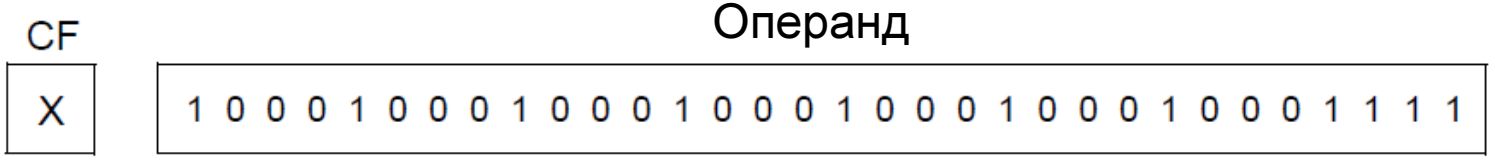
Лекция 7

2 марта

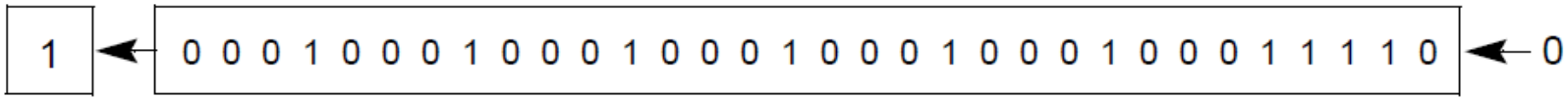
```
int pierce_arrow(int a, int b)
{
   int t = ~(a | b);
   return t;
}
```

```
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
```

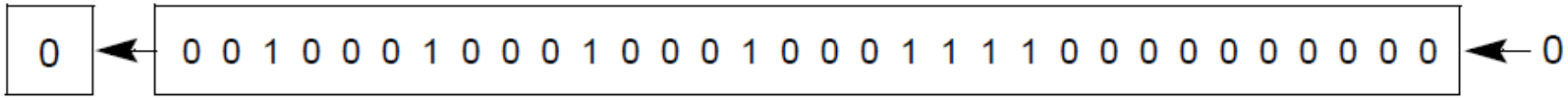
Начальное состояние



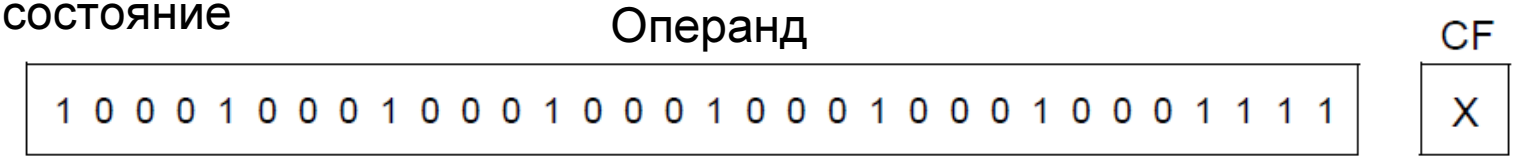
После сдвига на один разряд: SHL/SAL



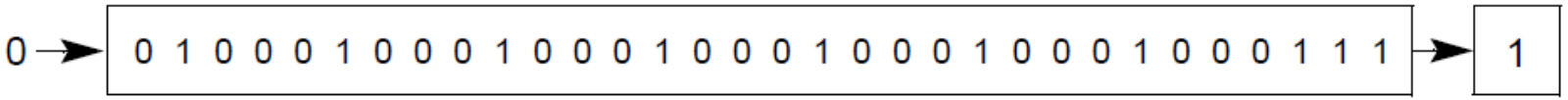
После сдвига на 10 разрядов: SHL/SAL



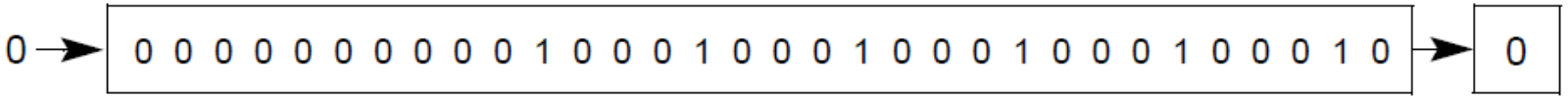
Начальное состояние



После сдвига на один разряд: SHR

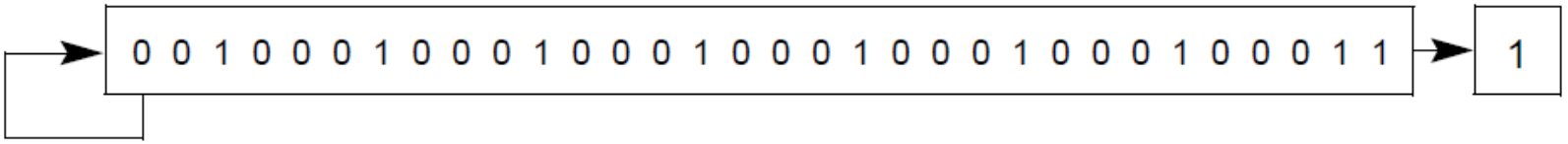


После сдвига на 10 разрядов: SHR

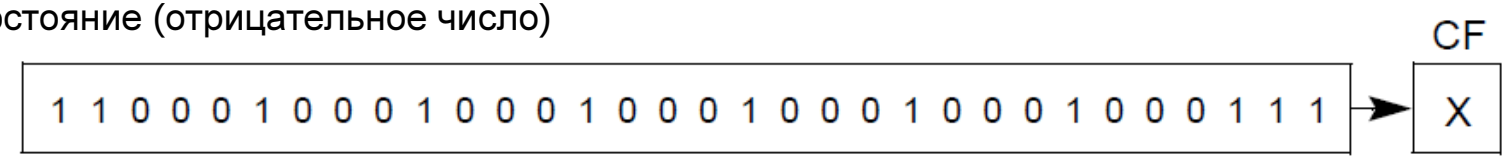


Начальное состояние (положительное число)

