МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпропетровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Технічна Кібернетика»

**Лабораторна робота №4**

**з дисципліни «Архітектура комп’ютера»**

**на тему: «**Реалізація розгалужень мовою Асемблер**»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Нежуміра О.И.

Дніпро, 2019

**Тема.** Реалізація розгалужень мовою Асемблер.

**Мета.** Набути практичних навичок розробки програм нелінійної структури мовою Асемблер.

# **Постановка задачі згідно з загальним та індивідуальним завданнями.**

Таблиця 1

|  |  |
| --- | --- |
| 8 | Визначити чи потрапляє точка с заданими координатами в третю або четверту чверть. Відповідь – повідомлення. |

# **Алгоритм розв’язання задачі індивідуального завдання.**

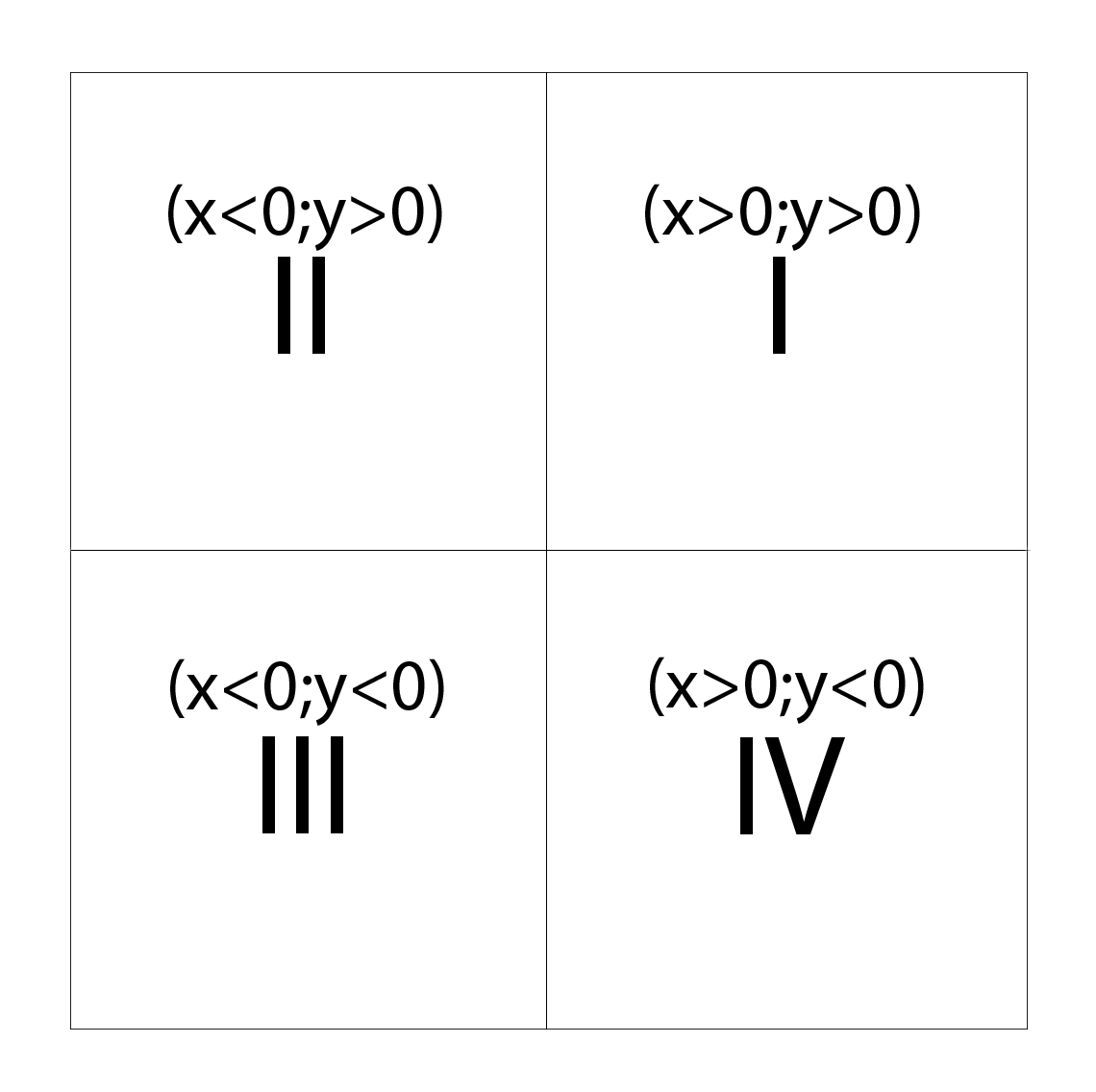


Рисунок 1

Якщо y<0, то точка належить до третьої або четвертої координатних чвертей, якщо ні – не належить. Осі не належать до чвертей тож якщо x=0, або y=0, то точка не належить до жодної чверті.

