Учебный курс. Часть 13. Циклы и команда LOOP

Автор: xrnd | Рубрика: Учебный курс | 19-04-2010 | Распечатать запись

До этой части все наши программы выполнялись последовательно — в них не было ветвлений и переходов. Сегодня мы научимся делать простейшие циклы. Циклом называется повторяющееся выполнение последовательности команд. Но для начала нужно научиться объявлять метки.

Синтаксис объявления меток

Метка представляет собой символическое имя, вместо которого компилятор подставляет адрес. В программе на ассемблере можно присвоить имя любому адресу в коде или данных. Обычно метки используются для организации переходов, циклов или каких-то манипуляций с данными. По сути имена переменных, объявленных с помощью директив объявления данных, тоже являются метками. Но с ними компилятор дополнительно связывает размер переменной. Метка объявляется очень просто: достаточно в начале строки написать имя и поставить двоеточие. Например:

```
m1: mov ax,4C00h
    int 21h
```

Теперь вместо имени *m1* компилятор везде будет подставлять адрес комады *mov ax,4C00h*. Можно объявлять метку на пустой строке перед командой:

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

Имя метки может состоять из латинских букв, цифр и символов подчёркивания, но должно начинаться с буквы. Имя метки должно быть уникальным. В качестве имени метки нельзя использовать директивы и ключевые слова компилятора, названия команд и регистров (в этом случае FASM покажет сообщение об ошибке). FASM различает регистр символов в именах меток. Можно также объявлять несколько меток на один адрес. Например:

```
no_error:
exit_app:
m1: mov ax,4C00h
```

Подробнее о синтаксисе объявления меток рассказывается в части 27.

Команда LOOP

Для организации цикла предназначена команда LOOP. У этой команды один операнд — имя метки, на которую осуществляется переход. В качестве счётчика цикла используется регистр CX. Команда LOOP выполняет декремент CX, а затем проверяет его значение. Если содержимое CX не равно нулю, то осуществляется переход на метку, иначе управление переходит к следующей после LOOP команде.

Содержимое CX интерпретируется командой как число без знака. В CX нужно помещать число, равное требуемому количеству повторений цикла. Понятно, что максимально может

быть 65535 повторений. Ещё одно ограничение связано с дальность перехода. Метка должна находиться в диапазоне -127…+128 байт от команды LOOP (если это не так, FASM сообщит об ошибке).

Пример цикла

В качестве примера я приведу простую программу, которая будет печатать все буквы английского алфавита. ASCII-коды этих символов расположены последовательно, поэтому можно выводить их в цикле. Для вывода символа на экран используется функция DOS *02h* (выводимый байт должен находиться в регистре DL).

```
 1 use16                 ;Генерировать 16-битный код
 2 org 100h              ;Программа начинается с адреса 100h
 3
 4     mov ah,02h        ;Для вызова функции DOS 02h - вывод символа
 5     mov dl,'A'        ;Первый выводимый символ
 6     mov cx,26         ;Счётчик повторений цикла
 7 metka:
 8     int 21h           ;Обращение к функции DOS
 9     inc dl            ;Следующий символ
10     loop metka        ;Команда цикла
11
12     mov ah,09h        ;Функция DOS 09h - вывод строки
13     mov dx,press      ;В DX адрес строки
14     int 21h           ;Обращение к функции DOS
15
16     mov ah,08h        ;Функция DOS 08h - ввод символа без эха
17     int 21h           ;Обращение к функции DOS
18
19     mov ax,4C00h      ;\
20     int 21h           ;/ Завершение программы
21 ;-------------------------------------------------------
22 press:
23     db 13,10,'Press any key...$'
```

