## ۳– دستورات مقایسه و پرش :

### ۳-۱) دستورات کنترلی و پرش

#### ۳-۱-۱) دستور پرش غیر شرطی jmp :

برای اجرای این دستور کامپیوتر آفست آدرس پرش را که جلوی دستور نوشته شده است در IP قرار می دهد.
بدین ترتیب اجرای دستورات از این IP دنبال می شود . البته می توان شماره آدرس را نیز قرار داد(مثلا JMP 2010H)

| الگو: |
| --- |
| JMP آدرس یا برچسب |

به مثال زیر دقت کنید . در خط سوم دستور پرش غیر شرطی آورده شده است . با اجرای این دستور ادامه برنامه از خط ششم ادامه می یابد وخطوط چهارم و پنجم اجرا نمی شود.

```
      MOV AL , 5
      ADD AL , BL
      JMP L1
      MUL BL
      INC BL
L1 : SUB CX , 2
```

مثال کتاب: برنامه ای بنویسید که :
الف)مقدار ثبات های dl,bl,al به ترتیب برابر با ۱و۲و۴ گردد.
ب) ثبات bl به alاضافه شود و نتیجه در ثبات dl ضرب گرددو این عملیات با استفاده از دستور jmp مرتبا تکرار شود.

```
          PAGE 100,110
TITLE     'jump.asm' jump  program
;-----------------------------------------
          .MODEL SMALL
          .STACK  64          ;Define stack
;-----------------------------------------
          .CODE               ;Define code segment
MAIN      PROC FAR
          MOV AX,@data        ;1- Set data segment
          MOV DS,AX           ;2- address
;
          MOV AL,1            ;3-AL=1
          MOV BL,2            ;4-BL=2
          MOV DL,4            ;5-DL=4
AGAIN:    ADD AL,BL           ;6-AL=AL+BL
          MUL DL              ;7-AX=AL*DL
          JMP AGAIN           ;8-jump AGAIN
;
          MOV AX,4C00H        ;9-End of
          INT 21H             ;10- processing
MAIN      ENDP                ; End of procedure
          END MAIN            ; End of program
```

دستور پرش معمولی شامل سه نوع زیر است:

الف) پرش **SHORT** که ازفاصله ۱۲۸- الی ۱۲۷+ نسبت به آدرس محل برچسب **begin**است.

**JMP SHORT begin**

ب) پرش ( **NEAR** در داخل سگمنت ) از ۳۲۷۶۸- الی ۳۲۷۶۷+ نسبت به آدرس فعلی پرش کند.

**JMP NEAR begin**

ج) پرش ( **FAR** از یک سگمنت به سگمنت دیگر پرش کند. )

**JMP FAR …**

### ۳-۱-۲)دستورات پرش شرطی :

تمامی دستورات پرش شرطی از نوع **SHORT** هستند.

الف) پرش شرطی مبتنی بر بیت های پرچم

ب )پرش شرطی مبتنی بر اعداد علامت دار

ج )پرش شرطی مبتنی بر اعداد بدون علامت

جدول پرش شرطی مبتنی بر بیت های پرچم

| دستور | شرط پرش | توضیحات |
|---|---|---|
| JZ ( JE ) | ZF=1 | پرش روی صفر |
| JNZ ( JNE ) | ZF=0 | پرش روی غیر صفر |
| JS | SF=1 | پرش روی علامت منفی |
| JNS | SF=0 | پرش روی علامت مثبت |
| JO | OF=1 | پرش روی سرریز |
| JNO | OF=0 | پرش روی عدم سرریز |
| JP( JPE ) | PF=1 | پرش روی ایجاد توازن |
| JNP ( JPO ) | PF=0 | پرش روی عدم ایجاد توازن |
| JC | CF=1 | پرش روی ایجاد رقم نقلی |
| JNC | CF=0 | پرش روی عدم ایجاد رقم نقلی |

**(J: JUMP ,Z: ZERO ,E:EQUAL , N:NOT ,S:SIGN ,O:OVERFLOW , P: PARIY,C:CARRY)**

در دستورات فوق اگر شرط برقرار باشد عمل پرش رخ می دهد , در غیر اینصورت دستور بعدی اجرا می شود.

**در مثال زیر دقت کنید:**

```
        MOV   AX,-100
        ADD   AX,BX
        SUB    AX,CX
        JNZ    NEXT
        MUL  BX
        MOV  AX,BX
NEXT :  MOV CX,10
```

در مثال فوق در خط چهارم یک دستور پرش شرطی آورده شده که می خواهیم ببینیم آیا پرش انجام می شود یا خیر ؟

شرط پرش این دستورآن است که **ZF=0** شود یعنی پرش را روی نتیجه غیر صفر انجام می دهد . برای کنترل پرچم **ZF** بایستی به دستور قبل از پرش نگاه کنیم یعنی دستور **SUB AX,CX** از آنجایی که در این دستور نتیجه در ثبات **AX** ذخیره می شود بنابراین دستور پرش این ثبات را ملاک پرش خود قرار می دهد . اگر محتوای این ثبات پس از اجرای دستور **SUB** صفر نشود در نتیجه **ZF=0** شده و پرش به برچسب **NEXT** انجام می شود و ادامه برنامه از آن خط دنبال می شود . اما هرگاه پس از اجرای دستور **SUB** ثبات **AX** صفر شود آنگاه **ZF=1** شده و درنتیجه عمل پرش انجام نمی شود و ادامه برنامه از خط بعد از پرش یعنی **MUL BL** دنبال می شود.

