Кодогенерация C++ кроссплатформенно

часть 2

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)

Вызов метода

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = obj->Method1(1) + 1;
```

x86

```
push    1
mov     ecx,esi ; this
call    ?Method1@BaseClass@@QAEHH@Z
mov     edi,eax
```

x64

```
mov             edx,1
mov             rcx,rbx
call            ?Method1@BaseClass@@QEAA_J_J@Z
mov             rdi,rax
```

Вызов метода

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = obj->Method1(1) + 1;
```

ARM

```
mov     r0, r4
movs    r1, #1
bl      <BaseClass::Method1(int)>
adds    r5, r0, #1
```

AVR

```
ldi         r22, 0x01    ; 1
ldi         r23, 0x00    ; 0
movw        r24, r28
call        <BaseClass::Method1(int)>
movw        r16, r24
```

Вызов метода

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = obj->Method1(1) + 1;
```

PowerPC

```
mr          r3,r31
li          r4,1
bl          BaseClass::Method1(int)
addi        r3,r29,1
```

MicroBlaze

```
addk     r19, r3, r0
addk     r5, r19, r0
brlid    r15, BaseClass::Method1(int)
addik    r6, r0, 1
```

Вызов константного метода

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = obj->Method2(2) + 2;
```

x86

```
push    2
mov     ecx,esi
call    ?Method2@BaseClass@@QBEHH@Z
add     eax, 2
```

x64

```
mov     edx,2
mov     rcx,rbx
call    ?Method2@BaseClass@@QEBA_J_J@Z
add     rax, 2
```

Вызов константного метода

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = obj->Method2(2) + 2;
```

ARM

```
mov         r0, r4
movs        r1, #2
bl          <BaseClass::Method2(int) const>
adds        r5, r0, #2
```

AVR

```
movw        r24, r28
call        <BaseClass::Method2(int) const>
movw        r16, r24
subi        r16, 0xFE
sbci        r17, 0xFF
```

Вызов константного метода

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = obj->Method2(2) + 2;
```

PowerPC

```
mr          r3,r31
li          r4,2
bl          <BaseClass::Method2(int) const>
addi        r3,r29,2
```

MicroBlaze

```
addk    r19, r3, r0
addk    r5, r19, r0
brlid   r15, <BaseClass::Method2(int) const>
addik   r22, r3, 2
```

Вызов оператора

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

BaseClass op = ~*obj;
```

x86

```
mov     ecx,esi
call    ??SBaseClass@@QAE?AV0@XZ
```

x64

```
mov     rcx,rbx
call    ??SBaseClass@@QEAA?AV0@XZ
```

Вызов оператора

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

BaseClass op = ~*obj;
```

ARM

```
mov         r1, r4
bl          <BaseClass::operator~()>
```

AVR

```
movw        r22, r16
movw        r24, r28
adiw        r24, 0x01
call        <BaseClass::operator~()>
```

Вызов оператора

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = ~*obj;
```

PowerPC

```
mr              r3,r31
bl              <BaseClass::operator~()>
```

MicroBlaze

```
addk    r19, r3, r0
addk    r6, r19, r0
brlid   r15, <BaseClass::operator~()>
```

Вызов конструктора

C++

```
class BaseClass
{
protected:
  ptrdiff_t x;
  ptrdiff_t y;
  ptrdiff_t z;
public:
  BaseClass();
  virtual ~BaseClass();
};
BaseClass::BaseClass()
  : x(), y(), z()
{}
```

x86

```
mov     dword ptr [ecx],offset ??_7BaseClass@@6B@
mov     eax,ecx
mov     dword ptr [ecx+4],0
mov     dword ptr [ecx+8],0
mov     dword ptr [ecx+0Ch],0
ret
```

x64

```
lea     rax,[??_7BaseClass@@6B@]
mov     qword ptr [rcx],rax
xor     eax,eax
mov     qword ptr [rcx+8],rax
mov     qword ptr [rcx+10h],rax
mov     qword ptr [rcx+18h],rax
mov     rax,rcx
ret
```

Вызов конструктора

C++

