МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ЧЕРЕПОВЕЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт информационных технологий |
| Кафедра математического и программного обеспечения |
| Программирование на ассемблере |
|  |

ЛАБОРАТОРНАЯ РАБОТА №3

|  |  |
| --- | --- |
| Исполнитель: | студент  группы 1ПИб-02-3оп-23  Богданов  Ренат Алексеевич |
| Руководитель: | Виноградова Людмила Николаевна |
| Оценка: |  |
| Подпись: |  |

2024 год

**Задание на лабораторную работу**

1. Даны три битовые переменные без знака a, b, c, d. Записать в d наибольшее из значений этих переменных.

2. Пусть a, b, c – числа размером в слово. Вычислить значение функции f при следующих условиях:

f=4\*a+b/c-6, если a>=b

f=6-c\*(a+b), если b>=c

f=3/a-(7+b)\*5, если c>=a

3. Пусть k – байтовая переменная со значением от 1 до 18. Записать в регистр AL количество двухзначных десятичных чисел (от 10 до 99), сумма цифр которых равна k.

**Задание 1**

**Код программы**

.model small

.stack 100h

.data

a db 1 ; Пример значения для a

b db 3 ; Пример значения для b

c db 2 ; Пример значения для c

d db ? ; Переменная для хранения наибольшего значения

.code

main:

mov ax, @data

mov ds, ax

; Сравнение a и b

mov al, a

mov bl, b

cmp al, bl

jbe b\_bigger\_or\_equal

mov d, al ; Если a > b, сохраняем a в d

jmp compare\_c

b\_bigger\_or\_equal:

mov al, bl ; Если b >= a, сохраняем b в al

mov d, al ; Затем сохраняем al в d

compare\_c:

; Сравнение текущего значения d и c

mov al, d

mov bl, c

cmp al, bl

jbe c\_bigger\_or\_equal

jmp end\_program ; Если d > c, d уже содержит наибольшее значение

c\_bigger\_or\_equal:

mov al, bl ; Если c >= d, сохраняем c в al

mov d, al ; Затем сохраняем al в d

end\_program:

mov al, d

; Exit program

mov ax, 4C00h

int 21h

end main

**Результаты работы программы**

Таблица 1. Результаты задания 1

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | c | d |
| 1 | 2 | 3 | 3 |
| 1 | 3 | 2 | 3 |
| 2 | 1 | 3 | 3 |
| 2 | 3 | 1 | 3 |
| 3 | 1 | 2 | 3 |
| 3 | 2 | 1 | 3 |

**Задание 2**

**Код программы**

.model small

.stack 100h

.data

a dw 10 ; Пример значения для a

b dw 5 ; Пример значения для b

c dw 2 ; Пример значения для c

f dw ? ; Переменная для хранения результата функции f

.code

main:

mov ax, @data

mov ds, ax

; Проверка условия a >= b

mov ax, a

mov bx, b

cmp ax, bx

jge condition\_a\_ge\_b

; Проверка условия b >= c

mov ax, b

mov bx, c

cmp ax, bx

jge condition\_b\_ge\_c

; Условие c >= a

mov ax, c

mov bx, a

cmp ax, bx

jge condition\_c\_ge\_a

; Если ни одно из условий не выполнено (теоретически не должно случиться)

jmp end\_program

condition\_a\_ge\_b:

; f = 4\*a + b/c - 6

mov ax, a

mov bx, 4

imul bx ; AX = 4\*a

mov bx, b

cwd ; Преобразуем AX в DX:AX для деления

idiv cx ; DX:AX = AX / C, AX = частное, DX = остаток

add ax, bx ; AX = AX + b/c

mov bx, 6

sub ax, bx ; AX = AX - 6

jmp store\_result

condition\_b\_ge\_c:

; f = 6 - c\*(a+b)

mov ax, a

add ax, b ; AX = a + b

mov bx, c

imul bx ; AX = c\*(a+b)

mov bx, 6

sub bx, ax ; BX = 6 - AX

mov ax, bx

jmp store\_result

condition\_c\_ge\_a:

