بسمه تعالی

# اتصال ریزپردازنده ۸۰۸۶ به LCD متنی در پروتئوس و چاپ یک پیغام متنی


نگارش:

سید علی مجتبوی

امیررضا فصیحی راد

امیرحسین کفاشی

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

تقدیم به

John W. Backus

خالق اولین زبان سطح بالا

Fortran

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

# فهرست مطالب


مقدمه ..... ۶

۱) 8086 Microprocessor ..... ۸

۱-۱) طرح پایه ..... ۸

۱-۲) منبع تغذیه ..... ۸

۱-۳) اتصالات پایه ..... ۹

۲- LCD 16*2 Matrix ..... ۱۱

۲-۱ چیدمان پایه های ماژول LCD ..... ۱۱

۲-۲) اتصالات پایه ..... ۱۲

۲-۳) ثبات ها ..... ۱۳

۲-۳-۲) پایه Register Select ..... ۱۳

۲-۴) کد های دستورالعملی lcd ..... ۱۴

۳- آشنایی با رابط قابل برنامه ریزی 8255A ..... ۱۵

۳-۱) پایه های کنترلی ..... ۱۶

۳-۲) روش برنامهریزی ۸۲۵۵: ..... ۱۸

۳-۲-۱) عملیات حالت صفر ..... ۱۹

۳-۲-۲) عملیات خروجی استروب شده (حالت ۱) ..... ۲۰

۳-۲-۳) تعاریف درگاه C برای خروجی استروب شده حالت ۱ ..... ۲۱

۳-۲-۴) عملیات دو طرفه (حالت ۲) ..... ۲۲

۳-۲-۵) حالت های مختلف برنامه ریزی 8255 : ..... ۲۳

۳-۲-۶) روش استفاده از 8255A ..... ۲۵

۴- آشنایی octal d-type transparent latch ..... ۲۶

۴-۱) اتصالات پایه ها ..... ۲۷

۴-۲) شرح عملکرد لچ ..... ۲۷

اتصال ماژول ها ..... ۲۹

۱) 8086 Microprocessor ..... ۳۰

۲) Latch ..... ۳۰

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD


۳) 8255A Programmable Peripheral Interface ..... ۳۰

۴) LCD ..... ۳۰

کد های برنامه ..... ۳۳

تحلیل کد: ..... ۳۴

پیشنهاد ..... ۳۸

بحث و نتیجه گیری ..... ۳۹

منابع: ..... ۴۰

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## مقدمه

در این پروژه قصد داریم با استفاده از ریزپردازنده ۸۰۸۶ و یک نمایشگر متنی استاندار ۲×۱۶ یک متن را چاپ کنیم که این عمل با استفاده از نرم افزار Proteus شبیه سازی کنیم. در طول این مقاله با پایه ها و عملکرد ریزپردازنده ۸۰۸۶ و نحوه اتصال آن به یک LCD متنی ۲×۱۶ با استفاده از یک ماژول I/O Control از مدل intel 8255A و آدرس دهی به پورت های آن از طریق یک octal D-Type Transparent latch – 3 state از نوع 74hc373 آشنا خواهیم شد همچنین کد برنامه که به زبان اسمبلی نوشته شده را بررسی می کنیم.

در بخش نخست هر یک از ماژول های استفاده شده در این پروژه یعنی:

۱) Intel 8086 microprocessor
۲) LCD 16*2 Display
۳) Intel 8255A Programmable Peripheral Interface (I/O Control Device)
٤) Octal D-type Transparent Latch – 3state (74hc373)

آشنا خواهیم شد و سعی داریم ساختار آن های را تا حدی مورد بررسی قرار دهیم. در بخش دوم نحوه اتصال این ماژول ها را به یکدیگر و شبیه سازی مدار در نرم افزار Proteus را مورد بررسی قرار می دهیم. در بخش سوم کد های برنامه را بررسی می کنیم و در نهایت در آخرین بخش درباره پروژه نتیجه گیری و بحث خواهیم کرد.

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

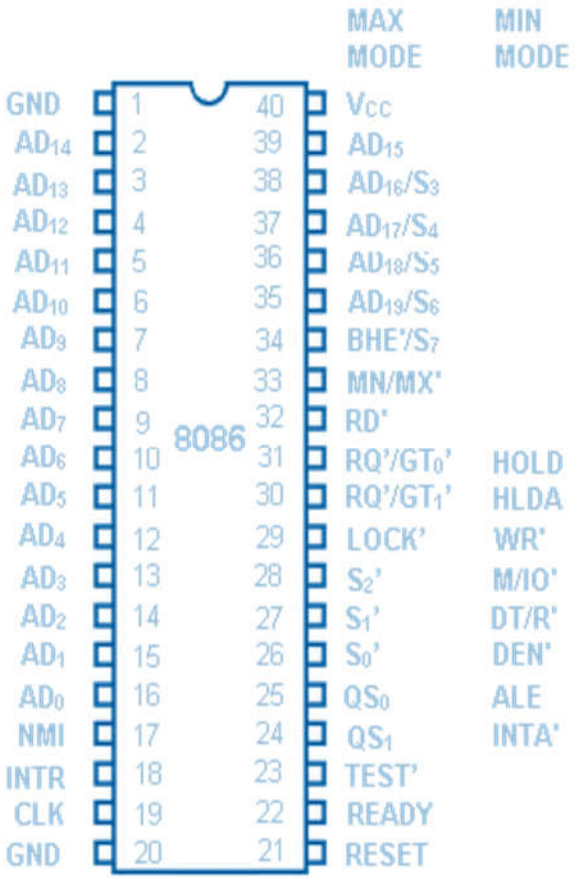
# بخش نخست

# آشنایی با ماژول ها

## ۱) 8086 MICROPROCESSOR

# ۱-۱) طرح پایه

۸۰۸۶ یک پردازنده ۱۶ بیتی با گذرگاه آدرس ۲۰ بیتی و گذرگاه داده ۱۶ بیتی مالتی پلکس شده با گذرگاه آدرس است.





# ۱-۲) منبع تغذیه

ریزپردازنده ۸۰۸۶ به ولتاژ تغذیه +5.0V با خطای $10\%V$ نیاز دارد و 360mA جریان می کشد.

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## ۳-۱) اتصالات پایه

| | |
|---|---|
| AD15-AD0 | خطوط گذرگاه آدرس/داده گذرگاه های مولتی پلکس شده برای داده و آدرس‌اند، و حاوی ۸ بیت سمت راست آدرس حافظه یا شماره پورت I/O در ازاء فعال بودن پایه ALE، و حاوی داده به هنگام غیر فعال بودن پایه ALE است. این پایه ها در هنگام تصدیق hold در وضعیت امپدانس بالا می باشند. |
| AD19/S6-A16/S3 | بیت های گذرگاه آدرس/وضعیت مولتی پلکس شده اند تا سیگنال های آدرس A16-A19 و نیز بیت های وضعیت S3-S6 فراهم شوند، این پایه ها هم در زمان تصدیق hold در امپدانس بالا هستند. |
| RD | هرگاه سیگنال read (خواندن) در یک منطق ۰ باشد داده از سمت حافظه یا دستگاه I/O متصل به سیستم دریافتی است. این پایه در زمان تصدیق hold در حالت امپدانس شناور است. |
| READY | ورودی READY برای تزریق حالات انتظار در زمان بندی ریزپردازنده کنترل می شود اگر پایه READY در سطح منطق ۰ واقع شود ریزپردازنده وارد یک حالت انتظار شده و بیکار باقی می ماند اگر پایه READY در منطق ۱ باشد. روی عملکرد پردازنده تاثیری ندارد. |
| INTR | درخواست وقفه - برای یک تقاضا وقفه سخت افزاری به کار می رود اگر INTR در سطح بالا به هنگام IF=۱ نگهداری شود ۸۰۸۶/۸۰۸۸ پس از تکمیل اجرای دستور خارجی وارد یک سیکل تصدیق وقفه می گردد. |
| TEST | پایه Test یک ورودی است که بوسیله دستور WAIT تست می شود. اگر TEST در منطق ۰ باشد.دستور WAIT همچون یک NOP عمل می کند و اگر TEST در منطق ۱ باشد به انتظار می ماند تا TEST به منطق ۰ برود. |
| NMI | ورودی وقفه پوشش ناپذیر مشابه INTR است. با این تفاوت که وقفه NMI بیت پرچم IF برای منطق ۱ چک نمی کند. اگر NMI فعال باشد. این ورودی وقفه بردار وقفه ۲ را بکار می برد. |
| RESET | ورودی reset سبب می شود تا ریزپردازنده ریست (بازنشانی) شود به شرطی که این پایه برای چهار پالس ساعت در سطح بالا نگهداری شود. هر وقت ۸۰۸۶ ریست شود.شروع به اجرای دستورات واقع در مکان حافظه FFFF0H نموده و برای غیرفعال کردن وقفه های آینده بیت پرچم IF را غیرفعال می نماید. |
| CLK | پایه clock سیگنال زمان بندی مبنا را برای ریزپردازنده فراهم می سازد. سیگنال clock یا ساعت باید سیکل یا چرخه کاری ۳۳٪ داشته باشد (برای مدت یک سوم دوره ساعت درسطح بالا و در دو سوم در سطح پایین باشد) تا زمان بندی داخلی صحیحی برای ۸۰۸۶ فراهم گردد. |
| Vcc | این ورودی منبع تغذیه را برای ریزپردازنده فراهم می کند. |
| GNB | سیم بازگشت منبع تغذیه است. توجه کنید که ریزپردازنده ۸۰۸۶ دو پایه با برچسب GNB دارد که برای عملکرد صحیح باید هردو به زمین وصل شوند. |
| MN/MX | پایه مد مینیمم ۸۰۸۶ با اتصال پایه MN/MX مستقیما به +5.0V وصل می گردد این پایه را از طریق مقاومت بالاکش متصل نکنید. در غیر اینصورت درست کار نخواهد کرد. |

