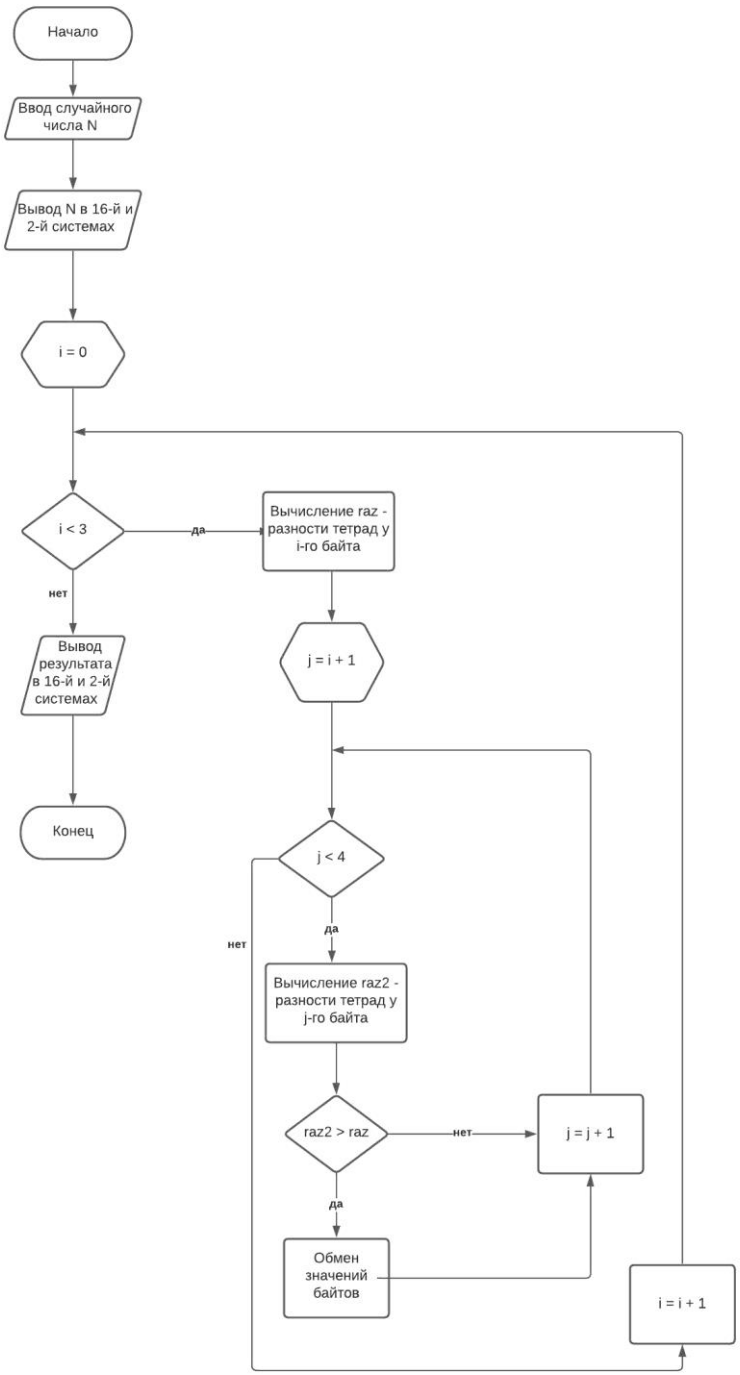
Поразрядные и логические операции

Задание: Назовем характеристикой байта разность между значениями младшей и старшей тетрад. Расположить байты числа в порядке убывания их характеристик.

Task: Let's call the characteristic of a byte the difference between the values of the lower and higher tetrads. Arrange the bytes of the number in descending order of their characteristics.

