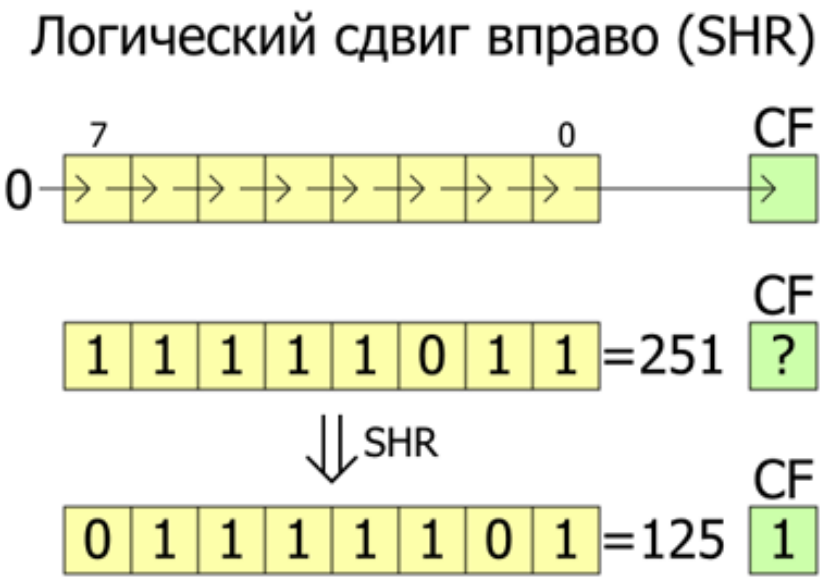
Учебный курс. Часть 18. Линейный сдвиг

Автор: xrnd | Рубрика: Учебный курс | 07-05-2010 | Распечатать запись

Сдвиги — это особые операции процессора, которые позволяют реализовать различные преобразования данных, работать с отдельными битами, а также быстро выполнять умножение и деление чисел на степень 2. В этой части мы рассмотрим операции линейного сдвига, а в следующей будут циклические.

Логический сдвиг вправо

Логический сдвиг всегда выполняется без учёта знакового бита. Для логического сдвига вправо предназначена команда SHR. У этой команды два операнда. Первый операнд представляет собой сдвигаемое значение и на его место записывается результат операции. Второй операнд указывает, на сколько бит нужно осуществить сдвиг. Этим операндом может быть либо непосредственное значение, либо регистр CL. Схема выполнения операции показана на рисунке:

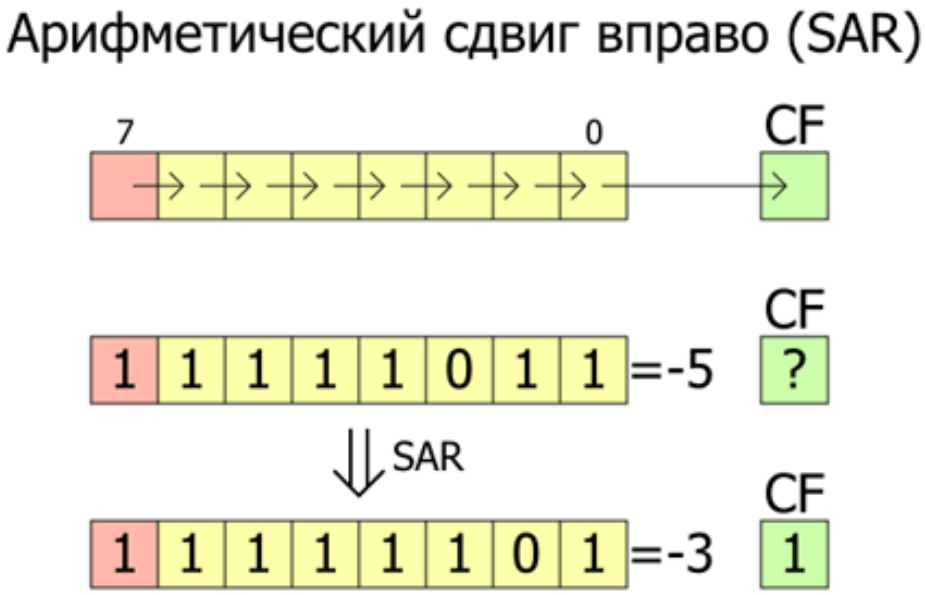


Все биты операнда сдвигаются вправо (от старших битов к младшим). Выдвинутый бит становится значением флага CF. Старший бит получает нулевое значение. Эта операция повторяется несколько раз, если второй операнд больше единицы. Логический сдвиг вправо можно использовать для деления целых чисел без знака на степень 2, причём сдвиг работает быстрее, чем команда деления DIV. Примеры:

```
    shr ax,1          ;Логический сдвиг AX на 1 бит вправо
    shr byte[bx],cl   ;Лог. сдвиг байта по адресу BX на CL бит вправо
    shr cl,4          ;CL = CL / 16 (для числа без знака)
```

