Tutor

Tutor for Linux

Windows : Тутор для виндовс пользователей

Пошаговое объяснение программы:

1. Макрос `putchar`

Этот макрос выводит символ на экран с использованием системного вызова **sys_write** в Linux (системный вызов 4). Аргументом макроса является регистр, содержащий символ для вывода. Макрос выполняет следующие шаги:

- Сохраняет регистры `eax`, `ebx`, `ecx`, и `edx` на стеке, чтобы сохранить их значения.
- Готовит параметры для системного вызова `sys_write`, который выводит символ на стандартный вывод (stdout):
  - **`eax = 4`** — это номер системного вызова **write**.
  - **`ebx = 1`** — это дескриптор файла (1 соответствует **stdout**).
  - **`ecx`** указывает на символ, который будет выведен, хранящийся на стеке.
  - **`edx = 1`** — количество символов для вывода (1 байт).
- После системного вызова, восстанавливает регистры из стека.

2. Сегмент данных (`section .data`)

- **`B dw 1, 2, 0, -5, 12, -45, 123, -1234, 30001`** — это массив с 9-ю целыми числами (16-битными), определенными с помощью директивы `dw` (define word).
- **`A dw 0 dup(9)`** — это массив, в который будет копироваться содержимое массива `B`. Он инициализирован нулями и содержит 9 элементов.

3. Сегмент кода (`section .text`)

Код программы начинается с метки `_start`, которая определяет точку входа программы.

a. Копирование массива B в массив A

```
mov esi, B            ; загружаем адрес массива B в регистр ESI
mov edi, A            ; загружаем адрес массива A в регистр EDI
mov ecx, 9            ; количество элементов массива
```

Цикл копирования:

```
copy_loop:
        mov ax, [esi]      ; загружаем элемент из массива B в регистр AX
        mov [edi], ax      ; копируем значение из AX в массив A
        add esi, 2         ; перемещаем указатель ESI к следующему элементу (размер слова - 2 байта)
        add edi, 2         ; перемещаем указатель EDI к следующему элементу
        dec ecx            ; уменьшаем счетчик элементов
jnz copy_loop          ; повторяем цикл, пока не скопируем все 9 элементов
```

b. Вывод содержимого массива A

После копирования начинается вывод массива A на экран.

```
mov esi, A            ; устанавливаем указатель на массив A
mov ecx, 9            ; размер массива (количество элементов)
```

Цикл обработки элементов массива:

```
array_loop:
    mov eax, [esi]    ; загружаем элемент из массива A в EAX
    test ax, ax       ; проверяем, является ли число отрицательным
    jns if_unsigned   ; если положительное (флаг знака не установлен), переходим к if_unsigned
        mov bx, '-'   ; если число отрицательное, выводим знак минус
        putchar bx    ; выводим символ '-'
        neg ax        ; инвертируем знак числа
```

c. Преобразование числа в ASCII и вывод

Каждое число преобразуется в строку символов (цифры) с помощью цикла деления на 10:

```
if_unsigned:
    mov bx, 10        ; делим на 10 для получения цифр
    xor di, di        ; DI — счетчик цифр

div_loop:
    xor dx, dx        ; очищаем старшую часть для деления 32-разрядного числа
    div bx            ; делим AX на 10 (остаток в DX, результат в AX)
    push dx           ; сохраняем цифру (остаток от деления) в стек
    inc di            ; увеличиваем счетчик цифр
    cmp ax, 0         ; если AX не равен 0, продолжаем делить
jnz div_loop
```

d. Вывод цифр

Цифры выводятся на экран в обратном порядке с использованием цикла:

```
print_digits_loop:
    pop ax            ; извлекаем цифру из стека
    add ax, '0'       ; преобразуем цифру в ASCII
    putchar ax        ; выводим цифру
    dec di            ; уменьшаем счетчик
jnz print_digits_loop ; повторяем, пока все цифры не будут выведены
```

e. Вывод пробела

После вывода числа выводится пробел:

```
mov bx, ' '           ; вывод пробела
putchar bx
```

f. Переход к следующему элементу массива

После вывода одного элемента массива, программа переходит к следующему:

```
add esi, 2            ; переходим к следующему элементу массива (на 2 байта вперед)
dec ecx               ; уменьшаем счетчик
```

```
jnz array_loop        ; повторяем цикл для каждого элемента массива
```

g. Вывод новой строки

После вывода всех элементов массива программа выводит символ новой строки:

```
mov bx, 10
putchar bx
```

h. Завершение программы

Программа завершает работу с использованием системного вызова `exit`:

```
mov eax, 1            ; системный вызов для завершения программы
mov ebx, 0            ; код возврата 0 (успешное завершение)
int 80h               ; вызов системного прерывания для завершения программы
```

Заключение:

Программа выполняет следующие действия:

1. Копирует массив целых чисел из одной области памяти в другую.
2. Преобразует каждое число в строку и выводит его на экран.
3. Если число отрицательное, сначала выводится знак минус.
4. Программа выводит числа по одному с пробелами между ними и завершает свою работу.