Діаграмма N-S:

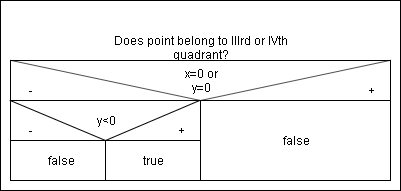


Рисунок 2

# **Лістинг програми.**

Turbo Assembler Version 3.0 11/04/19 19:20:16 Page 1

4v7.asm

1 0000 sseg segment stack 'stack'

2 0000 80\*(??) db 128 dup(?)

3 0080 sseg ends

4 ;-----------------------------------------------------------------------------------------------

5 ---

6 0000 dseg segment 'data'

7 0000 2D 24 msg\_neg db "-", '$'

8 0002 28 78 3B 79 29 20 63+ msg\_xy db "(x;y) coordinats for the point are:", 0Dh, 0Ah, '$'

9 6F 6F 72 64 69 6E 61+

10 74 73 20 66 6F 72 20+

11 74 68 65 20 70 6F 69+

12 6E 74 20 61 72 65 3A+

13 0D 0A 24

14 0028 78 3D 24 msg\_x db "x=", '$'

15 002B 79 3D 24 msg\_y db "y=", '$'

16 002E 0D 0A 24 msg\_endl db 0Dh, 0Ah, '$'

17 0031 00 x db 0

18 0032 FF y db -1

19 0033 54 72 75 65 2C 20 70+ msg\_t db "True, point belongs to IIIrd or IVth Quadrant", 0Dh, 0Ah, '$'

20 6F 69 6E 74 20 62 65+

21 6C 6F 6E 67 73 20 74+

22 6F 20 49 49 49 72 64+

23 20 6F 72 20 49 56 74+

24 68 20 51 75 61 64 72+

25 61 6E 74 0D 0A 24

26 0063 46 61 6C 73 65 2C 20+ msg\_f db "False, point doesn't belong to IIIrd or IVth Quadrant", 0Dh, 0Ah, '$'

27 70 6F 69 6E 74 20 64+

28 6F 65 73 6E 27 74 20+

29 62 65 6C 6F 6E 67 20+

30 74 6F 20 49 49 49 72+

31 64 20 6F 72 20 49 56+

32 74 68 20 51 75 61 64+

33 72 61 6E 74 0D 0A 24

34 009B dseg ends

35 ;-----------------------------------------------------------------------------------------------

36 ---

37 0000 cseg segment 'code'