Команды *«int 21h»* и *«inc dl»* (строки 8 и 9) будут выполняться в цикле 26 раз. Для того, чтобы программа не закрылась сразу, используется функция DOS *08h* — ввод символа с клавиатуры без эха, то есть вводимый символ не отображается. Перед этим выводится предложение нажать любую кнопку (но *Reset* лучше не нажимать). Для примера адрес строки объявлен с помощью метки. Символы с кодами 13 и 10 обозначают переход на следующую строку (символ 13(0Dh) называется CR — *Carriage Return* — возврат каретки, а символ 10(0Ah) LF — *Line Feed* — перевод строки . Эти символы унаследованы со времён древних телетайпов, когда текст печатался, как на печатной машинке). Так выглядит результат работы программы:

Вложенные циклы

Иногда требуется организовать вложенный цикл, то есть цикл внутри другого цикла. В этом случае необходимо сохранить значение CX перед началом вложенного цикла и восстановить после его завершения (перед командой LOOP внешнего цикла). Сохранить значение можно в другой регистр, во временную переменную или в стек. Следующая программа выводит все доступные ASCII-символы в виде таблицы 16×16. Значение счётчика внешнего цикла сохраняется в регистре BX.

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

 3
 4     mov ah,02h        ;Для вызова функции DOS 02h - вывод символа
 5     sub dl,dl         ;Первый выводимый символ
 6     mov cx,16         ;Счётчик внешнего цикла (по строкам)
 7 lp1:
 8     mov bx,cx         ;Сохраняем счётчик в BX
 9     mov cx,16         ;Счётчик внутреннего цикла (по столбцам)
10 lp2:
11     int 21h           ;Обращение к функции DOS
12     inc dl            ;Следующий символ
13     loop lp2          ;Команда внутреннего цикла
14
15     mov dh,dl         ;Сохраняем значение DL в DH
16     mov dl,13         ;\
17     int 21h           ; \
18     mov dl,10         ; / Переход на следующую строку
19     int 21h           ;/
20     mov dl,dh         ;Восстанавливаем значение DL
21
22     mov cx,bx         ;Восстанавливаем значение счётчика
23     loop lp1          ;Команда внешнего цикла
24
25     mov ah,09h        ;Функция DOS 09h - вывод строки
26     mov dx,press      ;В DX адрес строки
27     int 21h           ;Обращение к функции DOS
28
29     mov ah,08h        ;Функция DOS 08h - ввод символа без эха
```

```
30      int 21h             ;Обращение к функции DOS
31
32      mov ax,4C00h        ;\
33      int 21h             ;/ Завершение программы
34  ;-------------------------------------------------------
35  press db 13,10,'Press any key...$'
```

Как видите, всё довольно просто 🙂 Результат работы программы выглядит вот так:

Упражнение

Напишите программу для вычисления степени числа 3 по формуле $a = 3^n$. Число a — 16-битное целое без знака, число n — 8-битное целое без знака (используйте $n<11$, чтобы избежать переполнения). Проверьте работу программы в отладчике (нажимайте F7 на команде LOOP, чтобы осуществить переход). Результаты можете выкладывать в комментариях.

Следующая часть »

Комментарии:

RoverWWWorm
28-04-2010 21:54

:только я не совсем понял, надо ли выводить результат на экран
use16
org 100h

mov cx,n
mov bx,3
mov ax,bx
lp:

```
imul bx,ax
loop lp

mov [a],bx

mov ax,4c00h
int 21h

;———————-
n db 10
a dw ?
```

[Ответить]

xrnd
28-04-2010 22:46

Хорошо написано, но не совсем верно. Смотри сам, если n=1, то должно получиться 3, а твоя программа получит 9. На одно умножение больше получается. Кроме того, если a — число без знака, то результат лучше считать командой MUL.

Чтобы вывести число на экран, нужно сначала преобразовать его в текстовую строку (или в отдельные символы), поэтому тут всё не так просто. Но можешь попробовать 🙂 Я хотел этой теме отдельную статью посвятить.

[Ответить]

RoverWWWorm
07-05-2010 22:23

