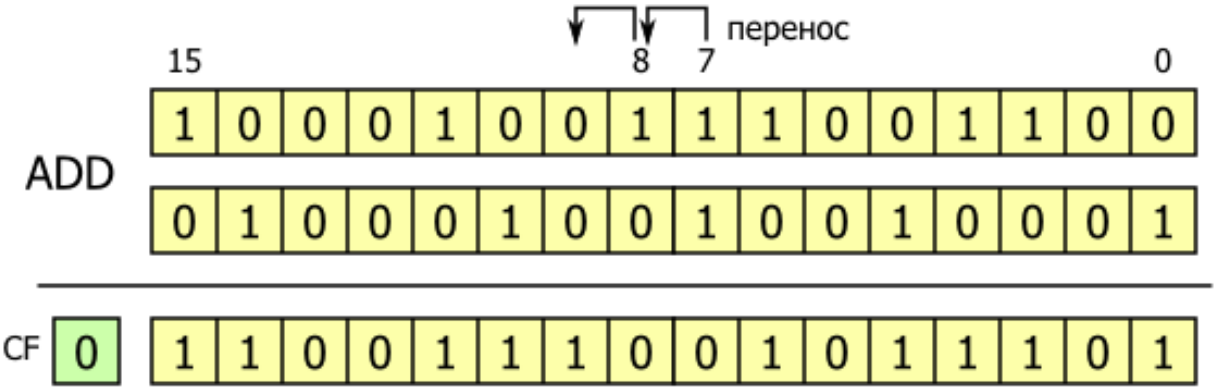
Учебный курс. Часть 10. Сложение и вычитание с переносом

Автор: xrnd | Рубрика: Учебный курс | 30-03-2010 | Распечатать запись

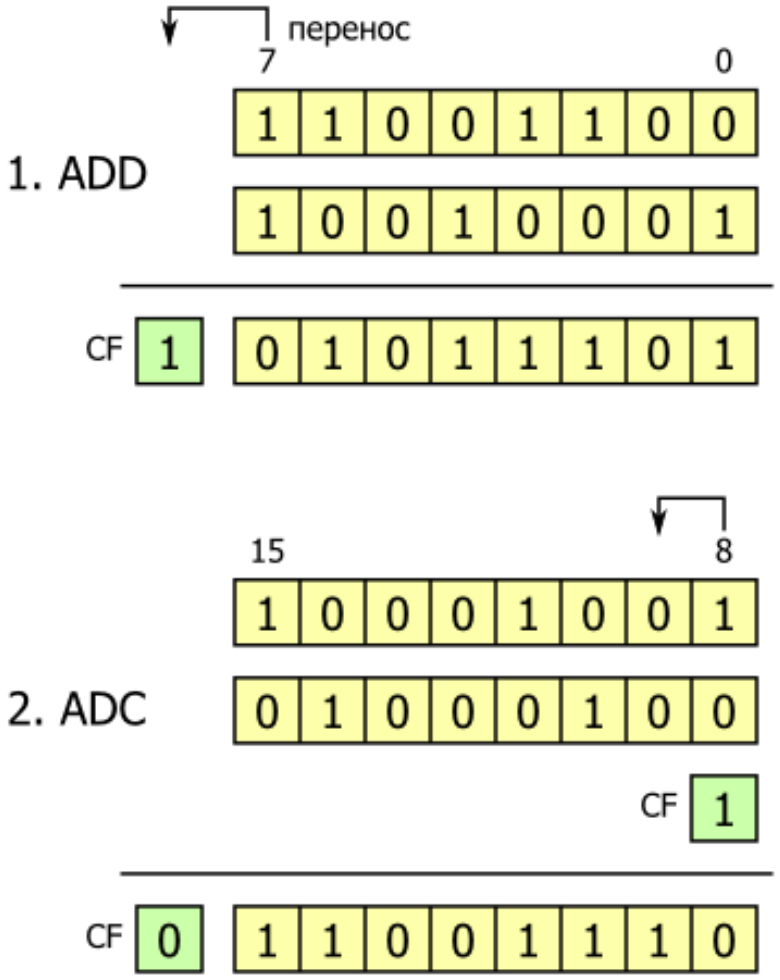
В системе команд процессоров x86 имеются специальные команды сложения и вычитания с учётом флага переноса (*CF*). Для сложения с учётом переноса предназначена команда ADC, а для вычитания — SBB. В общем, эти команды работают почти также, как ADD и SUB, единственное отличие в том, что к младшему разряду первого операнда прибавляется или вычитается дополнительно значение флага *CF*.

