Операция "стрелка Пирса"

```
int pierce_arrow(int a, int b) {
   int t = ~(a | b);
   return t;
}
```

```
section .text
global pierce_arrow
pierce_arrow:
        push    ebp
        mov     ebp, esp
        mov     eax, DWORD [ebp+12] ; (1)
        or      eax, DWORD [ebp+8]  ; (2)
        not     eax                 ; (3)
        pop     ebp
        ret
```

Операции арифметического сдвига

```
char upndown(char x) {
   return (x << 8) >> 8;
}
```

```
section .text
global upndown
upndown:
        push    ebp
        mov     ebp, esp
        movsx   eax, BYTE [ebp+8]
        sal     eax, 8
        sar     eax, 8
        pop     ebp
        ret
```

Инструкции вращения

```
unsigned sha256_f1(unsigned x) {
   unsigned t;
   t  = ((x >> 2)  | (x << ((sizeof(x) << 3) - 2)));  // (1)
   t ^= ((x >> 13) | (x << ((sizeof(x) << 3) - 13))); // (2)
   t ^= ((x >> 22) | (x << ((sizeof(x) << 3) - 22))); // (3)
   return t;
}
```

```
global sha256_f1
sha256_f1:
   push    ebp
   mov     ebp, esp
   mov     edx, DWORD [ebp+8] ; (1)
   pop     ebp                ; (2)
   mov     eax, edx           ; (3)
   mov     ecx, edx           ; (4)
   ror     eax, 13            ; (5)
   ror     ecx, 2             ; (6)
   xor     eax, ecx           ; (7)
   ror     edx, 22            ; (8)
   xor     eax, edx           ; (9)
   ret
```

Использование битовых операций

```
struct omg {
   int a : 3;
   int b : 5;
   int c : 2;

   unsigned cntr: 31;

   int sum : 8;
};

void f(struct omg *p) {
   p->cntr++;                      // 1
   p->b = (p->c << 3) | (p->a); // 2
   p->sum = p->a + p->b + p->c; // 3
}

static struct omg omg_p =
   {0, 0, 0, 0, 0};

int main() {
   f(&omg_p);
   return 0;
}
```

```
section .bss
   omg_s resb 12

section .text
global f
f:
        push    ebp                 ;  (1)
        mov     ebp, esp            ;  (2)
        push    esi                 ;  (3)
        push    ebx                 ;  (4)
        mov     esi, DWORD [ebp+8]  ;  (5)
        mov     eax, DWORD [esi+4]  ;  (6)
        lea     edx, [eax+1]        ;  (7)
        and     eax, -2147483648    ;  (8)
        and     edx,  2147483647    ;  (9)
        or      eax, edx            ; (10)
        mov     DWORD [esi+4], eax  ; (11)
        movzx   ebx, BYTE [esi+1]   ; (12)
        sal     ebx, 6              ; (13)
        sar     bl, 6               ; (14)
        lea     ecx, [0+ebx*8]      ;
        movzx   edx, BYTE [esi]
        mov     eax, edx
        and     edx, 7
        sal     eax, 5
        sar     al, 5
        or      ecx, eax
        sal     ecx, 3
        sar     cl, 3
        lea     eax, [0+ecx*8]
        or      edx, eax
        mov     BYTE [esi], dl
        sal     edx, 5
        sar     dl, 5
        add     ebx, edx
        add     ebx, ecx
        mov     BYTE [esi+8], bl
        pop     ebx
        pop     esi
        pop     ebp
        ret

global CMAIN
CMAIN:
        lea     ecx, [esp+4]
        and     esp, -16
        push    DWORD [ecx-4]
        push    ebp
        mov     ebp, esp
        push    ecx
        sub     esp, 4
        mov     DWORD [esp], omg_s
        call    f
        PRINT_DEC 4, eax
        NEWLINE
        mov     eax, 0
        add     esp, 4
        pop     ecx
        pop     ebp
        lea     esp, [ecx-4]
        ret
```