```
; а если так, то с n=1, должно вроде правильно показывать, но что то у меня
; отладчик много циклов показывает, если я ввожу к примеру n db 2
use16
org 100h

mov cx,n
mov bx,1
mov ax,3
lp:
imul bx,ax
loop lp

mov [a],bx

mov ax,4c00h
int 21h

;———————-
n db 10
a dw ?
```

[Ответить]

xrnd
08-05-2010 02:49

Я в первый раз не заметил. Да, циклов будет много 🙂 Потому что команда «mov cx,n» загружает в CX адрес переменной, а не её значение. Не хватает квадратных скобок. И лучше использовать команду movzx, так как размер n — 8 бит, а размер CX — 16 бит.

[Ответить]

RoverWWWorm
08-05-2010 10:34

```
use16
org 100h

movzx cx,[n]
mov bx,1
mov ax,3
lp:
imul bx,ax
loop lp

mov [a],bx

mov ax,4c00h
int 21h

;_____________-
n db 10
a dw ?
;уф, ну все теперь вроде правильно прога выводит 🙂
```

[Ответить]

xrnd
09-05-2010 17:30

Почти правильно 🙂 Тебе ещё нужно использовать команду MUL вместо IMUL, чтобы было совсем хорошо. 3^{10}=59049. Если ты умножаешь командой IMUL, то получится ошибка, потому что результат этой команды — слово со знаком. Предел его значения 32767.

[Ответить]

RoverWWWorm
10-05-2010 17:31

```
use16
org 100h

movzx cx,[n]
mov bx,3
mov ax,1
lp:
mul bx
loop lp
```

```
mov word[a],ax
mov word[a+2],dx

mov ax,4c00h
int 21h

;————————————-
n db 10
a dw ?
;ну, а сейчас хоть правильно 🙂
```

[Ответить]

xrnd
10-05-2010 18:18

Да теперь правильно 🙂 Только одна команда лишнаяя: «mov word[a+2],dx». Либо надо объявлять a как двойное слово с помощью dd.

[Ответить]

xrnd
07-05-2010 16:29

Как дела? Что-то давно не пишешь. Если не получается что-то, я подскажу.

[Ответить]

RoverWWWorm
07-05-2010 18:59

Салют! Да по учебе дела туго идут, практика, вот и времени нет, с документами каждый день вожусь. А так бы с радостью читал бы уроки. Просто каждый день захожу в твой сайт, смотрю какие новые уроки появились… Ну может сегодня почитаю малость)

[Ответить]

RoverWWWorm
07-05-2010 20:55

Ну если у меня вопросы будут, то сдесь на сайте долго все переписываться, вот если что icq : 417501814, добавляй 🙂

[Ответить]

levinter
28-06-2010 00:02

извени но немного не понятно loop ставится как конец метки или конец метки както подругому надо закончит в приведеном коде вроде вы оканчивали просто пустой строкой. а статья офиген я на основе етой статьи уже чуток поправил язык pure basic и добавил в него пару новых функций тока они не очень полные и недороботаные.

[Ответить]

xrnd
03-07-2010 01:55

Метка обозначает только адрес перехода. А loop — это команда процессора. Метка ей указывается для перехода на этот адрес, если цикл не завершён.
Конец цикла никак не обозначается.
В чистом ассемблере нет специальных конструкций для циклов или условий, просто указываются адреса переходов. Но адреса запоминать и писать неудобно, поэтому были придуманы метки.
Я ответил на твой вопрос? 🙂

[Ответить]

xrnd
03-07-2010 01:57

Пустую строку после loop я добавляю для удобства чтения кода. Проще говоря, мне так нравится 🙂 Но она не обязательна и никак на программу не влияет.

[Ответить]

levinter
01-07-2010 13:22

ну плиз ответьте!!!!!!!

[Ответить]

xrnd
03-07-2010 01:57

Ответил 🙂

[Ответить]

levinter
03-07-2010 19:05

да ответил получается метку не как нельзя закончить? тоесть написал метку и сразу ее испонять?

[Ответить]

xrnd
03-07-2010 19:52

Метка не исполняется. Это просто обозначение адреса 🙂 В ассемблере исполняются только команды процессора.

[Ответить]

levinter
03-07-2010 19:54

нет я имею ввиду что конец метки невозможно указать и поэтому будет исполнятся весь код от начала метки до команды loop?

[Ответить]

xrnd
03-07-2010 20:02

Да, именно так. Никаких концов метки не бывает)))

[Ответить]

levinter
03-07-2010 20:03

все терь все понял))))спс))

[Ответить]

Granus
15-08-2010 19:55