**مثال ۲: محتوای متغیر PH و ثبات CX را پس از اجرای قطعه برنامه زیر بنویسید.**

```
        PH  DW  '?'
        MOV  PH,0
        MOV  CX,10
Begin : ADD  PH,CX
        DEC   CX
        JNZ   begin
```

**جواب:**

همانطور که مشاهده می شود در خط آخر یک دستور پرش شرطی آورده شده ولی برچسب آن به خطوط ماقبل اشاره دارد یعنی اگر پرش انجام شود به خطوط ماقبل پرش انجام می پذیرد . شرط پرش نتیجه غیر صفر می باشد یعنی **ZF=0** و این نتیجه از دستور ماقبل پرش بدست می آید یعنی ثبات **CX** .

بار اول **PH=10** : و **CX=9** می بینیم که محتوای **CX** صفر نشده و در نتیجه **ZF=0** و پرش به برچسب **begin** انجام می شود.

بار دوم **PH=19** : و **CX=8** می بینیم که محتوای **CX** صفر نشده و در نتیجه **ZF=0** و پرش به برچسب **begin** انجام می شود.

بار سوم **PH=27** : و **CX=7** می بینیم که محتوای **CX** صفر نشده و در نتیجه **ZF=0** و پرش به برچسب **begin** انجام می شود.

بار چهارم **PH=34** : و **CX=6** می بینیم که محتوای **CX** صفر نشده و در نتیجه **ZF=0** و پرش به برچسب **begin** انجام می شود .

بار پنجم **PH=40** : و **CX=5** می بینیم که محتوای **CX** صفر نشده و در نتیجه **ZF=0** و پرش به برچسب **begin** انجام می شود.

بار ششم **PH=45** : و **CX=4** می بینیم که محتوای **CX** صفر نشده و در نتیجه **ZF=0** و پرش به برچسب **begin** انجام می شود.

بار هفتم **PH=49** : و **CX=3** می بینیم که محتوای **CX** صفر نشده و در نتیجه **ZF=0** و پرش به برچسب **begin** انجام می شود.

بار هشتم **PH=52** : و **CX=2** می بینیم که محتوای **CX** صفر نشده و در نتیجه **ZF=0** و پرش به برچسب **begin** انجام می شود.

بار نهم **PH=54** : و **CX=1** می بینیم که محتوای **CX** صفر نشده و در نتیجه **ZF=0** و پرش به برچسب **begin** انجام می شود.

بار دهم **PH=55** : و **CX=0** می بینیم که محتوای **CX** صفر شد و در نتیجه **ZF=1** و دیگر پرش به برچسب **begin** انجام نمی شود.

## جدول پرش شرطی مبتنی بر اعداد علامت دار

| دستور | توضیحات |
|---|---|
| JG ( JNLE ) | اگر عملوند اول بزرگتر باشد پرش انجام می شود ( در صورت مساوی و کوچکتر پرش انجام نمی شود ) |
| JGE ( JNL ) | اگر عملوند اول بزرگتر یا مساوی باشد پرش انجام می شود ( در صورت کوچکتر بودن پرش انجام نمی شود ) |
| JL ( JNGE ) | اگر عملوند اول کوچکتر باشد پرش انجام می شود ( در صورت مساوی و بزرگتر پرش انجام نمی شود ) |
| JLE ( JNG ) | اگر عملوند اول کوچکتر یا مساوی باشد پرش انجام می شود ( در صورت کوچکتر بودن پرش انجام نمی شود ) |

**(J: JUMP , L: LESS , G:GREATER , E:EQUAL , N:NOT)**

**در مثال زیر محتوای ثبات AL با عدد 20H مقایسه می شود(دستور cmp کارش مقایسه دو عملوند است) در صورتی که < AL 20Hباشد پرش به برچسب مورد نظر انجام می شود:**

```
CMP   AL , 20H
JL   NEXT
……
…….
NEXT: …..
```

جدول پرش شرطی مبتنی بر اعداد علامت دار

| توضیحات | دستور |
|---|---|
| اگر عملوند اول بزرگتر باشد پرش انجام می شود ( در صورت مساوی و کوچکتر پرش انجام نمی شود ) | JA ( JNBE ) |
| اگر عملوند اول بزرگتر یا مساوی باشد پرش انجام می شود ( در صورت کوچکتر بودن پرش انجام نمی شود ) | JAE ( JNB ) |
| اگر عملوند اول کوچکتر باشد پرش انجام می شود ( در صورت مساوی و بزرگتر پرش انجام نمی شود ) | JB ( JNAE ) |
| اگر عملوند اول کوچکتر یا مساوی باشد پرش انجام می شود ( در صورت کوچکتر بودن پرش انجام نمی شود ) | JBE ( JNA ) |

