Лекция 6

26 февраля

Операции над отдельными битами

- Логические инструкции
  - **AND**
  - **OR**
  - **XOR**
  - **NOT**
- Сдвиги и вращения
  - **SAR**
  - **SHR**
  - **SAL, SHL**
  - **ROR**
  - **ROL**
  - **RCR**
  - **RCL**
- Битовые и байтовые
  - BT
  - BTS
  - BTR
  - BTC
  - BSF
  - BSR
  - SET*cc*
  - TEST

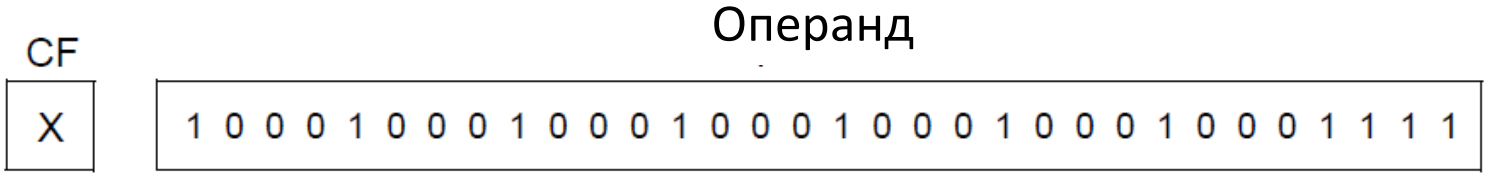
Пример

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

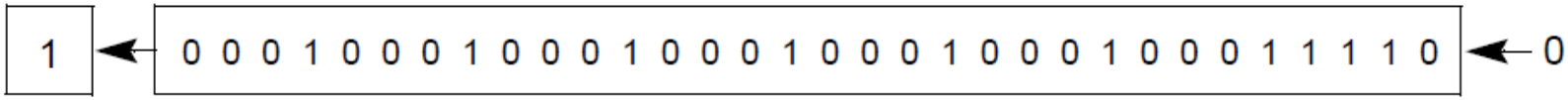
```
section .text
global pierce_arrow
pierce_arrow:
   push    ebp
   mov     ebp, esp
   mov     eax, dword [ebp+12]  ; (1)
   or      eax, dword [ebp+8]   ; (2)
   not     eax                  ; (3)
   pop     ebp
   ret
```

Сдвиг влево

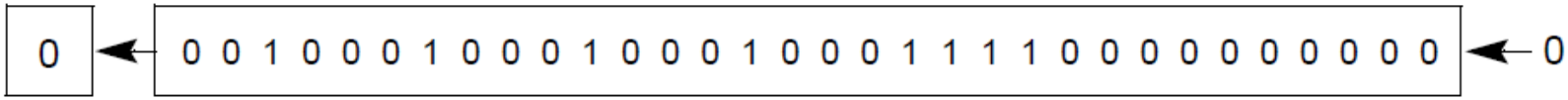
Начальное состояние



После сдвига на один разряд: SHL/SAL



После сдвига на 10 разрядов: SHL/SAL



Логический сдвиг вправо

Начальное состояние

Операнд

CF

1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 1 | X

После сдвига на один разряд: SHR

0 → 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 1 1 → 1

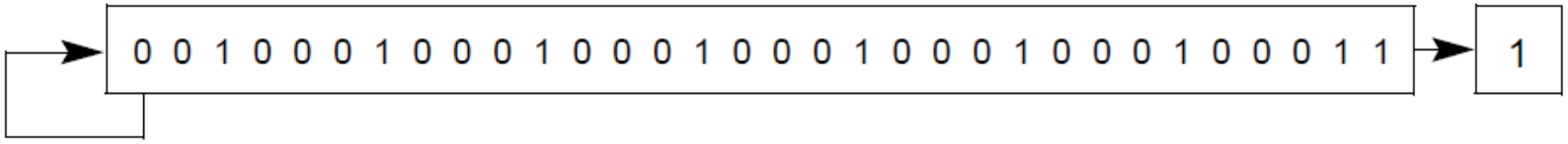
После сдвига на 10 разрядов: SHR

0 → 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 → 0

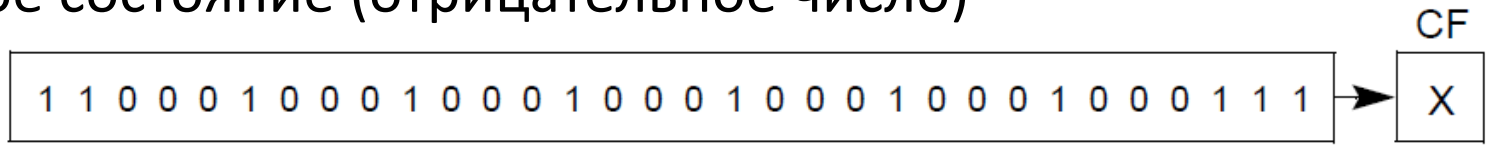
Арифметический сдвиг вправо

Начальное состояние (положительное число)