```
class BaseClass
{
protected:
  ptrdiff_t x;
  ptrdiff_t y;
  ptrdiff_t z;
public:
  BaseClass();
  virtual ~BaseClass();
};
BaseClass::BaseClass()
  : x(), y(), z()
{}
```

ARM

```
ldr     r2, [pc, #12]     ; @VPTR
movs    r3, #0
str     r3, [r0, #4]
str     r3, [r0, #8]
str     r2, [r0, #0]
str     r3, [r0, #12]
bx      lr
nop

@VPTR: .word     0x00009f60
```

Вызов конструктора

C++

```
class BaseClass
{
protected:
  ptrdiff_t x;
  ptrdiff_t y;
  ptrdiff_t z;
public:
  BaseClass();
  virtual ~BaseClass();
};
BaseClass::BaseClass()
  : x(), y(), z()
{}
```

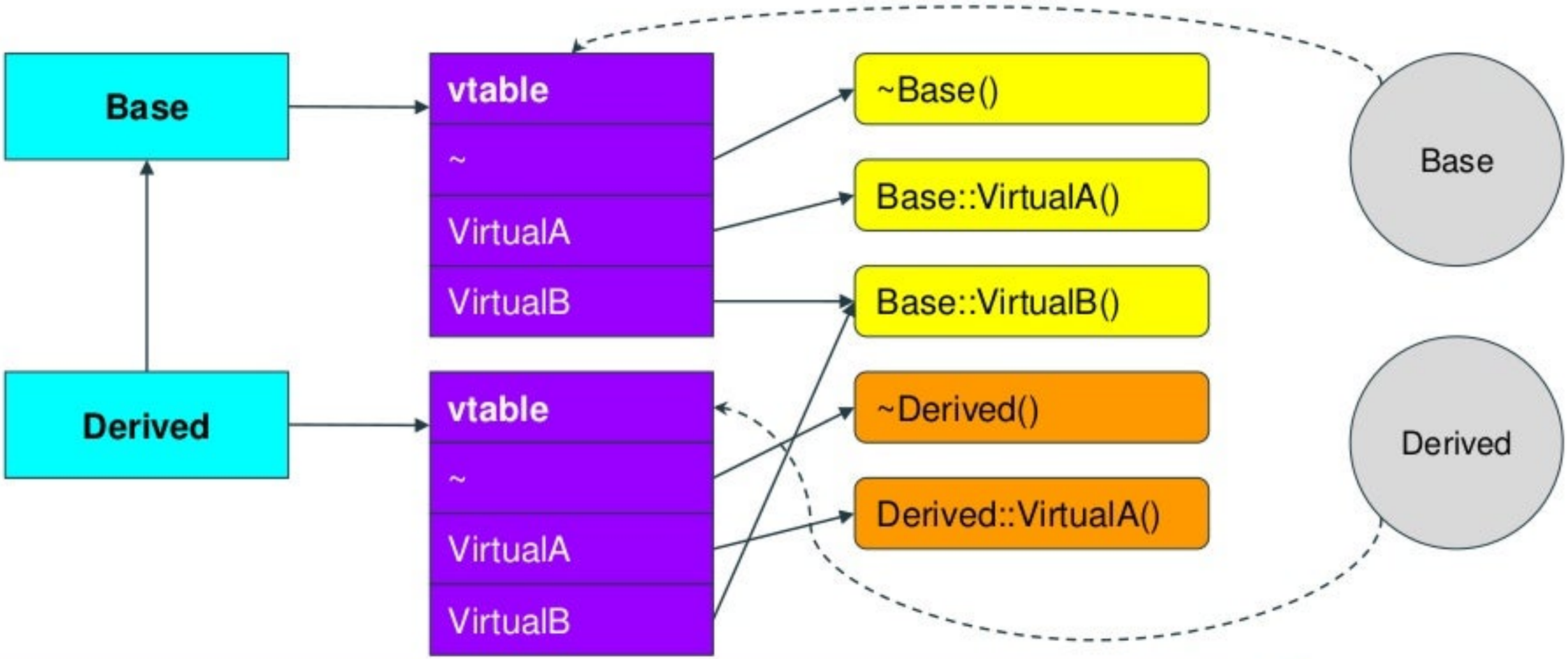
PowerPC

