МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра дискретной математики и информатики

ЛАБОРАТОРНАЯ РАБОТА №3

студента 2 курса 221 группы

направления 09.03.01 Информатика и вычислительная техника

факультета компьютерных наук и информационных технологий

Карасева Вадима Дмитриевича

Саратов 2025

**Текст задания**

Сначала программы должны печатать фамилию, имя и номер группы студента и переходить на новую строку. Используя рассмотренное упражнение, выполните следующие задания:

**Задание 3.1.** В регистре AX задано число от 0 до 65535. Выведите это число на экран. (Проверить программу для числа более 2600.)

**Задание 3.2.** Используя 32-битные регистры процессора (EAX, EBX, EDX), напишите программу, выводящую на экран число 65536. Число 65536 изначально поместить в регистр EAX.

**Задание 3.1.**

.model small

.stack 100h

.386 ;разрешение трансляции команд процессора 386

.data

Info db "Карасев Вадим 221", 13, 10, "$"

.code

start:

call print\_info

mov ax, 60155 ;число в al

mov bx, 10 ;система счисления в bl

mov cx, 0 ;разрядность

m1: mov dx, 0 ;помещаем в dx 0 (обнуляем перед каждым делением)

div bx ;(ax) / (bx), остаток в dx, неполное частное в ax

push dx ;сохраняем значения

inc cx ;cx += 1

cmp ax, 0 ;сравниваем (ax - 0)

jne m1

m2: pop dx ;выгружаем ax в dx

call print\_digits

loop m2 ;повторяем пока cx != 0

mov ax, 4C00h

int 21h

;------ Процедуры ------

print\_info proc

mov ax, @data

mov ds, ax

mov ah, 09h

mov dx, offset Info

int 21h

ret

print\_info endp

print\_digits proc

push ax

mov ah, 02h

add dl, '0'

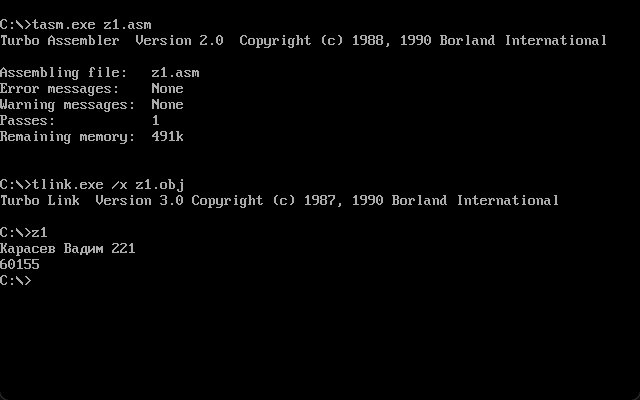
int 21h

pop ax

ret

print\_digits endp

end start



**Задание 3.2.**

.model small

.stack 100h

.386

.data

Info db "Карасев Вадим 221", 13, 10, '$'

.code

start:

call print\_info ;вывод информации

mov eax, 65536 ;загружаем число

mov ebx, 10 ;система счисления

mov ecx, 0

m1: mov edx, 0

div ebx

push edx ;сохраняем результат

inc ecx ;(ecx)++

cmp eax, 0 ;условие (eax) - 0

jne m1 ;если (eax) не равен 0, то идём в m1

m2: pop edx ;иначе выполняем код дальше

call print\_digits ;процедура вывода цифры

loop m2 ;пока ecx != 0

mov ax, 4C00h

int 21h

;------ Процедуры ------

print\_info proc

mov ax, @data

mov ds, ax

mov ah, 09h

mov dx, offset Info

int 21h

ret

print\_info endp

print\_digits proc

push eax

mov ah, 02h

add dl, '0'

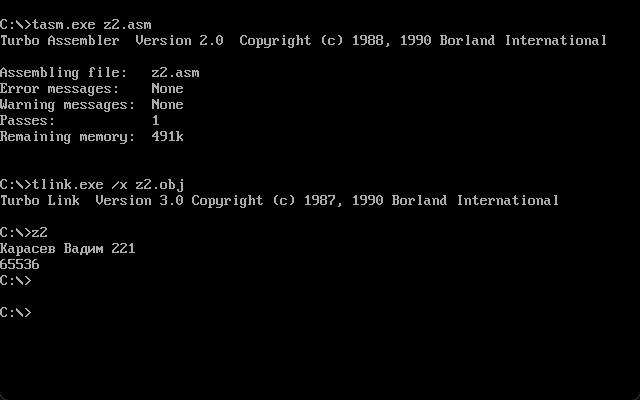
int 21h

pop eax

ret

print\_digits endp

end start



**Контрольные вопросы**

**Чем отличается деление на байт от деления на слово? (где должно располагаться делимое, куда попадут частное от деления и остаток от деления)**

Команда, работающая с байтами, делит 16-битовое делимое на 8-битовый делитель (из логики, что делимое в 2 раза размернее делителя), при этом делимое находится в AX. Целая часть от деления помещается в AL, остаток от деления помещается в AH.

