**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-32  
Воскобойников И. С.

Проверил: Осипов.О.В

**Белгород 2020**

**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).

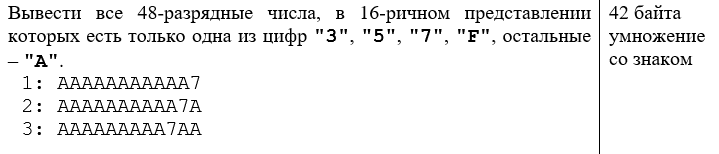
Входные параметры: a – адрес первого числа в памяти, n – степень двойки.

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.



**Задание 1**

.686 ; Тип процессора

.model flat, stdcall ; Модель памяти и стиль вызова подпрограмм

option casemap: none ; Чувствительность к регистру

; --- Подключение файлов с кодом, макросами, константами, прототипами функций и т.д.

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

; --- Подключаемые библиотеки ---

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

.data

digit db 6 dup(10101010b)

digitMask db 5 dup(00000000b), 00001111b

useN db 4

canUse db 10100011b, 10100101b, 10100111b, 10101111b

format db "%X", 0

formatN db 10, 0

.code

outputArray proc

PUSH EDI

PUSH ESI

MOV EDI, [ESP + 16] ; digit[0]

MOV ESI, [ESP + 12] ; n

PUSH EAX

PUSH EBX

PUSH EDX

PUSH ECX

MOV EBX, [ESP + 36] ; index

SUB ESP, 6 ; Буфер под локальную строку формата

MOV BYTE PTR [ESP], '%' ; Сформируем локальную строку формата для вывода

MOV BYTE PTR [ESP + 1], '2'

MOV BYTE PTR [ESP + 2], 'd'

MOV BYTE PTR [ESP + 3], ':'

MOV BYTE PTR [ESP + 4], ' '

MOV BYTE PTR [ESP + 5], 0h

MOV EBP, ESP

PUSH EBX

PUSH EBP

call crt\_printf

ADD ESP, 14

MOV ECX, 0

cycle:

XOR EAX, EAX

MOV AL, [EDI+ECX]

push ECX

push EAX

PUSH EAX

push offset format

call crt\_printf

ADD ESP, 8

pop EAX

pop ECX

INC ECX

CMP ESI, ECX

JNE cycle

push offset formatN

call crt\_printf

ADD ESP, 4

POP ECX

POP EDX

POP EBX

POP EAX

POP ESI

POP EDI

RET 12

outputArray endp

start:

MOV ECX, 4

XOR EBX, EBX

MOV ESI, 1

mainLoop:

PUSH ECX

DEC ECX

MOV BL, canUse[ECX]

MOV digit[5], BL

MOV ECX, 6

subLoop:

PUSH ECX

SUB ECX, 2

MOV EDX, ECX

MOV ECX, 2

doubleLoop:

PUSH ESI

PUSH offset digit

PUSH 6

CALL outputArray

INC ESI

MOV EBX, -1

CMP EBX, EDX

JNE swapNear

ROL digit[0], 4

JMP endDoubleLoop

swapNear:

ROL word ptr digit[EDX], 4

endDoubleLoop:

LOOP doubleLoop

POP ECX

LOOP subLoop

endLoop:

MOV digit[0], 10101010b

POP ECX

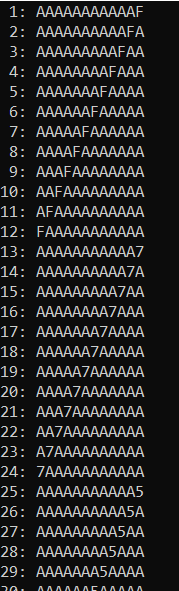
LOOP mainLoop

call crt\_\_getch

push 0

call ExitProcess

end start

****

**Задание 2**

.386 ; Тип процессора

.model flat, stdcall ; Модель памяти и стиль вызова подпрограмм

option casemap: none ; Чувствительность к регистру

; --- Подключение файлов с кодом, макросами, константами, прототипами функций и т.д.

include D:\my\_bstu\arch\_sys\masm32\include\windows.inc

include D:\my\_bstu\arch\_sys\masm32\include\kernel32.inc

include D:\my\_bstu\arch\_sys\masm32\include\user32.inc

include D:\my\_bstu\arch\_sys\masm32\include\msvcrt.inc

; --- Подключение файлов с кодом, макросами, константами, прототипами функций и т.д.

include c:\masm32\include\windows.inc

include c:\masm32\include\kernel32.inc

include c:\masm32\include\user32.inc

include c:\masm32\include\msvcrt.inc

; --- Подключаемые библиотеки ---

includelib c:\masm32\lib\user32.lib

includelib c:\masm32\lib\kernel32.lib

includelib c:\masm32\lib\msvcrt.lib

.code

multiplication proc ; multiplication (char\* a, int n, char\* res); -- Умножение беззнакового 40-байтового числа a на 2^n. Результат в res

PUSHF

PUSHAD

MOV EDI, [ESP + 38] ; Адрес числа a

MOV EBP, [ESP + 46] ; Адрес res

MOV EAX, [ESP + 42] ; Степень двойки, n

MOV EDX, 0h ; Для избежания переполнения при делении

MOV ECX, 8h ; Кол-во бит в байте

IDIV ECX ; Узнаем, на сколько бит сдвинется каждый байт, и сколько нулевых байтов мы запишем в результат

MOV ECX, EAX ; ECX = N / 8 (Количество смещенных байтов)

