Системное программирование

Лекция №4

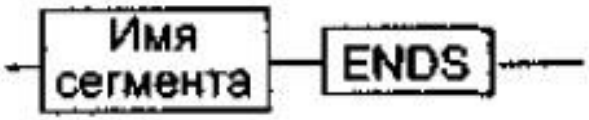
Основные директивы Ассемблера

Директивы сегментации

Процессор может одновременно работать:

- с одним сегментом кода;
- с одним сегментом стека;
- с одним сегментом данных;
- с тремя дополнительными сегментами данных.

Директивы сегментации



Синтаксис описания сегмента

Директивы сегментации

Атрибут выравнивания сегмента (тип выравнивания) сообщает компоновщику о том, что нужно обеспечить размещение начала сегмента на заданной границе.

Допустимые значения атрибута:

BYTE — выравнивание не выполняется. Сегмент может начинаться с любого адреса памяти;

WORD — сегмент начинается по адресу, кратному двум (выравнивание по границе слова);

DWORD — сегмент начинается по адресу, кратному четырем, (выравнивание по границе двойного слова);

PARA — сегмент начинается по адресу, кратному 16 (выравнивание по границе параграфа);

PAGE — сегмент начинается по адресу, кратному 256 (выравнивание по границе страницы размером 256 байт);

MEMPAGE — сегмент начинается по адресу, кратному 4 Кбайт (адрес следующей страницы памяти размером 4 Кбайт).

По умолчанию тип выравнивания имеет значение PARA

Директивы сегментации

Атрибут комбинирования сегментов (комбинаторный тип) сообщает компоновщику, как нужно комбинировать сегменты различных модулей, имеющие одно и то же имя.

Допустимые значения атрибута:

PRIVATE — сегмент не будет объединяться с другими сегментами с тем же именем вне данного модуля;

PUBLIC — заставляет компоновщик объединить все сегменты с одинаковым именем. Новый объединенный сегмент будет целым и непрерывным;

COMMON — располагает все сегменты с одним и тем же именем по одному адресу, то есть все сегменты с данным именем перекрываются;

AT xxxx — располагает сегмент по абсолютному адресу параграфа 0). Абсолютный адрес параграфа задается выражением xxxx;

STACK — определение *сегмента стека.* Заставляет компоновщик объединить все одноименные сегменты и вычислять адреса в этих сегментах относительно регистра SS.

По умолчанию тип выравнивания имеет значение PRIVATE

Директивы сегментации

Атрибут класса сегмента (тип класса) — это заключенная в кавычки строка, помогающая компоновщику определить нужный порядок следования сегментов при сборке программы из сегментов нескольких модулей. Компоновщик объединяет вместе в памяти все сегменты с одним и тем же именем класса.

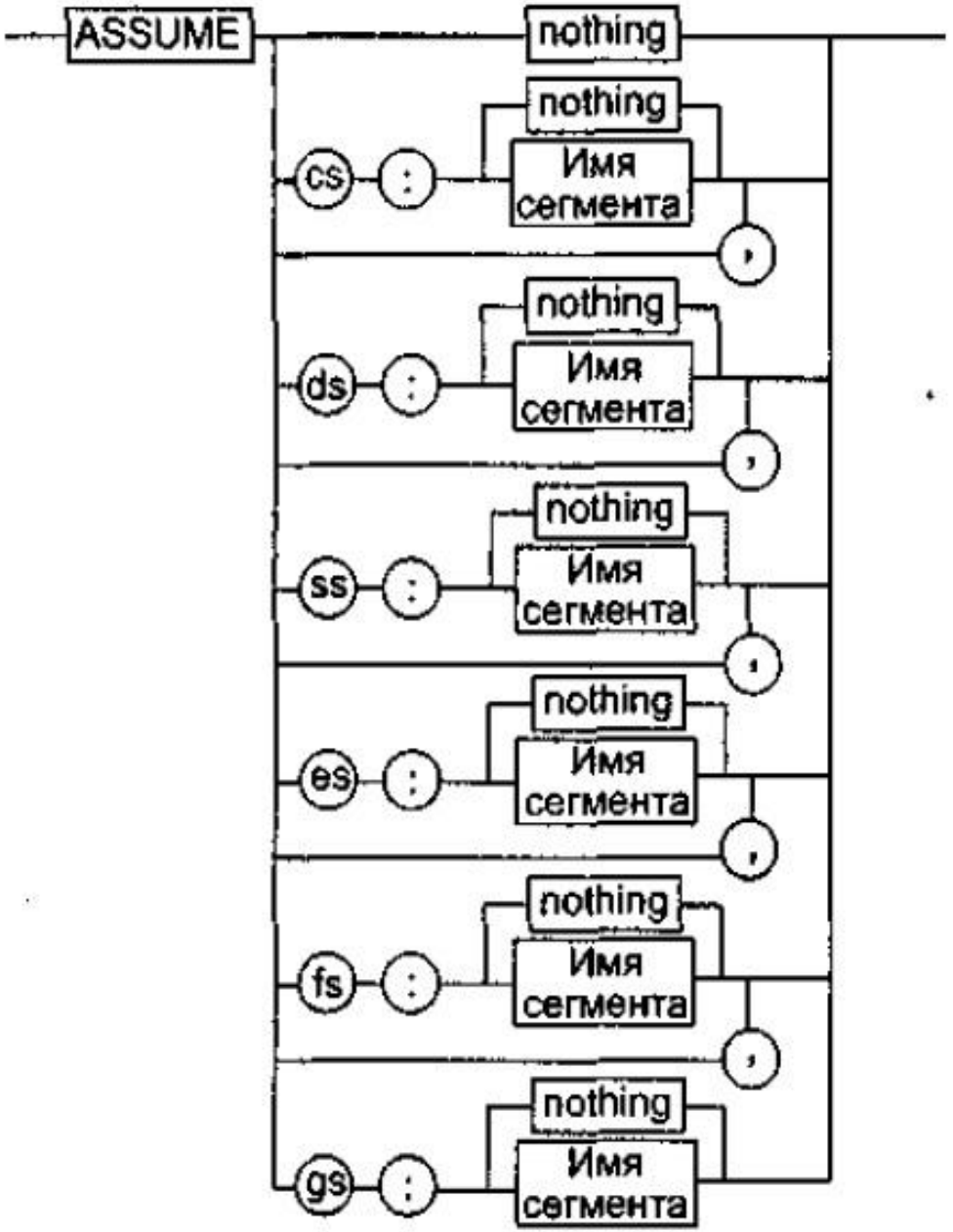
Атрибут размера сегмента. Для процессоров i80386 и выше сегменты могут быть 16- или 32-разрядными. Это влияет прежде всего на размер сегмента и порядок формирования физического адреса внутри него.