```
use16
org 100h

mov ah,01h
int 21h
sub al,'0'
mov cl,al
mov ax,1
mov bl,3
lp1:
mul bx
loop lp1
mov bl,10
xor cx,cx
lp2:
xor dx,dx
div bx
push dx
inc cx
test ax,ax
jnz lp2
mov ah,02h
mov dl,13
int 21h
mov dl,10
int 21h
lp3:
pop dx
add dl,'0'
int 21h
loop lp3
```

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

[Ответить]

xrnd
18-08-2010 15:48

Хорошая программа. Даже с вводом и выводом чисел 🙂
Есть только один небольшой недочёт. Ты предполагаешь, что регистры изначально содержат нулевые значения, а это может быть не так.
Лучше вместо

```
mov cl,al
```

сделать

```
movzx cx,al
```

Иначе, если CH не равно 0, то цикл будет выполняться больше, чем нужно.
Аналогично лучше заменить

```
mov bl,3
```

на

```
mov bx,3
```

[Ответить]

argir
04-12-2010 15:40

Я во второй программе удалил команды
mov dl,13 ;\
int 21h
Ответ получился тот же. Почему?

[Ответить]

xrnd
04-12-2010 16:29

Возможно, под Windows работает и с одним символом 0Ah. Однако, в программах для DOS правильнее всё-таки использовать 2 символа. В чистом досе или других эмуляторах может заглючить 🙂

В разных операционных системах существует 3 различных варианта обозначения перехода на новую строку:
1) 0Ah

2) 0Dh
3) 0Dh, 0Ah

[Ответить]

xrnd
04-12-2010 16:34

Кстати, такой же ответ получится, если поменять местами 13 и 10 🙂
Но самый правильный вариант — тот что в примере, я специально уточнял этот вопрос.
Также обозначается конец строки в текстовых файлах DOS и Windows.

[Ответить]

argir
05-12-2010 00:41

Пример который считает и 3°=1
use16
org 100h

mov ax,1
mov dx,1
mov cl,[n]
inc cl
sub bx,bx
stepen:
add dx,bx
mul dx
mov bx,3
loop stepen
mov [a],ax

mov ax,4C00h
int 21h
;————————————————————————-

n db 11
a dw ?

[Ответить]

xrnd
05-12-2010 10:10

Работает правильно, хотя я не сразу понял, как это происходит 🙂

Лучше не предполагать, что CH=0 при запуске программы, и писать:

```
movzx cx,[n]
inc cx
```

Для n = 11 результат не влезет в слово без знака: 3^11=17714, а максимально возможное значение 65535.

[Ответить]

argir
06-12-2010 23:25

Да прокололся. И сначала не понял, что 17714 опечатка, пока не достал калькулятор.

[Ответить]

xrnd
06-12-2010 23:59

Сорри, одна цифра потерялась.

[Ответить]

IgorKing
08-12-2010 15:19

Посмотри, пожалуйста (проблемное место помечено коментарием):

