

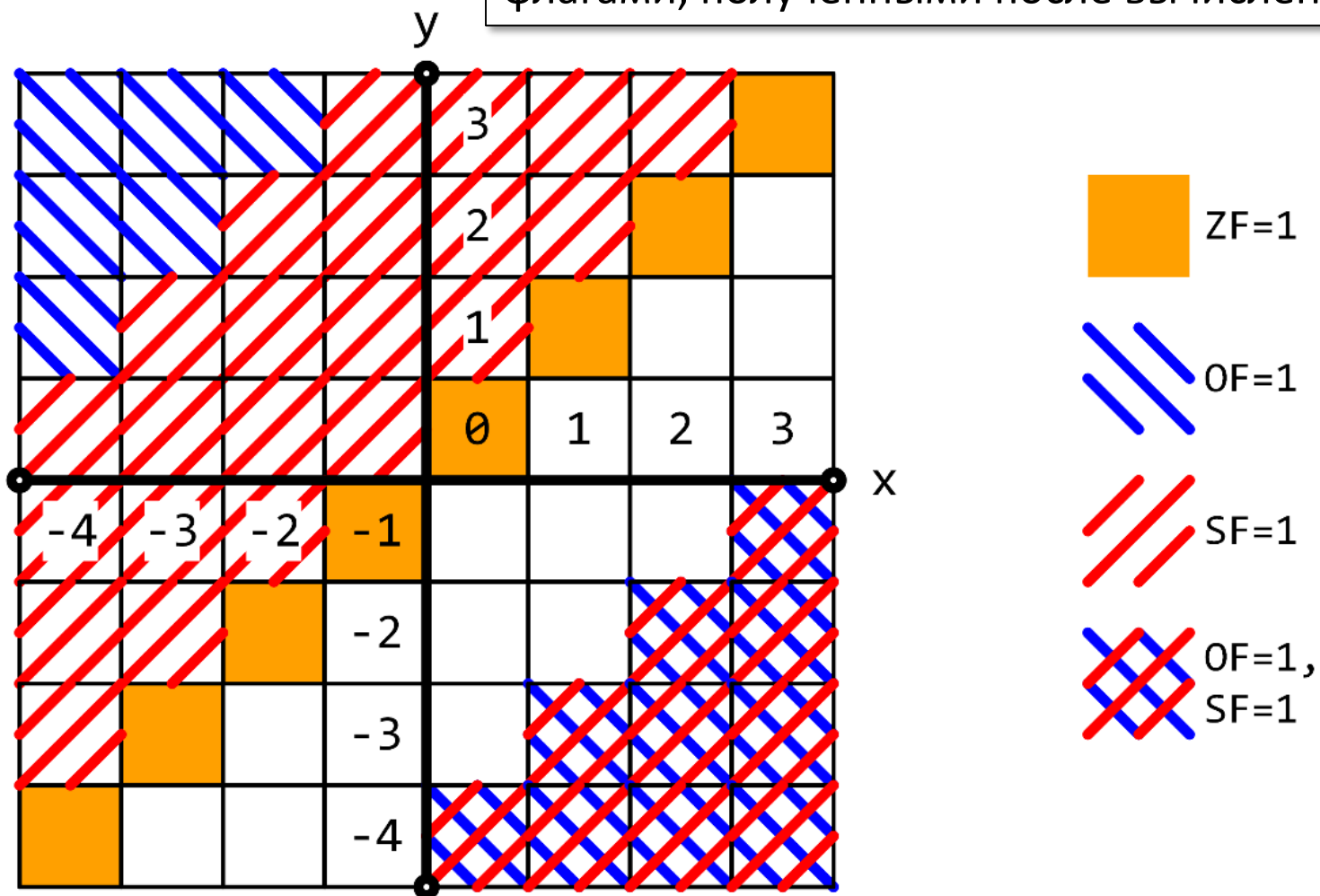
Лекция 7

29 февраля

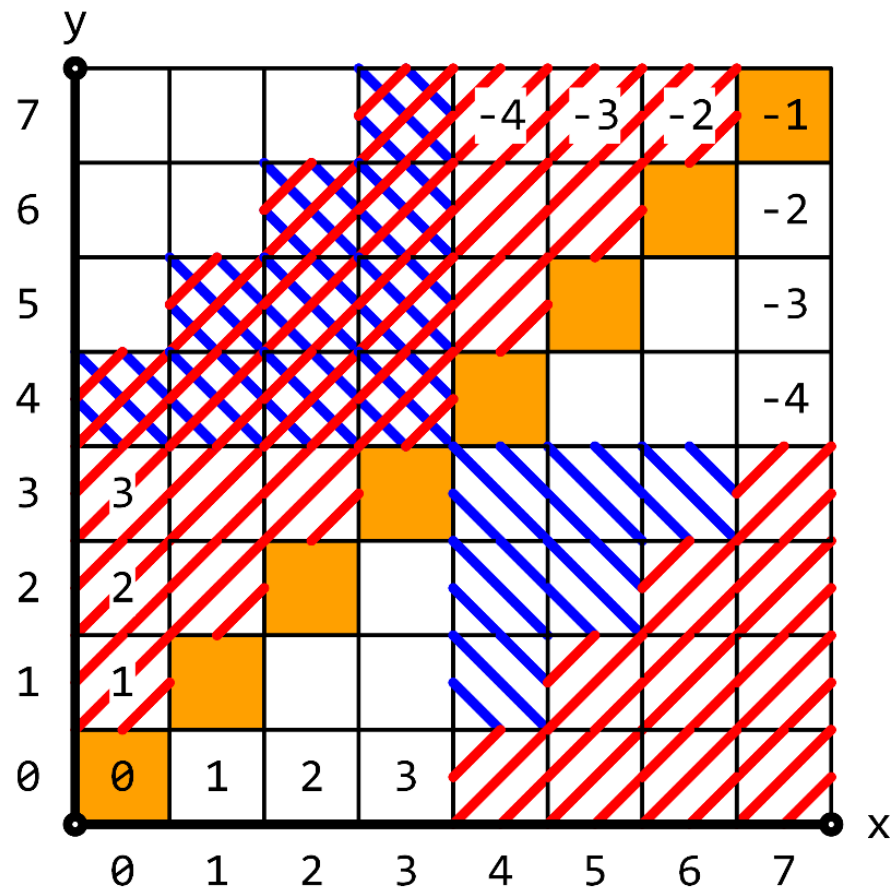
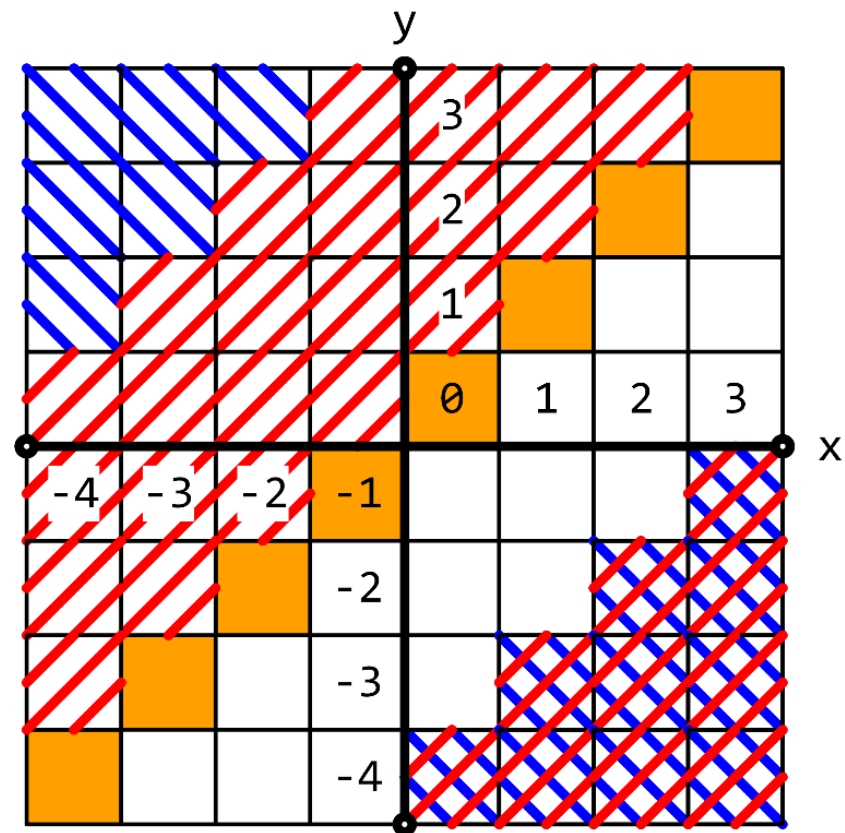
| Jcc | Условие | Описание |
|-----|--------------|---------------------------------------|
| JE | ZF | Равно / Ноль |
| JNE | ~ZF | Не равно / Не ноль |
| JS | SF | Отрицательное число |
| JNS | ~SF | Неотрицательное число |
| JG | ~(SF^OF)&~ZF | Больше (знаковые числа) |
| JGE | ~(SF^OF) | Больше либо 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

