بسمه تعالی

گزارش کار آزمایشگاه اصول میکرو کامپیوتر

MAT 85

استاد : آقای دکتر کلامی

دانشجو : سینا اژدرکش

گرایش:کارشناسی ناپیوسته مهندسی تکنولوژی الکترونیک

ISLAMIC AZAD UNIVERSITY OF URMIA

مقدمه

دستگاه MAT 85 براي آموزش زبان اسمبلي ميكروپروسسور 8085 مورد استفاده قرار ميگيرد.اين دستگاه داراي حافظه RAM و همچنين حافظه ROM بوده و برنامه ها نوشته شده در قسمت RAM ذخيره و اجرا مي شود. مبنای مورد استفاده در این دستگاه هگزا دسیمال میباشد (0-9,A-F) بنا براین کلیه آدرس ها و اطلاعات وارد شدهو نمایش داده شده توسط این دستگاه هگزادسیمال خواهد بود.

میکروپروسسور 8085: یک میکروپرسسور 8 بیتی با 16 خط آدرس میباشد بنابراین این دستگاه 4 سون سگمنت برای آدرس و 2 عدد سون سگمنت برای نمایش دیتا اختصاص داده است.

تذکر: اطلاعات ROM فقط قابل خواندن بوده و در صورت تغییر آن ها با پیغام خطا مواجه میشویم.

نحوه کار با دستگاه MAT85

برای این دستگاه 24 کلید در نظر گرفته شده که 0 تا 16 تای آنها مربوط به اعداد مبنای هگزادسیمال بوده و 8 تای آنها نیز مربوط به دستگاه است برای کار با این دستگاه دو روش کلی وجو دارد:

1.اتصال با استفاده از پورت کامپیوتر

2.استفاده از صفحه کلید دستگاه-----> ما در آزمایشگاه از این روش استفاده میکنیم.

برای مشخص کردن نحوه کار با دستگاه یک کلید دو حالته بر روی دستگاه وجود دارد که دارای حالت های KEY, SER است.اگر کلید در حالت KEY باشد با روشن کردن دستگاه پیغام 8085- بر روی صفحه ظاهر میشود ولی اگر در حالت SER باشد پیغام 8085 بر روی صفحه ظاهر میشود و هیچ یک از دکمه های دستگاه کار نخواهد کرد.

دستورات

دستور SUBST MEM : این دستور برای مشاهده حافظه RAM , ROM استفاده میشود همچنین با این دستور میتوان محتویات RAM را تغییر داد. صورت کلی دستور به شکل زیر است;

SUBST MEM < Order 16 Bit > NEXT <Data>

تذکر مهم:وقتی دستگاه این علامت را نشان داد منتظر وارد شدن دستور است (-)

وقتی دستگاه این علامت را نشان داد منتظر وارد شدن آدرس است (.)

دستور RESET : این دستور برای ریست کردن سخت افزاری دستگاه مورد استفاده قرار میگیرد.سعی شود از این دستور هیچگاه استفاده نشود فقط در مواقعی میخواهیم از حالت KEY به SER یا بلعکس تغییر حالت دهیم یا دستگاه هنگ کرده باشد انگاه از این دستور استفاده میکنیم.

دستور EXAM REG : این دستو ر مربوط به خود دستگاه است.از این دستور برای مشاهده محتویات رجیستر ها و در صورت لزوم تغییر محتویات آن ها استفاده میشود.شکل کلی دستور به شرح ذیل است;

EXAM REG (نام رجیستر مورد نظر)

رجیستر های 8085:

