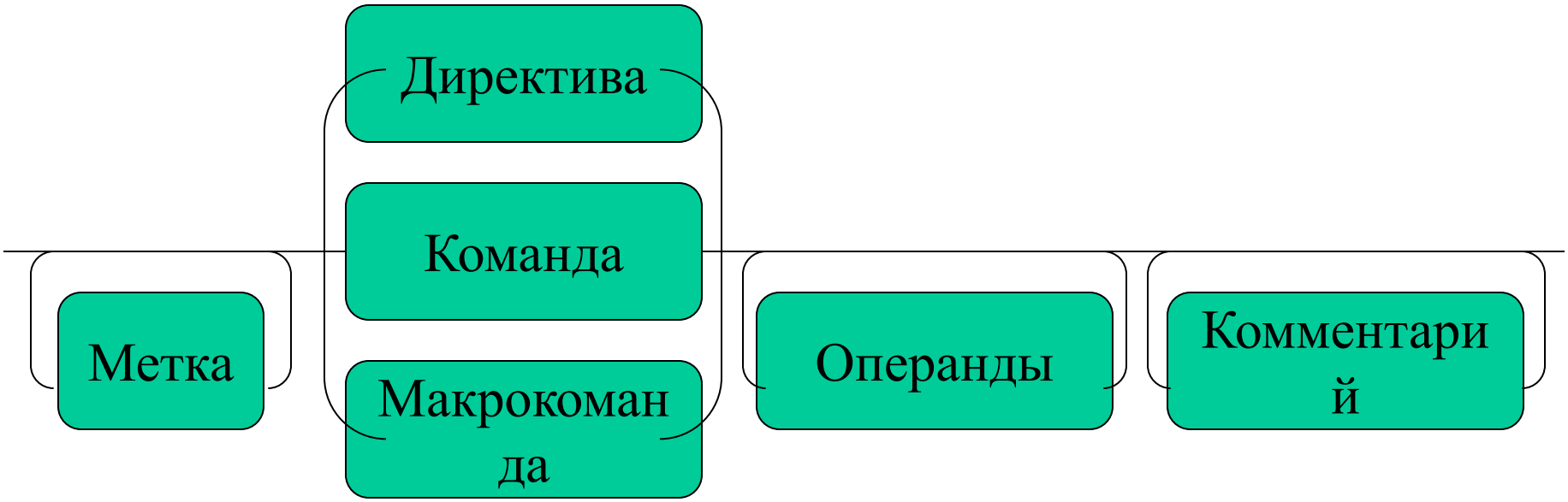
Введение в программирование на языке ассемблера

