**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ **«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. В. Г. ШУХОВА»**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

**ЛАБОРАТОРНАЯ РАБОТА №6**

**Дисциплина: Архитектура вычислительных систем**

**Тема: Логические команды и команды сдвига**

Выполнил: ст. группы ВТ-31

Подкопаев Антон Валерьевич

Проверил: Осипов Олег Васильевич

**Белгород 2019**

**Цель работы:** изучение команд поразрядной обработки данных.

**Задания для выполнения к работе**

1. Написать программу для вывода чисел на экран согласно варианту задания. При выполнении задания №1 все числа считать беззнаковыми. Написать и использовать функцию output(a) для вывода числа a на экран или в файл. Функция должна удовлетворять соглашению о вызовах. В функцию для вывода output передавать в качестве аргумента переменную размерности 32 или 64 бита, которой достаточно для хранения числа. К примеру, если в задании число указано как 15-разрядное, то аргументом функции должно быть число размером двойное слово, если 40-разрядное, то учетверённое слово. Функция должна выводить столько разрядов числа, сколько указано в задании, даже если старшие разряды равны нулю. Не допускается прямой перебор всех чисел с проверкой, удовлетворяет ли оно условию вывода (за исключением вариантов № 8, 12, 13). Числа выводить в порядке, который является удобным. Проверить количество выведенных чисел с помощью формул комбинаторики. В отчёт включить вывод формул и результаты работы программы.

2. Написать подпрограмму для умножения (multiplication) или деления (division) большого целого числа на 2*n* (в зависимости от варианта задания) с использованием команд сдвига. Подпрограммы должны иметь следующие заголовки:

multiplication(char\* a, int n, char\* res);

division(char\* a, int n, char\* res).

Входные параметры: *a* – адрес первого числа в памяти, *n* – степень двойки. Выходные параметры: *res* – адрес массива, куда записывается результат. В случае операции умножения, для массива *res* зарезервировать в два раза больше памяти, чем для множителей *a* и *b*. Числа *a*, *b*, *res* вывести на экран в 16-ричном виде. Подобрать набор тестовых данных для проверки правильности работы подпрограммы.

**Вариант 13**

|  |  |  |
| --- | --- | --- |
| 13 | Вывести все 48-разрядные числа, в 16-ричном представлении которых есть только одна из цифр "3", "5", "7", "F", остальные – "A".  1: AAAAAAAAAAA7  2: AAAAAAAAAA7A  3: AAAAAAAAA7AA  … | 42 байта  умножение  со знаком |

**Ход работы**

**Задание 1**

**Изображение выглядит как монитор, компьютер, снимок экрана, ноутбук

Автоматически созданное описаниеИзображение выглядит как снимок экрана, компьютер, монитор, ноутбук

Автоматически созданное описание**

**Задание 2**

|  |  |
| --- | --- |
| Входное значение | Вывод функции |
| 400000000002 | 800000000004 |
| 111111111111 | 222222222222 |
| 12345678 | 24691356 |

.386

.model flat, stdcall

option casemap: none

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

.DATA

alphabet db 3h,5h,7h,0Fh

count\_format db "%d. ", 0

half\_format db "%X", 0

number\_format db "%X",0Ah

new\_line\_format db 13, 10, 0

empty\_num dd 0AAAAAAAAh

half\_num dw 0AAAAh

.CODE

safe\_print proc

PUSH EAX

PUSH EBX

PUSH ECX

PUSH EDX

PUSH EBP

MOV EAX, [ESP+24]

mov EBX, [ESP+28]

;форматная строка

PUSH EBX

;операнд

PUSH EAX

CALL crt\_printf

POP EAX

POP EBX

POP EBP

POP EDX

POP ECX

POP EBX

POP EAX

RET 8

safe\_print endp

START:

XOR EAX, EAX ; очистка регистров

XOR ECX, ECX

XOR EBP, EBP

MOV EBP, 1 ; запишем 1 в EBP

MOV ECX, 1 ; запишем 1 в ECX

loop\_start:

