

## Лабораторная работа № 4

### МАКРОСРЕДСТВА АССЕМБЛЕРА

#### Цель работы

1. Изучение структуры и операторов макроопределения
2. Написание макроопределения

#### МАКРООПРЕДЕЛЕНИЯ

Макроопределения представляют собой последовательность операторов на языке ассемблера (команд и псевдооператоров), которые могут несколько раз появиться в программе. Подобно процедурам макроопределения имеют имена. После того как макроопределение задано, его имя можно использовать в исходной программе вместо последовательности программ. Макроопределения Ассемблера аналогичны функциям в языках высокого уровня. Макроопределения позволяют значительно повысить производительность программирования. Особенно это заметно в операциях ввода/вывода. Текст макроопределения можно вставлять в текст программы или включать их из библиотек макроопределений.

Основные правила подготовки макроопределения:

1. Макроопределение должно выполнять только одну функцию.
2. Макроопределение должно обеспечить изоляцию данных. Это означает, что все регистры, используемые в Макроопределении, перед их использованием должны быть сохранены в стеке, а перед выходом из Макроопределения должны быть восстановлены из стека.
3. Каждое макроопределение должно иметь одну точку входа и одну точку выхода.
4. Для эффективности программирования Макроопределения должны быть простыми и легко понимаемыми.
5. Для быстродействия Макроопределения должны занимать минимум памяти.
6. Если правила 4 и 5 конфликтуют, предпочтение отдается правилу 4.

Вызывается Макроопределение с помощью макрокоманды. Только в этом случае оно выполняется. При ассемблировании макрокоманда замещается макрорасширением. Макрорасширение это макроопределение, в котором формальные параметры заменены фактическими.

#### СОСТАВ МАКРООПРЕДЕЛЕНИЙ.

Каждое макроопределение имеет три части:

1. **Заголовок** - псевдооператор **MACRO**, в поле метки которого указано *имя* макроопределения, а в поле операнда - необязательный **список формальных параметров**. В списке формальных параметров указываются переменные - входные параметры, которые можно изменять при каждом вызове макроопределения.

2. **Тело** - последовательность операторов Ассемблера (команд и псевдооператоров), которые задают действия, выполняемые макроопределением.

3. **Концевик** - псевдооператор **ENDM**, который отмечает конец макроопределения. Если Вы опустите псевдооператор **ENDM**, то Ассемблер выдаст сообщение об ошибке **End of encountered file** (обнаружен конец входного файла).

Синтаксис макроопределения следующий:

*Имя* MACRO [список формальных параметров] ;начало макроопределения  
Тело макроопределения,  
Использующее формальные параметры  
ENDM ;ключевое слово – конец макроопределения

*Имя* используется для вызова макроопределения в макрокоманде с фактическими параметрами.

*Имя* [список фактических параметров]

Ниже приведено простое макроопределение для сложения значений размером в слово:

```
ADD_WORDS MACRO TERM1, TERM2, SUM
            MOV  AX, TERM1
            ADD  AX, TERM2
            MOV  SUM, AX
        ENDM
```

Для Ассемблера безразлично, что будет указано в качестве параметров макроопределения; имена регистров, ячейки памяти или непосредственные значения (конечно, непосредственное значение нельзя использовать в качестве операнда **SUM**).

Примеры макрокоманд вызова:

**ADD\_WORDS CX, DX, BX** ;сложить содержимое регистров CX и DX, результат поместить в регистр BX

**ADD\_WORDS 134, 357, CX** ;сложить 134 и 357, результат в регистр CX

**ADD\_WORDS X, Y, RES** ;сложить значения переменных X и Y, результат поместить в переменную RES

Макрокоманды используются в тексте программы. Каждая макрокоманда в тексте программы заменяется **макрорасширением**.

Вместо первой макрокоманды:

```
+      MOV  AX, CX
+      ADD  AX, DX
+      MOV  BX, AX
```

Вместо второй макрокоманды:

```
+      MOV  AX, 134
+      ADD  AX, 357
+      MOV  CX, AX
```

Вместо третьей макрокоманды:

```
+      MOV  AX, X
+      ADD  AX, Y
+      MOV  RES, AX
```

В листинге программы макрорасширения помечаются знаком +.