Зачем нужны такие команды? Они позволяют выполнять сложение и вычитание многобайтных целых чисел, длина которых больше, чем разрядность регистров процессора (в нашем случае 16 бит). Принцип программирования таких операций очень прост — длинные числа складываются (вычитаются) по частям. Младшие разряды складываются(вычитаются) с помощью обычных команд ADD и SUB, а затем последовательно складываются(вычитаются) более старшие части с помощью команд ADC и SBB. Так как эти команды учитывают перенос из старшего разряда, то мы можем быть уверены, что ни один бит не потеряется 🙂 Этот способ похож на сложение(вычитание) десятичных чисел в столбик.

На следующем рисунке показано сложение двух двоичных чисел командой ADD:



При сложении происходит перенос из 7-го разряда в 8-й, как раз на границе между байтами. Если мы будем складывать эти числа по частям командой ADD, то перенесённый бит потеряется и в результате мы получим ошибку. К счастью, перенос из старшего разряда всегда сохраняется в флаге CF. Чтобы прибавить этот перенесённый бит, достаточно применить команду ADC:



Аналогичная ситуация возникает с вычитанием чисел по частям. Чтобы было совсем понятно, приведу пример программы. Допустим, требуется вычислить значение формулы *k=i+j-n+1*, где переменные *k*, *i*, *j* и *n* являются 32-битными целыми числами без знака. Складывать и вычитать такие числа придётся в два этапа: сначала вычисления будут производиться с младшими словами операндов, а затем со старшими с учётом переноса.

