Кодогенерация С++ кроссплатформенно

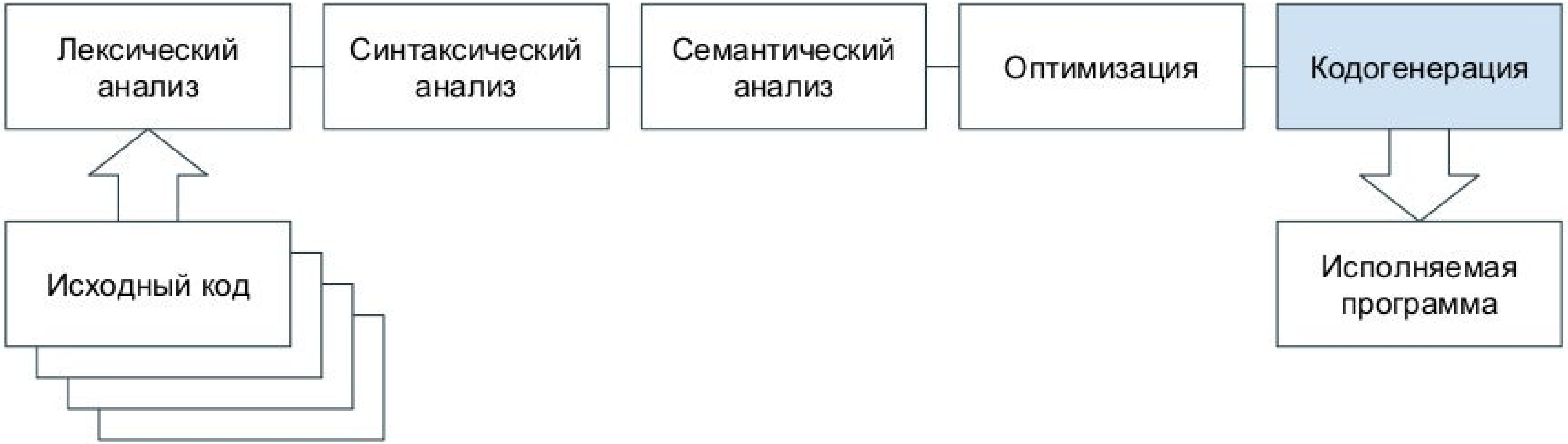
часть 1

Алексей Ткаченко, ОАО “Пеленг”
tkachenko@peleng.by
alexey.tkachenko@gmail.com

О чём доклад?

- сопоставление языковых конструкций C++ с генерируемым машинным кодом
- сравнение генерации для различных платформ и архитектур
- особенности архитектур в контексте кроссплатформенности
- развенчание мифов

Фазы компиляции



Платформы

Рассмотрим

- Intel x86 (CISC, 32bit) / MSVS 2015 (Windows 10)
- x86-64 (CISC, 64bit) / MSVS 2015 (Windows 10)
- ARM Cortex-A7 (RISC, 32bit) / GCC 4.6.3 (Debian 7 @ CubieBoard2)
- Atmel AVR (AVR RISC, 8bit) / GCC 4.8.1, Arduino (Windows 10)
- IBM PowerPC (RISC, 32bit) / Xilinx EDK 14.7 (Ubuntu 17)
- Xilinx Microblaze (RISC, 32bit, over FPGA) / Xilinx EDK 14.7 (Ubuntu 17)
- Atmel AVR (AVR RISC, 8bit) / GCC 4.8.1, Arduino (Windows 10)

Не рассматриваем

- IA-64/Itanium (VLIW, 64bit)
- ARM 64 (RICS, 32bit)
- Эльбрус-4С (VLIW, 64bit)
- AVR32
- Microchip PIC
- Сигнальные процессоры
- и многие другие

Загрузка константы

C++
```
return 42;
```

x86
```
mov     eax,2Ah
```

ARM
```
movs    r0, #42
```

PowerPC
```
li         r3,42
```

x64
```
mov     rax,2Ah
```

AVR
```
ldi    r24, 0x2A
ldi    r25, 0x00
```

MicroBlaze
```
addik    r3, r0, 42
```

Загрузка константы

C++
```
return 4200;
```

x86
```
mov     eax,1068h
```

x64
```
mov     rax,1068h
```

ARM
```
movs    r0, #4200
```

AVR
```
ldi    r24, 0x68
ldi    r25, 0x10
```

PowerPC
```
li          r3,4200
```

MicroBlaze
```
addik    r3, r0, 4200
```

Загрузка константы

C++
```
return 0xDEADBEEFL;
```

x86
```
mov    eax,0DEADBEEFh
```

x64
```
mov    rax,0DEADBEEFh
```

ARM
```
movw    r0, #0xBEEF
movt    r0, #0xDEAD
```

AVR
```
ldi    r22, 0xEF
ldi    r23, 0xBE
ldi    r24, 0xAD
ldi    r25, 0xDE
```

PowerPC
```
lis       r0,0xDEAD
ori       r0,r0,0xBEEF
```

MicroBlaze
```
imm     0xDEAD
addik   r5, r0, 0xBEEF
```

Глобальная переменная

C++
```
extern ptrdiff_t a;

return a;
```

x86
```
mov     eax,dword ptr [a]
```

ARM
```
movw    r3, #5804
movt    r3, #1
ldr     r0, [r3, #0]
```

PowerPC
```
lis     r9,5
lwz     r3,28980(r9)
```

x64
```
mov     rax,qword ptr [a]
```

AVR
```
lds     r24, 0x01C3
lds     r25, 0x01C4
```

MicroBlaze
```
imm     -28663
lwi     r3, r0, 10008
```

Глобальная переменная

C++
```
const ptrdiff_t ca = 43;

return ca;
```

x86
```
mov     eax,2Bh
```

ARM
```
movs    r0, #43
```

PowerPC
```
li          r3,43
```

x64
```
mov     rax,2Bh
```

AVR
```
ldi    r24, 0x2B
ldi    r25, 0x00
```

MicroBlaze
```
addik    r3, r0, 43
```

Глобальная переменная

C++
```
extern
volatile ptrdiff_t va;

return va;
```

x86
```
mov    eax,dword ptr [va]
```

ARM
```
movw    r3, #5808
movt    r3, #1
ldr    r0, [r3, #0]
```

PowerPC
```
lis     r9,5
lwz     r3,28984(r9)
```

x64
```
mov    rax,qword ptr [va]
```

AVR
```
lds    r24, 0x01C1
lds    r25, 0x01C2
```

MicroBlaze
```
imm    -28663
lwi    r3, r0, 10004
```

Глобальная переменная

C++
```
const volatile
ptrdiff_t cva = 'M' ^ 'E';

return cva;
```

x86
```
mov     eax,dword ptr [cva]
```

x64
```
mov     rax,qword ptr [cva]
```

ARM
```
movw    r3, #5828
movt    r3, #1
ldr     r0, [r3, #0]
```

AVR
```
lds     r24, 0x0112
lds     r25, 0x0113
```

PowerPC
```
lis     r9,5
lwz     r3,-13140(r9)
```

MicroBlaze
```
imm     -28664
lwi     r3, r0, 13196
```

Обращение по ссылке

C++
```
ptrdiff_t& ref = a;

return ref + 1;
```

x86
```
mov    eax,dword ptr [ref]
mov    eax,dword ptr [eax]
```

x64
```
mov    rax,qword ptr [ref]
mov    rax,qword ptr [rax]
```

ARM
```
movw   r3, #5804
movt   r3, #1
ldr    r0, [r3, #0]
```

AVR
```
lds    r24, 0x01C3
lds    r25, 0x01C4
```

PowerPC
```
lis     r9,5
lwz     r11,28544(r9)
lwz     r3,0(r11)
```

MicroBlaze
```
imm    -28663
lwi    r3, r0, 10008
```

Обращение по указателю

C++
```
ptrdiff_t *ptr = &a;

return *ptr;
```

x86
```
mov    eax,dword ptr [ptr]
mov    eax,dword ptr [eax]
```

x64
```
mov    rax,qword ptr [ptr]
mov    rax,qword ptr [rax]
```

ARM
```
movw   r3, #5628
movt   r3, #1
ldr    r3, [r3, #4]
ldr    r0, [r3, #0]
```

AVR
```
lds    r30, 0x0118
lds    r31, 0x0119
ld     r24, Z
ldd    r25, Z+1
```

PowerPC
```
lis    r9,5
lwz    r11,28568(r9)
lwz    r3,0(r11)
```

MicroBlaze
```
imm    -28664
lwi    r3, r0, 13208
lwi    r3, r3, 0
```

Чтение программной памяти (AVR)

C++

```
const PROGMEM uint16_t progdata[] = { 1, 2, 3 };
volatile int progindex;
uint16_t ReadProgData(int index)
{
   return pgm_read_word(progdata + index);
}


return ReadProgData(progindex);
```

AVR

```
lds    r30, 0x013B ; progindex
lds    r31, 0x013C ;
add    r30, r30
adc    r31, r31
subi   r30, 0x93 ; progdata
sbci   r31, 0xFF ;
lpm    r24, Z+
lpm    r25, Z
```

