**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

по теме Логические команды и команды сдвига

Выполнил: ст. группы ВТ-31  
Новожен Н.В

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 2n (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.

|  |  |  |
| --- | --- | --- |
| Вари-ант | Задание №1 | Задание №2 |
| 11 | Вывести все 16-разрядные числа, в двоичном представлении которых есть две или одна единицы, остальные – ноль.  1: 0000000000000001  2: 0000000000000011  3: 0000000000000101  … | 40 байт  умножение  без знака |

**Задание 1** (Вывести все 16-разрядные числа, в двоичном представлении которых есть две или одна единицы, остальные – ноль.)

**C*nk******=*n*!/ (*(n − k)*!\*k!) –число сочетаний из n по k***

***Длинна вектора 16 => n=16***

Одна единица k=1 => C161=16! / ((16-1)! \*1!) =16

Две единицы k=2 => C162=16! / ((16-2)! \*2!) =16\*15/2=120

**Всего вариантов A= A162 + A161 =120+16=136**

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

include E:\masm32\include\windows.inc

include E:\masm32\include\kernel32.inc

include E:\masm32\include\user32.inc

includelib E:\masm32\lib\user32.lib

includelib E:\masm32\lib\kernel32.lib

include E:\masm32\include\msvcrt.inc

includelib E:\masm32\lib\msvcrt.lib

.DATA

s db 13, 10, 0

.CONST

N dd 16

.CODE

; Процедура для вывода двоичного представления 16-битного числа

; void output (int a). Процедура в качестве аргумента

;принимает 32-разрядное целое число, но в процедуре

;используется только 2 младшиx байтf числа, остальные игнорируются

output proc

;Сохранить в стеке значения регистров, которые будут

;использованы

PUSH EDX ; Запомнить EAX

PUSH EAX ; Запомнить EAX

PUSH EBX ; Запомнить EBX

PUSH ECX ; Запомнить ECX

XOR EBX, EBX ; Обнулить EBX

MOV AX, [ESP+4\*5] ; Взять из стека аргумент, т.е. число, которое

;нужно вывести в двоичном представлении

MOV ECX, N ; Чтобы вывести 16-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 ; BX = AX

AND BX, 1b;0000 0000 0000 0001b ; Оставить только младший бит, остальные обнулить

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

; Команда для вывода символа на экран crt\_putch

;изменяет регистры EAX и ECX, поэтому нужно сохранить их в стеке

PUSH EAX

PUSH ECX

; Поместить выводимый символ в стек, т.е. передать его

;в качестве аргумента функции crt\_\_putch

PUSH EBX

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

POP EDX

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

output endp

new\_line proc

PUSH EDX

PUSH EAX

PUSH ECX

PUSH offset s

CALL crt\_printf

ADD ESP, 4

POP ECX

POP EAX

POP EDX

RET 0

new\_line endp

START:

XOR EAX,EAX

XOR EDX,EDX

XOR EDI,EDI

XOR ECX,ECX

MOV ECX,N; ECX=16

MOV EAX,1

j2:

PUSH ECX;SAVE

PUSH EAX;OUTPUT

CALL output

CALL new\_line;NEW LINE

MOV DX,AX; DX=AX

DEC ECX; --ECX

CMP ECX,0

je j4

j3:

SHL DX, 1

MOV DI,DX

OR DI,AX

PUSH EDX

PUSH EDI

CALL output

POP EDX

CALL new\_line

LOOP j3

j4:

SHL AX, 1

POP ECX ;RETURN ECX

LOOP j2

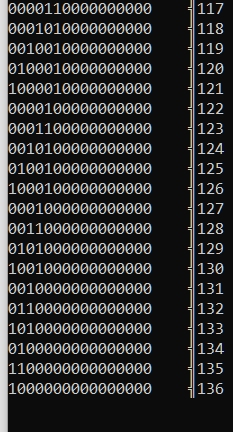
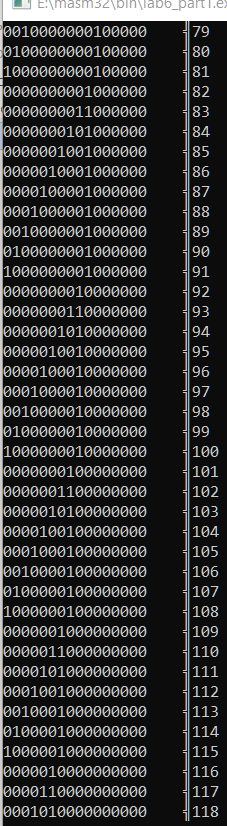
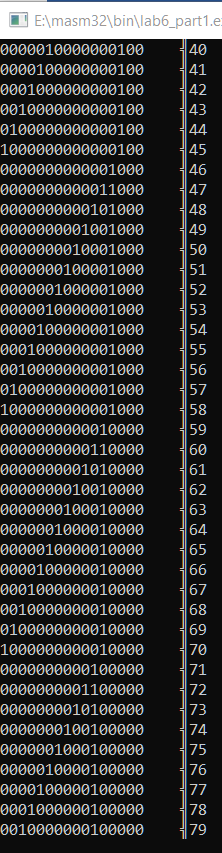
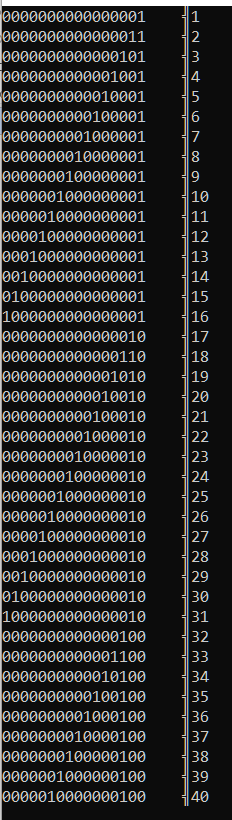
CALL crt\_\_getch

push NULL

call ExitProcess

END START

**Результат работы программы**



**Задание 2**(40 байт умножение без знака)

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);

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.

