ІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



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

Кафедра «Технічна Кібернетика»

**Лабораторна робота №5  
з дисципліни «Архітектура комп’ютера»**

**на тему: «**Реалізація циклічних алгоритмів мовою Асемблера**»**

Виконав:

студент гр.ПЗ1911

Сафонов Д.Є.

Прийняла:

Нежуміра О.И.

Дніпро, 2019

**Тема.** Реалізація циклічних алгоритмів мовою Асемблера.

**Мета.** Набути практичних навичок розробки програм циклічної структури.

Таблиця 1

|  |  |
| --- | --- |
| 8 | Задано натуральне число, діапазон значень – слово. Визначити кількість цифр кратних 3 у записі числа. |

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

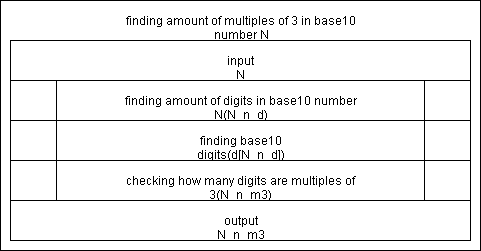


Рисунок 1

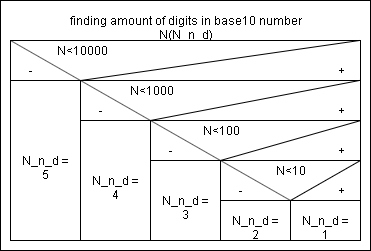


Рисунок 2

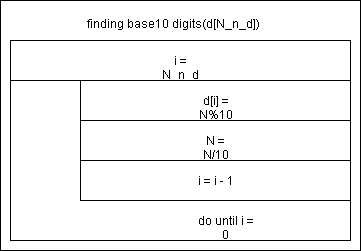


Рисунок 3

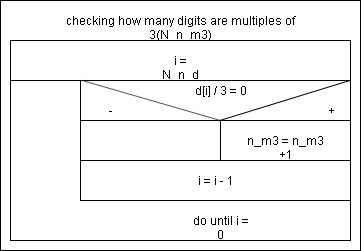


Рисунок 4

# **Лістинг програми.**

Turbo Assembler Version 3.0 11/09/19 18:58:40 Page 1

5v4.asm

1 0000 sseg segment stack 'stack'

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

3 0080 sseg ends

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

5 0000 dseg segment 'data'

6 0000 0000 N dw 0

7 0002 ?? N\_n\_d db ?;amount of digits for base10 N

8 0003 05\*(??) N\_d\_mas db 5 dup (?);array for digits of base10 N

9 0008 00 N\_d\_m3 db 0;amount of multiples of 3 in array N\_d\_mas

10 0009 49 6E 20 64 65 63 69+ msg\_res1 db "In decimal number N = ", '$'

11 6D 61 6C 20 6E 75 6D+

12 62 65 72 20 4E 20 3D+

13 20 24

14 0020 2C 20 6D 75 6C 74 69+ msg\_res2 db ", multiples of 3 are ", '$'

15 70 6C 65 73 20 6F 66+

16 20 33 20 61 72 65 20+

17 24

18 0036 20 64 69 67 69 74 73+ msg\_res3 db " digits", '$'

19 24

20 003E 0D 0A 24 msg\_endl db 0Dh, 0Ah, '$'

21 0041 dseg ends

22 ;-----------------------------------------------------------------------------

23 0000 cseg segment 'code'