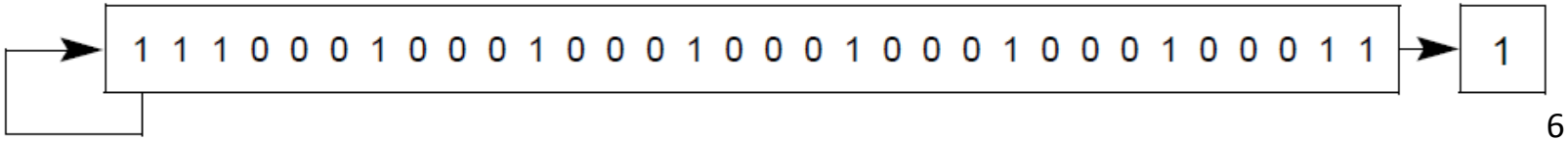
После сдвига на один разряд: SAR



Начальное состояние (отрицательное число)



После сдвига на один разряд: SAR



Пример

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

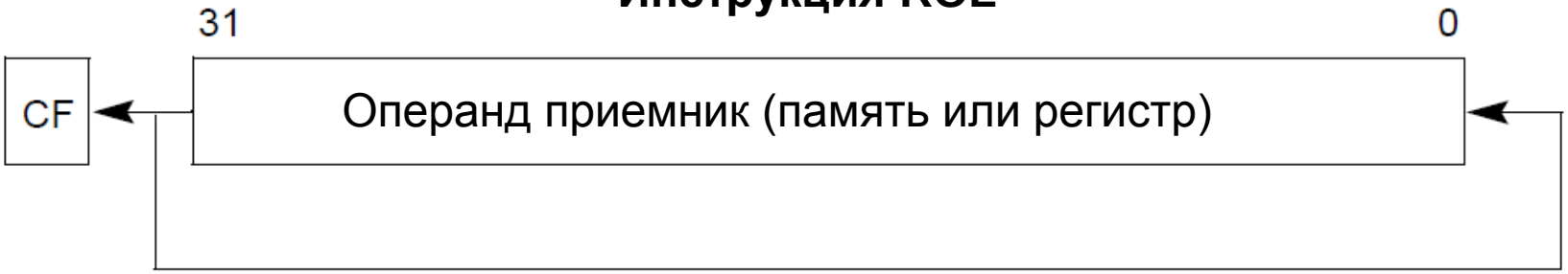
Что вернет `upndown(42);` ?

Пример

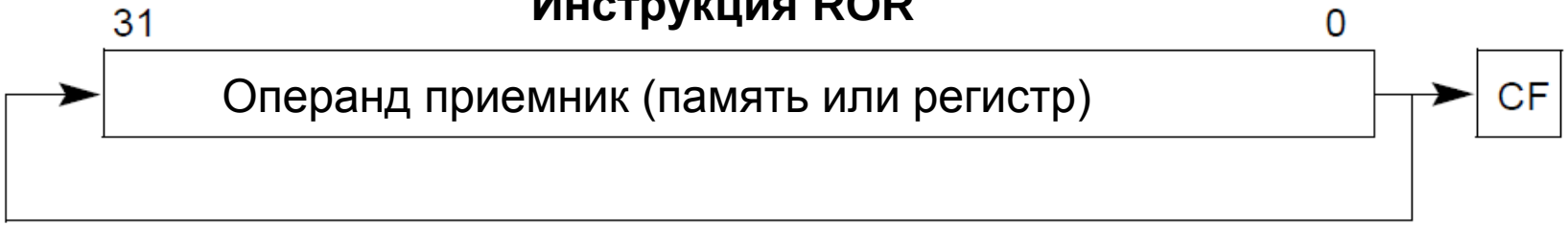
```
char upndown(char x) {
   return (x << 8) >> 8;
}
```