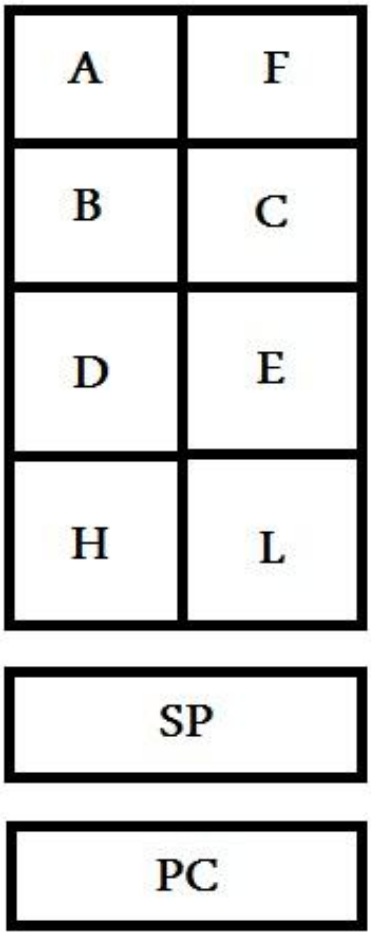
رجیستر های 8085 از یک سری رجیستر های 8 بیتی و یک سری 16 بیتی تشکیل شده است.

بطور کلی 8 رجیستر 8 بیتی و 2 رجیستر 16 بیتی که از این 8 رجیستر 6 رجیستر آن همه منظوره نام داشته و یک رجیستر تحت عنوان انباره و یک رجیستر دیگر به نام پرچم شناخته میشوند.

رجیستر ها عبارت اند از: B,C,D,E,H,L و 16 بیتی SP (اشاره گر پشته) و PC (شمارنده برنامه)

نکته:در صورت نیاز میتوان رجیستر های همه منظوره را به صورت جفتی و به عنوان یک رجیستر 16 بیتی نیز مورد استفاده قرار داد.

جفت رجیستر های ما عبارت اند از : BC,DE,HL



برنامه نویسی به زبان اسمبلی در میکروپرسسور 8085:

دستور ORG: از این دستور به عنوان راهنمای اسمبل استفاده میشود به این نحوه که آدرس شروع برنامه را برای ما مشخص میکند.شکل کلی این دستور به صورت ORG میباشد و هیچگونه UPCODE در 8085 ندارد.

MOV R1,R2:این دستور برای کپی کردن اطلاعات رجیستر R2 در R1 بکار میرود.همچنین محتویات R2 تغییر نمیکند. R2 ---------> R1

```
MOV   C , E
```

دستور بالا محتویات E را در C کپی میکند.این دستور یک دستور تک بایتی است.

دستور DATA , R , MVI:

این دستور اطلاعات 8 بیتی را در رجیسترهایش قرار میدهد. این دستور یک دستور دو بایتی بوده که یک بایت مربوط به UPCODE دستور و بایت دیگر مربوط به اطلاعات 8 بیتی است.

عدد 1B را در رجیستر E قرار میدهد

```
MVI    E , 1B
```

دستور END : این دستور نیز یک دستو راهنمای اسمبل است که انتهای برنامه را برای ما مشخص میکند ولی UPCODE در مجموعه دستورالعمل ها ندارد.

بنابراین از یک دستور معادل END استفاده میکنیم.

دستور RST1: این دستور یک دستور وقفه است ولی ما بجای END از ان استفاده میکنیم

UPCODE مربوط به دستور RST1 برابر با CF میباشد

مثال% برنامه ای بنویسید که محتویات رجیستر C را در B کپی کند.(آدرس شروع 2800 میباشد)

```
ORG    2800
MOV    B , C
END
```

دستور GO: از این دستور برای اجرای برنامه های نوشته شده در RAM دستگاه استفاده میشود.صورت کلی این دستور به شکل زیر است;

GO (آدرس شروع برنامه) EXEC

پس اجرای این دستور اگر پیغام 8085- ظاهر شود یعنی UPCODE برنامه صحیح میباشد ولی این دلیلی بر صحیح بودن برنامه نیست.بنابراین برای اطمینان از صحت برنامه با توجه به صورت مسئله باید امتحان شود.

مثال% برنامه ای بنویسید که عدد 4F را در رجیستر H قرار داده و سپس محتویات را در رجیستر C کپی کند؟

```
ORG 2800
MVI H , 4F
MOV C , H
END
```

دستور LDA ADDRESS: این دستور یک دستور 3 بایتی بوده که محتویات آدرس مقابل دستور را در انباره کپی میکند.یک بایت مربوط به UPCODE دستور و دو بایت مربوط به آدرس ان میباشد.

