# Использование Ассемблера в ЯВУ

## Соглашение о регистрах

Регистры, используемые в Turbo Pascal:

SS – сегмент стека

DS – сегмент данных

BP – для указания на список параметров, передаваемых через стек

SP – указатель стека (после завершения подпрограммы стек должен быть восстановлен до своего первоначального состояния на момент вызова подпрограммы)

# Ассемблер Intel 8086

## Соглашения о метках и типах

Все переменные и функции в ассемблерном модуле доступны по имени.
Метки Паскаля не являются доступными из ассемблерного модуля.
Подпрограммы Паскаля доступны для ассемблерного модуля, если для них в *.asm используется директива **EXTRN.**

Правила оформления программы с модулями на Ассемблере:

1) в модуле *.asm: метки должны объявляться с директивой **PUBLIC** или **GLOBAL**;
2) в программе *.pas:
   соответствующие подпрограммы должны быть объявлены с директивой **external;**
   в программе *.pas должна быть директива: {$L asm.obj}

# Ассемблер Intel 8086

## Соглашения о параметрах

# Ассемблер Intel 8086

## Многомодульные программы

Реальный проект состоит из нескольких модулей. Для согласования модулей между собой используются следующие директивы:

1) PUBLIC – указывает на метки текущего модуля, к которым могут иметь доступ другие модули проекта:
   PUBLIC <метка> [, <метка>]
2) EXTRN – объявление меток из других модулей, которые необходимы для работы данного модуля:
   EXTRN <объявление> [, <объявление>]
   где <объявление> - запись вида <метка>:<тип>
3) GLOBAL – директива, которая интерпретируется как PUBLIC, если объект определён в данном модуле, и как EXTRN, если определение объекта в данном модуле отсутствует:
   GLOBAL <объявление> [, <объявление>]
   где <объявление> - запись вида <метка>:<тип>, по формату совпадающая с такой записью в директиве EXTRN
4) INCLUDE – включение содержимого указанного файла в текущий файл:
   INCLUDE <имя файла>

# Ассемблер Intel 8086

## Многомодульные программы

В качестве типа принимаемого объекта могут указываться следующие:

ABS – имя постоянной величины;

BYTE – имя переменной величины байтового типа (1 байт);

WORD – имя переменной величины типа WORD (2 байта);

DWORD – имя переменной величины типа DWORD (4 байта);

FWORD – имя переменной величины типа FWORD (6 байтов);

QWORD – имя переменной величины типа QWORD (8 байтов);

TWORD – имя переменной величины типа TWORD (10 байтов);

NEAR – имя ближней процедуры или команды;

FAR – имя дальней процедуры или команды.

# Ассемблер Intel 8086

## Многомодульные программы: пример

```
;первый модуль
.Data
PUBLIC MemVar, Array1, Array_Length
Array_Length EQU 100
MemVar DW 10
Array1 DB Array_Length DUP(?)
. . .
.Code
PUBLIC NearProc, FarProc
NearProc PROC Near
. . .
NearProc ENDP

FarProc PROC Far
. . .
FarProc ENDP
```

```
;второй модуль
.Data
EXTRN MemVar: WORD, Array1: BYTE,
      Array_Length: ABS
. . .
.Code
EXTRN NearProc: NEAR, FarProc: FAR
. . .
mov ax, [MemVar]
mov bx, OFFSET Array1
moc cx, Array_Length
. . .
call NearProc
. . .
call FarProc
```

# Ассемблер Intel 8086

## Сегментные директивы

Программа может быть написана с использованием определений каждого сегмента в явном виде. Для этого предусмотрены сегментные директивы:

1) SEGMENT – указывает начало и атрибуты каждого сегмента программы. Это структурная директива, имеющая следующий вид:
   label SEGMENT align combine class
   label ENDS
   где **label** – имя сегмента;
   **align** – тип выравнивания сегмента (BYTE, WORD, DWORD, PARA, PAGE);
   **combine** – способ объединения нескольких сегментов (PRIVATE, PUBLIC, COMMON, STACK, MEMORY, AT exp);
   **class** – имя класса, к которому будет отнесён данный сегмент (заключается в апострофы или кавычки).

# Ассемблер Intel 8086

## Сегментные директивы

Программа может быть написана с использованием определений каждого сегмента в явном виде. Для этого предусмотрены сегментные директивы:

2) GROUP – директива, предназначенная для объединения различных сегментов таким образом, чтобы была возможной адресация внутри этих сегментов с помощью одного сегментного регистра, т.е. объединённый сегмент будет занимать не более 64 Кбайт памяти. Директива имеет следующий вид:
   name GROUP <segname> [, <segname>]
   где **name** – имя объединённого сегмента;
   **segname** – имена сегментов, которые будут объединены.
3) ASSUME – описание назначения сегментных регистров. Вид директивы:
   ASSUME <reg>: <name>[, <reg>: <name>]
   ASSUME <reg>: NOTHING
   ASSUME NOTHING
   где **reg** – имя сегментного регистра;
   **name** – имя сегмента или группы сегментов.

# Ассемблер Intel 8086

## Сегментные директивы: пример

```
ASSUME CS: Code
Code SEGMENT
    Fix DB 25
Begin:
    mov ax, Code
    mov ds, ax
    mov al, [Fix]
    mov ah, 4ch
    int 21h
Code ENDS
Stack_seg SEGMENT STACK
    DB 100h DUP(?)
Stack_seg ENDS
END Begin
```

Обращение к переменной Fix требует воспользоваться номером сегмента, в котором объявлена метка Fix. Несмотря на то что во время выполнения в этот номер находится в регистре DS, во время компиляции этот факт не был известен (по умолчанию предполагается директива ASSUME DS: NOTHING), следовательно, была сгенерирована команда с префиксом: mov al, [cs:Fix].

Этого можно избежать, если написать директиву ASSUME CS: Code, DS:Code