Возможные значения атрибута:

USE16 — сегмент допускает 16-разрядную адресацию. При формировании физического адреса может использоваться только 16-разрядное смещение;

USE32 — сегмент должен быть 32-разрядным. При формировании физического адреса может использоваться 32-разрядное смещение.

Директивы сегментации



Директива ASSUME

Директивы сегментации

Все сегменты сами по себе равноправны, так как директивы SEGMENT и ENDS не содержат информации о функциональном назначении сегментов. Для того чтобы использовать их как сегменты кода, данных или стека, необходимо предварительно сообщить транслятору об этом с помощью специальной директивы ASSUME. Привязка сегментов к сегментным регистрам осуществляется с помощью операндов этой директивы, в которых имя_сегмента должно быть именем сегмента, определенным в исходном тексте программы директивой SEGMENT или ключевым словом NOTHING. Если в качестве операнда используется только ключевое слово NOTHING, то предшествующие назначения сегментных регистров аннулируются, причем сразу для всех шести сегментных регистров.

Директивы сегментации

Листинг 1. Использование стандартных директив сегментации

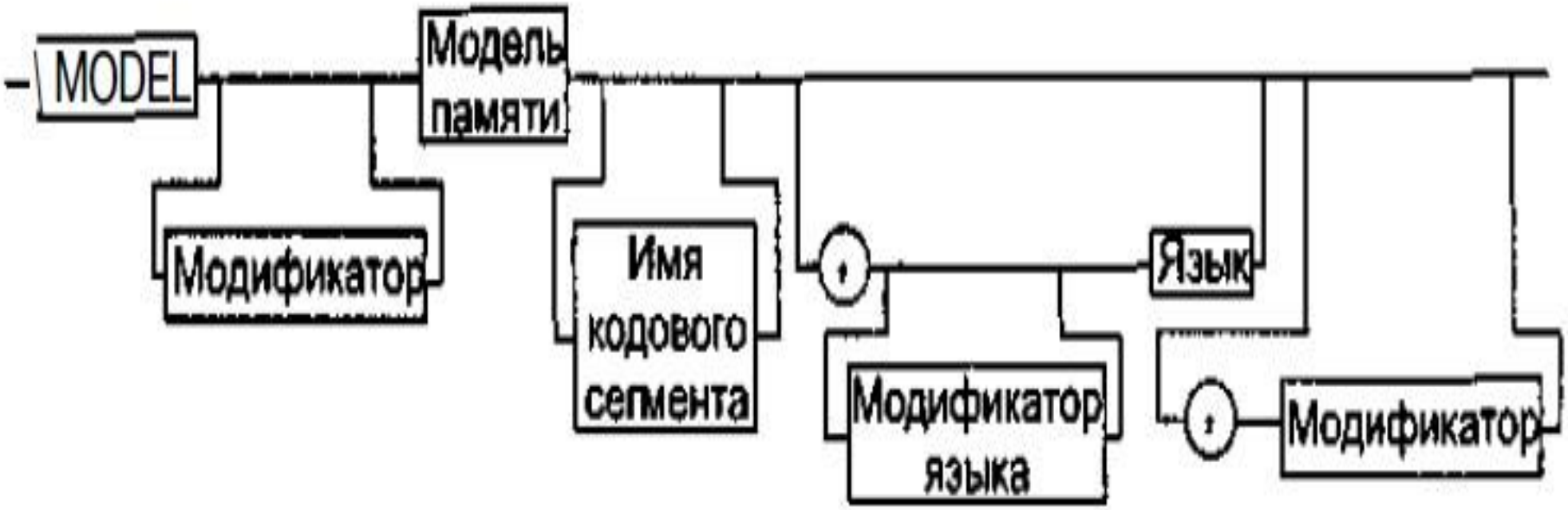
```
data    segment para public 'data'
message db  'Hello World! No war and bomb! Let us live friendly and learn
assembler language. $'
data    ends
stk segment stack
    db  256 dup ('?')                   ;сегмент стека
stk ends
code    segment para public 'code'  ; начало сегмента кода
main    proc                        ; начало процедуры main
    assume cs:code,ds:data,ss:stk
    mov ax,data                     ; адрес сегмента данных в регистр ax
    mov ds,ax                       ; ax в ds
    mov ah,9
    mov dx,offset message
    int 21h                         ; вывод сообщения на экран
    mov ax,4c00h                    ; пересылка 4c00h в регистр ax
    int 21h                         ; вызов прерывания с номером 21h
main    endp                        ; конец процедуры main
code    ends                        ; конец сегмента кода
end main                            ; конец программы с точкой входа main
```