| | |
|---|---|
| IO/M یا M/IO | این پایه مشخص می کند که گذرگاه آدرس ریزپردازنده حاوی یک آدرس حافظه یا آدرس یک پورت است.این پایه در هنگام تصدیق hold در امپدانس بالا خواهد بود. |
| WR | خط write یک آگاه‌گر است که مشخص می کند ۸۰۸۶ داده را برای حافظه یا I/O به خارج می فرستد.در زمانی که WR در منطق صفر است گذرگاه داده دارای داده معتبر برای حافظه یا I/O است. این پایه در هنگام تصدیق hold به امپدانس بالا شناور است. |
| INTA | سیگنال تصدیق وقفه INTA پاسخی به پایه ورودی INTR است. |
| ALE | سیگنال لچ آدرس فعال (ALE) نشان می دهد که گذرگاه آدرس/داده حاوی اطلاعات آدرس است. این آدرس می تواند آدرس یک حافظه یا شماره پورت I/O باشد. توجه کنید که سیگنال ALE در حین تصدیق HOLD، شناور نیست. |
| DT/R | سیگنال ارسال/دریافت DT/R داده نشان می دهد که گذرگاه داده ریزپردازنده داده را ارسال (DT/R = ۱) یا دریافت (DT/R = 0) می نماید. این سیگنال برای استفاده از بافر گذرگاه داده بکار می رود. |
| DEN | گذرگاه داده فعال (DEN) بافر های گذرگاه داده خارجی را فعال می کند. |
| HOLD | سیگنال HOLD در ۱ منطقی باشد ریزپردازنده اجرای نرم افزار را متوقف کرده و در منطق ۰ باشد ریزپردازنده اجرای معمولی نرم افزار را انجام دهد. |
| HLDA | تصدیق HODL یا (Hold Acknowledge) نشان می دهد که ریزپردازنده ۸۰۸۶ وارد حالت hold شده است. |
| SS0 | خط وضعیت SS۰ معادل با پایه S۰ در مد ماکزیمم عملکرد ریزپردازنده است. این سیگنال با IO/M یا DT/R برای دیکد کار سیکل جاری گذرگاه ترکیب می شود. |
| پایه مد حداکثر | برای دستیابی به مد حداکثر جه کار با ریزپردازنده های کمکی پایه MN/MX را به زمین وصل نمایید |
| RQ/GT1.RQ/GT0 | پایه های تقاضا/واگذاری دستیابی مستقیم به حافظه (DMA) را در عمل مد ماکزیمم تقاضا می کنند. این خطوط در طرفه اند و برای هر دو عمل تقاضا و واگذاری DMA به کار می رود. |
| LOCK | خروجی lock برای قفل کردن وسایل جانبی سیستم بکار می رود این پایه با پیشوند lock در هر دستورالعمل فعال می گردد. |
| QS1 و QS0 | بیت های وضعیت صف، وضعیت صف دستور داخلی را نشان می دهد. این پایه ها برای دستیابی به وسیله پردازنده کمکی استفاده می شود جدول ۷-۹ را برای مشاهده بیت های وضعیت صف ملاحظه کنید. |

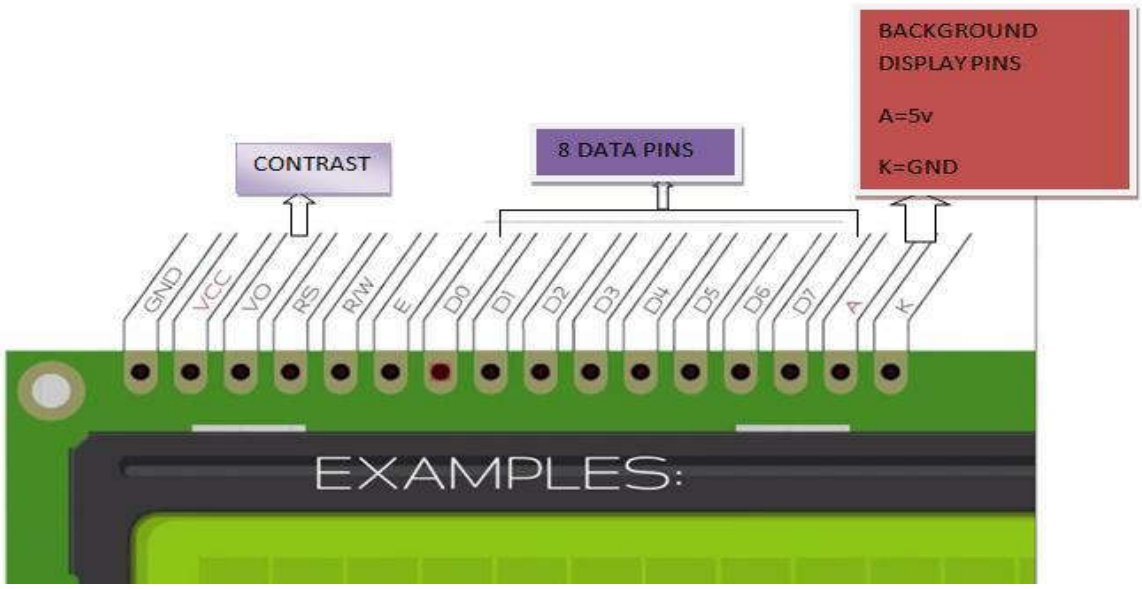
**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## LCD 16*2 MATRIX -۲

نمایشگر های LCD تقریبا در اطراف ما به فراوانی استفاده شده اند از کامپیوتر ها گرفته تا پرینتر ها، موبایل ها و ... .

در این پروژه از یک نمایشگر LCD ۲×۱۶ استفاده خواهد شد، که قابلیت نمایش ۱۶ کاراکتر در خط را دارا می باشد، و هر کاراکتر از ۵×۷ پیکسل ساخته شده است.


## ۲-۱ چیدمان پایه های ماژول LCD



**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## ۲-۲) اتصالات پایه

جدول زیر پایه های LCD را با اختصار توضیح می دهد.

| شماره پایه | نماد | توضیحات | عملکرد |
|---|---|---|---|
| ۱ | **VSS** | **GROUND** | **0V(GND)** |
| 2 | **VCC** | منبع تغذیه برای مدار منطقی | **+5V** |
| 3 | **VEE** | تنظیم کنتراست صفحه نمایش | |
| 4 | **RS** | **INSTRUCTION/DATA Register-Selection** | **RS = 0 : INSTR Register**<br>**RS = 1 : DATA Register** |
| 5 | **R/W** | **READ/WRITE Selection** | **R/W = 0 : Register WRITE**<br>**R/W = 1 : Register READ** |
| 6 | **E** | **ENABLE Signal** | **Sends data to data pins when a high to low pulse applied** |
| 7 | **D0** | **DATA INPUT/OUTPUT LINES** | |
| 8 | **D1** | | |
| 9 | **D2** | | |
| 10 | **D3** | | |
| 11 | **D4** | | |
| 12 | **D5** | | |
| 13 | **D6** | | |
| 14 | **D7** | | |
| 15 | **LED+** | **Backlight VCC (5V)** | **+5V** |
| 16 | **LED-** | **Backlight Ground (0V)** | **0V** |

## ۲-۳) ثبات ها

دو ریجستر اصلی در ماژول LCD وجود دارد.

- Instruction Register
- Data Register

ثبات دستور Instruction Register (IR) یک ثبات Write Only می باشد و برای ذخیره کد های دستور مانند "Display Clear" و "Cursor Shift" استفاده خواهد شد.

ثبات داده (Data register) یک ثبات read/write است که برای ذخیره موقت داده استفاده می شوند.

### ۲-۳-۲) پایه REGISTER SELECT

برای تعویض ثبات ها استفاده خواهد شد اگر RS=0 ثبات دستور و اگر RS=1 ثبات داده انتخاب خواهد شد.

| Operation | R/W | RS |
|---|---|---|
| IR write, internal operation (Display Clear etc.) | 0 | 0 |
| Busy flag (DB7) and Address Counter (DB0 ~ DB6) read | 1 | 0 |
| DR Write, Internal Operation (DR ~ DD RAM or CG RAM) | 0 | 1 |
| DR Read, Internal Operation (DD RAM or CG RAM) | 1 | 1 |

اگر بخواهیم یک کاراکتر را در صفحه نمایش نشان دهیم باید از ASCII استفاده کنیم.

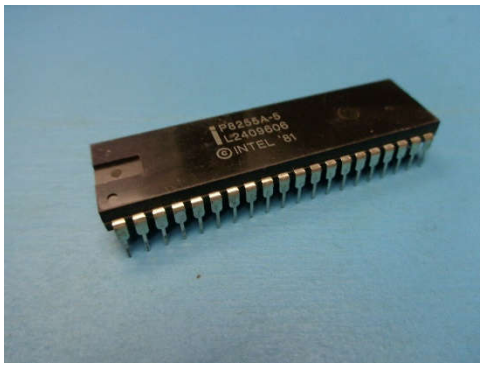
**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## ۴-۲) کد های دستورالعملی LCD

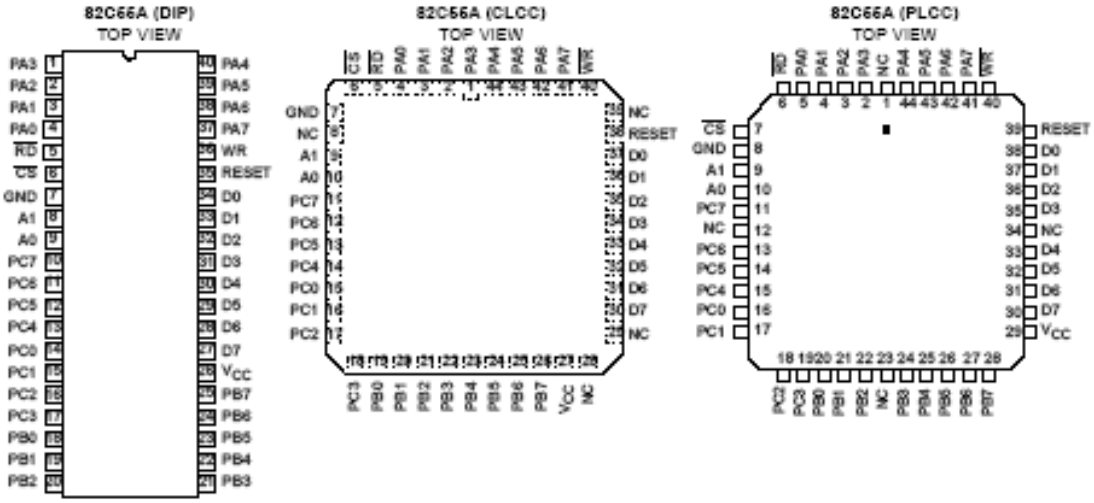
| Sr.No | Hex Code | دستورات ثبات دستورالعمل LCD |
|---|---|---|
| 1 | 01H | پاک کردن صفحه |
| 2 | 02H | بازگشت به خانه |
| 3 | 04H | عقب بردن نشانگر (جابجایی به چپ) |
| 4 | 06H | جلو بردن نشانگر (جابجایی به راست) |
| 5 | 05H | جابه جایی صفحه به راست |
| 6 | 07H | جابه جایی صفحه به چپ |
| 7 | 08H | صفحه روشن، نشانگر روشن |
| 8 | 0AH | صفحه خاموش، نشانگر روشن |
| 9 | 0CH | صفحه روشن، نشانگر خاموش |
| 10 | 0EH | صفحه روشن، نشانگر چشمک زننده |
| 11 | 0FH | صفحه روشن، نشانگر چشمک زننده |
| 12 | 10H | جابجایی موقعیت نشانگر به چپ |
| 13 | 14H | جابجایی موقعیت نشانگر به راست |
| 14 | 18H | جابجایی تمامی صفحه به چپ |
| 15 | 1CH | جابجایی تمامی صفحه به راست |
| 16 | 80H | اجبار نشانگر که از خط اول شروع کند |
| 17 | C0H | اجبار نشانگر که از خط دوم شروع کند |
| 18 | 38H | 2 lines and 5×7 matrix |

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## ۳- آشنایی با رابط قابل برنامه ریزی 8255A




intel 8255A یک I/O همه منظوره قابل برنامه ریزی است، این تراشه، ارزان قیمت و بسیار متداول است و به منظور استفاده با پردازنده های intel طراحی شده است. این تراشه دارای ۲۴ پایه برای I/O است که در گروه های ۱۲ پایه ای قابل برنامه ریزی هستند. هر گروهی می تواند در حالت های مجزا از جمله I/O ساده، IO استروب شده و I/O دو طرفه، کار کند، ۸۲۵۵ قادر است هر وسیله I/O موازی سازگار با TTL را به آسانی به ریز پردازنده Z80 ارتباط دهد.




