Язык ассемблера

Основы

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Общая характеристика

Язык ассемблера по существу аналогичен машинному языку, но представлен в форме, более понятной людям.

Состав:

- директивы определения данных
- команды
- директивы управления средой выполнения
- макросредства

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Директивы определения данных

Ассемблер позволяет программисту назначать имена переменным, т.е. работать не с их адресами, а с более понятными именами.

Для этого используются директивы вида DB, DW, DD – определить байт, слово, двойное слово. Примеры:

A DB 151 ; комментарий: ***char A = 151;***

B DB ?; ***char B;***

M DW 2001,-154,15; ***long M[ ] = {2001,-154,15};***

В отличие от языков высокого уровня память для переменных выделяется непосредственно по месту употребления директивы. Поэтому нежелательно их использовать между командами (но и не запрещено!).

Для определения массивов с повторяющимися элементами используется конструкция DUP (duplicate):

M DB 8 DUP(0); массив из 8 байтов со значениями 0

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Команды

Команды ассемблера это символическая запись машинных команд. Общий синтаксис:

[<метка>]: <мнемокод> [<операнды>] [; <комментарий>]

- Метки используются для указания цели перехода в операциях условного и безусловного перехода и вызовов подпрограмм;
- Мнемокод – один из набора команд процессора;
- Операнды определяют друг от друга запятыми. В качестве операнда могут быть число, регистр, имя переменной. Примеры:
  - **ADD AH, 12;** **AH = AH +12**
  - **SUB SI, Z;** **SI = SI – Z**
  - **INC BL;** **BL = BL + 1**

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Команды

В качестве операнда может использоваться ссылка (косвенная адресация):

MOV [BX], 300; *BX = 300, поместить по адресу, содержащемуся в BX, число 300

Часто используется модификация адресов (массивы):

MOV AX, A[SI]; индексная адресация: AX = [A+SI]

MOV AX, A[BX][SI]; базово-индексная AX = [A+BX+SI]

Для уточнения размеров непосредственных операндов используется ключевое слово **PTR:**

~~**MOV [BX], 0;**~~ непонятно «какой ноль» пересылать по адресу в BX

MOV [BX], byte ptr 0; пересылка байта

MOV [BX], word ptr 0; пересылка слова

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Команды

Организация переходов

Безусловный переход:

JMP L; переход к метке L

…

L: MOV AX, 0;

Условный переход: задача: ***if (A==B) {A = 0; B = 1;}***

```
MOV AX, A;
MOV BX, B;
CMP AX, BX; сравнение, устанавливается (или нет) флаг равенства E
JNE L;      если E=0 (не равно), переход к метке L
MOV A, 0;
MOV B, 1;
L:  …
```

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Команды

Циклы

Задача:

char A;

…

for(int i = 0;i<5;i++) A*=3;

Решение:

MOV AL, A;

MOV CX, 5;

L: MUL 3; AX = AL*3

LOOP L; CX--, переход на метку L, если CX!=0

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

Подпрограммы (процедуры) определяются с помощью ключевых слов **PROC** и **ENDP**:

< имя процедуры> PROC [<тип вызова>]

<код процедуры, включая команду RET>

< имя процедуры> ENDP

Тип вызова (**near** или **far**), как и тип возврата определяет, будет ли при вызове и возврате изменяться содержимое сегментного регистра **CS**.

far – дальний вызов с переходом в другой сегмент;

near – ближний вызов в пределах одного сегмента, изменяется только указатель команд IP.

В большинстве случаев тип вызова определяется автоматически.

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

Вызов процедуры:

CALL <имя процедуры>

Передача параметров между «основной программой» и подпрограммой (допустимы также и вложенные вызовы) может быть осуществлена самыми различными способами: через регистры, стек или как-нибудь иначе на усмотрение программиста.

Хорошо написанная процедура не должна изменять содержимое регистров (чтобы не испортить выполнение основной программы). Поэтому содержимое регистров обычно сохраняют в стеке командами **push** или **pusha**, а потом восстанавливают командами **pop** или **popa**.

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

Пример №1 Передача параметров через регистры

```
; процедура AX = max(AX,BX)
MAX    PROC
       CMP AX,BX
       JGE MAX1; GE – greater or equal
       MOV AX,BX
MAX1:  RET; выход из подпрограммы
MAX    ENDP
```

Перед вызовов процедуры необходимо поместить параметры в регистрах AX и BX.

Результат окажется в регистре AX.

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

Решение задачи с помощью процедуры

; c = max(a,b) + max(5,a-1)

```
MOV AX,A;  AX = A
MOV BX,B;  BX  = B
CALL MAX; AX = max(AX,BX)
MOV C,AX;  C = AX
MOV AX,5;  AX = 5
MOV BX,A;  BX = A
DEC BX;      BX = BX-1
CALL MAX; AX = max(AX,BX)
ADD C,AX;  C = C+AX
```

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

Что на самом деле делают инструкции CALL и RET?

Команда CALL:

- сохраняет *адрес возврата* в стеке (при ближнем вызове только смещение offset, при дальнем – также и сегментную часть);
- записывает в указатель команд IP смещение адреса первой команды процедуры; в случае дальнего вызова записывает также в сегментный регистр CS сегментную часть адреса.

Команда RET:

- считывает из стека смещение *адреса возврата* и записывает его в указатель команд IP; в случае дальнего вызова считывает сегментную часть адреса из стека и записывает в сегментный регистр CS.

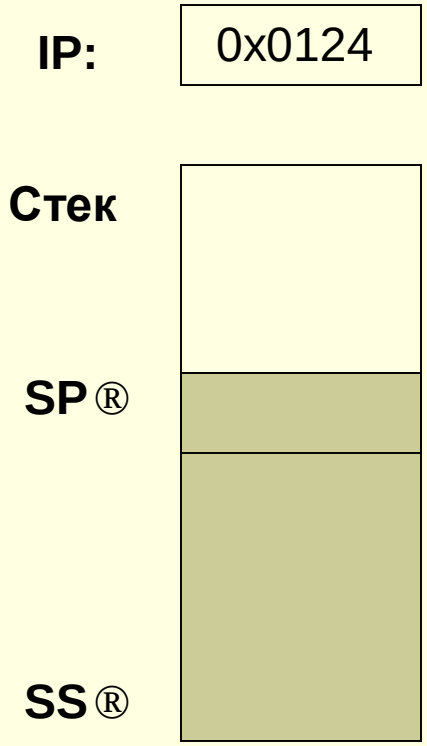
Таким образом обеспечивается возврат к выполнению основной программы.

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация (ближний вызов): шаг 1

MAX PROC
0x0011 **CMP AX,BX**
0x0012 **JGE MAX1;**
0x0015 **MOV AX,BX**
0x0016 **MAX1: RET;**
MAX ENDP
Основная программа:

0x0121 **MOV AX,A;**
0x0124 **MOV BX,B;**
0x0127 **CALL MAX;**
0x012A **MOV C,AX;**

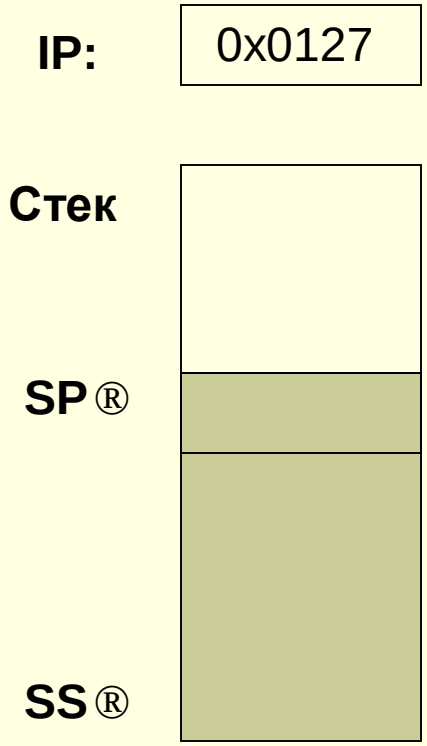


PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация (ближний вызов): шаг 2