Арифметический сдвиг вправо

Арифметический сдвиг отличается от логического тем, что он не изменяет значение старшего бита, и предназначен для чисел со знаком. Арифметический сдвиг вправо выполняется командой SAR. У этой команды тоже 2 операнда, аналогично команде SHR. Схема выполнения операции показана на рисунке:

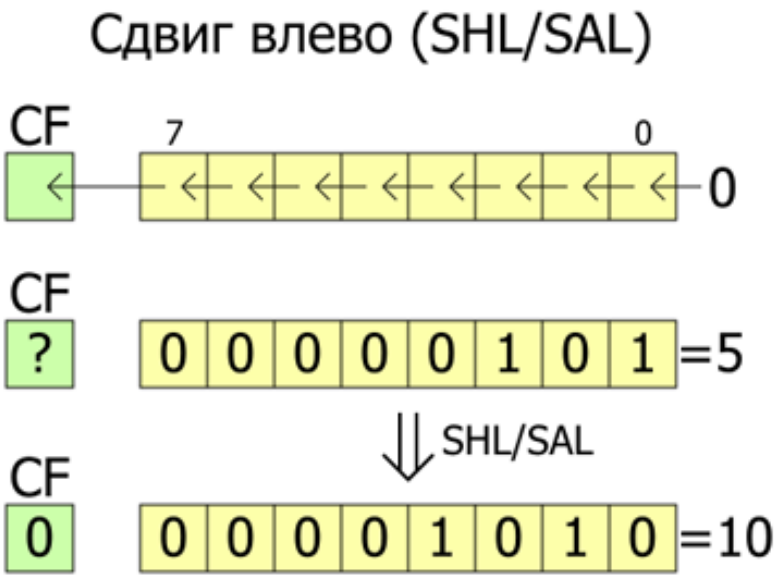


Выдвинутый бит становится значением флага CF. Знаковый бит не изменяется. При сдвиге на 1 бит сбрасывается флаг OF. Эту команду можно использовать для деления целых чисел со знаком на степень 2 (обратите внимание, что «округление» всегда в сторону меньшего числа, поэтому для отрицательных чисел результат будет отличаться от результата деления с помощью команды IDIV). Примеры:

```
    sar bx,1        ;Арифметический сдвиг BX на 1 бит вправо
    sar di,cl       ;Арифметический сдвиг DI на CL бит вправо
    sar [x],3       ;x = x / 8 (x - 8-битное значение со знаком)
```

Логический и арифметический сдвиг влево

Логический сдвиг влево выполняется командой SHL, а арифметический — командой SAL. Однако, на самом деле это просто синонимы для одной и той же машинной команды. Сдвиг влево одинаков для чисел со знаком и чисел без знака. У команды 2 операнда, аналогично командам SHR и SAR.Схема этой операции показана на рисунке:

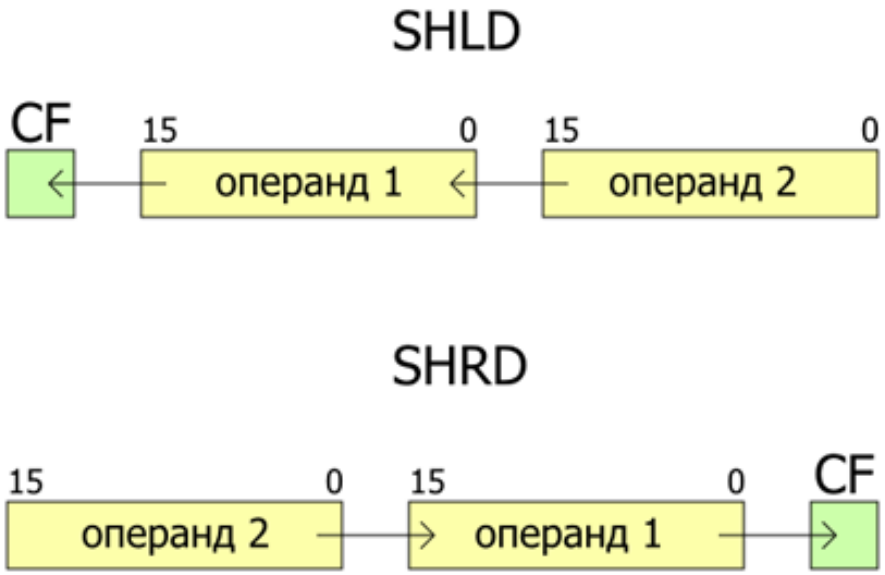


Старший бит становится значением флага CF, а младший получает нулевое значение. С помощью сдвига влево можно быстро умножать числа на степень 2. Но будьте внимательны, чтобы не получить в результате переполнение. Если при сдвиге на 1 бит меняется значение старшего бита, то устанавливается флаг OF. Примеры использования команды:

```
    shl dx,1          ;Сдвиг DX на 1 бит влево
    sal dx,1          ;То же самое
    shl ax,cl         ;Сдвиг AX на CL бит влево
    sal [x],2         ;x = x * 4
```

Сдвиги двойной точности

Существуют ещё две команды, осуществляющие более сложные сдвиги. SHRD — сдвиг двойной точности вправо, SHLD — сдвиг двойной точности влево. У этих команд 3 операнда. Первый операнд — сдвигаемое значение и место для записи результата, должен иметь размер 16 бит. Второй операнд — источник вдвигаемых битов, тоже должен иметь размер 16 бит и находится в одном из регистров. Значение второго операнда не меняется. Третий операнд — счётчик сдвигов, может быть непосредственным значением или находиться в регистре CL. Схемы работы этих команд показаны на рисунке:



Небольшой пример использования команды SHLD:

```
    shld ax,bx,3      ;Сдвинуть ax на 3 бита влево,
                      ;3 старших бита BX становятся младшими битами AX
```

Пример программы

Программа печатает переменную размером 16-бит в двоичном виде. Используется команда сдвига влево на 1 бит, после чего анализируется значение флага CF. Если CF=0, выводим символ ‘0’, если CF=1 выводим символ ‘1’. Проверка битов осуществляется в цикле.

```
1 use16                 ;Генерировать 16-битный код
2 org 100h              ;Программа начинается с адреса 100h
3     jmp start         ;Безусловный переход на метку start
4 ;-- Данные ------------------------------------------------------------
5 v   dw 12345
6 pak db 13,10,'Press any key...$'
```

```
 7 ;---------------------------------------------------------------------
 8 start:
 9     mov bx,[v]        ;BX = v
10     mov ah,2          ;Функция DOS 02h - вывод символа
11     mov cx,16         ;Инициализация счётчика цикла
12 lp:
13     shl bx,1          ;Сдвиг BX на 1 бит влево
14     mov dl,'0'        ;dl = '0'
15     jnc print         ;Переход, если выдвинутый бит равен 0
16     inc dl            ;dl = dl + 1 = '1'
17 print:
18     int 21h           ;Обращение к функции DOS 02h
19     loop lp           ;Команда цикла
20
21     mov ah,9          ;\
22     mov dx,pak        ; > Вывод строки 'Press any key...'
23     int 21h           ;/
24
25     mov ah,8          ;\
26     int 21h           ;/ Ввод символа без эха
27
28     mov ax,4C00h      ;\
29     int 21h           ;/ Завершение программы
```

Результат работы программы:

Упражнение

Объявите массив из 8 слов без знака. Сдвиньте первый элемент на 1 бит влево, второй элемент — на 2 бита вправо (логическим сдвигом), третий элемент — на 3 бита влево и т.д. до конца массива. Используйте циклы. Проверьте работу программы в отладчике. Результаты можете выкладывать в комментариях.

Следующая часть »

Комментарии:

RoverWWWorm
18-05-2010 20:09

```
use16
org 100h

jmp start
;—————————————
array dw 1111,2222,3333,4444,5555,6666,7777,8888
n db 8
array2 dw ?
;—————————————
start:
mov di,0
movzx cx,[n]
mov bx,0 ;для манипуляций с cl, только почему xor bh,bh не работает или xor bx,bx
lp:
mov ax,word[array+di]

mov bl,cl ;сохраняем значение cl, чтобы не нарушить цикл
inc bh
mov cl,bh

shl ax,cl
mov word[array2+di],ax ;
add di,2

mov cl,bl ;востанавливаем значение cl

loop lp

mov ax,4c00h
int 21h
```