Директивы сегментации

Листинг 2. Использование упрощенных директив сегментации

```
masm                    ; режим работы для TASM - masm, для MASM - не нужно
model   small           ; модель памяти
.data                   ; сегмент данных
message db  'Hello World! No war and bomb! Let us live friendly and learn
assembler language. $'
.stack  256h            ; сегмент стека
.code                   ; сегмент кода
main    proc            ; начало процедуры main
    mov ax,@data        ; заносим адрес сегмента данных в регистр ax
    mov ds,ax           ; ax в ds
    mov ah,9
    mov dx,offset message
    int 21h             ; вывод сообщения на экран
    mov ax,4c00h        ; пересылка 4c00h в регистр ax
    int 21h             ; вызов прерывания с номером 21h
main    endp            ; конец процедуры main
end main                ; конец программы с точкой входа main
```

Директивы сегментации



Синтаксис директивы MODEL

Директивы сегментации

Модели памяти

Модель	Количество и размер сегментов		Тип указателя	
	Code	Data	Code	Data
MS DOS, Win16				
Tiny	ОДИН, <=64Kb		Near	
Small	ОДИН, <=64Kb	ОДИН, <=64Kb	Near	Near
Medium	НЕСКОЛЬКО, <=64Kb	ОДИН, <=64Kb	Far	Near
Compact	ОДИН, <=64Kb	НЕСКОЛЬКО, <=64Kb	Near	Far
Large	НЕСКОЛЬКО, <=64Kb	НЕСКОЛЬКО, <=64Kb	Far	Far
Huge	НЕСКОЛЬКО, >64Kb	НЕСКОЛЬКО, >64Kb	Huge	Huge
Win32 (начиная с i386)				
Flat	Ограничено возможностями процессора и материнской платы	Ограничено возможностями процессора и материнской платы	Flat	Flat

Директива Model

Язык — необязательный операнд, принимающий значения C, PASCAL, BASIC, FORTRAN, SYSCALL и STDCALL. Если он указан, подразумевается, что процедуры рассчитаны на вызов из программ на соответствующем языке высокого уровня, следовательно, если указан язык C, все имена ассемблерных процедур, объявленных как PUBLIC, будут изменены так, чтобы начинаться с символа подчеркивания, как это принято в C.

Модификатор — необязательный операнд, принимающий значения NEARSTACK (по умолчанию) или FARSTACK. Во втором случае сегмент стека не будет объединяться в одну группу с сегментами данных.

Директивы сегментации

Упрощенные директивы определения сегмента

Формат директивы (режим **MASM**)	Назначение
.CODE [имя]	Начало или продолжение сегмента кода
.DATA	Начало или продолжение сегмента инициализированных данных. Также используется для определения данных типа near 1
.CONST	Начало или продолжение сегмента постоянных данных (констант) модуля
.DATA?	Начало или продолжение сегмента неинициализированных данных. Также используется для определения данных типа near
.STACK [размер]	Начало или продолжение сегмента стека модуля. Параметр [размер] задает размер стека
.FARDATA [имя]	Начало или продолжение сегмента инициализированных данных тина far
.FARDATA? [имя]	Начало или продолжение сегмента неинициализированных данных типа far

Директивы сегментации

Идентификаторы, создаваемые директивой MODEL

Имя идентификатора	Значение переменной
@code	Физический адрес сегмента кода
@data	Физический адрес сегмента данных типа near
@fardata	Физический адрес сегмента данных типа far
@fardata?	Физический адрес сегмента неинициализированных данных типа far
@curseg	Физический адрес сегмента инициализированных данных типа far
@stack	Физический адрес сегмента стека

Сегменты, объявленные упрощенными директивами, не требуется закрывать директивой ENDS — они закрываются автоматически, как только ассемблер обнаруживает новую директиву определения сегмента или конец программы.

Директива END

Директива ENDS завершает сегмент, а директива ENDP завершает процедуру. Директива END отмечает конец исходной программы и указывает Ассемблеру, где завершить трансляцию. Поэтому директива END должна присутствовть в каждой исходной программе. Она имеет формат:

END [метка точки входа]

Операнд может быть опущен, если программа не предназначена для выполнения, например, если ассемблируются только определения данных, или эта программа должна быть скомпонована с другим (главным) модулем. Для обычной программы с одним модулем операнд содержит имя, указанное в директиве PROC, которое было обозначено как FAR.

Директивы определения идентификаторов

Псевдооператоры (EQU и =) присваивают выражению символическое имя (идентификатор). Выражение может быть 16-битовой константой, ссылкой на адрес, другим символическим именем, префиксом сегмента и операндом, меткой команды.

1. Определенные знаком = идентификаторы можно переопределять, а определенные директивой EQU – нельзя.

2. Директиву EQU можно использовать как с числовыми, так и с текстовыми выражениями, а знак = только с числовыми.

```
K       EQU   1024          ; Присвоить имя константе
TABLE   EQU   DS:[BP][SI]   ; Присвоить имя комбинации адресов
SPEED   EQU   RATE          ;Определить синоним
count   EQU   CX            ;Присвоить имя регистру
```

