
Ассемблер: Системные вызовы и адресация

— преподаватель Л. М. Подольская —

Ассемблер: Системные вызовы Linux

Для использования системных вызовов Linux необходимо выполнить следующие шаги:

- Поместите номер системного вызова в регистр EAX.
- Сохраните аргументы системного вызова в регистрах EBX, ECX и т. д.
- Вызовите соответствующее прерывание (80h).
- Результат обычно возвращается в регистр EAX.

Примеры использования системного вызова

ИСПОЛЬЗОВАНИЕ СИСТЕМНОГО ВЫЗОВА **sys_exit**

```
1 | mov eax,1          ; номер системного вызова (sys_exit)
2 | int 0x80          ; вызов ядра
```

ИСПОЛЬЗОВАНИЕ СИСТЕМНОГО ВЫЗОВА **sys_write**

```
1 | mov edx,4          ; длина сообщения
2 | mov ecx,msg        ; сообщение для записи
3 | mov ebx,1          ; файловый дескриптор (stdout)
4 | mov eax,4          ; номер системного вызова (sys_write)
5 | int 0x80          ; вызов ядра
```

Посмотрите на следующую простую программу, чтобы понять использование регистров в программировании на Ассемблере.

Эта программа отображает 9 звезд на экране вместе с простым сообщением.

```
1  section .text
2      global _start      ;нужно указать для линкера (gcc)
3
4  _start:                ;говорит линкеру о точке входа
5      mov  edx,len      ;длина сообщения
6      mov  ecx,msg      ;сообщение для записи
7      mov  ebx,1        ;файловый дескриптор (stdout)
8      mov  eax,4        ;номер системного вызова (sys_write)
9      int  0x80        ;вызов ядра
10
11     mov  edx,9        ;длина сообщения
12     mov  ecx,s2       ;сообщение для записи
13     mov  ebx,1        ;файловый дескриптор (stdout)
14     mov  eax,4        ;номер системного вызова (sys_write)
15     int  0x80        ;вызов ядра
16
17     mov  eax,1        ;номер системного вызова (sys_exit)
18     int  0x80        ;вызов ядра
19
20  section .data
21  msg db 'Сейчас покажем 9 звёзд',0xa ;сообщение
22  len equ $ - msg     ;длина сообщения
23  s2 times 9 db '*'
```

```
1  Сейчас покажем 9 звёзд
2  *****
```

Следующий пример читает
число с клавиатуры и
отображает его на экране:

```
1 section .data ;Сегмент Data
2 userMsg db 'Пожалуйста, введите любые цифры: ' ;Просим пользователя ввести число
3 lenUserMsg equ $-userMsg ;Длина сообщения
4 dispMsg db 'Вы ввели: '
5 lenDispMsg equ $-dispMsg
6
7 section .bss ;Неинициализированные данные
8 num resb 5
9
10 section .text ;Сегмент Code
11 global _start
12
13 _start: ;Запрос пользователю на ввод
14 mov eax, 4
15 mov ebx, 1
16 mov ecx, userMsg
17 mov edx, lenUserMsg
18 int 80h
19
20 ;Считываем и сохраняем пользовательский ввод
21 mov eax, 3
22 mov ebx, 2
23 mov ecx, num
24 mov edx, 5 ;5 байт (числовой, 1 для знака) этой информации
25 int 80h
26
27 ;Вывод сообщения 'Вы ввели: '
28 mov eax, 4
29 mov ebx, 1
30 mov ecx, dispMsg
31 mov edx, lenDispMsg
32 int 80h
33
34 ;Вывод введённого числа
35 mov eax, 4
36 mov ebx, 1
37 mov ecx, num
38 mov edx, 5
39 int 80h
40
41 ; Код выхода
42 mov eax, 1
43 mov ebx, 0
44 int 80h
```

Ассемблер: Режимы адресации

Адресации на регистр

```
1 | MOV DX, TAX_RATE ; Регистр в первом операнде
2 | MOV COUNT, CX ; Регистр во втором операнде
3 | MOV EAX, EBX ; Оба операнда в регистрах
```

Немедленная адресация

```
1 | BYTE_VALUE DB 150 ; Определена величина byte
2 | WORD_VALUE DW 300 ; Определена величина word
3 | ADD BYTE_VALUE, 65 ; Добавлен немедленный операнд 65
4 | MOV AX, 45H ; Немедленная константа 45H передана на AX
```

Ассемблер: Режимы адресации

Адресация на память

Прямая адресация со смещением

```
1 | BYTE_TABLE DB 14, 15, 22, 45 ; Таблица bytes  
2 | WORD_TABLE DW 134, 345, 564, 123 ; Таблица words
```