[Ответить]

xrnd
19-05-2010 00:18

«почему xor bh,bh не работает или xor bx,bx»

Не знаю. Я проверил — работает. Также можно делать xor di,di

У тебя здесь есть серьёзная ошибка. array2 получается, что состоит только из одного элемента, поэтому при записи элементов программа будет стирать свои команды. Надо написать «dw 8 dup(?)» или «rw 8»

И ты все элементы сдвигаешь влево. А надо то влево, то вправо 🙂 Нужно либо делать условный переход внутри цикла. Либо попробуй сделать цикл, который будет обрабатывать по 2 элемента внутри (один сдвигать влево, второй — вправо) и повторяться n/2 раз.

[Ответить]

xrnd
19-05-2010 00:19

Я конечно тут извратился с заданием, но зато подумать можно немного 🙂

[Ответить]

RoverWWWorm
19-05-2010 19:03

да сойдет сложность задачи, вот в 19 уроке посложнее будет написать код 🙂

[Ответить]

RoverWWWorm
19-05-2010 19:00

«И ты все элементы сдвигаешь влево. А надо то влево, то вправо…»
Даа, не внимательно я прочел задачу 🙂

```
org 100h

jmp start
;_____________________
array dw 1111,2222,3333,4444,5555,6666,7777,8888
n db 8
array2 rw 8
;_____________________
start:
xor di,di
movzx ax,[n]
mov bh,2
div bh
movzx cx,al
xor bh,bh ;для манипуляций с cl
lp:
mov ax,word[array+di]

mov bl,cl ;сохраняем значение cl, чтобы не нарушить цикл
inc bh
mov cl,bh

shl ax,cl
mov word[array2+di],ax
add di,2
;_____________________________
mov ax,word[array+di] ;теперь двигаем вправо
inc cl
shr ax,cl
mov word[array2+di],ax
add di,2
;_____________________________
mov cl,bl ;востанавливаем значение cl

loop lp
```

mov ax,4c00h
int 21h

[Ответить]

xrnd
21-05-2010 10:49

Хорошо, теперь почти всё верно. Но тебе ещё надо добавить одну команду «inc bh» внутри цикла, а то ты «inc cl» делаешь второй раз, а bh не изменяется.

И деление на 2 очень сложно написал. Можно разделить сдвигом вправо на 1 бит. Вместо:

```
    movzx ax,[n]
    mov bh,2
    div bh
    movzx cx,al
```

написать:

```
    movzx cx,[n]
    shr cx,1
```

[Ответить]

RoverWWWorm
21-05-2010 17:58

….
;———————————————
mov ax,word[array+di] ;теперь двигаем вправо
inc cl
shr ax,cl
mov word[array2+di],ax
add di,2
;———————————————
inc bh ; здесь добавить да

mov cl,bl ;востанавливаем значение cl

loop lp

; а с делением да, намудрил малость

[Ответить]

fufel
23-06-2010 21:02

use16
org 100h
jmp start
;————————————-
array dw 1AAh,1ABh,1ACh,1ADh,1AEh,1AFh,1BBh,1BCh

```
n db 1
;————————————-
start:
mov cx,4
mov di,2
lp:
mov dx,cx
sub cx,cx
mov cl,[n]
mov ax,[array+si]
shl ax,cl
inc cl
mov bx,[array+di]
shr bx,cl
add si,4
add di,4
inc cl
mov [n],cl
mov cx,dx
loop lp

mov ax,4c00h
int 21h
```

[Ответить]

xrnd
24-06-2010 02:42

Привет! Хороший вариант, но есть недостатки.

Во-первых, регистр si перед началом цикла надо обнулить. Когда смотришь программу в отладчике, он обнуляет регистры, но реально при запуске программы это может быть не так. Поэтому всегда лучше явно инициализировать регистры.

Во-вторых, сдвиг выполняется правильно, но результат нигде не сохраняется 🙂 Неплохо было бы сохранить сдвинутое значение на место исходного.

[Ответить]

fufel
25-06-2010 22:18

Спасбо, учту))).