```
lis         r9,5
li          r0,0
addi        r9,r9,0xCDE8
stw         r0,12(r3)
stw         r9,0(r3)
stw         r0,4(r3)
stw         r0,8(r3)
blr
```

MicroBlaze

```
imm    0x9008
addik  r3, r0, 0x45E4
swi    r0, r5, 4
swi    r0, r5, 8
swi    r3, r5, 0
rtsd   r15, 8
swi    r0, r5, 12
```

Виртуальные методы: диспетчеризация



Вызов виртуального метода

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = obj->Method3(3) + 3;
```

x86

```
mov         edi,eax
mov         eax,dword ptr [edi]
mov         ecx,edi
push        3
call        dword ptr [eax+4]
```

x64

```
mov         rdi,rax
mov         rax,qword ptr [rdi]
mov         edx,3
mov         rcx,rdi
call        qword ptr [rax+8]
```

Вызов виртуального метода

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = obj->Method3(3) + 3;
```

ARM

```
ldr     r3, [r4, #0]
movs    r1, #3
mov     r0, r4
ldr     r3, [r3, #8]
blx     r3
```

AVR

```
ld      r30, Y
ldd     r31,
Y+1
ldd     r0,
Z+4
ldd     r31,
Z+5
mov     r30, r0
```

```
ldi     r22,
0x03
ldi     r23,
0x00
movw    r24,
r28
icall
```

Вызов виртуального метода

C++

```
class BaseClass
{
  ptrdiff_t Method1(ptrdiff_t a);
  ptrdiff_t Method2(ptrdiff_t a) const;
  BaseClass operator~();
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new BaseClass();

ptrdiff_t result = obj->Method3(3) * 3;
```

PowerPC

```
lwz     r9,0(r31)
li      r4,3
mr      r3,r31
lwz     r9,8(r9)
mtctr   r9
bctrl
```

MicroBlaze

```
addk          r19, r3, r0
lwi           r3, r19, 0
addk          r5, r19, r0
lwi           r3, r3, 8
brald         r15, r3
```

Вызов перекрытого виртуального метода

C++

```
class BaseClass
{
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}
class DerivedClass : public BaseClass
{
public:
  virtual ptrdiff_t Method3(ptrdiff_t
a);
}

BaseClass* obj = new DerivedClass();

ptrdiff_t result = obj->Method3(4) + 4;
```

x86

```
mov         edi,eax
mov         eax,dword ptr [edi]
mov         ecx,edi
push        4
call        dword ptr [eax+4]
```

x86 (virtual in base class)

```
mov         edi,eax
mov         eax,dword ptr [edi]
mov         ecx,edi
push        3
call        dword ptr [eax+4]
```

Указатель на поле

C++

```
class BaseClass
{
  ptrdiff_t x;
  ptrdiff_t y;
  ptrdiff_t z;
public:
  typedef ptrdiff_t BaseClass::*
FieldPtr;
 FieldPtr GetField(ptrdiff_t ptr);
};

BaseClass* obj = new BaseClass(1, 2,
3);
BaseClass::FieldPtr field = obj-
>GetField(1);
ptrdiff_t value = obj->*field;
```

x86

```
push    3
push    2
push    1
mov     ecx,eax
call    ??0BaseClass@@QAE@HHH@Z
mov     esi,eax
jmp     00403090
xor     esi,esi
push    1
mov     ecx,esi
call    ?GetField@BaseClass@@QAEPQ1@HH@Z
mov     edi,dword ptr [eax+esi]
```

Указатель на поле

C++

