

Учебный курс. Часть 24. Команды управления флагами

Автор: xrnd | Рубрика: [Учебный курс](#) | 10-09-2010 |  [Распечатать запись](#)

В предыдущей части учебного курса мы использовали флаг CF, чтобы вернуть из процедуры информацию об ошибке. Чтобы у вас сложилась полная картина, я решил в этой части подробнее рассказать о командах управления флагами.

Как вы, наверно, помните флаги изменяются в результате выполнения арифметических и логических команд, а также команд сдвига. Регистр флагов можно сохранить в стек с помощью команды [PUSHF](#) и восстановить из стека с помощью команды [POPF](#). Кроме того, в процессоре существуют специальные команды, которые позволяют явно установить или сбросить флаги CF, DF и IF. Это очень простые команды: у них нет операндов и результатом является только изменение значения соответствующего флага.

Флаг переноса CF

Команда [CLC](#) сбрасывает флаг CF.

Команда [STC](#) устанавливает флаг CF в единицу.

Команда [CMC](#) инвертирует значение флага CF.

Флаг направления DF

Этот флаг определяет направление обработки данных цепочечными командами (о них подробно расскажу в отдельной статье). Он должен устанавливаться или сбрасываться перед использованием этих команд.

Команда [CLD](#) сбрасывает флаг DF.

Команда [STD](#) устанавливает флаг DF в единицу.

Флаг прерывания IF

Этот флаг определяет, разрешены в данный момент прерывания или нет (о прерываниях тоже будет отдельная статья).

Команда [CLI](#) сбрасывает флаг IF (запрещает прерывания).

Команда [STI](#) устанавливает флаг IF в единицу (разрешает прерывания).

Команды LAHF и SAHF

Команда [LAHF](#) загружает младший байт регистра флагов в АН. Её удобно использовать, когда нужно получить значения сразу нескольких флагов. Порядок расположения флагов представлен на рисунке:

7	6	5	4	3	2	1	0
SF	ZF	0	AF	0	PF	1	CF

Команда [LAHF](#) выполняет обратную операцию — загружает содержимое АН в младший байт регистра флагов. Это позволяет одновременно изменить значения нескольких флагов. При этом биты 1, 3, 5 регистра АН игнорируются.

Пример программы

В качестве примера использования команды [LAHF](#) я написал процедуру, которая получает и выводит на консоль значения флагов SF, ZF, AF, PF и CF. Подобную процедуру можно использовать в отладочных целях. Достаточно добавить её вызов в то место программы, где хочется проверить состояние флагов.

Чтобы выводить значения отдельных битов удобно написать отдельную процедуру для печати флага CF в виде символа. Нужный бит будет помещаться в флаг CF при помощи команды сдвига [SHL](#).

```
;Процедура вывода значения флага CF в виде символа
print_cf:
    push ax
    push dx
    mov ah,2                ;Функция DOS 02h - вывод символа
    mov dl,'0'              ;DL = '0'
    adc dl,0                 ;Если CF = 1, то в DL будет символ '1'
    int 21h                 ;Обращение к функции DOS
    pop dx
    pop ax
    ret
```

А вот собственно процедура вывода флагов. Как видите, здесь нет ничего сложного 😊 Процедура сохраняет регистр флагов и восстанавливает его перед возвратом управления, поэтому можно вызывать её где угодно, не волнуясь, что она нарушит выполнение основной программы.

```
;Процедура вывода состояния флагов на консоль
print_flags:
    push ax
    push cx
    push dx
    pushf                   ;Сохранение регистра флагов
    lahf                    ;Загрузка младшего байта FLAGS в AH
    mov cl,ah               ;CL = AH
    mov ah,9                ;Функция DOS 09h - вывод строки
    mov dx,s_sf             ;DX = адрес строки 'FLAGS: SF='
    int 21h                 ;Обращение к функции DOS
    shl cl,1                ;Сдвиг CL влево на 1 бит
    call print_cf           ;Печать выдвинутого бита
    mov dx,s_zf
    int 21h                 ;Вывод строки ' ZF='
    shl cl,1                ;Сдвиг CL влево на 1 бит
    call print_cf           ;Печать выдвинутого бита
    mov dx,s_af
    int 21h                 ;Вывод строки ' AF='
    shl cl,2                ;Сдвиг CL влево на 2 бита
    call print_cf           ;Печать выдвинутого бита
    mov dx,s_pf
    int 21h                 ;Вывод строки ' PF='
    shl cl,2                ;Сдвиг CL влево на 2 бита
    call print_cf           ;Печать выдвинутого бита
    mov dx,s_cf
    int 21h                 ;Вывод строки ' CF='
```

<code>shl cl,2</code>	<i>;Сдвиг CL влево на 2 бита</i>
<code>call print_cf</code>	<i>;Печать выдвинутого бита</i>
<code>mov dx,s_endl</code>	
<code>int 21h</code>	<i>;Вывод конца строки</i>
<code>popf</code>	<i>;Восстановление регистра флагов</i>
<code>pop dx</code>	
<code>pop cx</code>	
<code>pop ax</code>	
<code>ret</code>	

Полный исходный код примера — [printflags.asm](#). Вывод программы выглядит так:

```

C:\PRINTF~1.COM
FLAGS: SF=1 ZF=0 AF=1 PF=0 CF=0
FLAGS: SF=1 ZF=0 AF=1 PF=0 CF=1
FLAGS: SF=0 ZF=1 AF=0 PF=1 CF=0
Press any key..._

```

Упражнение

Чтобы потренироваться в работе с флагами напишите следующую программу. Вычислите сумму значений флагов CF, SF, ZF и выведите на консоль результат сложения. Если результат больше 1, инвертируйте значение флага CF 😊

[Следующая часть »](#)

Комментарии:

fufel
16-09-2010 19:41

Здравствуйте!