.386

.MODEL FLAT, STDCALL

OPTION CASEMAP: NONE

include E:\masm32\include\windows.inc

include E:\masm32\include\kernel32.inc

include E:\masm32\include\user32.inc

includelib E:\masm32\lib\user32.lib

includelib E:\masm32\lib\kernel32.lib

include E:\masm32\include\msvcrt.inc

includelib E:\masm32\lib\msvcrt.lib

.DATA

s db 13, 10, 0

t db "%.2X:", 0

a db 40 dup(1)

b dd 0

res db 80 dup(?)

.CONST

N dd 16

.CODE

; Процедура для вывода двоичного представления 16-битного числа

; void output (int a). Процедура в качестве аргумента

;принимает 32-разрядное целое число, но в процедуре

;используется только 2 младшиx байтf числа, остальные игнорируются

output proc

;Сохранить в стеке значения регистров, которые будут

;использованы

PUSH EDX ; Запомнить EAX

PUSH EAX ; Запомнить EAX

PUSH EBX ; Запомнить EBX

PUSH ECX ; Запомнить ECX

XOR EBX, EBX ; Обнулить EBX

MOV EAX, [ESP+4\*5]

MOV ECX, [ESP+4\*6]; Взять из стека аргумент, т.е. число, которое

;нужно вывести в двоичном представлении

;MOV ECX, N ; Чтобы вывести 16-битное число, необходим цикл.

;Помещаем в ECX количество итераций

XOR EDI,EDI

j1:

;ROL AX, 1 ; Сделать циклический сдвиг числа на один разряд влево.

;Таким образом старший бит попадёт на место младшего

;MOV BX, AX ; BX = AX

;AND BX, 1b;0000 0000 0000 0001b ; Оставить только младший бит, остальные обнулить

XOR EBX,EBX

MOV BL, [EAX+EDI] ;[EAX+EDI];[EAX+ECX-1] ; Прибавить к BX код символа "0"

; Команда для вывода символа на экран crt\_putch

;изменяет регистры EAX и ECX, поэтому нужно сохранить их в стеке

INC EDI

PUSH EAX

PUSH ECX

; Поместить выводимый символ в стек, т.е. передать его

;в качестве аргумента функции crt\_\_putch

PUSH EBX

PUSH offset t

CALL crt\_printf ; Вызвать функцию

ADD ESP, 8; 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

POP EDX

RET 8 ; Возврат к основной программе и очистка стека от аргумента размером 4 байта

output endp

new\_line proc

PUSH EDX

PUSH EAX

PUSH ECX

PUSH offset s

CALL crt\_printf

ADD ESP, 4

POP ECX

POP EAX

POP EDX

RET 0

new\_line endp

;Входные параметры: a – адрес первого числа в памяти, n – степень двойки.

;Выходные параметры: res – адрес массива, куда записывается результат.

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

PUSH EAX; SAVE

PUSH EDX; SAVE

PUSH ESI; SAVE

PUSH EDI; SAVE

PUSH ECX; SAVE

PUSH EBX; SAVE

MOV ESI,[ESP +4\*7]; GET CHAR \*a

MOV ECX,[ESP +4\*8]; GET INT n

MOV EDI,[ESP +4\*9]; GET CHAR \*res

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

CMP ECX,80

JGE j9

PUSH ECX

MOV ECX,40

jk:

MOV DL,[ESI+ECX-1]

MOV[EDI+ECX-1],DL

LOOP jk

POP ECX

fr:

PUSH ECX

MOV ECX ,80

DEC ECX

SHR BYTE PTR[EDI],1

MOV EBX,0

lp:

INC EBX

RCR BYTE PTR[EDI+EBX],1

LOOP lp

POP ECX

LOOP fr

; SHR [ESI],1

j9:

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

POP EBX;RETURN VALUE

POP ECX;RETURN VALUE

POP EDI;RETURN VALUE

POP ESI;RETURN VALUE

POP EDX;RETURN VALUE

POP EAX;RETURN VALUE

RET 12

multiplication endp

START:

;AFTER

MOV EAX,40

PUSH EAX

PUSH OFFSET a

CALL output

CALL new\_line

MOV ECX,2

PUSH OFFSET res

PUSH ECX

PUSH OFFSET a

CALL multiplication

CALL new\_line

;BEFORE

MOV EAX,80

PUSH EAX

PUSH OFFSET res

CALL output

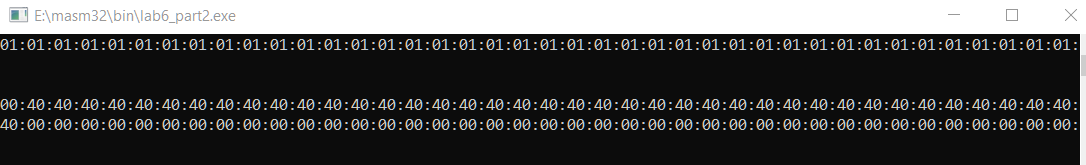
CALL new\_line

CALL crt\_\_getch

push NULL

call ExitProcess

END START



Вывод: Мы изучили команды поразрядной обработки данных.