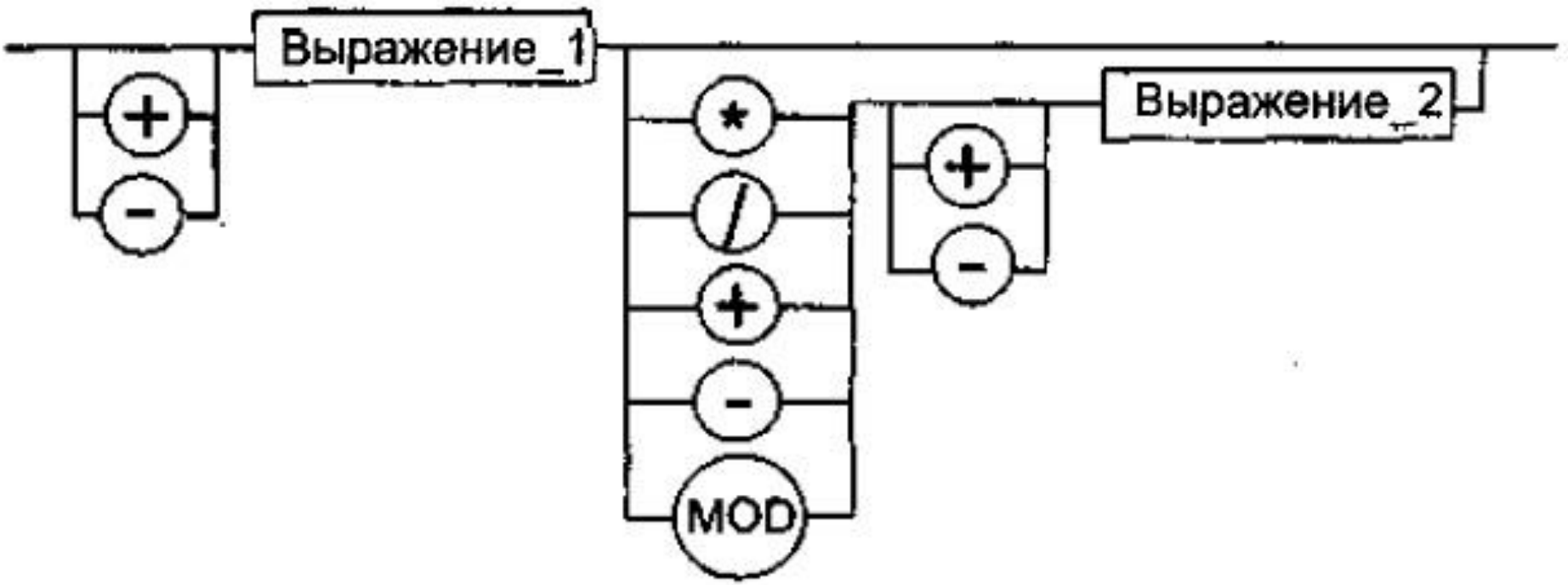
Выражения

Выражение — это набор чисел, меток или строк, связанных друг с другом операторами. Результатом вычисления выражения может быть адрес некоторой ячейки памяти или некоторое константное (абсолютное) значение.

Операторы Ассемблера

Оператор	**Приоритет**
LENGTH, SIZE,WIDTH,MASK,( ),[ ],< >	1
.	2
:	3
PTR, OFFSET, SEG, TYPE, THIS	4
HIGH, LOW	5
+, - (унарные)	6
*, /, MOD, SHL, SHR	7
+, - (бинарные)	8
EQ, NE, LT, LE, GT, GE	9
NOT	10
AND	11
OR, XOR	12
SHORT, .TYPE	13

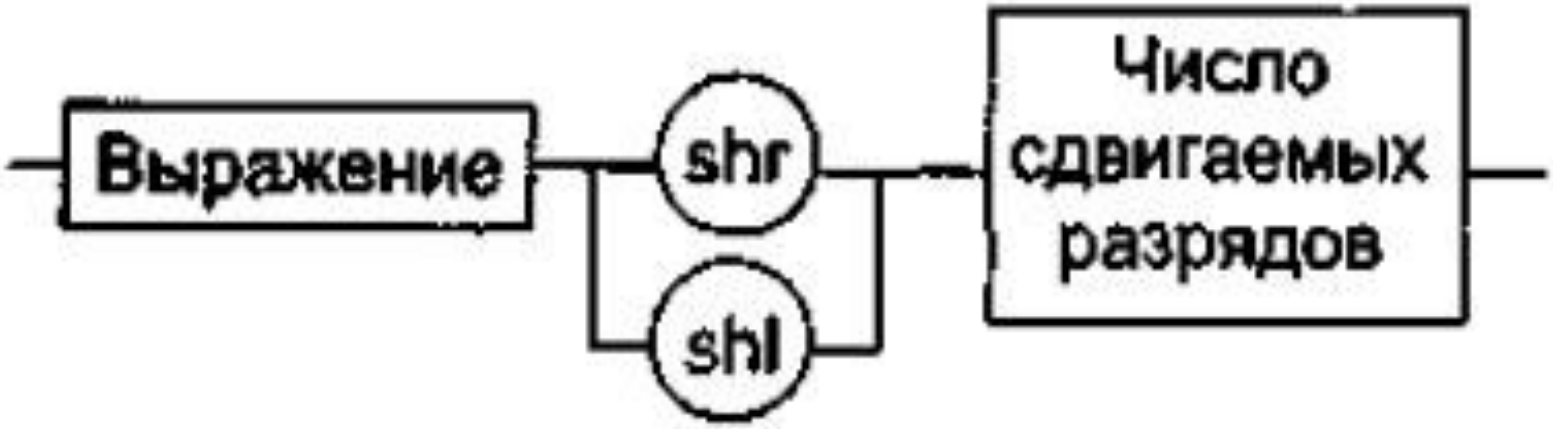
Арифметические операторы



Синтаксис арифметических операторов

```
X           DW    1,2,3,4,5
Y           DB    ?
SIZE_X      EQU   Y-X      ;SIZE_X = 10
```