Формат инструкции на языке ассемблера



Директивы – команды управления работой транслятора

- Управление листингом

  PAGE К_стр К_симв

  9 < К_стр=66 < 256

  59 < К_симв=80 < 133

- Переход к новой странице

  PAGE

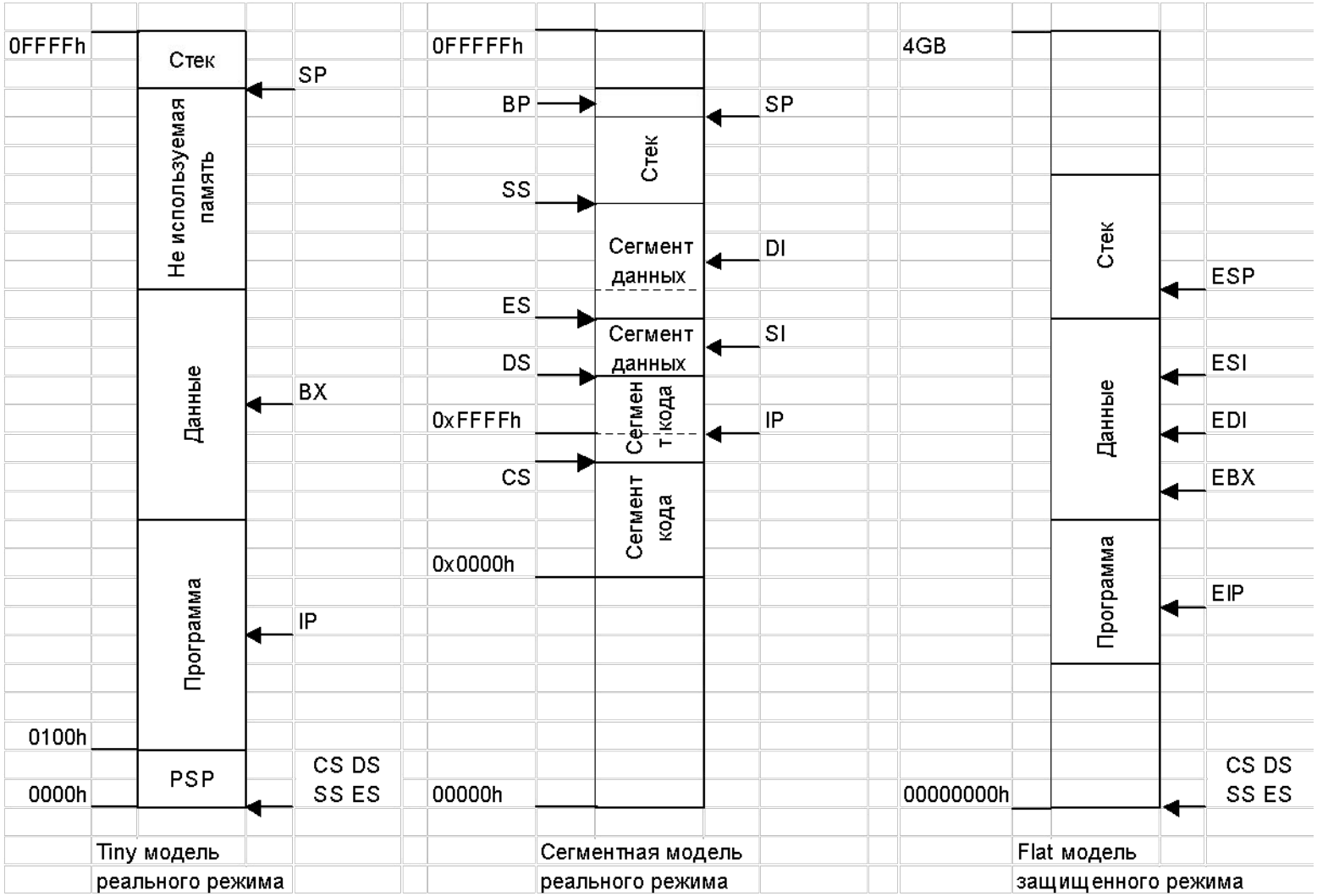
- Заголовок

  TITLE текст

- Подзаголовок

  SUBTTL текст

Три модели организации адресного пространства



Директива описания сегмента

Имя SEGMENT $[_1]$ $[_2]$ $[_3]$ $[_4]$

<инструкции языка>

Имя ENDS

Здесь $[_1]$ - тип выравнивания

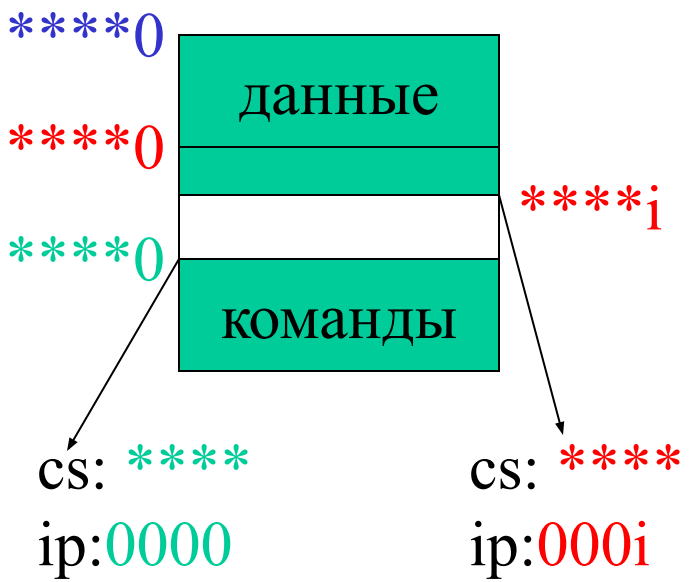
$[_2]$ - тип объединения

$[_3]$ - класс

$[_4]$ - размер адреса (для i386 и выше)

Имя – константа, содержащая номер параграфа начала сегмента

- Тип выравнивания
  - ✔ BYTE x 1
  - ✔ WORD x 2
  - ✔ DWORD x 4
  - ✔ PARA x 16
  - ✔ PAGE x 256
  - ✔ MEMPAGE x 1024



- Тип объединения
  - ✔ PRIVATE
  - ✔ PUBLIC (MEMORY)
  - ✔ COMMON
  - ✔ STACK
  - ✔ AT ****
- Размер адреса
  - ✔ USE16
  - ✔ USE32
- Класс

  ‘имя_класса’

Обозначения

- SR – сегментный регистр
- r8, r16, r32 – регистр общего назначения
- m8, m16,m32 – адрес области памяти
- i8,i16,i32 – непосредственное значение (константа)

Директива указания содержимого сегментных регистров

ASSUME SR:имя [,sr:имя]…

Имя – имя сегмента или NOTHING

Директива «Конец модуля»

END [точка входа в программу]