Сделал, но только вот какая фигня, если запускаю программу из винды, то сумма=0, а если запускаю в отладчике, то сумма=1. С чего такая разница не понимаю.

```

use16
org 100h
jmp start
;_____
summ_flags db 'Summa Flagov = $'

```

```
;-----  
start:  
mov dx,summ_flags  
mov ah,09h  
int 21h  
xor ax,ax  
lahf  
shr ah,1  
call summa  
shl ah,2  
call summa  
shl ah,1  
call summa  
cmp cl,1  
jg inv_cf  
vivod:  
xor dx,dx  
mov ax,cx  
mov bx,10  
div bx  
  
add dl,'0'  
mov ah,02h  
int 21h  
mov ah,08h  
int 21h  
mov ax,4c00h  
int 21h  
  
summa:  
jc summ  
ret  
summ:  
add cl,1  
ret  
inv_cf:  
cmc  
jmp vivod
```

[\[Ответить\]](#)

[xrnd](#)

20-09-2010 16:41

Так как у тебя перед командой LAHF идёт XOR AX,AX, то сумма флагов должна быть равна 1 (ZF = 1, CF = 0, SF = 0).

Разница из-за того, что ты явно не обнуляешь регистр CL. Отладчик при запуске меняет значения регистров, а когда запускаешь без отладчика, в регистрах может быть другое значение.

Кстати, на подобном принципе делают защиту от отладки. Если при запуске программы какие-то регистры нулевые, то это запуск в отладчике)))

[\[Ответить\]](#)

[xrnd](#)

20-09-2010 16:45

```
summa:
    jc summ
    ret
summ:
    add c1,1
    ret
```

А этот кусок кода можно заменить одной командой 😊

```
adc c1,0
```

[\[Ответить\]](#)

fufel

20-09-2010 18:49

Всё понятно, спасибо.

[\[Ответить\]](#)

argir

30-12-2010 22:28

org 100h ;Программа начинается с адреса 100h
jmp start

```
s_sum db 'CF+SF+ZF=$'
press db 13,10,'Press any key...$'
sr_sum db 0,'$'
```

```
start:
mov ax,64000
add ax,63999;пример устанавливающий флаги
pushf;их сохранение
lahf; в ah — флаги
xor al,al;обнуление al
ror ah,1;подготовка к подсчету — биты подряд
mov cx,3;счетчик циклов=3
sym: shl ah,1;значение флага в CF
adc al,0;суммирование значения флагов
loop sym
cmp al,1;сравнение суммы с 1
jna sr;если меньше или равно,не меняем CF
porf;если больше
cmc;инвертируем CF
pushf;сохраняем измененное значение
sr: add al,'0';преобразуем сумму в символ
mov [sr_sum],al;сохраняем как строку
mov dx,s_sum
call print;выводим
```

```
mov dx,sr_sum
call print;выводим
mov dx,press
call print;выводим
popf; восстанавливаем флаги
mov ah,08h ;Функция DOS 08h — ввод символа без эха
int 21h ;Обращение к функции DOS
```

```
mov ax,4C00h ;\
int 21h ;/ Завершение программы
```

```
print:
push ax
mov ah,09h
int 21h
pop ax
ret
```

[\[Ответить\]](#)

[xrnd](#)

02-01-2011 18:41

Очень хорошо.

Мне понравился алгоритм нахождения суммы флагов. Я даже не думал, что можно их складывать в цикле 😊

[\[Ответить\]](#)

Гость

05-02-2011 14:38

```
use16
org 100h
add cx,1; устанавливаем флаги чтобы были не 0
sub cx,2 ; устанавливаем флаги чтобы были не 0
call Flag ; вызов процедуры
xor ax,ax
mov al,[CFZ] ; помещаем значение флага c
mov ah,[SFZ]; помещаем значение флага s
add al, ah ; c=c+s
mov ah,[ZFZ] ; помещаем значение флага s
add al, ah ; c=c+s
cmp al,1 ; проверяем значение
ja m ; если al>1 то переходим
jmp m3 ; если al
m:
xor ax,ax
mov al,[CFZ];
cmp al,1 ; инверсия состояния флага до вызова процедуры
jz m1 ; c>1
jb m2; c <1
jmp m3
```

m1:
clc ; меняется текущие состояние флага
jmp m3
m2:
stc ; меняется текущие состояние флага
jmp m3
m3:
mov ax,4C00h
int 21h
SFZ rb 1
ZfZ rb 1
AFZ rb 1
PFZ rb 1
CFZ rb 1
Ds10 rb 10

