МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Организация ЭВМ»

Тема: Представление и обработка целых чисел. Организация ветвящихся процессов

Студентка гр. 0382 _____ Андрющенко К.С

Преподаватель _____ Ефремов М.А.

Санкт-Петербург

2021

Цель работы.

Изучение организации ветвления в языке ассемблера. Изучить операторы сравнения, условного и безусловного перехода. Научится описывать функции с условием используя ветвление.

Задание.

Разработать на языке Ассемблера программу, которая по заданным целочисленным значениям параметров a, b, i, k вычисляет:

a) значения функций i1 = f1(a,b,i) и i2 = f2(a,b,i);

b) значения результирующей функции res = f3(i1,i2,k), где вид функций f1 и f2 определяется из табл. 2, а функции f3 - из табл.3 по цифрам шифра индивидуального задания (n1,n2,n3), приведенным в табл.4. Значения a, b, i, k являются исходными данными, которые должны выбираться студентом самостоятельно и задаваться в процессе исполнения программы в режиме отладки. При этом следует рассмотреть всевозможные комбинации параметров a, b и k, позволяющие проверить различные маршруты выполнения программы, а также различные знаки параметров a и b.

Вариант №2: 1.3.2

```
        / 15-2*i , при a>b
f1 = <
        \ 3*i+4 , при a<=b

        / 7 - 4*i , при a>b
f3 = <
        \ 8 -6*i , при a<=b

        / max(i1,10-i2), при k<0
f2 = <
         \ |i1 - i2| , при k>=0
```

Замечания:

1) при разработке программы нельзя использовать фрагменты, представленные на ЯВУ, в частности, для ввода-вывода данных. Исходные данные должны вводиться, а результаты контролироваться в режиме отладки;
2) при вычислении функций f1 и f2 вместо операции умножения следует использовать арифметический сдвиг и, возможно, сложение;

3) при вычислении функций f1 и f2 нельзя использовать процедуры;

4) при разработке программы следует минимизировать длину кода, для чего, если надо, следует преобразовать исходные выражения для вычисления функций.

Выполнение работы.

Для организации условных операторов будем последовательно использовать cmp и команды условного.

cmp – команда процессора для сравнения чисел. Реально она изменяет значения флагов.

Команда условного перехода анализирует значение нужных флагов, и в случае если они установлены, выполняют переход по указанному адресу.

Рассмотрим используемые команды условного перехода.

JG/JNLE (SF = OF и ZF = 0) - переход если больше.

JLE/JNG (SF ≠ OF или ZF = 1) - переход если меньше или равно.

JGE (SF = OF) - переход если больше или равно.

JL (SF ≠ OF) - переход если меньше.

JMP – безусловный переход к указанной метке.

Учитывая замечание 2, рассмотрим простейшую процедуру оптимизации умножения. Чтобы умножить число на степень двойки, его достаточно просто сдвинуть влево на необходимое число двоичных (битовых) позиций.

Исходный код программы см. ПРИЛОЖЕНИЕ А.

Файл диагностических сообщений см.

ПРИЛОЖЕНИЕ В.

Тестирование.

Значения переменных a, b, i, k для 4 тестов см. Таблица 1.

Таблица 1

№ теста	a	b	k	i
1	2	3	-1	1
2			1	
3	3	2	-1	
4			1	

Значение переменных i1, i2 и res (помещаем в регистр общего назначения dx в конце выполнения всех необходимых арифметических для

удобного наблюдения за результатом) см. Таблица 2

Таблица 2

№ теста	I1	I2	res
1	7	2	8
2	7	2	5
3	13	3	13
4	13	3	10

Значение регистров после выполнения программы под управлением отладчика.