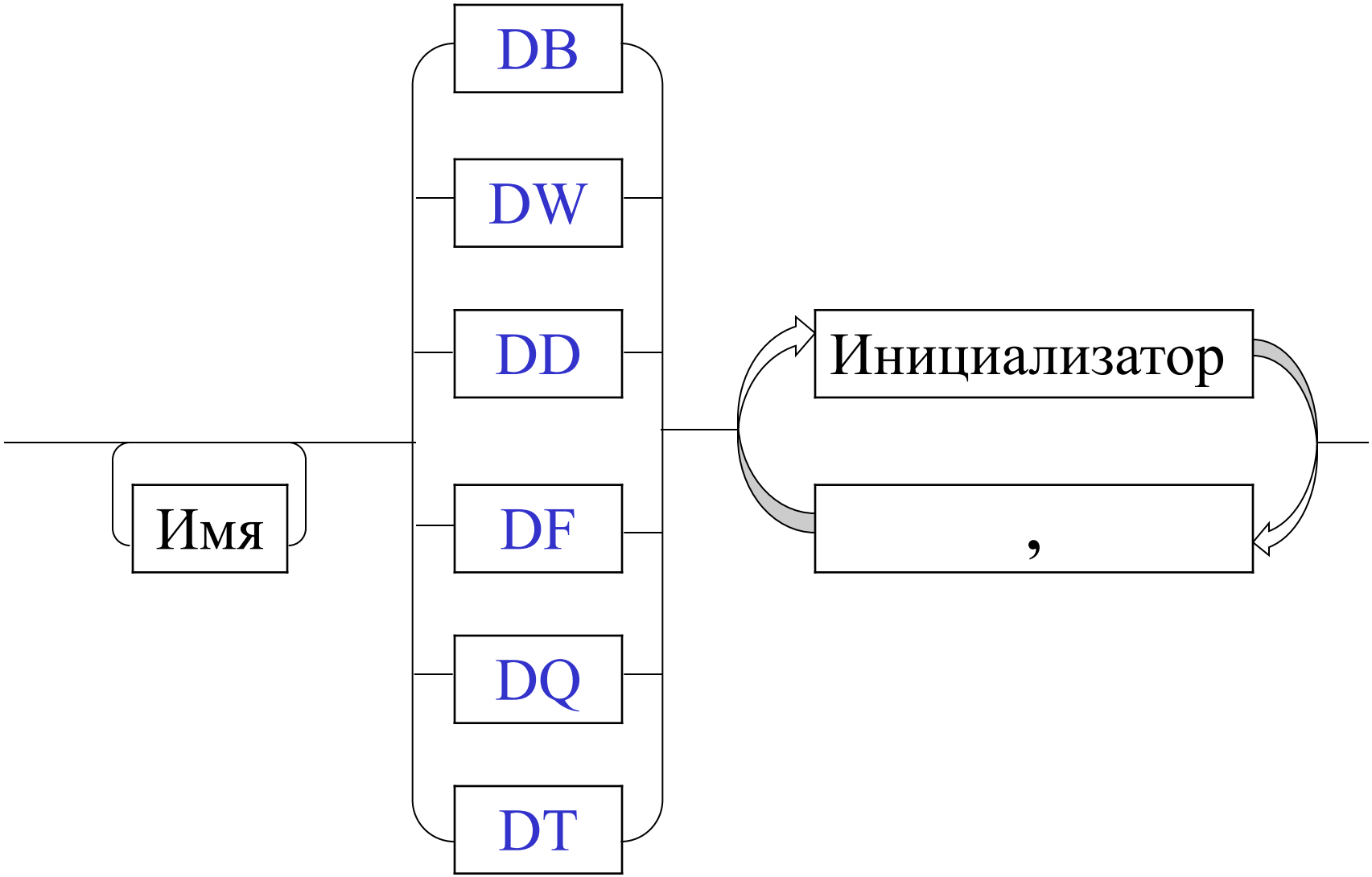
Директива описания процедуры

```
Имя     PROC   [FAR]
    Тело_процедуры
Имя     ENDP
```

Far – для входа в процедуру используется полный адрес (адресная пара CS:IP)

Иначе – только смещение (IP)