## ПСЕВДООПЕРАТОРЫ МАКРОАССЕМБЛЕРА

### ПСЕВДООПЕРАТОР *LOCAL*

Если Ваше макроопределение содержит помеченные команды или псевдооператоры, то Вы должны указать Ассемблеру, чтобы он изменял метки при каждом расширении макроопределения. В противном случае Вы столкнетесь с сообщениями об ошибках **Symbol is Multi-Defined** (символ многократно определен). Псевдооператор **LOCAL** сообщает Ассемблеру, какие метки должны изменяться при каждом расширении.

Например, следующее ниже макроопределение *MSUM* выполняет сложение элементов массива *MAS* длиной *LEN*. Значение суммы помещается в переменную *MASS*. Указание метки *NEXT* в операторе *LOCAL* позволяет нам пользоваться этим макроопределением в программе более одного раза:

*MSUM MACRO MAS, LEN, MASS*

*LOCAL NEXT*

*PUSH CX* ;сохранить текущее значение *CX*

*MOV CX,LEN* ;поместить счетчик в *CX*

*MOV AX,0* ;обнулить регистр для суммы

*MOV SI,0* ;обнулить индексный регистр

*NEXT: ADD AX,MAS[SI]* ;добавить очередной элемент

*INC SI* ;увеличить значение индексного регистра

*LOOP NEXT* ;повторять, пока счетчик не обратится в 0

*MOV MASS, AX*

*POP CX* ;восстановить исходное значение *CX*

*ENDM*

Оператор **LOCAL** следует непосредственно за оператором **MACRO**.

### ЗАДАНИЕ МАКРООПРЕДЕЛЕНИЙ В ИСХОДНЫХ ПРОГРАММАХ

Существует два способа использования макроопределений: их можно задавать в начале программы или считывать в программу из отдельного файла с библиотекой макроопределений.

### Структура программы с макроопределениями

*ADD\_WORDS MACRO TERM1, TERM2, SUM*

*MOV AX, TERM1*

*ADD AX, TERM2*

*MOV SUM, AX*

*ENDM*

*sss segment stack*

*db 128 dup (?)*

*sss ends*

*dss segment*

*x1 db 13*

*x2 db 11*

*ys db ?*

*a1 db 44*

```
b1 db 33  
ab db ?  
dss ends
```

```
cseg segment  
assume cs:cseg,ds:dss,ss:sss  
push ax  
mov bx,dss  
mov ds,bx  
beg proc  
add_words x1,x2,ys  
add_words a1,b1,ab  
retf  
beg endp  
cseg ends  
end beg
```

Задание макроопределений непосредственно в тексте программы удобно, но имеет недостаток - ими можно воспользоваться только в этой программе. Чтобы они были доступны и другим программам, их надо помещать в библиотеку макроопределений.

?????????? ???????????????????

Библиотека макроопределений представляет собой файл. Его содержимое можно считывать в любую исходную программу. Тем самым все макроопределения библиотеки становятся доступными для этой программы. Чтобы использовать какое-либо из них, достаточно точно указать его имя.

### **СЧИТЫВАНИЕ БИБЛИОТЕКИ МАКРООПРЕДЕЛЕНИЙ В ПРОГРАММУ**

Для считывания библиотеки макроопределений в исходную программу надо передать Ассемблеру ее имя в операторе **INCLUDE**.

Чтобы избежать повторения считывания, оператор **INCLUDE** надо поместить в условную структуру **IF1**, например

```
IF1  
INCLUDE MACRO.LIB  
ENDIF
```

Это заставит Ассемблер считать библиотеку во время прохода 1. Но при этом указанный в операторе **INCLUDE** текст в листинг не попадет, поскольку Ассемблер выдает его во время прохода 2.