تذکر:هنگام کمپایل آدرس ابتدا 8 بیت پایین و سپس 8 بیت بالا بایستی وارد شود.

مثلا ---------- > LDA 28D0

دستور STA ADDRESS: این دستور نیز یک دستور 3 بایتی بوده و کاری درست بر عکس LDA انجام میدهد. به این ترتیب که محتویات انباره را در آدرس مورد نظر کپی میکند.

دستور ADD R :این دستور یک دستور تک بایتی بوده که محتویات رجیستر R را با انباره جمع کرده و نتیجه را در انباره قرار میدهد.

```
ADD B
```

دستور بالا محتویات رجیستر B را با انباره جمع کرده و در انباره قرار داده است

به این شکل A+B=A

مثال% برنامه ای بنویسید که محتویات خانه حافظه های 28D1 , 28D0 را با هم جمع کرده و نتیجه را در آدرس 8000 قرار دهد؟(آدرس شروع 2800 میباشد)

```
        ORG   2800
LDA   28D0
        MOV  B , A
        LDA  28D1
            ADD   B
          STA   8000
          END
```

دستور INR R: این دستور محتویات رجیستر نوشته شده در مقابل آن را یک واحد افزایش میدهد و یک دستور تک بایتی میباشد.

محتویات رجیستر B را یک واحد افزایش میدهد INR B

دستور DCR R: این دستور محتویات رجیستر نوشته شده در مقابل آن را یک واحد کاهش میدهد و یک دستور تک بایتی میباشد.

محتویات رجیستر B را یک واحد کاهش میدهد DCR B

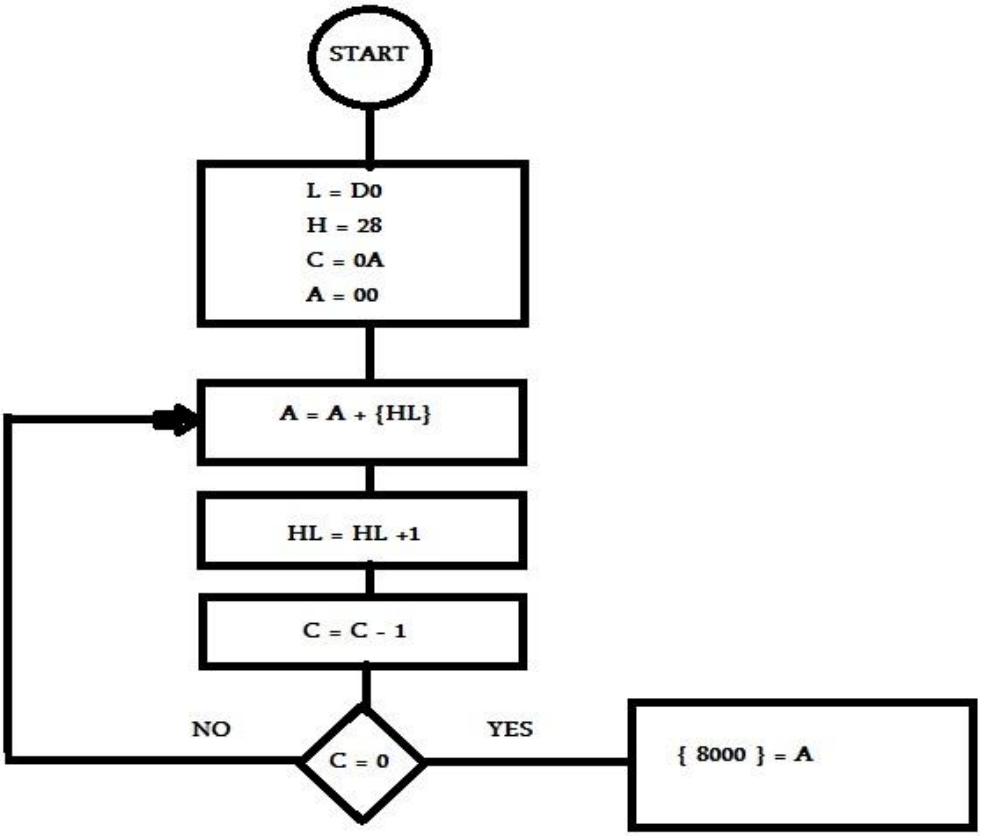
دستورJZ ADDRES S: این دستور یک دستور 3 بایتی هست و در صورت صفر شدن عملیات محاسبات قبلی به آدرس نوشته شده در مقابل آن پرش میکند