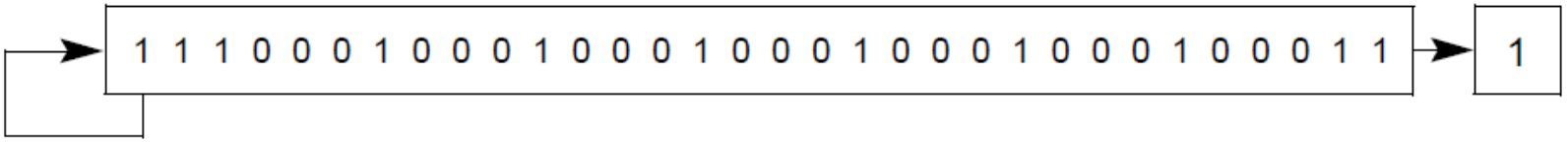
После сдвига на один разряд: SAR



Начальное состояние (отрицательное число)



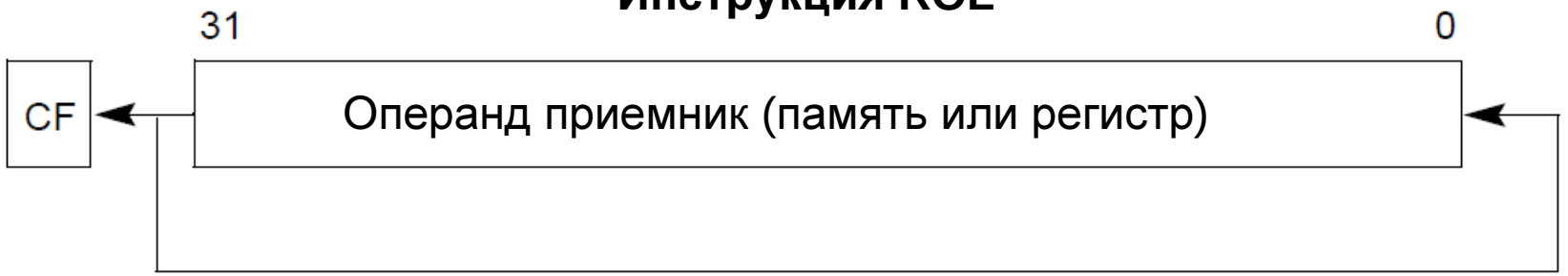
После сдвига на один разряд: SAR



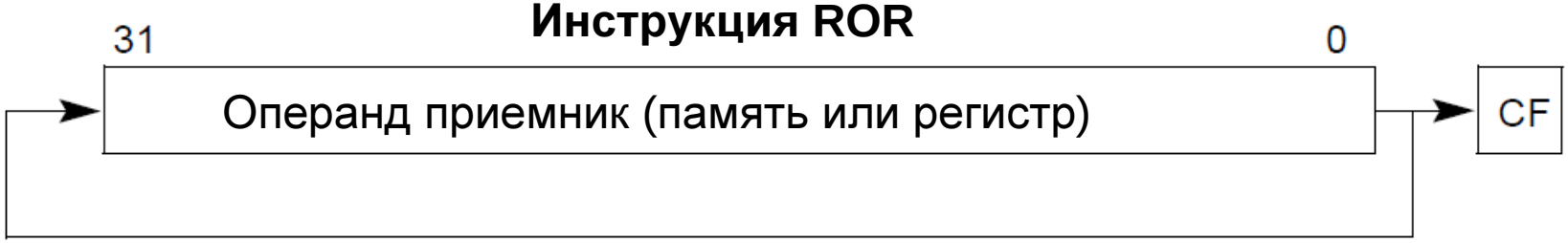
```
char upndown(char x) {
   return (x << 8) >> 8;
}
```

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

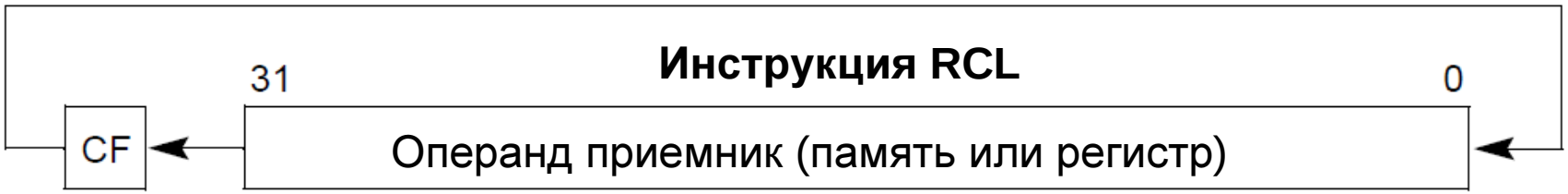
Инструкция ROL



Инструкция ROR



Инструкция RCL



Инструкция RCR

```
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
```

```
int arith(int x,
          int y,
          int z) {
   int t1 = x + y;
   int t2 = z * 48;
   int t3 = t1 & 0xFFFF;
   int t4 = t2 * t3;
   return t4;
}
```

```
; начало функции пропущено
mov eax, dword [ebp + 16] ; (1)
lea eax, [eax + 2 * eax]  ; (2)
sal eax, 4                ; (3)
mov edx, dword [ebp + 12] ; (4)
add edx, dword [ebp + 8]  ; (5)
and edx, 65535            ; (6)
imull eax, edx            ; (7)
; конец функции пропущен
```