Обращение к элементу массива

C++
```
extern ptrdiff_t arr[];

return arr[0];
```

x86
```
mov     eax,dword ptr [arr]
```

x64
```
mov     rax,qword ptr [arr]
```

ARM
```
movw    r3, #5812
movt    r3, #1
ldr     r0, [r3, #0]
```

AVR
```
lds     r24, 0x01B9
lds     r25, 0x01BA
```

PowerPC
```
lis       r9,5
lwz       r3,29340(r9)
```

MicroBlaze
```
imm     -28663
lwi     r3, r0, 9988
```

Обращение к элементу массива

C++
```
extern ptrdiff_t arr[];

return arr[1];
```

x86
```
mov     eax,dword ptr
[004060FCh]
```

ARM
```
movw    r3, #5812
movt    r3, #1
ldr     r0, [r3, #4]
```

PowerPC
```
lis      r9,5
lwz      r3,29344(r9)
```

x64
```
mov     rax,qword ptr
[140006188h]
```

AVR
```
lds    r24, 0x01BB
lds    r25, 0x01BC
```

MicroBlaze
```
imm    -28663
lwi    r3, r0, 9992
```

Обращение к элементу массива

C++

```
extern ptrdiff_t arr[];
extern size_t arr_idx;

return arr[arr_idx];
```

x86

```
mov  eax,dword ptr [arr_idx]
mov  eax,dword ptr arr[eax*4]
```

x64

```
mov     rax,qword ptr [arr_idx]
lea     rcx,[arr]
mov     rax,qword ptr [rcx+rax*8]
```

Обращение к элементу массива

C++

```
extern ptrdiff_t arr[];
extern size_t arr_idx;

return arr[arr_idx];
```

ARM

```
movw    r2, #5828
movt    r2, #1
movw    r3, #5812
movt    r3, #1
ldr     r2, [r2, #0]
ldr.w   r0, [r3, r2, lsl #2]
```

AVR

```
lds     r30, 0x01B7
lds     r31, 0x01B8
add     r30, r30
adc     r31, r31
subi    r30, 0x47
sbci    r31, 0xFE
ld      r24, Z
ldd     r25, Z+1
```

Обращение к элементу массива

C++
```
extern ptrdiff_t arr[];
extern size_t arr_idx;

return arr[arr_idx];
```

PowerPC
```
lis      r9,5
lwz      r0,28988(r9)
lis      r9,5
addi     r9,r9,29340
rlwinm  r0,r0,2,0,29
lwzx     r3,r9,r0
```

MicroBlaze
```
imm    -28663
lwi    r3, r0, 9984
bslli   r3, r3, 2
imm    -28663
addik   r3, r3, 9988
lwi    r3, r3, 0
```

Поля структуры

C++
```
struct Test
{
  uint16_t a;
  uint8_t b;
  uint32_t c;
  uint32_t d;
};

extern Test t;

return t.a + t.b + t.c + t.d;
```

x86
```
movzx ecx,byte ptr [40610Ah]
movzx eax,word ptr [t]
add   ecx,dword ptr [406110h]
add   eax,ecx
add   eax,dword ptr [40610Ch]
```

x64 MSVS
```
movzx   ecx,byte ptr [1400061A2h]
mov     eax,dword ptr [1400061A8h]
movzx   edx,word ptr [t]
add     eax,ecx
add     eax,edx
add     eax,dword ptr [1400061A4h]
```

Поля структуры

C++
```
struct Test
{
  uint16_t a;
  uint8_t b;
  uint32_t c;
  uint32_t d;
};

extern Test t;

return t.a + t.b + t.c + t.d;
```

ARM
```
movw    r3, #5832
movt    r3, #1

ldr     r0, [r3, #8]
ldr     r1, [r3, #4]
ldrh    r2, [r3, #0]
ldrb    r3, [r3, #2]

adds    r0, r0, r1
adds    r3, r2, r3
adds    r0, r0, r3
```

AVR
```
lds    r18, 0x01AE
lds    r24, 0x01AF
lds    r25, 0x01B0
add    r24, r18
adc    r25, r1
lds    r18, 0x01AC
lds    r19, 0x01AD
add    r24, r18
adc    r25, r19
lds    r18, 0x01B3
lds    r19, 0x01B4
add    r24, r18
adc    r25, r19
```

Поля структуры

C++

```
struct Test
{
  uint16_t a;
  uint8_t b;
  uint32_t c;
  uint32_t d;
};

extern Test t;

return t.a + t.b + t.c + t.d;
```

PowerPC

```
lis     r11,5
addi    r9,r11,29356
lhz     r0,29356(r11)
lwz     r10,8(r9)
lbz     r11,2(r9)
lwz     r3,4(r9)
add     r0,r0,r11
add     r3,r3,r10
add     r3,r0,r3
```

MicroBlaze

```
imm    -28663
addik   r5, r0, 9980
imm    -28663
addik   r4, r0, 9972
lwi    r7, r5, 0
lwi    r6, r5, -4
lbui    r3, r4, 2
lhui    r5, r4, 0
addk    r4, r7, r6
addk    r3, r5, r3
rtsd    r15, 8
addk    r3, r4, r3
```

Поля структуры

C++
```
Test t = {
create_rand<uint16_t>(),
create_rand<uint8_t>(),
create_rand<uint32_t>(),
create_rand<uint32_t>()
};
return t.a + t.b + t.c + t.d;
```

x86
```
push    ebx
push    esi
push    edi

call    _create_rand
mov     di,ax
call    _create_rand
mov     bl,al
call    _create_rand
mov     esi,eax
call    _create_rand
```

```
movzx       ecx,bl
movzx       edx,di
add     eax,ecx
add     eax,edx
pop     edi
add     eax,esi
pop     esi
pop     ebx
```

Поля структуры

C++
```
Test t = {
create_rand<uint16_t>(),
create_rand<uint8_t>(),
create_rand<uint32_t>(),
create_rand<uint32_t>()
};
return t.a + t.b + t.c + t.d;
```

x64 MSVS
```
mov     qword ptr [rsp+8],rbx
mov     qword ptr [rsp+10h],rsi
push    rdi
sub     rsp,20h

call    qword ptr [__imp_rand]
mov     esi,eax
call    qword ptr [__imp_rand]
mov     ebx,eax
call    qword ptr [__imp_rand]
mov     edi,eax
call    qword ptr [__imp_rand]
```

```
movzx       ecx,bl
movzx       edx,di
add         eax,ecx
add         eax,edx
pop         edi
add         eax,esi
pop         esi
pop         ebx
```

Поля структуры

C++

```
Test t = {
create_rand<uint16_t>(),
create_rand<uint8_t>(),
create_rand<uint32_t>(),
create_rand<uint32_t>()
};
return t.a + t.b + t.c + t.d;
```

ARM

```
push   {r4, r5, r6, lr}
bl   8cb0
mov   r4, r0
bl   8cb0
mov   r6, r0
bl   8cb0
uxtb   r6, r6
uxtah   r4, r6, r4
mov   r5, r0
bl   8cb0
adds   r5, r0, r5
adds   r0, r5, r4
pop   {r4, r5, r6, pc}
```

AVR

```
push   r15
push   r16
push   r17
push   r28
push   r29
call   0x394
movw   r28, r24
call   0x394
mov   r15, r24
call   0x394
movw   r16, r24
call   0x394
movw   r18, r16
add   r18, r15
adc   r19, r1
add   r18, r28
adc   r19, r29
add   r24, r18
adc   r25, r19
pop   r29
pop   r28
pop   r17
pop   r16
pop   r15
```

Поля структуры

C++

```
Test t = {
create_rand<uint16_t>(),
create_rand<uint8_t>(),
create_rand<uint32_t>(),
create_rand<uint32_t>()
};
return t.a + t.b + t.c + t.d;
```

PowerPC

```
stwu    r1,-32(r1)
mflr    r0
stw     r27,12(r1)
stw     r0,36(r1)
stw     r28,16(r1)
stw     r29,20(r1)
bl      20334
mr      r29,r3
bl      20310
mr      r27,r3
bl      2030c
mr      r28,r3
```

```
bl      2030c
lwz     r0,36(r1)
add     r29,r29,r27
add     r28,r28,r3
lwz     r27,12(r1)
add     r3,r29,r28
lwz     r28,16(r1)
lwz     r29,20(r1)
mtlr    r0
```

MicroBlaze

```
addik   r1, r1, -40
swi     r15, r1, 0
swi     r19, r1, 28
swi     r22, r1, 32
brlid   r15, -1188
swi     r23, r1, 36
brlid   r15, -1196
addk    r23, r3, r0
brlid   r15, -1204
addk    r22, r3, r0
brlid   r15, -1212
```

```
addk    r19, r3, r0
imm     0
andi    r5, r23, -1
andi    r4, r22, 255
addk    r3, r3, r19
lwi     r15, r1, 0
lwi     r19, r1, 28
lwi     r22, r1, 32
lwi     r23, r1, 36
addk    r4, r5, r4
addk    r3, r3, r4
addik   r1, r1, 40
```

