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)

Лабораторная работа №6
дисциплина «Архитектура вычислительных систем»
по теме «Логические команды и команды сдвига»

Выполнил: студент группы ВТ-31 Макаров Д.С
Проверил: Осипов О.В.

Белгород 2019

Лабораторная работа №6

«Логические команды и команды сдвига»

Цель работы:изучение команд поразрядной обработки данных.

Вариант 9

Задание:

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.

Ход работы

Вывод задания 1

1. AAAAAAAAAA3

2. AAAAAAAAA3A

3. AAAAAAAA3AA

4. AAAAAAA3AAA

5. AAAAAA3AAAA

6. AAAAA3AAAAA

7. AAAA3AAAAAA

8. AAA3AAAAAAA

9. AA3AAAAAAAA

10. A3AAAAAAAAA

11. 3AAAAAAAAAA

12. AAAAAAAAAA5

13. AAAAAAAAA5A

14. AAAAAAAA5AA

15. AAAAAAA5AAA

16. AAAAAA5AAAA

17. AAAAA5AAAAA

18. AAAA5AAAAAA

19. AAA5AAAAAAA

20. AA5AAAAAAAA

21. A5AAAAAAAAA

22. 5AAAAAAAAAA

23. A5AAAAAAAAAA

24. 5AAAAAAAAAAA

25. AAAAAAAAAAA7

26. AAAAAAAAAA7A

27. AAAAAAAAA7AA

28. AAAAAAAA7AAA

29. AAAAAAA7AAAA

30. AAAAAA7AAAAA

31. AAAAA7AAAAAA

32. AAAA7AAAAAAA

33. AAA7AAAAAAAA

34. AA7AAAAAAAAA

35. A7AAAAAAAAAA

36. 7AAAAAAAAAAA

37. AAAAAAAAAAAF

38. AAAAAAAAAAFA

39. AAAAAAAAAFAA

40. AAAAAAAAFAAA

41. AAAAAAAFAAAA

42. AAAAAAFAAAAA

43. AAAAAFAAAAAA

44. AAAAFAAAAAAA

45. AAAFAAAAAAAA

46. AAFAAAAAAAAA

47. AFAAAAAAAAAA

48. FAAAAAAAAAAA

Тестовые данные к заданию 2.

	Входное значение	Вывод функции
1	400000000002	800000000004
2	111111111111	222222222222
3	12345678	24691356

Приложение

Содержимое файла task2.asm

```
.386
.model flat, stdcall
option casemap: none

include c:\masm32\include\windows.inc


.DATA
    format db "%x%x"
    big_number dd 2d,0d,0d,0d,0d,0d,0d,0d,0d,0d,4d
.CODE

mult_arr proc
    ;x=x<<n
    ;4 байта - n
    ;4 байта - x
    ;4 байта - адрес возврата
    push eax
    push ebx
    push ecx
    push edx

    ;16 байт - сохранение переменных
    xor edx,edx
    mov ecx,[esp+20]
outer_loop:
    push ecx
    mov ecx,0
    clc
    inner_loop:
        ;адрес начала массива x
        mov ebx,[esp+28]
        ;x[ecx]
         mov eax,dword ptr [ecx*4+ebx]
        ;прибавить CF
        mov edx,0
        adc edx,0
        or eax,edx
        ;сдвиг влево (старший бит в CF)
        shl eax,1
        ;запись обратно
        mov [ecx*4+ebx],eax
        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 1
    call mult_arr

END START
```

Содержимое файла task1.asm

```
.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 eax
   pop ebp
   pop edx
   pop ecx
   pop ebx
   pop eax
   ret 8
safe_print endp

START:
    xor eax,eax
    xor ecx,ecx
    xor ebp,ebp
    mov ebp,1
    mov ecx,1
```

```
loop_start:
    ;вложенный цикл
    mov bl,alphabet[ecx-1]
    push ecx
    mov ecx,7

    ;установка первого символа
    xor edx,edx
    mov dx,half_num
    mov eax,empty_num
    shl eax, 4
    add al,bl

    ;вывод первого числа
    push ebp
    push offset count_format
    call safe_print
    inc ebp
    push edx
    push offset half_format
    call safe_print
    push eax
    push offset number_format
    call safe_print
    ex_loop:
        shl eax, 4
        add al,000Ah

        push ebp
        push offset count_format
        call safe_print
        inc ebp
        push edx
        push offset half_format
        call safe_print
        push eax
        push offset number_format
        call safe_print

    loop ex_loop

    mov ecx,3
    mov eax,empty_num

    shl edx,4
    add dl,bl
    and edx,0000FFFFh

    push ebp
    push offset count_format
    call safe_print
    inc ebp
    push edx
    push offset half_format
    call safe_print
    push eax
    push offset number_format
    call safe_print

    ex_loop2:
```

```
        shl edx,4
        add dl,000Ah
        and edx,0000FFFFh

        push ebp
        push offset count_format
        call safe_print
        inc ebp
        push edx
        push offset half_format
        call safe_print
        push eax
        push offset number_format
        call safe_print

    loop ex_loop2
    pop ecx
    inc ecx
cmp ecx,5
jl loop_start

END START
```