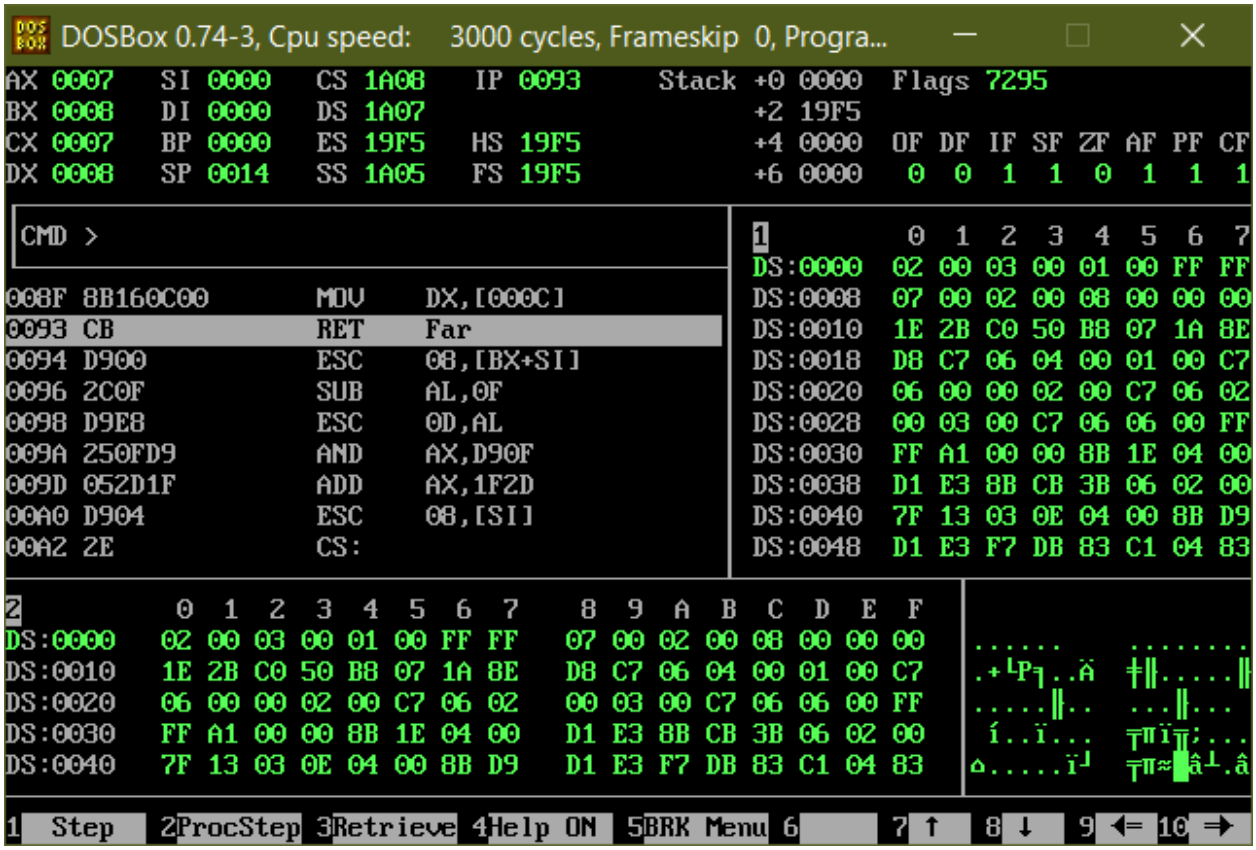

Рисунок 1 – тест №1

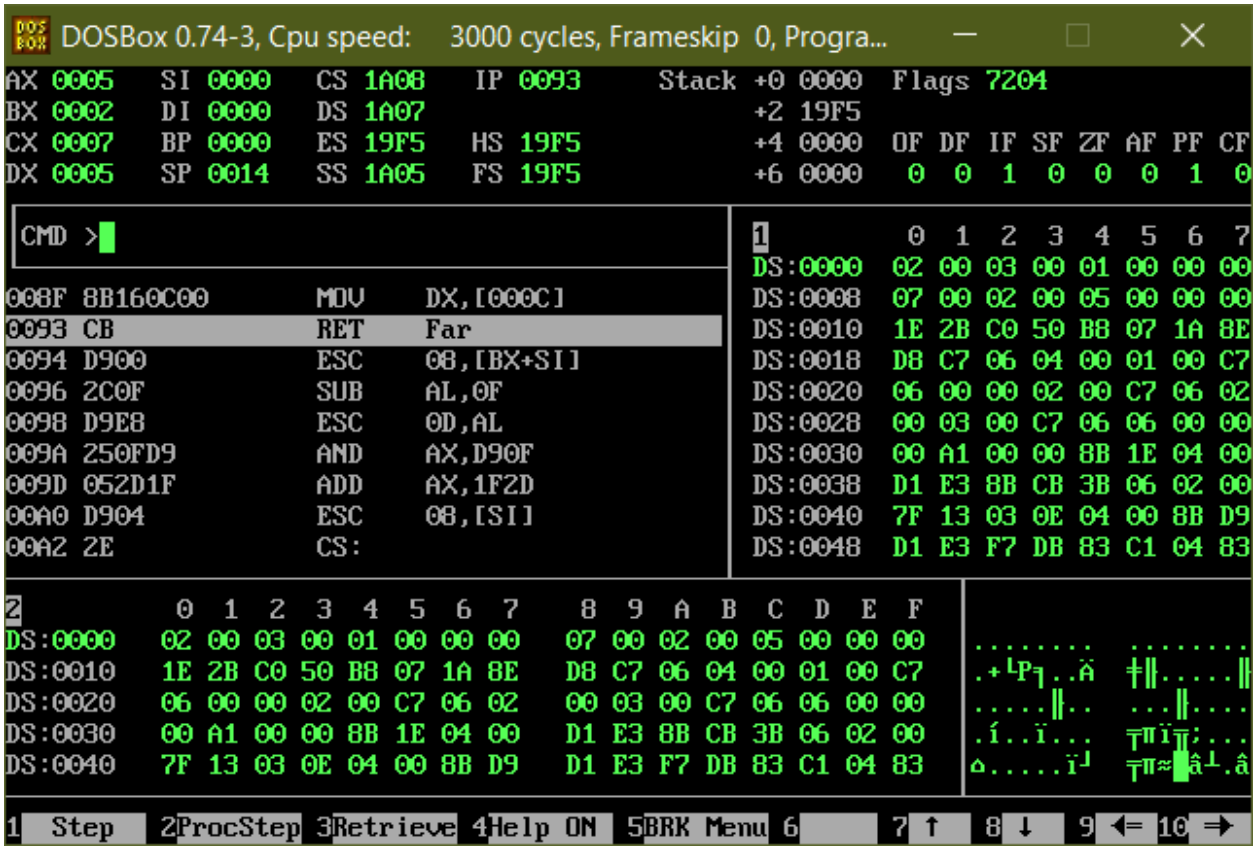