[Ответить]

Shindo
30-11-2010 22:07

```
use16
org 100h
```

```
mov cx, 8
mov di, 0

jmp start
hello db ‘Hello!!!$’
parametr dw 1 dup(10,15,20,25,30,35,40,45)
parametrnot rw 8
;————————————————————-

start:

mov bx, word[parametr+di]
cmp di, 0
jnp metka1
cmp di, 0
jz metka2
shl bx, di
mov word[paramerrnot+di], bx
inc di
loop start
jmp exit
;————————————————————

metka1:
shr bx, di
mov word[paramerrnot+di], bx
inc di
loop start
;————————————————————
metka2:
shl bx, di
mov word[paramerrnot+di], bx
inc di
loop start

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

я запутался. Подскажите что не так ?

[Ответить]

xrnd
01-12-2010 16:15

```
    shl bx, di
    shr bx, di
```

Счётчик сдвигов может быть только в регистре CL.

Тебе нужно либо сохранить CX, поместить в него счетчик сдвигов, а потом восстановить, либо сделать цикл не командой loop, а переходами.

Например в конце цикла сделать сравнение и переход.
Объявить массив можно проще:

```
parametr dw 10,15,20,25,30,35,40,45
```

Тут ещё надо учесть, что поскольку массив состоит из слов, то смещения элементов от начала будут в 2 раза больше: 0, 2, 4 и т.д.

```
    mov bx,[parametr+di]
    add di,2
```

[Ответить]

Shindo
01-12-2010 19:39

спасибо!)

[Ответить]

argir
13-12-2010 09:49

Чтобы использовать при сдвиге регистр CL, начал программу с конца массива.

```
use16
org 100h
jmp start

array1 dw 810h,820h,830h,840h,850h,860h,870h,880h

start:
mov cx,8
mov si,cx
dec si
sal si,1 ;адрес последнего элемента
star:
test cl,1 ;проверка на четность
jz cona
shl [array1+si],cl
jmp con
cona: shr [array1+si] ,cl
con : dec si
dec si
loop star

mov ax,4C00h
int 21h
```

[Ответить]

xrnd
13-12-2010 20:02

Да, это самый оптимальный вариант.
Ошибок нет, всё правильно.

[Ответить]

Гость
17-01-2011 02:09

```
use16
org 100h
jmp start
;— Äàíûå ——————————————————————————
vk db 'pacs$','dva$','pacs$','dva$','pacs$','dva$','pacs$','$'
pak db 13,10,'Press any key…$'
m db 1
;————————————————————————————————————-
start:
mov si,0 ; ствик от начала
mov di,0 ; количества $ чтобы понять конец элемента масмиве
mov cx,40 ; 40 что количество букв в массиве не больше 40
s4t4ik1: ; цикл ищет $
mov dl,byte[vk+si];
inc si ; +1
cmp dl, 24h ; 24h ($)
Jz s4t4ik2 ; если в $ делаем переход
loop s4t4ik1
jmp exit
s4t4ik2:
inc di ; 1 первый элемент
mov bx ,cx ;сохраняем cx
mov cx, 0
cmp di,2 ; если второй элемент делаем переход
Jz Metka1
mov cx ,di ; помешаем в cl номер элемента
sal dl,cl ; сдвигаем на номер элемента
mov [vk+si],dl ; сохраняем изменения
mov cx,bx ; восстанавливаем cx
cmp di,8 ; если это 8 элемент выходим из програмы
Jz exit
jmp s4t4ik1 ;
Metka1:
shr dl,1
mov [vk+si], dl ; сдвигаем на номер элемента
mov [vk+si],dl ; сохраняем изменения
jmp s4t4ik1
exit:
mov ax,4C00h ;\
int 21h
```

[Ответить]

xrnd
18-01-2011 21:43

Программа хорошая, но слова без знака опять меня порадовали 🙂
Слово без знака объявляется директивой dw и представляет собой 16-битное целое (два байта).

[Ответить]

Knight212
21-02-2011 21:19

```
use16
org 100h
jmp start

array dw 35135, 1654, 0, 5341
dw 135, 12, 8443, 54632

start:
mov bx, array+14
mov cx, 8

lpr:
mov ax, [bx]
shr ax, cl
sub bx, 4
sub cx, 1
loop lpr

mov bx, array+12
mov cx, 7

lpl:
mov ax, [bx]
shl ax, cl
sub bx, 4
sub cx, 1
cmp cx, 0
je exit
loop lpl

exit:
mov ax, 4C00h
int 21h
```

[Ответить]

Shov
31-03-2011 15:54

```
use16
org 100h
jmp start
;_____
arr1 dw 1111h, 2222h, 3ed4h, 4a45h, 5555h, 6666h, 7777h, 8888h
arr2 dw 8 dup(?)
```

