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.

```
section .text
   global _start     ;нужно указать для линкера (gcc)

_start:              ;говорит линкеру о точке входа
   mov  edx,len      ;длина сообщения
   mov  ecx,msg      ;сообщение для записи
   mov  ebx,1        ;файловый дескриптор (stdout)
   mov  eax,4        ;номер системного вызова (sys_write)
   int  0x80         ;вызов ядра

   mov  edx,9        ;длина сообщения
   mov  ecx,s2       ;сообщение для записи
   mov  ebx,1        ;файловый дескриптор (stdout)
   mov  eax,4        ;номер системного вызова (sys_write)
   int  0x80         ;вызов ядра

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

section .data
msg db 'Сейчас покажем 9 звёзд',0xa ;сообщение
len equ $ - msg  ;длина сообщения
s2 times 9 db '*'
```

```
Сейчас покажем 9 звёзд
*********
```

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

```
section .data                           ;Сегмент Data
   userMsg db 'Пожалуйста, введите любые цифры: ' ;Просим пользователя ввести число
   lenUserMsg equ $-userMsg             ;Длина сообщения
   dispMsg db 'Вы ввели: '
   lenDispMsg equ $-dispMsg

section .bss            ;Неинициализированные данные
   num resb 5

section .text           ;Сегмент Code
   global _start

_start:                 ;Запрос пользователю на ввод
   mov eax, 4
   mov ebx, 1
   mov ecx, userMsg
   mov edx, lenUserMsg
   int 80h

   ;Считываем и сохраняем пользовательский ввод
   mov eax, 3
   mov ebx, 2
   mov ecx, num
   mov edx, 5           ;5 байт (числовой, 1 для знака) этой информации
   int 80h

   ;Вывод сообщения 'Вы ввели: '
   mov eax, 4
   mov ebx, 1
   mov ecx, dispMsg
   mov edx, lenDispMsg
   int 80h

   ;Вывод введёного числа
   mov eax, 4
   mov ebx, 1
   mov ecx, num
   mov edx, 5
   int 80h

   ; Код выхода
   mov eax, 1
   mov ebx, 0
   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:

```
section .text
global _start    ;нужно объявить для линкера
_start:          ;указывает компоновщику точку входа

   ;пишем имя 'Zara Ali'
   mov  edx,9        ;длина сообщения
   mov  ecx, name    ;сообщение для записи
   mov  ebx,1        ;файловый дескриптор (stdout)
   mov  eax,4        ;номер системного вызова (sys_write)
   int  0x80         ;вызов ядра

   mov  [name],  dword 'Nuha'    ; изменение имени Nuha Ali

   ;запись имени 'Nuha Ali'
   mov  edx,8        ;длина сообщения
   mov  ecx,name     ;сообщение для записи
   mov  ebx,1        ;файловый дескриптор (stdout)
   mov  eax,4        ;номер системного вызова (sys_write)
   int  0x80         ;вызов ядра

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

section .data
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

```
choice      DB  'y'
number      DW  12345
neg_number  DW  -12345
big_number  DQ  123456789
real_number1    DD  1.234
real_number2    DQ  123.456
```

Пример 2

```
section .text
   global _start          ;нужно указать для линкера (gcc)

_start:                   ;показывает линкеру точку входоа
   mov  edx,1           ;длина сообщения
   mov  ecx,choice         ;сообщение для записи
   mov  ebx,1           ;файловый дескриптор (stdout)
   mov  eax,4           ;номер системного вызова (sys_write)
   int  0x80            ;вызов ядра

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

section .data
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

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

1

Множественность инициализаций

```
marks   TIMES   9   DW   0
```

2

СПАСИБО ЗА ВНИМАНИЕ!