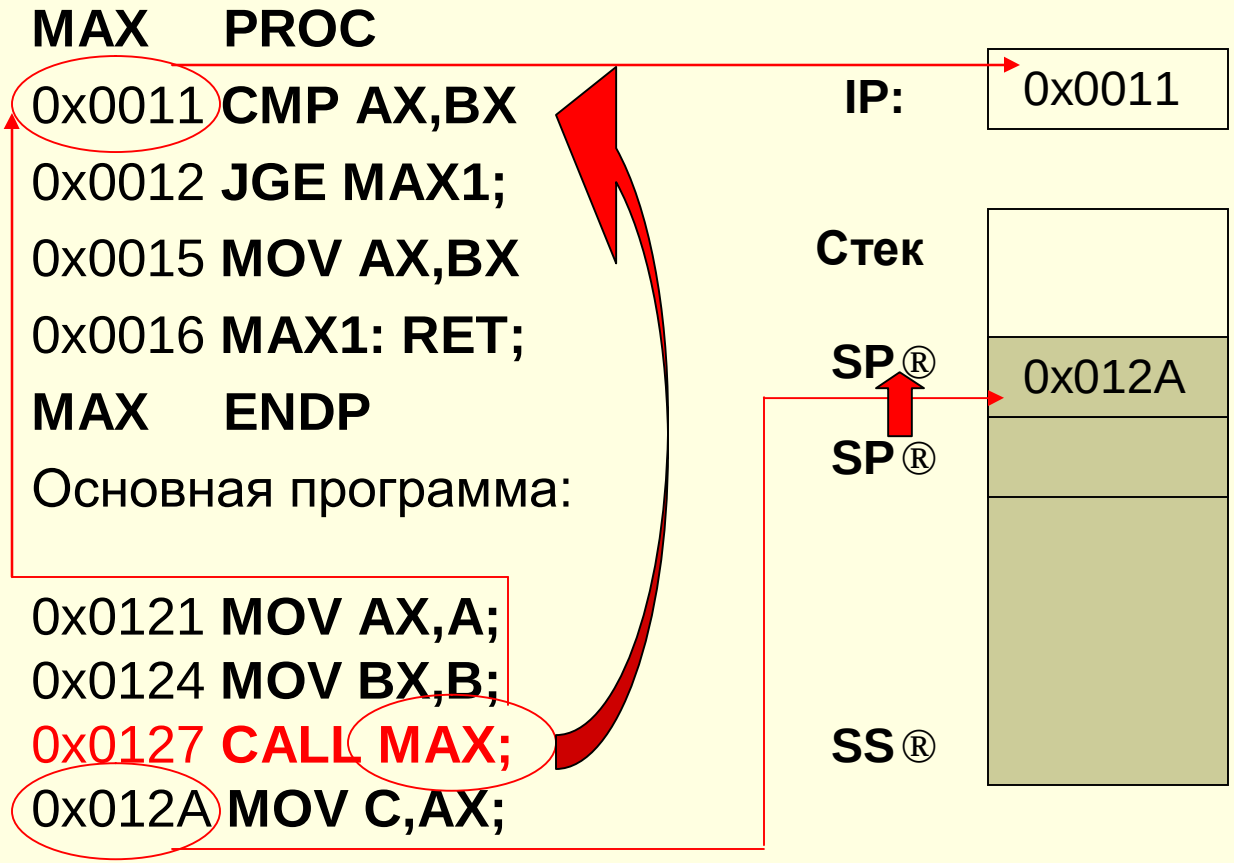
MAX PROC
0x0011 **CMP AX,BX**
0x0012 **JGE MAX1;**
0x0015 **MOV AX,BX**
0x0016 **MAX1: RET;**
MAX ENDP
Основная программа:

0x0121 **MOV AX,A;**
0x0124 **MOV BX,B;**
0x0127 **CALL MAX;**
0x012A **MOV C,AX;**



PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация (ближний вызов): шаг 3



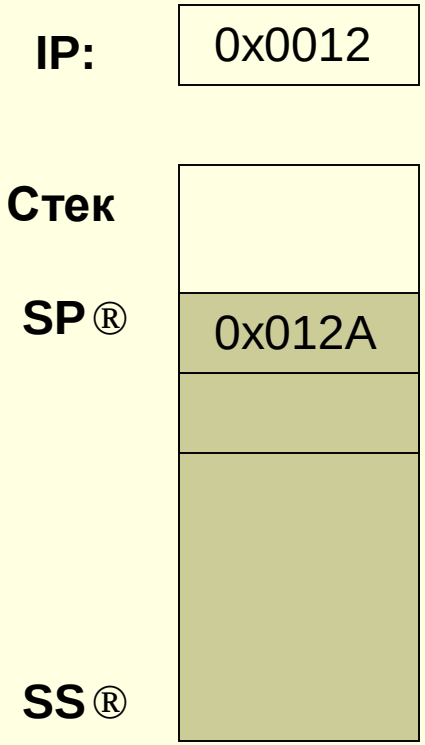
PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация (ближний вызов): шаг 4

```
MAX     PROC
0x0011 CMP AX,BX
0x0012 JGE MAX1;
0x0015 MOV AX,BX
0x0016 MAX1: RET;
MAX     ENDP
```

Основная программа:

```
0x0121 MOV AX,A;
0x0124 MOV BX,B;
0x0127 CALL MAX;
0x012A MOV C,AX;
```



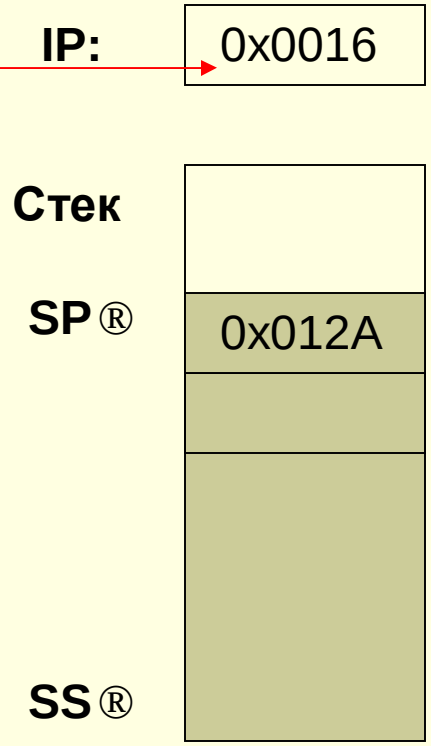
PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация (ближний вызов): шаг 5

```
MAX     PROC
0x0011 CMP AX,BX
0x0012 JGE MAX1;
0x0015 MOV AX,BX
0x0016 MAX1: RET;
MAX     ENDP
```

Основная программа:

```
0x0121 MOV AX,A;
0x0124 MOV BX,B;
0x0127 CALL MAX;
0x012A MOV C,AX;
```