وضعیت پایه ها و شمای کلی 8255A در شکل ۱ نشان داده شده است. پایه ها نشان می دهد که ۸۲۵۵ سه درگاه I/O (A,B,C) دارد که در دو گروه ۱۲ پایه ای برنامه ریزی می شوند. گروه A از درگاه A (PA0-PA7) و نیمه بالایی درگاه C ( PC4-PC7) و گروه B ( PB0-PB7) و نیمه پائینی درگاه C (PC0-PC3) تشکیل می شود.

انتخاب درگاه بوسیله پایه CS و پایه های آدرس A0 و A1 ، انجام می شود. که با هم بطور داخلی یک درگاه I/O یا ثبات فرمان را بر می گزینند.

ورودی RESET باعث می شود که درگاه I/O به عنوان درگاه ورودی برنامه ریزی شود تا آسیبی به مدارهای متصل به پایه های درگاه نرسد. اگر این درگاه بخواهد به عنوان درگاه خروجی بکار رود. 8255A را باید از طریق درگاه فرمان برنامه ریزی نمود تا به عنوان خروجی عمل کنند.

## ۱-۳) پایه های کنترلی

### CS :

با اعمال ولتاژ low به این پایه ارتباط بین چیپ و ریزپردازنده و برای انتخاب پورت از طریق پایه هایA0 وA1 امکان پذیر خواهد شد.

| CS | A1 | A0 | نتیجه |
|---|---|---|---|
| ۰ | 0 | 0 | PORT A |
| ۰ | 0 | 1 | PORT B |
| ۰ | 1 | 0 | PORT C |
| ۰ | 1 | 1 | Control Register |
| ۱ | X | X | No Selection |

### WR :

برای عمل write استفاده خواهد شد. سیگنال کنترلی عمل نوشتن (write) را فعال خواهد نمود وقتی این سیگنال در حالت low قرار بگیرد. ریز پردازنده عمل نوشتن را روی پورت I/O یا control register انجام خواهد داد.

### RD :

برای عمل خواندن استفاده خواهد شد این سیگنال عمل خواندن (read) را انجام خواهد داد اگر این سیگنال در حالت low باشد ریزپردازنده داده را از پورت انتخاب شده می خواند.

### RESET :

این پایه یک سیگنال از نوع high است که در صورت فعال شدن control register را پاک خواهد نمود و همه پورت های در حالت input قرار خواهد داد.

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## A0 – A1 :

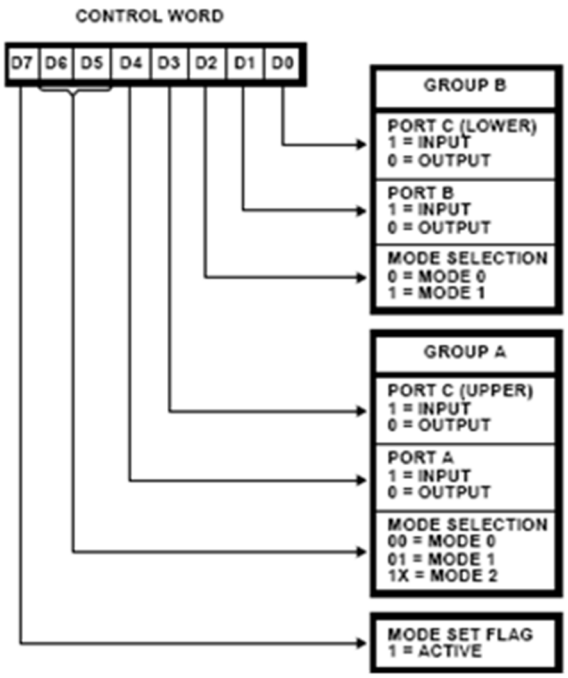
این سیگنال ها با سیگنال های RD و WR کار خواهند کرد جدول زیر انواع سیگنال ها با نتیجه شان را نشان خواهد داد.

| A1 | A0 | RD | WR | CS | نتیجه |
|---|---|---|---|---|---|
| ۰ | ۰ | ۰ | ۱ | ۰ | **INPUT OPERATION**<br>PORT A -> DATA BUS |
| ۰ | ۱ | ۰ | ۱ | ۰ | PORT B -> DATA BUS |
| ۱ | ۰ | ۰ | ۱ | ۰ | PORT C -> DATA BUS |
| ۰ | ۰ | ۱ | ۰ | ۰ | **OUTPUT OPERATION**<br>PORT A <- DATA BUS |
| ۰ | ۱ | ۱ | ۰ | ۰ | PORT B <- DATA BUS |
| ۱ | ۰ | ۱ | ۰ | ۰ | PORT C <- DATA BUS |
| ۱ | ۱ | ۱ | ۰ | ۰ | PORT D <- DATA BUS |

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## ۲-۳) روش برنامه‌ریزی ۸۲۵۵:

برنامه ریزی ۸۲۵۵ کار نسبتا ساده ای است زیرا تراشه فقط دارای ۲ ثبات فرمان داخلی است که برنامه ریزی می شوند. شکل زیر ثبات فرمان اصلی را نشان می دهد که به استفاده کننده اجازه می دهد تا درگاه I/O گروههای A و B را به طور جداگاه برنامه ریزی نماید.




این ثبات فرمان را می توان با ارسال یک به محل آخرین بیت سمت چپ آن انتخاب نمود .

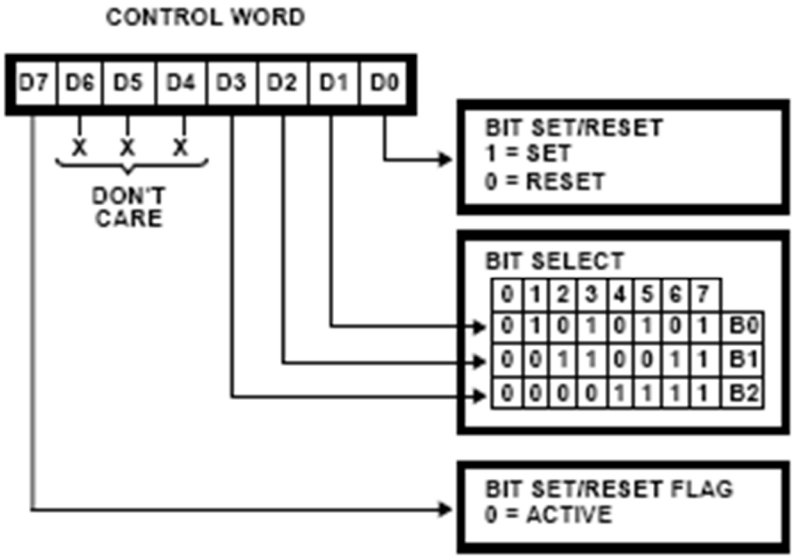
پایه های گروه B به عنوان پایه های ورودی و یا به عنوان پایه های خروجی قابل برنامه ریزی هستند. گروه B همچنین می تواند در یکی از حالت های ۰ یا ۱ کار کند.

حالت ۰ مربوط به عملیات *ورودی یا خروجی* ساده و حالت ۱ مربوط به عملیات ورودی-خروجی استروب شده است در این حالت درگاه B به عنوان ورودی خروجی به همراه سیگنالهای کنترلی انتقال اطلاعات (دست دادن) که بوسیله درگاه C تامین می شود، عمل کند.

پایه های گروه A به عنوان پایه های ورودی یا خروجی برنامه ریزی می شوندکه در یکی از حالات ۰ و ۱ و ۲ کار کنند. در همه حالات به جز حالت ۰ درگاه A به عنوان یک درگاه ورودی یا خروجی برنامه ریزی می شود که درگاه C به عنوان سیگنال های کنترل انتقال اطلاعات برای آن عمل می کند. حالات ۰و۱ مشابه حالات در گروه B هستند. و حالت ۲ درگاه A را به عنوان یک درگاه I/O دو طرفه بکار می گیرد که درگاه C تامین کننده سیگنال های کنترل انتقال اطلاعات برای آن است.

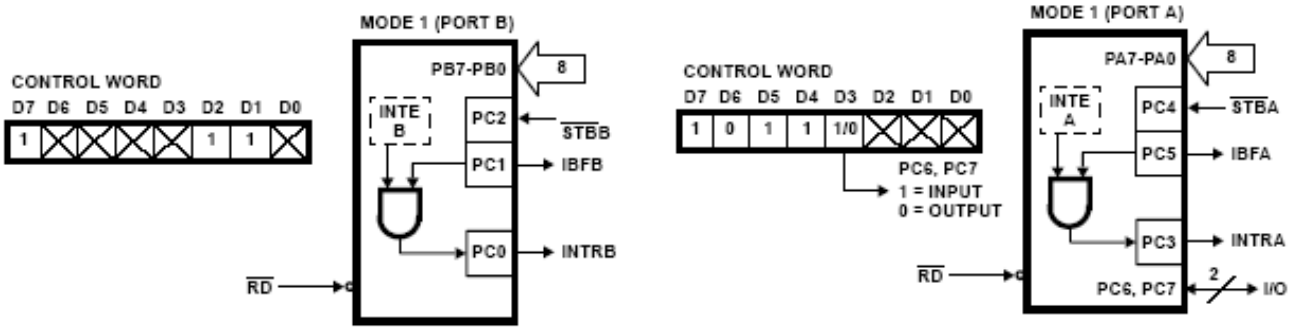
اگر یک صفر در آخرین بیت ثبات فرمان (از سمت چپ) قرار داده شود بیت های درگاه C را میتوان به طور مجزا آدرس داد، به نحوی که در حالات ۱و۲ می توان هر یک از آنها را نشاند و یا پاک نمود. در حالت ۰، اطلاعات مستقیما به درگاهC فرستاده می شوند.

