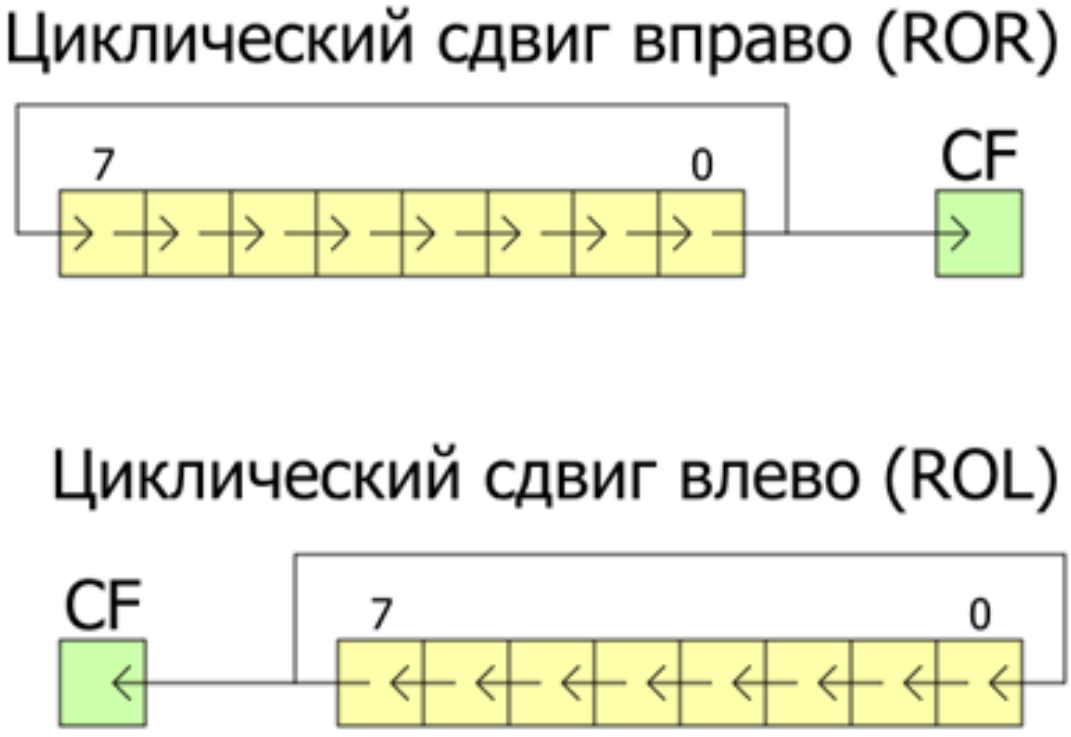
Учебный курс. Часть 19. Циклический сдвиг

Автор: xrnd | Рубрика: Учебный курс | 13-05-2010 |
Распечатать запись

Циклический сдвиг отличается от линейного тем, что выдвигаемые с одного конца биты вдвигаются с другой стороны, то есть движутся по кольцу. В процессора x86 существует 2 вида циклического сдвига: простой и через флаг переноса (CF). У всех команд, рассматриваемых в этой части учебного курса, по 2 операнда, таких же, как у команд линейного сдвига. Первый операнд — сдвигаемое значение и место для записи результата. Второй операнд — счётчик сдвигов, который может находится в регистре CL или указываться непосредственно.

Простой циклический сдвиг

Циклический сдвиг вправо выполняется командой ROR, а влево — командой ROL. Схема работы этих команд представлена на рисунке (на примере 8-битного операнда):



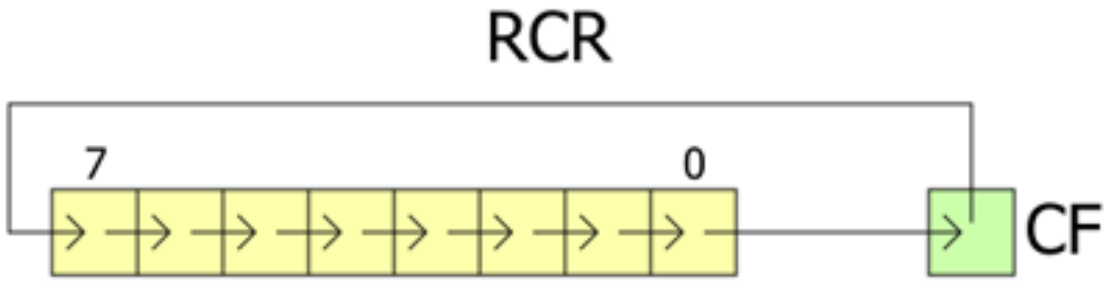
Значение последнего выдвигаемого бита копируется в флаг CF. Для сдвигов на 1 бит устанавливается флаг OF, если в результате сдвига