### **Удаление макроопределений**

Чтобы избежать считывания всех макроопределений, необходимо *удалить* ненужные макроопределения и тем самым освободить занятую для них часть рабочей области. Для этого непосредственно за оператором **INCLUDE** надо перечислить имена подлежащих удалению макроопределений, например:

```
INCLUDE MACRO.LIB  
PURGE MAC1, MAC2, MAC3
```

### **Макроопределение Вывод на экран в заданную позицию**

Для установки курсора в заданную позицию используется прерывание **10H**. Сначала выполняем очистку экрана с помощью функции **AH=6**.

```

Set_cursor MACRO row, col
Push ax, bx, cx, dx  Все регистры в стек
mov  ax, 0600h      ;(al=0) очистить весь экран
mov  bh, 07         ;атрибут нормальный ч./б
mov  cx, 0000       ;координаты от 00,00
mov  dx, 184fh      ; до 24,79 (весь экран)
int  10h
mov  ah, 02         ;Установка курсора
mov  bh, 00         ;страница
mov  dh, row        ;номер строки в DH
mov  dl, col        ;номер столбца в DL
int  10h
pop  ax, bx, cx, dx ;Все регистры из стека
Set_cursor ENDM

```

### Выполнение лабораторной работы

1. Напишите программу с макроопределениями на Ассемблере в соответствии с заданием.
2. Выполните компиляцию с помощью **MASM** или **TASM**.
3. Распечатайте листинг программы
4. Получите выполняемый модуль с помощью программы **LINK** или **TLINK**
4. Распечатайте результаты работы программы
5. Подготовьте отчет по лабораторной работе.

### Варианты заданий

1. Написать макроопределение ввода с клавиатуры десятичного числа и перевода его в двоичное число. Максимальное число вводимых разрядов 4, если число содержит меньшее число разрядов, то признак конца ввода - нажатие клавиши ввод. Параметр макроопределения: переменная, где находится двоичное число (слово или байт).
2. Написать макроопределение вывода двоичного числа на экран в заданную позицию. Параметры макроопределения: переменная или регистр (16 битов), где находится двоичное число и позиция вывода (номер строки, номер столбца).
3. Написать макроопределение вывода значения таймера на экран в заданную позицию. Параметры: номер строки, номер столбца.
4. Написать макроопределение вывода двоичного представления символа на экран в заданную позицию. Параметры: номер строки, номер столбца, символ.

### Контрольные вопросы

1. Каковы свойства макроопределения?
2. Преимущества и недостатки макроопределений?
3. Зачем нужен псевдооператор **LOCAL**?
4. Как выполняется вывод в заданную позицию экрана?