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

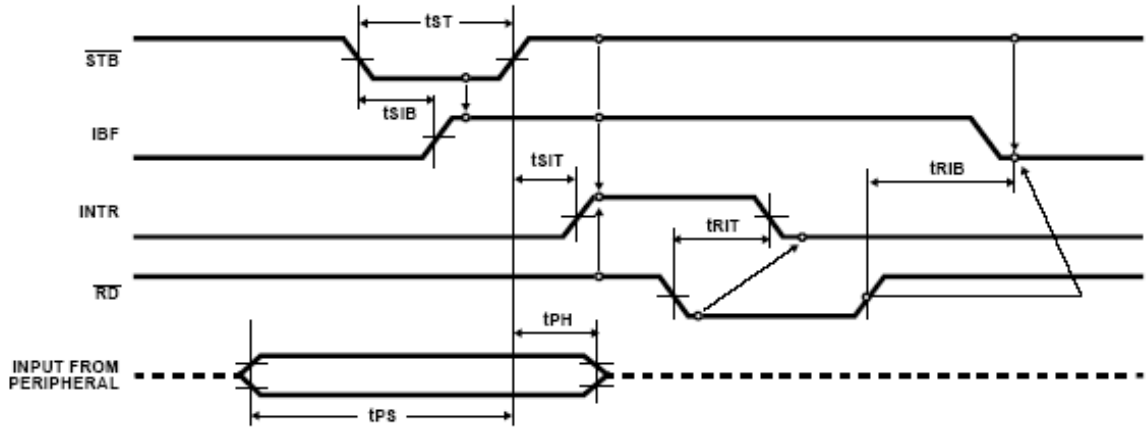



## ۱-۲-۳) عملیات حالت صفر

در حالت صفر اطلاعات را می توان به درگاه A,B,C فرستاد و در آنجا از آنها تا دستورالعمل بعدی (OUT) نگهداری می شود. حالت ۱ ورودی استروب شده در ۸۲۵۵ با بکار گیری پایه های درگاه C بعنوان سیگنال های کنترل انتقال اطلاعات، عملیات دست تکانی را برای درگاه B و A انتخاب می کند. شکل زیر ساختار داخلی ۸۲۵۵ را برای عملیات ورودی استروب شده و همچنین سیگنالهای زمانبندی مربوطه را در حالت ۱ نشان می دهد.



**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD




$\overline{STB}$[1]- **استروب :** این ورودی برای وارد کردن اطلاعات بداخل قفل درگاه A یا B بکار می رود. این اطلاعات تا زمانیکه بوسیله یک دستورالعمل IN بداخل ریز پردازنده برده نشود در آنجا نگهداری می شود.

**IBF = (INPUT BUFFER FULL) :** این خروجی نشان می دهد که قفل ورودی ، اطلاعاتی برای ریز پردازنده در بر دارد. سیگنال $\overline{STB}$ این پایه خروجی را می نشاند و دستورالعمل IN آنرا پاک می کند .

**INTR[2]- (در خواست وقفه) :** این خروجی برای درخواست نمودن یک وقفه بکار میرود. وقتی سیگنال $\overline{STB}$ به سطح منطقی ۱ می رود این خروجی ۱ می شود و با اجرا شدن دستورالعمل IN پاک می گردد .

**INTE[3]- فعال کننده وقفه :** این فعال کننده یک ورودی یا خروجی نیست، بلکه یک بیت داخلی است که بوسیله ثبات فرمان BSR[4] برنامه ریزی می شود.INTE A به عنوان PC4 و INTE B به عنوان PC2 برنامه ریزی می شود.

در این حالت PC6,PC7 به عنوان I/O همه منظوره در عملیات ورودی استروب شده حالت ۱ هستند. این پایه ها وقتی به عنوان پایه های خروجی به کار می روند بوسیله BSR کنترل می شوند، و وقتی به عنوان پایه های ورودی بکار می روند، از طریق درگاه C خوانده می شوند.

## ۲-۲-۳) عملیات خروجی استروب شده (حالت ۱)

ساختار داخلی ۸۲۵۵ برای عملیات خروجی استروب شده، و سگنال های زمانبندی مربوطه را نشان می دهد.

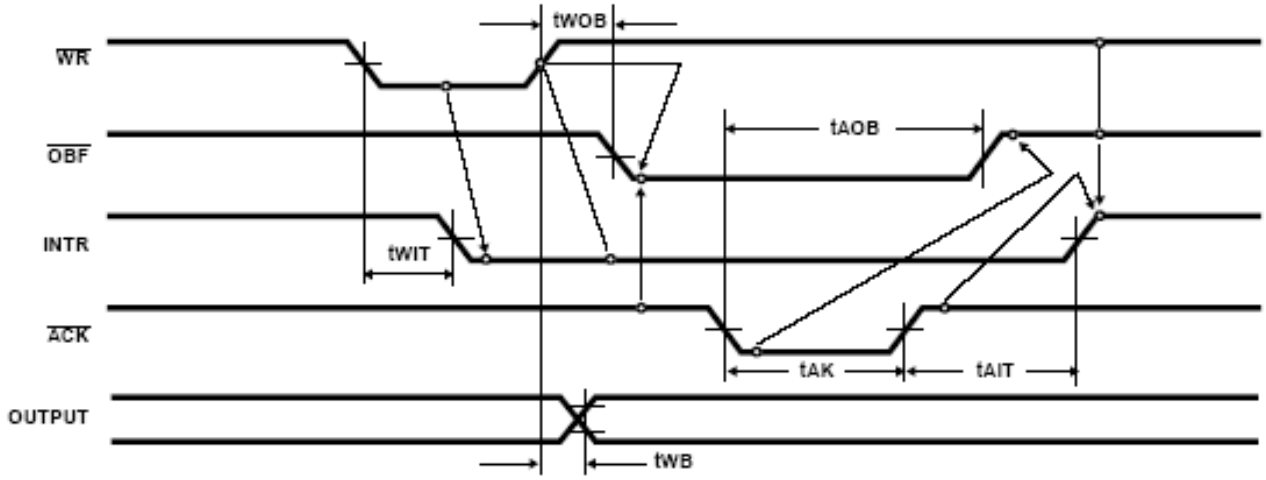
[1] **STROBE**

[2] **INTERUPT REQUEST**

[3] **INTRUPT ENABLE**

[4] **BIT DET RESET**

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD




## ۳-۲-۳) تعاریف درگاه C برای خروجی استروب شده حالت ۱

$\overline{OBF}$[5] – **بافر خروجی پر است :** هر گاه اطلاعاتی به قفل درگاه A یا B فرستاده شود این خروجی پایین می رود. این سیگنال با برگشتن پالس $\overline{ACK}$[6] از یک وسیله خارجی نشانده می شود و وقتی اطلاعات با یک دستور العمل OUT در درگاه نوشته می شود، پاک می گردد.

$\overline{ACK}$ – **ورودی اعلام دریافت اطلاعات :** سیگنالی که باعث می شود پایه $\overline{OBF}$ به سطح منطقی ۱ برگردد. $\overline{ACK}$ پاسهای از یک وسیله خارجی است که نشان می دهد وسیله اطلاعات را از ۸۲۵۵ دریافت کرده است.

**INTR– در خواست وقفه :** هرگاه یک وسیله خارجی، دریافت اطلاعات خروجی را اعلام کند ( $\overline{ACK}$ ) از این سیکنال می توان برای دادن وقفه به ریز پردازنده استفاده کرد.

**INTE فعال کننده وقفه :** این فعال کننده یک ورودی یا خروجی نیست، بلکه یک بیت داخلی است که بوسیله ثبات فرمان BSR[7] برنامه ریزی می شود.

INTE A به عنوان بیت PC6 و INTE B به عنوان PC2 برنامه ریزی می شود.

[5] OUTPUT BUFFER FULL

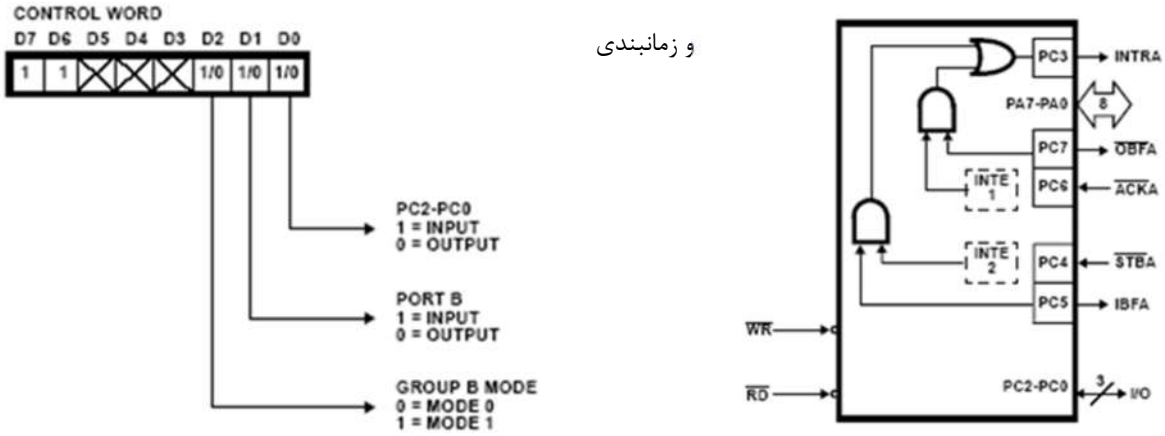
[6] ACKNOLEDGE MANT

[7] BIT DET RESET

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

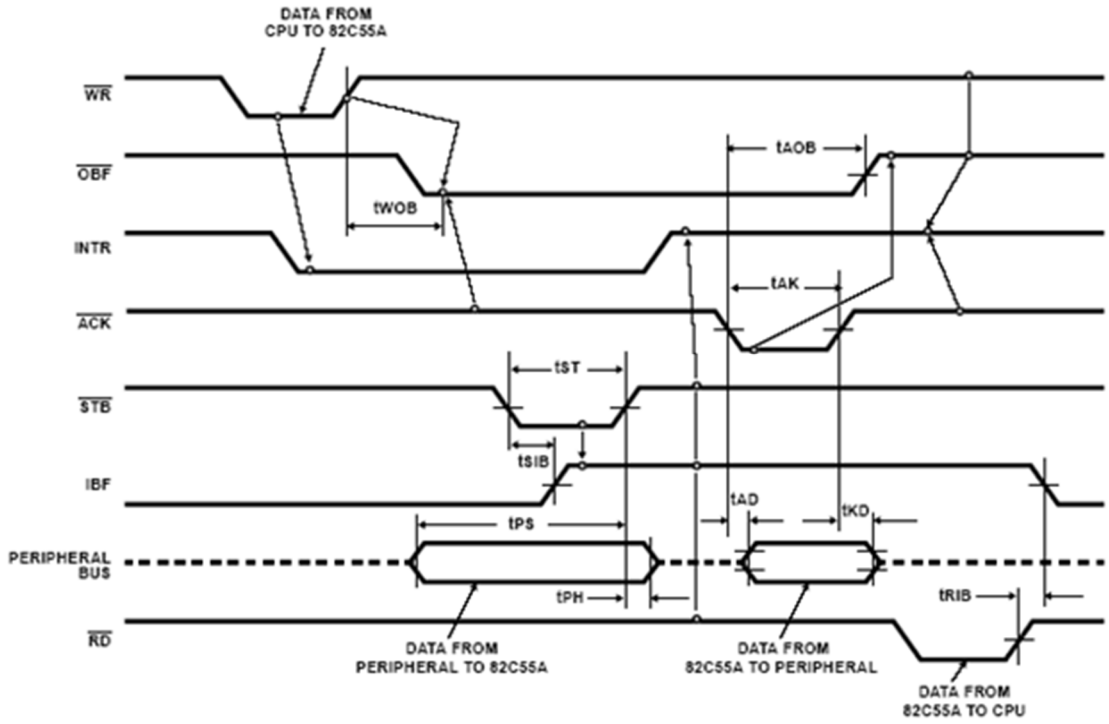
## ۴-۲-۳) عملیات دو طرفه (حالت ۲)

عملیات دو طرفه فقط برای پایه های گروه A انجام می شود. درگاه A به صورت یک درگاه دو طرفه در می آید که امکان ارسال و دریافت اطلاعات را بر روی هشت سیم مشترک بوجود می آورد . اطلاعات دو طرفه در ارتباط دهی دو کامپیوتر مفید است شکل زیر ساختار داخلی و سیگنال های زمانبندی را برای عملیات دو طرفه درگاه A نشان می دهد.در این حالت نیز سیگنالهای کنترلی از تعاریف در ۲ حالت قبل پیروی میکنند.