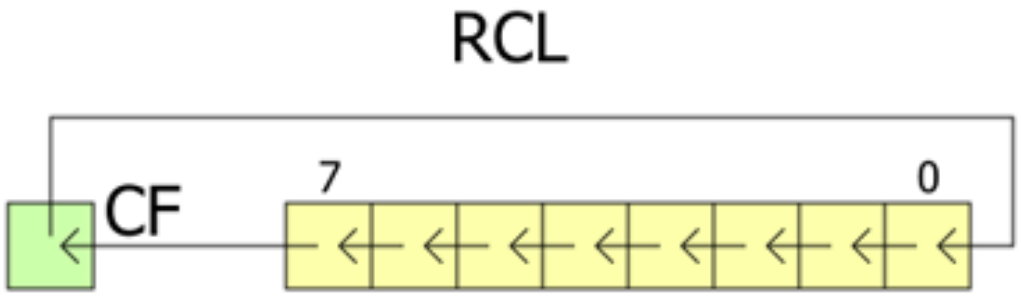
изменяется знаковый бит операнда. (Честно говоря, не помню, чтобы я этим когда-то пользовался. Но может вам пригодится 🙂) Примеры использования команд:

```
rol bl,1          ;Циклический сдвиг BL на 1 бит влево
ror word[si],5    ;Циклический сдвиг слова по адресу в SI на 5 бит вправ
rol ax,cl         ;Циклический свдиг AX на CL бит влево
```

Циклический сдвиг через флаг переноса

Отличие от простого циклического сдвига в том, что флаг CF участвует в сдвиге наравне с битами операнда. При сдвиге на 1 бит выдвигаемый бит помещается в CF, а значение CF вдвигается в операнд с другой стороны. При сдвиге на несколько бит эта операция повторяется многократно. Циклический сдвиг через флаг переноса выполняется командами RCR (вправо) и RCL (влево).





Опять же для сдвигов на 1 бит устанавливается флаг OF, если в результате сдвига изменяется знаковый бит операнда. Примеры использования команд:

```
rcr dh,3          ;Цикл. сдвиг DH на 3 бита вправо через флаг CF
rcl byte[bx],cl   ;Цикл. сдвиг байта по адресу в BX на CL бит влево чере
```

```
    rcl dx,1          ;Цикл. сдвиг DX на 1 бит влево через флаг CF
```

Пример программы

В качестве примера напишем программу для подсчёта единичных битов в байте. Для анализа битов используется команда ROL в цикле. Цикл выполняется 8 раз — по числу битов в байте. Если очередной бит равен 1, то выполняем инкремент счётчика единичных битов.

```
 1 use16              ;Генерировать 16-битный код
 2 org 100h           ;Программа начинается с адреса 100h
 3
 4     mov al,[x]     ;AL = x
 5     xor bl,bl      ;BL = 0 (Здесь будем считать единичные биты)
 6     mov cx,8       ;Инициализация счётчика цикла
 7 lp:
 8     rol al,1       ;Цилический сдвиг AL на 1 бит влево
 9     jnc bit0       ;Переход, если CF=0
10     inc bl         ;Инкремент счетчика единичных битов
11 bit0:
12     loop lp        ;Команда цикла
13     mov [n],bl     ;Сохраняем результат в n
14
15     mov ax,4C00h   ;\
16     int 21h        ;/ Завершение программы
17 ;-------------------------------------------------------------------
18 x  db 89h          ;Байт
19 n  db ?            ;Количество единичных битов в байте
```

Эту программу можно немного оптимизировать. Во-первых, использовать вместо условного перехода команду ADC со нулевым вторым операндом. Это позволит прибавить 1, если CF=1 и прибавить 0, если CF=0. Во-вторых, считать единичные биты можно в регистре AH, а обнулить его в начале с помощью команды MOVZX, совместив с загрузкой байта в регистр AL. Ноль для команды ADC можно взять в регистре CH, это делает команду короче и быстрее, чем при использовании непосредственного операнда. CH равен 0 во время выполнения цикла, так как CX изменяется от 8 до 0. Вот что получилось в итоге:

```
1 use16              ;Генерировать 16-битный код
2 org 100h           ;Программа начинается с адреса 100h
3
4     movzx ax,[x]   ;AL = x, AH = 0
```

```
 5      mov cx,8        ;Инициализация счётчика цикла
 6  lp:
 7      rol al,1        ;Цилический сдвиг AL на 1 бит влево
 8      adc ah,ch       ;Прибавляем флаг CF к AH, так как CH = 0
 9      loop lp         ;Команда цикла
10      mov [n],bl      ;Сохраняем результат в n
11
12      mov ax,4C00h    ;\
13      int 21h         ;/ Завершение программы
14  ;-------------------------------------------------------------------
15  x   db 89h          ;Байт
16  n   db ?            ;Количество единичных битов в байте
```

Всего 6 команд для подсчета битов в байте. А попробуйте написать то же самое на языке высокого уровня 😉

Упражнение

Объявите в программе строку. Длина строки должна быть больше 8 символов и храниться в байте без знака. Напишите цикл для шифрования строки по алгоритму: первый символ циклически сдвигается вправо на 1 бит, второй символ — на 2 бита, …, 7-й — на 7 битов, 8-й — снова на 1 бит, 9-й на 2 бита и т.д. Затем напишите цикл для расшифровки строки и выведите её на экран.