```
n db 8
;———
start:
xor si,si
mov cl,1
body:

mov bx,word[arr1+si]
test cl,01
jnz toleft

;— if CF=1 to right
shr bx,cl
mov word[arr2+si],bx
inc cl
add si,2

cmp cl,9
je exit
jmp body

;— if CF=0 to left
toleft:
shl bx,cl
mov word[arr2+si],bx
inc cl
add si,2

cmp cl,9
je exit
jmp body

exit:
mov ax,4C00h
int 21h
```

[Ответить]

xrnd
01-04-2011 15:56

Код написан без ошибок, но тут два куска кода практически идентичны — отличие только в команде сдвига.
Можно сделать проще:

```
    ...
    jnz toleft
    shr bx,cl
    jmp save
toleft:
    shl bx,cl
save:
    mov word[arr2+si],bx
    inc cl
    add si,2
    ...
```

[Ответить]

plan4ik
03-04-2011 17:20

```
use16
org 0x100
jmp Start

;== [DATA] ===================================

arr dw 0x1111, 0x2222, 0x3333, 0x4444
dw 0x5555, 0x6666, 0x7777, 0x8888
res_arr rw 8

;=============================================

Start:
mov ch, 4
xor cl, cl

Parity:
movzx si, cl
mov bx, word [arr+si]
; 0-ой элемент отдЫхает 😉
shl bx, cl
mov word [res_arr+si], bx
add cl, 2
dec ch
jnz Parity

mov ch, 4
mov cl, 1
NParity:
movzx si, cl
mov bx, word [arr+si]
shl bx, cl
mov word [res_arr+si], bx
add cl, 2
dec ch
jnz NParity

mov ax, 4c00h
int 21h
```

[Ответить]

plan4ik
03-04-2011 17:30

оЙ ;(я заметил … у меня смещение SI в байт
Исправлю так:

…
mov ch, 4
mov cl, 1
NParity:
movzx si, cl
shl si, 1
mov bx, word [arr+si]
shl bx, cl
mov word [res_arr+si], bx
add cl, 2
dec ch
jnz NParity
…

сдвиг влево умножает значение в двое SI<<1 == (2, 4, 6 ,8)

[Ответить]

plan4ik
03-04-2011 17:34

блин я задачу не внимательно читал и поэтому в цикле NParity есть ошибка … сдвиг влево вместо в право … чЁрт надеюсь ошибок больше нету 😉

[Ответить]

plan4ik
03-04-2011 17:44

блин еще одна извини за флуд

но я еще не учел то что у меня смещение не только в слово но я еще прыгаю через одно слово по этому смещение будет (1<<si)+si , короче я запутался скину нормальный исходник позже

[Ответить]

plan4ik
03-04-2011 19:06

у меня с математикой плохо и я се чуть голову не сломал

Start:
mov ch, 4
xor cl, cl

Parity:
movzx si, cl
add si, si
mov bx, word [arr+si]
; 0-îé ýëåìåíò îòâûâàò 😉
shl bx, cl
mov word [res_arr+si], bx
add cl, 2

```
dec ch
jnz Parity

mov ch, 4
mov cl, 1
NParity:
movzx si, cl
add si, si
mov bx, word [arr+si]
shr bx, cl
mov word [res_arr+si], bx
add cl, 2
dec ch
jnz NParity

mov ax, 4c00h
int 21h
```

[Ответить]

plan4ik
03-04-2011 19:07

```
;== [DATA] ===================================

arr dw 0×1111, 0×2222, 0×3333, 0×4444
dw 0×5555, 0×6666, 0×7777, 0×8888
res_arr rw 8

;=============================================
```

[Ответить]

xrnd
08-04-2011 18:59

Интересный алгоритм получился.
Но работает правильно 🙂

[Ответить]

NimRoen
18-05-2011 13:53

```
;последовательный сдвиг элементов массива размерностью 8 слов
;нечетные элементы влево на значение индекса элемента, четные — вправо
use16
org 100h
   jmp main

;=========================;
arrVar dw 1111,2222,3333,4444,5555,6666,7777,8888
;=========================;
```

```
;=========================;
main:
    mov cx,8
    mov ax,arrVar
    add ax,0eh
    mov si,ax
    mov di,ax
    std
slide_element:
    lodsw
    test cl,1
    jne slide_right
    shl ax,cl
    jmp short save_element
slide_right:
    shr ax,cl
save_element:
    stosw
    loop slide_element
    mov ax,4c00h
    int 21h
;=========================;
```

[Ответить]

xrnd
23-06-2011 14:23

Отличный код, но только сдвиг получается наоборот: чётные влево, нечётные вправо 🙂
Нужно поменять «jne slide_right» на «je»

[Ответить]

алекс
15-03-2012 22:07