## Приложение 1

```
;VV.ASM
;ВВОД/ВЫВОД ЧИСЛА

SSEG SEGMENT STACK
  DB 128 DUP(?)
SSEG ENDS
;
DSEG SEGMENT
  CHISLO DW 0 ;ВЫДЕЛЯЕМ ПАМЯТЬ ПОД ЧИСЛО И ОБНУЛЯЕМ
DSEG ENDS
;
ERROR MACRO
  MOV AH,2
  MOV DL,7
  INT 21H
  MOV AH,2
  MOV DL,8
  INT 21H
ENDM
;
CSEG SEGMENT
  ASSUME CS: CSEG, DS:DSEG, SS:SSEG
  MAIN PROC ; НАЧАЛО ПРОЦЕДУРЫ MAIN(ТОЧКА ВХОДА В ПРОГРАММУ)
  MOV AX, DSEG
  MOV DS, AX
  CALL VVOD ; ВЫЗОВ ПРОЦЕДУРЫ ВВОДА
  MOV DL,0AH ; ВЫВОДИМЫЙ СИМВОЛ (ПЕРЕВОД СТРОКИ) - В DL
  MOV AH,2 ; ВЫВОД
  INT 21H ; ПО ПРЕРЫВАНИЮ 21H
  CALL VIVOD ; ВЫЗОВ ПРОЦЕДУРЫ ВЫВОДА
  MOV AX,4C00H ; СТАНДАРТНЫЙ ВЫХОД ИЗ ПРОГРАММЫ
  INT 21H ; ПО ПРЕРЫВАНИЮ 21H
  MAIN ENDP ; КОНЕЦ ПРОЦЕДУРЫ MAIN
;
VVOD PROC
  MOV DX,0AH ; ЗАНОСИМ В DX МНОЖИТЕЛЬ - 10
  MOV BX,0 ; ОБНУЛЯЕМ BX, В КОТОРОМ БУДЕТ ВВЕДЁНОЕ ЧИСЛО
L: MOV AH,1 ; ВВОД С КЛАВИАТУРЫ
  INT 21H ; ПО ПРЕРЫВАНИЮ 21H
  CMP AL,0DH ; ПРОВЕРЯЕМ ВВЕДЁННЫЙ СИМВОЛ НЕ СИМВОЛ ЛИ ЭТО КЛАВИША enter
  JE L2 ; ЕСЛИ ДА,ТО ПЕРЕХОДИМ НА МЕТКУ
  CMP AL,30H
  JB ERR1
  CMP AL,39H
  JA ERR1
  AND AL,0FH ; УДАЛЕНИЕ ПРИЗНАКА ASCII
  MOV AH,0 ; ОБНУЛЯЕМ AH
  ADD BX,AX ; СУММИРУЕМ( ДЛЯ ТОГО И ОБНУЛЯЛИ AH)
L1:MOV AH,1
  INT 21H
  CMP AL,0DH
  JE L2
  CMP AL,30H
  JB ERR2
  CMP AL,39H
  JA ERR2
  AND AL,0FH
  MOV CL,AL ; ЗАНОСИМ ВВЕДЁННОЕ ЧИСЛО В CL
```

```

MOV AX,BX      ; ЗАНОСИМ ЧИСЛО, ПОЛУЧЕННОЕ РАНЕЕ, В AX
MUL DX        ; УМНОЖАЕМ ЕГО НА 10(ПРИ ЭТОМ DX ОБНУЛИТЬСЯ)
MOV DX,0AH    ; ЗАНОСИМ В DX МНОЖИТЕЛЬ
MOV BX,AX     ; ПОЛУЧЕННЫЙ РЕЗУЛЬТАТ ВОЗВРАЩАЕМ В BX
MOV CH,0
ADD BX,CX     ; СУММИРУЕМ ПОЛУЧЕННОЕ РАНЕЕ ЧИСЛО С ВВЕДЁННЫМ
JMP L1        ; ПЕРЕХОД К ВВОДУ СЛЕД СИМВОЛА
L2:MOV CHISLO,BX ; ЗАНОСИМ ВВЕДЁННОЕ ОКОНЧАТЕЛЬНОЕ ЧИСЛО В ПЕРЕМЕННУЮ
JMP KON
ERR1:ERROR
JMP L
ERR2:ERROR
JMP L1
KON: RET      ; ВОЗВРАТ К ТОЧКЕ ВЫЗОВА
VIVOD ENDP
;
VIVOD PROC
MOV CX,0      ; ОБНУЛЯЕМ CX - КОЛИЧЕСТВО ЦИФР
MOV BX,0AH    ; ДЕЛИТЕЛЬ В BX - 10
MOV DX,0      ; ЗАНОСИМ ЧИСЛО В ДВА РЕГИСТРА
MOV AX,CHISLO ; ЧТОБЫ МОЖНО БЫЛО ВЫВОДИТЬ ЧИСЛА РАЗМЕРОМ В СЛОВО
L3:DIV BX     ; ДЕЛИМ ЧИСЛО НА 10
INC CX        ; УВЕЛИЧИВАЕМ НА 1 СЧЁТЧИК ЦИФР
OR DX,30H    ; ДОБАВЛЯЕМ К ОСТАТКУ ПРИЗНАК ASCII
PUSH DX      ; И ЗАНОСИМ ЕГО В СТЕК
MOV DX,0     ; ОБНУЛЯЕМ DX, ТАК КАК ОН УЧАСТВУЕТ В ДЕЛЕНИИ
CMP AX,9     ; СРАВНИВАЕМ ЧАСТНОЕ С 9
JA L3        ; ЕСЛИ БОЛЬШЕ 9, ТО ПЕРЕХОД НА МЕТКУ
OR AX,30H    ; ДОБАВЛЯЕМ К ЧАСТНОМУ ПРИЗНАК ASCII
INC CX
PUSH AX      ; ЗАНОСИМ ПЕРВУЮ ЦИФРУ В СТЕК
L5:POP DX    ; ИЗВЛЕКАЕМ ЦИФРУ ИЗ СТЕКА
MOV AH,2     ; ВЫВОД
INT 21H     ; ПО ПРЕРЫВАНИЮ 21H
LOOP L5     ; ПЕРЕХОД К ВЫВОДУ СЛЕД ЦИФРЫ
RET         ; ВОЗВРАТ К ТОЧКЕ ВЫЗОВА
VIVOD ENDP
CSEG ENDS
END MAIN    ; КОНЕЦ ПРОГРАММЫ С ТОЧКОЙ ВХОДА MAIN