Поля структуры

C++
```
struct SmallTest
{
   uint16_t a;
   uint8_t b;
};

extern SmallTest st;

return st.a + st.b;
```

x86
```
movzx eax,byte ptr[406116h]
movzx ecx,word ptr [st]
add      eax,ecx
```

x64
```
movzx rax,byte ptr
[1400061AEh]
movzx rcx,word ptr [st]
add      rax,rcx
```

ARM
```
movw    r3, #5844
movt    r3, #1
ldrh    r0, [r3, #0]
ldrb    r3, [r3, #2]
adds    r0, r0, r3
```

AVR
```
lds   r18, 0x01AB
lds   r24, 0x01A9
lds   r25, 0x01AA
add   r24, r18
adc   r25, r1
```

PowerPC
```
lis      r9,5
addi     r11,r9,28992
lhz      r0,28992(r9)
lbz      r3,2(r11)
add      r3,r0,r3
```

MicroBlaze
```
imm    -28663
addik  r4, r0, 9968
lhui   r5, r4, 0
lbui   r3, r4, 2
rtsd   r15, 8
addk   r3, r5, r3
```

Поля структуры

C++
```
SmallTest st = {
create_rand<uint16_t>(),
create_rand<uint8_t>()
};
return st.a + st.b;
```

x86
```
push    esi
call    _create_rand
mov     si,ax
call    _create_rand
movzx   ecx,si
movzx   eax,al
add     eax,ecx
pop     esi
```

x64 MSVS
```
push    rbx
sub     rsp,20h
call    __imp_rand
mov     ebx,eax
call    __imp_rand
movzx   rax,al
movzx   rcx,bx
add     rax,rcx
add     rsp,20h
pop     rbx
```

ARM
```
push    {r4, lr}
bl      8cb0
mov     r4, r0
bl      8cb0
uxtb    r0, r0
uxtah   r0, r0, r4
pop     {r4, pc}
```

Поля структуры

C++
```
SmallTest st = {
create_rand<uint16_t>(),
create_rand<uint8_t>()
};
return st.a + st.b;
```

AVR
```
push    r28
push    r29
call    0x394
movw    r28, r24
call    0x394
movw    r18, r28
add     r18, r24
adc     r19, r1
movw    r24, r18
pop     r29
pop     r28
```

PowerPC
```
stwu    r1,-24(r1)
mflr    r0
stw     r29,12(r1)
stw     r0,28(r1)
bl      20334
mr      r29,r3
bl      20310
lwz     r0,28(r1)
add     r3,r29,r3
lwz     r29,12(r1)
addi    r1,r1,24
mtlr    r0
```

MicroBlaze
```
addik   r1, r1, -32
swi     r15, r1, 0
brlid   r15, -1124
swi     r19, r1, 28
brlid   r15, -1132
addk    r19, r3, r0
andi    r4, r3, 255
lwi     r15, r1, 0
imm     0
andi    r3, r19, -1
lwi     r19, r1, 28
addk    r3, r3, r4
addik   r1, r1, 32
```

Массив структур

C++

```
struct SmallTest
{
   uint16_t a;
   uint8_t b;
};

extern SmallTest arr_st[];
extern size_t arr_st_idx;

SmallTest st =
arr_st[arr_st_idx];
return st.a + st.b;
```

x86

```
mov     ecx,dword ptr [arr_st_idx]
mov     ecx,dword ptr arr_st[ecx*4]
mov     eax,ecx
shr     eax,10h
movzx   eax,al
movzx   ecx,cx
add     eax,ecx
```

x64 MSVS

```
mov     rax,qword ptr [arr_st_idx]
lea     rcx,[arr_st]
mov     ecx,dword ptr [rcx+rax*4]
mov     eax,ecx
movzx   edx,cx
shr     eax,10h
movzx   eax,al
add     rax,rdx
```

Массив структур

C++

```
struct SmallTest
{
   uint16_t a;
   uint8_t b;
};

extern SmallTest arr_st[];
extern size_t arr_st_idx;

SmallTest st =
arr_st[arr_st_idx];
return st.a + st.b;
```

ARM

```
movw    r2, #5624
movt    r2, #1
movw    r3, #5960
movt    r3, #1

ldr     r2, [r2, #0]
add.w   r1, r3, r2, lsl #2
ldrh.w  r0, [r3, r2, lsl #2]
ldrb    r3, [r1, #2]
adds    r0, r0, r3
```

AVR

```
lds     r24, 0x010E
lds     r25, 0x010F
movw    r30, r24
add     r30, r30
adc     r31, r31
add     r30, r24
adc     r31, r25
subi    r30, 0x6F
sbci    r31, 0xFE
ldd     r18, Z+2
ld      r24, Z
ldd     r25, Z+1
add     r24, r18
adc     r25, r1
```

Массив структур

C++
```
struct SmallTest
{
   uint16_t a;
   uint8_t b;
};

extern SmallTest arr_st[];
extern size_t arr_st_idx;

SmallTest st =
arr_st[arr_st_idx];
return st.a + st.b;
```

PowerPC
```
lis     r11,5
lis     r9,5
lwz     r0,28520(r11)
addi    r9,r9,29368
rlwinm  r0,r0,2,0,29
add     r11,r0,r9
lhzx    r10,r9,r0
lbz     r3,2(r11)
add     r3,r10,r3
```

MicroBlaze
```
imm    -28664
lwi    r4, r0, 13188
bslli  r4, r4, 2
imm    -28663
addik  r4, r4, 9936
lhui   r5, r4, 0
lbui   r3, r4, 2
rtsd   r15, 8
addk   r3, r5, r3
```

Битовые поля

C++

```
struct BitTest
{
  uint16_t a : 1;
  uint16_t b : 2;
  uint16_t c : 3;
  uint16_t d : 4;
  uint16_t e : 5;
};
extern BitTest bt;

return bt.a + bt.b + bt.c +
bt.d + bt.e;
```

x86

```
movzx  esi,word ptr [bt]
mov    edx,esi
mov    eax,esi
shr    eax,6
mov    ecx,esi
and    eax,0Fh
shr    ecx,3
shr    edx,0Ah
and    ecx,7
and    edx,1Fh
add    eax,edx
add    eax,ecx
```

```
mov    ecx,esi
shr    ecx,1
and    esi,1
and    ecx,3
add    eax,ecx
add    eax,esi
```

Битовые поля

C++

```
struct BitTestS
{
 int16_t a : 1;
 int16_t b : 2;
 int16_t c : 3;
 int16_t d : 4;
 int16_t e : 5;
};
extern BitTestS bts;

return bts.a + bts.b + bts.c
+ bts.d + bts.e;
```

x86

```
mov     cx,word ptr [bts]
mov     ax,cx
mov     dx,cx
shl     ax,0Fh
push    esi
movsx   esi,ax
mov     ax,cx
shl     ax,0Dh
cwde
sar     eax,0Eh
sar     esi,0Fh
add     esi,eax
shl     dx,6
```

```
mov     ax,cx
movsx           edx,dx
shl     ax,0Ah
add     cx,cx
cwde
sar     eax,0Dh
add     eax,esi
sar     edx,0Ch
movsx           ecx,cx
add     eax,edx
sar     ecx,0Bh
add     eax,ecx
pop     esi
```

Битовые поля

C++

```
struct BitTest
{
  uint16_t a : 1;
  uint16_t b : 2;
  uint16_t c : 3;
  uint16_t d : 4;
  uint16_t e : 5;
};
extern BitTest bt;

return bt.a + bt.b + bt.c +
bt.d + bt.e;
```

x64

```
movzx   ecx,word ptr [bt]
mov     edx,ecx
mov     eax,ecx
shr     eax,6
and     eax,0Fh
shr     edx,0Ah
and     edx,1Fh
add     edx,eax
mov     eax,ecx
shr     eax,3
and     eax,7
add     edx,eax
```

```
mov     eax,ecx
shr     eax,1
and     ecx,1
and     eax,3
add     eax,edx
add     eax,ecx
```

Битовые поля

C++

```
struct BitTest
{
  uint16_t a : 1;
  uint16_t b : 2;
  uint16_t c : 3;
  uint16_t d : 4;
  uint16_t e : 5;
};
extern BitTest bt;

return bt.a + bt.b + bt.c +
bt.d + bt.e;
```

ARM

```
movw    r3, #5848
movt    r3, #1
ldrh    r3, [r3, #0]
ubfx    r2, r3, #1, #2
and.w   r0, r3, #1
adds    r0, r0, r2
ubfx    r2, r3, #3, #3
adds    r0, r0, r2
ubfx    r2, r3, #6, #4
adds    r0, r0, r2
ubfx    r3, r3, #10, #5
adds    r0, r0, r3
```

PowerPC

```
lis       r9,5
lhz       r0,28996(r9)
rlwinm  r9,r0,19,30,31
rlwinm  r11,r0,17,15,31
rlwinm  r10,r0,31,27,31
add       r11,r11,r9
rlwinm  r3,r0,22,29,31
rlwinm  r0,r0,26,28,31
add       r3,r3,r0
add       r11,r11,r10
add       r3,r3,r11
```

MicroBlaze

```
imm    -28663
lhui   r4, r0, 9836
bsrli  r3, r4, 13
bsrli  r7, r4, 15
bsrli  r6, r4, 10
bsrli  r5, r4, 6
andi   r3, r3, 3
addk   r3, r7, r3
andi   r6, r6, 7
addk   r3, r3, r6
andi   r5, r5, 15
srl    r4, r4
addk   r3, r3, r5
andi   r4, r4, 31
```