Сравнение x и y реализуется как формула над флагами, полученными после вычисления $(x-y)$



Сравнение: со знаком и без



| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
|-----|-----|-----|-----|-----|-----|-----|-----|

0 1 2 3 4 5 6 7 Беззнаковое число

0 1 2 3 -4 -3 -2 -1 Знаковое число

«Основные» команды IA-32

| Пересылка данных | Двоичная арифметика ✓ | Передача управления | Логические ✓ | Сдвиги и вращения | Битовые и байтовые | Прочее |
|---|--|---|---|--|--|--|
| <ul style="list-style-type: none"> • MOV • XCHG • BSWAP • MOVSX • MOVZX • CDQ • CWD • CBW • PUSH • POP • CMOVCc | <ul style="list-style-type: none"> • ADD • ADC • SUB • SBB • NEG • IMUL • MUL • IDIV • DIV • INC • DEC • CMP | <ul style="list-style-type: none"> • JMP • Jcc • CALL • RET | <ul style="list-style-type: none"> • AND • OR • XOR • NOT | <ul style="list-style-type: none"> • SAR • SHR • SAL, SHL • ROR • ROL • RCR • RCL | <ul style="list-style-type: none"> • SETcc • TEST | <ul style="list-style-type: none"> • LEA • NOP |

Красным выделены не упоминавшиеся ранее на лекциях ассемблерные инструкции. Команды двоичной арифметики и логические команды представлены в полном составе.

```
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
```

```

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;
}

```

```

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;
```



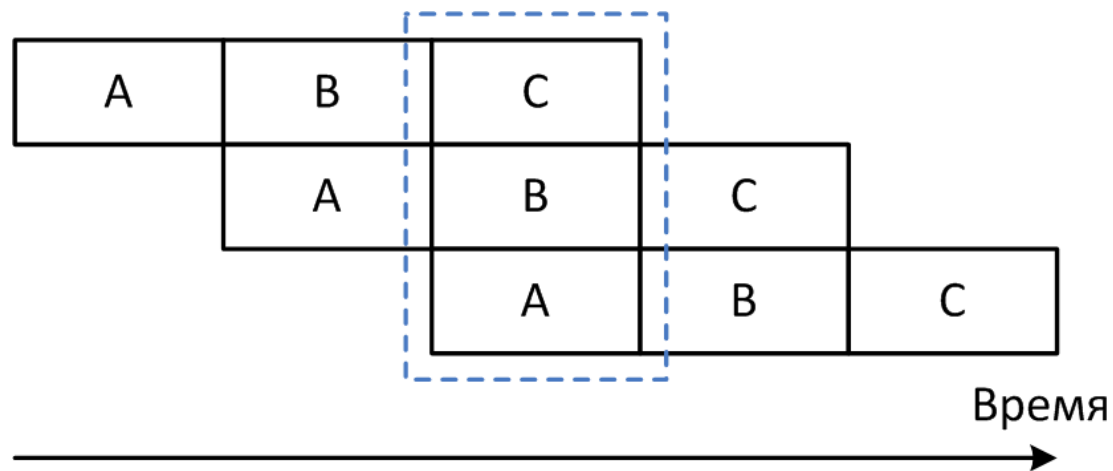
```
nt = !(Test);  
if (nt) goto Else;  
val = Then_Expr;  
goto Done;  
Else:  
    val = Else_Expr;  
Done:  
    ...
```