```

**; ВЫВОД ЛИНИИ НА ЭКРАН**

```

;
.MODEL SMALL
.STACK 64
.DATA
VVC1 DB 'Vvedite cvet fona(v intervale 0-15) ',20H
VVC2 DB 'Vvedite cvet linii(v intervale 0-15) ',20H
VVKT1 DB 'Vvedite koordinati nachala linii(v intervale x: 0-639, y: 0-479) ',20H
VVKT2 DB 'Vvedite koordinati konca linii(v intervale x: 0-639, y: 0-479) ',20H
ERR DB 'Dannie vne intervala! Povtorite vvod!',20H
ERR_COLOR DB 'Cveta fona i linii sovpadaui! Povtorite vvod!',20H
X1 DW ?
X2 DW ?
Y1 DW ?
Y2 DW ?
X_INC DW ?
Y_INC DW ?
COLOR_FON DB ?
COLOR_LINE DB ?
STR DB 0

ERROR MACRO
MOV AX,1301H
MOV BX,0016H
LEA BP,ERR
MOV CX,38
MOV DH,STR
MOV DL,0
INT 10H
INC STR
ENDM
.CODE
MAIN PROC FAR
MOV AX,@DATA ;ИНИЦИАЛИЗАЦИЯ СЕГМЕНТНЫХ РЕГИСТРОВ
MOV DS,AX
MOV ES,AX
MOV AH,6
MOV AL,0
MOV BH,7
MOV CX,0
MOV DH,25
MOV DL,80
INT 10H
CALL VVOD_ATR ;ВЫЗОВ ПРОЦЕДУРЫ ВВОДА АТТРИБУТОВ
CALL VVOD_KOORD ;ВЫЗОВ ПРОЦЕДУРЫ ВВОДА КООРДИНАТ ТОЧЕК
MOV AH,00H ;УСТАНОВКА ГРАФИЧЕСКОГО РЕЖИМА
MOV AL,12H ; 640*480
INT 10H ;ПО ПРЕРЫВАНИЮ 10H
CALL VIVOD_LINE ;ВЫЗОВ ПРОЦЕДУРЫ ВЫВОДА ЛИНИИ НА ЭКРАН

```

```

MOV AH,1      ;ЗАДЕРЖКА ДО НАЖАТИЯ КЛАВИШИ
INT 21H
MOV AH,00H    ;УСТАНОВКА ИСХОДНОГО ГРАФИЧЕСКОГО РЕЖИМА
MOV AL,3
INT 10H
MOV AX,4C00H  ;СТАНДАРТНЫЙ ВЫХОД ИЗ ПРОГРАММЫ ПО
ПЕРЕРЫВАНИЮ 21H
INT 21H
MAIN ENDP
;
VVDOT_ATR PROC NEAR ;ПРОЦЕДУРА ВВОДА АТТРИБУТОВ
MOV STR,0
V1: MOV AX,1301H ;ВЫВОД СТРОКИ СИМВОЛОВ
MOV BX,0016H ;ПО КОМАНДЕ 13H
LEA BP,VVC1 ;ПЕРЕРЫВАНИЯ 10H
MOV CX,37
MOV DH,STR
MOV DL,0
INT 10H
CALL VVDOT_CHISLO ;ВЫЗОВ ПРОЦЕДУРЫ ВВОДА ЧИСЛА
INC STR
CMP BX,15
JA ERR3
MOV COLOR_FON,BL ;ЦВЕТ ФОНА
V2: MOV AX,1301H
MOV BX,0016H
LEA BP,VVC2
MOV CX,38
MOV DL,0
MOV DH,STR
INT 10H
CALL VVDOT_CHISLO
INC STR
CMP BX,15
JA ERR4
MOV COLOR_LINE,BL ;ЦВЕТ ЛИНИИ
CMP BL,COLOR_FON
JNE END1
MOV AX,1301H
MOV BX,0016H
LEA BP,ERR_COLOR
MOV CX,47
MOV DL,0
MOV DH,STR
INT 10H
INC STR
JMP V1
ERR3:ERROR
JMP V1
ERR4:ERROR
JMP V2
END1: RET ;ВОЗВРАТ К ТОЧКЕ ВЫЗОВА