Битовые поля

C++
```
struct BitTestS
{
  int16_t a : 1;
  int16_t b : 2;
  int16_t c : 3;
  int16_t d : 4;
  int16_t e : 5;
};
extern BitTestS bts;

return bts.a + bts.b + bts.c
+ bts.d + bts.e;
```

ARM
```
movw    r3, #5988
movt    r3, #1
ldrh    r3, [r3, #0]
sbfx    r2, r3, #1, #2
sbfx    r0, r3, #0, #1
adds    r0, r0, r2
sbfx    r2, r3, #3, #3
adds    r0, r0, r2
sbfx    r2, r3, #6, #4
adds    r0, r0, r2
sbfx    r3, r3, #10, #5
adds    r0, r0, r3
```

Битовые поля

C++

```
struct BitTest
{
 uint16_t a : 1;
 uint16_t b : 2;
 uint16_t c : 3;
 uint16_t d : 4;
 uint16_t e : 5;
};
extern BitTest bt;

return bt.a + bt.b + bt.c +
bt.d + bt.e;
```

AVR

```
lds    r18, 0x0137
mov    r24, r18
andi   r24, 0x01
ldi    r25, 0x00
mov    r19, r18
lsr    r19
andi   r19, 0x03
add    r24, r19
adc    r25, r1
mov    r19, r18
```

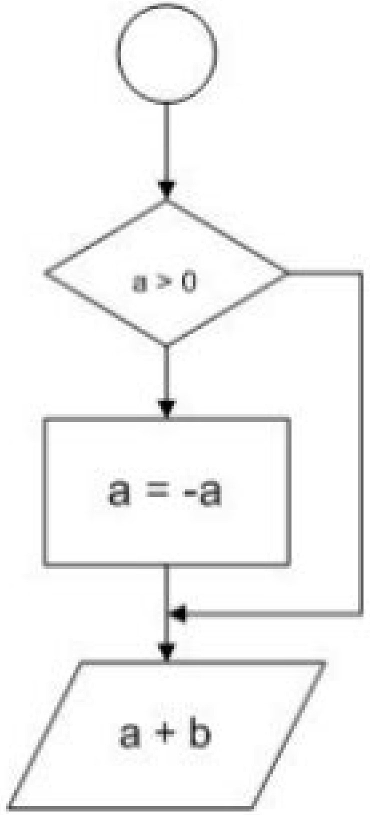
```
lsr    r19
lsr    r19
lsr    r19
andi   r19, 0x07
add    r24, r19
adc    r25, r1
mov    r20, r18
swap   r20
lsr    r20
lsr    r20
andi   r20, 0x03
lds    r19, 0x0138
mov    r18, r19
```

```
andi   r18, 0x03
add    r18, r18
add    r18, r18
or     r18, r20
add    r24, r18
adc    r25, r1
mov    r18, r19
lsr    r18
lsr    r18
andi   r18, 0x1F
add    r24, r18
adc    r25, r1
```

Условное выполнение (if)

C++
```
extern ptrdiff_t a;
extern ptrdiff_t b;

if (a > 0)
{
  a = -a;
}
return a + b;
```



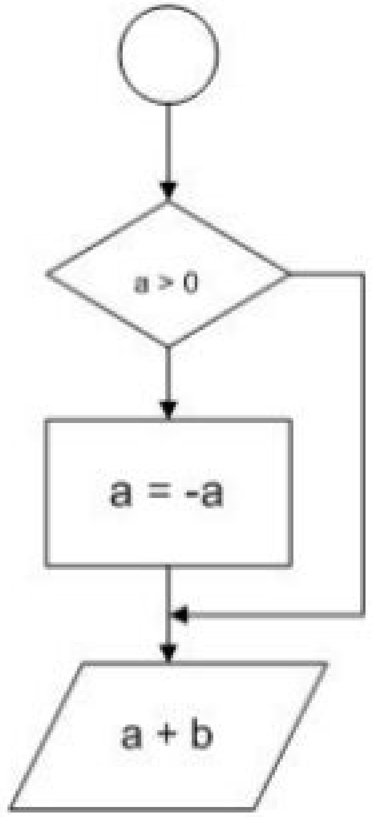
x86
```
mov     ecx,dword ptr [a]
mov     eax,dword ptr [b]
test    ecx,ecx
jle     @1
neg     ecx
mov     dword ptr [a],ecx
@1:
add     eax,ecx
```

x64
```
mov     rcx,qword ptr [a]
mov     rax,qword ptr [b]
test    rcx,rcx
jle     @1
neg     rcx
mov     qword ptr [a],rcx
@1:
add     rax,rcx
```

Условное выполнение (if)

C++
```
extern ptrdiff_t a;
extern ptrdiff_t b;

if (a > 0)
{
  a = -a;
}
return a + b;
```



ARM
```
ldr    r2, [pc, #16]
ldr    r3, [r2, #0]
cmp    r3, #0
ble.n  @1
negs   r3, r3
str    r3, [r2, #0]
@1:
ldr    r2, [pc, #8]
ldr    r0, [r2, #0]
adds   r0, r3, r0
```

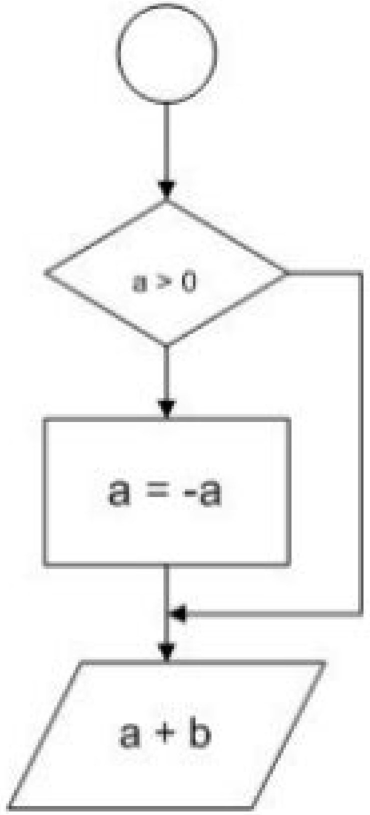
AVR
```
lds    r24, 0x010A
lds    r25, 0x010B
cp     r1, r24
cpc    r1, r25
brge   .+14 ; @1
neg    r25
neg    r24
sbc    r25, r1
sts    0x010B, r25
sts    0x010A, r24
```

```
@1:
lds    r18, 0x010A
lds    r19, 0x010B
lds    r24, 0x0108
lds    r25, 0x0109
add    r24, r18
adc    r25, r19
```

Условное выполнение (if)

C++
```
extern ptrdiff_t a;
extern ptrdiff_t b;

if (a > 0)
{
  a = -a;
}
return a + b;
```



PowerPC
```
lis      r11,5
lis      r9,5
lwz      r3,28528(r11)
lwz      r0,28532(r9)
cmpwi    cr7,r3,0
ble-     cr7, @1
neg      r3,r3
stw      r3,28528(r11)
@1:
add      r3,r3,r0
```

MicroBlaze
```
imm   -28664
lwi   r3, r0, 13180
blei  r3, 16 // @1
rsubk  r3, r3, r0
imm   -28664
swi   r3, r0, 13180
@1:
imm   -28664
lwi   r4, r0, 13176
```

Условное выполнение (if)

C++
```
extern ptrdiff_t a;
extern ptrdiff_t b;

if (a > 0)
{
  return b;
}
return 1;
```

x86
```
cmp     dword ptr [a],0
mov     eax,1
cmovg   eax,dword ptr [b]
```

x64
```
cmp     qword ptr [a],0
mov     eax,1
cmovg   rax,qword ptr [b]
```

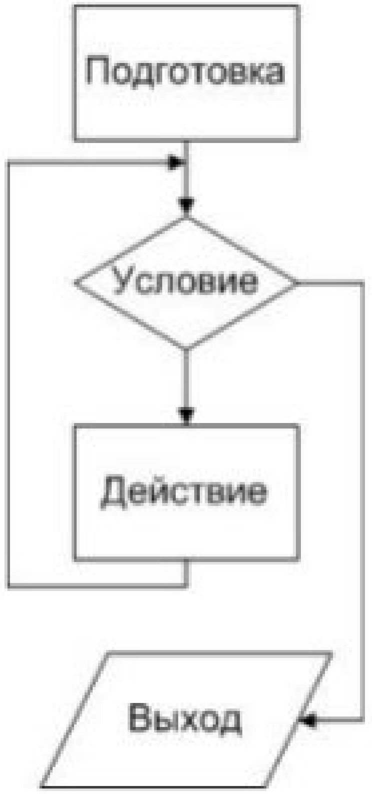
ARM
```
ldr    r3, [pc, #12]
ldr    r3, [r3, #0]
cmp    r3, #0
itte   gt
ldrgt  r3, [pc, #8]
ldrgt  r0, [r3, #0]
movle  r0, #1
```