```
class BaseClass
{
  ptrdiff_t x;
  ptrdiff_t y;
  ptrdiff_t z;
public:
  typedef ptrdiff_t BaseClass::*
FieldPtr;
 FieldPtr GetField(ptrdiff_t ptr);
};

BaseClass* obj = new BaseClass(1, 2,
3);
BaseClass::FieldPtr field = obj-
>GetField(1);
ptrdiff_t value = obj->*field;
```

x64

```
mov                     edx,1
lea                     r9d,[rdx+2]
lea                     r8d,[rdx+1]
mov                     rcx,rax
call                    ??0BaseClass@@QEAA@_J00@Z
mov                     rbx,rax
mov                     edx,1
mov                     rcx,rbx
call
        ?GetField@BaseClass@@QEAAPEQ1@_J_J@Z
movsxd  rcx,eax
mov                     rdi,qword ptr [rcx+rbx]
```

Указатель на поле

C++

```
class BaseClass
{
  ptrdiff_t x;
  ptrdiff_t y;
  ptrdiff_t z;
public:
  typedef ptrdiff_t BaseClass::*
FieldPtr;
 FieldPtr GetField(ptrdiff_t ptr);
};

BaseClass* obj = new BaseClass(1, 2,
3);
BaseClass::FieldPtr field = obj-
>GetField(1);
ptrdiff_t value = obj->*field;
```

ARM

```
movs    r1, #1
movs    r2, #2
movs    r3, #3
mov     r4, r0
bl              <BaseClass::BaseClass(int, int,
int)>
mov     r0, r4
movs    r1, #1
bl              <BaseClass::GetField(int)>
ldr     r5, [r4, r0]
```

Указатель на поле

```
C++
class BaseClass
{
  ptrdiff_t x;
  ptrdiff_t y;
  ptrdiff_t z;
public:
  typedef ptrdiff_t BaseClass::*
FieldPtr;
 FieldPtr GetField(ptrdiff_t ptr);
};

BaseClass* obj = new BaseClass(1, 2,
3);
BaseClass::FieldPtr field = obj-
>GetField(1);
ptrdiff_t value = obj->*field;
```

```
PowerPC
li          r4,1
li          r5,2
li          r6,3
bl          <BaseClass::BaseClass(int, int,
int)>
mr          r3,r31
li          r4,1
bl          <BaseClass::GetField(int)>
lwzx        r29,r31,r3
```

Указатель на поле

C++

```
class BaseClass
{
  ptrdiff_t x;
  ptrdiff_t y;
  ptrdiff_t z;
public:
  typedef ptrdiff_t BaseClass::*
FieldPtr;
 FieldPtr GetField(ptrdiff_t ptr);
};

BaseClass* obj = new BaseClass(1, 2,
3);
BaseClass::FieldPtr field = obj-
>GetField(1);
ptrdiff_t value = obj->*field;
```

MicroBlaze

```
addk    r5, r3, r0
addik   r6, r0, 1
addik   r7, r0, 2
addik   r8, r0, 3
imm     ...
brlid   r15, BaseClass::BaseClass(int, int, int)>
addk    r19, r3, r0
addk    r5, r19, r0
imm     ...
brlid   r15, <BaseClass::GetField(int)>
lw      r22, r3, r19
```

Указатель на метод

C++

```
class BaseClass
{
public:
  typedef ptrdiff_t
(BaseClass::*MethodPtr) (ptrdiff_t);

  MethodPtr GetMethod(ptrdiff_t);
};

BaseClass* obj = new DerivedClass();
BaseClass::MethodPtr method = obj-
>GetMethod(0);
ptrdiff_t value = (obj->*method)(42) +
1;
```

x86

```
push    0
mov     ecx,esi
call    ?GetMethod@BaseClass@@QAEP81@AEHH@ZH@Z
push    2Ah
mov     ecx,esi
call    eax
```

Указатель на виртуальный метод

C++

```
class BaseClass
{
public:
  typedef ptrdiff_t
(BaseClass::*MethodPtr) (ptrdiff_t);

  MethodPtr GetMethod(ptrdiff_t);
};