PUSH EBP

INC EBP ; ADD EBP, ECX ; Смещаем начало числа на байты

PUSH EAX

MOV ECX, 41

moveBytes:

MOV AL, BYTE PTR [EDI + ECX] ; a

MOV BYTE PTR [EBP + ECX], AL ; res

LOOP moveBytes

MOV AL, BYTE PTR [EDI] ; a

MOV BYTE PTR [EBP], AL ; res

AND BYTE PTR [EBP], 01111111b ; Пока что убираем знаковый бит

POP EAX

POP EBP

CMP EDX, 0

JE endLoop

MOV ECX, EDX

MOV ESI, 0h ; Номер текущего байта для для записи

outer\_loop:

PUSH ECX

PUSH EAX

MOV ECX, 42

ADD ECX, EAX

mov AL, byte ptr [EBP+ECX]

mov BL, byte ptr [EBP+ECX]

rol BL, 1

and BL, 00000001b ; сдвинутый бит

shl AL, 1 ; готовый байт

mov byte ptr [EBP+ECX], AL

inner\_loop:

dec ESI

mov AL, byte ptr [EBP+ECX]

mov DL, byte ptr [EBP+ECX]

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 [EBP+ECX], AL

mov BL, DL ; запомнить сдвинутый бит для младшего байта

loop inner\_loop

POP EAX

pop ECX

loop outer\_loop

endLoop:

MOV AL, BYTE PTR [EDI] ; a

AND AL, 10000000b

MOV BYTE PTR [EBP], AL ; res

POPAD

POPF

RET 12

multiplication endp

output\_res proc ; output\_res(char\* a, int n, char\* res) -- Вывод на экран числа a и res в 16-ричном виде

PUSHF

PUSHAD

SUB ESP, 14h ; Буфер для локальной строки формата

MOV BYTE PTR [ESP], 'a'

MOV BYTE PTR [ESP + 1], ' '

MOV BYTE PTR [ESP + 2], '='

MOV BYTE PTR [ESP + 3], ' '

MOV BYTE PTR [ESP + 4], 0h

MOV BYTE PTR [ESP + 5], '%'

MOV BYTE PTR [ESP + 6], '0'

MOV BYTE PTR [ESP + 7], '2'

MOV BYTE PTR [ESP + 8], 'X'

MOV BYTE PTR [ESP + 9], ' '

MOV BYTE PTR [ESP + 10], 0h

MOV BYTE PTR [ESP + 11], 10 ; Символ перехода на новую строку

MOV BYTE PTR [ESP + 12], 13

MOV BYTE PTR [ESP + 13], 0h

; Выведем число a длиною в 42 байт

MOV EBP, ESP ; Сохранили адрес локальной строки формата

PUSH EBP

CALL crt\_printf ; Вывод пояснения о том, что выводится число a

ADD ESP, 4h

ADD EBP, 5h ; Переход к след. строке формата

MOV EDI, [ESP + 58] ; Достали указатель на число a

MOV ESI, 0h ; Индекс для вывода числа a

MOV ECX, 42 ; Выводим 21 раз по 2 байта

output\_a\_again:

PUSH ECX ; printf изменит ECX, поэтому мы сохраним его копию

XOR ECX, ECX

MOV CL, BYTE PTR [EDI + ESI] ;ch

PUSH ECX

PUSH EBP ; Положили в стек адрес строки формата

CALL crt\_printf

ADD ESP, 8h

INC ESI

POP ECX ; Вернули состояние ECX

LOOP output\_a\_again

ADD EBP, 6h ; Вывод символа перехода на новую строку

PUSH EBP

CALL crt\_printf

ADD ESP, 4h

MOV EBP, ESP ; Возвращаемся к первой строке формата

; Выведем число res длиною 42 + n / 4 + 1 байт

MOV BYTE PTR [ESP], 'r' ; Заменим пояснение вывода a на r

PUSH EBP

CALL crt\_printf

ADD ESP, 4h

ADD EBP, 5h ; Перешли к след. строке формата

MOV EAX, [ESP + 62] ; Достали n

MOV ECX, 8h ; Кол-во бит в записи

MOV EDX, 0h ; Для избежания переполнения при делении

IDIV ECX ; EAX = n / 8

ADD EAX, 42 ; Теперь в EAX действительный размер числа res в байтах

MOV ECX, 1h ; Кол-во байт для одновременного вывода

MOV EDX, 0h ; Для избежания переполнения при делении

IDIV ECX ; Теперь в EAX кол-во итераций вывода байт числа res по 2

ADD EAX, 1h ; С округлением вверх

MOV ECX, EAX

MOV EDI, [ESP + 66] ; Достали указатель на число res

MOV ESI, 0h ; Индекс для вывода числа res

output\_res\_again:

PUSH ECX ; printf изменит ECX, поэтому мы сохраним его копию

XOR ECX, ECX

MOV CL, BYTE PTR [EDI + ESI]

PUSH ECX

PUSH EBP ; Положили в стек адрес строки формата

CALL crt\_printf

ADD ESP, 8h

POP ECX ; Вернули состояние ECX

INC ESI

LOOP output\_res\_again

ADD ESP, 14h ; Удалили буфер для локальной строки формата

POPAD

POPF

RET 12

output\_res endp

start:

push offset res

push n

push offset a

call multiplication

PUSH OFFSET res

PUSH n

PUSH OFFSET a

CALL output\_res

call crt\_\_getch ; задержка ввода с клавиатуры

push 0

call ExitProcess

end start

C:\Users\500a5\Desktop\03-12-2020 16-20-05.png

Вывод: Мы изучили команды поразрядной обработки данных.