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

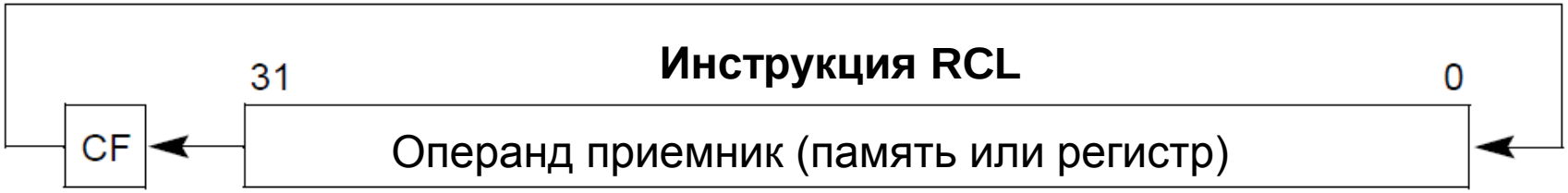
Инструкция ROL



Инструкция ROR



Инструкция RCL



Инструкция RCR

Пример

В 2008 году Intel были заявлены новые команды, поддерживающие шифрование AES: AESENC, AESDEC, AESENCLAST, AESKEYGENASSIST, …

```
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
```

Криптографические хеш-функции H(X):

- Необратимость
  H(X) = m, m – задано
- Стойкость к коллизиям первого рода
  H(M) = H(N), M – задано
- Стойкость к коллизиям второго рода
  $H(M_1) = H(M_2)$

Обратная задача

```
static int a, b, c, d;

???
```

```
mov     eax, dword [b] ; (1)
mov     edx, dword [c] ; (2)
or      al, -1         ; (3)
sal     eax, 3         ; (4)
add     edx, eax       ; (5)
mov     dword [a], eax ; (6)
mov     eax, edx       ; (7)
sar     edx, 31        ; (8)
idiv    dword [d]      ; (9)
mov     dword [c], eax ; (10)
```

Операции над отдельными битами

- Логические инструкции
  - AND
  - OR
  - XOR
  - NOT
- Сдвиги и вращения
  - SAR
  - SHR
  - SAL, SHL
  - ROR
  - ROL
  - RCR
  - RCL
- Битовые и байтовые
  - **BT**
  - **BTS**
  - **BTR**
  - **BTC**
  - **BSF**
  - **BSR**
  - **SET*cc***
  - **TEST**