Блок-схема алгоритма для программы на С



Программа на С

```c
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <locale.h>
int main()
{
	setlocale(LC_ALL, "Russian");
	srand(time(0));
	int N = rand();
	printf("Двоичный код: ");
	for (int i = 0; i < 32; i++)		// Вывод битов
	{
		printf("%d", (N >> (31 - i)) & 1);
	}
	printf("\n");
	printf("Шестнадцатиричный код: %08X\n", N);
	for (int i = 0; i < 3; i++)
	{
		int b = (N >> (24 - 8 * i)) & 0xFF;		// Получение i-го байта
		int raz = abs((b & 0xF) - ((b >> 4) & 0xF));	// Разность тетрад i-го байта
		for (int j = (i + 1); j < 4; j++)
		{
			int b2 = (N >> (24 - 8 * j)) & 0xFF;	// Получение j-го байта
			int raz2 = abs((b2 & 0xF) - ((b2 >> 4) & 0xF));	// Разность тетрад
j-го байта
			if (raz2 > raz)				// Сравнение разностей
			{
				N &= (0xFFFFFF00 << (24 - 8 * i) | 0xFFFFFF00 >> (8 + 8 * i));
// Обнуление i-го байта
				N &= (0xFFFFFF00 << (24 - 8 * j) | 0xFFFFFF00 >> (8 + 8 * j));
	// Обнуление j-го байта
				N |= (b2 << (24 - 8 * i));	// Заносим j-й байт га место i-го
				N |= (b << (24 - 8 * j));	// Заносим i-й байт га место j-го
				b = b2;
				raz = raz2;
			}
		}
	}
	printf("Результат: %08X\n", N);
	printf("Двоичный код: ");
	for (int i = 0; i < 32; i++)		// Вывод битов
	{
		printf("%d", (N >> (31 - i)) & 1);
	}
	return 0;
}
```