Следующая часть »

Комментарии:

RoverWWWorm
20-05-2010 22:03

;ура, вроде верно
use16
org 100h

jmp start

;_____
text db ‘hello asmworld!$’
len db 15
rezult1 rb 15

```
rezult2 rb 15
;————————————-

start:
movzx cx,[len]
xor di,di
xor bh,bh
xor bl,bl

lp1:
mov bh,cl ;сохраняем значение cl, чтобы не нарушить цикл
inc bl

cmp bl,8 ;если 8-й символ, то снова двигаем на 1 бит
jnz metka1
mov bl,1

metka1:
mov cl,bl

mov al,byte[text+di]
ror al,cl
mov byte[rezult1+di],al
add di,1

mov cl,bh ;останавливаем счетчик цикла

loop lp1
;————————————

;решил вывести закодированные символы на экран
xor di,di
movzx cx,[len]
mov ah,02h

lp3:
mov dl,byte[rezult1+di]
int 21h
inc di

loop lp3
;————————————————
```

```
;———расшифровка
movzx cx,[len]
xor di,di
xor bh,bh
xor bl,bl

lp2:
mov bh,cl ;сохраняем значение cl, чтобы не нарушить цикл
inc bl

cmp bl,8 ;если 8-й символ, то снова двигаем на 1 бит
jnz metka2
mov bl,1

metka2:
mov cl,bl

mov al,byte[rezult1+di]
rol al,cl
mov byte[rezult2+di],al
add di,1

mov cl,bh ;останавливаем счетчик цикла

loop lp2
;————————————
;вывод символов на экран:
xor di,di
movzx cx,[len]

mov ah,02h
mov dl,13
int 21h
mov dl,10
int 21h

lp4:
mov dl,byte[rezult2+di]
int 21h
inc di
```

loop lp4
;————————————-
mov ah,08h
int 21h

mov ax,4C00h
int 21h

[Ответить]

xrnd
21-05-2010 10:59

Ух. Ты молодец, упорный 🙂
Всё правильно.
Можно немного оптимизировать. Например, заменить

```
    xor bh,bh
    xor bl,bl
```

одной командой

```
    xor bx,bx
```

И «add di,1» лучше делать как «inc di»

[Ответить]

RoverWWWorm
21-05-2010 18:05

aa, точно блин 🙂

[Ответить]

irina
13-06-2010 10:46

помогите пожалуйста найти ошибку в программе, надо выполнить деление натуральных чисел не используя команды div, idiv

[Ответить]

xrnd
14-06-2010 14:35

Я тебе помогу. Такое деление можно написать с помощью вычитаний и сдвигов в цикле. Какой размер чисел для деления? Числа со знаком или без? Можешь в комментарий написать свою программу с ошибкой)

[Ответить]

fufel
28-06-2010 21:01

Безбожно тупил, но кажется осилил.

```
use16
org 100h
jmp start
;————————————————————————-
string db ‘Hello_world!’
string_2 rb 12
string_3 rb 12
;————————————————————————-
start:
mov cx,12
mov dl,1
sub di,di

lp_kod:
mov al,[string+di]
mov bx,cx
sub cx,cx
mov cl,dl
ror al,cl
mov [string_2+di],al
cmp cl,7
je sbros
inc dl
lp:
inc di
sub ax,ax
mov cx,bx
```

```
loop lp_kod
jmp dekod

sbros:
sub dl,6
jmp lp

dekod:
mov cx,12
mov dl,7
sub di,di

lp_dekod:
mov bx,cx
mov al,[string_2+di]
mov cl,dl
ror al,cl
mov [string_3+di],al
cmp cl,1
je nabros
dec dl
lp2:
inc di
mov cx,bx
loop lp_dekod
jmp vivod

nabros:
add dl,6
jmp lp2

vivod:
sub di,di
mov cx,12
mov ah,02h

lp_vivod:
mov dl,[string_2+di]
int 21h
inc di
loop lp_vivod
```

```
sub di,di
mov cx,12
mov ah,02h
mov dl,13
int 21h
mov dl,10
int 21h

lp_vivod_2:
mov dl,[string_3+di]
int 21h
inc di
loop lp_vivod_2

mov ah,08h
int 21h

mov ax,4c00h
int 21h
```

[Ответить]

xrnd
03-07-2010 02:14

Как-то даже взрывает мозг 🙂 Особенно сброс и наброс. Что-то здесь не правильно. Отпишусь, когда проверю.