;вложенный цикл

MOV BL, alphabet[ECX-1] ; запишем в BL символ для перебора

PUSH ECX ; сохраним ECX в стеке

MOV ECX, 7 ; запишем 7 в ECX

;установка первого символа

XOR EDX, EDX ; очистка регистра EDX

MOV DX, half\_num ; запишем 0AAAAh в DX

MOV EAX, empty\_num ; запишем 0AAAAAAAAh в EAX

shl EAX, 4 ; выполним побитовый сдвиг влево на 4 регистра EAX

ADD AL, BL ; сложим регистры AL и BL, результат в AX

;вывод первого числа

PUSH EBP ; запишем в стек EBP

PUSH offset count\_format

CALL safe\_print

INC EBP ; прибавим 1 к EBP

PUSH EDX ; запишем EDX в стек

PUSH offset half\_format

CALL safe\_print

PUSH EAX ; запишем EAX в стек

PUSH offset number\_format

CALL safe\_print

ex\_loop:

shl EAX, 4 ; выполним побитовый сдвиг влево на 4 регистра EAX

ADD AL, 000Ah ; прибавим к AL 000Ah

PUSH EBP ; запишем EBP в стек

PUSH offset count\_format

CALL safe\_print

INC EBP ; прибавим 1 к EBP

PUSH EDX

PUSH offset half\_format

CALL safe\_print

PUSH EAX ; запишем EAX в cn

PUSH offset number\_format

CALL safe\_print

LOOP ex\_loop

MOV ECX, 3 ; запишем 3 в ECX

MOV EAX, empty\_num ; запишем 0AAAAAAAAh в EAX

shl EDX, 4 ; выполним побитовый сдвиг влево на 4 регистра EDX

ADD DL, BL ; сложим регистры DL и BL, результат в DX

and EDX, 0000FFFFh ; выполним логическое И над EDX и 0000FFFFh

PUSH EBP ; запишем EBP в стек

PUSH offset count\_format

CALL safe\_print

INC EBP ; прибавим 1 к EBP

PUSH EDX ; запишем EDX

PUSH offset half\_format

CALL safe\_print

PUSH EAX ; запишем EAX в стек

PUSH offset number\_format

CALL safe\_print

ex\_loop2:

shl EDX, 4 ; выполним побитовый сдвиг влево на 4 регистра EDX

ADD DL, 000Ah ; прибавим к DL 000Ah

and EDX, 0000FFFFh ; выполним логическое И над EDX и 0000FFFFh

PUSH EBP ; запишем EBP в стек

PUSH offset count\_format

CALL safe\_print

INC EBP ; прибавим 1 к EBP

PUSH EDX ; запишем EDX

PUSH offset half\_format

CALL safe\_print

PUSH EAX ; запишем EAX в стек

PUSH offset number\_format

CALL safe\_print

LOOP ex\_loop2

POP ECX ; вернем ECX з

INC ECX ; прибавим 1 к ECX

CMP ECX, 5 ; стравним ECX и 5

JL loop\_start ; если ECX меньше, повторить цикл

END START

.386

.model flat, stdcall

option casemap: none

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

.DATA

format db "%x%x"

big\_number dd 1d,1d,1d,1d,1d,1d,1d,1d,1d,1d,1d

.CODE

mult\_arr proc

PUSH EAX

PUSH EBX

PUSH ECX

PUSH EDX

XOR EAX, EAX

MOV ECX, [ESP+20]

outer\_loop:

PUSH ECX

MOV ECX, 0

clc

inner\_loop:

MOV EBX, [ESP+28]

MOV EDX, dword ptr [ECX\*4+EBX]

sahf

ADC EDX,0

shl EDX,1

lahf

MOV [ECX\*4+EBX], EDX

INC ECX

CMP ECX, 11

JL inner\_loop

POP ECX

LOOP outer\_loop

POP EDX

POP ECX

POP EBX

POP EAX

RET 8

mult\_arr endp

START:

PUSH offset big\_number

PUSH 2

CALL mult\_arr

END START