Программа на NASM

```
section .data
x dd 0 ; длина числа (количество разрядов)
i dd 0
j dd 0
x1 dd 0
n dd 2 ; основание системы счисления
num dd 0 ; переменная, в которой хранится число
m1 dd 1 ; переменная, в которой хранится разряд числа
m2 dd 1
ms_e dd 10 ; номер символа «перенос строки» в таблице ; ASCII

section .text
global _start
_start:
        mov r14d, 0x0deadbeef        ; заносим число
        mov dword [num], r14d
        mov byte [i], 3
        mov byte [j], 4
        mov eax, dword [num]
;Вывод значения до преобразований
        mov edx, 0
        loop1: ; занесение цифр в стек
                div dword [n]
                push rdx
                inc dword [x1]
                mov edx, 0
                sub eax, 0
```

```
        jnz loop1 ;
loop2:
        pop r10 ; извлечение цифр из стека
        add r10d, 48 ; перевод символов в цифры
        mov [m2], r10d
        dec dword [x1]
        mov rax, 1
        mov rdi, 1
        mov rsi, m2
        mov rdx, 1
        syscall ; вывод на экран
        sub dword [x1], 0 ; проверка конца цикла
        jnz loop2
mov rsi, ms_e
mov rdx, 1
syscall

mov bh, 0                    ; счетчик для первого цикла, i
for1:
mov dl, 8
mov al, bh
mul dl                  ; 8 * i
mov dl, 24
sub dl, al              ; 24 - 8 * i
mov cl, dl
mov r10d,  [num]       ; заносим число
shr r10, cl
and r10, 0x0FF     ; значение левого байта
```

```
mov r11, r10       ; заносим левый байт
and r11, 0x0F        ; (b & 0xF) - правая тетрада
mov rdx, r11

mov r11, r10       ; заносим левый байт
mov cl, 4
shr r11, cl
and r11, 0x0F       ; (b >> 4) & 0xF) - левая тетрада
mov rax, r11

cmp rdx, rax        ; сравниваем тетрады для получения неотрицательной разности
jg lab1
sub rax, rdx
mov rdx, rax
jmp lab2
lab1:
sub rdx, rax
lab2:
mov r11, rdx      ; разность тетрад

mov bl, bh                                        ; счетчик для второго цикла, j
inc bl    ; j = i + 1
for2:

        mov dl, 8
        mov al, bl
        mul dl                  ; 8 * j
```

```
mov dl, 24
sub dl, al              ; 24 - 8 * j
mov cl, dl
mov r12d, [num]       ; заносим число
shr r12, cl
and r12, 0x0FF        ; правый байт


mov r13, r12       ; заносим правый байт
and r13, 0x0F         ; (b2 & 0xF)  правая тетрада
mov rdx, r13

mov r13, r12       ; заносим правый байт
mov cl, 4
shr r13, cl
and r13, 0x0F       ; (b2 >> 4) & 0xF) левая тетрада
mov rax, r13

cmp rdx, rax
jg lab3
sub rax, rdx
mov rdx, rax
jmp lab4
lab3:
sub rdx, rax
lab4:
mov r13, rdx   ; разность тетрад
```

```
cmp r13, r11       ; сравнение разностей
jle lab5


mov r8, 0x0FFFFFF00
mov dl, 8
mov al, bh
mul dl                  ; 8 * i
mov dl, 24
sub dl, al             ; 24 - 8 * i
mov cl, dl
shl r8, cl              ;  0xFFFFFF00 << (24 - 8 * i)


mov r9, 0x0FFFFFF00
mov dl, 8
mov al, bh
mul dl          ; 8 * i
add al, 8       ; 8 + 8 * i
mov cl, al
shr r9, cl      ; 0xFFFFFF00 >> (8 + 8 * i)

or r8, r9
and  [num], r8d      ; N &= (0xFFFFFF00 << (24 - 8 * i) | 0xFFFFFF00
>> (8 + 8 * i))

mov r8, 0x0FFFFFF00
mov dl, 8
mov al, bl
```

```
mul dl                  ; 8 * j
mov dl, 24
sub dl, al              ; 24 - 8 * j
mov cl, dl
shl r8, cl              ;  0xFFFFFF00 << (24 - 8 * j)


mov r9, 0x0FFFFFF00
mov dl, 8
mov al, bl
mul dl          ; 8 * j
add al, 8       ; 8 + 8 * j
mov cl, al
shr r9, cl      ; 0xFFFFFF00 >> (8 + 8 * j)

or r8, r9
and [num], r8d     ; N &= (0xFFFFFF00 << (24 - 8 * j) | 0xFFFFFF00
>> (8 + 8 * j))

mov dl, 8
mov al, bh
mul dl                  ; 8 * i
mov dl, 24
sub dl, al              ; 24 - 8 * i
mov cl, dl
mov r15, r12
shl r15, cl
or [num], r15d     ; N |= (b2 << (24 - 8 * i))
```

```
                mov dl, 8
                mov al, bl
                mul dl                  ; 8 * j
                mov dl, 24
                sub dl, al              ; 24 - 8 * j
                mov cl, dl
                mov r15, r10
                shl r15, cl
                or [num], r15d      ; N |= (b << (24 - 8 * j))

                mov r10, r12   ;b = b2
                mov r11, r13   ; raz = raz2;

                lab5:
                inc bl
                cmp bl, [j]
                jl for2

inc bh
cmp bh, [i]
jl for1

mov eax, [num]
;Вывод получившегося значения
        mov edx, 0
        loop3: ; занесение цифр в стек
                div dword [n]
                push rdx
                inc dword [x1]
```

```
        mov edx, 0
        sub eax, 0
        jnz loop3 ;
loop4:
        pop r10 ; извлечение цифр из стека
        add r10d, 48 ; перевод символов в цифры
        mov [m2], r10d
        dec dword [x1]
        mov rax, 1
        mov rdi, 1
        mov rsi, m2
        mov rdx, 1
        syscall ; вывод на экран
        sub dword [x1], 0 ; проверка конца цикла
        jnz loop4
mov rsi, ms_e
mov rdx, 1
syscall
mov eax, 60
xor rdi, rdi
syscall
```