```
use16
org 100h

jmp start ;переход на метку start

array dw 23476,23598,12397,23685 ;описание данных
dw 47593,56774,38432,23457

lenght db 8

start: ;метка start начало программы

movzx cx,[lenght] ;помещаем в щетчик цикла длину массива
xor si,si ;инициализируем si нулем
xor dl,dl ;инициализируем dl нулем

cyk: ;метка начала цикла
```

```
mov bx,cx ;сохраняем значение щетчика в bx
inc dl ;увеличиваем dl на единицу
mov cl,dl ;помещаем в cl значение dl
test dl,01h ;проверяем значение dl на четность
je par ;если dl четное переход на метку par
shl [array+si],cl ;здвиг влево на cl бит
jmp el ;переход на метку el
par: ;метка par
shr [array+si],cl ;сдвиг вправо на cl бит
el: ;метка el
add si,2 ;увеличиваем si на 2
mov cx,bx ;востанавливаем значение щетчика их bx

loop cyk ;конец цикла,возврат на метку cyk
mov ax,4c00h ;завершение программы
int 21h
```

[Ответить]

Smith
26-03-2012 09:34

```
use16
org 100h

jmp start

ar1 dw 25,54,75,12,64,35,18
ar2 rw 7

right:
shr ax,cl
jmp return

start:

mov cx,7
xor bx,bx
xor di,di
lp:
mov ax,[ar1+di]
inc bl
add di,2
mov bh,cl
mov cl,bl

test cl,1
jz right
shl ax,cl
return:

mov word[ar2+di-2],ax
```

```
mov cl,bh
loop lp

int 20h
```

[Ответить]

X
17-03-2013 19:05

```
use16
org 100h
jmp start
;======================
arr1 dw 1,2,3,4,5,6,7,8
arr2 dw 8 dup(?)
sz db 8
;======================
start:

movzx cx,[sz]
mov si,-1
xor cl,cl
lp1:
inc si
inc cl
mov ax,word[arr1+si]
mov bx,si
shr bx,1
jnc left
jc right
loop lp1

left:
shl ax,cl
mov word[arr2+si],ax
jmp lp1

right:
shr ax,cl
mov word[arr2+si],ax
jmp lp1

mov ah,08h
int 21h
mov ax,4c00h
int 21h
```

[Ответить]

kotya
15-11-2015 04:16

самый компактный код пока что у меня x)
можно сократить еще на одну строку, но тогда вместо 48 байт будет 49

```
use16
org 100h
jmp start

arr dw 564,394,152,342,255,662,447,188

start:
mov si,14
mov cx,8
l1:
test cl,1
jnz left
shr [arr+si],cl
jmp end1
left:
shl [arr+si],cl
end1:
dec si
dec si
loop l1

mov ax,4C00h
int 21h
```

[Ответить]

Александр
13-11-2016 02:39

```
use16; Оригинальности нет, увы.
org 100h
jmp start
;_____________________________________
array1 dw -1,-1,-1,-1,-1,-1,-1,-1
array2 rw 8
;_____________________________________
start:
mov bx,1
mov cx,8
xor di,di
mov si,array1
lp:
mov ax,[si]
mov dx,cx
mov cx,bx

shl ax,cl
mov [array2+di],ax

inc cl
add di,2
```

```
shr ax,cl
mov [array2+di],ax

mov cx,dx
add bl,2
add di,2
loop lp

mov ax, 4C00h
int 21h
```

[Ответить]

Олег
26-03-2019 12:52

Здравствуйте. Решил написать программку после 18 уроков, чтобы закрепить пройденный материал. Два дня писал. Программка рисует спрайт, который можно перемещать по экрану клавишами со стрелками. Esc — выход. Правда, мерцает при перемещении. До синхронизации с развёрткой мне ещё далековато.