Рисунок 2 – тест №2

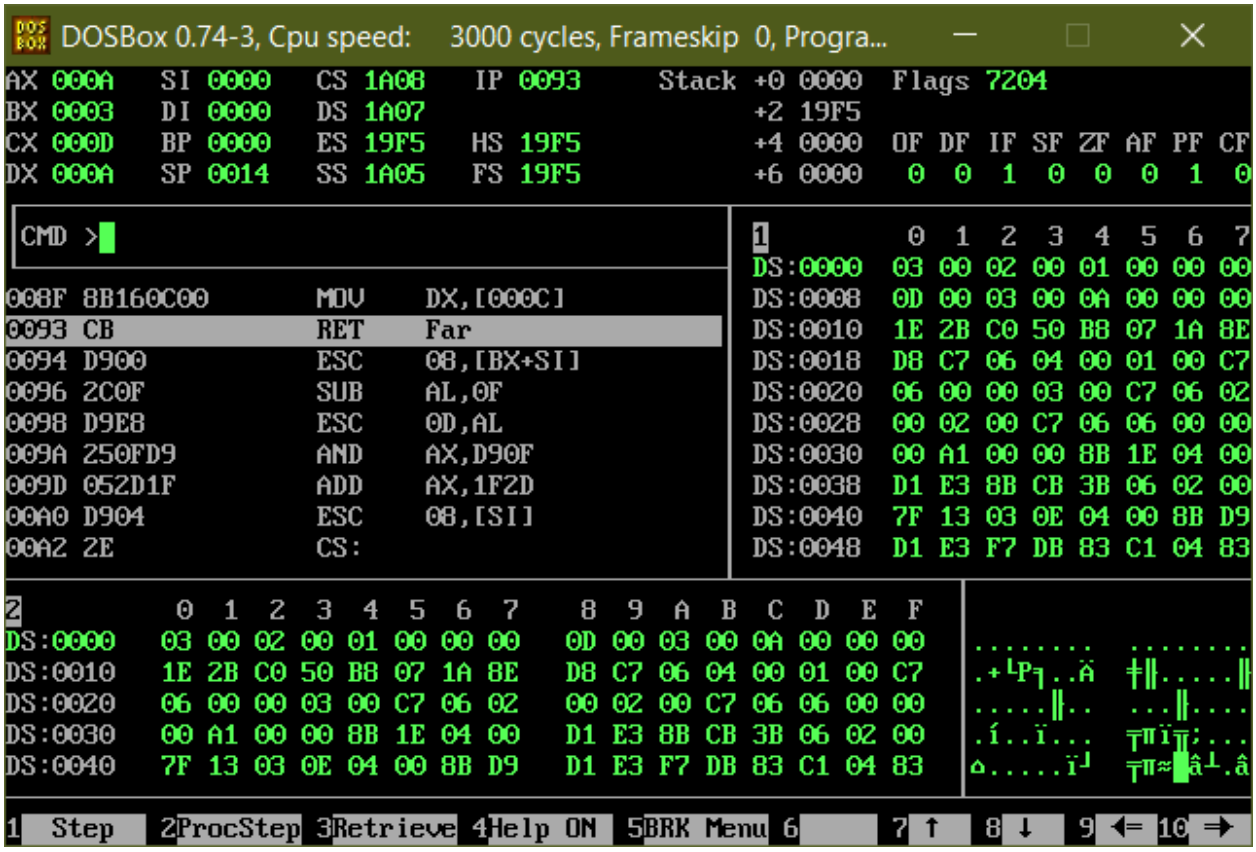