**(J: JUMP , A: ABOVE , B:BELOW , E:EQUAL , N:NOT)**

**در مثال زیر محتوای ثبات AH با عدد AL مقایسه می شودو در صورتی که AH =< AL باشد پرش به برچسب مورد نظر انجام می شود:**

```
CMP   AH,AL
JBE   CONT1
……
…….
CONT1: …..
```

**نکته : تمام دستورات پرش شرطی از نوع SHORT هستند ,یعنی حداکثر به ۱۲۸+ بایت پایین تر یا به ۱۲۸- بایت بالاتر می توانند پرش کنند.**

### ۳-۲) دستور مقایسه ای CMP:

**این دستور دو عملوند را باهم مقایسه مب کند و روی بیتهای پرچم اثر می گذارد.**

**الگو:**

**CMP عملوند ۱ , عملوند ۲**

**این دستور مانند دستور SUB عمل می کند با این تفاوت که نتیجه در جایی ذخیره نمی گردد بلکه مقادیر بیتهای ثبات پرچم را تغییر می دهد ,در این دستور عملوندها تغییر نکرده و عملوند مقصد نبایستی یک عدد ثابت باشد.**

جدول تغییر بیتهای پرچم در دستور CMP

| | CF | SF | ZF |
|---|---|---|---|
| عملوند۲ > عملوند۱ | 0 | 0 | 0 |
| عملوند۲ = عملوند۱ | 0 | 0 | 1 |
| عملوند۲ < عملوند۱ | 1 | 1 | 0 |

مثال : برنامه زیر چه عملی را انجام می دهد ؟

```
CMP AL,20
JZ NEXT
```

جواب : در برنامه ابتداء محتوای ثبات AL با عدد 20 مقایسه می شود . در صورت برابر بودن از آنجا که ZF=1 شده و شرط پرش فراهم می شود ادامه برنامه از برچسب NEXT دنبال می شود . اما هرگاه محتوای AL با عدد 20 برابر نباشد آنگاه ZF=0 شده و درنتیجه پرش انجام نمی شود.

نکته مهم در مورد تبدیل کد نویسی:

اگر در زبانهای سطح بالا مشاهده کرده باشید یک دستور حلقه شرطی وجود دارد به نام while , که تازمانیکه شرط برقرار باشد دستورات داخل حلقه را اجرا میکند ولی اگر شرط غلط شد از حلقه خارج می شود.

مثلا دستور(a>10) while به این معنی است که متغییر حافظه a را که قبلا مقدار دهی شده است را بررسی کن و اگر مقدارش از ۱۰ بزرگتر بود دستورات داخل حلقه را اجرا کن و درغیر اینصورت از حلقه خارج شو.

حالا می خواهیم به کمک دستورات یادگرفته شده در اسمبلی این کد را تبدیل کنیم که به شرح زیر است:

```
        Mov  ax,a
While: cmp ax,10
        Ja  while_body      ---> اگر شرط درست بود برو به برچسب مربوطه
        Jmp  end_while      ---> اگر شرط غلط بود برو به برچسب مربوطه
While_body: ……
        …….
        …….
        …….
        Jmp  while
End_while: …….
        …….
        ……
        ……
```

در اینجا تا وقتی که مقدار متغیر بزرگتر از ۱۰ هست دستورات حلقه اجرا می شود

دستورات فوق به یک روش دیگر به شرح زیر هم قابل نوشتن است:

```
            Mov  ax,a
      While: cmp ax,a
             Jbe  end_while
             .
             .
             .
             .
             Jmp  while
End_while :  .
             .
             .
             .
```

اگر کوچکتر مساوی بود برو به پایان حلقه

اگر بزرگتر از ۱۰ بود و شرط درست بود بدنه حلقه اجرا شود

تمرین کلاسی۱:

بازسازی حلقه **while** روبرو به زبان اسمبلی:

```
While  (a>10  &&  b<=10)
{
.
.
.
}
```

جواب:

```
            Mov ax,a
            Mov bx,b
      While:cmp  ax,10
             Jbe  end_while
             Cmp  bx,10
              Ja   end_while
             .
             .
             .
             .
             Jmp  while
End_while :  .
             .
             .
             .
```

**تمرین کلاسی ۲:**

**بازسازی حلقه while روبرو به زبان اسمبلی:**

```
While  (b<5  ||  a>=10)
{
.
.
.
}
```

**جواب:**

```
            Mov ax,a
            Mov bx,b
      While:cmp  bx,5
            Jb  while_body
            Cmp  ax,10
             Jae   while_body
             Jmp  end_while
While_body:.
            .
            .
            .
            .
            Jmp  while
End_while :  .
            .
            .
            .
```