[Ответить]

fufel
03-07-2010 21:43

Да, точно, надо кометарии в програму добавить, а то сам забуду чего нагородил))).

[Ответить]

xrnd
04-07-2010 20:31

Извиняюсь. Долго проверял, но работает всё верно. Хотя конечно можно проще написать тоже самое.

В первом цикле есть код:

```
    sub cx,cx
    mov cl,dl
```

Здесь CX можно не обнулять, так как новое значение всё равно затирает старое. Кстати, во втором цикле у тебя нет этой команды.
Чуть ниже:

```
    sub ax,ax
```

Тоже обнулять не нужно. Эту команду можно просто убрать.

И гораздо проще можно сделать «сброс» и «наброс». У тебя 3 перехода получается. Можно сделать только один. Например, вот так:

```
    cmp cl,7
    jne ne_sbros
    xor dl,dl
ne_sbros:
    inc dl
    inc di
```

[Ответить]

fufel
04-07-2010 21:01

Регистры я для себя обнулял, мне так проще в отладчике смотреть какие числа куда идут. А про «сброс», «наброс» — хорошее замечание , спасибо.

[Ответить]

argir
19-12-2010 22:59

Почему в предыдущих примерах резервируется под строковую переменную количество байтов без знака $?

Вот, что у меня получилось:

```
use16
org 100h
jmp start

arr db ‘Telephone$’
arrz rb 10 ;резервируем под шифровку
arrd rb 10 ;резервируем под дешифровку
l db 10
pak db 13,10,'(de)coding press any key…$’
pak1 db 13,10,’out press any key…$’
start:
mov ah,09h
mov dx,arr
int 21h ;Выводим слово на экран
mov ah,9
mov dx,pak
int 21h
mov ah,8
int 21h ;Предлагаем его (де)шифровать
movzx bx,[l]
mov cx,7
sub bx,cx
mov si,cx
shifr: dec si
mov al,[arr+si]
rol al,cl ;шифруем символы с 7 по 1
mov [arrz+si],al;сохраняем первую часть шифровки
cmp bx,cx
jna men_str ;если в строке меньше 14символов, то начнем шифровать
символы >8 попозже
mov ah,[arr+si+7]
rol ah,cl ;шифруем символы с 14 по 8
jmp kon
men_str: jnz con_str ;если это последний символ строки, то записываем $
mov ah,24h
kon: mov [arrz+si+7],ah ;сохраняем вторую часть шифровки

con_str: loop shifr

mov ah,09h
mov dx,arrz
int 21h ;выводим шифровку на экран
```

```
mov ah,9
mov dx,pak
int 21h
mov ah,8
int 21h ; предлагаем опять (де)шифровать

mov cx,7
mov si,cx
deshifr: dec si ;дальше тоже самое, но со сдвигом вправо
mov al,[arrz+si]
ror al,cl
mov [arrd+si],al
cmp bx,cx
jna men_str1
mov ah,[arrz+si+7]
ror ah,cl
jmp kon1
men_str1: jnz con_str1
mov ah,24h
kon1: mov [arrd+si+7],ah

con_str1: loop deshifr

mov ah,09h
mov dx,arrd
int 21h ;выводим расшифровку
mov ah,9
mov dx,pak1
int 21h
mov ah,8
int 21h

mov ax,4C00h
int 21h ; happy end
```

[Ответить]

xrnd
20-12-2010 20:11

Там строка выводится по одному символу функцией 02h, поэтому конец строки не нужен.

Вообще не понял, как твоя программа работает 🙂
А что будет, если строка длиннее 14 символов?

[Ответить]

argir
21-12-2010 09:41

Больше 14 символов в строке программа не обработает. Идея была такая — разбить строку по 7символов и обрабатывать символы 1,8;2,9;…. за один цикл, так как они кодируются одинаково. Несложно доработать программу на любую длину строки.

[Ответить]

argir
19-12-2010 23:47

И почему-то русские слова не показываются на экране правильно в моей программе?

[Ответить]

xrnd
20-12-2010 19:35

Дело в том, что при выводе на консоль используется кодировка cp866 (DOS), а строки в программе в кодировке win1251. Если набрать исходник в каком-нибудь редакторе с поддержкой кодировки дос, то будет выводиться нормально.
Например, можно набрать код в программе FAR.