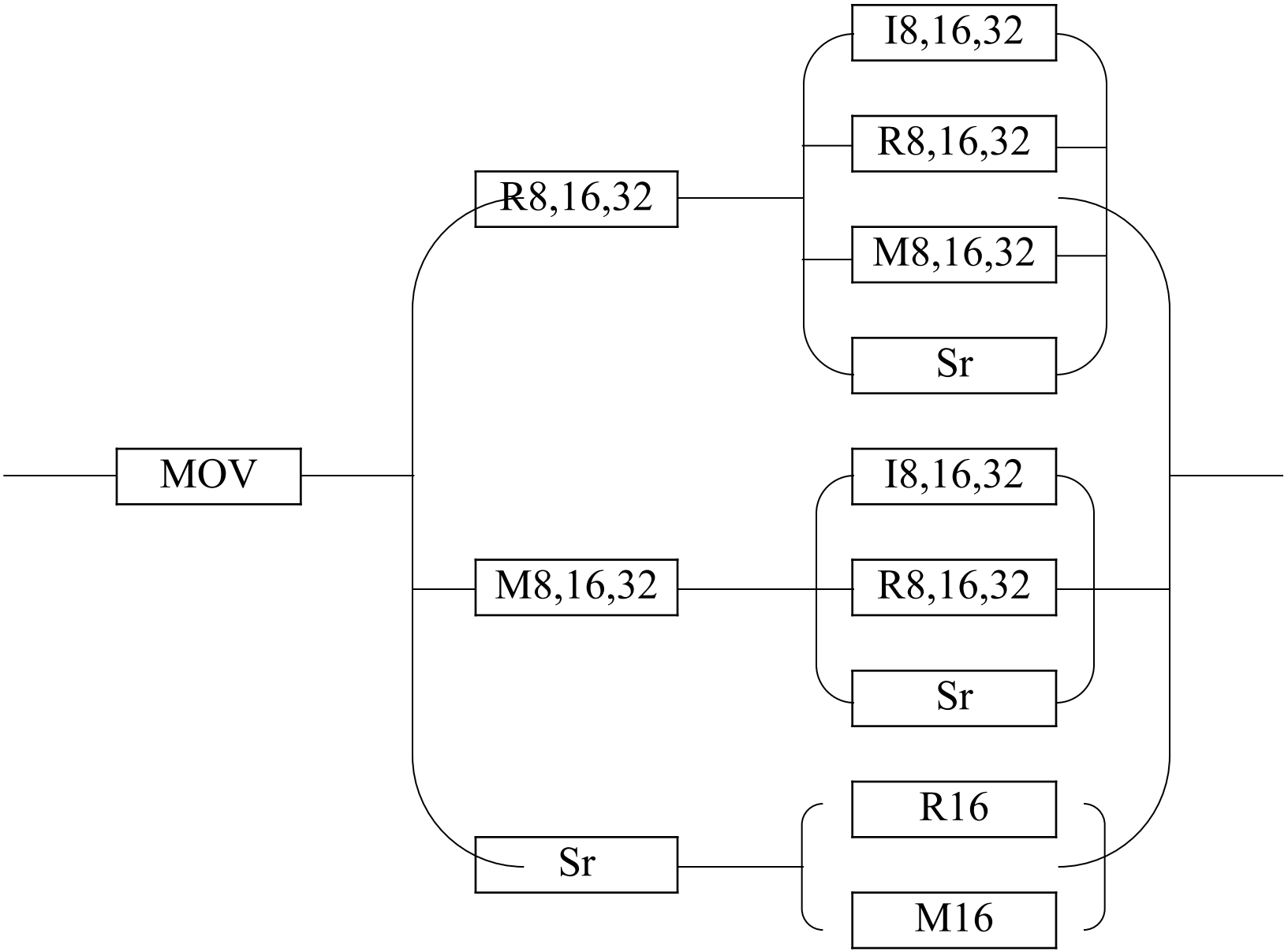
Директивы описания данных



Пример

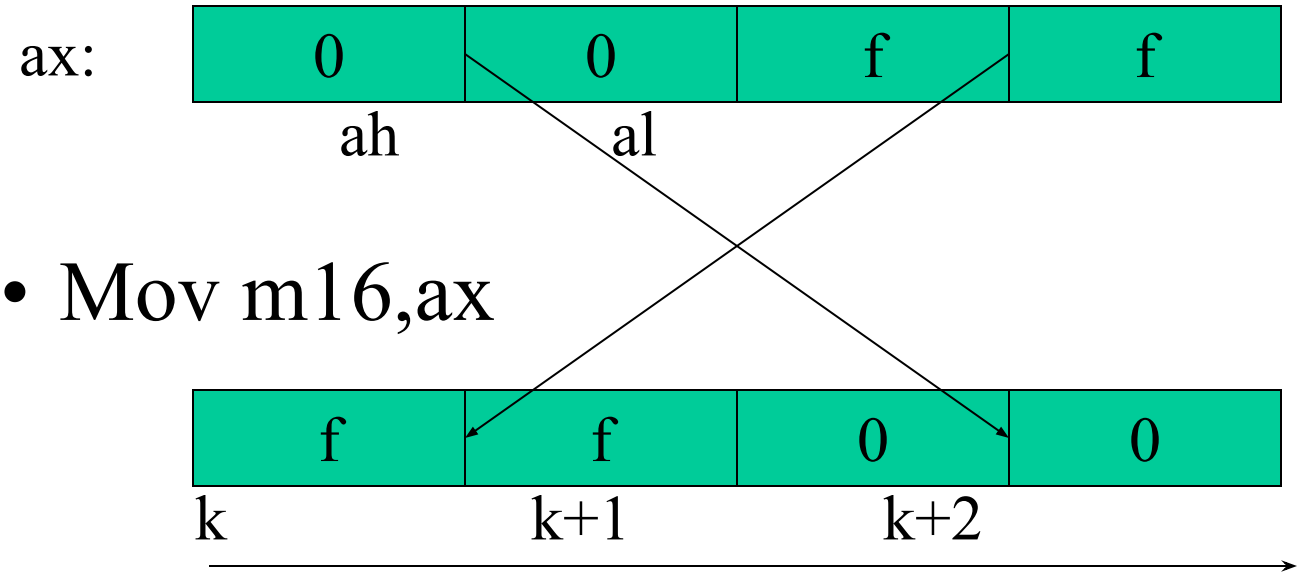
```
Data segment
A       db ?
B       db ‘abcd’,4 dup(‘*’)            ;”abcd****”
C       dw-1,0,1
D       dwC                             ;Offset C
E       dd D                            ;Seg D:Offset D
F       db 0fh,15,17q,1111b             ;4 dup(15)
G       dd -1.5
H       dq ‘hgfedcba’                   ;”abcdefgh”
Data    ends
```

Инструкция MOV



N.B.