Дизассемблерный листинг программы на C

```
.LC0:
        .string "Russian"
.LC1:
        .string
"\320\224\320\262\320\276\320\270\321\207\320\275\321\213\320\271
\320\272\320\276\320\264: "
.LC2:
        .string "%d"
.LC3:
        .string
"\320\250\320\265\321\201\321\202\320\275\320\260\320\264\321\206\320\260\3
21\202\320\270\321\200\320\270\321\207\320\275\321\213\320\271
\320\272\320\276\320\264: %08X\n"
.LC4:
        .string
"\320\240\320\265\320\267\321\203\320\273\321\214\321\202\320\260\321\202:
%08X\n"
main:
        push    rbp
        mov     rbp, rsp
        sub     rsp, 48
        mov     esi, OFFSET FLAT:.LC0
        mov     edi, 6
        call    setlocale
        mov     edi, 0
        call    time
        mov     edi, eax
        call    srand
        call    rand
        mov     DWORD PTR [rbp-4], eax
```

```
        mov     edi, OFFSET FLAT:.LC1
        mov     eax, 0
        call    printf
        mov     DWORD PTR [rbp-8], 0
        jmp     .L2
.L3:
        mov     eax, 31
        sub     eax, DWORD PTR [rbp-8]
        mov     edx, DWORD PTR [rbp-4]
        mov     ecx, eax
        sar     edx, cl
        mov     eax, edx
        and     eax, 1
        mov     esi, eax
        mov     edi, OFFSET FLAT:.LC2
        mov     eax, 0
        call    printf
        add     DWORD PTR [rbp-8], 1
.L2:
        cmp     DWORD PTR [rbp-8], 31
        jle     .L3
        mov     edi, 10
        call    putchar
        mov     eax, DWORD PTR [rbp-4]
        mov     esi, eax
        mov     edi, OFFSET FLAT:.LC3
        mov     eax, 0
        call    printf
        mov     DWORD PTR [rbp-12], 0
```

```
        jmp     .L4
.L8:
        mov     eax, 3
        sub     eax, DWORD PTR [rbp-12]
        sal     eax, 3
        mov     edx, DWORD PTR [rbp-4]
        mov     ecx, eax
        sar     edx, cl
        mov     eax, edx
        and     eax, 255
        mov     DWORD PTR [rbp-16], eax
        mov     eax, DWORD PTR [rbp-16]
        and     eax, 15
        mov     edx, eax
        mov     eax, DWORD PTR [rbp-16]
        sar     eax, 4
        and     eax, 15
        mov     ecx, eax
        mov     eax, edx
        sub     eax, ecx
        cdq
        xor     eax, edx
        mov     DWORD PTR [rbp-20], eax
        sub     DWORD PTR [rbp-20], edx
        mov     eax, DWORD PTR [rbp-12]
        add     eax, 1
        mov     DWORD PTR [rbp-24], eax
        jmp     .L5
.L7:
```

```
mov     eax, 3
sub     eax, DWORD PTR [rbp-24]
sal     eax, 3
mov     edx, DWORD PTR [rbp-4]
mov     ecx, eax
sar     edx, cl
mov     eax, edx
and     eax, 255
mov     DWORD PTR [rbp-32], eax
mov     eax, DWORD PTR [rbp-32]
and     eax, 15
mov     edx, eax
mov     eax, DWORD PTR [rbp-32]
sar     eax, 4
and     eax, 15
mov     ecx, eax
mov     eax, edx
sub     eax, ecx
cdq
xor     eax, edx
mov     DWORD PTR [rbp-36], eax
sub     DWORD PTR [rbp-36], edx
mov     eax, DWORD PTR [rbp-36]
cmp     eax, DWORD PTR [rbp-20]
jle     .L6
mov     eax, 3
sub     eax, DWORD PTR [rbp-12]
sal     eax, 3
mov     edx, -256
```

```
mov     ecx, eax
sal     edx, cl
mov     eax, DWORD PTR [rbp-12]
add     eax, 1
sal     eax, 3
mov     esi, -256
mov     ecx, eax
shr     esi, cl
mov     eax, esi
or      edx, eax
mov     eax, DWORD PTR [rbp-4]
and     eax, edx
mov     DWORD PTR [rbp-4], eax
mov     eax, 3
sub     eax, DWORD PTR [rbp-24]
sal     eax, 3
mov     edx, -256
mov     ecx, eax
sal     edx, cl
mov     eax, DWORD PTR [rbp-24]
add     eax, 1
sal     eax, 3
mov     esi, -256
mov     ecx, eax
shr     esi, cl
mov     eax, esi
or      edx, eax
mov     eax, DWORD PTR [rbp-4]
and     eax, edx
```

```
        mov     DWORD PTR [rbp-4], eax
        mov     eax, 3
        sub     eax, DWORD PTR [rbp-12]
        sal     eax, 3
        mov     edx, DWORD PTR [rbp-32]
        mov     ecx, eax
        sal     edx, cl
        mov     eax, edx
        or      DWORD PTR [rbp-4], eax
        mov     eax, 3
        sub     eax, DWORD PTR [rbp-24]
        sal     eax, 3
        mov     edx, DWORD PTR [rbp-16]
        mov     ecx, eax
        sal     edx, cl
        mov     eax, edx
        or      DWORD PTR [rbp-4], eax
        mov     eax, DWORD PTR [rbp-32]
        mov     DWORD PTR [rbp-16], eax
        mov     eax, DWORD PTR [rbp-36]
        mov     DWORD PTR [rbp-20], eax
.L6:
        add     DWORD PTR [rbp-24], 1
.L5:
        cmp     DWORD PTR [rbp-24], 3
        jle     .L7
        add     DWORD PTR [rbp-12], 1
.L4:
        cmp     DWORD PTR [rbp-12], 2
```