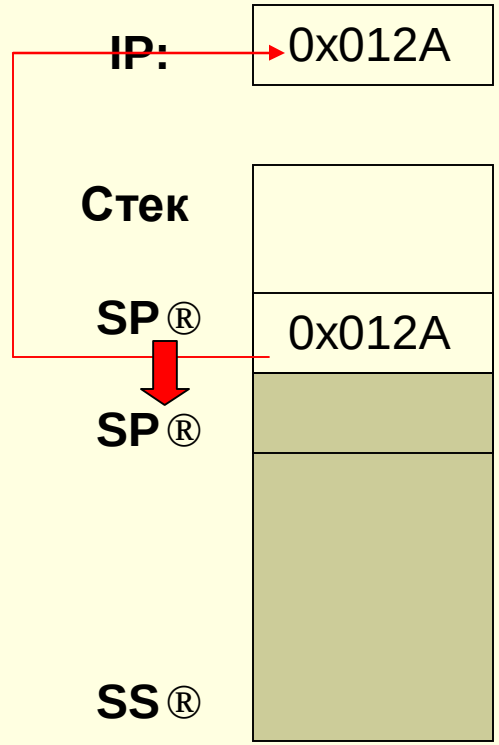
в примере A > B

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация (ближний вызов): шаг 6

MAX PROC
0x0011 **CMP AX,BX**
0x0012 **JGE MAX1;**
0x0015 **MOV AX,BX**
0x0016 **MAX1: RET;**
MAX ENDP
Основная программа:

0x0121 **MOV AX,A;**
0x0124 **MOV BX,B;**
0x0127 **CALL MAX;**
0x012A **MOV C,AX;**



PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация (ближний вызов): шаг 7

MAX PROC
0x0011 **CMP AX,BX**
0x0012 **JGE MAX1;**
0x0015 **MOV AX,BX**
0x0016 **MAX1: RET;**
MAX ENDP
Основная программа:

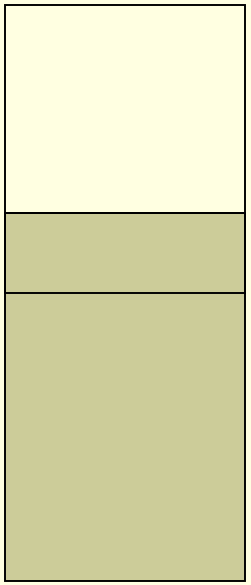
0x0121 **MOV AX,A;**
0x0124 **MOV BX,B;**
0x0127 **CALL MAX;**
0x012A **MOV C,AX;**

IP: 0x012D

Стек

SP®

SS®



PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

Пример №2. Процедура поиска максимального элемента в массиве

Задача:

DL = max(X[i]) + max(Y[i])

Исходные данные:

X DB 100 DUP (?); массив из 100 байт с неопределенными данными

Y DB 25 DUP (?); массив из 25 байт с неопределенными данными

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

```
; процедура MAX: AL = max(W[0…N-1)
; где BX – начальный адрес W, CX = N
MAX     PROC
        PUSH BX;        спасти в стеке регистр BX
        PUSH CX;        спасти в стеке регистр CX
        MOV AL,0;       AL = 0
MAX1:   CMP [BX],AL;    сравнить элемент массива с AL
        JLE MAX2;       если W[i]<AL перейти к метке MAX2
        MOV AL, [BX];   сохранить в AL «новый максимум»
MAX2:   INC BX;         BX++, перейти к следующему элементу массива
        LOOP MAX1;      CX--, если CX!=0, перейти к метке MAX1
        POP CX;         восстановить регистр CX
        POP BX;          восстановить регистр BX
        RET;            выход из подпрограммы
MAX     ENDP
```

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

Решение задачи DL = max(X[i]) + max(Y[i])

```
LEA BX,X;      записать адрес массива X в регистр BX
MOV CX,100;    записать в CX число элементов массива X
CALL MAX;      определить максимальный элемент массива X
MOV DL,AL;     DL = AL
LEA BX,Y;      записать адрес массива Y в регистр BX
MOV CX,25;     записать в CX число элементов массива Y
CALL MAX;      определить максимальный элемент массива Y
ADD DL,AL;      DL += AL
```

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

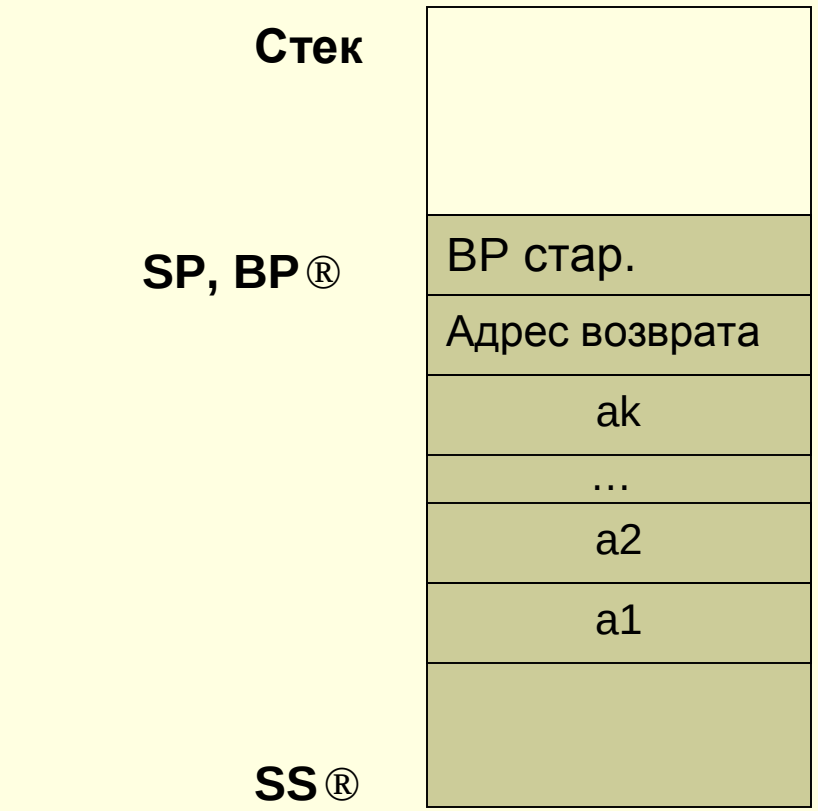
Передача параметров через стек (принята в большинстве языков программирования)

Вызов процедуры: P(a1, a2,…, ak)

```
PUSH a1;
PUSH a2;
…
PUSH ak;
CALL P;
```

Сама процедура:

```
P       PROC
        PUSH BP;
        MOV BP, SP
; обращение к параметру ak:
[BP+4]
```



PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

Подпрограмма NULL(A,N) – обнуление N байт, начиная с адреса A, A и N передаются через стек

```
NULL    PROC;                          PUSH BP; спасти BP
        MOV BP,SP; BP = SP
        PUSH BX;  спасти BX
        PUSH CX;  спасти CX
        MOV CX, [BP+4];  CX = N;
        MOV BX, [BP+6];  BX = A;
NULL1:  MOV BYTE PTR [BX], 0;  *BX = 0;
        INC BX;  BX++
        LOOP NULL1; CX--;  if(!CX) goto NULL1;
        POP CX; восстановить CX
        POP BX; восстановить BX
        POP BP; восстановить BP
        RET 4; возврат из подпрограммы с «очисткой» 4 байт в стеке
NULL    ENDP
```

на C++:

```
void nulling(char A[ ], int N)
        {
  for(int i=0; i<N;i++) A[i] = 0;
}
```