Команда, работающая со словами, делит 32-битовое делимое на 16-битовый делитель, при этом делимое находится в паре регистров: DX - старшая (значимая) часть, а AX - младшая часть. Целая часть помещается в AX, а остаток - в DX.

**Каков механизм действия команды cmp? В паре с какими командами она обычно используется?**

Команда cmp является командой сравнения путём вычитания второго оператора из первого. cmp не сохраняет результат, а лишь меняет флажки состояния в зависимости от ожидаемого результата. Основным применением данной команды является организация циклов ветвлений. Обычно использование команды cmp подразумевает в дальнейшем использование команды условного перехода jcc.

**На какие флаги реагируют команды условного перехода для чисел со знаком и для чисел без знака?**

Первая группа команд Jcc (кроме JCXZ/JECXZ ) проверяет текущее состояние регистра флагов (не изменяя его) и в случае соблюдения условия осуществляет переход на смещение, указанное в качестве операнда. Флаги, проверяемые командой, кодируются в ее мнемонике, например: JC– переход, если установлен CF. Сокращения «L» (less – меньше) и «G» (greater – больше) применяются для целых со знаком, а «A» (above – над) и «B» (below – под) для целых без знака.

Цифра 0 означает, что флаг должен быть сброшен, а цифра 1 означает, что флаг должен быть установлен, чтобы условие было выполнено (переход произошел).

Команды условного перехода можно разделить на три подгруппы:

- Непосредственно проверяющие один из флагов на равенство 0 или 1.

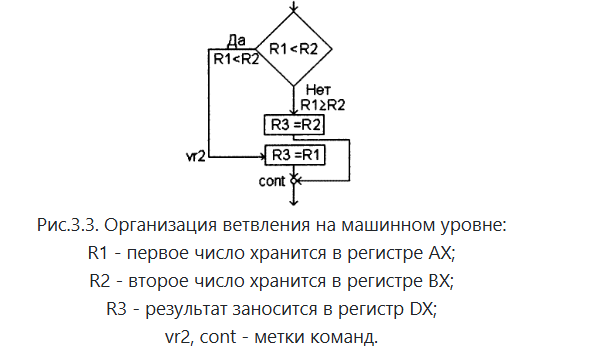
- Арифметические сравнения со знаком. Существуют 4 условия, которые могут быть проверены:

меньше (JL), меньше или равно (JLE), больше (JG), больше или равно (JGE). Эти команды проверяют одновременно три флага: знака, переполнения и нуля.

- Арифметические без знака. Здесь также существует 4 возможных соотношения между операндами: меньше (JB), меньше или равно (JBE), больше (JA), больше или равно (JAE).

Учитываются только два флага. Флаг переноса показывает какое из двух чисел больше. Флаг нуля определяет равенство.

**С помощью команд условного и безусловного перехода выполните программную реализацию алгоритма ветвления для определения наименьшего числа из двух заданных. Алгоритм изображен в виде блок-схемы, приведенной на рис.3.3.**



.model small

.stack 100h

.386

.data ;трансляция команд процессора 386

.code ;код

start: ;точка входа в программу

mov AX, @data

mov DS, AX ;т.к. напрямую в DS нельзя загрузить

mov AX,60155 ;число

mov BX,10 ;система счисления

cmp AX,BX

JL m1

mov DX, BX

jmp m2

m1: mov DX, AX

m2: mov ax,4C00h

int 21h ;вызов кода 0

end start

**Каков механизм работы команды организации цикла LOOP?**

Все команды цикла используют регистр CX в качестве счетчика цикла. Простейшая из них– команда LOOP. Она в конце каждой итерации уменьшает содержимое CX на 1 и передает управление на метку (указанную в команде), если содержимое CX не равно 0. Если вычитание 1 из CX привело к нулевому результату, выполняется следующая команда.

Команда LOOPNE (цикл пока не равно) выходит из цикла, если не установлен флаг нуля или если в регистре CX получился 0.

Команда LOOPE (цикл пока равно) выполняет обратную к описанной проверку флага нуля: цикл здесь завершается, если регистр CX достиг 0 или если установлен флаг 0.

**Как с помощью команды сдвига можно умножить знаковое число, хранящееся в АХ, на 2 в n-ой степени?**

Для этого нужно сдвинуть содержимое регистра влево на n битов. Достигается это с помощью команды Ассемблера shl <op1>, <счетчик>.

**Как с помощью команды сдвига проверить содержимое регистра ВХ на четность?**

Для этого нужно сделать логический сдвиг вправо на 1 бит, после сдвига младший разряд числа перейдет в флаг CF, проверяем его с помощью команды условного переход. За этот сдвиг отвечает команда shr <op1>, <счетчик>.