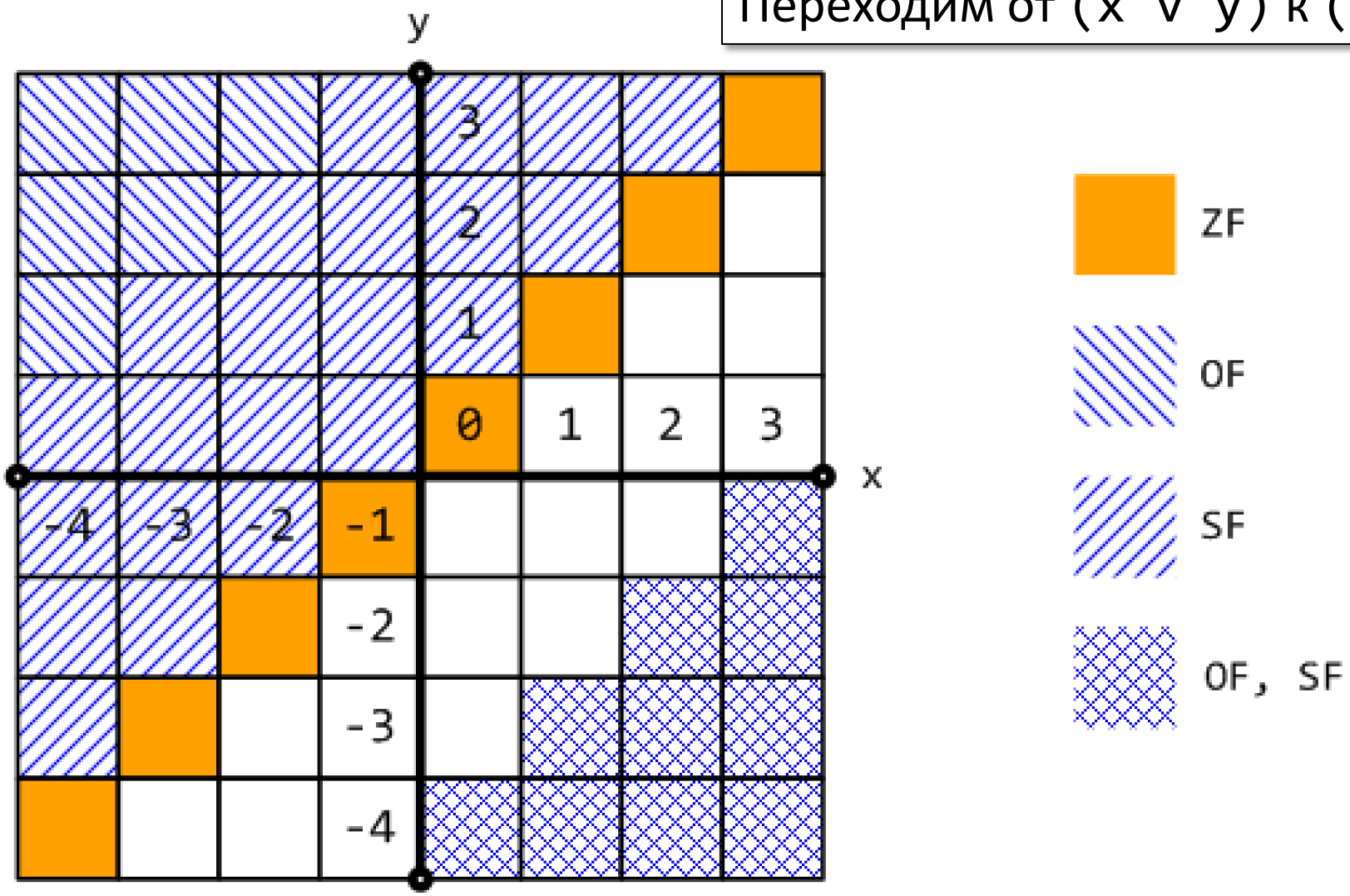
Далее…

- Арифметика
- Операции над машинными словами
- Реализация управляющих операторов языка Си
  - Условная передача данных
  - Организация циклов
  - Оператор switch

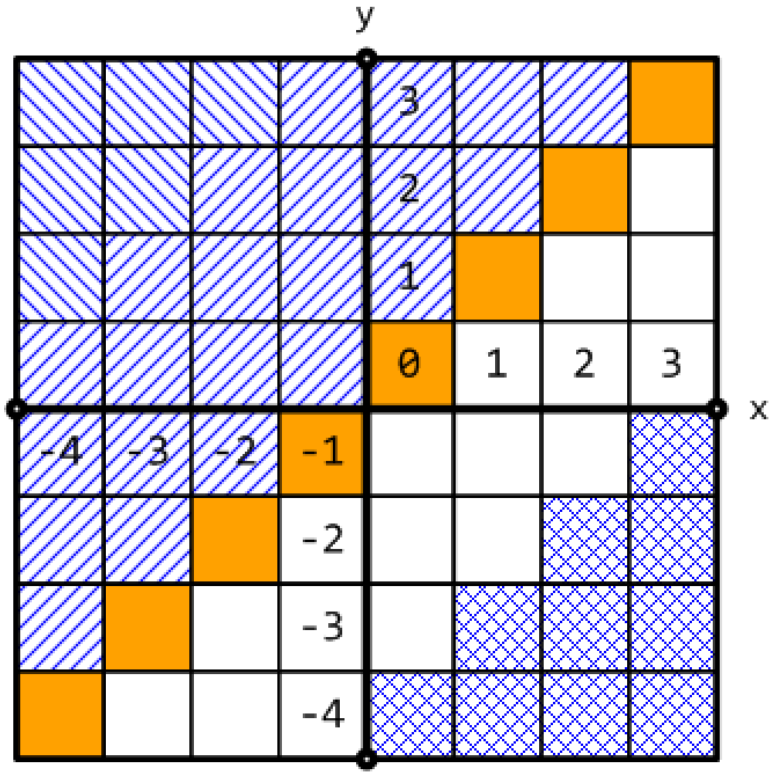
J*cc*	Условие	Описание
JE	ZF	Равно / Ноль
JNE	~ZF	Не равно / Не ноль
JS	SF	Отрицательное число
JNS	~SF	Неотрицательное число
JG	~(SF^OF)&~ZF	Больше (знаковые числа)
JGE	~(SF^OF)	Больше либо равно (знаковые числа)
JL	(SF^OF)	Меньше (знаковые числа)
JLE	(SF^OF)|ZF	Меньше либо равно (знаковые числа)
JA	~CF&~ZF	Больше (числа без знака)
JB	CF	Меньше (числа без знака)