38 assume cs:cseg, ds:dseg, ss:sseg

39

40 0000 B8 0000s start: mov ax, dseg

41 0003 8E D8 mov ds, ax

42 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

43 ++++

44 0005 BA 0002r out\_xy: lea dx, msg\_xy

45 0008 B4 09 mov ah, 09h

46 000A CD 21 int 21h

47 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

48 ++++

49 000C BA 0028r out\_x\_msg: lea dx, msg\_x

50 000F B4 09 mov ah, 09h

51 0011 CD 21 int 21h

52

53 0013 80 3E 0031r 00 process\_x: cmp x, 0

54 0018 7D 07 jge out\_x\_pos

55 001A F6 1E 0031r neg x;making -x +x to output it with ascii

56 001E EB 16 90 jmp out\_x\_neg

57

Turbo Assembler Version 3.0 11/04/19 19:20:16 Page 2

4v7.asm

58 0021 8A 16 0031r out\_x\_pos: mov dl, x

59 0025 80 C2 30 add dl, 30h

60 0028 B4 02 mov ah, 02h

61 002A CD 21 int 21h

62

63 002C BA 002Er lea dx, msg\_endl

64 002F B4 09 mov ah, 09h

65 0031 CD 21 int 21h

66 0033 EB 1E 90 jmp out\_y\_msg

67

68 0036 BA 0000r out\_x\_neg: lea dx, msg\_neg

69 0039 B4 09 mov ah, 09h

70 003B CD 21 int 21h

71

72 003D 8A 16 0031r mov dl, x

73 0041 80 C2 30 add dl, 30h

74 0044 B4 02 mov ah, 02h

75 0046 CD 21 int 21h

76

77 0048 BA 002Er lea dx, msg\_endl

78 004B B4 09 mov ah, 09h

79 004D CD 21 int 21h

80

81 004F F6 1E 0031r neg x;making +x back -x, for future processing

82 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

83 ++++

84 0053 BA 002Br out\_y\_msg: lea dx, msg\_y

85 0056 B4 09 mov ah, 09h

86 0058 CD 21 int 21h

87

88 005A 80 3E 0032r 00 process\_y: cmp y, 0

89 005F 7D 07 jge out\_y\_pos

90 0061 F6 1E 0032r neg y;making -y +y to output it with ascii

91 0065 EB 16 90 jmp out\_y\_neg

92

93 0068 8A 16 0032r out\_y\_pos: mov dl, y

94 006C 80 C2 30 add dl, 30h

95 006F B4 02 mov ah, 02h

96 0071 CD 21 int 21h

97

98 0073 BA 002Er lea dx, msg\_endl

99 0076 B4 09 mov ah, 09h

100 0078 CD 21 int 21h

101 007A EB 1E 90 jmp if\_xz\_yz

102

103 007D BA 0000r out\_y\_neg: lea dx, msg\_neg

104 0080 B4 09 mov ah, 09h

105 0082 CD 21 int 21h

106

107 0084 8A 16 0032r mov dl, y

108 0088 80 C2 30 add dl, 30h

109 008B B4 02 mov ah, 02h

110 008D CD 21 int 21h

111

112 008F BA 002Er lea dx, msg\_endl

113 0092 B4 09 mov ah, 09h

114 0094 CD 21 int 21h

Turbo Assembler Version 3.0 11/04/19 19:20:16 Page 3

4v7.asm

115

116 0096 F6 1E 0032r neg y;making +y back -y, for future processing

117 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

118 ++++

119 009A 80 3E 0031r 00 if\_xz\_yz: cmp x, 0

120 009F 74 18 je out\_f

121 00A1 80 3E 0032r 00 cmp y,0

122 00A6 74 11 je out\_f; testing x and y for =0

123

124 00A8 80 3E 0032r 00 if\_yn: cmp y, 0

125 00AD 7F 0A jg out\_f; testing y for <0

126

127 00AF BA 0033r out\_t: lea dx, msg\_t

128 00B2 B4 09 mov ah, 09h

129 00B4 CD 21 int 21h

130 00B6 EB 08 90 jmp end\_

131

132 00B9 BA 0063r out\_f: lea dx, msg\_f

133 00BC B4 09 mov ah, 09h

134 00BE CD 21 int 21h

135 00C0 end\_:

136 ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

137 ++++

138 00C0 B4 4C mov ah, 4Ch

139 00C2 CD 21 int 21h

140 00C4 cseg ends

141 end start

Turbo Assembler Version 3.0 11/04/19 19:20:16 Page 4

Symbol Table

Symbol Name Type Value

??DATE Text "11/04/19"

??FILENAME Text "4v7 "

??TIME Text "19:20:16"