BaseClass* obj = new DerivedClass();
BaseClass::MethodPtr method = obj-
>GetMethod(0);
ptrdiff_t value = (obj->*method)(42) +
1;
```

x86

```
push    1
mov     ecx,esi
call    ?GetMethod@BaseClass@@QAEP81@AEHH@ZH@Z
push    2Ah
mov     ecx,esi
call    eax


mov     eax,dword ptr [ecx]
jmp     dword ptr [eax+4]
```

Указатель на метод

C++

```
class BaseClass
{
public:
  typedef ptrdiff_t
(BaseClass::*MethodPtr) (ptrdiff_t);

  MethodPtr GetMethod(ptrdiff_t);
};

BaseClass* obj = new DerivedClass();
BaseClass::MethodPtr method = obj-
>GetMethod(0);
ptrdiff_t value = (obj->*method)(42) +
1;
```

x64

```
mov     edx,1
mov     rcx,rbx
call    ?GetMethod@BaseClass@@QEAAP81@EAA_J_J@Z0@Z
mov     edx,2Ah
mov     rcx,rbx
call    rax
```

Указатель на виртуальный метод

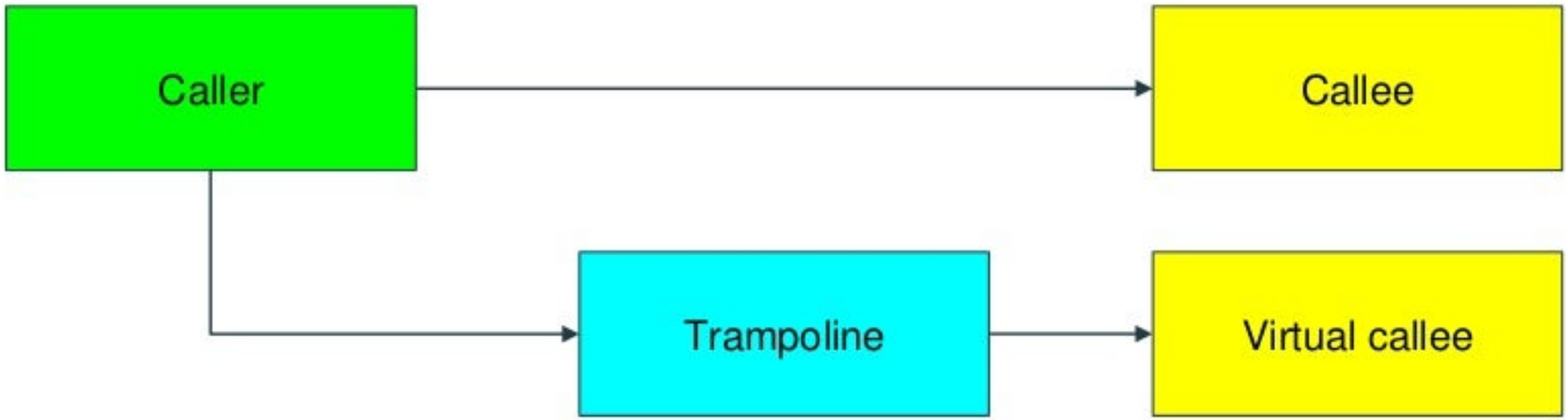
C++

```
class BaseClass
{
public:
  typedef ptrdiff_t
(BaseClass::*MethodPtr) (ptrdiff_t);

  MethodPtr GetMethod(ptrdiff_t);
};

BaseClass* obj = new DerivedClass();
BaseClass::MethodPtr method = obj-
>GetMethod(0);
ptrdiff_t value = (obj->*method)(42) +
1;
```

x64

```
xor     edx,edx
mov     rcx,rbx
call    ?GetMethod@BaseClass@@QEAAP81@EAA_J_J@Z0@Z
mov     edx,2Ah
mov     rcx,rbx
call    rax


mov     rax,qword ptr [rcx]
jmp     qword ptr [rax+8]
```

Указатель на метод



Указатель на метод

C++