[Ответить]

plan4ik
03-04-2011 19:49

я буду первый в подкатегории форума «Ассемблер в DOS» 😉 нуже промедаретируй мой расказ 😉

[Ответить]

anton
19-12-2010 23:49

Вот, написал, можно сказать, «в лоб». Два куска кода совершенно одинаковы,
отличие в одной команде. Так и не смог придумать, как избежать повтора 🙁

```
use16
org 100h
mov bx,msg ;в bx — адрес строки
xor si,si ;si указывает номер символа
counter_1:
mov cl,1 ;считает 7 символов
code_msg:
mov al,[bx+si] ;символ — в al
cmp al,0dh ;это конец строки ?
je print_msg_code ;да, напечатать
ror al,cl ;кодируем символ
mov [bx+si],al ;заменим символ на закодированный
inc si
inc cl
cmp cl,7 ;7 символов закодировали ?
jna code_msg ;нет еще, продолжим
jmp counter_1 ;да, кодируем след. группу символов
print_msg_code: ;печать закодированной строки
mov ah,09h
mov dx,bx
int 21h
;Декодирую, все то же самое, за исключением
;направления циклического сдвига:
xor si,si
counter_2:
mov cl,1
decode_msg:
mov al,[bx+si]
cmp al,0dh
je print_msg_decode
rol al,cl
mov [bx+si],al
inc si
```

```
inc cl
cmp cl,7
jna decode_msg
jmp counter_2
print_msg_decode: ;печать декодированной строки
int 21h
mov ax,4c00h ;выходим
int 21h
;-------------------------------------------------------
msg db 'Pochemu,esli ya pishu kirilitzey,'
db 'vivoditsa kakaya-to abrakadabra ?',0dh,0ah,'$'
```

Не стал использовать loop, т. к. счетчик уже занят под роры-ролы.

[Ответить]

xrnd
20-12-2010 20:22

Не так уж и «в лоб» получилось. Хорошая программа и работает правильно со строкой любой длины.
LOOP использовать не обязательно. В этом упражнении проще цикл сделать командами условных переходов.

Про кириллицу я ответил на предыдущий комментарий. Дело в кодировке. Windows использует win1251 и в редакторе FASM тоже эта кодировка. А на консоль выводится в кодировке cp866. Поэтому и получается абракадабра.
Нужно исходник набрать в 866 кодировке и тогда русские буквы будут отображаться. Например, можно файл набрать в FARe.

[Ответить]

anton
20-12-2010 21:11

Спасибо, тогда буду набирать код в FARe.

[Ответить]

plan4ik
03-04-2011 19:42

не обязательно можно в реестре изменить значение кодировки в консольных приложениях … HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\Control\Nls\CodePage->OEMCP поменять с 866 на 1251 и не надо прибегать к фар’у 😉

как одобрят скину ссылку на форум 😉

[Ответить]

xrnd
08-04-2011 19:06

Тогда нормальные консольные приложения будут кракозябы печатать.

[Ответить]

plan4ik
08-04-2011 19:59

tak nado nastroykah prilojenia nastroit shrift na Lucida Console i sohranit kak postoianuu nastroyku i vse budet ok 😉 uje provereno godami)))

[Ответить]

xrnd
20-12-2010 20:25

Избежать повтора можно, написав процедуру. Один из переданных параметров будет определять направление циклического свига. Вправо — шифрование, влево — расшифровка.

[Ответить]

anton
20-12-2010 21:13

А как можно в качестве параметра передать направление сдвига ?

[Ответить]

IgorKing
21-12-2010 13:39

1-влево
2-вправо

[Ответить]

anton
21-12-2010 16:43

IgorKing, что-то я не совсем понял твое объяснение, я бы даже сказал, совсем не понял 🙂 В виде кода можешь показать ?

[Ответить]

IgorKing
21-12-2010 17:11

Ну в смысле если значение параметра 1, то будет сдвиг влево, если 2, то вправо.
А в самой процедуре просто сравниваешь параметр с 1, если да сдвиг влево, иначе вправо. Хотя, если честно код не читал поэтому вот просто пример.

…….

```
cmp [arg],1
jnz Ror
rol al,1
jmp Contine
Ror:
ror al,1
Contine:
```

…….

[Ответить]

anton
21-12-2010 18:12

А, ну понятно. Но так можно обойтись и без подпрограммы. Просто после шифровки строки пометить какой-нибудь байт в памяти, и потом уже смотреть: байт помечен — ага, ror обходим, используем rol(расшифровываем). Вопрос был немножко в другом, как передать в

подпрограмму параметр(направление сдвига), внутри самой же подпрограммы уже не заботиться о нем.

[Ответить]

IgorKing
21-12-2010 19:15

Т.е. без проверки?
Незнаю, идея бредовая, но может быть можно в качестве параметра передавать код операции или что-то вроде…В таких вопросах нам может помочь только наш Сенсей!