Рисунок 3 – тест №3

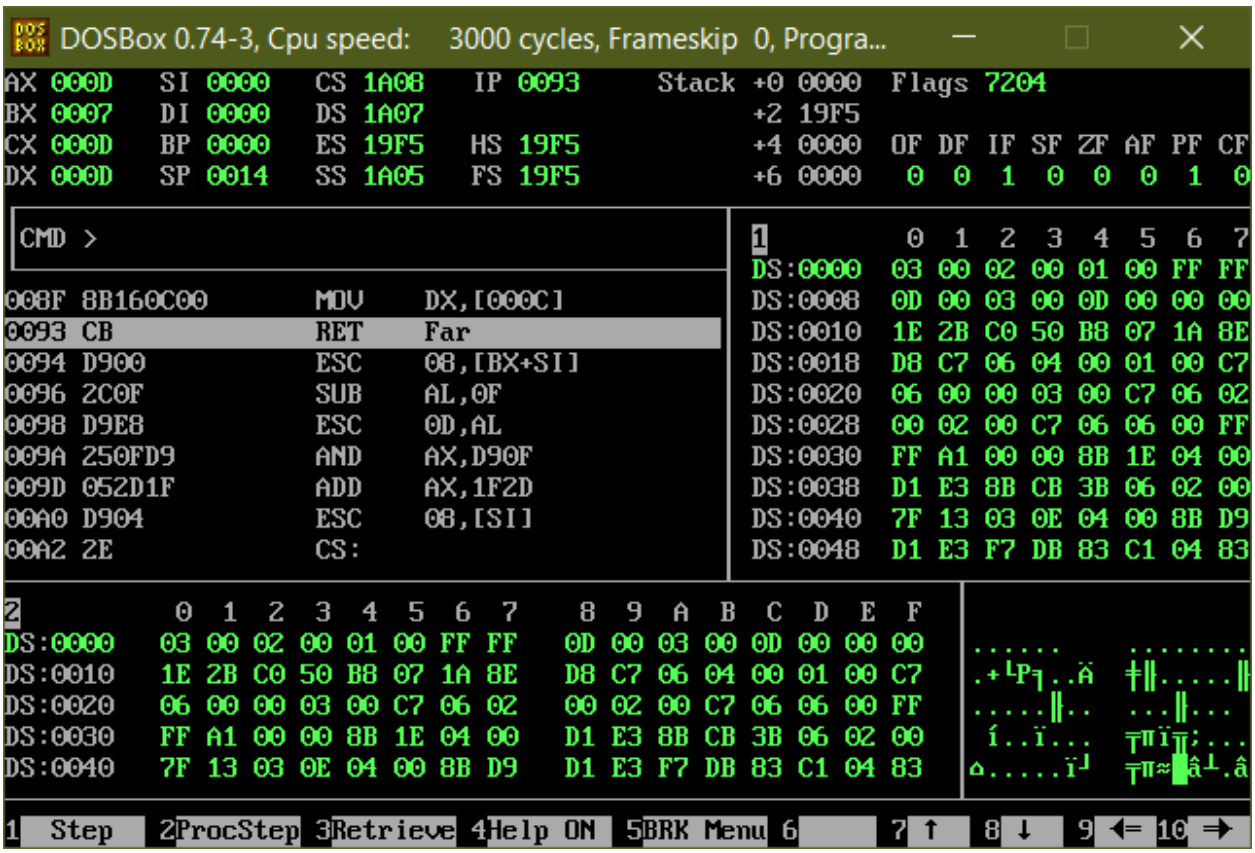

Рисунок 4 – тест №4

Вывод.