دستورJNZ ADDRES S : این دستور یک دستور 3 بایتی هست و در صورت صفر نشدن عملیات محاسبات قبلی به آدرس نوشته شده در مقابل آن پرش میکند

دستور ADD M : این دستور محتویات انباره را با محتویات آدرسی که در جفت رجیستر HL قرار دارد جمع کرده و نتیجه را در انباره قرار میدهد.

مثال% برنامه ای بنویسید که اعداد موجود در خانه حافظه 28D9 ، 28D0 را (10 عدد) با هم جمع کرده و نتیجه را در خانه حافظه 80000 قرار دهد؟(آدرس شروع 2800 میباشد)

```
ORG     2800
MVI     L , D0
MVI     H , 28
MVI     C , 0A
MVI     A , 00
ADD     M
INR     L
DCR     C
JNZ     2808
STA     8000
END
```



حلقه های تو در تو: برای ایجاد تاخیر های زمانی از حلقه های تو در تو استفاده میشود.این حلقه ها بایستی دارای نقاط مشترک با سایر حلقه ها باشند.شکل کلی برنامه برای سه حلقه تو در تو به شرح ذیل است:

```
MVI   D , 04
MVI   C , FF    (LOOP3)
MVI   E , FF    (LOOP2)
DCR   E         (LOOP1)
JNZ    LOOP1
DCR   C
JNZ   LOOP2
DCR   D
JNZ   LOOP3
```

زیر برنامه در بسیاری از برنامه های زبان اسمبلی یک یا چند قسمت برنامه به دفعات تکرار می شود برای جلوگیری از این نوع تکرار ها میتوان آن قسمت از برنامه را درون زیر برنامه قرار داد و در صورت لزوم داخل برنامه آن را فراخوانی کرد.

دستور CALL: این دستور یک دستو سه بایتی بوده و برای فراخوانی زیر برنامه استفاده میشود. یک بایت برای UPCODE دستور و دو بایت برای آدرس زیر برنامه استفاده میشود.

```
CALL    اسم زیر برنامه
```

تذکر مهم: آخر زیر برنامه حتما باید به دستور RET ختم شود

آدرس زیر برنامه میتواند هر آدرسی غیر از آدرس خود برنامه اصلی باشد.به شرح ذیل;

در بدنه اصلی برنامه این دستوررات با ید گنجانده شود;

```
CALL   DELAY
CALL   DELAY
END
```

تمرین% برنامه ای بنویسید که دو عدد موجود در خانه حافظه 28D1 , 28D0 را با هم جمع کرده و پس از یک تاخیر در خانه حافظه 8000 قرار دهد؟ (آدرس شروع 2800 است)

```
ORG   2800
LDA   28D0
MOV  B , A
   LDA  28D1
   ADD   B
    STA  8000
CALL  DELAY
CALL  DELAY
END
```

```
DELAY
        MVI   D , 04
        MVI   C , FF   (LOOP3)
        MVI   E , FF   (LOOP2)
        DCR   E        (LOOP1)
        JNZ    LOOP1
        DCR   C
        JNZ   LOOP2
        DCR   D
        JNZ   LOOP3
        RET
```

نحوه استفاده از سویچ ها و LED ها بصورت باینری:

در دستگاه MAT 85 هشت سویچ برای وارد کردن اعداد باینری بصورت هشت بیتی و هشت عدد LED برای نمایش DATA بصورت باینری هشت بیتی در نظر گرفته شده است.که بعنوان کد های ورودی و خروجی دستگاه عمل میکنند.