[Ответить]

anton
21-12-2010 20:14

Получилось! 🙂 Правда, делал в MASMe (в нем пока поуверенней себя чувствую), вместо подпрограммы использовал макрос.

```
cseg segment

org 100h

code_decode macro shft

local counter,code_decode,print_msg

mov bx,offset msg
xor si,si
counter:
mov cl,1
code_decode:
mov al,[bx+si]
cmp al,0dh
je print_msg
shft al,cl
mov [bx+si],al
inc si
inc cl
cmp cl,7
```

```
jna code_decode
jmp counter
print_msg:
mov ah,09h
mov dx,bx
int 21h
endm

start:
code_decode ror
code_decode rol

int 20h

;-------------------------------------------------------------------
msg db 'Goodbye,Amerika!Hello,word!',0dh,0ah,'$'

cseg ends

end start
```

[Ответить]

xrnd
21-12-2010 21:11

Программа получилась короче, но код будет таким же, потому что в каждом вызове макроса подставляются все команды.

А с процедурой код получился бы меньше 🙂

На FASMe тоже можно написать аналогичный макрос (см. Часть 28)

[Ответить]

xrnd
21-12-2010 21:13

Можно сделать самомодифицирующийся код.
Сначала зашифровать, потом пропатчить команду в коде, чтобы изменить направление сдвига, и снова запустить.

Но это уже извращение получается 🙂

[Ответить]

anton
21-12-2010 22:18

Нет уж, на сегодня с меня хватит извращений, голова уже пухнет.
Спасибо, xrnd и IgorKing за толковые ответы, буду двигаться дальше 🙂

[Ответить]

xrnd
21-12-2010 21:04

В общем, правильно. Но я бы сделал 0 или не ноль.
Параметр можно передать через регистр или через стек.
Например, у тебя не используется регистр DI, можно в него поместить значение, которое будет определять направление сдвига.

[Ответить]

юля
21-12-2010 21:58

Как производится циклический арифметический сдвиг?

[Ответить]

xrnd
22-12-2010 00:26

Не очень понятно, что имеется в виду. Я о таком не слышал.
При арифметическом сдвиге знаковый бит сохраняется, поэтому неясно, как он может быть ещё и циклическим? 🙂

Может быть, циклически сдвигаются все биты кроме старшего?
Тогда это можно сделать с помощью сдвига и логических операций:

```
    mov al,10001111b   ;AL = 10001111b
    sar al,1  ;Арифметический сдвиг, CF=выдвинутый бит
    jc set_bit
    and al,10111111b ;Сбросить 6-й бит
    jmp continue
set_bit:
    or al,01000000b ;Установить 6-й бит
```

```
continue:
    ;Результат AL = 11000111b
```

[Ответить]

Гость
17-01-2011 21:25

use16
org 100h
jmp start
;————————————————————————————
vk db ‘parolpa$’
m du 8
;————————————————————————————
start:
mov si,[m] ;конец строки +1
dec si ; конец строки
mov cx,[m] ; cx =8
s4t4ik1:
mov dl,byte[vk+si]; перемешается от 7 символа к 0
cmp cx,8 ; если символ 7 перемешаемся делаем смешение 1
Jz metca ;
ror dl,cl ; смешение
mov [vk+si],dl ; сохраняем смешение
dec si ; si -1
loop s4t4ik1
mov si,[m] ; востонавливаем значения
mov cx,[m] ; для
dec si ; расшифровки
jmp s4t4ik2 ; переходим к расшифровки
;————————————————
s4t4ik2:
mov dl,byte[vk+si];
cmp cx,8
Jz metca1
rol dl,cl
mov [vk+si],dl
dec si
loop s4t4ik2
jmp Print ; выводим конечный расшифрованный результат

```
;_____________________
metca:; необходима для 7симола
mov dl,byte[vk+si]
ror dl,1
mov [vk+si],dl
dec cx
dec si
jmp s4t4ik1
;_____________________
metca1:; необходима для 7симола
mov dl,byte[vk+si]
rol dl,1
mov [vk+si],dl
dec cx
dec si
jmp s4t4ik2
;_____________________
Print:
mov ah,02h
mov cx,[m]
mov di ,0
s4t4ik3:
mov dl,byte[vk+di];
int 21h
inc di
loop s4t4ik3
mov ah,08h
int 21h
mov ax,4C00h
int 21h
```

[Ответить]

Гость
17-01-2011 21:43

Чучуть с циклом перепутал.
Он 1-7
А у меня 1-7 ,+8элемент сдвинуть на 1 , -это как 1 цикл.
Поэтому вышил такой страный
После 9 символа нет действий , так как если цикл , не зная конца строки

, просто смешение задать.
А если с конец строки , то там длинно слишком выходит

[Ответить]

xrnd
18-01-2011 22:02

Замороченно как-то получилось.
Проще всё-таки начинать с первого символа строки. Можно сделать цикл командами перехода, а счетчик держать не в CX.