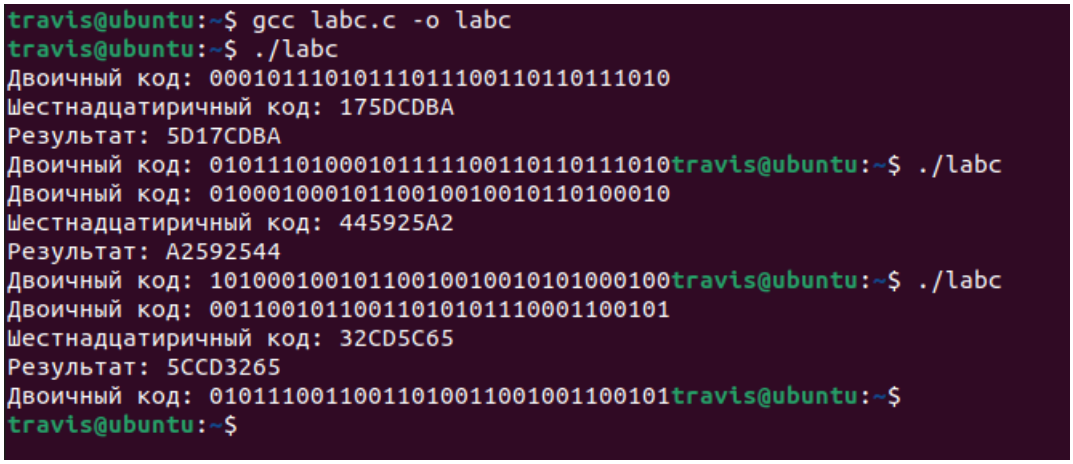
```
        jle     .L8
        mov     eax, DWORD PTR [rbp-4]
        mov     esi, eax
        mov     edi, OFFSET FLAT:.LC4
        mov     eax, 0
        call    printf
        mov     edi, OFFSET FLAT:.LC1
        mov     eax, 0
        call    printf
        mov     DWORD PTR [rbp-28], 0
        jmp     .L9
.L10:
        mov     eax, 31
        sub     eax, DWORD PTR [rbp-28]
        mov     edx, DWORD PTR [rbp-4]
        mov     ecx, eax
        sar     edx, cl
        mov     eax, edx
        and     eax, 1
        mov     esi, eax
        mov     edi, OFFSET FLAT:.LC2
        mov     eax, 0
        call    printf
        add     DWORD PTR [rbp-28], 1
.L9:
        cmp     DWORD PTR [rbp-28], 31
        jle     .L10
        mov     eax, 0
        leave
```

ret

Анализ программ:

Дизассемблированная программа не использует регистры r8 – r15 и работает со стеком памяти, что позволяет экономить память. Программа, написанная на NASM оказалась почти в 11 раз быстрее программы на C. Стоит заметить, что дизассемблированная программа использует не системные вызовы (syscalls), а функции из библиотеки C (call putchar, call printf, …).

Скриншот выводов программы на C



Скриншот выводов программы на NASM