```
PowerMetod:
push bp
mov bp,sp
push bx
push dx
push di
push si
push cx
mov bx,[bp+10]
mov dx,[bp+8]
xor ax,ax
xor di,di
xor si,si
mov cx,[bp+6]
sub cx,[bp+4]
inc cx
.Symbol_Check:
mov al,byte[bx+di]
xchg bx,dx
cmp al,byte[bx+si]
inc di
jnz .Discrepancy
mov ax,di
inc si
cmp byte[bx+si],'$'
jz .Exit
jnz .Coincidence
.Discrepancy:
xor si,si
.Coincidence:
xchg bx,dx
loop .Symbol_Check ; Почему не происходит переход? Т.е. доходит и cx
; обнуляется?
```

xor ax,ax
.Exit:
pop cx
pop si
pop di
pop dx
pop bx
pop bp
ret 8

[Ответить]

xrnd
08-12-2010 22:19

Переход происходит 🙂
Ты наверно нажимаешь F8 в отладчике. Я в упражнении писал, что для перехода надо нажимать F7.
При нажатии F8 Turbo Debugger прогоняет все итерации цикла и переходит к следующей команде.

Кстати, нашёл ошибку в твоей процедуре:

```
    cmp al,byte[bx+si]
    inc di
    jnz .Discrepancy
```

Команда INC изменит флаги и здесь всё время будет переход.

Ещё можно здесь оптимизировать немного:

```
    cmp byte[bx+si],'$'
    jz .Exit
    jnz .Coincidence
```

Либо равно, либо не равно — третьего быть не может 🙂 Если немного по-другому разместить метки, то можно одной командой перехода обойтись.

[Ответить]

Борис
21-12-2010 23:11

Приветствую мои 5 копеек, считает, dx вобще не учитывал
use16 ;gen 16-bit cod
org 100h ;begin 100h
movzx cx,[n] ;loop = n
movzx bx,[b]
mov al,1 ;b^0
my_loop:
mul bx
loop my_loop
mov [a],ax

```
exit_progr:
mov ax,4C00h ;\
int 21h ;/ Çàâåðøåíèå ïðîãðàììû
;————————————————————————————-
b db 3
n db 10 ;0<n<11
a dw ?
```

[Ответить]

xrnd
22-12-2010 00:31

Всё правильно, хорошая программа.
Есть только один маленький недочет: вместо команды «mov al,1» надо использовать «mov ax,1». Если в AH будет какой-то мусор, то он испортит результат вычисления.

[Ответить]

Гость
12-01-2011 19:23

```
use16 ;Ãåíåðèðîâàòü 16-áèòíûé êîä
org 100h ;Ïðîãðàììà íà÷èíàåòñÿ ñ àäðåñà 100h
mov ax, 0; ax=0000
mov al,3 ; ax=0003
mov dl,3 ; dx=0003
mov cl,[n]; cx =00[n]
x:
mul dl ; al=al*dl
loop x
mov [a], ax ; a = al
mov ax,4C00h
int 21h
;————————————-
n db 4
a dw ?
```

[Ответить]

xrnd
13-01-2011 17:01

Принцип ты правильно понял, но есть ошибки.

Во-первых, не нужно предполагать, что в регистрах изначально находятся нули.
Turbo Debugger регистры обнуляет, но когда программа запускается без него, в регистрах могут быть другие значения.

Вместо:

```
    mov cl,[n]
```

Нужно написать:

```
    movzx cx,[n]
```

Во-вторых, самая первая команда не нужна. При умножении получится AX = AL*DL. Вообще лучше умножать не байты а слова, это даст больший диапазон результата. Например DX:AX = AX*BX.

В-третьих, считать твоя программа будет неправильно. Если n = 1, то получится a = 9. Если n = 2, a = 27.

[Ответить]

Гость
13-01-2011 17:42

А теперь
use16
org 100h
mov al,3 ;
mov dl,3 ;
movzx cx,[n];
movzx dx,dl ;
mov [a] ,dx ;1 степень
dec cx ; меньшаем на 1 схсчётчик
x:
mul dl ; al=al*dl
loop x
mov [a], ax ; a = al
mov ax,4C00h
int 21h
;————————-
n db 4
a dw ?

[Ответить]

xrnd
15-01-2011 00:51

Этот вариант лучше, хотя для n = 1 будет считаться неправильно. Если CX = 0, то цикл выполнится 65536 раз 😉

команды