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Организация подпрограмм

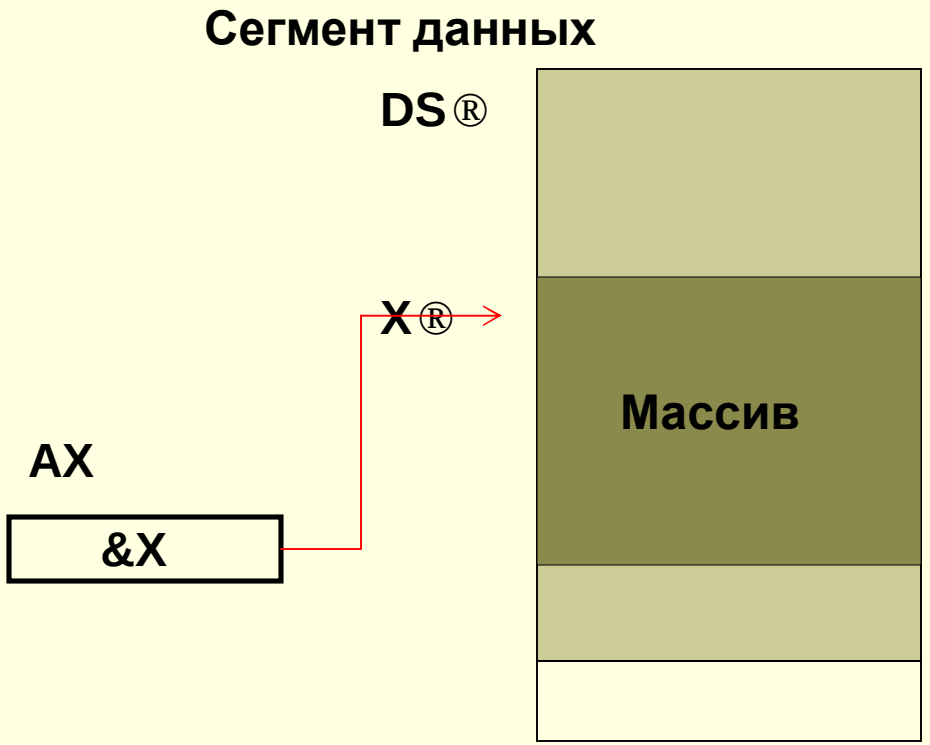
Задача: обнулить массив X

```
X DB 100 DUP (?)
....
LEA AX,X;       AX = &X
PUSH AX;        записать AX в стек
MOV AX, 100;    AX = 100
PUSH AX;        записать AX в стек
CALL NULL;      вызвать подпрограмму
```

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №1

```
    LEA AX,X;
    PUSH AX;
    MOV AX, 100;
    PUSH AX;
    CALL NULL;
AB: ....
```

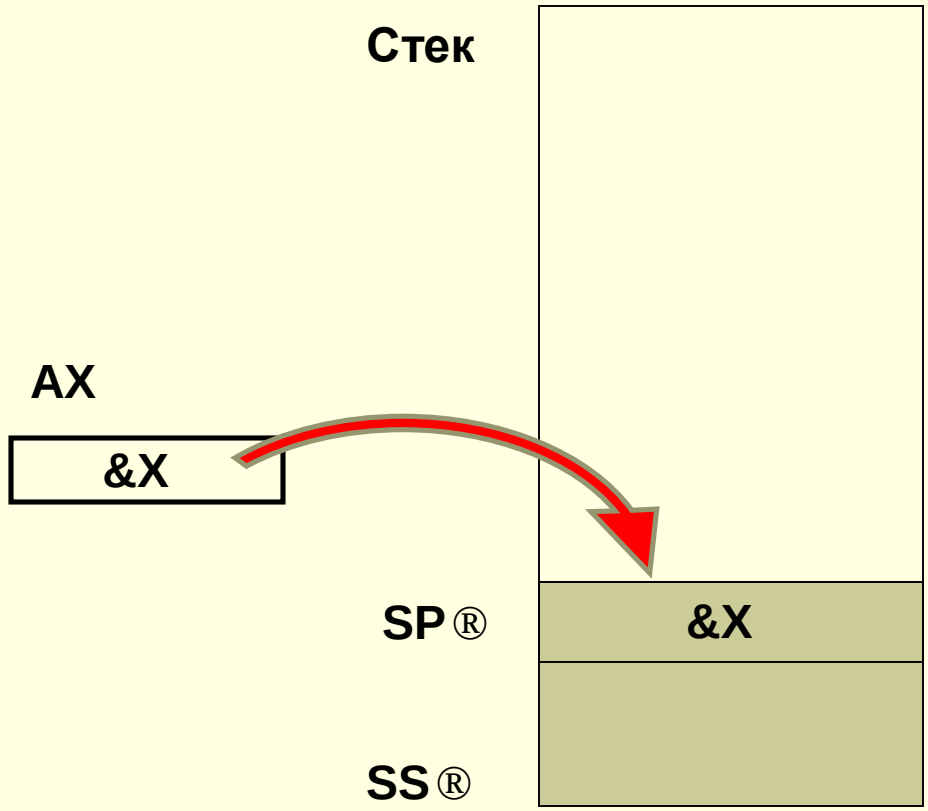


В регистр AX записывается адрес первого элемента массива

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №2

```
    LEA AX,X;
    PUSH AX;
    MOV AX, 100;
    PUSH AX;
    CALL NULL;
AB: ....
```

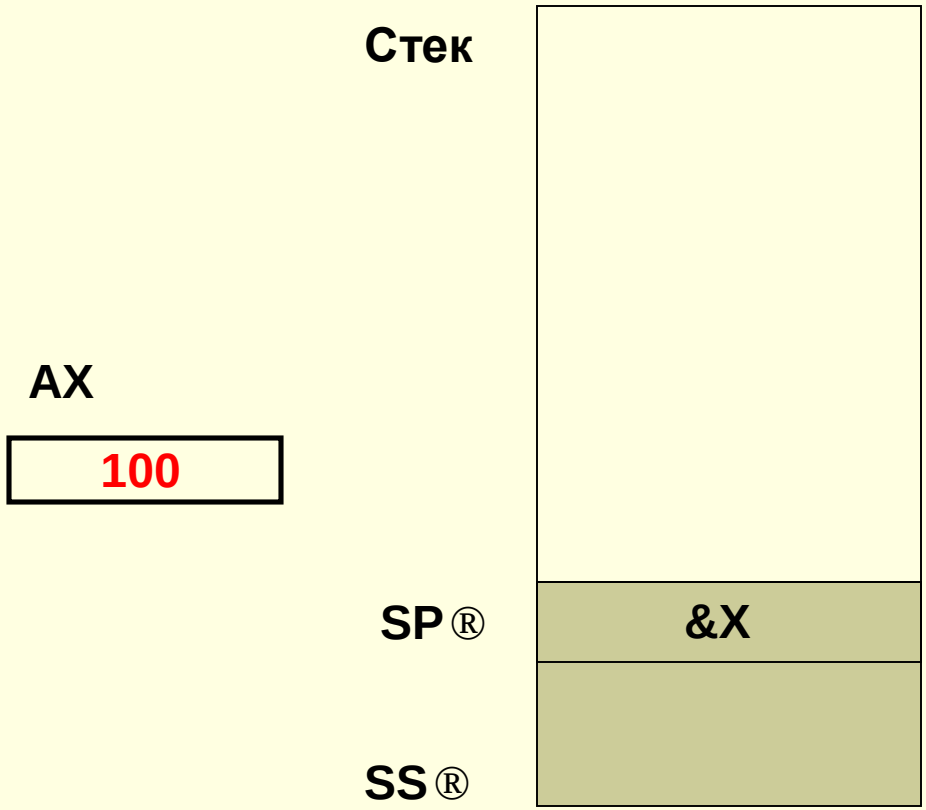


В стек записывается адрес первого элемента массива

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №3

```
LEA AX,X;
PUSH AX;
MOV AX, 100;
PUSH AX;
CALL NULL;
AB: ....
```

