**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ**

**ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»**

**(БГТУ 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 ПВ-31

Донцов А.А.

Проверил: Осипов О.В.

Белгород 2019

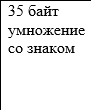
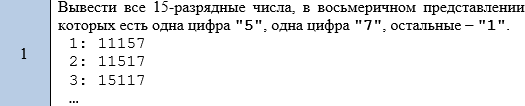
**Цель работы:** изучение команд поразрядной обработки данных.

**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).



.486

.model flat, stdcall

option casemap: none

include c:\masm32\include\kernel32.inc

include c:\masm32\include\msvcrt.inc

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

.data

number\_format db "%d:", 0

format\_8 db "-%o", 13, 10, 0

mask1 dw 001b

mask5 dw 101b

mask7 dw 111b

.code

; Процедура для вывода двоичного представления 8-битного числа

; void output (unsigned int a). Процедура в качестве аргумента принимает не 8-разрядное, а 32-разрядное целое число, но в процедуре используется только младший байт числа, остальные игнорируются

output proc

;Сохранить в стеке значения регистров, которые будут использованы

PUSH EAX ; Запомнить EAX

PUSH EBX ; Запомнить EBX

PUSH ECX ; Запомнить ECX

XOR EBX, EBX ; Обнулить EBX

MOV AX, [ESP+4\*4] ; Взять из стека аргумент, т.е. число, которое нужно вывести в двоичном представлении

ROL AX, 1

MOV ECX, 15 ; Чтобы вывести 8-битное число, необходим цикл. Помещаем в ECX количество итераций

j1:

ROL AX, 1 ; Сделать циклический сдвиг числа на один разряд влево. Таким образом старший бит попадёт на место младшего

MOV BX, AX ; BL = AL

AND BX, 00000001b ; Оставить только младший бит, остальные обнулить

ADD BX, '0' ; Прибавить к BL код символа "0"

PUSH EAX ; Команда для вывода символа на экран crt\_putch изменяет регистры EAX и ECX, поэтому нужно сохранить их в стеке

PUSH ECX

PUSH EBX ; Поместить выводимый символ в стек, т.е. передать его в качестве аргумента функции crt\_\_putch

CALL crt\_\_putch ; Вызвать функцию

ADD ESP, 4 ; Удалить аргумент из стека, так как функция crt\_\_putch этого не делает

POP ECX ; Восстановить ECX

POP EAX ; Восстановить EAX

LOOP j1 ; ECX = ECX - 1. Выполнять цикл пока ECX ? 0

POP ECX ; Восстановить ECX

POP EBX ; Восстановить EBX

POP EAX ; Восстановить EAX

RET 4 ; Возврат к основной программе и очистка стека от аргумента размером 4 байта

output endp

start:

MOV ESI, 1 ; номер последовательности

MOV EBX, 5 ;количество символов в 8меричной системе счисления

SUB ESP, 2 ; выделение 2 байт памяти в стеке

MOV EBP, ESP ; сохранение адреса локальной переменной

MOV dword ptr [EBP], 0 ; обнуление локальной переменной

MOV EDX, EBX ; передаём в счётчик количество итераций, на каком месте встретится 5

j1:

MOV EAX, 5 ; передаём в счётчик количество итераций, на каком месте встретится 7

j2:

MOV ECX, EBX

j3:

CMP EDX, EAX ; сравнение

JE j5 ; пропускаем последовательность

CMP EDX, ECX

JE jmask5

CMP EAX, ECX

JE jmask7

jmp jmask1

j4:

loop j3

jmp joutput

j6:

MOV dword ptr [EBP], 0 ; обнуление локальной переменной

j5:

DEC EAX

CMP EAX, 0 ; сравнение

JG j2 ; если больше нуля, то перейти по ссылке

DEC EDX

CMP EDX, 0 ; сравнение

JG j1 ; если больше нуля, то перейти по ссылке

jmp exit1

jmask1:

push EAX

MOV AX, word ptr [EBP]

SHL AX, 3 ; смещение влево на 3 бита

OR AX, mask1 ; добавление 1 в конец

MOV word ptr [EBP], AX

POP EAX

jmp j4

jmask5:

push EAX

MOV AX, word ptr [EBP]