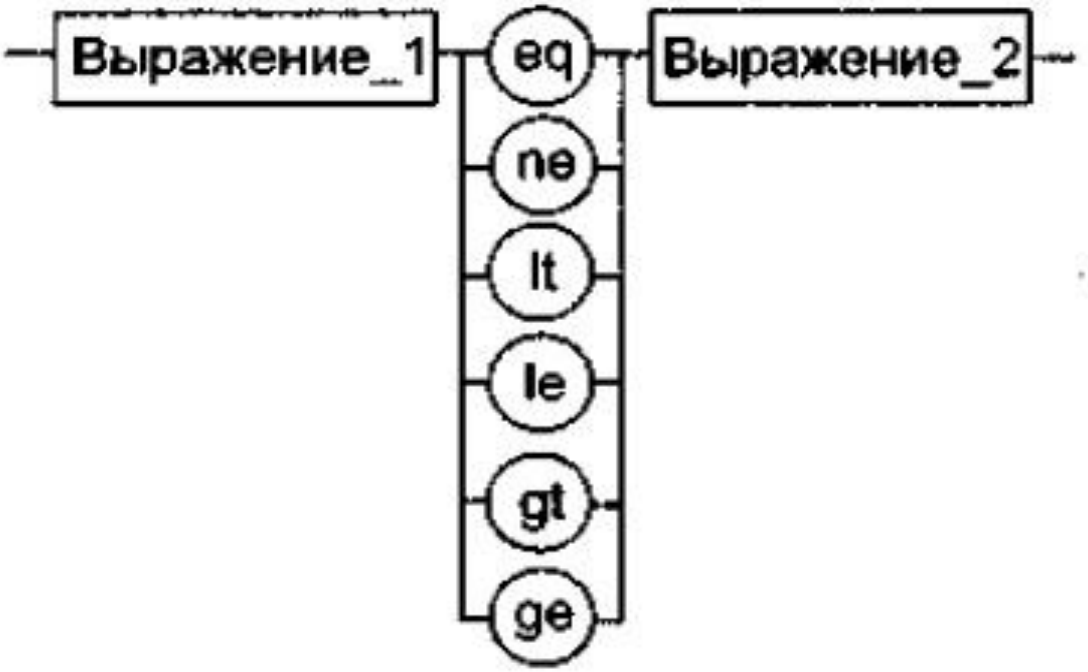
Операторы сдвига



Синтаксис операторов сдвига

```
MAS     EQU     10111011
        MOV     AL,MAS SHR 3            ;AL = 00010111
```

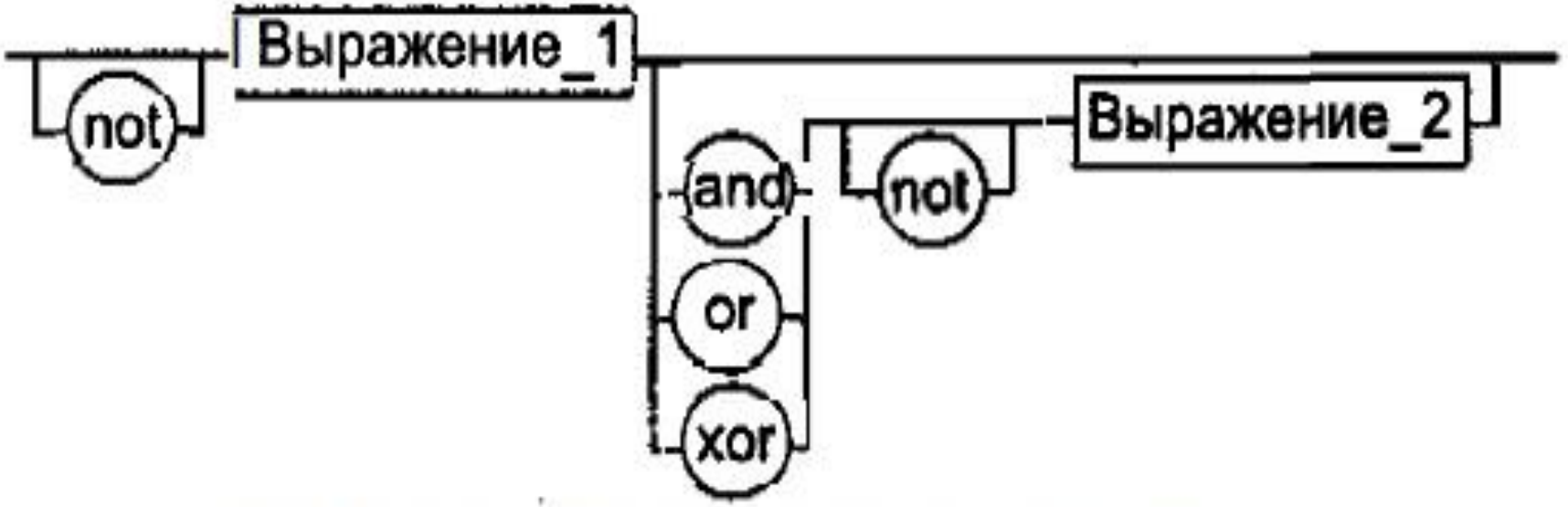
Операторы сравнения



Синтаксис операторов сравнения

```
tab_size    equ 30                  ; размер таблицы
            mov al,tab_size ge 50   ; загрузка размера таблицы в al
            cmp al,0                ; если tab_size < 50, то
            je m1                   ; переход на m1
m1: ...
```

