**Приложение к лабораторной работе №2**

1. Псевдо-инструкции NASM

**DB, DW, DD, DQ**: используются для объявления **инициализированных** данных в выходном файле.

Примеры использования:

db 0x55 ; байт со значением 0x55

db 0x55, 0x56, 0x57 ; 3 последовательно записанных байта

db ‘a’, 0x55 ; код символа и байт 0x55

db ‘hello’, 0 ; нуль-терминированная строка

dw 0xAA55 ; слово 0xAA55

dw ‘a’ ; 0x41 (‘a’) 0x00

dw ‘ab’ ; 0x41 (‘a’) 0x42 (‘b’)

dw ‘abc’ ; 0x41 0x42 0x43 0x00

и т. п.

**RESB, RESW, RESD, RESQ**: разработаны для использования в BSS-секции модуля, они объявляют **неинициализированное** пространство для хранения данных. Каждая из них принимает один операнд, который является числом резервируемых байт, слов, двойных слов, учетверенных слов соответственно.

NASM не поддерживает синтаксис резервирования неинициализированного пространства, реализованный в MASM/TASM, где можно было писать **dw ?** и тому подобное.

Примеры использования:

mybuf: resb 64 ; зарезервировать 64 байта

myword: resw 1 ; зарезервировать 1 слово

qarray: resq 10 ; 10 учетверенных слов

**EQU**: определяет символ для указанного **константного выражения**.  
Смысл EQU — связать имя метки со значением ее операнда.

Пример использования

hello\_msg: db ‘Hello, World’, 0

hello\_len equ $-hello\_msg ; обратите внимание, нет ‘:’ после имени метки

**TIMES**: обозначает, что следующая инструкция должна быть записана несколько раз. TIMES является эквивалентом псевдо-инструкции DUP в MASM/TASM.

Пример использования:

buf: **times 64** db 0

Аргумент TIMES является **числовым выражением**, что позволяет писать таким образом:

; дополнить размер strbuf до 64 байт

strbuf: db ‘Hello, World’, 0  
 times **64 - $ + strbuf** db 0

**TIMES нельзя применять в макросах, т. к. обрабатывается после макро-фазы. Для повторения кода в макросах используйте директиву препроцессора %rep.**

*Примечание: псевдо-инструкции RESB(W/D/Q), EQU и TIMES являются критическими выражениями NASM. Это значит, что их аргументами могут быть только выражения, которые зависят от ранее определенных символов.   
Например, такой код написан с ошибкой:*

*mov ax, symbol*

*const\_sym equ symbol*

*symbol: ; …*

*Инструкция mov ax, symbol является правильной, в то время как   
const\_sym equ symbol является ошибкой.*

1. Вычисление эффективного адреса

В NASM эффективные адреса имеют простой синтаксис:

myvar: dw 123  
 mov ax, [myvar] ; занести в ax значение 123  
 mov bx, [myvar+1]  
 mov dx, [es:myvar+bx]  
 mov eax, [ebx\*2+ecx+offset]

1. Выражения $ и $$

**$** представляет собой позицию начала (адрес) текущей строки. Например, можно написать подобный бесконечный цикл: *JMP $*

**$$** является меткой начала текущей секции кода. Ее можно использовать, чтобы, например, узнать, как далеко текущая инструкция находится от начала секции: *($ - $$)*.

1. Директивы NASM

Директива **SECTION** (или **SEGMENT** — синоним для SECTION) изменяет текущую секцию кода.

Пример определения новой секции:

section .bss ; секция неинициализированных данных

Можно также задавать начальный адрес секции:

BOOT\_ADDR equ 0x7C00  
 BOOT\_SIZE equ 512  
SIGNATURE\_ADDR equ BOOT\_ADDR + BOOT\_SIZE - 2

section signature start=SIGNATURE\_ADDR  
 dw 0xAA55

Однако в таком случае следует отслеживать, чтобы секции не пересекались.

Директива **BITS** (синоним для **USE16, USE32, USE64>**): определяет для какого режима процессора написан следующий после нее код.   
Синтаксис: BITS <16|32|64>.

1. Макросы NASM

**STRUC, ENDSTRUC**: объявление структур.  
Пример объявления структуры:

struc mytype  
mt\_long resd 1  
mt\_word resw 1  
mt\_byte resb 1  
mt\_buf resb 32  
 endstruc

Поля структур должны иметь уникальные имена среди всех структур.

**ISTRUC, AT, IEND**: создание экземпляров структур.

my\_instance\_addr: istruc mytype

at mt\_long, dd 101010

at mt\_word, dw 1024

at mt\_byte, db 0xAA

at mt\_buf, db ‘Buffer contents’, 0

iend

**%macro, %endmacro**: создание нового макроса

%macro print 2 ; 2 – количество аргументов  
 push eax  
 push esi

mov esi, %1 ; поместить первый аргумент в ESI  
 mov ah, %2  
%%print\_loop: ; о “%%” далее

lodsb  
 or al, al  
 jz %%print\_ret  
 stosw  
 jmp %%print\_loop

%%print\_ret:  
 pop esi  
 pop eax

%endmacro  
  
Метки в макросах являются глобальными. Чтобы сделать метки локальными, нужно дописать к началу имени метки “%%”.

**%rep, %endrep**Используется для повторения кода.

С помощью %rep можно создать сразу несколько экземпляров структур.  
Например,

%rep 32 ; 32 – количество повторений

istruc idt\_entry  
 at idt\_offset\_l, dw (dummy\_isr – protmode\_start)  
 at idt\_selector, dw CODE\_32\_SEGMENT  
 at idt\_reserved, db 0  
 at idt\_attr, db 0x8E  
 at idt\_offset\_h, dw 0  
 iend

%endrep