[Ответить]

Shov
01-04-2011 00:44

```
use16
org 100h
jmp start
;_________
wor_d db ‘this is word!’
code_w db 12 dup(?)
;_________________
start:
mov cl,1
xor di,di

body:
mov al,byte[wor_d+di]
ror al,cl
mov byte[code_w+di],al
inc di
inc cl
cmp di,8
je cl2one
cmp di,12
je output
jmp body

cl2one:
mov cl,1
```

```
jmp body
output:
xor di,di
xor ax,ax

oloop:
mov dl,byte[wor_d+di]
mov ah,02h
int 21h
inc di
cmp di,12
jne oloop

mov dl,10
mov ah,02h
int 21h
mov dl,13
mov ah,02h
int 21h
xor di,di
oloop2:
mov dl,byte[code_w+di]
mov ah,02h
int 21h
inc di
cmp di,12
jne oloop2

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

[Ответить]

xrnd
01-04-2011 16:30

Шифрование написано хорошо, но нет расшифровки строки.

[Ответить]

plan4ik
04-04-2011 17:36

я в бешенстве xrnd помоги пожалуйста почему не работает такой код:
mov [deStr+di], dl ???

```
use16
org 100h
jmp enCode

;==[data]=======================================

enStr db «HeL1o asmworld and fasm!», 0
deStr db 32 dup (‘$’)
max db 7

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

enCode:
mov si, enStr ; адресс строки
xor di, di ; общий счетчик символов
en_loop:
xor cx, cx
mov ch, [max] ; ch = счетчик
_encode:
inc cl ; cl = для сдвига
mov dl, [si]
cmp dl, 0 ; конец строки
je Exit_encode ; подобие break
ror dl, cl
inc si
mov [deStr+di], dl ; ОШИБКА не записывает
inc di ; общий счетчик символов
dec ch
jnz _encode
jmp en_loop

Exit_encode:

; вывод закодированой строки
mov ah, 2
mov si, deStr-1
```

```
pr_deStr:
inc si
mov dl, [si]
cmp dl, '$'
jz Exit
int 21h
jmp pr_deStr

Exit:
mov ah, 8
int 21h

mov ax, 4c00h
int 21h
```

программа еще не закончена … ;((

[Ответить]

plan4ik
04-04-2011 18:05

;))) оказывается я написал баг программу)) 'H' ror 1 == '$')))
поэтому на выводе строки — первый же символ заканчивал мою программу)) , блин придется менять

[Ответить]

plan4ik
04-04-2011 18:58

Ура почти все работает!!! 😉 не могу тока понять почему не выводится строка
0ah, 0dh, 'Decoded: $' а так все работает как швейцарские часы, можно даже зашифрованые сообщения в пентагон посылать)))

```
use16
org 100h
jmp enCode

;==[data]==========================================
```

```
source db «HeL1o asmworld and fasm!», 0
enStr db 32 dup (0)
deStr db 32 dup (0)
max db 7
str1 db «Encoded: $»
str2 db 0dh, 0ah, «Decoded: $»

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

; —[Кодирование строки]————————————-

enCode:

mov si, source ; адресс строки
xor di, di ; общий счетчик символов
en_loop:
xor cx, cx
mov ch, [max] ; ch = счетчик
_encode:
inc cl ; cl = для сдвига
mov dl, [si]
cmp dl, 0 ; конец строки
je enCode_print ; подобие break
ror dl, cl
inc si
mov [enStr+di], dl
inc di ; общий счетчик символов
dec ch
jnz _encode
jmp en_loop

;-[Декодирование строки]——————————

deCode: ;

mov si, enStr ; адресс закодированой строки
xor di, di ; общий счетчик символов
de_loop:
xor cx, cx
mov ch, [max] ; ch = счетчик
_decode:
inc cl
```

```
mov dl, [si]
cmp dl, 0
jz deCode_print ; вывод декодированой строки
rol dl, cl
mov [deStr+di], dl
inc si
inc di
dec ch
jnz _decode
jmp de_loop

;—[вывод закодированой строки]——
enCode_print: ;

mov ah, 9
mov dx, str1
int 21h
mov ah, 2
mov si, enStr-1

lp_enStr:
inc si
mov dl, [si]
cmp dl, 0
jz deCode ; декодируем и выводим оригинал
int 21h
jmp lp_enStr

;-[вывод декодированой строки]————————

deCode_print:

mov ah, 9
mov dx, str2
int 21
mov ah, 2
mov si, deStr-1

lp_deStr:
inc si
mov dl, [si]
cmp dl, 0
```

jz Exit
int 21h
jmp lp_deStr

;—[Выход]———————————-

Exit:
mov ah, 8
int 21h
mov ax, 4c00h
int 21h

[Ответить]

xrnd
08-04-2011 20:04