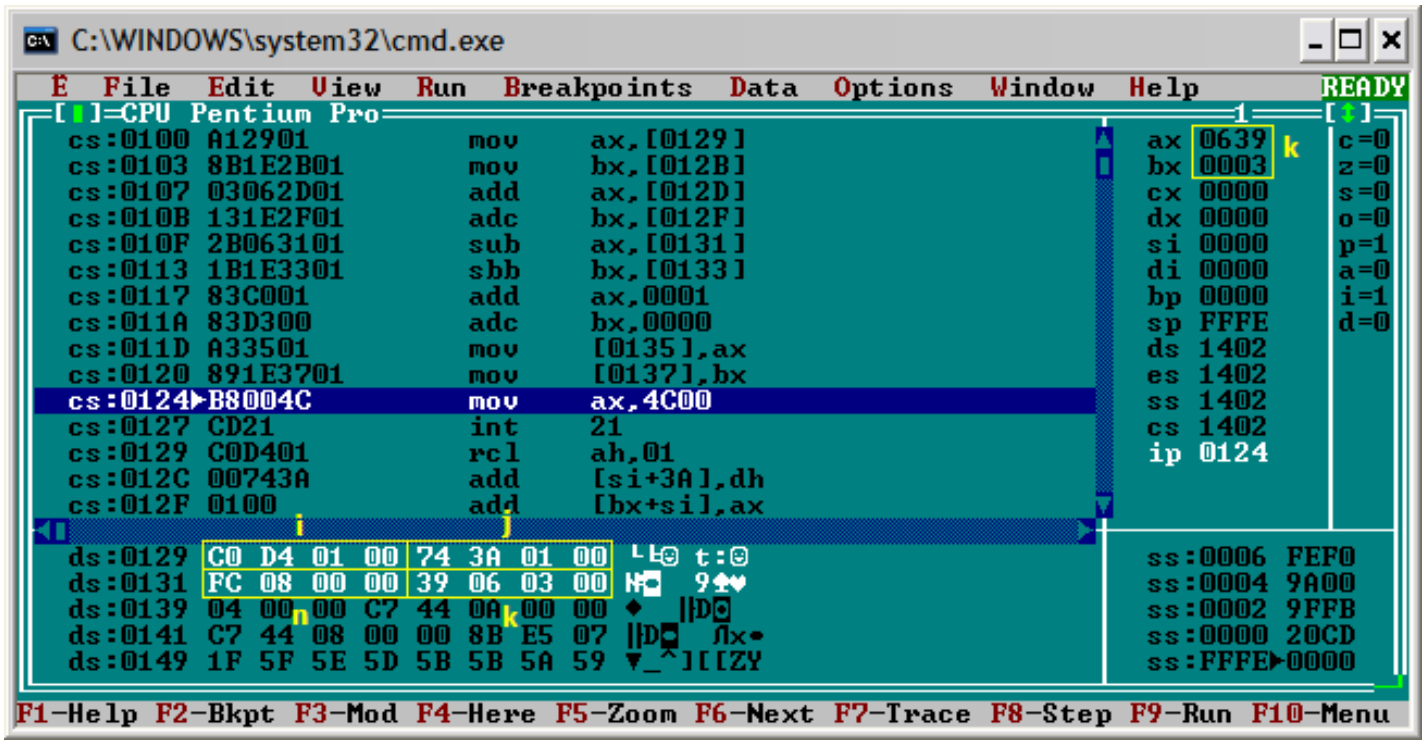
Для прибавления единицы в данном примере нельзя использовать команду INC, так как она не влияет на флаг *CF* и мы можем получить ошибку в результате!

```
use16                ;Генерировать 16-битный код
org 100h             ;Программа начинается с адреса 100h

    mov ax,word[i]   ;Загружаем младшую часть i в AX
    mov bx,word[i+2] ;Загружаем старшую часть i в BX

    add ax,word[j]   ;Складываем младшие части i и j
    adc bx,word[j+2] ;Складываем старшие части i и j

    sub ax,word[n]
    sbb bx,word[n+2] ;BX:AX = i+j-n

    add ax,1         ;Команда INC здесь не подходит!
    adc bx,0         ;BX:AX = i+j-n+1

    mov word[k],ax   ;\
    mov word[k+2],bx ;/ Сохраняем результат в k

```

```
19      mov ax,4C00h        ;\
20      int 21h             ;/ Завершение программы
21  ;-------------------------------------------------------
22  i dd 120000
23  j dd  80500
24  n dd   2300
25  k dd      ?
```

Запись *word[i]* означает, что мы переопределяем размер переменной (она объявлена как DWORD) и обращаемся к младшему слову. Старшее слово расположено в памяти после младшего, поэтому к адресу переменной надо прибавить 2, и соответствующая запись будет иметь вид *word[i+2]*.

Посмотреть работу программы можно в отладчике:



Обратите внимание, как хранятся переменные в памяти. В процессорах Intel младший байт всегда хранится по младшему адресу, поэтому получается, что в окне дампа значения надо читать справа налево. В регистрах же числа записываются в нормальном виде. Сравните, как выглядит одно и то же значение *k* в памяти и в регистрах (старшая часть находится в BX, а младшая — в AX).

Одно из преимуществ ассемблера в том, что на нём можно реализовать работу с собственными форматами чисел, например с очень длинными целыми. А в языках высокого уровня выбор всегда ограничен компилятором. Следующая программа складывает два 7-байтных значения (для разнообразия я использовал только один регистр).

```
1  use16                ;Генерировать 16-битный код
2  org 100h             ;Программа начинается с адреса 100h
3
4      mov ax,word[x]
5      add ax,word[y]
6      mov word[z],ax
7
8      mov ax,word[x+2]
9      adc ax,word[y+2]
10     mov word[z+2],ax
```

```
11
12      mov ax,word[x+4]
13      adc ax,word[y+4]
14      mov word[z+4],ax
15
16      mov al,byte[x+6]
17      adc al,byte[y+6]
18      mov byte[z+6],al
19
20      mov ax,4C00h      ;\
21      int 21h           ;/ Завершение программы
22  ;-------------------------------------------------------
23  x dd 0xF1111111
24    dw 0xF111
25    db 0x11
26  y dd 0x22222222
27    dw 0x2222
28    db 0x22
29  z rb 7
```

Обращение к старшему байту записывается как *byte[x+6]*. Наверно, вы уже догадались, почему 🙂 Команда MOV не меняет состояние флагов, поэтому её можно ставить между командами сложения.