Цикл while

C++

```
extern size_t limit;

size_t a = 1, b = 1;
size_t result = a + b;
while (result < limit)
{
  size_t sum = a + b;
  a = b;
  b = sum;
  result += sum;
}
return result;
```



x86

```
mov     edi,dword ptr [limit]
mov     eax,2
mov     esi,1
mov     edx,esi
cmp     edi,eax
jbe     @1

@2:
lea     ecx,[edx+esi]
add     eax,ecx
lea     esi,[edx]
mov     edx,ecx
cmp     eax,edi
jb      @2
@1:
```

x64

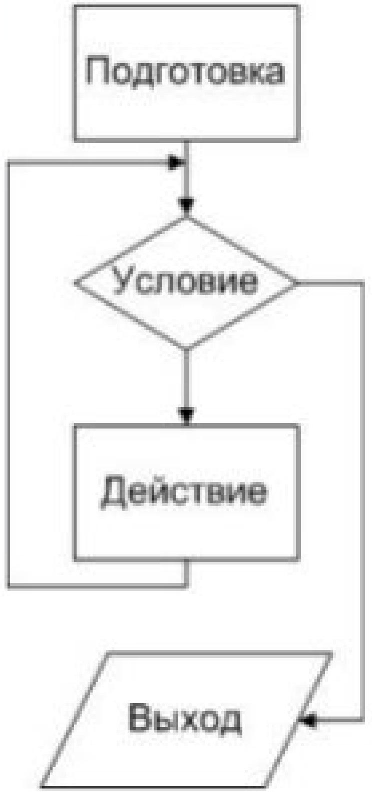
```
mov     r9,qword ptr [limit]
mov     rax,2
mov     r8d,1
mov     edx,r8d
cmp     rax,r9
jae     @1
nop
@2:
lea     rcx,[rdx+r8]
add     rax,rcx
lea     r8,[rdx]
mov     rdx,rcx
cmp     rax,r9
jb      @2
@1
```

Цикл while

C++

```
extern size_t limit;

size_t a = 1, b = 1;
size_t result = a + b;
while (result < limit)
{
  size_t sum = a + b;
  a = b;
  b = sum;
  result += sum;
}
return result;
```



ARM

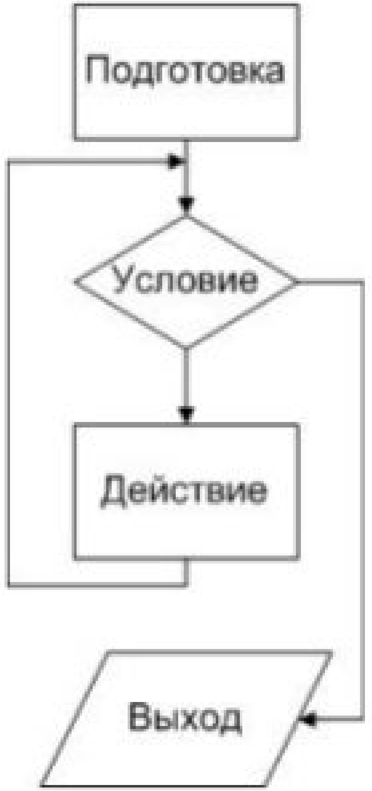
```
ldr    r3, [pc, #32]
movs   r0, #2
push   {r4}
ldr    r4, [r3, #0]
cmp    r4, #2
bls.n  @1
movs   r3, #1
movs   r0, #2
mov    r1, r3
```

```
@2:
adds   r2, r3, r1
mov    r1, r3
adds   r0, r0, r2
cmp    r0, r4
mov    r3, r2
bcc.n  @2
@1:
pop    {r4}
```

Цикл while

C++
```
extern size_t limit;

size_t a = 1, b = 1;
size_t result = a + b;
while (result < limit)
{
 size_t sum = a + b;
 a = b;
 b = sum;
 result += sum;
}
return result;
```



AVR
```
lds   r30, 0x0106
lds   r31, 0x0107
ldi   r24, 0x02
ldi   r25, 0x00
ldi   r18, 0x01
ldi   r19, 0x00
ldi   r20, 0x01
ldi   r21, 0x00
```

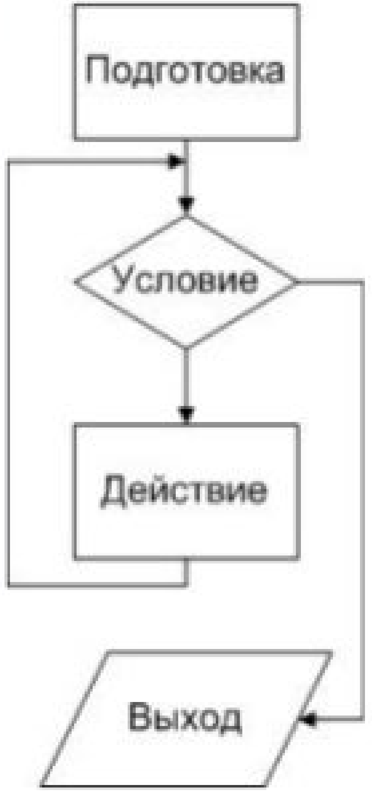
```
@2:
cp    r24, r30
cpc   r25, r31
brcc  @1
movw  r22, r20
add   r22, r18
adc   r23, r19
add   r24, r22
adc   r25, r23
movw  r20, r18
movw  r18, r22
rjmp  @2
@1:
```

Цикл while

C++

```
extern size_t limit;

size_t a = 1, b = 1;
size_t result = a + b;
while (result < limit)
{
  size_t sum = a + b;
  a = b;
  b = sum;
  result += sum;
}
return result;
```



PowerPC

```
lis      r9,5
li       r3,2
lwz      r10,28536(r9)
cmplwi  cr7,r10,2
blelr   cr7
li       r11,1
li       r9,1
@1:
add      r0,r11,r9
mr       r11,r9
add      r3,r3,r0
mr       r9,r0
cmplw   cr7,r3,r10
blt+     cr7, @1
```

MicroBlaze

```
imm    -28664
lwi    r7, r0, 13172
addik   r5, r0, 2
cmpu    r18, r7, r5
bgeid   r18, @1
addk   r3, r5, r0
addik   r4, r0, 1
addk   r6, r4, r0
```

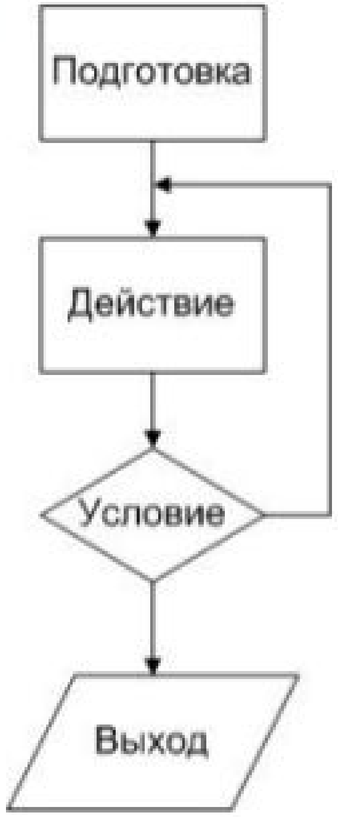
```
@2:
addk    r5, r4, r6
addk    r3, r3, r5
addk    r6, r4, r0
cmpu    r18, r7, r3
bltid   r18, @2
addk    r4, r5, r0
@1:
```

Цикл do/while

C++

```
extern size_t limit;

size_t a = 1, b = 1;
size_t result = a + b;
do
{
  size_t sum = a + b;
  a = b;
  b = sum;
  result += sum;
} while (result < limit);
return result;
```



x86

```
push    esi
mov     esi,1
push    edi
mov     edi,dword ptr [limit]
mov     edx,esi
lea     eax,[esi+1]
@1:
lea     ecx,[edx+esi]
add     eax,ecx
lea     esi,[edx]
mov     edx,ecx
cmp     eax,edi
jb      @1
pop     edi
pop     esi
```

x64 MSVS

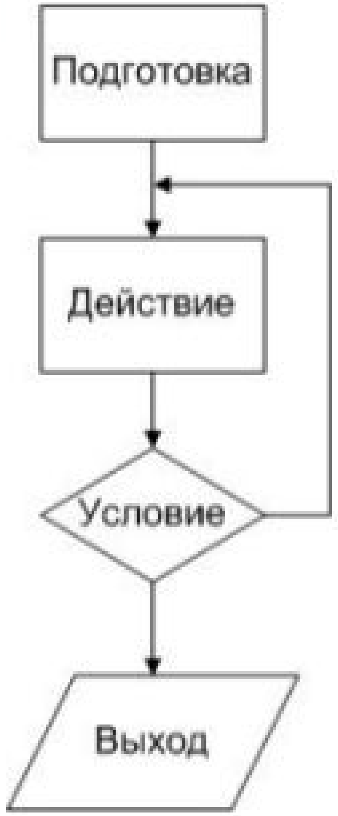
```
mov     r9,qword ptr [limit]
mov     r8d,1
mov     rdx,r8d
lea     rax,[r8+1]
nop     dword ptr [rax]
nop     dword ptr [rax+rax]
@1:
lea     rcx,[rdx+r8]
add     rax,rcx
lea     r8,[rdx]
mov     rdx,rcx
cmp     rax,r9
jb      @1
```