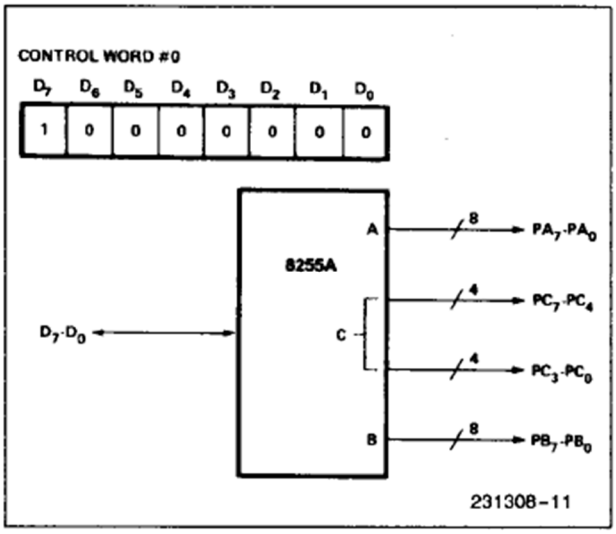

و زمانبندی

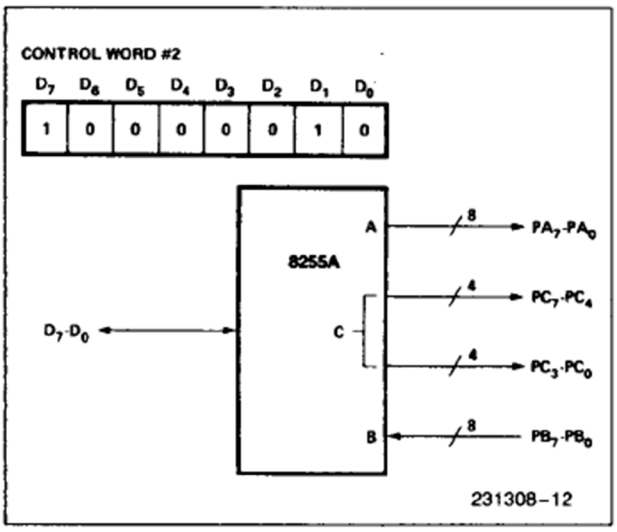


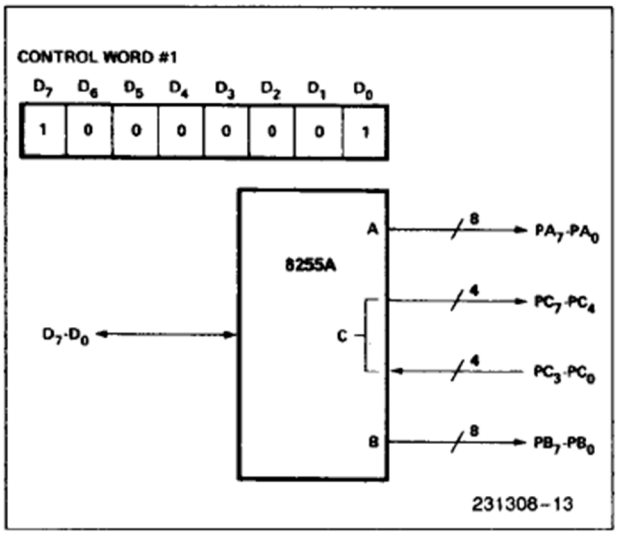



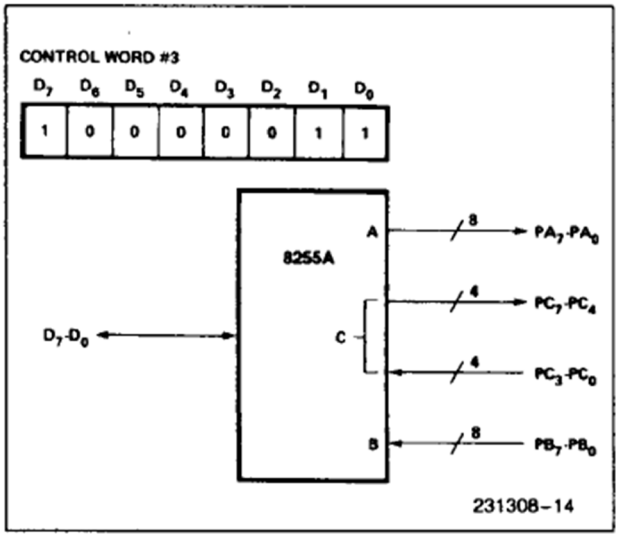
**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

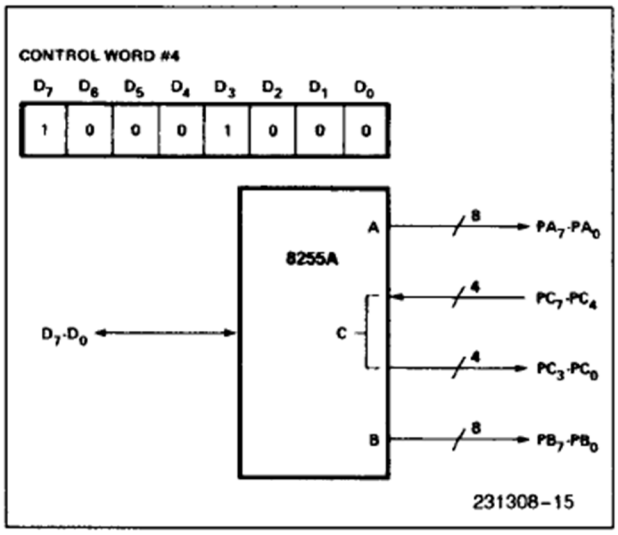
## ۵-۲-۳) حالت های مختلف برنامه ریزی 8255 :