1

```
1 | MOV CL, BYTE_TABLE[2] ; Получает 3й элемент BYTE_TABLE  
2 | MOV CL, BYTE_TABLE + 2 ; Получает 3й элемент BYTE_TABLE  
3 | MOV CX, WORD_TABLE[3] ; Получает 4й элемент WORD_TABLE  
4 | MOV CX, WORD_TABLE + 3 ; Получает 4й элемент WORD_TABLE
```

2

Косвенная адресация на память

```
1 | MY_TABLE TIMES 10 DW 0 ; Выделено 10 words (по 2 байта) каждое инициализировано на 0  
2 | MOV EBX, [MY_TABLE] ; Эффективный адрес MY_TABLE в EBX  
3 | MOV [EBX], 110 ; MY_TABLE[0] = 110  
4 | ADD EBX, 2 ; EBX = EBX + 2  
5 | MOV [EBX], 123 ; MY_TABLE[1] = 123
```

3

Инструкция MOV

Синтаксис инструкции MOV:

```
1 | MOV пункт_назначения, источник
```

Инструкция MOV может иметь одну из следующих пяти форм:

```
1 | MOV регистр, регистр
2 | MOV регистр, непосредственное_значение
3 | MOV память, непосредственное_значение
4 | MOV регистр, память
5 | MOV память, регистр
```

Инструкция MOV порой вызывает двусмысленность. Например, посмотрите на утверждения:

```
1 | MOV EBX, [MY_TABLE] ; Эффективный адрес MY_TABLE в EBX
2 | MOV [EBX], 110 ; MY_TABLE[0] = 110
```


Пример:

```
1 section .text
2 global _start ;нужно объявить для линкера
3 _start: ;указывает компоновщику точку входа
4
5 ;пишем имя 'Zara Ali'
6 mov edx,9 ;длина сообщения
7 mov ecx, name ;сообщение для записи
8 mov ebx,1 ;файловый дескриптор (stdout)
9 mov eax,4 ;номер системного вызова (sys_write)
10 int 0x80 ;вызов ядра
11
12 mov [name], dword 'Nuha' ; изменение имени Nuha Ali
13
14 ;запись имени 'Nuha Ali'
15 mov edx,8 ;длина сообщения
16 mov ecx,name ;сообщение для записи
17 mov ebx,1 ;файловый дескриптор (stdout)
18 mov eax,4 ;номер системного вызова (sys_write)
19 int 0x80 ;вызов ядра
20
21 mov eax,1 ;номер системного вызова (sys_exit)
22 int 0x80 ;вызов ядра
23
24 section .data
25 name db 'Zara Ali '
```

Переменные

Существует пять основных форм директивы определения:

Директива	Цель	Размер хранения
DB	Определить Byte	выделяет 1 байт
DW	Определить Word	выделяет 2 байта
DD	Определить Doubleword	выделяет 4 байта
DQ	Определить Quadword	выделяет 8 байт
DT	Определить Ten Bytes	выделяет 10 байт

Синтаксис для оператора распределения памяти для инициализированных данных

[имя-переменной] директива-определения начальное-значение [, начальное-значение]...

Примеры

Пример 1

```
1 choice      DB 'y'
2 number      DW 12345
3 neg_number  DW -12345
4 big_number  DQ 123456789
5 real_number1 DD 1.234
6 real_number2 DQ 123.456
```

Пример 2

```
1 section .text
2     global _start                ;нужно указать для линкера (gcc)
3
4     _start:                      ;показывает линкеру точку входа
5         mov     edx,1            ;длина сообщения
6         mov     ecx,choice       ;сообщение для записи
7         mov     ebx,1            ;файловый дескриптор (stdout)
8         mov     eax,4            ;номер системного вызова (sys_write)
9         int     0x80            ;вызов ядра
10
11        mov     eax,1            ;номер системного вызова (sys_exit)
12        int     0x80            ;вызов ядра
13
14 section .data
15 choice DB 'y'
```

Выделение дискового пространства для неинициализированных данных

Существует пять основных форм директив резервирования

Директива	Цель
RESB	Зарезервировать Byte
RESW	Зарезервировать Word
RESD	Зарезервировать Doubleword
RESQ	Зарезервировать Quadword
REST	Зарезервировать 10 байт

Множественность определений

```
1 | choice    DB    'Y'                ;ASCII of y = 79H  
2 | number1   DW    12345              ;12345D = 3039H  
3 | number2   DD    12345679          ;123456789D = 75BCD15H
```

1

Множественность инициализаций

```
1 | marks    TIMES 9 DW 0
```

2

СПАСИБО ЗА ВНИМАНИЕ!