Цикл do/while

C++

```
extern size_t limit;

size_t a = 1, b = 1;
size_t result = a + b;
do
{
  size_t sum = a + b;
  a = b;
  b = sum;
  result += sum;
} while (result < limit);
return result;
```



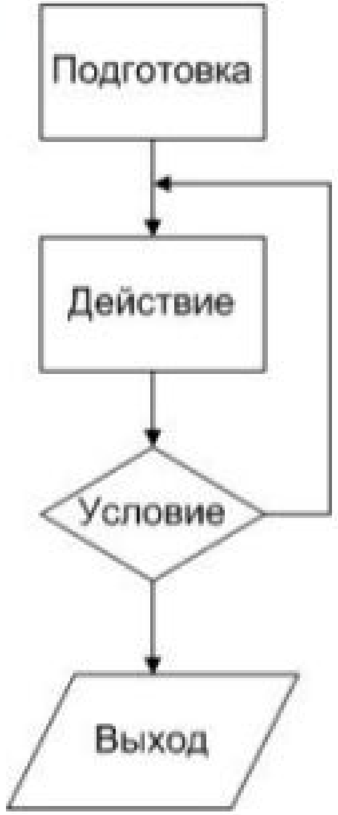
ARM

```
ldr    r2, [pc, #24]
movs   r3, #1
push   {r4}
movs   r0, #2
ldr    r4, [r2, #0]
mov    r1, r3
@1:
adds   r2, r3, r1
mov    r1, r3
adds   r0, r0, r2
cmp    r0, r4
mov    r3, r2
bcc.n  @1
pop    {r4}
```

Цикл do/while

C++
```
extern size_t limit;

size_t a = 1, b = 1;
size_t result = a + b;
do
{
 size_t sum = a + b;
 a = b;
 b = sum;
 result += sum;
} while (result < limit);
return result;
```



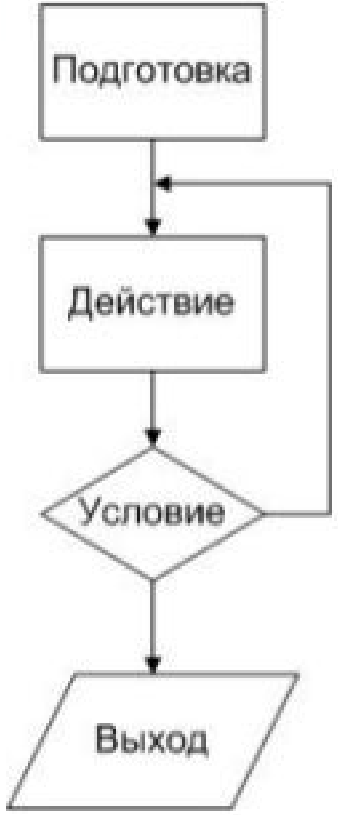
AVR
```
lds   r30, 0x0106
lds   r31, 0x0107
ldi   r24, 0x02
ldi   r25, 0x00
ldi   r18, 0x01
ldi   r19, 0x00
ldi   r22, 0x01
ldi   r23, 0x00
```

```
@2:
movw   r20, r22
add    r20, r18
adc    r21, r19
add    r24, r20
adc    r25, r21
cp     r24, r30
cpc    r25, r31
brcc   @1
movw   r22, r18
movw   r18, r20
rjmp   @2
@1:
```

Цикл do/while

C++
```
extern size_t limit;

size_t a = 1, b = 1;
size_t result = a + b;
do
{
  size_t sum = a + b;
  a = b;
  b = sum;
  result += sum;
} while (result < limit);
return result;
```



PowerPC
```
lis      r9,5
li       r3,2
lwz      r10,28536(r9)
li       r11,1
li       r9,1
@1:
add      r0,r9,r11
mr       r11,r9
add      r3,r3,r0
mr       r9,r0
cmplw    cr7,r3,r10
blt+     cr7,@1
```

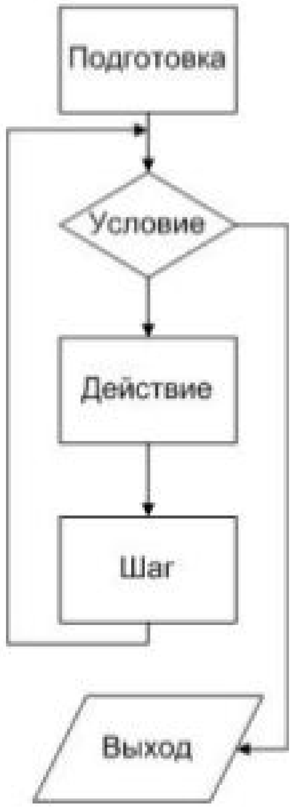
MicroBlaze
```
iaddik   r4, r0, 1
imm      -28664
lwi      r7, r0, 13172
addk     r6, r4, r0
addik    r3, r0, 2
@1:
addk     r5, r4, r6
addk     r3, r3, r5
addk     r6, r4, r0
cmpu     r18, r7, r3
bltid    r18, @1
addk     r4, r5, r0
```

Цикл for

C++
```
extern size_t for_steps;
size_t sum = 0;

for (
 size_t i = 0;
  i < for_steps;
  ++i)
{
 sum += i * i % (i + 1);
}

return sum;
```



x86
```
sub     esp,8
mov     eax,dword ptr
[for_steps]
xor     edx,edx
push    ebx
push    esi
xor     esi,esi
xor     ebx,ebx
xor     ecx,ecx
mov     dword ptr [ebp-4],esi
cmp     eax,2
jb      @1
dec     eax
push    edi
mov     dword ptr [ebp-8],eax
mov     edi,1
```

```
@2:
<step>
<step>
cmp     ecx,dword ptr [ebp-8]
jb      @2
mov     eax,dword ptr [for_steps]
xor     edx,edx
pop     edi
@1:
cmp     ecx,eax
jae     @3
<step>
@3:
lea     eax,[esi+ebx]
pop     esi
add     eax,edx
pop     ebx
```

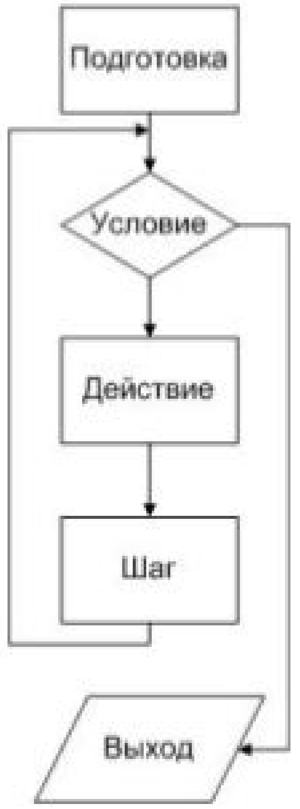
Цикл for

C++

```
extern size_t for_steps;
size_t sum = 0;

for (
 size_t i = 0;
  i < for_steps;
  ++i)
{
 sum += i * i % (i + 1);
}

return sum;
```



x64 MSVS

```
push    rdi
xor     r10d,r10d
mov     qword ptr [rsp+10h],rbx
mov     rbx,qword ptr [for_steps]
mov     r11d,r10d
mov     rdi,r10d
mov     rcx,r10d
cmp     rbx,2
jb      @1
mov     qword ptr [rsp+18h],rsi
mov     r9d,1
lea     rsi,[rbx-1]
nop
@2:
xor     rdx,rdx
lea     r8,[r9+1]
<step>
<step>
cmp     rcx,rsi
jb      @2
mov     rsi,qword ptr [rsp+18h]
@1:
cmp     rcx,rbx
mov     rbx,qword ptr [rsp+10h]
jae     @3
mov     rax,rcx
<step>
@3:
lea     rax,[r11+r10]
add     rax,rdi
pop     rdi
```

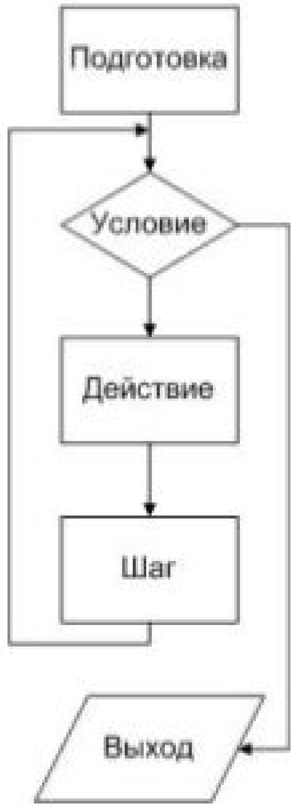
Цикл for

C++

```
extern size_t for_steps;
size_t sum = 0;

for (
 size_t i = 0;
  i < for_steps;
  ++i)
{
 sum += i * i % (i + 1);
}

return sum;
```



ARM

```
ldr    r3, [pc, #36]
push   {r4, r5, r6, lr}
ldr    r6, [r3, #0]
cbz    r6, @1
movs   r4, #0
mov    r5, r4
@1:
mul.w  r0, r4, r4
adds   r4, #1
mov    r1, r4
blx    8720 <_init+0x88>
```