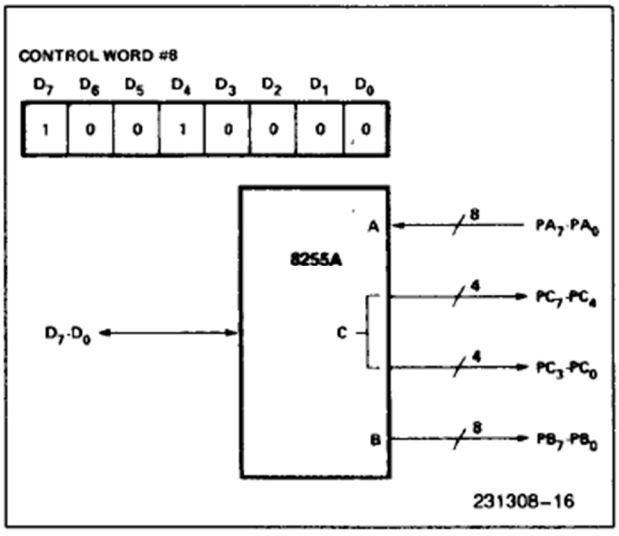


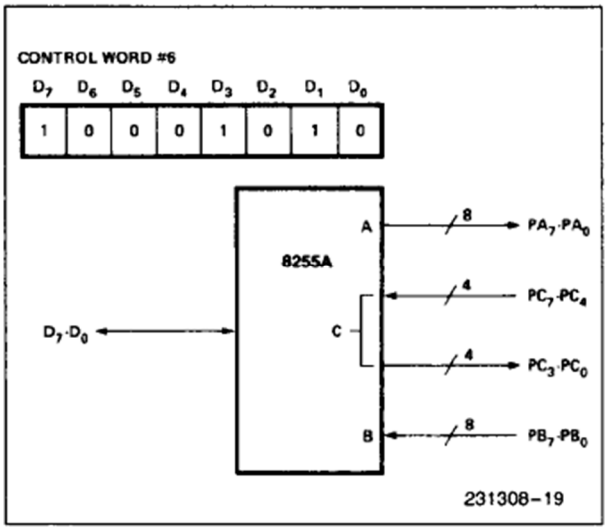



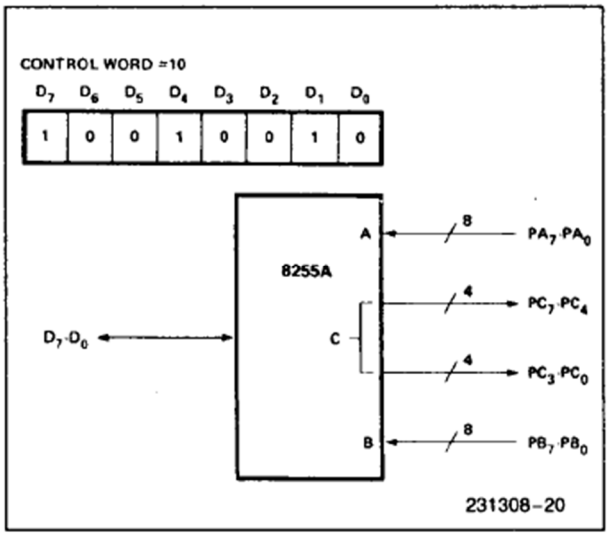



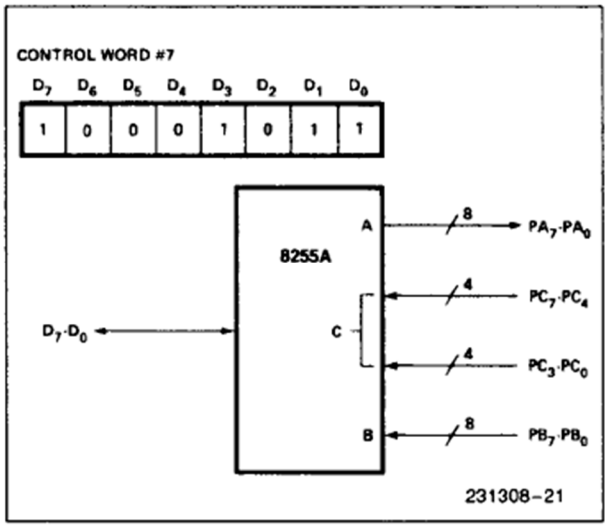



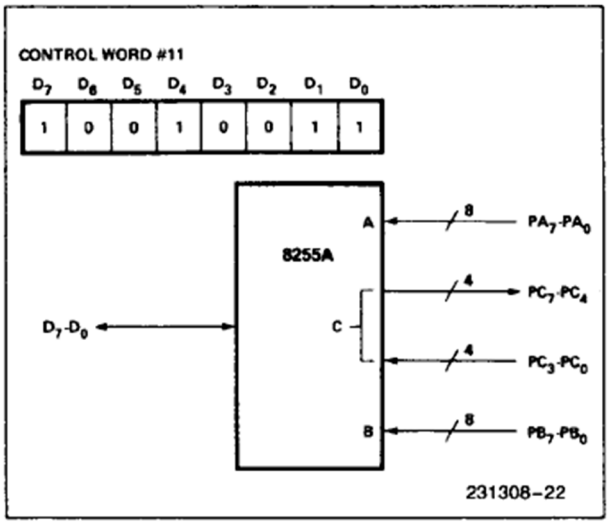



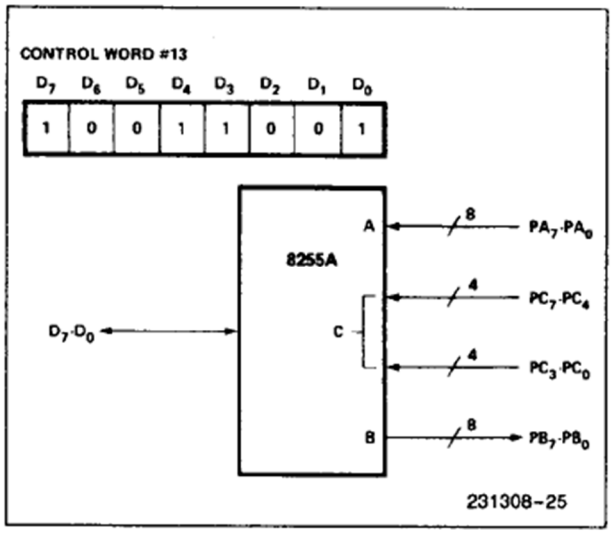
**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

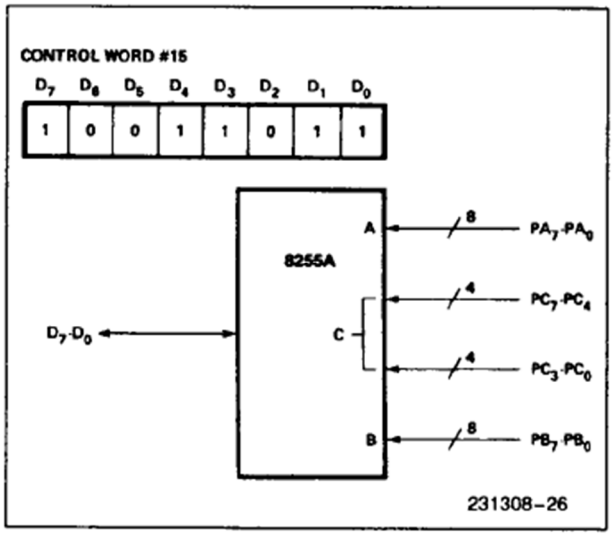


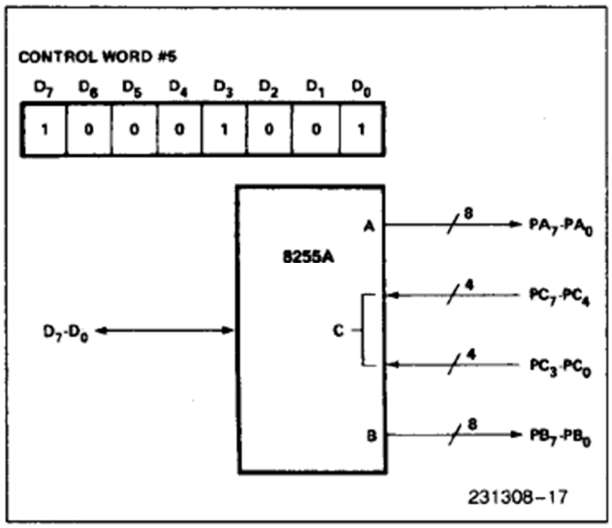



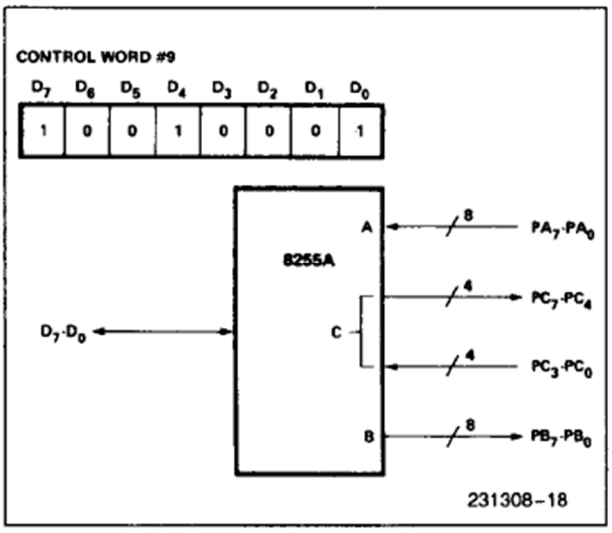



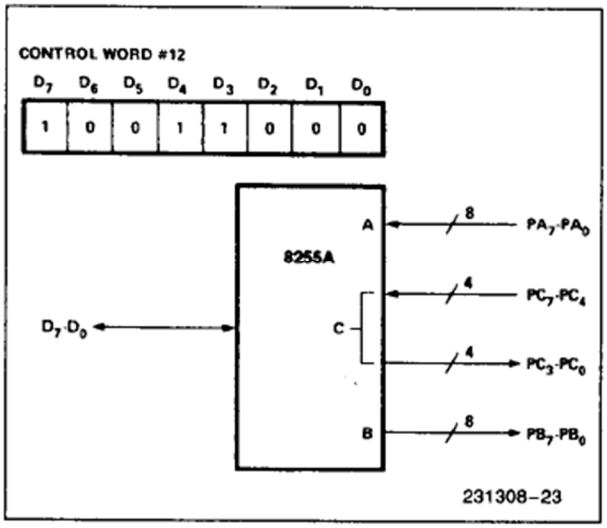



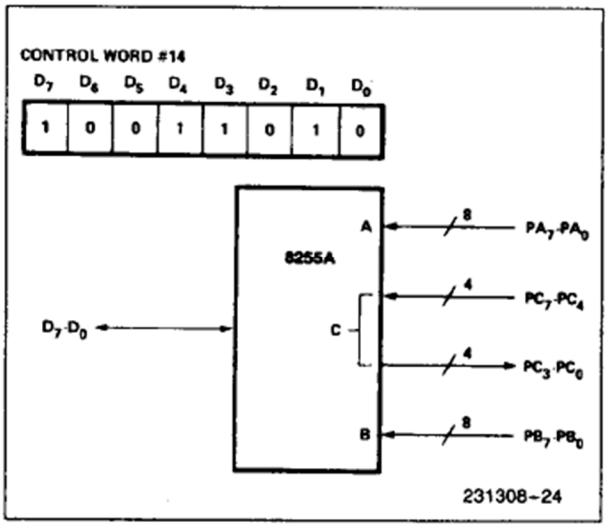



**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD




## ۶-۲-۳) روش استفاده از 8255A

برای راه اندازی و استفاده از ۸۲۵۵ باید VCC را به +۵ ولت و GND را به ۰ ولت متصل می کنیم.CS باید در سطح منطقی صفر (۰) باشد. حالIC آماده فعالیت است. ابتدا باید آن را برنامه ریزی کرد. برای این کار A0 و A1 را ۱ می کنیم و کلمه کنترلی را روی گذرگاه قرار می دهیم. سپس کلید WR را به سطح منطقی صفر آورده و دوباره به ۱ باز می گردانیم.

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## ۴- آشنایی OCTAL D-TYPE TRANSPARENT LATCH

**: 74HC373**

*74HCT373* یک لچ ۳ حالته از نوع D با هشت خروجی است. ورودی های این دستگاه از ویژگی های فعال کردن لچ (LE) و فعال کردن خروجی (OE) برخوردار هستند. وقتی LE در حالت high است، داده های ورودی وارد لچ می شوند. در این شرایط لچ مانند یه کلنال عبوری عمل می کند ، هر بار که ورودی متناظر D تغییر کند، خروجی لچ نیز تغییر می کند. وقتی LE در حالت low است، لچ ها اطلاعاتی را که در ورودی ها وجود دارد ذخیره می کنند.