```
    mov dl,3
    movzx dx,dl
```

Можно заменить одной:

```
    mov dx,3
```

[Ответить]

Knight212
08-02-2011 16:54

use16
org 100h

mov ax, 3
mov cl, [n]
cbw
mov bx, 3

m:
mul bx
loop m

mov [a], ax

mov ax, 4C00h
int 21h

n db 3
a dw ?

[Ответить]

Knight212
08-02-2011 17:04

Вместо
mov ax, 3
нужно
mov ax, 1

[Ответить]

xrnd
09-02-2011 17:38

Хорошая программа, но есть один небольшой недочет.

Команда CBW работает с регистром AX, поэтому она тут бесполезна.
Чтобы обнулить старшую часть счетчика цикла, нужно написать:

```
    movzx cx,[n]
```

При запуске в отладчике, регистры содержат нулевые значения. Но на самом деле может быть не так. CH надо обнулить явно, иначе программа будет работать не правильно, если при запуске там какой-то мусор.

[Ответить]

AlexKor_KAA
10-02-2011 12:52

Здравствуйте! Классные статьи — хотя я опытный программист, читаю Ваши (твои?) уроки с удовольствием и прорабатываю их.
Кажется, по моему в этой статье у тебя опечатка (или что?) — странно что никто не заметил. Ты пишешь — «Символы с кодами 13 и 10 обозначают переход на следующую строку (символ 13(0Dh) называется LF – Line Feed – перевод строки, а символ 10(0Ah) CR – Carriage Return – возврат каретки.» Но кажется наоборот — код 10 (0Ah) это LF, а код 13 (0Dh) это CR. Это в написано в любой таблице символов.

[Ответить]

xrnd
10-02-2011 17:14

Можно на «ты».

Да, я действительно перепутал CR и LF 🙂 Спасибо, исправлю.

[Ответить]

Георгий
10-02-2011 20:10

```
;Еще раз здравствуйте:)
use16
org 100h

dec [n];
movzx cx,[n]
mov al,3
mov bl,3
lp:
mul bl
loop lp

mov [a],ax

mov ax,4C00h
int 21h
; —init data —
n db 4; дана степень
a dw ?
```

[Ответить]

xrnd
11-02-2011 17:22

Цикл написан правильно.
Но лучше всё-таки умножать 16-битные регистры, а не 8-битные. Диапазон результата будет больше.

[Ответить]

Георгий
11-02-2011 20:27

```
; немного странную реализацию сделал, вот
use16
org 100h

movzx cx,[n]
mov ax,1
mov bx,3
lp:
mul bx
loop lp

mov [a],ax

mov ax,4C00h
int 21h
; –init data –
n db 3; дана степень
a dw ?
```

[Ответить]

xrnd
14-02-2011 12:39

Да, этот вариант лучше.

[Ответить]

A13R42
20-02-2011 16:28

Привет. Прогнал в отладчике — вроде работает. Увидел в комментах, что можно попроще сделать, но сам не догадался =)).
Всё же, проверь пожалуйста на наличие ошибок. Заранее спасибо.

```
;————————
use16
org 0x100

movzx cx,[n]
mov bx,3
mov ax,3
loop lab
mov [a],3
mov ax,0x4c00
int 0x21
lab:
mul bx
loop lab
mov [a],ax
mov ax,0x4c00
int 0x21
;————————————
n db 3
a dw ?
```

[Ответить]

xrnd
22-02-2011 20:25

Ошибок нет. Интересно используется первая команда loop 🙂

[Ответить]

annihilator
25-04-2011 18:12

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

mov ax, 1
movzx bx, [x]
movzx cx, [n]

lb:
mul bx
loop lb

mov [a], ax

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

;---------------------------------------------------------
a dw ?
x db 3
n db 10
```

[Ответить]

Ваш комментарий

Имя *

Почта (скрыта) *

Сайт

Добавить

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