Языки программирования (Asm)

Занятие №8. Препроцессор. Процедуры.

МИРЭА - РТУ, БК №252

Москва 2020

Однострочные макросы

Однострочные макросы в NASM выглядят так

```
1 %define mask 0xFF &
2 %define func(a,b) ((a*a)+(a*b))
```

По принципу своей работы они крайне похожи на работу директивы препроцессора #define из языка C
Строчка кода

```
1 mov word[x], mask 0xFF31
```

будет преобразована в

```
1 mov word[x], 0x31
```

Операция побитового логического И с маской 0b11111111 будет выполнена на этапе препроцессорной обработки.
Для отмены макроопределения используется команда %undef

```
1 %undef mask
```

Многострочные макросы

Многострочные макросы лучше выносить в отдельный файл. Например, назовем файл mac.inc

```
%macro print 1
  mov eax, 4
  mov ebx, 1
  mov ecx, %1
  mov edx, 1
  int 0x80
%endmacro
```

После имени макроса идет число параметров. В данном случае, параметр единственный. Обращение к нему происходит через символ процент % и номер. Например, так будет выглядеть макрос с 2 параметрами

```
%macro push2 2
  push %1
  push %2
%endmacro
```

Использование многострочных макросов

В основном коде макрос будет использоваться так

```
1  %include "mac.inc"
2
3  section .bss
4    char resb 1
5
6  section .text
7    global _start
8
9  _start:
10   mov byte[char], 'N'
11
12   print char
13
14   mov eax,1
15   mov ebx,0
16   int 0x80
```

Условия в макросах

В макросах также можно использовать условия. Важно помнить, что все сравниваемые данные должны быть известны на момент сборки программы. %0 хранит число переданных параметров.

```
%macro test_numbers 3
  %if %0 == 3 ; Количество параметров
    mov eax, 4
    mov ebx, 1
    mov ecx, msg1
    mov edx, len1
    int 0x80
  %endif
  %if %1 < 5 && %2 > 5 ; Сравнение значений параметров
    mov eax, 4
    mov ebx, 1
    mov ecx, msg2
    mov edx, len2
    int 0x80
  %endif
%endmacro
```

Препроцессорные циклы: %REP

Помимо циклов, которые реализуются командами процессора, есть возможность создавать куски повторяющихся команд средствами препроцессора

```
1  _start:
2    mov byte[x], 0x30
3    %rep 80
4    mov eax, 4
5    mov ebx, 1
6    mov ecx, x
7    mov edx, 1
8    int 0x80
9    mov eax, [x]
10   inc eax
11   mov [x], eax
12   %endrep
13
14   mov eax, 1
15   int 0x80
```

Внешние процедуры

Внешняя процедура strlen

```
1  global strlen
2
3  section .text
4    strlen:
5      push ebp
6      mov ebp, esp
7      xor eax,eax
8      mov esi, [ebp+8]
9    .lp:
10     cmp byte[esi], 0
11     jz .quit
12     inc esi
13     inc eax
14     jmp short .lp
15   .quit:
16     pop ebp
17     ret
```

Использование процедуры и макроса print

```
%include "mac.inc"
extern strlen

section .data
hello db "Hello", 0xA

section .bss
  char resb 1

section .text
  global _start

_start:
  mov esi, hello
  push esi
  call strlen
  add eax, 0x30
  mov [char], eax
  print char
  mov eax,1
  mov ebx,0
```

Сборка

Для сборки программы из нескольких модулей нам необходимо получить объектные модули для дальнейшей компоновки. Вызываем nasm для каждого модуля, после чего выполняем компоновку ("линковку")

```
1 nasm -felf32 strlen.asm
2 nasm -felf32 main.asm
3 ld -m elf_i386 strlen.o main.o -o test
```

Компоновщик собирает из нескольких объектных модулей с их относительными адресами один исполняемый файл, в котором все адреса выставлены заново, и модули могут взаимодействовать друг с другом.