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

25

26 0000 B8 0000s start: mov ax, dseg

27 0003 8E D8 mov ds, ax

28 ;N\_n\_d++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

29 0005 81 3E 0000r 2710 cmp N, 10000;checking how many digits N has

30 000B 7F 1E jg N\_n\_d\_5

31

32 000D 81 3E 0000r 03E8 cmp N, 1000

33 0013 7F 1E jg N\_n\_d\_4

34

35 0015 83 3E 0000r 64 cmp N, 100

36 001A 7F 1F jg N\_n\_d\_3

37

38 001C 83 3E 0000r 0A cmp N, 10

39 0021 7F 20 jg N\_n\_d\_2

40

41 0023 C6 06 0002r 01 mov N\_n\_d, 1

42 0028 EB 21 90 jmp end\_N\_n\_d

43

44 002B C6 06 0002r 05 N\_n\_d\_5:mov N\_n\_d, 5

45 0030 EB 19 90 jmp end\_N\_n\_d

46

47 0033 C6 06 0002r 04 N\_n\_d\_4:mov N\_n\_d, 4

48 0038 EB 11 90 jmp end\_N\_n\_d

49

50 003B C6 06 0002r 03 N\_n\_d\_3:mov N\_n\_d, 3

51 0040 EB 09 90 jmp end\_N\_n\_d

52

53 0043 C6 06 0002r 02 N\_n\_d\_2:mov N\_n\_d, 2

54 0048 EB 01 90 jmp end\_N\_n\_d

55

56 004B end\_N\_n\_d:

57 ;N\_d\_mas++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Turbo Assembler Version 3.0 11/09/19 18:58:40 Page 2

5v4.asm

58 004B 8A 0E 0002r mov cl, N\_n\_d

59 004F 32 ED xor ch, ch; cx = N\_n\_d

60 0051 BE 0000 mov si, 0

61 0054 A1 0000r mov ax, N

62 0057 33 D2 xor dx, dx; [dx:ax] = N

63

64 0059 BB 000A d\_loop: mov bx, 10

65 005C F7 F3 div bx

66 005E 88 94 0003r mov N\_d\_mas[si], dl; dl = dx, cuz 9<255

67 0062 33 D2 xor dx, dx;[dx:ax] = ax

68 0064 46 inc si

69 0065 E2 F2 loop d\_loop

70 ;N\_d\_m3+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

71 0067 8A 0E 0002r mov cl, N\_n\_d

72 006B 32 ED xor ch, ch; cx = N\_n\_d

73 006D BE 0000 mov si, 0

74 0070 m3\_loop:

75 0070 8A 84 0003r mov al, N\_d\_mas[si]

76 0074 32 E4 xor ah, ah; ax = N\_d\_mas[si]

77 0076 B7 03 mov bh, 3

78 0078 F6 F7 div bh

79 007A 80 FC 00 cmp ah, 0;checking if remainder =0 (N\_d\_mas[si] % 3 == 0)

80 007D 75 04 jne end\_iter\_m3\_loop

81

82 007F FE 06 0008r inc N\_d\_m3

83 0083 end\_iter\_m3\_loop:

84 0083 46 inc si

85 0084 E2 EA loop m3\_loop

86 ;out++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

87 ;print msg\_res1

88 0086 BA 0009r lea dx, msg\_res1

89 0089 B4 09 mov ah, 09h

90 008B CD 21 int 21h

91 008D print\_N:

92 008D 8A 0E 0002r mov cl, N\_n\_d

93 0091 32 ED xor ch, ch; cx = N\_n\_d

94 0093 8B F1 mov si, cx

95 0095 4E dec si

96 0096 print\_N\_loop:

97 0096 8A 94 0003r mov dl, N\_d\_mas[si]

98 009A 80 C2 30 add dl, 30h

99 009D B4 02 mov ah, 02h

100 009F CD 21 int 21h

101

102 00A1 4E dec si

103 00A2 E2 F2 loop print\_N\_loop

104 ;print msg\_res2

105 00A4 BA 0020r lea dx, msg\_res2

106 00A7 B4 09 mov ah, 09h

107 00A9 CD 21 int 21h

108 ;print N\_d\_m3

109 00AB 8A 16 0008r mov dl, N\_d\_m3

110 00AF 80 C2 30 add dl, 30h

111 00B2 B4 02 mov ah, 02h

112 00B4 CD 21 int 21h

113 ;print msg\_res3

114 00B6 BA 0036r lea dx, msg\_res3

Turbo Assembler Version 3.0 11/09/19 18:58:40 Page 3

5v4.asm

115 00B9 B4 09 mov ah, 09h

116 00BB CD 21 int 21h

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

118 00BD B4 4C mov ah, 4Ch

119 00BF CD 21 int 21h

120 00C1 cseg ends

121 end start

Turbo Assembler Version 3.0 11/09/19 18:58:40 Page 4

Symbol Table

Symbol Name Type Value

??DATE Text "11/09/19"