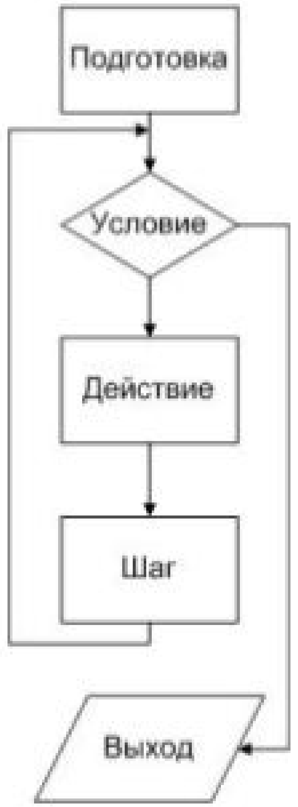
```
cmp    r4, r6
add    r5, r1
bne.n  @2
mov    r0, r5
pop    {r4, r5, r6, pc}
@1:
mov    r0, r6
pop    {r4, r5, r6, pc}
```

Цикл for

C++
```
extern size_t for_steps;
size_t sum = 0;

for (
  size_t i = 0;
   i < for_steps;
   ++i)
{
  sum += i * i % (i + 1);
}

return sum;
```



AVR
```
push   r28
push   r29
lds    r28, 0x0104
lds    r29, 0x0105
ldi    r18, 0x00
ldi    r19, 0x00
ldi    r30, 0x00
ldi    r31, 0x00
@2:
cp     r18, r28
cpc    r19, r29
breq   @1
mul    r18, r18
movw   r24, r0
mul    r18, r19
add    r25, r0
add    r25, r0
eor    r1, r1
subi   r18, 0xFF
sbci   r19, 0xFF
movw   r22, r18
call   __udivmodhi4
add    r30, r24
adc    r31, r25
rjmp   @2
@1:
movw   r24, r30
pop    r29
pop    r28
```

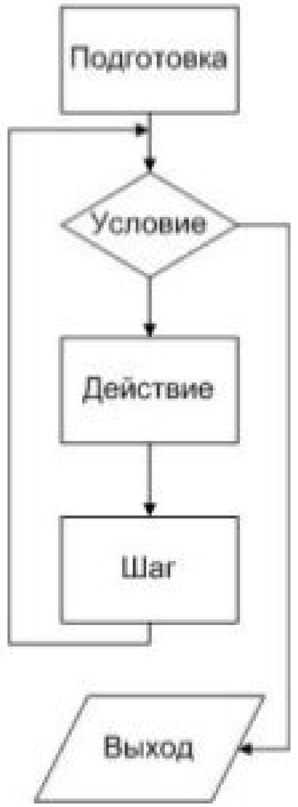
Цикл for

C++

```
extern size_t for_steps;
size_t sum = 0;

for (
  size_t i = 0;
   i < for_steps;
   ++i)
{
  sum += i * i % (i + 1);
}

return sum;
```



PowerPC

```
lis       r9,5
li        r3,0
lwz       r0,28540(r9)
cmpwi  cr7,r0,0
beqlr  cr7
mtctr  r0
li        r11,0
@1:
mullw  r9,r11,r11
addi     r11,r11,1
divwu  r0,r9,r11
mullw  r0,r0,r11
subf     r9,r0,r9
add      r3,r3,r9
bdnz+  @1
```

MicroBlaze

```
addik   r1, r1, -40
swi    r23, r1, 36
imm    -28664
lwi    r23, r0, 13168
swi    r15, r1, 0
swi    r19, r1, 28
swi    r22, r1, 32
beqid   r23, @1
addk   r3, r23, r0
addk   r19, r0, r0
addk   r22, r19, r0
@2:
mul    r5, r19, r19
addik   r19, r19, 1
brlid   r15,
__umodsi3
addk   r6, r19, r0
cmpu   r18, r23, r19
bltid   r18, @2
addk   r22, r22, r3
addk   r3, r22, r0
@1:
lwi   r15, r1, 0
lwi   r19, r1, 28
lwi   r22, r1, 32
lwi   r23, r1, 36
```

Вызов функции

C++

```
ptrdiff_t x_call();

return 1 + x_call();
```

x86

```
call    x_call
inc     eax
```

ARM

```
push    {r3, lr}
bl      x_call
adds    r0, #1
pop     {r3, pc}
```

x64 MSVS

```
call    x_call
inc     rax
```

AVR

```
call  x_call
subi  r24, 0xFF
sbci  r25, 0xFF
```

Вызов функции

C++
```
ptrdiff_t x_call();

return 1 + x_call();
```

PowerPC
```
mflr    r0
stw     r0,12(r1)
bl      x_call
lwz     r0,12(r1)
addi    r3,r3,1
addi    r1,r1,8
mtlr    r0
blr
```

MicroBlaze
```
addik   r1, r1, -28
swi     r15, r1, 0
brlid   r15, x_call
or      r0, r0, r0
lwi     r15, r1, 0
addik   r3, r3, 1
rtsd    r15, 8
addik   r1, r1, 28
```

Вызов функции с параметрами

C++
```
ptrdiff_t x_call_params
(ptrdiff_t a,
ptrdiff_t b,
ptrdiff_t c);

return 2 + x_call_params
(1, 2, 3);
```

x86 / caller
```
push    3
push    2
push    1
call    x_call_params
add     esp,0Ch
add     eax,2
```

x86 / callee
```
push            ebp
mov             ebp,esp
; body
mov             esp,ebp
pop             ebp
ret
```

x64
```
mov     rdx,2
lea     r8d,[rdx+1]
lea     rcx,[rdx-1]
call    x_call_params
add     rax,2
```

Вызов функции

C++
```
ptrdiff_t x_call();

return 1 + x_call();
```

ARM
```
push    {r3, lr}
movs    r0, #1
movs    r1, #2
movs    r2, #3
bl      x_call_params
adds    r0, #2
pop     {r3, pc}
```

PowerPC
```
stwu    r1,-8(r1)
mflr    r0
li      r3,1
li      r4,2
li      r5,3
stw     r0,12(r1)
bl      x_call_params
lwz     r0,12(r1)
addi    r3,r3,2
addi    r1,r1,8
mtlr    r0
blr
```

MicroBlaze
```
addik   r1, r1, -28
addik   r5, r0, 1
addik   r6, r0, 2
swi     r15, r1, 0
brlid   r15, x_call_params
addik   r7, r0, 3
lwi     r15, r1, 0
addik   r3, r3, 2
rtsd    r15, 8
addik   r1, r1, 28
```

Конвенции вызова x86

C++

```
ptrdiff_t __cdecl x_call_cdecl(ptrdiff_t a, ptrdiff_t b, ptrdiff_t c);
ptrdiff_t __stdcall x_call_stdcall(ptrdiff_t a, ptrdiff_t b, ptrdiff_t c);
ptrdiff_t __fastcall x_call_fastcall(ptrdiff_t a, ptrdiff_t b, ptrdiff_t c);


return 2 + x_call_cdecl(1, 2, 3);
return 3 + x_call_stdcall(1, 2, 3);
return 4 + x_call_fastcall(1, 2, 3);
```

cdecl / caller

```
push    3
push    2
push    1
call    x_call_cdecl
add     esp,0Ch
add     eax,2
ret
```

cdecl / callee

```
push    ebp
mov     ebp,esp
sub     esp,0Ch ; locals
; body
mov     esp,ebp
pop     ebp
ret
```

Конвенции вызова x86

C++

```
ptrdiff_t __cdecl x_call_cdecl(ptrdiff_t a, ptrdiff_t b, ptrdiff_t c);
ptrdiff_t __stdcall x_call_stdcall(ptrdiff_t a, ptrdiff_t b, ptrdiff_t c);
ptrdiff_t __fastcall x_call_fastcall(ptrdiff_t a, ptrdiff_t b, ptrdiff_t c);


return 2 + x_call_cdecl(1, 2, 3);
return 3 + x_call_stdcall(1, 2, 3);
return 4 + x_call_fastcall(1, 2, 3);
```

stdcall / caller

```
push    3
push    2
push    1
call    x_call_stdcall
add     eax,3
ret
```

stdcall / callee

```
push    ebp
mov     ebp,esp
sub     esp,0Ch ; locals
; body
mov     esp,ebp
pop     ebp
ret     0Ch
```

Конвенции вызова x86

C++

```
ptrdiff_t __cdecl x_call_cdecl(ptrdiff_t a, ptrdiff_t b, ptrdiff_t c);
ptrdiff_t __stdcall x_call_stdcall(ptrdiff_t a, ptrdiff_t b, ptrdiff_t c);
ptrdiff_t __fastcall x_call_fastcall(ptrdiff_t a, ptrdiff_t b, ptrdiff_t c);


return 2 + x_call_cdecl(1, 2, 3);
return 3 + x_call_stdcall(1, 2, 3);
return 4 + x_call_fastcall(1, 2, 3);
```

fastcall / caller

```
mov     edx,2
push    3
lea     ecx,[edx-1]
call    x_call_fastcall
add     eax,4
ret
```

fastcall / callee

```
push    ebp
mov     ebp,esp
sub     esp,0Ch ; locals
; body
mov     esp,ebp
pop     ebp
ret     0Ch
```