Упражнение

Напишите программу для вычисления формулы *d=b-1+a-c*. Все числа — 3-х байтные целые без знака. Проверьте работу программы в отладчике. Результаты можете выкладывать в комментариях.

Следующая часть »

Комментарии:

RoverWWWorm
04-04-2010 20:35

;d=b-1+a-c. Все числа – 3-х байтные целые без знака
use16
org 100h

mov ax,word[a]
mov bx,word[a+2]

sub ax,word[c]
sbb bx,word[c+2]

add ax,word[b]
adc bx,word[b+2]

sub ax,1
sbb bx,0

mov word[d],ax
mov word[d+2],bx

```
mov ax,4c00h
int 21h
;_____________
a dw 30000
b dw 2000
c dw 100

d dw ?

;в директивах объявления данных запутался блин, думаю в проге ошибки..
кстати, хотел ввести, например : a dw 1777777 выходит ошибка, вне диапазона, но вроде дипазон dw(двойное слово) от -2 147 483 648 до 2 147 483 647 ???
```

[Ответить]

xrnd
05-04-2010 03:05

Для начала нужно правильно объявить 3-х байтные числа. У тебя все числа объявлены как двухбайтные. Их можно объявить либо как 3 байта подряд, либо как комбинацию из слова и байта:

```
a dw 123
db 45
```

И складывать сначала слово (первые 2 байта). а потом оставшийся 3-й байт.
У тебя получается что word[a+2] — это тоже самое, что word[b], потому что a занимает 2 байта в памяти, а потом за ним идёт b.

dw 1777777 — получается ошибка, потому что ты объявляешь слово, а максимальное значение для слова 65535.

[Ответить]

RoverWWWorm
05-04-2010 12:14

```
;может так
use16
org 100h

mov ax,word[a]
mov bh,byte[a+2]

sub ax,word[c]
sbb bh,byte[c+2]

add ax,word[b]
adc bh,byte[b+2]

sub ax,1
sbb bh,0

mov word[d],ax
mov byte[d+2],bh
```

```
mov ax,4c00h
int 21h
;_____________
a dw 30000
db 35

b dw 2000
db 25

c dw 100
db 15

d rb 3
```

[Ответить]

xrnd
05-04-2010 17:12

Замечательно! На этот раз всё правильно.

[Ответить]

RoverWWWorm
05-04-2010 12:19

кстати, а как число, объявить как 3 байта подряд

[Ответить]

xrnd
05-04-2010 17:16

Очень просто, как массив
a db 56h, 34h, 12h ;a = 123456h (младший байт пишется первым)

или можно в десятичном виде:
b db 3,2,1 ; b = (1 * 65536 + 2 * 256 + 3)

[Ответить]

RoverWWWorm
05-04-2010 17:32

Ура!
Жду следущих уроков.

[Ответить]

mc-black
11-05-2010 20:04

Что-то никогда не задумывался, как вычисляются целые длиной больше, чем разрядность регистров. Раньше полагал, что там какой-то хитрый алгоритм, а оказалось, что куда уж проще! =)

[Ответить]

xrnd
12-05-2010 00:08

Кстати, это полезная вещь. Например, есть 8-битные микроконтроллеры и целые числа больше байта там только так и складываются. Команды по-другому называются, но принцип тот же.

[Ответить]

mc-black
11-05-2010 23:10

Какими BBCODE можно здесь оформлять код?

[Ответить]

xrnd
12-05-2010 00:05

Хм. Я так понял, они вообще тут не поддерживаются :)))
Можешь писать теги HTML, вот так: <pre lang="asm" line="1">код</pre>

[Ответить]

IgorKing
27-08-2010 12:17

В записи word[i], i это адрес старшего байта слова, а i+1 младшего, так?