SHL AX, 3 ; смещение влево на 3 бита

OR AX, mask5 ; добавление 5 в конец

MOV word ptr [EBP], AX

POP EAX

jmp j4

jmask7:

push EAX

MOV AX, word ptr [EBP]

SHL AX, 3 ; смещение влево на 3 бита

OR AX, mask7 ; добавление 1 в конец

MOV word ptr [EBP], AX

POP EAX

jmp j4

joutput:

PUSH EAX

PUSH EBX

PUSH ECX

PUSH EDX

PUSH ESI

PUSH offset number\_format

call crt\_printf

ADD ESP, 8

push dword ptr[EBP]

call output

PUSH dword ptr[EBP]

PUSH offset format\_8

call crt\_printf

ADD ESP, 8

POP EDX

POP ECX

POP EBX

POP EAX

INC ESI

jmp j6

exit1:

call crt\_\_getch ; Задержка ввода

push 0

call ExitProcess

end start

2 номер

.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

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

include c:\masm32\include\msvcrt.inc

includelib c:\masm32\lib\msvcrt.lib

.data

n dd 6

a dd 1, 0, 0, 0, 0, 0, 0, 0, 0

res dd 18 dup(0)

fmt db "n = %hu", 13, 10, 0

number\_format\_a db "a = %08X %08X %08X %08X %08X %08X %08X %08X %08X", 13, 10, 0

number\_format\_res db "res = %08X %08X %08X %08X %08X %08X %08X %08X %08X %08X %08X %08X %08X %08X %08X %08X %08X %08X", 13, 10, 0

.code

multiplication proc

;Сохранить в стеке значения регистров, которые будут использованы

push eax

push ebx

push ecx

push esi

push edi

push ebp

; достать данные

xor eax, eax

xor ebx, ebx

mov esi, [esp+8+4\*7] ; res

mov edi, [esp+4\*7] ; a

mov ecx, [esp+4+4\*7] ; n

push ecx

mov ecx, 34

; заполнение массива результата изначальными числами

init\_res\_loop:

mov al, byte ptr[edi + ecx]

mov byte ptr[esi + ecx], al

loop init\_res\_loop

mov al, byte ptr[edi]

mov byte ptr[esi], al

pop ecx

outer\_loop:

push ecx

mov ecx, 34

mov ebp, ecx

mov AL, byte ptr[esi+ebp]

mov BL, byte ptr[esi+ebp]

rol BL, 1

and BL, 00000001b; сдвинутый бит

shl AL, 1; готовый байт

mov byte ptr[esi+ebp], AL

inner\_loop:

dec ebp

mov AL, byte ptr[esi+ebp]

mov DL, byte ptr[esi+ebp]

rol DL, 1

and DL, 00000001b; сдвинутый бит

sal AL, 1; cмещенный байт

or AL, BL; готовый байт с учетом сдвинутого бита старшего байта

mov byte ptr[esi+ebp], AL

mov BL, DL; запомнить сдвинутый бит для младшего байта

loop inner\_loop

pop ecx

loop outer\_loop

pop ebp

pop edi

pop esi

pop ecx

pop ebx

pop eax

ret 12

multiplication endp

start:

push dword ptr a[0]

push dword ptr a[4]

push dword ptr a[8]

push dword ptr a[12]

push dword ptr a[16]

push dword ptr a[20]

push dword ptr a[24]

push dword ptr a[28]

push dword ptr a[32]

push offset number\_format\_a

call crt\_printf

add esp, 10\*4

push word ptr n

push offset fmt

call crt\_printf

add ESP, 8

push offset res

push n

push offset a

call multiplication

; вывод результата

push dword ptr res[0]

push dword ptr res[4]

push dword ptr res[8]

push dword ptr res[12]

push dword ptr res[16]

push dword ptr res[20]

push dword ptr res[24]

push dword ptr res[28]

push dword ptr res[32]

push dword ptr res[36]

push dword ptr res[40]

push dword ptr res[44]

push dword ptr res[48]

push dword ptr res[52]

push dword ptr res[56]

push dword ptr res[60]

push dword ptr res[64]

push dword ptr res[68]

push offset number\_format\_res

call crt\_printf

add esp, 10\*4

call crt\_\_getch ; задержка ввода с клавиатуры

push 0

call ExitProcess

end start