В результате работы была написана программа, реализующая вычисление значения функции по некоторым параметрам. Были изучены способы организации ветвления в языке Ассемблер, условные и безусловные переходы.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Файл LB3_CODE.ASM

```
; Стек программы
AStack SEGMENT STACK
    DW 12 DUP(?)
AStack ENDS
; Данные программы
DATA SEGMENT
  a   DW 0 ;определяет переменную размером в слово.
  b   DW 0
  i   DW 0
  k   DW 0
  i1  DW 0
  i2  DW 0
  res DW 0
DATA ENDS
; Код программы
CODE SEGMENT
  ASSUME CS:CODE, DS:DATA, SS:AStack
; Головная процедура
Main PROC FAR
  push DS
  sub AX,AX
  push AX
  mov AX,DATA
  mov DS,AX
  mov i, 1
  mov a, 2
  mov b, 3
  mov k, -1
f1_and_f2:
  mov ax, a
  mov bx, i ; bx = i
  shl bx, 1 ; bx = 2i
  mov cx, bx; cx = 2i
  cmp ax, b
  jg a_more_b ; если a > b выполним блок a_more_b
  add cx, i ; cx = 3i
  mov bx, cx ; bx = 3i
  shl bx, 1 ; bx = 6i
  neg bx ; bx = -6i
  add cx, 4 ; cx = 3i + 4
  add bx, 8 ; bx = 8 - 6i
  jmp f1_f2_result ; безусловный переход к сохранению результата
f1_result
a_more_b:
  shl bx, 1 ; bx = 4i
  neg cx ; cx = -2i
  neg bx ; bx = -4i
  add cx, 15 ; cx = -2i + 15
  add bx, 7 ; bx = 7 - 4i
f1_f2_result:
  mov i1, cx ; i1 = f1(i)
  mov i2, bx ; i2 = f2(i)
```

```
f3:
  mov ax, i1
  mov bx, i2
  cmp k, 0
  jl f3_k_less_0 ; если k < 0 переход на метку f3_k_less_0
  cmp ax, bx ; i1 >= i2
  jge i1_more_i2
  sub bx, ax
  mov res, bx
  jmp f3_result
i1_more_i2:
  sub ax, bx ; i1 = i1 - i2
  mov res, ax
  jmp f3_result
f3_k_less_0:
  neg bx
  add bx, 10
  cmp ax, bx
  jge max_i1 ; если i1 >= i2 переход к метке max_i1
  mov res, bx
  jmp f3_result
max_i1:
  mov res, ax
f3_result:
  mov dx, res
  ret
Main ENDP
CODE ENDS
END Main
```