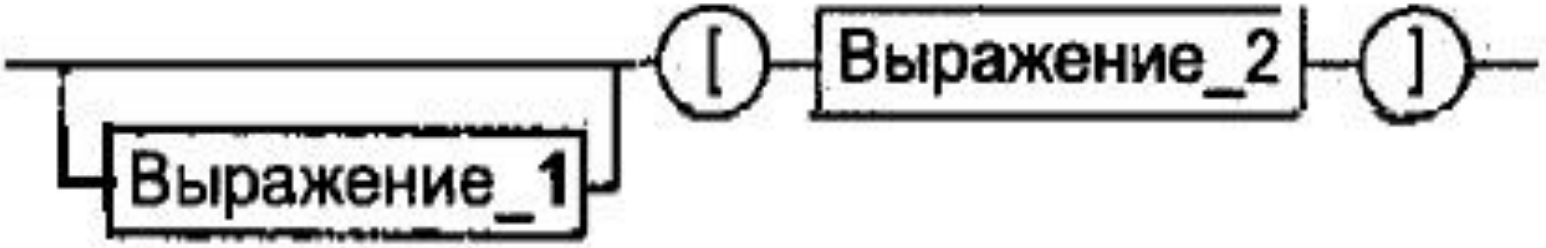
Логические операторы



Синтаксис логических операторов

```
flags   equ 10010011b
        mov al,flags xor 01h    ; al = 10010010    ;пересылка в al поля flags
                                                   ; с инвертированным
                                                   ;правым битом
```

Индексный оператор

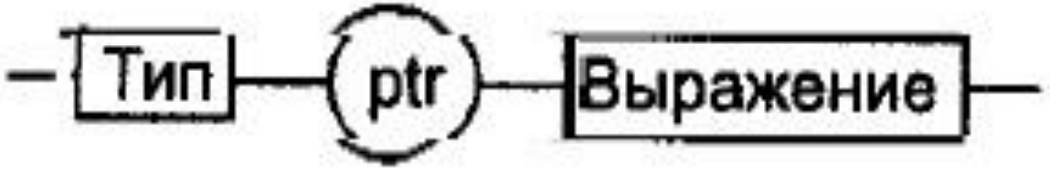


Синтаксис индексного оператора

Если выражение expression1 задано, оно может быть целым значением, перемещаемым операндом или абсолютным символом. Expression2 может быть целым значением или абсолютным символом, а также перемещаемым операндом, в том случае, когда выражение expression1 не задано.

mov a x , m a s [s i] ; пересылка слова по адресу mas + (si) в регистр ax

Оператор переопределения типа



Синтаксис оператора переопределения типа

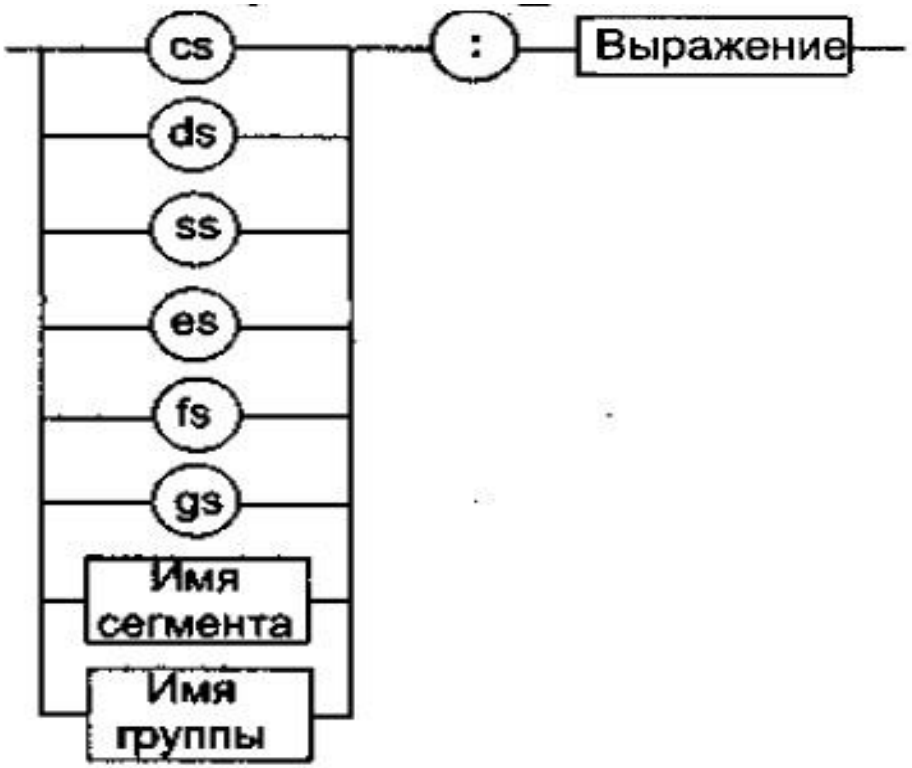
Тип может принимать одно из следующих значений: BYTE, WORD, DWORD, QWORD, TBYTE, NEAR, FAR

```
d_wrd dd      0
      mov     al,byte ptr d_wrd+1    ;пересылка второго байта из
                                     ;двойного слова

P     dw      10  dup     (?)
Q     equ     byte ptr P

S:    mov     cx,10
S1    equ     far ptr S
```

Оператор переопределения сегмента



Синтаксис оператора переопределения сегмента

```
.code
    jmp metl     ; обход обязателен, иначе поле ind
                 ; будет трактоваться как очередная команда
ind db 5         ; описание поля данных в сегменте команд,
metl:
    mov al,cs:ind ; переопределение сегмента позволяет работать с данными,
                 ; определенными внутри сегмента кода
```

Операторы получения составляющих адреса

SEG expression

Выдает значение сегмента, в котором расположено expression. Выражение может быть переменной, меткой, именем сегмента, именем группы либо другим символом.

OFFSET expression

Выдает число байт между выражением и началом сегмента, в котором оно определено. Выражение может быть меткой, именем сегмента или переменной.

```
.data
pole dw 5
. . .
.code
. . .
      mov ax,seg pole
      mov es,ax
      mov dx,offset pole      ;теперь в паре es:dx полный адрес pole
```

Операторы получения байтов выражения

HIGH expression .