```

```

VVID_ATR ENDP
;
VVID_KOORD PROC NEAR
    MOV AX,1301H
    MOV BX,0016H
    LEA BP,VVKT1
    MOV CX,63
    MOV DL,0
    MOV DH,STR
    INT 10H
    INC STR
V3: MOV AH,2
    MOV DL,0AH      ;ПЕРЕЙТИ НА СЛЕДУЮЩУЮ СТРОКУ
    INT 21H
    MOV AH,2
    MOV DL,0DH      ;И УСТАНОВИТЬ КУРСОР В ЕЁ НАЧАЛО
    INT 21H
    MOV AH,2
    MOV DL,'X'
    INT 21H
    CALL VVID_CHISLO
    CMP BX,639
    JA ERR5
    MOV X1,BX      ;КООРДИНАТА X1
V4: MOV AH,2
    MOV DL,0Ah
    INT 21H
    MOV AH,2
    MOV DL,'Y'
    INT 21H
    CALL VVID_CHISLO
    CMP BX,479
    JA ERR6
    MOV Y1,BX      ;КООРДИНАТА Y1
    ADD STR,2
    MOV AX,1301H
    MOV BX,0016H
    LEA BP,VVKT2
    MOV CX,63
    MOV DL,0
    MOV DH,STR
    INT 10H
V5: MOV AH,2
    MOV DL,0AH
    INT 21H
    MOV AH,2
    MOV DL,0DH
    INT 21H
    MOV AH,2
    MOV DL,'X'
    INT 21H
    INC STR

```

```

CALL VVOD_CHISLO
CMP BX,639
JA ERR7
MOV X2,BX      ;КООРДИНАТА X2
V6: MOV AH,2
MOV DL,0Ah
INT 21H
MOV AH,2
MOV DL,'Y'
INT 21H
CALL VVOD_CHISLO
CMP BX,479
JA ERR8
MOV Y2,BX     ;КООРДИНАТА Y2
JMP END2
ERR5:INC STR
ERROR
JMP V3
ERR6:INC STR
INC STR
ERROR
DEC STR
MOV AH,2
MOV DL,0
MOV DH,STR
MOV BH,0
INT 10H
JMP V4
ERR7:INC STR
ERROR
JMP V5
ERR8:INC STR
INC STR
ERROR
DEC STR
MOV AH,2
MOV DL,0
MOV DH,STR
MOV BH,0
INT 10H
JMP V6
END2: RET      ;ВОЗВРАТ К ТОЧКЕ ВЫЗОВА
VVOD_KOORD ENDP
;
VVOD_CHISLO PROC NEAR
MOV DX,0AH    ; ЗАНОСИМ В DX МНОЖИТЕЛЬ - 10
MOV BX,0      ; ОБНУЛЯЕМ BX, В КОТОРОМ БУДЕТ ВВЕДЁНОЕ ЧИСЛО
BEGIN:MOV AH,1 ; ВВОД С КЛАВИАТУРЫ
INT 21H      ; ПО ПРЕРЫВАНИЮ 21H
CMP AL,0DH   ; ПРОВЕРЯЕМ ВВЕДЁННЫЙ СИМВОЛ НА СОВПАДЕНИЕ С
ВОЗРАТОМ КАРЕТКИ
JE L2        ; ЕСЛИ СОВПАЛ, ТО ПЕРЕХОДИМ НА МЕТКУ