??FILENAME Text "5v4 "

??TIME Text "18:58:40"

??VERSION Number 0300

@CPU Text 0101H

@CURSEG Text CSEG

@FILENAME Text 5V4

@WORDSIZE Text 2

D\_LOOP Near CSEG:0059

END\_ITER\_M3\_LOOP Near CSEG:0083

END\_N\_N\_D Near CSEG:004B

M3\_LOOP Near CSEG:0070

MSG\_ENDL Byte DSEG:003E

MSG\_RES1 Byte DSEG:0009

MSG\_RES2 Byte DSEG:0020

MSG\_RES3 Byte DSEG:0036

N Word DSEG:0000

N\_D\_M3 Byte DSEG:0008

N\_D\_MAS Byte DSEG:0003

N\_N\_D Byte DSEG:0002

N\_N\_D\_2 Near CSEG:0043

N\_N\_D\_3 Near CSEG:003B

N\_N\_D\_4 Near CSEG:0033

N\_N\_D\_5 Near CSEG:002B

PRINT\_N Near CSEG:008D

PRINT\_N\_LOOP Near CSEG:0096

START Near CSEG:0000

Groups & Segments Bit Size Align Combine Class

CSEG 16 00C1 Para none CODE

DSEG 16 0041 Para none DATA

SSEG 16 0080 Para Stack STACK

# **Тести.**

Таблиця 2

|  |  |  |  |
| --- | --- | --- | --- |
| № | Назва | Вхідні дані | Очікувані результати |
| 1 | Додатне п’ятирозрядне число | 33333 | Кількість цифр кратних трьом у десятичному записі числа 33333 дорівнює 5 |
| 2 | Додатне чотирирозрядне число | 1234 | Кількість цифр кратних трьом у десятичному записі числа 1234 дорівнює 1 |
| 3 | Додатне трирозрядне число | 657 | Кількість цифр кратних трьом у десятичному записі числа 657 дорівнює 1 |
| 4 | Додатне дворозрядне число | 36 | Кількість цифр кратних трьом у десятичному записі числа 36 дорівнює 2 |
| 5 | Додатне одноразрядне число | 1 | Кількість цифр кратних трьом у десятичному записі числа 1 дорівнює 0 |
| 6 | Нуль | 0 | Кількість цифр кратних трьом у десятичному записі числа 0 дорівнює 1 |

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



Рисунок 5



Рисунок 6



Рисунок 7



Рисунок 8



Рисунок 9



Рисунок 10

Результати програми відповідають очікуваним. Нуль сприймається програмою, як число кратне трьом через те, що .

# **Висновки щодо можливості реалізацій циклічних конструкцій у мові Асемблера.**

У мові асемблера можна зробити такі види циклів:

1. З передумовою(перевірка у початку цикла).
2. З післяумовою(перевірка у кінці циклу).
3. Цикл з лічильником(виконується певну кількість разів).
4. Безкінечний цикл(цикл можна завершити тільки якщо «вбити» процес программи.

Перший цикл має такий синтаксис:

Mov ax, 0

Cmp ax, 8

Je end\_loop1

Add ax, 2

End\_loop1:

Другий цикл має такий синтаксис:

Mov ax, 0

Loop1: add ax, 2;

Cmp ax, 8; приклад умови

Jne loop1; У цьому прикладі вихід з цикла виконається коли ax = 8. Перевага цього циклу у тому, що він обов’язково виконується один раз(у деяких програмах це потрібно).

Третій цикл має такий синтаксис:

Mov ax, 0

Mov bx, 4

Loop1: add ax, 2

Dec bx

Cmp bx, 0

Je loop1

Але набагато простіше виконувати його завдяки команді loop:

Mov ax, 0

Mov cx, 4

Loop1: add ax, 2

Loop loop1; Цей цикл дуже добре підходить для обробки массивів, саме для цього я і використовував у своїй програмі.

Четвертий можна виконати, якщо використовувати безумовний перехід, наприклад:

lea dx, text

Mov ah, 09h

Loop1: Int 21h

Jmp loop1; цей цикл будет безкінечно виводити повідомлення, яке знаходится по мітці text.