; f = 3/a - (7+b)\*5

mov ax, 3

cwd ; Преобразуем AX в DX:AX для деления

mov bx, a

idiv bx ; DX:AX = 3 / A, AX = частное, DX = остаток

mov bx, ax ; BX = AX (частное от 3/a)

mov ax, 7

add ax, b ; AX = 7 + b

mov cx, 5

imul cx ; AX = (7 + b) \* 5

sub bx, ax ; BX = BX - AX

mov ax, bx

store\_result:

mov f, ax ; Сохраняем результат в f

end\_program:

; Exit program

mov ax, 4C00h

int 21h

end main

**Результаты работы программы**

Таблица 1. Результаты задания 1

|  |  |  |  |
| --- | --- | --- | --- |
| a | b | c | d |
| 1 | 2 | 3 | -42 |
| 1 | 3 | 2 | -2 |
| 2 | 1 | 3 | -5 |
| 2 | 3 | 1 | 1 |
| 3 | 1 | 2 | -5 |
| 3 | 2 | 1 | -4 |

**Задание 3**

**Код программы**

.model small

.stack 100h

.data

k db = 5

.code

main:

mov ax, @data

mov ds, ax

mov dh, k

mov ch, 1

xor al, al

loop1: mov bh, ch

xor cl,cl

jmp loop2

loop1n: inc ch

cmp ch, 10

jl loop1

jmp end\_program

loop2: mov bh, ch

mov bl, cl

add bh, cl

cmp bh, dh

jz incal

loop2n: inc cl

cmp cl, 10

jl loop2

jmp loop1n

incal:

inc al

jmp loop2n

end\_program:

mov ax, 4C00h

int 21h

end main

**Результаты работы программы**

Таблица 1. Результаты задания 1

|  |  |
| --- | --- |
| k | al |
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 5 | 5 |
| 6 | 6 |
| 7 | 7 |
| 8 | 8 |
| 9 | 9 |
| 10 | 9 |
| 11 | 8 |
| 12 | 7 |
| 13 | 6 |
| 14 | 5 |
| 15 | 4 |
| 16 | 3 |
| 17 | 2 |
| 18 | 1 |

**Контрольные вопросы**

1. *Сколько операндов имеют команды условного и безусловного переходов?*

Команды условного и безусловного переходов имеют один операнд, который определяет адресную метку или смещение, на которое должно передаваться управление.

1. *Опишите результат выполнения команды CMP?*

Команда CMP сравнивает два операнда, выполняя операцию вычитания второго операнда из первого без сохранения результата. На основе этого сравнения устанавливаются флаги процессора, такие как флаг переноса (CF), флаг нуля (ZF), флаг знака (SF), флаг переполнения (OF) и флаг четности (PF).

1. *Что такое дальний переход?*

Дальний переход представляет собой передачу управления из одного физического сегмента в другой. При этом меняются как содержимое регистра IP, так и содержимое регистра CS, что указывает на переход в другой физический сегмент. Дальние переходы могут использовать прямую и косвенную формы адресации и требуют указания сегмента и смещения.

1. *Влияют ли команды перехода на значения регистра флагов?*

Команды перехода не оказывают влияния на значения регистра флагов. Они просто передают управление на другую команду в зависимости от условий или всегда (в случае безусловных переходов).

1. *Для чего используется оператор SHORT?*

Оператор SHORT используется для явного указания, что переход должен быть коротким. Короткие переходы передают управление на команды, расположенные в пределах 127 байт вперед или 128 байт назад от команды перехода. Если метка находится в пределах этого диапазона, использование оператора SHORT позволяет убедиться, что будет использован короткий переход, а не ближний. Это важно для того, чтобы избежать ошибок компиляции, если метка находится слишком далеко для короткого перехода.

**Вывод**