[Ответить]

xrnd
27-08-2010 12:30

Не совсем так.
word обозначает, что выражение в квадратных скобках используется как адрес слова. Это нужно, например, если переменная i объявлена как байт (с помощью директивы db) или как двойное слово (dd).

byte обозначает, что выражение используется как адрес байта.
Тогда для байтов в слове будет так:
byte[i] — младший байт слова
byte[i+1] — старший байт слова.
(Младший байт в архитектуре Intel располагается по младшему адресу в памяти).

[Ответить]

VanOk
27-10-2010 13:41

Извини пожалуйста за глупый вопрос а как разделять 3байт. числа на слово и байт
;————————

a dw 30000
db 35

b dw 2000
db 25

c dw 100
db 15

d rb 3
чет я не понял
Надеюсь мне ответят

[Ответить]

IgorKing
27-10-2010 14:31

Запись не понял?
Например такую:

a dw 30000
db 35

Можно, для удобства, заменить на:

a dw 7530h
db 1Dh

Тогда в памяти это будет выглядеть так: 30 75 1D
a — это адрес младшего байта твоего числа (т.е. 30h)
A dw,db просто записывают значения в память.

[Ответить]

VanOk
27-10-2010 16:41

Извини пожалуйста за глупый вопрос а как разделять 3байтовые числа на слово и байт
;_____
a dw 30000
db 35

b dw 2000
db 25

c dw 100
db 15

d rb 3
чет я не понял

[Ответить]

xrnd
27-10-2010 21:29

Ты имеешь в виду, как задать определённое значение такому числу?

Дело в том, что специальной директивы для 3-х байтных чисел нет. Можно объявить либо 3 байта друг за другом, либо слово и байт, либо байт и слово.

В твоём варианте объявлено слово и байт. Слово — это 16 младших битов числа, а байт — 8 старших. В шестнадцатеричной системе получается просто, а в десятичной сложнее.

Допустим, мы хотим присвоить такому числу значение 200 000.
Тогда старшая часть будет равна 200 000 / 65536 = 3 (без дробной части)
младшая часть будет равна 200 000 % 65536 = 3 392 (остаток от деления)

```
x dw 3392 ; x = 200000
db 3
```

Вообще, конкретные значения чисел тут не важны. Главное — написать код для действий с ними.

[Ответить]

VanOk
02-11-2010 00:29

```
;d=b-1+a-c
use16
org 100h
mov ax,word[b]
mov bl,byte[b+2]
sub ax,1
sbb bl,0
add ax,word[a]
adc bl,byte[a+2]
sub ax,word[c]
sbb bl,byte[c+2]
mov word[d],ax
mov byte[d+2],bl

mov ax,4C00h
int 21h
;---------------------------------
b dw 1000
db 15
a dw 3300
db 8
c dw 1100
db 14
d rb 3
```

;кажется получилось спосибо огромное за верхне обяснение и за «Вообще, ;конкретные значения чисел тут не важны. Главное – написать код для действий ;с ними.»

[Ответить]

xrnd
02-11-2010 19:17

Всё правильно, молодец 🙂

[Ответить]

oleg
16-11-2010 00:18

Спасибо за ответы на предыдущие мои коменты. Ну и за сам курс Асма. 🙂

имхо, в этом уроке не хватает картинок по флагу переноса и действию команд adc sbb, трудно в уме всё это понять. хотя чуток подумать, то понять можно.

Ну и вернуться к предыдущему уроку «Часть 5. Директивы объявления данных» и туда тоже картинок с ячейками памяти и наполнением их значенями. Для полноты картины.

А так, Отлично.

[Ответить]

xrnd
16-11-2010 03:59

Это хорошая идея. Добавлю картинки в ближайшее время.

Если есть ещё пожелания, замечания, предложения — не стесняйся писать. Мне очень важна обратная связь. На ассемблере я программирую давно, поэтому не всегда знаю, что может быть непонятно 🙂

[Ответить]

argir
28-11-2010 09:09

При вычитании не может быть переноса из старшего разряда.