## 4. Functional diagram

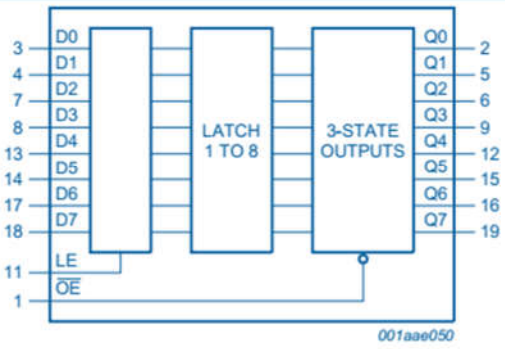



Fig. 1. Functional diagram

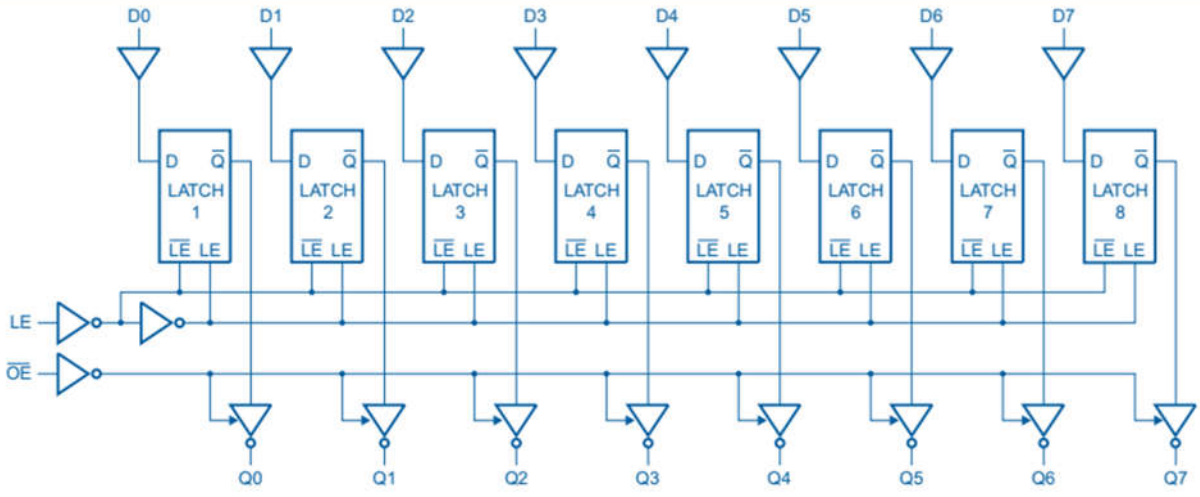



Fig. 5. Logic diagram

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## ۴-۱) اتصالات پایه ها

| Symbol | Pin | Description |
|---|---|---|
| OE’ | 1 | 3-state output enable input (active LOW) |
| Q0,Q1,Q2,Q3,Q4,Q5,Q6,Q7 | 2,5,6,9,12,15,16,19 | 3-state latch output |
| D0,D1,D2,D3,D4,D5,D6,D7 | 3,4,7,8,13,14,17,18 | Data input |
| GND | 10 | Ground (0 V) |
| LE | 11 | Latch enable input (active HIGH) |
| Vcc | 20 | Supply voltage |

## ۴-۲) شرح عملکرد لچ

Table 3. Functional description

*H = HIGH voltage level; h = HIGH voltage level one set-up time prior to the HIGH-to-LOW LE transition;*
*L = LOW voltage level; l = LOW voltage level one set-up time prior to the HIGH-to-LOW LE transition;*
*X = don't care; Z = high-impedance OFF-state.*

| Operating mode | Control | | Input | Internal latches | Output |
|---|---|---|---|---|---|
| | $\overline{OE}$ | LE | Dn | | Qn |
| Enable and read register (transparent mode) | L | H | L | L | L |
| | | | H | H | H |
| Latch and read register | L | L | l | L | L |
| | | | h | H | H |
| Latch register and disable outputs | H | X | X | X | Z |

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

# بخش دوم

# نحوه اتصال ماژول ها

# و

# شبیه سازی در PROTEUS

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

# اتصال ماژول ها

در بخش قبل با تک تک اجزای مدار آشنا شدیم و عملکرد پایه های آن را فرا گرفتیم حال در این بخش باید اجزا را به یکدیگر متصل کنیم و وارد فاز شبیه سازی شویم در ابتدا توضیح می دهیم که به کدام پایه از هریک از این ماژول ها احتیاج داریم و کدام پایه ها را باید به یکدیگر متصل کنیم و سپس مدار را در نرم افزار Proteus شبیه سازی کنیم در بخش بعدی با شروع برنامه نویسی مدار بیشتر با عملکرد مدار آشنا خواهیم شد.

## ۱) 8086 MICROPROCESSOR

**: RESET**

اگر این پایه را برای ۴ پالس high کنیم پردازنده reset (بازنشانی) خواهد شد. ولی در این پروژه نیاز به بازنشانی ریزپردازنده نداریم به همین دلیل این پایه را به زمین وصل می کنیم تا low باقی بماند.

**: READY**

چون نیازی نداریم که ریزپردازنده وارد حالت انتظار کنیم پایه READY را به منبع ولتاژ وصل می کنیم در نتیجه این پایه high باقی خواهد ماند و پردازنده هیچوقت وارد حالت انتظار نخواهد شد.

**: MN/MX′**

لازم نیست که پردازنده را وارد مد ماکزیمم کنیم در نیجه باید پایه **MN/MX′** را به **+5.0v** را وصل می کنیم در نتیجه پردازنده وارد مد مینیمم می گردد.

**: AD0 – AD15**

گذرگاه آدرس/داده را به گذرگاه مشترک وصل می کنیم تا با دیگر اجزا و ماژول های مدار بتوانیم داده و آدرس رد و بدل کنیم.

**: ALE**

هنگامی که آدرس روی باس است پردازنده سیگنال **ALE** را فعال می کند با استفاده از این می توانیم لچ را آگاه کنیم که آدرس روی باس است.

**: RD′ و WR′**

این پایه را باید به پایه های **RD′** و **WR′** ماژول **8255A (I/O Control)** متصل می کنیم تا عمل خواندن و نوشتن ریزپردازنده را با همدیگر هماهنگ کنیم.

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## ۲) Latch

در واقع از لچ استفاده می کنیم تا هر یک از Port های I/O Control را بخواهیم آدرس دهی کنیم. همانطور که در بخش قبل گفته شد لچ مورد استفاده دو پایه کنترلی OE و LE دارد.

OE′ سیگنالی است که خروجی لچ را فعال می کند این پایه را به زمین متصل می کنیم تا همیشه خروجی فعال باشد.

LE را باید به پایه ALE ریزپردازنده وصل کنیم تا هر وقت که فقط آدرس روی باس بود، ورودی لچ فعال شده و در نتیجه یکی از پورت های I/O را کنترل کند.

پایه های Q1 و Q2 را به ترتیب به A0 و A1 وصل می کنیم تا بتوانیم به وسیله این پایه های کنترلی ریزپردازنده، PORT های را کنترل کنیم.

## ۳) 8255A PROGRAMMABLE PERIPHERAL INTERFACE

پایه های D0 – D7 را به گذرگاه مشترک وصل می کنیم تا ورودی (داده ها) را از ریزپردازنده بگیریم RESET و CS′ را به زمین وصل می کنیم تا وارد فاز • منطقی شوند.

و پورت های را به صورت زیر در فاز برنامه نویسی آدرس دهی خواهیم کرد:

اگر آدرس را برابر 00H (0000 0000) قرار دهیم پورت **A** فعال خواهد شد.

اگر آرس را برابر 02H (0000 0010) قرار دهیم پورت **B** فعال خواهد شد.

اگر آدرس را برابر 04H (0000 0100) قرار دهیم **C** فعال خواهد شد.

اگر آدرس را برابر 06H (0000 0110) قرار دهیم کنترل ریجستر فعال خواهد شد.

به این صورت می توانیم پورت های را کنترل و برنامه نویسی کنیم.

## ۴) LCD

**VSS :**

این پایه را به GND وصل می کنیم.

**VDD و VEE :**

این دو پایه یکی برای تغزیه LCD و دیگری برای تغییر میزان CONTRAST صفحه نمایش است هر دو را به منبع تغذیه وصل می کنیم.

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

**: D0 – D7**

این پایه ها را به پورت I/O A وصل می کنیم (PA0 – PA7) تا بتوانیم داده ها و دستورالعمل از ریزپردازنده که از طریق BUS به پورت های ورودی I/O متصل شده اند را به LCD منتقل کنیم.

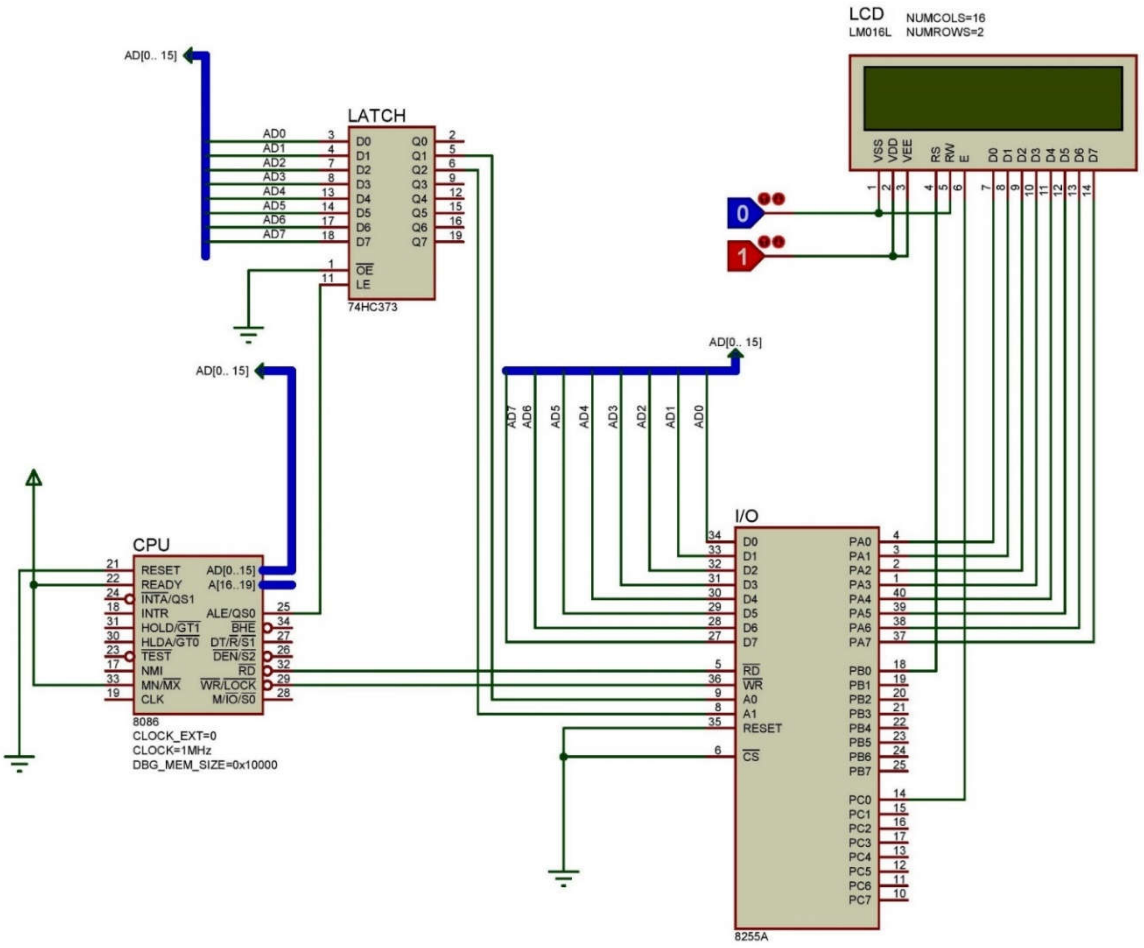
**: RS**

با استفاده از این پایه می توانیم بین ثبات های دستورالعمل و داده LCD جابه جا شویم. اگر بخواهیم کاراکتر ها را به LCD بفرستیم باید در ثبات داده ذخیره کنیم و درنتیجه RS را وارد فاز منطقی ۱ می کنیم، اگر بخواهیم به LCD دستور دهیم باید RS را وارد فاز منطقی ۰ کنیم، تا دستورالعمل ها وارد ثبات دستورالعمل شود. درنتیجه این پایه را به PB0 وصل می کنیم.

