دستورات اسمبلی - دستورات جمع و تفریق

تعدادی از دستورات ٨٠٨٦ که برای انجام عملیات جمع و تفریق بر روی اعداد صحیح بکار می روند عبارتند از :

ADD
ADC
INC
SUB
SBB
DEC
NEG
CMP

ADD

این دستورالعمل حاصل جمع صحیح دو عملوند خود را محاسبه و نتیجه را در عملوند اول قرار می دهد .

add dest, src

دستورالعمل add محتوای عملوند src را با عملوند dest جمع می کند و نتیجه را در dest ذخیره می کند (dest := dest + src).

دستورالعمل add به شکل های زیر می تواند استفاده شود :

add register, register
add register, memory
add memory, register
add register, immediate data
add memory, immediate data
add AX/AL, immediate data

مثال. دستور زیر محتوای ثبات های AX و BX را جمع کرده و حاصل را در ثبات AX ذخیره می کند .

Add AX,BX

فلگ های زیر با توجه به نتیجه دستورالعمل add تاثیر می پذیرند:

- فلگ Overflow اگر یک شوددلالت بر سرریزی در محاسبات علامتدار است.
- فلگ Carry اگر یک شود دلالت بر سرریزی در محاسبات بدون علامت دارد.
- فلگ Sign اگر یک شود نشان می دهد که نتیجه منفی بوده است. یعنی با ارزش ترین بیت عدد یک است.
- فلگ Zero اگر یک شود بیان کننده این است که نتیجه جمع صفر بوده است.
- فلگ Auxiliary Carry شامل سرریزی BCD از نیبل پایین است.
- فلگ Parity با توجه به ٨ بیت پایین نتیجه تغییرمی کند. اگر تعداد بیت های یک نتیجه زوج باشد این فلگ یک می شود. و اگر تعداد فردی بیت ١ در نتیجه باشد این فلگ صفر می شود.

روی بقیه فلگ ها اثر ندارند.

نکاتی که درمورد دستور mov باید رعایت شود در مورد دستور add نیز صادق است. علاوه بر این که با این دستور نمی توان یک ثبات سگمنت را با مقداری جمع کرد .

وقتی داده فوری با یک عملوند بایت یا کلمه جمع می شود داده فوری به اندازه عملوند گسترش پیدا می کند .

چون جمع حافظه و حافظه وجود ندارد اگر بخواهید دو متغیر را با هم جمع کنید باید عملوندهای حافظه را در ثبات ها منتقل کنید .

مثال. یک حالت ممکن برای انجام عمل جمع متغیرها به صورت J := K + M + N + P; می تواند به شکل زیر باشد :

```
mov  BX, K
mov  AX, M
add  BX, N
add  AX, P
add  AX, BX
mov  J, AX
```

مثال. می توان یک ثبات را با محلی از حافظه جمع کرد. اجباری نیست که هردو عملوند ثبات باشد .

```
mov  AX, K
add  J, AX
```

مثال. می توان یک مقدار ثابت را با حافظه جمع کرد.

```
add  J, 2
```

ADC

دستورالعمل adc (add with carry) مشابه دستورالعمل add است با این تفاوت که حاصل جمع دو عملوند و فلگ Carry را محاسبه می کند (dest :=dest+source+CF). بنابراین اگر Carry صفر باشد نتیجه مشابه add می شود .

INC

دستورالعمل inc (increment) یک واحد به عملوند خود اضافه می کند. شکل کلی آن به صورت زیر است :

```
inc dest
```

این دستورالعمل عدد ۱ را با dest جمع و حاصل را در خود dest ذخیره می کند .

دستور inc به شکل های زیر می تواند باشد:

```
inc register
inc memory
```

عملوند دستور می تواند ثبات یا مکانی از حافظه باشد. اندازه عملوند می تواند ۸ یا ۱۶ بیتی باشد .

دستور inc فشرده تر و اغلب سریع تر از دستور add است .

به استثنای فلگ Carry بقیه فلگ ها مشابه دستورالعمل add تغییر می کنند .توجه کنید که این دستور بر روی فلگ Carry تاثیر ندارد و برای تاثیر روی فلگ Carry باید از دستورالعمل ADD استفاده شود .

افزایش شمارنده حلقه و اندیس آرایه یکی از متداولترین کاربردهای دستور inc است .

SUB

دستورالعمل sub (subtract) حاصل تفریق عملوند دوم از عملوند اول را محاسبه می کند. شکل کلی آن به صورت زیر است:

```
sub dest, src
```

دستورالعمل sub مقدار src را از dest کم کرده حاصل را در dest ذخیره می کند .

مشابه دستورالعمل add ، دستور sub به صورت های زیر می تواند باشد:

```
sub register, register
sub register, memory
sub memory, register
sub register, immediate data
sub memory, immediate data
sub AX/AL, immediate data
```

دستور sub به طریق زیر فلگ ها را تغییر می دهد:

- اگر نتیجه صفر شود فلگ Zero یک می شود. این در حالتی اتفاق می افتد که عملوندها با هم برابر باشند.
- اگر نتیجه منفی شود فلگ sign یک می شود.
- اگر سرریزی رخ دهد فلگ overflow یک می شود.
- فلگ Auxiliary Carry در صورت نیاز برای عملیات BCD یک می شود.
- فلگ Parity با توجه به تعداد بیت های یک نتیجه تنظیم می شود.
- فلگ Carry در صورت بروز سرریزی در محاسبات بدون علامت یک می شود.