Можно тогда заменить в ответах

sub ax,1
sbb bl,0

на
dec ax ?

[Ответить]

argir
28-11-2010 09:49

Да уже сам разобрался, что нельзя. Флаг CF выставляется не только при переполнении, но и при вычитании большего числа из меньшего.

[Ответить]

xrnd
28-11-2010 14:38

При вычитании большего числа из меньшего тоже возникает перенос, потому что в процессоре вычитание реализуется с помощью сложения 🙂

[Ответить]

Борис
16-12-2010 23:23

Приветствую, спасибо за занятия, сначала с трудом вкурил о порядке старших байтов, но потом вроде с помощью калькулятора виндового все встало на свои места. Вот моя версия считалки:

```
use16 16-bit cod
org 100h ;begin 100h
;d=b-1+a-c
mov ax, word[a]
mov bl, byte[a+2]

sub ax, word[c] ;
sbb bl, byte[c+2]; =a-c

add ax,word[b]
adc bl,byte[b+2] ;=a-c+b

sub ax,0x0001
sbb bl,0x0000 ;=a-c+b-1

mov word[d], ax
mov byte[d+2], bl

mov ax,4C00h ;\
int 21h ;/ end
;————————————————————————————-
a dw 0xff00
db 0xff
b dw 0x0023
db 0x01
c dw 0x3443
db 0x8
d dw ?
```

[Ответить]

xrnd
17-12-2010 17:02

Хорошая считалка.

А FASM не ругается на эту команду?

```
    sbb bl,0x0000
```

Из 8-битного регистра вычитается 16-битный шестнадцатеричный ноль O_o

[Ответить]

Борис
17-12-2010 21:30

Не, не ругается :-), и в дебугере работвет

[Ответить]

Борис
17-12-2010 21:34

и еще нашел баг, d 2-х байтное, а 3-й в воздухе висит !!!

[Ответить]

xrnd
17-12-2010 21:46

Наверно, 0 он и есть ноль, размер определяется по регистру 🙂

Конкретно в этой программе это даже не баг, 3-й байт там всегда будет 🙂

[Ответить]

Philin
13-01-2011 18:01

Привет… и что я тут намудрил…?))

```
use16
org 100h
;d=b-1+a-c

mov ax,word[b]
mov bx,word[b+2]
mov cl,byte[b+4]

sub ax,1
sbb bx,0

add ax,word[a]
adc bx,word[a+2]
adc cl,byte[a+4]

sub ax,word[c]
sbb bx,word[c+2]
sub cl,byte[c+4]

mov word[d],ax
mov word[d+2],bx
mov byte[c+4],cl
```

```
mov ax,4c00h
int 21h
;————————-
b dw 0xBBBB
db 0xBB
a dw 0xAAAA
db 0xAA
c dw 0xCCCC
db 0xCC
d db ?
```

[Ответить]

xrnd
15-01-2011 00:55

Привет. У тебя переменные 3-х байтные, а код выполняет действия с 5-байтными числами.
d лучше тоже объявить как 3 байта, а не один.

[Ответить]

Philin
02-02-2011 13:55

Привет сансэй..)) с первого раза не дошло…

```
use16
org 100h

mov ax,word[b]
mov bl,byte[b+2]

sub ax,1
sbb bl,0

add ax,word [a]
adc bl,byte [a+2]

sub ax,word [c]
sbb bl,byte [c+2]

mov word [d], ax
mov byte [d+2],bl

mov ax,4c00h
int 21h
;————————————
b dw 0xBBBB
db 0xBB
a dw 0xAAAA
db 0xAA
c dw 0xCCCC
db 0xCC
d rb 3
```

есть вопрос такой,

mov ax,word[b]
mov bl,byte[b+2]

sub ax,1
sbb bl,0

add ax,word [a]
adc bl,byte [a+2]

mov cx, word [c]
mov dl, byte [c+2]

mov word [d], ax-cx
mov byte [d+2],bl-dl

вопрос заключается в двух последних строчках, может глупо, но любопытно (расчитывал что код покороче станет, ан нет)… спасибо…

[Ответить]

xrnd
09-02-2011 15:12

Привет 🙂

Да, теперь всё правильно.