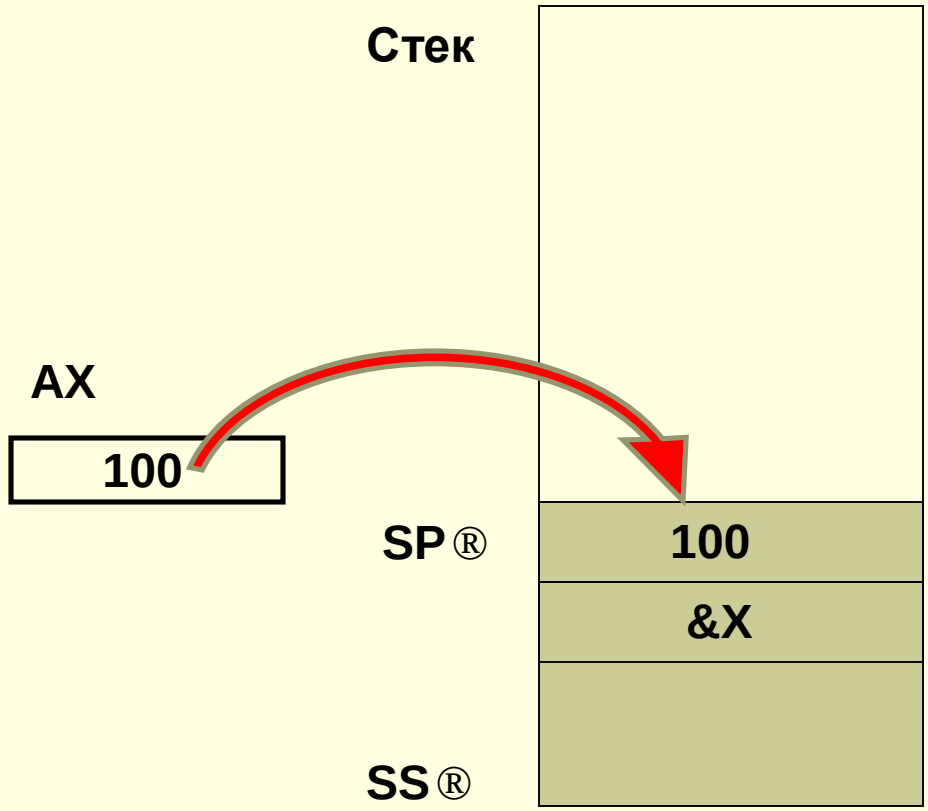


В AX записывается число элементов в массиве

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №4

```
    LEA AX,X;
    PUSH AX;
    MOV AX, 100;
    PUSH AX;
    CALL NULL;
AB: ....
```



В стек записывается число элементов массива

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №5

```
LEA AX,X;
PUSH AX;
MOV AX, 100;
PUSH AX;
CALL NULL;
AB: ....
```