```
class BaseClass
{
public:
  typedef ptrdiff_t
(BaseClass::*MethodPtr) (ptrdiff_t);

  MethodPtr GetMethod(ptrdiff_t ptr);
};

BaseClass* obj = new DerivedClass();
BaseClass::MethodPtr method = obj-
>GetMethod(0);
ptrdiff_t value = (obj->*method)(42) +
1;
```

ARM

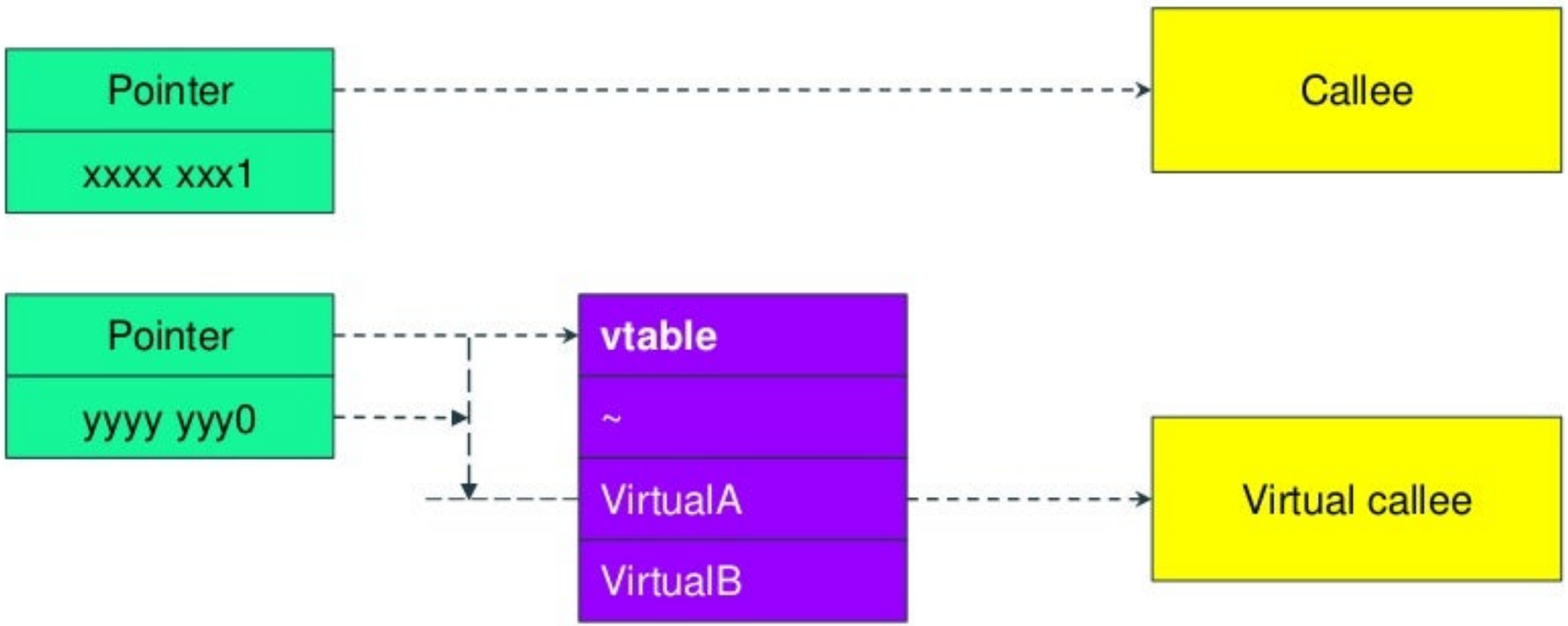
```
movs        r2, #1
add         r0, sp, #16
mov         r1, r4
bl          BaseClass::GetMethod(int)
ldr         r0, [sp, #20]
ldr         r3, [sp, #16]
lsls        r2, r0, #31
it                      pl
addpl.w     r0, r4, r0, asr #1
bpl.n       @1
asrs        r2, r0, #1
adds        r0, r4, r2
ldr         r2, [r4, r2]
ldr         r3, [r2, r3]