```
    mov dx, str2
    int 21
```

Должно быть 21h, иначе это другое прерывание.

[Ответить]

NimRoen
19-05-2011 12:30

;шифровка строки по алгоритму циклического сдвига каждого байта
;строки на количество битов равное индексу кодируемого байта
;с основанием 7, т.е. 1-7,1-7,..
use16
org 100h
  jmp main
;=====================================;
strDecode db ‘Пример зашифровываемой строки’,0
strBR db 10,13,0
;=====================================;

;=====================================;
main:
  mov bx,strDecode ;
  mov ax,bx ;
  call proc_strOut ;

```
  mov bx,strBR ;
  call proc_strOut ;выводим начальную строку
  mov bx,ax
  xor ax,ax ;
  call proc_strDE ;шифруем строку, выводим
  mov ah,1 ;
  call proc_strDE ;дешифруем строку, выводим
;_____________________;
exit:
  mov ax,4c00h
  int 21h
;=====================================;

;=====================================;
;кодируем/декодируем строку
;bx — указатель на строку
;ah — флаг работы подпрограммы (0 — шифруем, 1 — дешифруем)
proc_strDE:
  xor cx,cx
  mov si,cx
get_char:
  inc cl
  js slide_more7 ;если счетчик сдвига равен 8
slide:
  mov al,[bx+si] ;получаем текущий символ
  cmp al,0
  je display_string ;если конец строки, выводим на экран
  cmp ah,0 ;проверяем флаг работы подпрограммы
  jne decode
  ror al,cl ;шифруем, если ah=0
  jmp short end_de_char
decode:
  rol al,cl ;дешифруем, если ah0
end_de_char:
  mov [bx+si],al
  inc si
  jmp short get_char
slide_more7:
  mov cx,1 ;начинаем счетчик сдвига сначала
  jmp short slide
display_string:
```

```
  mov ax,bx ;
  call proc_strOut ;
  mov bx,strBR ;
  call proc_strOut ;отображаем строку
  mov bx,ax
proc_strDE_end:
  ret
;=====================================;

;=====================================;
;посимвольный вывод кириллической строки на экран (0 —
завершающий символ)
;bx — указатель на строку
proc_strOut:
  push ax
  push dx
  push si
  mov si,-1
proc_strOut_getChar:
  inc si mov al,[bx+si]
  cmp al,0xf0
  jb proc_strOut_charLessF0
  and al,0xef
  jmp short proc_strOut_displayChar
proc_strOut_charLessF0:
  cmp al,0xef
  ja proc_strOut_displayChar
  cmp al,0xc0
  jnb proc_strOut_charC0EF
  cmp al,0
  je proc_strOut_end
  jmp short proc_strOut_displayChar
proc_strOut_charC0EF:
  and al,0xbf
proc_strOut_displayChar:
  mov ah,02h
  mov dl,al
  int 21h
  jmp short proc_strOut_getChar
proc_strOut_end:
  pop si
```

```
  pop dx
  pop ax
  ret
;=====================================;
```

[Ответить]

xrnd
23-06-2011 14:51

В принципе всё хорошо.
Вроде есть одна ошибка:

```
    inc cl
    js slide_more7 ;если счетчик сдвига равен 8
```

Переход по флагу SF будет выполнен, если старший бит равен 1, то есть CL=128 и больше.

Очень интересная процедура proc_strOut. Я так понял, она преобразует кодировку?
Есть более простой способ — сделать табличку из 256 байт и заменять байты по этой таблице (существует даже специальная команда XLAT)

[Ответить]

NimRoen
24-06-2011 10:02

Да, надо вместо js вставить cmp cl,8 и jb.
Опять тетраду с байтом перепутал 🙂
а xlat моя пока не знать 🙂

[Ответить]

NimRoen
24-06-2011 10:03

сорри, не jb, а jnb

[Ответить]

алекс
16-03-2012 20:09

```
use16
org 100h

jmp start

string db 13,10,'Horosho v derevne letom$'
press db 13,10,'Press any key…$'
length db 23

start:

mov ah,09h
mov dx,string
int 21h
int 21h
movzx cx,[length]
xor si,si
mov dl,1

incript:
mov ax,cx
mov cl,byte[si+1]
ror [string+si],cl
inc si
mov cx,ax
loop incript
mov ah,09h
mov dx,string
int 21h

movzx cx,[length]
xor si,si
mov dl,1

decript:
mov ax,cx
mov cl,byte[si+1]
rol [string+si],cl
inc si
```

```
mov cx,ax
loop decript

mov ah,09h
mov dx,string
int 21h
mov ah,09h
mov dx,press
int 21h
mov ah,08h
int 21h
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.