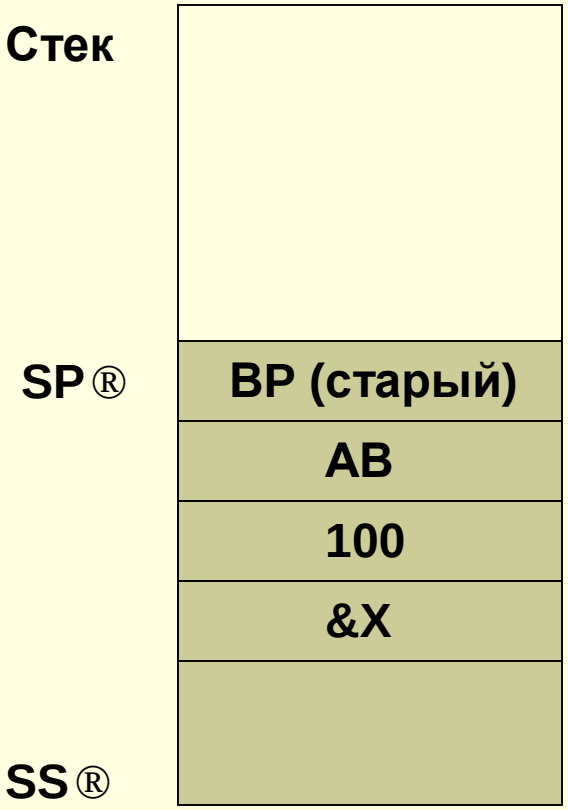
Передача управления NULL

В стек записывается адрес возврата и управление передается процедуре NULL

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №6

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

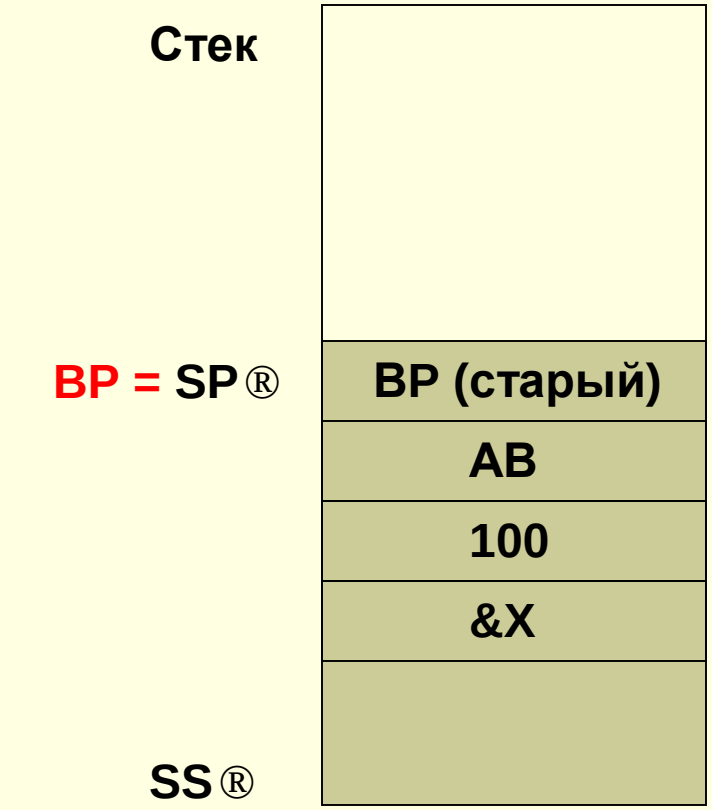


В стеке сохраняется содержимое регистра BP

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №7

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

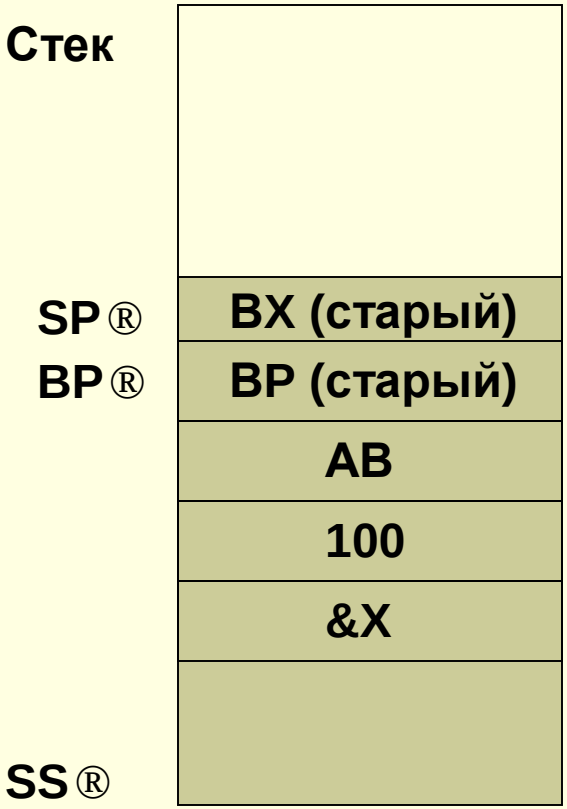


В регистр BP записывается адрес вершины стека

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №8

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

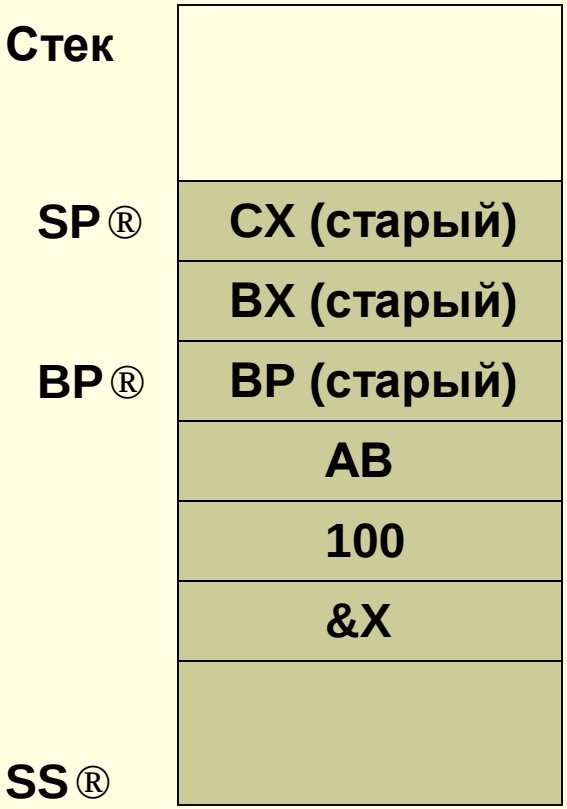


В стеке сохраняется содержимое регистра BX

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №9

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

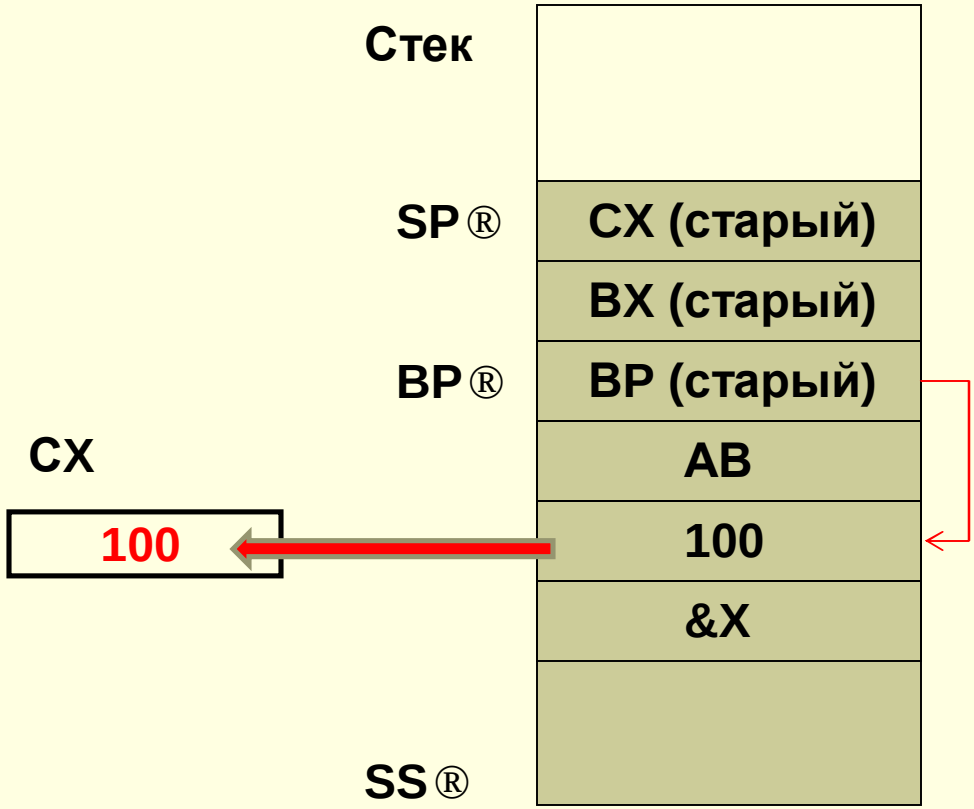


В стеке сохраняется содержимое регистра CX

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №10

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

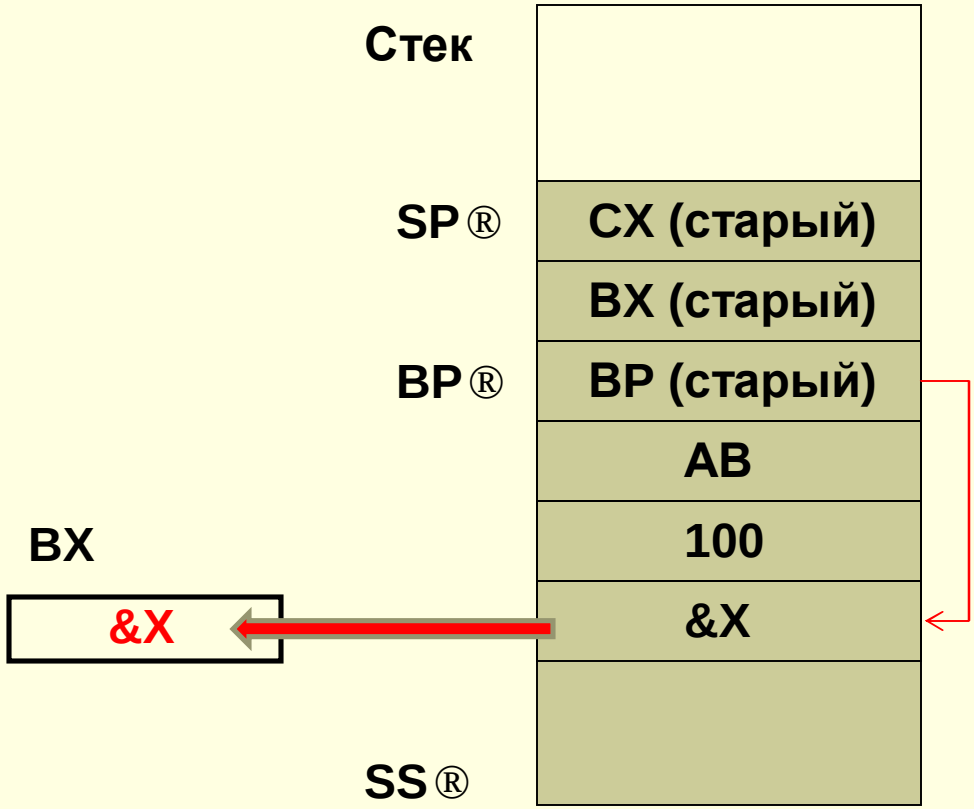


В CX записывается число элементов массива

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №11

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

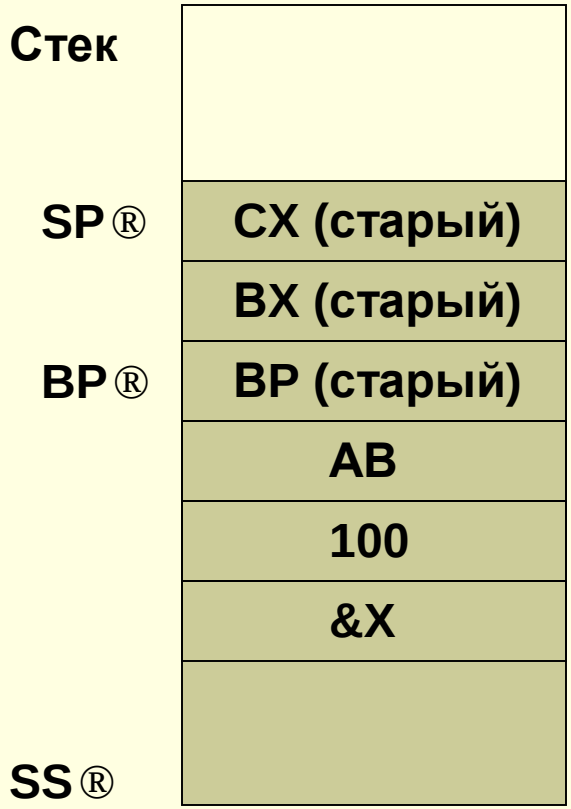


В BX записывается адрес первого элемента массива

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №12 (укрупненный)