Этот оператор возвращает старшие 8 бит выражения expression.

```
mov  bl, high 1234h ;Эта команда поместит 12h в bl.
```

LOW expression .

Этот оператор возвращает старшие 8 бит выражения expression.

```
mov  al, low 1234h ;Эта команда поместит 34h в al.
```

Операторы получения размера переменной

LENGTH переменная .

Возвращает число единиц типа BYTE, WORD, DWORD, QWORD или TBYTE, занимаемых переменной. Тип переменной определяет, в каких единицах измерения возвращается ее длина.

TYPE выражение .

Выдает число байт, необходимых для хранения переменной того типа, каким является выражение; для метки NEAR выдает 0FFFFh, а для метки FAR - 0FFFEh.

SIZE переменная .

Выдает число байт, занимаемых переменной. Значение, возвращаемое этим оператором, равно длине (LENGTH) переменной, повторенной TYPE (тип) раз:

SIZE переменная=(LENGTH переменная)*(TYPE переменная)

```
C dw 23, 45 dup (24)
T equ length C          ; Значение T равно 46.
R equ type C            ; Значение R равно 2.
D equ size C            ; Значение D равно 92.
```

Процедуры

Процедура (подпрограмма) — основная функциональная единица декомпозиции задачи. Варианты размещения в программе:

- в начале программы (до первой исполняемой команды)
- в конце
- внутри другой процедуры или основной программы
- в другом модуле

Процедуры

```
имя_процедуры PROC [[модификатор_языка] язык] [расстояние]
[ARG список_аргументов]
[RETURNS список_аргументов]
[LOCAL список_аргументов]
[USES список_регистров]

        команды,
        директивы
        языка
        ассемблера

[имя_процедуры] ENDP
```

Заголовок процедуры

Тело процедуры

Конец процедуры

Процедуры

```
model small
.stack 100h
.data
.code
my_proc  proc   near
ret
my_proc  endp
start:
end start
```

Процедуры

```
model small
.stack 100h
.data
.code
start:
mov ax,4c00h
int 21h         ;возврат управления операционной
                ;системе
my_proc proc  near
ret
my_proc endp
end start
```

Процедуры

```
model small
.stack 100h
.data
.code
start:
jmp ml
my_proc proc near
ret
my_proc endp
ml:
mov ax,4c00h
int 21h ;возврат управления операционной
        ;системе
end start
```

Процедуры. Команды работы с контекстом

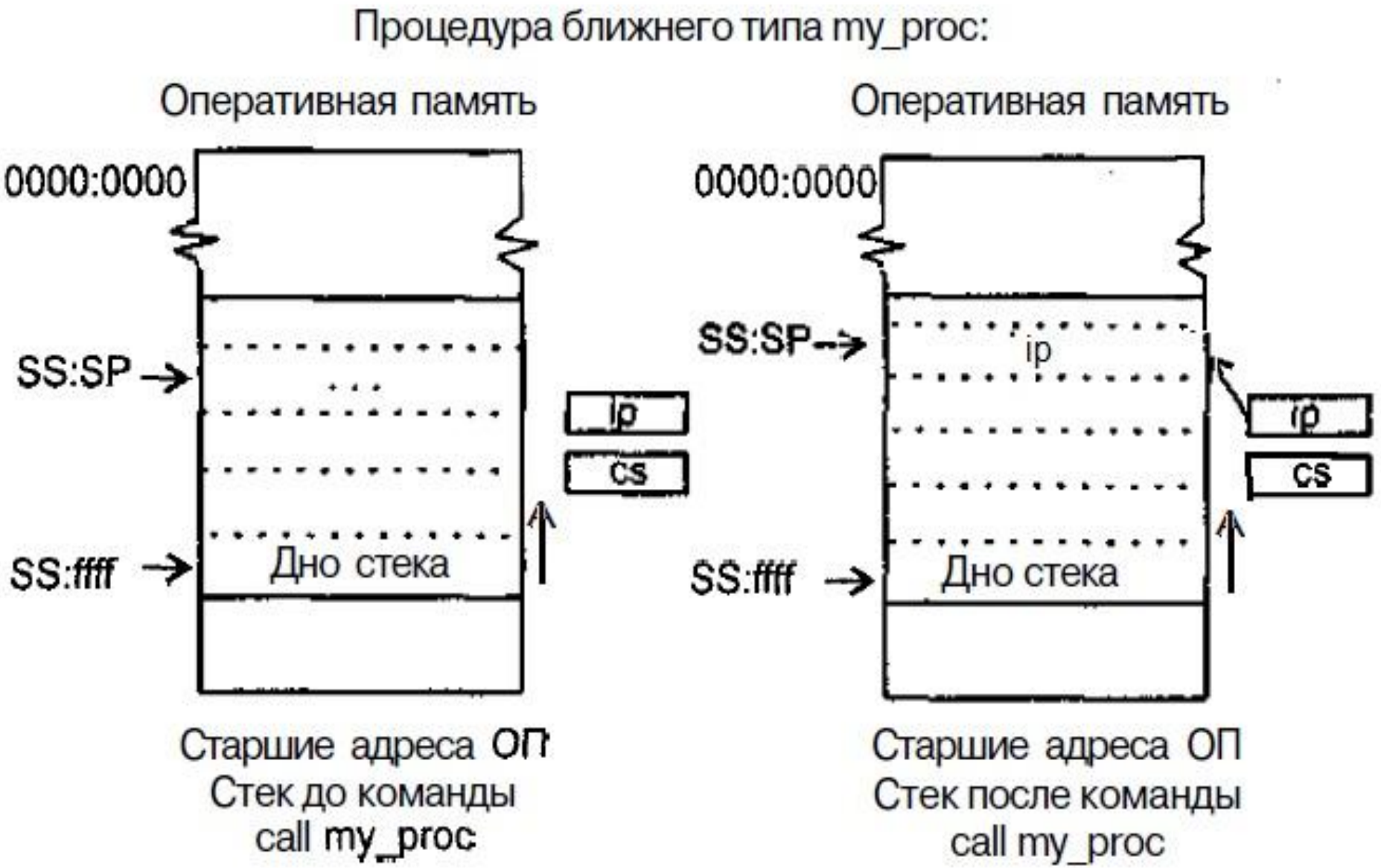
Контекст - информация о состоянии программы в точке вызова процедуры.