@1:
movs        r1, #42
blx         r3
```

Указатель на метод

```
C++
class BaseClass
{
public:
  typedef ptrdiff_t
(BaseClass::*MethodPtr) (ptrdiff_t);

  MethodPtr GetMethod(ptrdiff_t ptr);
};

BaseClass* obj = new DerivedClass();
BaseClass::MethodPtr method = obj-
>GetMethod(0);
ptrdiff_t value = (obj->*method)(42) +
1;
```

```
PowerPC
mr                      r3,r31
li                      r4,1
bl                      BaseClass::GetMethod(int)
mr                      r0,r3
add                     r11,r31,r4
andi.       r9,r0,1
beq-        @1

lwzx        r9,r31,r4
add                     r9,r9,r0
lwz                     r0,-1(r9)

@1:
mr                      r3,r11
li                      r4,42
mtctr       r0
bctrl
```

Указатель на метод



Тривиальные глобальные данные

```C++
int Trivial;

Point EmptyPoint;

int Initialized = 42;
```

Ничего не стоят в процессе выполнения.

Неинициализированные глобальные данные заполняются нулевыми байтами и не требуют хранения в загрузочном образе.

Инициализированные данные хранятся в виде начальных значений в загрузочном образе.

Статические переменные функции

C++

```
int& GetStaticA()
{
    static int s;
    return s;
}

int& GetStaticB()
{
    static int s = 42;
    return s;
}
```

x86

```
mov         eax,407544h
ret

mov         eax,407040h
ret
```

x64

```
lea         rax,[140008918h]
ret

lea         rax,[14000807Ch]
ret
```

Статические переменные функции

C++

```
int& GetStaticA()
{
    static int s;
    return s;
}

int& GetStaticB()
{
    static int s = 42;
    return s;
}
```

ARM

```
movw    r0, #11528    ; 0x2d08
movt    r0, #1
bx      lr

movw    r0, #10404
movt    r0, #1
bx      lr
```

AVR

```
ldi     r24, 0x4E
ldi     r25, 0x01
ret

ldi     r24, 0x00     ; 0
ldi     r25, 0x01     ; 1
ret
```

Статические переменные функции

C++

```
int& GetStaticA()
{
    static int s;
    return s;
}

int& GetStaticB()
{
    static int s = 42;
    return s;
}
```

PowerPC

```
lis         r3, 0x6
addi        r3,r3,0x964C
blr

lis         r3,0x5
addi        r3,r3,0xECC8
blr
```

MicroBlaze

```
imm    0x9009
addik  r3, r0, 0x7030
rtsd   r15, 8

imm    0x9008
addik  r3, r0, 0x7520
rtsd   r15, 8
```

Статические переменные функции

```
C++
int InitStaticC();

int& GetStaticC()
{
    static int s = InitStaticC();
    return s;
}
```

```
x86
mov     eax,fs:[0000002Ch]
mov     ecx,[__tls_index]
mov     ecx,[eax+ecx*4]
mov     eax,[0040754Ch]
cmp     eax,[ecx+4]
jle     @1
push    40754Ch
call    __Init_thread_header
add     esp,4
cmp     [40754Ch],0FFFFFFFFh
jne     @1
mov     dword ptr [ebp-4],0
```

```
call    InitStaticC
push    40754Ch
mov     dword ptr [00407548h],eax
call    __Init_thread_footer
add     esp,4

@1:
mov     eax,407548h
ret
```

Статические переменные функции

C++

```
BaseClass* GetStaticD()
{
    static DerivedClass obj;
    return &obj;
}
```

x86