Вызов вариативной функции

C++

```
ptrdiff_t
x_call_va(ptrdiff_t a,
ptrdiff_t b, ptrdiff_t c,
...);

return 5 + x_call_va(1, 2,
3, 4, 5);
```

x86

```
push        5
push        4
push        3
push        2
push        1
call    x_call_va
add     esp,14h
add     eax,5
ret
```

x64

```
sub     rsp,38h
mov     rdx,2
mov     dword ptr [rsp+20h],5
lea     r9d,[rdx+2]
lea     r8d,[rdx+1]
lea     rcx,[rdx-1]
call    x_call_va
add     rax,5
add     rsp,38h
ret
```

ARM

```
push    {r4, lr}
sub     sp, #8
movs    r0, #1
movs    r1, #2
movs    r2, #3
movs    r3, #4
movs    r4, #5
str     r4, [sp, #0]
bl      x_call_va
adds    r0, r0, r4
add     sp, #8
pop     {r4, pc}
```

Вызов вариативной функции

C++

```
ptrdiff_t
x_call_va(ptrdiff_t a,
ptrdiff_t b, ptrdiff_t c,
...);

return 5 + x_call_va(1, 2,
3, 4, 5);
```

PowerPC

```
stwu    r1,-8(r1)
mflr    r0
li      r3,1
li      r4,2
li      r5,3
li      r6,4
li      r7,5
stw     r0,12(r1)
bl      x_call_va
lwz     r0,12(r1)
addi    r3,r3,5
addi    r1,r1,8
mtlr    r0
blr
```

MicroBlaze

```
addik   r1, r1, -28
addik   r5, r0, 1
addik   r6, r0, 2
addik   r7, r0, 3
addik   r8, r0, 4
swi     r15, r1, 0
brlid   r15, x_call_va
addik   r9, r0, 5
lwi     r15, r1, 0
addik   r3, r3, 5
rtsd    r15, 8
addik   r1, r1, 28
```

Вызов функции по указателю

C++

```
typedef
ptrdiff_t(*fptr)(ptrdiff_t,
ptrdiff_t);

ptrdiff_t x_call_ptr(fptr,
ptrdiff_t, ptrdiff_t);

static ptrdiff_t sum(ptrdiff_t a,
ptrdiff_t b)
{
   return sizeof(fptr) + a + b;
}
return 6+x_call_ptr(sum,1,-2);
```

x86 / caller

```
push    0FFFFFFFEh
push    1
push    402B20h ; sum
call    x_call_ptr
add     esp,0Ch
add     eax,6
ret
```

x86 / callee

```
push    ebp
mov     ebp,esp
push    dword ptr [ebp+10h]
push    dword ptr [ebp+0Ch]
call    dword ptr [ebp+8]
add     eax,dword ptr [ebp+0Ch]
add     esp,8
add     eax,dword ptr [ebp+10h]
pop     ebp
ret
```

Вызов функции по указателю

C++

```
typedef
ptrdiff_t(*fptr)(ptrdiff_t,
ptrdiff_t);

ptrdiff_t x_call_ptr(fptr,
ptrdiff_t, ptrdiff_t);

static ptrdiff_t sum(ptrdiff_t a,
ptrdiff_t b)
{
   return sizeof(fptr) + a + b;
}
return 6+x_call_ptr(sum,1,-2);
```

x64 / caller

```
sub     rsp,28h
mov      rdx,1
lea     rcx,[140002A80h] ; sum
lea     r8,[rdx-3]
call    x_call_ptr
add     rax,6
add     rsp,28h
ret
```

x64 / callee

```
mov     qword ptr [rsp+8],rbx
push    rdi
sub      rsp,20h
mov     rbx,rdx
mov     rax,rcx
mov     rcx,rbx
mov     rdx,r8
mov     rdi,r8
call     rax
add      rax,rbx
mov     rbx,qword ptr [rsp+30h]
add      rax,rdi
add      rsp,20h
pop      rdi
ret
```

Вызов функции по указателю

C++

```
typedef
ptrdiff_t(*fptr)(ptrdiff_t,
ptrdiff_t);

ptrdiff_t x_call_ptr(fptr,
ptrdiff_t, ptrdiff_t);

static ptrdiff_t sum(ptrdiff_t a,
ptrdiff_t b)
{
   return sizeof(fptr) + a + b;
}
return 6+x_call_ptr(sum,1,-2);
```

ARM / caller

```
push    {r3, lr}
movs    r1, #1
mvn.w   r2, #1
movw    r0, #38401    ; sum
movt    r0, #0
bl      x_call_ptr
adds    r0, #6
pop     {r3, pc}
```

ARM / callee

```
push    {r4, lr}
mov     r3, r0
adds    r4, r1, r2
mov     r0, r1
mov     r1, r2
blx     r3
adds    r0, r4, r0
pop     {r4, pc}
```

Вызов функции по указателю

C++

```
typedef
ptrdiff_t(*fptr)(ptrdiff_t,
ptrdiff_t);

ptrdiff_t x_call_ptr(fptr,
ptrdiff_t, ptrdiff_t);

static ptrdiff_t sum(ptrdiff_t a,
ptrdiff_t b)
{
   return sizeof(fptr) + a + b;
}
return 6+x_call_ptr(sum,1,-2);
```

PowerPC / caller

```
stwu    r1,-8(r1)
mflr    r0
lis     r3,0
li      r4,1
addi    r3,r3,5676
li      r5,-2
stw     r0,12(r1)
bl      x_call_ptr
lwz     r0,12(r1)
addi    r3,r3,6
addi    r1,r1,8
mtlr    r0
blr
```

PowerPC / callee

```
stwu    r1,-24(r1)
mflr    r0
mtctr   r3
stw     r29,12(r1)
mr      r29,r4
stw     r28,8(r1)
mr      r3,r29
stw     r0,28(r1)
mr      r4,r5
mr      r28,r5
add     r29,r29,r28
bctrl
```

```
lwz     r0,28(r1)
add     r3,r29,r3
lwz     r28,8(r1)
lwz     r29,12(r1)
addi    r1,r1,24
mtlr    r0
blr
```

Вызов функции по указателю

C++

```
typedef
ptrdiff_t(*fptr)(ptrdiff_t,
ptrdiff_t);

ptrdiff_t x_call_ptr(fptr,
ptrdiff_t, ptrdiff_t);

static ptrdiff_t sum(ptrdiff_t a,
ptrdiff_t b)
{
   return sizeof(fptr) + a + b;
}
return 6+x_call_ptr(sum,1,-2);
```

MicroBlaze / caller

```
imm     -28672
addik   r5, r0, 3412
addik   r1, r1, -28
addik   r6, r0, 1
swi     r15, r1, 0
brlid   r15, x_call_ptr
addik   r7, r0, -2
lwi     r15, r1, 0
addik   r3, r3, 6
rtsd    r15, 8
addik   r1, r1, 28
```

MicroBlaze / callee

```
addk    r4, r6, r0
addk    r3, r5, r0
addk    r5, r6, r0
addk    r6, r7, r0
addik   r1, r1, -32
swi     r15, r1, 0
swi     r19, r1, 28
brald   r15, r3
addk    r19, r4, r7
addk    r3, r19, r3
lwi     r15, r1, 0
lwi     r19, r1, 28
rtsd    r15, 8
addik   r1, r1, 32
```

Вызов шаблонной функции

C++

```
template<typename T>
T Subtract(T a, T b)
{
   return a - b;
}

extern ptrdiff_t a;
extern ptrdiff_t b;
return 7 + Subtract(a, b);

extern char c;c
extern char d;
return 8 + Subtract(c, d);
```

x86

```
push    dword ptr [b]
push    dword ptr [a]
call  ??$Subtract@H@test_4@@YAHHH@Z
add       esp,8
add       eax,7
ret
```

x86

```
movzx             eax,byte ptr [d]
push      eax
movzx             eax,byte ptr [c]
push      eax
call   ??$Subtract@D@test_4@@YADDD@Z
movsx             eax,al
add       esp,8
add       eax,8
ret
```

Вызов шаблонной функции

C++
```
template<typename T>
T Subtract(T a, T b)
{
   return a - b;
}

extern ptrdiff_t a;
extern ptrdiff_t b;
return 7 + Subtract(a, b);

extern char c;c
extern char d;
return 8 + Subtract(c, d);
```

x64
```
sub     rsp,28h
mov     rdx,qword ptr [b]
mov     rcx,qword ptr [a]
call    ??$Subtract@_J@test_4@@YA_J_J0@Z
add     rax,7
add     rsp,28h
ret
```

x64
```
sub     rsp,28h
movzx           edx,byte ptr [d]
movzx           ecx,byte ptr [c]
call    ??$Subtract@D@test_4@@YADDD@Z
movsx           rax,al
add     rax,8
cdqe
add     rsp,28h
ret
```

Пока всё...

Спасибо за внимание!
Продолжение следует…

А теперь - вопросы!

Алексей Ткаченко, ОАО “Пеленг”
tkachenko@peleng.by
alexey.tkachenko@gmail.com