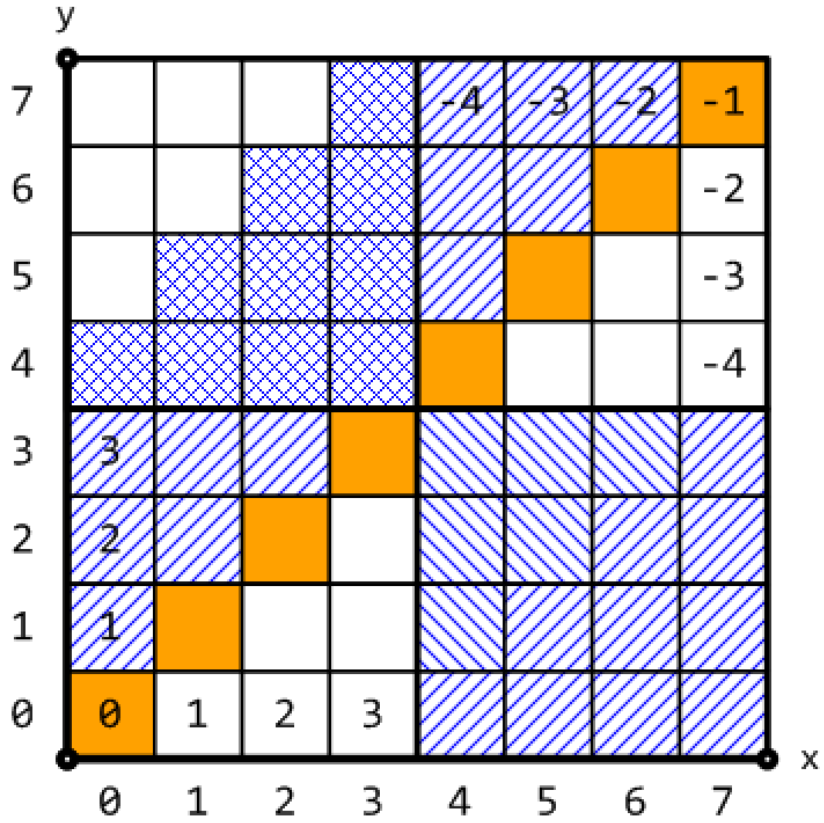
Сравнение знаковых чисел

Переходим от (x v y) к (x-y)

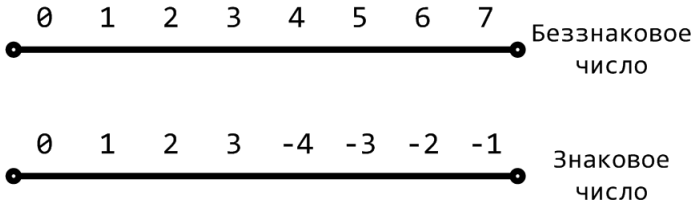


Сравнение: со знаком и без





000	001	010	011	100	101	110	111



```c
int absdiff(int x, int y) {
    int result;
    if (x > y) {
      result = x-y;
    } else {
      result = y-x;
    }
    return result;
}
```

```
absdiff:
   push ebp
   mov  ebp, esp
   mov  edx, dword [8 + ebp]  ; (1)
   mov  eax, dword [12 + ebp] ; (2)
   cmp  edx, eax              ; (3)
   jle  .L6                   ; (4)
   sub  edx, eax              ; (5)
   mov  eax, edx              ; (6)
   jmp  .L7                   ; (7)
.L6:                          ; (8)
   sub  eax, edx              ; (9)
.L7:                          ; (10)
   pop  ebp
   ret
```

```c
int goto_ad(int x, int y) {
  int result;
  if (x <= y) goto Else;
  result = x-y;
  goto Exit;
Else:
  result = y-x;
Exit:
  return result;
}
```

```asm
absdiff:
   push ebp
   mov  ebp, esp
   mov  edx, dword [8 + ebp]  ; (1)
   mov  eax, dword [12 + ebp] ; (2)
   cmp  edx, eax              ; (3)
   jle  .L6                   ; (4)
   sub  edx, eax              ; (5)
   mov  eax, edx              ; (6)
   jmp  .L7                   ; (7)
.L6:                          ; (8)
   sub  eax, edx              ; (9)
.L7:                          ; (10)
   pop  ebp
   ret
```