توجه داشته باشید که تفریق خاصیت جابجائی ندارد.

مثال. دستورات زیر عمل J := J - K; را انجام می دهند.

```
mov  AX, K
sub  J, AX
```

مثال. دستورات زیر عمل J := K - J; را انجام می دهند.

```
mov  AX, K
sub  AX, J
mov  J, AX
```

بعد از عمل تفریق از مقادیر فلگ های Carry ، Sign، Overflow و Zero می توان برای بررسی مساوی، نامساوی، بزرگتر یا کوچکتر بودن هر عملوند با دیگری استفاده کرد. جزئیات بیشتر در دستور cmp گفته خواهد شد .

SBB

دستورالعمل sbb (subtract with borrow) مشابه دستور sub است با این تفاوت که حاصل تفریق عملوند دوم و CF از عملوند اول را محاسبه می نماید(dest:=dest-src-CF).

DEC

دستورالعمل dec (decrement) یک واحد از عملوند خود کم می کند و حاصل را در خود عملوند ذخیره می نماید.

```
dec dest
```

عملوند دستور dec می تواند ثبات یا حافظه باشد.

به استثنای فلگ Carry بقیه فلگ ها مشابه دستورالعمل sub تغییر می کنند.

NEG

دستورالعمل neg (negate) مکمل۲ عملوند خود را محاسبه می کند. فرم کلی ان به صورت زیر است:

```
neg dest
```

دستور neg حاصل تفریق تنها عملوند خود را از عدد صفر را محاسبه کرده) عملوند را منفی می کند) و نتیجه را در آن ذخیره می کند. درنتیجه اجرای دستور علامت عملوند عکس می شود .

عملوند دستور neg می تواند ثبات یا محلی از حافظه باشد:

```
neg register
neg memory
```

این دستور روی فلگ ها به صورت زیر تاثیرمی گذارد:

- اگر نتیجه برابر با صفر شود فلگ Carry صفر و در غیر این صورت یک می شود. اگر عملوند صفر بوده باشد دستور اثری روی آن نمی گذارد ولی فلگ Carry را صفر می کند. منفی کردن هر مقدار دیگر فلگ Carry را یک می کند.
- اگر عملوند یک بایتی و حاوی مقدار 128- باشد و یا دوبایتی و حاوی عدد 32768- باشد، منفی کردن عملوند را تغییر نمی دهد اما فلگ Overflow را یک می کند.
- روی فلگ های S، P و Z مانند دستور Sub اثر می گذارد.

مثال. دستور زیر علامت متغیر J عکس می شود.

```
neg J
```

مثال. دستورات اسمبلی زیر مکمل K را محاسبه و در J ذخیره می کند.(J=-k;)

```
mov AX, K
neg AX
mov J, AX
```

CMP

دستوالعمل cmp (compare) مانند دستور sub است با این تفاوت که حاصل تفریق را ذخیره نمی کند .(dest-src) نحوه کلی آن به صورت زیر است :

```
cmp dest.src
```

به صورت های زیر می تواند استفاده شود:

```
cmp register, register
cmp register, memory
cmp memory, register
cmp register, immediate data
cmp memory, immediate data
cmp AX/AL, immediate data
```

فلگ ها مانند دستورالعمل sub با توجه به نتیجه تفریق تغییر می کنند.

دستور زیر را درنظر بگیرید:

```
cmp AX, BX
```

این دستور حاصل AX-BX را محاسبه می کند و با توجه به حاصل فلگ ها را تنظیم می کند. فلگ ها به صورت زیر تغییر می کنند و می توانند برای بررسی نتیجه مقایسه بکار برده شوند :

- فلگ Zero یک می شود اگر AX=BX باشد. مساوی یا نامساوی بودن دو عملوند را مشخص می کند.
- فلگ Carry وقتی یک می شود که در محاسبات بدون علامت AX<BX باشد.یعنی تفریق BX از AX احتیاج به رقم قرضی داشته باشد.
- فلگ Sign به همراه فلگ Overflow در محاسبات علامتدار نشان می دهد کدام عملوند بزرگتر است.

دستورالعمل cmp روی فلگ های Parity و Auxiliary Carry هم تاثیر دارد ولی بندرت هنگام مقایسه مورد بررسی قرار می گیرند. به طور خلاصه برای مقایسه دو عملوند با توجه به فلگ های زیر می توان نتیجه گیری کرد :

عملوندهای بدون علامت	فلگ ها
AX = BX	Z=1
AX ≠ BX	Z=0
AX < BX	C=1
AX >= BX	C=0
عملوندهای علامتدار	**فلگ ها**
AX = BX	Z=1
AX ≠ BX	Z=0
AX < BX	(S=0 and O=1)or(S=1 and O=0)
AX >= BX	(S=0) and (O=0) or (S=1) and (O=1)