**: E**

با اعمال شدن یک سیگنال روی این پایه داده ها به داخل ثبات های LCD هدایت می شود در نتیجه این پایه را به PC0 وصل میکنیم تا بتوانیم روی این پایه نیز با استفاده برنامه نویسی کنترل داشته باشیم. در واقع این پایه را به پایه I/O interrupt متصل می کنیم.

تا به اینجا تا حدودی با معماری مدار آشنا شدیم در بخش بعدی و با شروع برنامه نویسی بیشتر با عملکرد مدار آشنا خواهیم شد.



**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

# بخش سوم

# بررسی کدهای برنامه

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## کد های برنامه

**Delay :**

دلیل اصلی برای ایجاد وقفه اختلاف سرعت پردازنده با سایر قطعات از جمله I/O و LCD است.

برای ایجاد وقفه از یک حلقه کمک می گیریم، منطق استفاده از حلقه در زبان اسمبلی به صورت مختصر این است که تعداد دفعات اجرا این حلقه را در ثبات CX قرار می دهیم، و بعد از هر بار اجرای این حلقه از این مقدار ۱ واحد کم شده تا به ۰ برسد.

دستور LOOP در پردازنده ۸۰۸۶، با هربار اجرا ۱۷.۵ پالس ساعت به طول می انجامد. درصورتی که پردازنده در فرکانس 1MHz تنظیم شده باشد زمان هر پالس ساعت برابر $\mathbf{1}\boldsymbol{s} \div \mathbf{1}\,\boldsymbol{MHz} = \mathbf{1}\boldsymbol{\mu_S}$ خواهد بود. بنابراین برای ایجاد یک وقفه ۱ میلی ثانیه $\frac{\mathbf{1}\boldsymbol{ms}}{\mathbf{1}\boldsymbol{\mu s} \times \mathbf{17.5}} \cong \mathbf{58}$ باید مقدار ۵۸ دسیمال برابر با 3A هگز را در ثبات CX قرار دهیم.

**OUT :**

با این دستور می توان محتویات یک ثبات همه منظوره را در یک ثبات I/O کپی کرد. فرمت این دستور به دو صورت است. در صورتی که عدد مربوط به پورت کمتر از ۲۵۵ دسیمال یا FF هگز باشد، می توان به صورت مستقیم استفاده کرد :

```
OUT PORT_NUMBER, Value
```

و در صورتی که عدد مربوط به پورت از ۲۵۵ بیشتر باشد ، باید ابتدا آن را در ثبات DX قراردهیم، سپس با استفاده از OUT ، مقدار را در پورت مورد نظر بنویسیم البته در این حالت فقط یک بایت قابل انتقال است :

```
MOV DX, PORT_NUMBER

OUT DX, Value
```

برای ارسال یک عدد دو بایتی باید به سراغ نیم ثبات AX برویم . به اینصورت که ابتدا ۸ بایت کم ارزش (سمت راست) را در AL قرار داده و به پورت میفرستیم . بعد بلافاصله ۸ بایت باارزش را به همین ترتیب در پورت قرار میدهیم :

```
MOV DX, PORT_NUMBER

MOV AX, Value

OUT DX, AL

OUT DX, AH
```

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

# تحلیل کد:

در ابتدا باید با توجه به مدار متغییر هایی را در قسمت داده به نام های پورت ها ایجاد کرده و مقدار دهی کنیم :

با دستور EQU مقادیر را برابر با متغییر ها قرار می دهیم.

```
DATA SEGMENT

    PORTA EQU 00H

    PORTB EQU 02H

    PORTC EQU 04H

    PORT_CON EQU 06H

DATA ENDS
```

سپس وارد قسمت کد شده و ابتدا آدرس شروع قسمت داده را در ثبات DS وارد می کنیم :

```
CODE SEGMENT

    MOV AX, DATA

    MOV DS, AX
```

دستور زیر نیز مشخص خواهد کرد که کد سگمت از کدام قسمت حافظه شروع شود در این برنامه آفست 00H یعنی از ابتدای حافظه شروع می شود

```
ORG 0000H
```

در این قسمت مقدار ۱۰،۰۰۰،۰۰۰ باینری را با استفاده از دستور OUT بروی I/O قرار می دهیم، در حالی که آدرس آن بروی مقدار PORT_CON تنظیم شده است این آدرس موجب ۱ شدن دو پایه A0 و A1 تراشه I/O شده و آن را در وضعیت دریافت برنامه قرار می دهد و دستور تعریف شده در آن فرآخوانی می شود همچنین این کار تمامی پورت ها را برابر مقدار منطقی ۰ قرار می دهد :

```
Start:

    MOV DX, PORT_CON

    MOV AL, 10000000B

    OUT DX, AL
```

در این قسمت باید ابتدا LCD را در وضعیت روشن قرار دهیم :

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

همان طور که در بخش های قبل اشاره شد وظیفه پورت C ایجاد یک سیگنال interrupt است، بنابراین ابتدا باید مقدار این پورت را بروی • منطقی قرار داد.

```
Display_On:
        MOV AL, 00H
        MOV DX, PORTC
        OUT DX, AL
```

سپس با دستورات زیر، CPU ابتدا آدرس پورت A را بروی گذرگاه داده/دستور قرار می دهد و سپس مقدار E هگز را. این امر موجب قرار گرفتن این مقدار بروی پورت A است. این چرخه برای بقیه پورت ها هم صادق است.

مقدار E هگز را بروی پورت A قرار می دهیم، درحالی که مقدار پورت B بروی • قرار دارد موجب فرمان روشن شدن LCD می شود:

```
        MOV AL, 0EH
        MOV DX, PORTA
        OUT DX, AL
```

سپس باید با استفاده از یک تاخیر از تصبیت مقادیر بروی پورت مورد نظر اطمینان حاصل کرد :

```
MOV CX, 00FFH
Delay0: LOOP Delay0
```

بعد از اتمام این حلقه باید مقدار پورت Cرا برابر با ۱ قرار داده و دوباره صفر کنیم این کار موجب ایجاد سیگنال شده محتوای پورت A بروی ثبات داخلی LCD بارگیری شود:

```
        MOV AL, 0FFH
        MOV DX, PORTC
        OUT DX, AL

        MOV AL, 00H
        MOV DX, PORTC
        OUT DX, AL
```

سپس به همین ترتیب مقدار 38 هگز را بروی LCD قرار می دهیم :

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

```
        MOV AL, 038H
        MOV DX, PORTA
        OUT DX, AL
MOV CX,00FFH
Delay2: LOOP Delay2
        MOV AL, 0FFH
        MOV DX, PORTC
        OUT DX, AL
MOV CX, 00FFH
Delay3: LOOP Delay3
        MOV AL, 00H
        MOV DX, PORTC
        OUT DX, AL
```

در این قسمت LCD آماده دریافت کارکتر ها می باشد بنابراین باید بروی ثبات داده LCD سوئیچ کرد، برای این منظور بایستی پورت B برابر ۱ قرار می دهیم:

```
Change_LCD_mode:
        MOV AL, 0FH
        MOV DX, PORTB
        OUT DX, AL
```

بعد از تغییر ثبات حالا می توان یک کارکتر را بروی I/O قرار داد و در نتیجه بر روی LCD چاپ کرد :

```
Write_Chars:
        MOV AL, 'H'
        MOV DX, PORTA
        OUT DX, AL
```

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

```
MOV CX, 00FFH
Delay4: LOOP Delay4
        MOV AL, 0FFH
        MOV DX, PORTC
        OUT DX, AL
MOV CX, 00FFH
Delay5: LOOP Delay5
        MOV AL, 00H
        MOV DX, PORTC
        OUT DX, AL
MOV CX, 0FFFFH ; A Big Delay
Delay100: LOOP Delay100
```

سپس به همین ترتیب می تون تمامی کارکتر ها را بر روی LCD نمایش داد.

در ادامه دوباره به روی ثبات کنترلی LCD سویچ می کنیم:

```
LCD_Mod_0:
        MOV AL, 0FFH
        MOV DX, PORTC
        OUT DX, AL
        MOV AL, 00H
        MOV DX, PORTB
        OUT DX, AL
```

حال LCD را بر روی حالت شیفت به راست قرار می دهیم:

```
Shift_on:
        MOV AL, 00H
```

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

```
        MOV DX, PORTC
        OUT DX, AL
        MOV AL, 1CH
        MOV DX, PORTA
        OUT DX, AL
```

در این جا با هر پالس که بر روی پای E اعمال می شود، کل LCD به راست جابه جا می شود. بنابراین با استفاده از دستور JMP به قسمت Shifting پرش می کنیم:

```
Shifting:
        MOV AL, 00H
        MOV DX, PORTC
        OUT DX, AL
        MOV AL, 0FFH
        MOV DX, PORTC
        OUT DX, AL
        MOV CX, 01900H
    Delay200: loop Delay200
JMP Shifting
```

در انتها نیز باید قسمت کد بسته و کد خاتمه یابد :

```
 CODE ENDS
END
```

## پیشنهاد

برای انجام این پروژه می توانستیم از پرچم های busy نیز استفاده کنیم. همچنین می توانستیم از تنها دو پورت استفاده کنیم. و همچنین خواهیم توانست با اتصال سنسور های آنالوگ همچون ولت سنج و سنسور دما و یک تبدیل گر سیگنال آنالوگ به دیجیتال مدار های پیچیده تری بسازیم. که خود پروژه ای دیگر است.

**Source:** https://github.com/Fasihi-Rad/intel_8086_and_LCD

## بحث و نتیجه گیری

در طول این پروژه با اجزای سازنده ۸۰۸۶ و دستگاه های جانبی قابل اتصال به آن همچون I/O Control، LCD متنی تا حدودی آشنا شدیم و نحوه برنامه نویسی و کنترل هر یک را فرا گرفتیم. همچنین مشاهده کردیم که چگونه یک سیستم دیجیتال را قبل از مرحله ساخت توسط نرم افزار Proteus شبیه سازی و آزمایش کنیم.

این پروژه دید تازه ای از برنامه نویسی اسمبلی و کار با مدار های دیجیتال به ما داد.

# منابع:


۱- میکرو کنترلر خانواده 8051 ، آی اسکات مکنزی - حسن سید رضی

۲- ریز پردازنده Z80، بری .بی بری - دکتر سعید حسین نیا

۳- اصول طراحی VLSI، داگلاس پاکنل، کامران اشراقیان - مرتضی صاحب‌الزمانی

4- The Intel Microprocessors - Barry B. Brey

5- LM016L Display Datasheet - **HITACHI©**

6- Intel 8255A Datasheet - **Intel© Corporation**

7- 74HC373; 74HCT373 Octal D-type transparent-latch 3-state Rev. 7 -2020 Datasheet - **Nexperia© B.V**