Условная передача данных

```
val = Test ? Then_Expr : Else_Expr;

val = x>y ? x-y : y-x;
```

```
   nt = !(Test);
   if (nt) goto Else;
   val = Then_Expr;
   goto Done;
Else:
   val = Else_Expr;
Done:
   ...
```

```
tmp_val = Then_Expr;
val = Else_Expr;
t = Test;
if (t) val = tmp_val;
```

```c
int absdiff(int x, int y) {
    int result;
    if (x > y) {
      result = x-y;
    } else {
      result = y-x;
    }
    return result;
}
```

```c
int absdiff(int x, int y) {
    return (x > y)? x-y: y-x;
}
```

Более короткая запись …

Регистр	Значение
edi	**x**
esi	**y**

```
absdiff:
   ...
   mov   edx, edi
   sub   edx, esi   ; tmp_val:edx = x-y
   mov   eax, esi
   sub   eax, edi   ; result:eax = y-x
   cmp   edi, esi   ; Compare x:y
   cmovg eax, edx   ; If >, result:eax = tmp_val:edx
   ...
```

Оператор do-while

```
int pcount_do(unsigned x) {
  int result = 0;
  do {
    result += x & 0x1;
    x >>= 1;
  } while (x);
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
  return result;
}
```

Оператор do-while

Регистр	Значение
edx	**x**
ecx	**result**

```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
  return result;
}
```

```
   mov ecx, 0     ;   result = 0
.L2:              ; loop:
   mov eax, edx
   and eax, 1     ;   t = x & 1
   add ecx, eax   ;   result += t
   shr edx, 1     ;   x >>= 1
   jne .L2        ;   If !0, goto loop
```

Оператор while

```
int pcount_while(unsigned x) {
  int result = 0;
  while (x) {
    result += x & 0x1;
    x >>= 1;
  }
  return result;
}
```

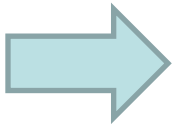
```
int pcount_do(unsigned x) {
  int result = 0;
loop:
  if (!x) goto done;
  result += x & 0x1;
  x >>= 1;
  goto loop;
done:
  return result;
}
```

```
int pcount_do(unsigned x) {
  int result = 0;
  if (!x) goto done;
loop:
  result += x & 0x1;
  x >>= 1;
  if (x)
    goto loop;
done:
  return result;
}
```

Оператор for

```
#define WSIZE 8*sizeof(int)

int pcount_for(unsigned x) {
  int i;
  int result = 0;
  for (i = 0; i < WSIZE; i++) {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  return result;
}
```



```
int pcount_for_gt(unsigned x) {
  int i;
  int result = 0;
  i = 0;
  if (!(i < WSIZE))
    goto done;
 loop:
  {
    unsigned mask = 1 << i;
    result += (x & mask) != 0;
  }
  i++;
  if (i < WSIZE)
    goto loop;
 done:
  return result;
}
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

Регистр	Значение
ecx	**x**
edx	**p_pred**
ebx	**pred**
eax	**res**

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

Регистр	Значение
ecx	x
edx	p_pred
ebx	pred
eax	res

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

Регистр	Значение
ecx	x
edx	p_pred
ebx	pred
eax	res

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

```c
int fib(int x) { // x >= 1
  int i;
  int p_pred = 0;
  int pred = 1;
  int res = 1;
  x--;
  for (i = 0; i < x; i++) {
     res = p_pred + pred;
     p_pred = pred;
     pred = res;
  }
  return res;
}
```

Регистр	Значение
ecx	x
edx	p_pred
ebx	pred
eax	res

```
fib:
   push  ebp
   mov   ebp, esp
   push  ebx

   mov   ecx, dword [ebp + 8] ; x
   xor   edx, edx             ; p_pred
   mov   ebx, 1               ; pred
   mov   eax, 1               ; res
   dec   ecx

   jecxz .end
.loop:
   lea   eax, [edx + ebx]
   mov   edx, ebx
   mov   ebx, eax
   loop .loop

.end:
   pop ebx
   pop ebp
   ret
```

Обратная задача

```
f:
   ...
   mov edx, dword [ebp+8] ; (1)
   mov eax, 0             ; (2)
   test edx, edx          ; (3)
   je .L7                 ; (4)
 .L10:                    ;
   xor eax, edx           ; (5)
   shr edx, 1             ; (6)
   jne .L10               ; (7)
 .L7:                     ;
   and eax, 1             ; (8)
   ...
```

```
int f(unsigned x) {
   int val = 0;
   while (________) {
      ____________;
   }
   return ________;
}
```