- Mov ax,0ffh



- Mov m16,ax

Инструкция генерации программного прерывания int

- Вызов функции BIOS

  Int 10h

- Вызов функции DOS

  Int 21h

Номер функции задается в регистре ah

02h – установить курсор
dh – строка, dl – колонка

03h – получить позицию курсора на bh странице
dh – строка, dl - колонка

01h – ввод символа в al

02h – вывод символа из dl

09h – вывод строки, до ‘$’
адрес в dx

4ch – завершение программы с кодом ошибки al

Пример

```
        title       Первый файл
        subttl Основная программа и сегмент данных

Stkseg  segment  stack
        db      256 dup(?)
Stkseg  ends

Dtseg   segment  public
x       db      "abcdef"
Dtseg   ends
```

```
Cdseg segment
    assume   cs:cdseg,ds:dtseg,ss:stkseg,es:nothing
    main   proc    far
        mov    ax,dtseg     ; инициализация сегментного
        mov    ds,ax        ; регистра данных
        mov    ah,9         ; функция DOS вывод строки
        mov    dx,offset x ; адрес строки
        int  21h
        mov    ah,2         ; функция DOS вывод символа
        mov    dl,2ah       ; код символа *
        int  21h
        mov    ah,4ch       ; функция завершения задачи
        mov    al,0      ; код ошибки
        int  21h
    main   endp
Cdseg ends
         end   main
```

```
    title   Второй файл
    subttlЕще один сегмент данных
Dtseg    segmentbyte public
y            db '$'
Dtseg    ends
         end
```

Результат: abcdef*

```
    title   Второй файл
    subttlЕще один сегмент данных
Dtseg    segmentpublic
y            db '$'
Dtseg    ends
         end
```

Результат: abcdef *

tst.asm
Первый файл
Основная программа и сегмент данных

```
 1  0000                    stkseg  segment stack
 2  0000  0100*(??)                 db      256 dup(?)
 3  0100                    stkseg  ends
 4
 5  0000                    dtseg   segment public
 6  0000  61 62 63 64 65 66         x       db      "abcdef"
 7  0006                    dtseg   ends
 8
 9  0000                    cdseg   segment
10                      assume  cs:cdseg,ds:dtseg,ss:stkseg,es:nothing
11  0000                    main    proc    far
12  0000  B8 0000s                  mov     ax,dtseg
13  0003  8E D8                     mov     ds,ax
14
15  0005  B4 09                     mov     ah,9
16  0007  BA 0000r                  mov     dx,offset x
17  000A  CD 21                     int     21h
18  000C  B4 02                     mov     ah,2
19  000E  B2 2A                     mov     dl,2ah
20  0010  CD 21                     int     21h
21
22  0012  B4 4C                     mov     ah,4ch
23  0014  B0 00                     mov     al,0    ; код ошибки
24  0016  CD 21                     int     21h
25  0018                    main    endp
26  0018                cdseg   ends
27
28                          end     main
```

Symbol Table
Первый файл

Symbol Name	Type	Value
??DATE	Text	"22/02/04"
??FILENAME	Text	"tst "
??TIME	Text	"14:26:27"
??VERSION	Number	040A
@CPU	Text	0101H
@CURSEG	Text	CDSEG
@FILENAME	Text	TST
@WORDSIZE	Text	2
MAIN	Far	CDSEG:0000
X	Byte	DTSEG:0000

Groups & Segments	Bit	Size	Align	Combine	Class
CDSEG	16	0018	Para	none	
DTSEG	16	0006	Para	Public	
STKSEG	16	0100	Para	Stack	

```
Turbo Assembler    Version 4.1     22/02/04 14:26:27          Page 1
tst1.asm
Второй файл
Еще один сегмент данных
     1  0000                dtseg segment byte     public
     2  0000  24                    y       db     '$'
     3  0001                dtseg   ends
     4
     5                      end
Turbo Assembler    Version 4.1     22/02/04 14:26:27          Page 2
Symbol Table
Второй файл
Symbol Name                       Type  Value

??DATE                            Text  "22/02/04"
??FILENAME                        Text  "tst1    "
??TIME                            Text  "14:26:27"
??VERSION                         Number     040A
@CPU                              Text  0101H
@CURSEG                           Text  DTSEG
@FILENAME                         Text  TST1
@WORDSIZE                         Text  2
Y                              Byte  DTSEG:0000

Groups & Segments                 Bit Size Align  Combine Class
DTSEG                          16  0001 Byte  Public
```

Распределение памяти (.mem)

Start	Stop	Length	Name	Class
00000H	000FFH	00100H	STKSEG	
00100H	00106H	00007H	DTSEG	
00110H	00127H	00018H	CDSEG	

Program entry point at 0011:0000

Использование оператора mov

```
#include <iostream.h>
   int mema;
void main() {
   _asm mov mema,5; /*то же, что и  mema=5; */
   cout << “mema=“ << mema << endl;
   }
```

Результат: mema=5

Использование стека

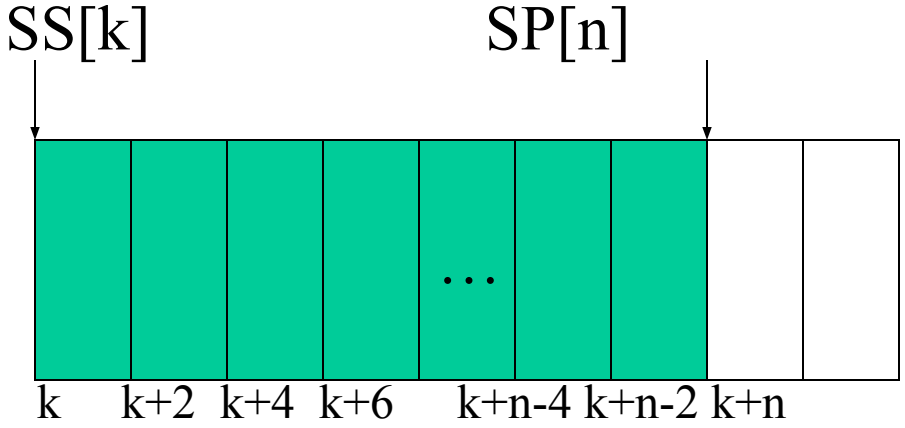
- Используется для:

  Хранения адреса возврата из вызванной программы

  Передачи параметров между программами

  Временного хранения данных

- Единица данных – слово
- Регистры, связанные со стеком: ss, sp, bp




```
PUSH  1
PUSH  2
POP   bx
POP   ax
NOP
```

Использование стека

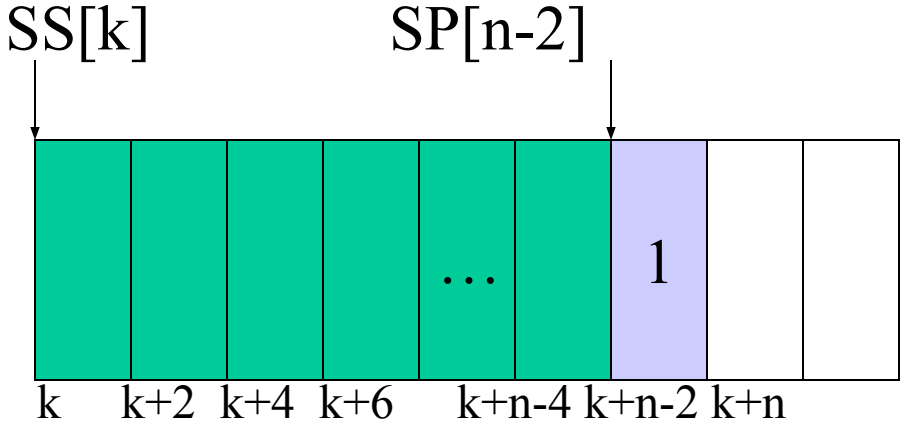
- Используется для:

  Хранения адреса возврата из вызванной программы

  Передачи параметров между программами

  Временного хранения данных

- Единица данных – слово
- Регистры, связанные со стеком: ss, sp, bp




```
PUSH  1
PUSH  2
POP   bx
POP   ax
NOP
```

Использование стека

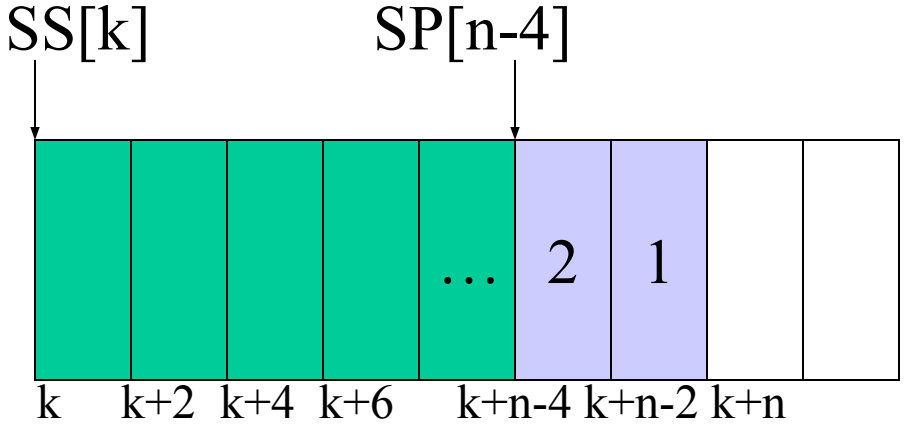
- Используется для:

  Хранения адреса возврата из вызванной программы

  Передачи параметров между программами

  Временного хранения данных

- Единица данных – слово
- Регистры, связанные со стеком: ss, sp, bp




```
PUSH  1
PUSH  2
POP   bx
POP   ax
NOP
```

Использование стека

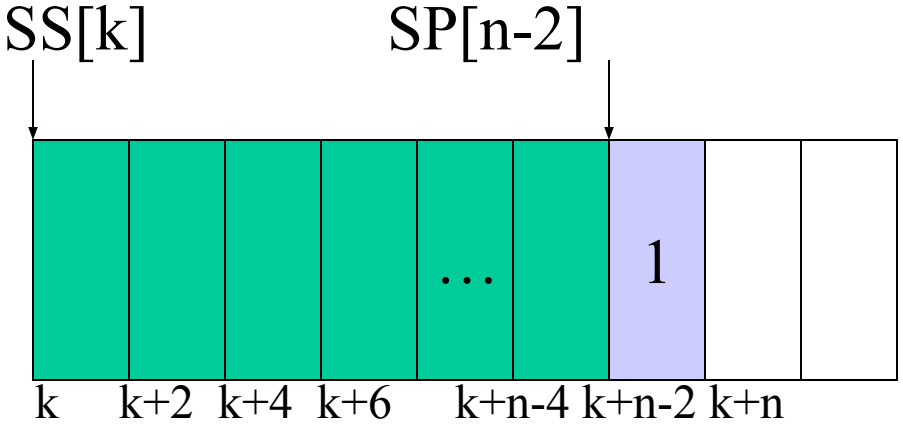
- Используется для:

  Хранения адреса возврата из вызванной программы

  Передачи параметров между программами

  Временного хранения данных

- Единица данных – слово
- Регистры, связанные со стеком: ss, sp, bp




```
PUSH  1
PUSH  2
POP   bx  ; =2
POP   ax
NOP
```

Использование стека

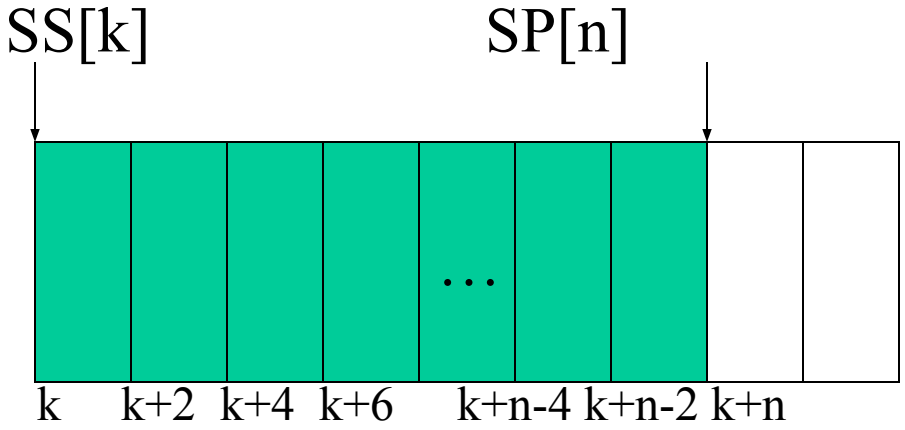
- Используется для:

  Хранения адреса возврата из вызванной программы

  Передачи параметров между программами

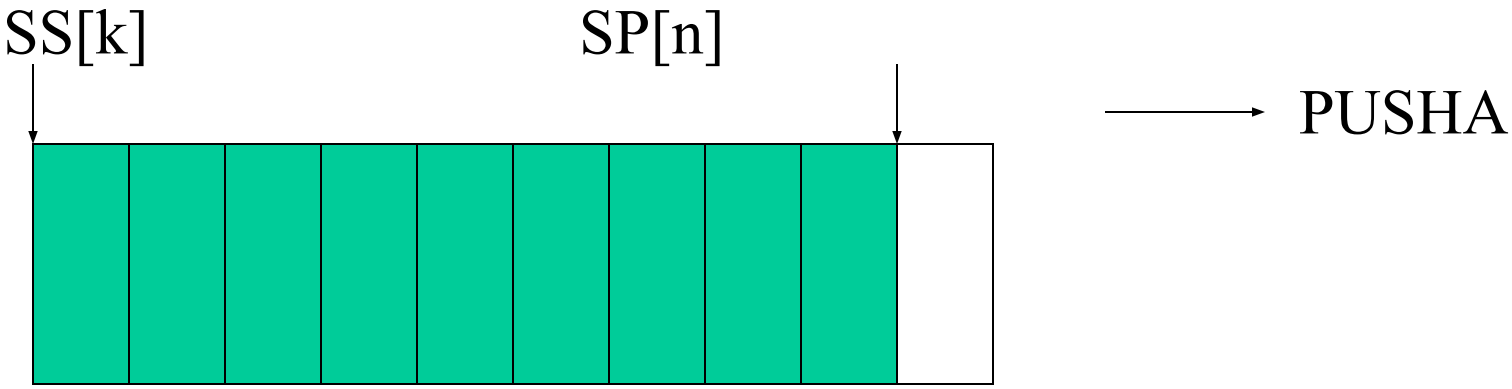
  Временного хранения данных

- Единица данных – слово
- Регистры, связанные со стеком: ss, sp, bp

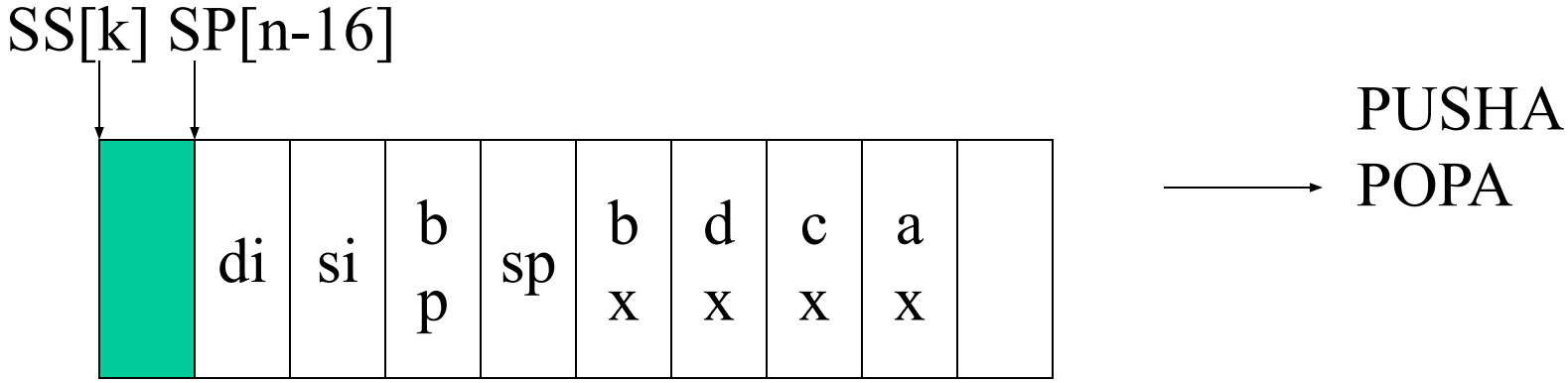



```
PUSH  1
PUSH  2
POP   bx  ; =2
POP   ax  ; =1
NOP
```