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

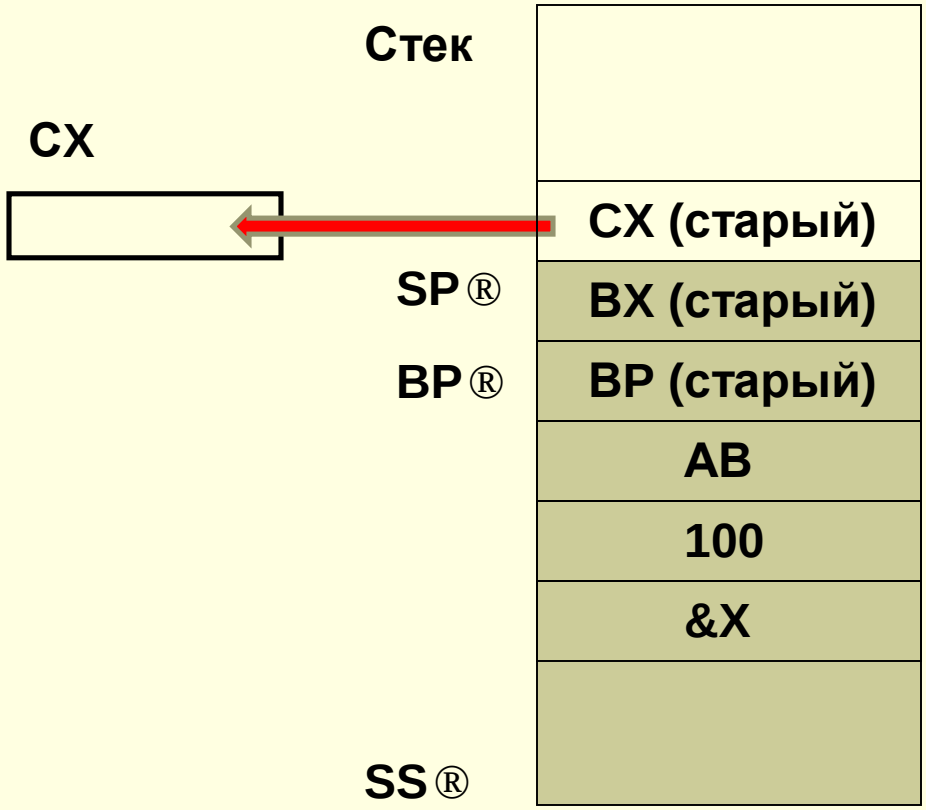


Обнуляются элементы массива

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №13

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

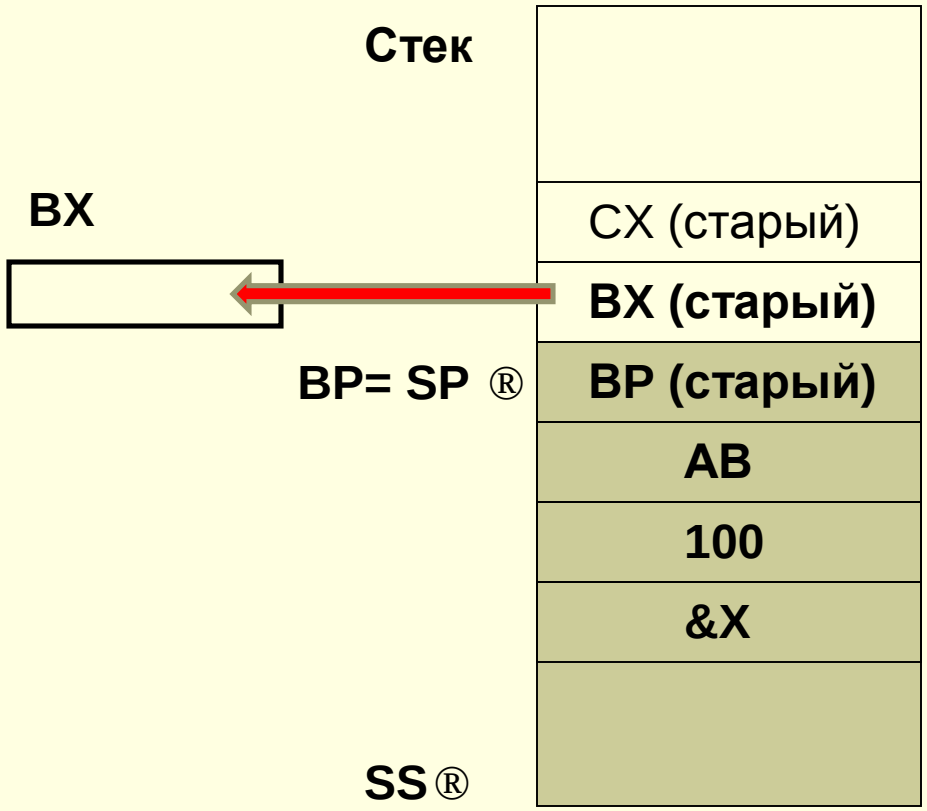


Восстанавливается содержимое регистра CX

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №14

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

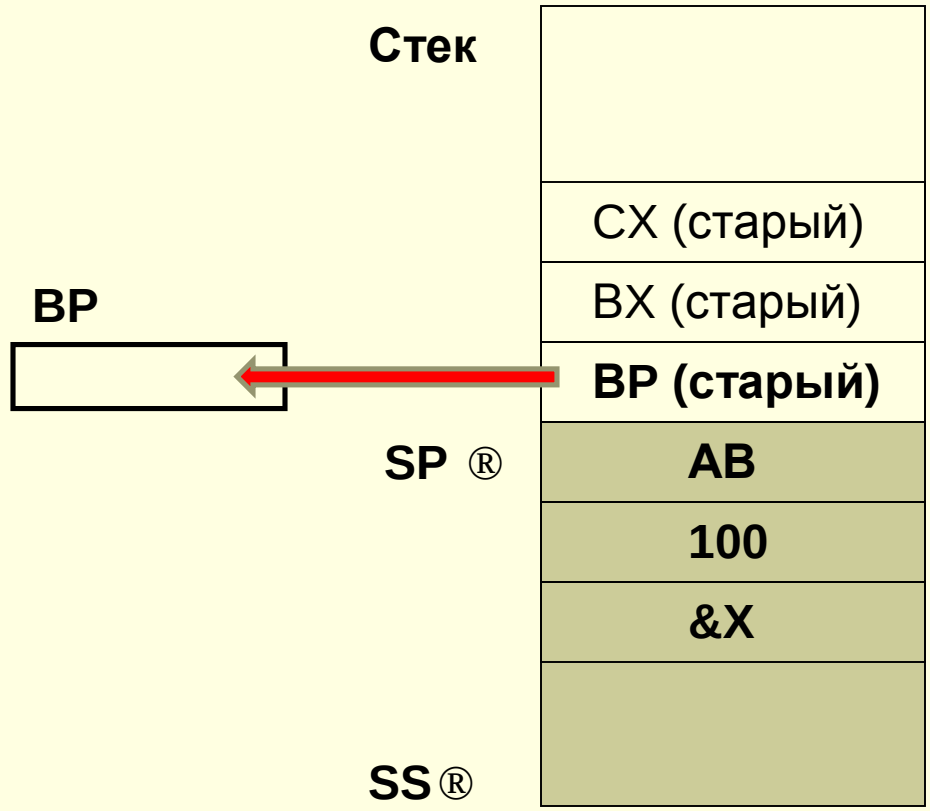


Восстанавливается содержимое регистра BX

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №15

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```

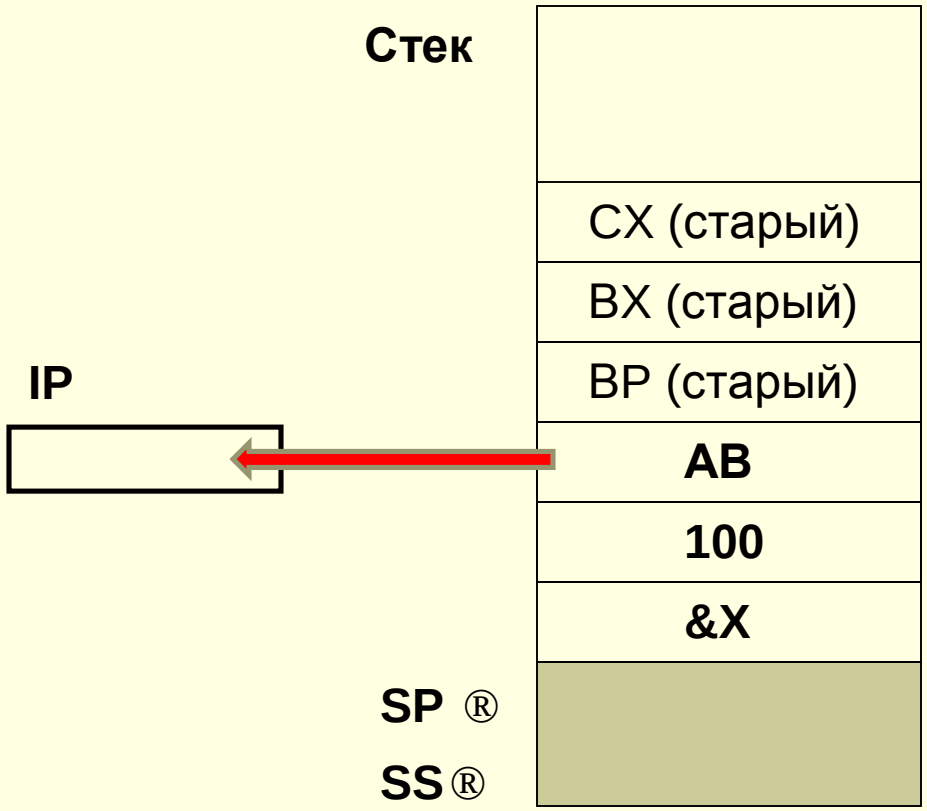


Восстанавливается содержимое регистра BP

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №15

```
NULL    PROC;
        PUSH BP;
        MOV BP,SP;
        PUSH BX;
        PUSH CX;
        MOV CX, [BP+4];
        MOV BX, [BP+6];
NULL1:  MOV BYTE PTR [BX], 0;
        INC BX;
        LOOP NULL1;
        POP CX;
        POP BX;
        POP BP
        RET 4;
NULL    ENDP
```



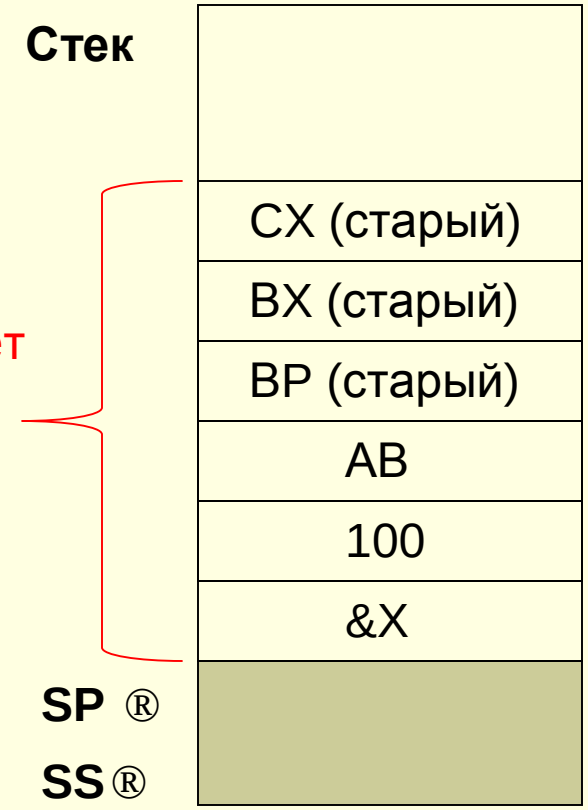
Адрес возврата записывается в указатель команд IP (возврат управления), стек «очищается» от параметров

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Демонстрация работы, шаг №16

```
LEA AX,X;
PUSH AX;
MOV AX, 100;
PUSH AX;
CALL NULL;
AB: ....
```

Это уже никому не нужно и будет затерто при дальнейшем использовании стека



В результате: задача выполнена, регистры не пострадали

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Передача параметров через стек. Выводы.

- для адресации параметров используется регистр BP (base pointer), который настраивается на вершину стека, где хранится в данный момент старое содержимое этого регистра;
- для доступа к последнему записанному в стек параметру используется конструкция [BP+4] при ближнем вызове (2 байта для хранения старого значения BP + 2 байта для хранения смещения адреса возврата) или [BP+6] при дальнем вызове (2 байта для хранения старого значения BP + 4 байта для хранения полного адреса возврата);
- если процедура использует локальные переменные, они также размещаются в стеке и адресуются с помощью BP. Так, если бы процедура NULL использовала локальную переменную, то она бы разместила ее в стеке выше ячейки «CX старый» (командой PUSH), а для доступа использовала бы конструкцию [BP-4] .