Насчет двух последних строчек — ну это же не бэйсик, а ассемблер.
Каждая строка кода соответствует машинной команде.
А в процессоре просто нет такой команды, которая вычитает регистры и записывает результат в память. Для этого нужно сначала вычесть командой SUB, а потом поместить результат в память командой MOV.

```
    sub ax,cx
    mov word [d], ax
```

[Ответить]

Knight212
03-02-2011 23:33

Привет. Спасибо за занятия.

use16
org 100h

mov ax, word[b]
sub ax, 1
add ax, word[a]
sub ax, word[c]
mov word[d], ax

```
mov al, byte[b+2]
sbb al, 0
adc al, byte[a+2]
sbb al, byte[c+2]
mov byte[d+2], al

mov ax, 4C00h
int 21h

a dw 1111h
db 11h
b dw 5555h
db 0DDh
c dw 3333h
db 99h
d rb 3
```

[Ответить]

xrnd
09-02-2011 15:22

Привет.

Все команды правильные, а вот их порядок — нет.
Команды SBB и ADC должны выполняться после соответствующих SUB и ADD.
Перенос из младшей части сохраняется в флаге CF и он будет потерян при выполнении любой команды, изменяющей флаг CF.

```
mov ax, word[b]
sub ax, 1       ; Заём из старшего разряда в флаге CF.
add ax, word[a] ; Флаг CF будет изменен командой ADD :(
```

[Ответить]

A13R42
12-02-2011 22:42

Здравствуй. Вот, намудрил что-то, проверь пожалуйста =)

```
;——————————————
use16
org 100h

mov ah,byte[b]
mov bx,word[b+1]

sub ah,1
sbb bx,0

add ah,byte[a]
adc bx,word[a+1]

sub ah,byte[c]
sbb bx,word[c+1]
```

```
mov byte[d],ah
mov word[d+1],bx

mov ax,0x4C00
int 0x21
;_____________
a db 0xFF, 0xAA, 0xAB
b db 0x22, 0x44, 0x99
c db 0x66, 0x55, 0x10
d rb 3
```

[Ответить]

xrnd
14-02-2011 12:55

Привет.
Ошибок нет, код написан правильно.

[Ответить]

plan4ik
16-03-2011 04:37

xrnd пожалуйста исправь код в уроке

```
mov ax,word[i] ;Загружаем младшую часть i в AX
mov bx,word[i+2] ;Загружаем старшую часть i в BX

add ax,word[j] ;Складываем младшие части i и j
adc bx,word[j+2] ;Складываем старшие части i и j

sub ax,word[n]
sbb bx,word[n+2] ;BX:AX = i+j-n
```

на:

```
mov ax,word[i] ;Загружаем младшую часть i в AX
mov bx,word[i+1] ;Загружаем старшую часть i в BX

add ax,word[j] ;Складываем младшие части i и j
adc bx,word[j+1] ;Складываем старшие части i и j

sub ax,word[n]
sbb bx,word[n+1] ;BX:AX = i+j-n
```

а то я почти себя убедил что в fasm’е можно передвигатся только по байтам
что бы другие не попадались в похожее заблуждение … 😉

С уважением aka plan4ik

[Ответить]

xrnd
22-03-2011 16:39

Мой код был правильным.

i, j, n — объявлены как dd, то есть 4 байта.
Сначала складывается по 2 младших, потом по 2 старших.
Поэтому смещение +2

Если сделать как у тебя i+1, j+1, n+1, то получится ошибка
mov bx,word[i+1] поместит в bx два байта из середины двойного слова, а не старшую часть. 2 и 3 байт вместо 3 и 4.

[Ответить]

plan4ik
30-03-2011 04:18

а я чета подумал что рас уж я указал указатель на слово то и перемещаюсь со смещением по 16 байт 😉

Спасибо, учту на будущее ….