```
use16
org 100h
jmp begin
;const
step_x dw 1 ;шаг в пикселях по x
step_y dw 1 ;шаг в пикселях по y

;var
sprite db 33h,33h,33h,33h,0CCh,0CCh,0CCh,0CCh ;Битмап
db 3Fh,0F0h,3Fh,0F0h,0CFh,0F0h,0CFh,0F0h ;спрайта.
db 3Fh,0F0h,3Fh,0F0h,0CFh,0F0h,0CFh,0F0h ;Размер
db 30h,00h,30h,00h,0C0h,00h,0C0h,00h ;16×16
db 33h,33h,33h,33h,0CCh,0CCh,0CCh,0CCh ;точек
db 3Fh,0F0h,3Fh,0F0h,0CFh,0F0h,0CFh,0F0h ;
db 3Fh,0F0h,3Fh,0F0h,0CFh,0F0h,0CFh,0F0h ;
db 30h,00h,30h,00h,0C0h,00h,0C0h,00h ;
x dw 0 ;Координаты спрайта на экране. X — горизонталь,
y dw 0 ;Y — вертикаль. Начало координат — верхний левый угол
stroka rb 5

begin:
mov ax,4 ;Графический режим CGA, 320×200, 4 цвета
int 10h ;BIOS. Видео сервис.
mov ax,0b800h ;Сегмент адреса видеопамяти
mov es,ax
mov di,0
mov bp,0
jmp put_sprite_start

;Procedure Put_sprite
;———————использованные регистры:———————————
;di — смещение относительно начала массива sprite
; (номер элемента,0 тоже считается)
```

```
;si — смещение в видеопамяти с началом $b800:$0000 (CGA)
;bp — счётчик циклов от 0 до 15. За один цикл рисуется
; одна строка спрайта. Спрайт имеет 16 строк
;bl — хранит смещение на количество пикселей в байте.
; Возможные значения: 0,1,2,3. Один байт кодирует 4 пикселя.
;————————————————————————————

put_sprite_start:
mov ax,[y]
add ax,bp
shr ax,1
mov cx,80
jc half_1
mul cx
mov si,ax
jmp continue_1
half_1:
mov si,2000h
mul cx
add si,ax
continue_1:
mov ax,[x]
mov cl,4
div cl
mov bl,ah
cbw
add si,ax
cmp bl,0
ja sdvig
mov cx,4
rec_vram_1:
mov al,[sprite+di]
mov [es:si],al
inc di
inc si
loop rec_vram_1
inc bp
cmp di,64
jnz put_sprite_start
mov bp,0
mov di,0
jmp key_start
sdvig:
mov cx,5
mov ah,bl
mov bx,stroka
mov al,0
loop_2:
mov byte[bx],al
inc bx
loop loop_2
```

```
mov cl,ah
shl cl,1
mov dx,0
sub bx,2
add di,3
mov ch,3
loop_3:
mov ax,0
mov ah,[sprite+di]
dec di
mov dl,[sprite+di]
shrd ax,dx,cl
mov [bx],ah
inc bx
or [bx],al
sub bx,2
dec ch
cmp ch,0
jnz loop_3

mov dl,[sprite+di]
shr dl,cl
mov [bx],dl
mov cx,5
loop_4:
mov dl,[bx]
mov [es:si],dl
inc bx
inc si
loop loop_4

add di,4
inc bp
cmp bp,16
jnz put_sprite_start
mov di,0
mov bp,0
jmp key_start

;endProcedure

;Procedure Clear_Screen
clear:
mov si,0
loop_1:
mov byte[es:si],0
inc si
cmp si,4000h
jnz loop_1
jmp put_sprite_start
;EndProcedure

;Procedure Key
key_start:
```

```
;очистка буфера клавиатуры
mov ax,40h
mov es,ax
mov si,1ah
mov ax,[es:si]
mov si,1ch
mov [es:si],ax
mov ax,0b800h
mov es,ax
;———————————
mov ah,8
int 21h
cmp al,75
jz left
cmp al,77
jz right
cmp al,27
jz exit
cmp al,72
jz up
cmp al,80
jz down
jmp key_start
left:
mov ax,[x]
cmp ax,[step_x]
js key_start
sub ax,[step_x]
mov [x],ax
jmp clear
right:
mov ax,[x]
add ax,[step_x]
cmp ax,304
ja key_start
mov [x],ax
jmp clear
up:
mov ax,[y]
cmp ax,[step_y]
js key_start
sub ax,[step_y]
mov [y],ax
jmp clear
down:
mov ax,[y]
add ax,[step_y]
cmp ax,184
ja key_start
mov [y],ax
jmp clear
;endProcedure
```

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

[Ответить]

Ваш комментарий

Имя *

Почта (скрыта) *

Сайт

Добавить

Уведомить меня о новых комментариях по email.

Уведомлять меня о новых записях почтой.