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

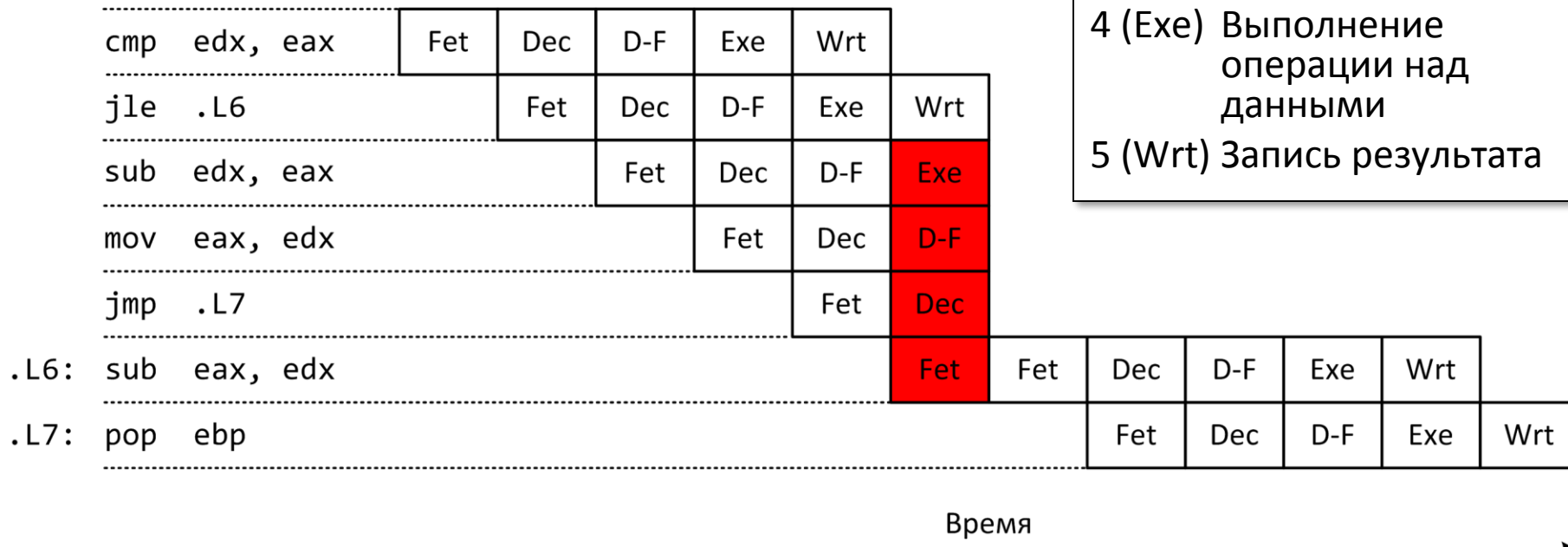

Конвейер – совмещение разных действий в один момент времени

- Общая для различных предметных областей методика
- Длительность обработки неизменна или несколько увеличивается
- Увеличение пропускной способности



Опустошение конвейера при передаче управления

- 1 (Fet) Извлечение инструкции из памяти
- 2 (Dec) Декодирование команды
- 3 (D-F) Загрузка операндов
- 4 (Exe) Выполнение операции над данными
- 5 (Wrt) Запись результата



```

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

```

```

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;
}
```



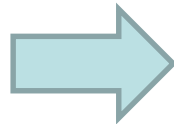
```
int pcount_do(unsigned x) {
    int result = 0;
    if (!x) goto done;
loop:
    result += x & 0x1;
    x >>= 1;
    if (x)
        goto loop;
done:
    return result;
}
```



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;
}
```

```

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

```



```

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

```

```

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

```

```

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

```