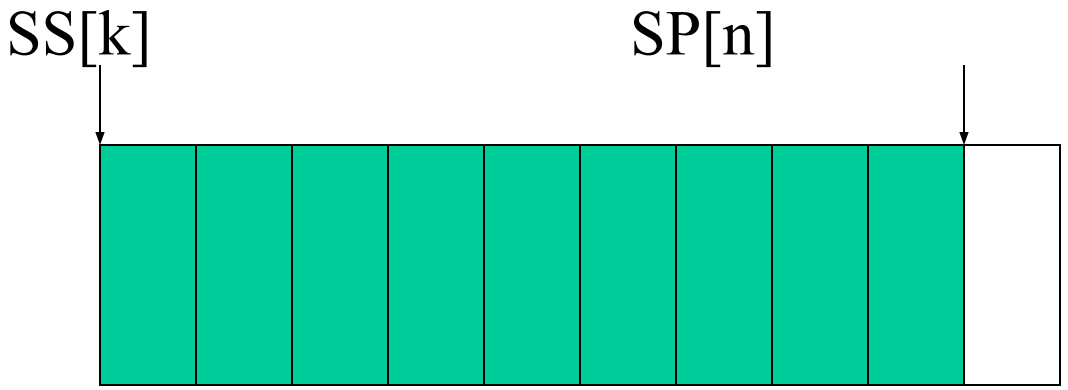
Команды работы со стеком



Команды работы со стеком

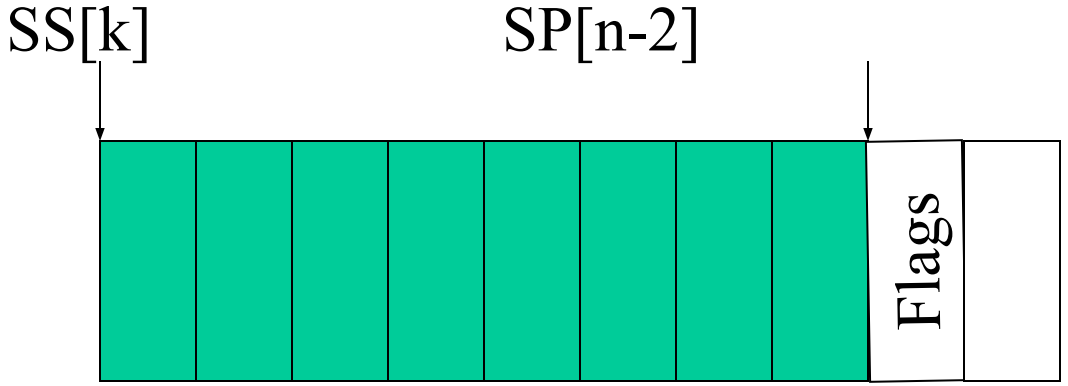


Команды работы со стеком



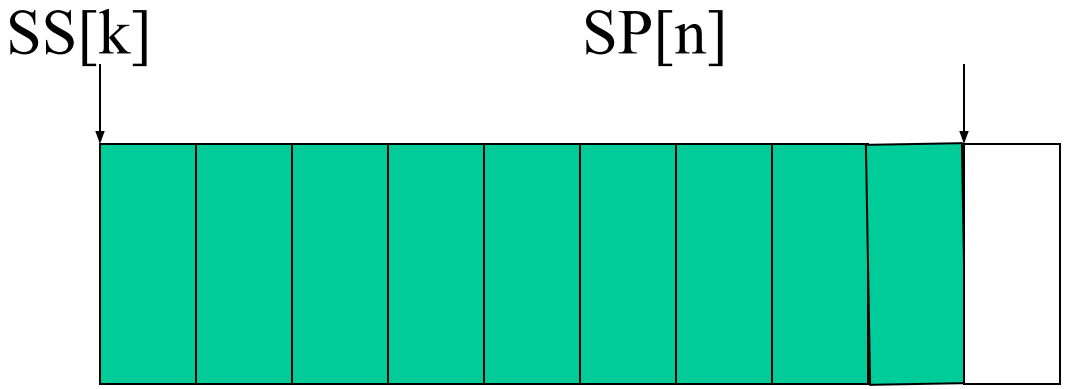
PUSHA
POPA
PUSHF

Команды работы со стеком



PUSHA
POPA
PUSHF
POPF

Команды работы со стеком



PUSHA
POPA
PUSHF
POPF
→ PUSHAD
POPAD