PPI نقش مهمی در مموری مپینگ دارد و همینطور رابط بین 8051 و کیبورد و 7Segment هست.



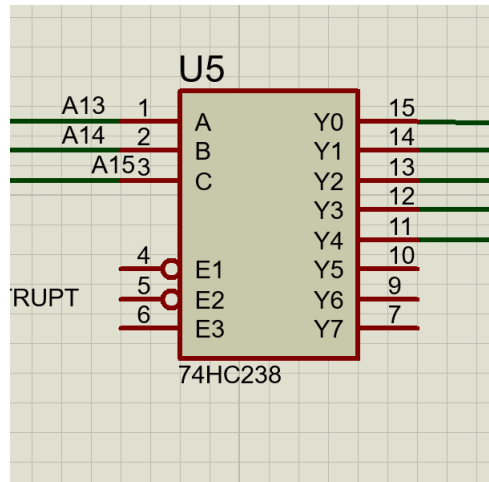
PPI



7-Segment



Keyboard



Decoder 3-8

- همانطور که در کلاس نیز مطرح شد برای استفاده ی همزمان از باس آدرس و باس دیتا میتوان از یک Latch استفاده کرد که با سیگنال ALE کنترل می شود.

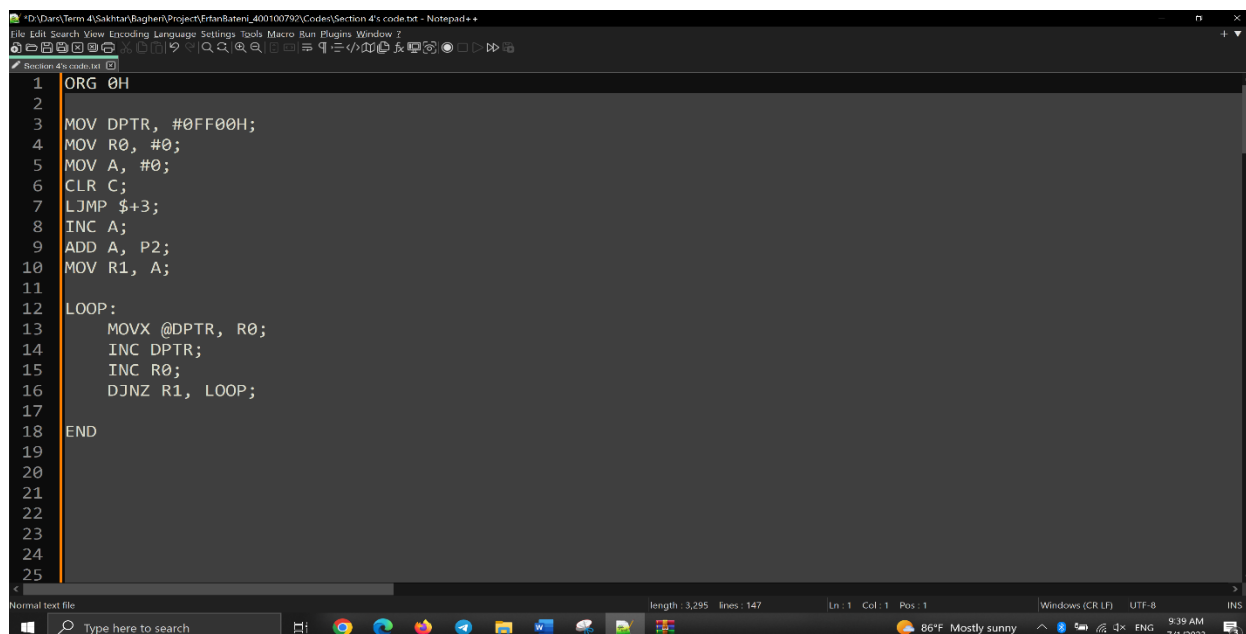
بخش 4)

کد این بخش به طور کامل در پوشه ی Codes آورده شده است.

توضیحات کد :

در اینجا کدی به زبان اسمبلی 8051 وجود دارد که عدد n را از کاربر دریافت می کند و اعداد 1 تا n را در مکان های حافظه با شروع از آدرس n می نویسد. در این کد از کاربر انتظار می رود که عدد n را از طریق پورت P2 وارد کند و سپس در کد از یک حلقه استفاده می شود و مقدار رجیستر R0 را از 1 تا n افزایش می دهد. هر بار، مقدار R0 را در محل حافظه نشان داده شده توسط DPTR می نویسد و DPTR را برای مکان حافظه بعدی افزایش می دهد. این حلقه تا زمانی ادامه می یابد که همه اعداد از 1 تا n در حافظه نوشته شوند. ولی ممکن است که ایرادی پیش بیاید که این بازه نوشتن با کد لود شده برای شبیه سازی تلاقی داشته باشد.

راه حل : برای حل این موضوع می توان آدرس مکان ها و خانه هایی که از قبل پر است (یعنی اگر در آنجا بنویسیم اطلاعاتی از بین می رود) را ذخیره کنیم. با این کار دیگر این مشکل پیش نمی آید.



```
1 ORG 0H
2
3 MOV DPTR, #0FFF00H;
4 MOV R0, #0;
5 MOV A, #0;
6 CLR C;
7 LJMP $+3;
8 INC A;
9 ADD A, P2;
10 MOV R1, A;
11
12 LOOP:
13     MOVX @DPTR, R0;
14     INC DPTR;
15     INC R0;
16     DJNZ R1, LOOP;
17
18 END
19
20
21
22
23
24
25
```


بخش 5)

قطعا این پروژه ابعاد بسیار بزرگتری دارد و چالش هایی فراتر از چالش های ذکر شده ولی به دلیل کمبود وقت و حجم خواسته شده در همین حد قناعت می کنیم. از دیگر چالش ها می توان به محدودیت حجم حافظه اشاره کرد که با افزایش تعداد حافظه خارجی یا حجم آن تا حدودی این چالش را برطرف کرد. البته راه حل هایی که در قسمت های مختلف ذکر شد بهینه ترین حالت ممکن نیست و قطعا میتوان راه حل های بهینه تر و بهتری ارائه داد.

منابع :

- <https://www.computinghistory.org.uk/det/16197/Micro-Professor-EPB-MPF-1B/>
- <https://ostad.nit.ac.ir/payaidea/ospic/file7828.pdf>
- https://en.wikipedia.org/wiki/Intel_8255

پاپان