??VERSION Number 0300

@CPU Text 0101H

@CURSEG Text CSEG

@FILENAME Text 4V7

@WORDSIZE Text 2

END\_ Near CSEG:00C0

IF\_XZ\_YZ Near CSEG:009A

IF\_YN Near CSEG:00A8

MSG\_ENDL Byte DSEG:002E

MSG\_F Byte DSEG:0063

MSG\_NEG Byte DSEG:0000

MSG\_T Byte DSEG:0033

MSG\_X Byte DSEG:0028

MSG\_XY Byte DSEG:0002

MSG\_Y Byte DSEG:002B

OUT\_F Near CSEG:00B9

OUT\_T Near CSEG:00AF

OUT\_XY Near CSEG:0005

OUT\_X\_MSG Near CSEG:000C

OUT\_X\_NEG Near CSEG:0036

OUT\_X\_POS Near CSEG:0021

OUT\_Y\_MSG Near CSEG:0053

OUT\_Y\_NEG Near CSEG:007D

OUT\_Y\_POS Near CSEG:0068

PROCESS\_X Near CSEG:0013

PROCESS\_Y Near CSEG:005A

START Near CSEG:0000

X Byte DSEG:0031

Y Byte DSEG:0032

Groups & Segments Bit Size Align Combine Class

CSEG 16 00C4 Para none CODE

DSEG 16 009B Para none DATA

SSEG 16 0080 Para Stack STACK

# **Тести.**

Таблиця 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Назва | Вхідні дані | | Очікуваний результат |
|  |  | x | y | Повідомлення |
| [1](#t1) | I(x>0;y>0) | 1 | 1 | “False, point doesn’t belong to IIIrd or IVth Quadrant” |
| [2](#t2) | II(x<0;y>0) | -1 | 1 | “False, point doesn’t belong to IIIrd or IVth Quadrant” |
| [3](#t3) | III(x<0;y<0) | -1 | -1 | “True, point belongs to IIIrd or IVth Quadrant” |
| [4](#t4) | IV(x>0;y<0) | 1 | -1 | “True, point belongs to IIIrd or Ivth Quadrant” |
| [5](#t5) | x=0 | 0 | 1 | “False, point doesn’t belong to IIIrd or Ivth Quadrant” |
| [6](#t6) | y=0 | 1 | 0 | “False, point doesn’t belong to IIIrd or Ivth Quadrant” |
| [7](#t7) | x=0;y=0 | 0 | 0 | “False, point doesn’t belong to IIIrd or Ivth Quadrant” |
| [8](#t8) | x=0;y<0 | 0 | -1 | “False, point doesn’t belong to IIIrd or Ivth Quadrant” |

# **Результати виконання програми та їх аналіз.**

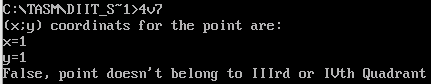


Рисунок 3

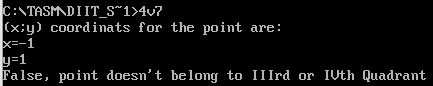


Рисунок 4

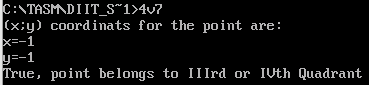


Рисунок 5

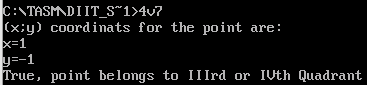


Рисунок 6

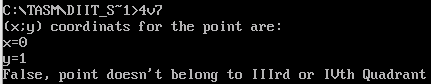


Рисунок 7

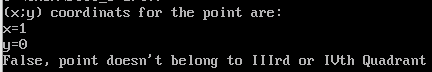


Рисунок 8

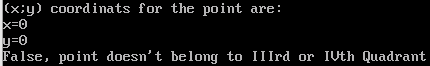


Рисунок 9

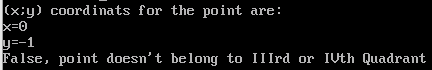


Рисунок 10

Для зручності я зробив, щоб програма також виводила на єкран значення x та y. Результати програми співпадають з очікуваними.

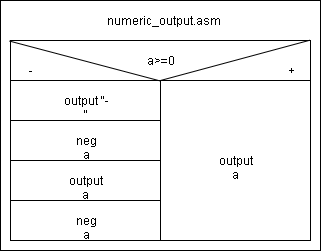


Рисунок 11

# **Висновки щодо можливості альтернативних реалізацій керуючих конструкцій у мові асемблер.**

Стандартна бібліотека інструкцій asm 8086 налічує достатньо багато команд умовного переходу, усі вони є варіаціями команді безумовного переходу jmp. Вони перевіряють флаги FR задані варіацією. Більшість з них перевіряє відношення одного числа до іншого. Усі вони починаються з символа j, а після нього йде умова. Усі умови – скорочення від англійської, наприклад:

A – above (>)

B – below (<)

L – less (<)

G – greater (>)

E – equal (=)

N – not (!)

Z – zero (ZF = 1)

A, B – для чисел без знаку (>= 0).

G, L – та сама функція для чисел за знаком (для усіх).

З цих 7 можна скласти майже усі умови, деякі з яких повторюються, наприклад JNAE теж саме, що й JB.

За бажанням ціми командами замінити цикли.