دستورات ضرب و تقسیم

استفاده از دستورات ضرب و تقسیم کمی پیچیده تر از دستورات دیگر است. زیرا بسته به اندازه و نوع عملوند (بدون علامت و علامتدار) متفاوت از هم هستند. در این دستورات یک عملوند همیشه ثبات انباشتگر (AL/AX) است.

MUL
IMUL
DIV
IDIV

MUL

دستورالعمل mul (multiply) عمل ضرب بدون علامت را انجام می دهد. فرم کلی دستور به شکل زیر است :

mul src

دستور دارای یک عملوند است درحالیکه عمل ضرب به دو مقدار نیاز دارد .عملوند دیگر همیشه ثبات انباشتگر (AL/AX) درنظر می گرفته می شود .

حاصل ضرب ۸ بیت در ۸ بیت به بیش از یک بایت و ۱۶ بیت در ۱۶ بیت به بیش از دو بایت احتیاج دارد .

دستور ضرب بسته به اندازه تک عملوند خود به دو صورت ممکن است عمل کند :

- اگر عملوند یک بایتی باشد؛ عملوند را در AL ضرب کرده نتیجه را در AX ذخیره می کندد.

AX← AL*src(8)

- اگر عملوند دوبایتی باشد؛ عملوند را در AX ضرب کرده نتیجه را در ثبات DX:AX قرار می دهد.

DX:AX←AX*src(16)

دستور mul به صورت های زیر می تواند باشد:

mul register
mul memory

توجه کنید که عملوند دستور mul نمی تواند یک داده فوری باشد.

دستور mul روی فلگ های Carry و Overflow را تاثیر می گذارد. بعد از اجرای دستور اگر نیمه بالای حاصلضرب برابر با صفر باشد فلگ های Carry و Overflow صفر می شوند در غیر اینصورت یک می شوند .
بقیه فلگ ها وضعیت نامعینی دارند خصوصا فلگ های Sign و Zero مقدار معنی داری ندارند. برای بررسی علامت و نتیجه صفر می توان فلگ های Carry و Overflow را بررسی کرد .

IMUL

دستور imul (integer multiply) مشابه دستور mul است با این تفاوت که عمل ضرب علامتدار را انجام می دهد و روی عملوندهای علامتدار عمل می کند. علامت حاصل ضرب با توجه به بیت علامت عملگرها تنظیم می شود .

مثال. دستورات زیر حاصل عبارت 6 * (J*7 + K) را محاسبه می کند .نتیجه در DX:AX قرار می گیرد .

```
mov AL,7
mov BL,J
imul BL
add AX, K
mov BX,AX
mov AX,6
imul BX
```

DIV

دستورالعمل div (division) برای تقسیم اعداد بدون علامت استفاده می شود. شکل کلی دستور به صورت زیر است :

div src

تک عملوند دستورالعمل div مقسوم علیه تقسیم است، مقسوم بستگی به سایز عملوند یکی از دو حالت زیر ممکن است باشد :

- اگر عملوند ۸بیتی باشد ثبات AX بر عملوند تقسیم می شود. خارج قسمت تقسیم در AL و باقیمانده در AH قرار می گیرند.
- اگر عملوند ۱۶بیتی باشد مقدار ۳۲بیتی DX:AX بر عملوند تقسیم می شود. خارج قسمت تقسیم در AX و باقیمانده در DX ذخیره می شوند.

برای تقسیم بدون علامت یک عدد ۸بیتی بر یک عدد ۸بیتی دیگر مقسوم باید گسترش داده شده ۱۶ بیتی شود. برای این کار مقسوم را در ثبات AL و مقدار صفر را در ثبات AH ذخیره کنید. سپس دستور div را بنویسید. برای تقسیم بدون علامت یک عدد ۱۶بیتی بر یک عدد ۱۶بیتی دیگر باید مقسوم در ثبات AX و مقدار صفر در ثبات DX ذخیره شود .

دستور div به یکی از دو صورت زیر می تواند باشد. تقسیم بر عملوند فوری امکان ندارد و داده فوری باید در یکی از ثبات ها قرار گیرد .

div register
div memory

فلگ هایCarry، Auxiliary carry ، Overflow، Parity، Sign و Zero نامعین هستند .

نکته. اگر مقسوم علیه صفر باشد یا خارج قسمت بزرگتر از مقصد باشد خطا اتفاق می افتد و برنامه سقط می شود. معمولا پیغام "division by zero"یا "divide error" صادر می شود .

مثال. دستورات زیر حاصل عبارت J=K/M را محاسبه می کند. متغیرها ۸ بیتی و بدون علامت هستند .

```
mov AL, K
mov AH, 0
div M
mov J, AX
```

IDIV

دستورالعمل idiv (integer division) مشابه دستورالعمل div است با این تفاوت که تقسیم علامتدار را انجام می دهد .

قبل از عمل تقسیم مقدار مقسوم (AL یا AX) در صورت لزوم به طور ریاضی باید گسترش داده شوند. اینکار توسط دستورات cbwو cwd انجام می پذیرد .

مثال. دستورات زیر حاصل عبارت J := (K*M)/P را محاسبه می کنند. متغیرها ۱۶بیتی و علامتدار درنظرگرفته شده اند .

```
mov AX, K
imul M
idiv P
mov J, AX
```

جدول زیر به طور خلاصه کاربرد دستورات گسترش در عملیات ضرب و تقسیم را نشان می دهد .

ضرب	تقسیم
mov AL, byte1 imul byte2	mov AX, word1 idiv byte1
mov AX, word1 imul word2	mov AL, byte1 cbw idiv byte2
mov AL,byte1 cbw imul word1	mov AX, word1 cwd idiv word2

برنامه محاسبه ضرب و تقسیم دو عدد صحیح علامتدار توسط دستورات اسمبلی در محیط C++ تحت ویندوز.

```
#include <iostream.h>
#include <stdlib.h>

void main ()
{

   short int a,b;
   unsigned short int high_mul, low_mul;
   short int quotient, remainder;
   long   mul_result;

   cout<<"Input First Number:";
   cin>>a;
   cout<<endl<<"Input Second Number:";
   cin>>b;

   asm {
      mov ax,a
      mov bx,b

      imul bx

      mov low_mul,ax
      mov high_mul,dx

      mov ax,a
      mov bx,b
      cwd
      idiv bx
      mov quotient, ax
      mov remainder, dx
   }

   mul_result =high_mul;
   mul_result <<=16;
   mul_result +=low_mul;

   cout<<"\nMultiply Result="<< mul_result<<endl;
   cout<<"Quotient ="<< quotient <<endl;
   cout<<"Remainder ="<< remainder <<endl;
}
```