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Директивы подготовки сегментов

При программировании на ассемблере (для ПК) необходимо подготовить сегменты данных, кода и стека. Локализация сегмента производится с помощью ключевых слов SEGMENT и ENDS:

A SEGMENT; начало сегмента

A1 DB 20DUP(?)

A2 DW 8

A ENDS; конец сегмента

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Директивы подготовки сегментов

При адресации переменных всегда используется полные их адреса в формате *сегментный регистр: смещение*. Для того, чтобы указать ассемблеру, какой именно регистр нужно использовать, применяется ключевое слово

ASSUME ES:A, DS:B, CS:C

- сегмент с именем A будет сегментируется с использованием сегментного регистра ES и т.д.

Использование директивы освобождает от необходимости работать с полными адресами, т.е.

MOV AX, ES:A2

можно упростить до

MOV AX, A2

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Директивы подготовки сегментов. Общая структура программы

```
STACK SEGMENT STACK;          сегмент стека
        DB 128 DUP (?);       резервируем место для стека
STACK ENDS
DATA SEGMENT;                 сегмент данных
        <описание переменных>
DATA ENDS
CODE SEGMENT
        ASSUME CS:CODE, DS:DATA; SS:STACK
START:MOV AX,DATA
        MOV DS,AX;            загрузка системного регистра DS
        <программа>
CODE ENDS
        END START;            конец программы, точка входа
```

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Макросредства. Общие сведения

Макросредства расширяют исходный язык ассемблера благодаря тому, что трансляция осуществляется в два этапа:

- макрогенерация – приведение программы к виду, в котором нет макросредств, выполняется макрогенератором;
- ассемблирование, выполняется собственно ассемблером.

Макрогенерация и ассемблирование могут выполняться по очереди или чередоваться.

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Макросредства. Общие сведения

Макросредства:

- блоки повторения;
- макросы;
- условное ассемблирование;
- и др.

Примеры:

Блоки повторения:

REPT k

<тело>; повторяется k раз

ENDM

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com

Макросредства. Общие сведения

Макрос:

SUM MACRO X,Y; X = X+Y

MOV AX,Y

ADD X,AX

ENDM

Вызов:

SUM A,B; A = A+B

По существу макросредства ассемблера аналогичны препроцессору C, C++

PDF создан испытательной версией pdfFactory Pro www.pdffactory.com