call [модификатор] имя_процедуры

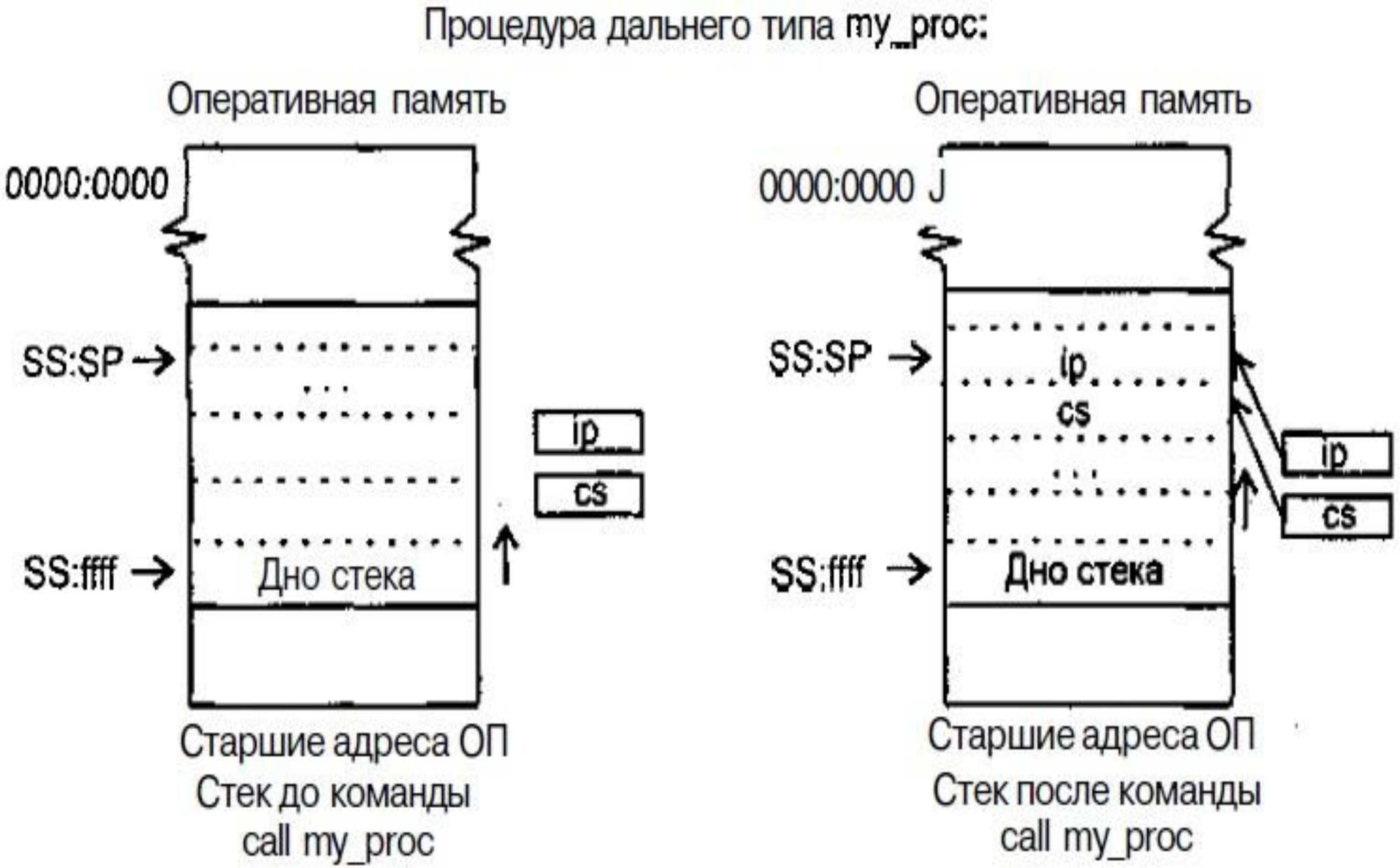
ret [число]

Необязательный параметр [число] обозначает количество элементов, удаляемых из стека при возврате из процедуры. Размер элемента определяется параметрами директивы SEGMENT — use16 и use32 (или соответствующим параметром упрощенных директив сегментации). Если указан параметр usel6, то [число] — это значение в байтах; если use32 — в словах.

Процедуры. Внутрисегментный вызов



Процедуры. Межсегментный вызов



Оператор USES

Позволяет перечислить имена всех регистров, значение которых изменяется в процедуре. При его обработке компилятор ассемблера выполняет две вещи. Во-первых, в начале процедуры автоматически генерируется последовательность команд PUSH, с помощью которых в стеке сохраняются значения регистров, указанных в операторе USES. Во-вторых, при выходе из процедуры (точнее, перед каждой командой RET, если в процедуре их несколько) автоматически восстанавливаются значения этих регистров. Имена регистров разделяются пробелом или символом табуляции (не запятой!).