[Ответить]

plan4ik
30-03-2011 13:07

и мой код

```
use16
org 100h

mov al, byte [a+2]
sub al, byte [c+2]
mov [d+2], al

mov al, byte [a+1]
sbb al, byte [c+1]
mov [d+1], al

mov al, byte [a]
sbb al, byte [c]
mov [d], al

dec [b+2]

mov al, byte [b+2]
add al, byte [d+2]
mov [d+2], al

mov al, byte [b+1]
adc al, byte [d+1]
mov [d+1], al

mov al, byte [b]
adc al, byte [d]
mov [d], al
```

```
mov ax, 4c00h
int 21h
;================================
;d=b+(a-c)-1 d = 0x91928

d rb 3
a db 0x0A, 0x1a, 0x2A
b db 0x0B, 0x1b, 0x2B
c db 0x0C, 0x1c, 0x2C
```

[Ответить]

Pecenita
24-03-2011 10:44

Здравствуйт! Нужна очень программка SBB для вычитания чисел BAh,2Eh,26h,22h,18h…друг с другом

Адрес Машинный код Мнемоника Комментарий
8000 21 LXI H,8025H
8001 25
8002 80
8003 79 MOV A,C
8004 96 SBB M
8005 32 STA 8125H
8006 25
8007 81
8008 23 INX H
800A 7A MOV A,D
800B 32 STA 8126H
800C 26
800D 81
8008 E7 RST4

помогите ее доделать, плиз, чтобы она работала (это на учебном устройстве Электроника-580)

[Ответить]

Pecenita
24-03-2011 10:45

просто, как мне сказали, там должна быть команда STC=1…где?

[Ответить]

xrnd
25-03-2011 14:25

Как должны вычитаться числа?
И где они хранятся? Я так понял, что они должны быть по адресу 8025H.

[Ответить]

Alexandr1230
24-03-2011 16:40

А вот мой вариант… Он получился больше, но использует только AX :

```
use16
org 100h

mov ax,[a]
sub ax,[c]
mov [a],ax

mov al,byte[a+2]
sbb al,byte[c+2]
mov byte[a+2],al

mov ax,[b]
sub ax,1
mov [b],ax

mov al,byte[b+2]
sbb al,0
mov byte[b+2],al

mov ax,[a]
add ax,[b]
mov [d],ax

mov al,byte[a+2]
adc al,byte[b+2]
mov byte[d+2],al

mov ax,4c00h
int 21h

a dw 2200h
db 44h
b dw 3300h
db 66h
c dw 1100h
db 22h
d dw ?
db ?
```

[Ответить]

Pecenita
27-03-2011 17:05

а что такое ax, al, dw, int

и что значит?

```
a dw 2200h
db 44h
```

b dw 3300h
db 66h
c dw 1100h
db 22h
d dw ?
db ?

[Ответить]

xrnd
01-04-2011 13:27

ax и al — названия регистров
dw — директива для объявления слова.
int — команда программного прерывания. Используется для вызова функций DOS.

Код объявляет 4 трехбайтные переменные: a, b, c и d.

[Ответить]

xrnd
01-04-2011 13:25

Хороший вариант, если остался один свободный регистр 😉

[Ответить]

mustdie
24-03-2011 18:07

use16
org 100h
mov ax,word[b]
mov bh,byte[b+2]
sub ax,1
add ax,word[a]
adc bh,byte[a+2]
sub ax,word[c]
sbb bh,byte[c+2]
mov word[d],ax
mov byte[d+2],dh
mov ax,4C00h
int 21h
;——data——
a db 3 dup(?)
b db 3 dup(?)
c db 3 dup(?)
d db 3 dup(?)

я потерян для общества?

[Ответить]

Pecenita
28-03-2011 22:08

Здравствуйте! Вот такая задача есть: нужна для любого массива чисел, в котором может быть больше 255 чисел, программка, программка, перемещающая этот массив с адреса??? какие идеи есть?

[Ответить]

Ваш комментарий

Имя *

Почта (скрыта) *

Сайт

Добавить

Уведомить меня о новых комментариях по email.

Уведомлять меня о новых записях почтой.