Flag:
PUSHF ; сохраняет флаги в стёке
xor bx,bx
xor ax,ax
xor si,si
mov cx,8 ; счётчик =8
POPF; восстанавливаем
PUSHF ; сохраняет флаги в стёк
LAHF
;—————SAL ah,0
push ax
SHR ah,7 ; сдвигаем на 7 бит в право остаётся 7бит
mov [Ds10+si],ah ; сохраняем
pop ax
inc si
;—————
Zl:
SAL ah,1 ; смешаем на 1 бит , или на x=x+1
push ax
SHR ah,7 ; сдвигает всё лишние оставляя только 1 бит
mov [Ds10+si],ah ; сохраняем в память
xor ax,ax
pop ax
inc si
Loop Zl
mov bl,[Ds10+0]
mov [SFZ],bl ; SFZ=S
mov bl,[Ds10+1]
mov [ZfZ],bl ; ZfZ=Z
mov bl,[Ds10+3]
mov [AFZ],bl ; AFZ=A
mov bl,[Ds10+5]
mov [PFZ],bl ; PFZ=P
mov bl,[Ds10+7]
mov [CFZ],bl ; CFZ=C
POPF
ret

[\[Ответить\]](#)

[xrnd](#)

09-02-2011 17:05

Как ни странно, это работает 😊

Но простых путей ты точно не искал.
Тут тоже много ненужных обнулений регистров.

```
PUSHF ; сохраняет флаги в стёке
xor bx,bx
xor ax,ax
xor si,si
mov cx,8 ; счётчик =8
POPF ; восстанавливаем
PUSHF ; сохраняет флаги в стёк
LAHF
```

Если убрать ненужный XOR, то и POPF/PUSHF делать не нужно, так как команда MOV флаги не изменяет.

Далее, зачем тебе 10 байт Ds10, если всего используется 8? Значения переменных-флагов дублируют байты массива Ds10. Можно было обойтись тем, что просто объявить по-другому:

```
Ds10:
SFZ db ?
ZFZ db ?
    db ?
AFZ db ?
    db ?
PFZ db ?
    db ?
CFZ db ?
```

Тогда не нужен код, копирующий значения.

Много лишних команд переходов. Инвертировать флаг CF проще командой CMC.
Ещё одна проблема в том, что флаги изменяются при вычислении суммы, поэтому инвертируется уже другое значение CF.

[\[Ответить\]](#)

RoverWWorm
04-03-2011 15:14

не могу понять зачем помещать в dl строку '0'
mov dl,'0' ;DL = '0'

[\[Ответить\]](#)

RoverWWorm
04-03-2011 18:42

или эта операция преобразует число находящийся в dl в символьный вид

[\[Ответить\]](#)

[xrnd](#)

05-03-2011 16:18

В dl помещается не строка, а символ '0', точнее его код 30h.

Дальше выполняется команда adc dl,0. Если CF=0, то значение в DL не изменится. Если CF=1, то в DL будет 31h, код символа '1'.

[\[Ответить\]](#)

алекс

31-03-2012 15:17

```
use16
org 100h
jmp start
summ db 0
sout rb 2
resul db 'Flags sum =$'
press db 13,10,'Press any key...$'
start:
lahf
mov al,ah
and al,11000001b
mov cx,8
xor dl,dl
loop1:
ror al,1
jnb next
inc dl
next:
loop loop1
mov [summ],dl
cmp dl,1
jna next1
xor ah,00000001b
next1:
sahf
sum_out:
mov dl,[summ]
add dl,'0'
mov [sout],dl
mov [sout+1], '$'
mov ah,09h
mov dx,resul
int 21h
mov dx,sout
int 21h
mov dx,press
int 21h
mov ah,08h
int 21h
```

```
mov ax,4c00h
int 21h
```

[\[Ответить\]](#)

Leoscoder
07-02-2014 09:08

```
use16
org 100h
xor dx,dx
;Изменяем флаги для проверки
mov ah,11000001b ;sf,zf,0,af,0,pf,1,cf
sahf
;читаем флаги и проверяем
lahf ; ah=flags
shl ah,1 ;sf
adc dh,0
shl ah,1 ;zf
adc dh,0
shr ah,3 ;cf
adc dh,0
xchg dh,dl
and dl,11111111b;проверка на 0
jz m1 ;если dl=0->m1
cmc ;иначе, инвентируем CF
m1:
;Вывод на консоль
print:

push dx
mov dx,sum
mov ah,9
int 21h
pop dx
add dl,48
mov ah,2
int 21h
exit:
mov dx,press
mov ah,9
int 21h
mov ah,8
int 21h
mov ax,4c00h
int 21h
sum db 'sum = $'
press db 13,10,'Press any key...$'
```

[\[Ответить\]](#)

Ваш комментарий

Имя *

Почта (скрыта) *

Сайт

Добавить

- ☐ Уведомить меня о новых комментариях по email.
- ☐ Уведомлять меня о новых записях почтой.