**Московский авиационный институт**

**(национальный исследовательский университет)**

Институт №3

Системы управления, информатики и электроэнергетики

Кафедра 304

Вычислительные машины, системы и сети

**Отчет по лабораторной работе**

**по учебной дисциплине**

**«Языки Ассемблера»**

**на тему:**

***«***Блочная структура

ассемблер-программы***»***

*Группа:* ***М30-109Б-19***

*Выполнил:*

***Кузнецов И.И.***

[*iluxandro@gmail.com*](mailto:iluxandro@gmail.com)

*Принял:*

***Ратников Максим Олегович***

***Москва 2020***

**Содержание.**

**Задание\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3**

**Код программы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4**

**Ручной счет\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5**

**Тесты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6-7**

**Вывод\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_8**

***Задание***

**Вариант №7**

Разработать ассемблер-программу реализации линейного процесса в соответствии с вариантом задания. Результат выводить в 16-ричной системе счисления. Вычислить эталонный результат.

3.PNG

4.PNG

***Код программы***

org 100h

mov ax, [c] ; ax = c

not ax ; ax = NOT(c)

mov bx, [d] ; bx = d

or ax, bx ; ax = ax OR bx = d OR NOT(C)

mov bx, [a] ; bx = a

and bx, ax ; bx = bx AND ax = a AND (d OR NOT(c))

mov ax, [b] ; ax = b

not ax ; ax = NOT(b)

or bx, ax ; bx = bx OR ax = a AND (d OR NOT(c)) OR NOT(b)

mov ax, [c] ; ax = c

and ax, bx ; ax = ax AND bx = a AND (d OR NOT(c)) OR NOT(b) AND c

call print ; вызываем функцию

ret ; return

print proc ; начало описания функции

mov cx, 16 ; cx = 16

cycle:

xor dx, dx ; очищаем мусор лежащий в dx

div cx ; ax - остаток, dx - частное

mov bx, dx ; bx = dx

xchg ax, dx ; меняем значения ax с dx

cmp dx, 9 ; сравниваем остаток с 9

jg hex ; если больше 9 идем на метку hex

add dx, 30h ; если меньше, то добавляем 30h чтобы попасть в значения символов = 0 - 9

jmp next ; прыгаем на вывод

hex:

add dx, 37h ; добавляем 30h чтобы попасть в значения символов букв HEX’a

next:

mov ah, 2h ; вывод

int 21h ; вывод

cmp bx, 16 ; сравниваем частное с 16

jge cycle ; если частное больше или равно 16 повторяем

cmp bx, 9 ; P.S.: Теперь понял что нет смысла от такой проверки в рамках данной

xchg bx,dx ; программы, т.к. на 8 бит максим 2 тетрады, т.е. два 16-ричных

jg hex2 ; символа

add dx, 30h

jmp next2

hex2:

add dx, 37h

next2:

mov ah, 2h

int 21h

mov ah, 0 ; ожидание от пользователя

int 16h ; ввода любого символа

ret

print endp ; конец описания функции

a dw 00001001b ; 9 in decimal

b dw 00010010b ; 18 in decimal

c dw 11101000b ; -24 in decimal

d dw 00101100b ; 44 in decimal

**Ручной счет**

**a ^ (d v !c) v !b ^ c**

**a** = 9 = 1001, **b** = 18 = 10010, **d** = 44 = 101100, **c** = -24

24 = 00011000, обратный код = 11100111

1

**с** = -24 = 11101000

NOT(c) = NOT(11101000) = 00010111

NOT(b) = NOT(00010010) = 11101101

d v !c a ^ (d v !c)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| **0** | **0** | **1** | **1** | **1** | **1** | **1** | **1** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| **0** | **0** | **0** | **0** | **1** | **0** | **0** | **1** |

(a ^ (d v !c)) v !b ( (a ^ (d v !c)) v !b) ^ c

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |
| **1** | **1** | **1** | **0** | **1** | **0** | **0** | **0** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| **1** | **1** | **1** | **0** | **1** | **1** | **0** | **1** |

11101000**2** = 232**10** = E8**16**

**a** = -15 = 11110001, **b** = 7 = 00000111, **d** = -8 = 11111000 , **c** = 22 = 00010110

NOT(c) = NOT (00010110) = 11101001

NOT(b) = NOT(00000111) = 11111000

d v !c a ^ (d v !c)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| **1** | **1** | **1** | **1** | **1** | **0** | **0** | **1** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| **1** | **1** | **1** | **1** | **0** | **0** | **0** | **1** |

(a ^ (d v !c)) v !b ( (a ^ (d v !c)) v !b) ^ c

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| **0** | **0** | **0** | **1** | **0** | **0** | **0** | **0** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| **1** | **1** | **1** | **1** | **1** | **0** | **0** | **1** |

10 000**2** = 16 **10** = 10**16**

***Тесты***

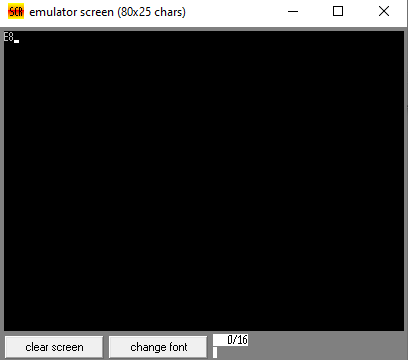
**Тест 1**

***Цель:*** проверить работу программы при изначальных исходных данных

a = 9, b = 18, c = -24, d = 44

***Ожидаемый результат****:* корректная работа (см.ручной счет = E8)

***Полученный результат:***

******

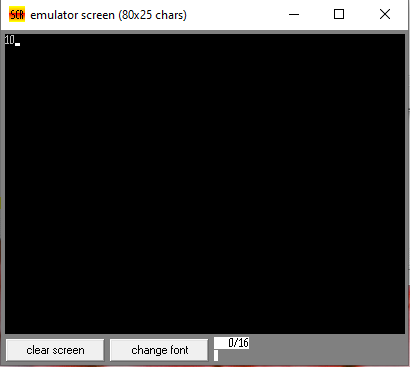
**Тест 2**

***Цель:*** проверить работу программы с другими исходными данными

a = -15, b = 7, c = 22, d = -8

***Ожидаемый результат****:* корректная работа (см.ручной счет = 10)

***Полученный результат:***



**Вывод:**

Программа работает без ошибок. Набор тестов считают достаточным.