پورت 21H برای سویچ ها و 22H برای LED ها تعریف شده است.

تذکر مهم:برای فعال کردن این پورت در ابتدای هر برنامه بایستی دو دستور زیر را به ترتیب بنویسیم:

```
MVI     A , 0A
OUT     20
```

دستور IN PORT :این دستو محتویات شماره پورت مورد نظر را به انباره منتقل میکند و یک دستور دو بایتی بوده که یک بایت مربوط به UPCODE دستور و یک بایت مربوط به شماره پورت مورد نظر است به عنوان مثال برای خواندن اطلاعات سوئیچ ها در دستگاه MAT85 دستور زیر را وارد میکنیم:

```
IN   21
```

دستور OUT PORT :این دستور اطلاعات موجود در انباره را در پورت مورد نظر منتقل میکند و یک دستور دو بایتی بوده که یک بایت مربوط به UPCODE دستور و یک بایت مربوط به شماره پورت مورد نظر است به عنوان مثال برای خواندن اطلاعات سوئیچ ها در دستگاه MAT85 دستور زیر را وارد میکنیم:

```
OUT   22
```

تمرین% برنامه ای بنویسید که محتویات سوئیچ ها را بر روی LED نمایش دهد ؟

```
MVI   A , 0A
OUT   20
IN
OUT
```

مثال% برنامه ای بنویسید که دو عدد موجود در خانه حافظه های 8000 و 8001 را باهم جمع کرده و نتیجه را بر روی ال ای دی ها نشان دهد؟.

```
ORG     2800
MVI     A  ,  0A
OUT     20
LDA     8000
MOV     B  ,  A
LDA     8001
ADD     B
OUT     22
END
```

مثال% برنامه ای بنویسید که LED ها 10 با روشن خاموش کند؟

```
ORG    2800
MVI    A  ,  0A
OUT    20
MVI   B  ,  0A
LOOP1: CALL   DELAY
MVI    A  , FF
OUT    22
CALL   DELAY
MVI    A  , 00
OUT    22
DCR    B
JNZ    LOOP1
END

DELAY
MVI   C  ,  FF
MVI   E  ,  FF    (LOOP3)
DCR    C          (LOOP2)
JNZ    LOOP2
DCR    C
JNZ   LOOP3
RET
```

دستور CMP R: این دستور محتویات انباره را با رجیستر مورد نظر مقایسه کرده و با توجه به شرایط پرچم کری و Z را تغییر میدهد.
اگر محتویات رجیستر بزرگتر از انباره باشد پرچم کری یک میشود اگر محتویات انباره با رجیستر برابر باشد پرچم کری و Z هر دو صفر میشوند.
اگر محتویات انباره بزرگتر از رجیستر باشد پرچم کری صفر میشود.

دستور JC ADDRESS:این دستور در صورت SET شدن کری به آدرس مقابل آن پرش میکند.ویک دستور سه بایتی بوده که یک بایت UPCODE دستور و دو بایت دیگر مربوط به آدرس است.

دستورJNC ADDRESS : این دستور در صورت RESET شدن کری به آدرس مقابل آن پرش میکند.ویک دستور سه بایتی بوده که یک بایت UPCODE دستور و دو بایت دیگر مربوط به آدرس است.

دستور JMP ADDRESS: این دستور یک دستور پرش غیر شرطی است که به آدرس مقابل آن بدون هیچ شرطی پرش میکند و یک دستور سه بایتی است.

مثال% برنامه ای بنویسید که اعداد موجود در خانه حافظه های 28D0, 28D1 , 28D2 را با هم مقایسه و کوچکترین آن ها را انتخاب کند و بر روی LED نشان دهد؟

```
ORG  2800
MVI   A  ,  0A
OUT  20
LDA  28D0
MOV  B  ,  A
LDA   28D1
CMP  B
A >B ,  C=0  JNC   LOOP1
MOV   B , A
LDA   28D2
CMP  B
A >B   ,  C=0   JNC    LOOP1
JMP    LOOP3
LOOP2=MOV    A  ,  B
LOOP3=OUT    22
END
```