```

```

CMP AL,30H
JB ERR1
CMP AL,39H
JA ERR1
AND AL,0FH      ; УДАЛЕНИЕ ПРИЗНАКА ASCII
MOV AH,0        ; ОБНУЛЯЕМ AH
ADD BX,AX       ; СУММИРУЕМ( ДЛЯ ТОГО И ОБНУЛЯЛИ AH)
L1:MOV AH,1
INT 21H
CMP AL,0DH
JE L2
CMP AL,30H
JB ERR2
CMP AL,39H
JA ERR2
AND AL,0FH
MOV CL,AL       ; ЗАНОСИМ ВВЕДЁННОЕ ЧИСЛО В CL
MOV AX,BX       ; ЗАНОСИМ ЧИСЛО, ПОЛУЧЕННОЕ РАНЕЕ, В AX
MUL DX          ; УМНОЖАЕМ ЕГО НА 10(ПРИ ЭТОМ DX ОБНУЛИТЬСЯ)
MOV DX,0AH      ; ЗАНОСИМ В DX МНОЖИТЕЛЬ
MOV BX,AX       ; ПОЛУЧЕННЫЙ РЕЗУЛЬТАТ ВОЗВРАЩАЕМ В BX
MOV CH,0
ADD BX,CX       ; СУММИРУЕМ ПОЛУЧЕННОЕ РАНЕЕ ЧИСЛО С ВВЕДЁННЫМ
JMP L1          ; ПЕРЕХОД К ВВОДУ СЛЕД СИМВОЛА
ERR1:MOV AH,2
MOV DL,7H
INT 21H
MOV AH,2
MOV DL,8H
INT 21H
JMP BEGIN
ERR2:MOV AH,2
MOV DL,7H
INT 21H
MOV AH,2
MOV DL,8H
INT 21H
JMP L1
L2:RET          ; ВОЗВРАТ К ТОЧКЕ ВЫЗОВА
VVDOD_CHISLO ENDP
;
VIVOD_LINE PROC NEAR ;ВЫВОД ЛИНИИ НА ЭКРАН ПРИ ПОМОЩИ
АЛГОРИТМА БРЕЗЕНХАМА
MOV AH,0BH      ;УСТАНОВИТЬ ЦВЕТ ФОНА
MOV BH,00       ;ПО ПРЕРЫВАНИЮ 10H
MOV BL,COLOR_FON
INT 10H
MOV CX,X2
MOV DX,Y2
MOV AX,CX
SUB AX,X1
JNS DX_POS

```

```

    NEG AX
    MOV WORD PTR X_INC,1
    JMP DX_NEG
DX_POS:MOV WORD PTR X_INC,-1
DX_NEG:MOV BX,DX
    SUB BX,Y1
    JNS DY_POS
    NEG BX
    MOV WORD PTR Y_INC,1
    JMP DY_NEG
DY_POS:MOV WORD PTR Y_INC,-1
DY_NEG:SHL AX,1
    SHL BX,1
    PUSH AX
    PUSH BX
    MOV AL,COLOR_LINE
    MOV AH,0CH
    MOV BH,0
    INT 10H
    POP BX
    POP AX
    CMP AX,BX
    JNA DX_DY
    MOV DI,AX
    SHR DI,1
    NEG DI
    ADD DI,BX ;DI=2*DY-DX
BEGIN1:CMP CX,X1
    JE EXIT_LINE
    CMP DI,0
    JL L3
    ADD DX,Y_INC
    SUB DI,AX
L3: ADD CX,X_INC
    ADD DI,BX
    PUSH AX
    PUSH BX
    MOV AL,COLOR_LINE
    MOV AH,0CH
    MOV BH,0
    INT 10H
    POP BX
    POP AX
    JMP BEGIN1
DX_DY: MOV DI,BX
    SHR DI,1
    NEG DI
    ADD DI,AX ;DI=2*DX-DY
BEGIN2:CMP DX,Y1
    JE EXIT_LINE
    CMP DI,0
    JL L4

```

```
    ADD CX,X_INC
    SUB DI,BX
L4:  ADD DX,Y_INC
    ADD DI,AX
    PUSH AX
    PUSH BX
    MOV AL,COLOR_LINE
    MOV AH,0CH
    MOV BH,0
    INT 10H
    POP BX
    POP AX
    JMP BEGIN2
EXIT_LINE:
RET
VIVOD_LINE ENDP
END MAIN
```