ПРИЛОЖЕНИЕ В
ДИАГНОСТИЧЕСКИЕ СООБЩЕНИЯ

Файл LB3_CODE.LST

```
Microsoft (R) Macro Assembler Version 5.10                  11/4/21
01:26:22
                                                             Page
1-1


                        ; Стек программы
0000                    AStack SEGMENT STACK
0000  000C[                     DW 12 DUP(?)
       ????
                ]

0018                    AStack ENDS
                        ; Данные программы
0000                    DATA SEGMENT
0000  0000                      a   DW 0 ;определяет перемен
                        ную размером в слово.
0002  0000                      b   DW 0
0004  0000                      i   DW 0
0006  0000                      k   DW 0
0008  0000                      i1  DW 0
000A  0000                      i2  DW 0
000C  0000                      res DW 0
000E                    DATA ENDS
                        ; Код программы
0000                    CODE SEGMENT
                          ASSUME CS:CODE, DS:DATA, SS:AStack
                        ; Головная процедура
0000                    Main PROC FAR
0000  1E                  push DS
0001  2B C0                    sub AX,AX
0003  50                  push AX
0004  B8 ---- R           mov AX,DATA
0007  8E D8                    mov DS,AX
0009  C7 06 0004 R 0001        mov i, 1 ; задаем значения пе
                        ременных для тестировани
                        и отладки
000F  C7 06 0000 R 0002        mov a, 2
0015  C7 06 0002 R 0003        mov b, 3
001B  C7 06 0006 R FFFF        mov k, -1
0021                    f1_and_f2:
0021  A1 0000 R           mov ax, a
0024  8B 1E 0004 R             mov bx, i ; bx = i
0028  D1 E3                    shl bx, 1 ; bx = 2i
002A  8B CB                    mov cx, bx; cx = 2i
002C  3B 06 0002 R             cmp ax, b
0030  7F 13                    jg a_more_b ; если a > b выполним
                         блок a_more_b
0032  03 0E 0004 R             add cx, i ; cx = 3i
0036  8B D9                    mov bx, cx ; bx = 3i
0038  D1 E3                    shl bx, 1 ; bx = 6i
003A  F7 DB                    neg bx ; bx = -6i
003C  83 C1 04                 add cx, 4 ; cx = 3i + 4
```

```
003F  83 C3 08            add bx, 8 ; bx = 8 - 6i
0042  EB 0D 90            jmp f1_f2_result ; безусловный п
                        реход к сохранению резул
                        тата f1_result
0045                    a_more_b:
0045  D1 E3               shl bx, 1 ; bx = 4i
0047  F7 D9               neg cx ; cx = -2i
```

```
0049  F7 DB                 neg bx ; bx = -4i
004B  83 C1 0F              add cx, 15 ; cx = -2i + 15
004E  83 C3 07              add bx, 7 ; bx = 7 - 4i
0051                  f1_f2_result:
0051  89 0E 0008 R          mov i1, cx ; i1 = f1(i)
0055  89 1E 000A R          mov i2, bx ; i2 = f2(i)
0059                  f3:
0059  A1 0008 R       mov ax, i1
005C  8B 1E 000A R          mov bx, i2
0060  83 3E 0006 R 00             cmp k, 0
0065  7C 15                 jl f3_k_less_0 ; если k < 0 перехо
                    на метку f3_k_less_0
0067  3B C3                 cmp ax, bx ; i1 >= i2
0069  7D 09                 jge i1_more_i2
006B  2B D8                 sub bx, ax
006D  89 1E 000C R          mov res, bx
0071  EB 1C 90              jmp f3_result
0074                  i1_more_i2:
0074  2B C3                 sub ax, bx ; i1 = i1 - i2
0076  A3 000C R       mov res, ax
0079  EB 14 90              jmp f3_result
007C                  f3_k_less_0:
007C  F7 DB                 neg bx
007E  83 C3 0A              add bx, 10
0081  3B C3                 cmp ax, bx
0083  7D 07                 jge max_i1 ; если i1 >= i2 переход
                     к метке max_i1
0085  89 1E 000C R          mov res, bx
0089  EB 04 90              jmp f3_result
008C                  max_i1:
008C  A3 000C R       mov res, ax
008F                  f3_result:
008F  8B 16 000C R          mov dx, res
0093  CB              ret
0094                  Main ENDP
0094                  CODE ENDS
                      END Main
```

Segments and Groups:

N a m e	Length	Align	Combine	Class
ASTACK	0018	PARA	STACK	
CODE	0094	PARA	NONE	
DATA	000E	PARA	NONE	

Symbols:

N a m e	Type	Value	Attr
A	L WORD	0000	DATA
A_MORE_B	L NEAR	0045	CODE
B	L WORD	0002	DATA
F1_AND_F2	L NEAR	0021	CODE
F1_F2_RESULT	L NEAR	0051	CODE
F3	L NEAR	0059	CODE
F3_K_LESS_0	L NEAR	007C	CODE
F3_RESULT	L NEAR	008F	CODE
I	L WORD	0004	DATA
I1	L WORD	0008	DATA
I1_MORE_I2	L NEAR	0074	CODE
I2	L WORD	000A	DATA
K	L WORD	0006	DATA
MAIN	F PROC	0000	CODE Length = 0094
MAX_I1	L NEAR	008C	CODE
RES	L WORD	000C	DATA
@CPU	TEXT	0101h	
@FILENAME	TEXT	LB3_CODE	
@VERSION	TEXT	510	

80 Source Lines
80 Total Lines
24 Symbols

47954 + 461353 Bytes symbol space free

0 Warning Errors
0 Severe Errors