```
mov     eax,fs:[0000002Ch]
mov     ecx,[__tls_index]
mov     ecx,[eax+ecx*4]
mov     eax,[0040756Ch]
cmp     eax,[ecx+4]
jle     @1
push    40756Ch
call    __Init_thread_header
add     esp,4
cmp     [40756Ch],0FFFFFFFFh
jne     @1
mov     dword ptr [ebp-4],0
mov     ecx,407550h
call    ??0DerivedClass@@QAE@XZ
push    @2
call    _atexit
mov     [407550h],eax
call    __Init_thread_footer
add     esp,4
@1:
mov     eax,407550h
ret

@2:
mov     ecx,407550h
jmp     ??1BaseClass@@UAE@XZ
```

Статические переменные функции

C++

```
int InitStaticC();

int& GetStaticC()
{
    static int s = InitStaticC();
    return s;
}
```

x64

```
mov     ecx,[_tls_index]
mov     rax,gs:[58h]
mov     edx,4
mov     rcx,[rax+rcx*8]
mov     eax,[rdx+rcx]
cmp     [140008920h],eax
jle     @1

lea     rcx,[140008920h]
call    _Init_thread_header
cmp     [140008920h], -1
jne     @1
```

```
call    InitStaticC
mov     [14000891Ch],eax

lea     rcx,[140008920h]
call    _Init_thread_footer

@1:
lea     rax,[14000891Ch]
ret
```

Статические переменные функции

C++

```
BaseClass* GetStaticD()
{
    static DerivedClass obj;
    return &obj;
}
```

x64

```
mov     ecx,[_tls_index]
mov     rax,gs:[58h]
mov     edx,4
mov     rcx,[rax+rcx*8]
mov     eax,[rdx+rcx]
cmp     [140008960h],eax
jle     @1
lea     rcx,[140008960h]
call    _Init_thread_header
cmp     [140008960h],-1
jne     @1

lea     rbx,[14000891Ch]
mov     rcx,rbx
call    ??0DerivedClass@@QEAA@XZ
```

```
lea     rcx,[@3]
call    atexit
nop
lea     rcx,[140008960h]
call    _Init_thread_footer
mov     rax,rbx
jmp      @2

@1:
lea     rax,[14000891Ch]

@2:
ret

@3:
lea     rcx,[140008928h]
jmp     ??1BaseClass@@UEAA@XZ
```

Статические переменные функции

C++

```
int InitStaticC();

int& GetStaticC()
{
    static int s = InitStaticC();
    return s;
}
```

ARM

```
push   {r4, lr}
ldr    r4, [pc, #56] // @d1
ldr.w  r3, [r4, #200]
lsls   r1, r3, #31
bpl.n  @1
@2:
ldr    r0, [pc, #48]    ; @d2
pop    {r4, pc} // return

@1:
add.w  r0, r4, #200
blx    8958 <_init+0x54>
cmp    r0, #0
beq.n  @2
bl     <test_6::InitStaticC()>

str.w  r0, [r4, #204]
add.w  r0, r4, #200
```

```
blx    89a4 <_init+0xa0>
              // atexit

ldr    r0, [pc, #16]
pop    {r4, pc} // return

ldr    r0, [pc, #16] // @d1
blx    8a10 <_init+0x10c>
blx    898c <_init+0x88>
nop

@d1: .word    0x00012dd4
      //GetStaticC()::s

@d2: .word    0x00012dd0
      // guard GetStaticC()::s
```

Статические переменные функции

C++

```
int InitStaticC();

int& GetStaticC()
{
    static int s = InitStaticC();
    return s;
}
```

AVR

```
push    r28
push    r29
lds     r24, 0x0152 ; <guard GetStaticC()::s>
cpse    r24, r1
rjmp    @1
call          InitStaticC
sts     0x0151, r25
sts     0x0150, r24 ; <GetStaticC()::s>
ldi     r24, 0x01
sts     0x0152, r24 ; <guard GetStaticC()::s>
@1:
ldi     r24, 0x50     ; 80
ldi     r25, 0x01     ; 1
pop     r29
pop     r28
ret

00800152 00000008 b guard variable for test_6::GetStaticC()::s
```

Статические переменные функции

C++

```
int InitStaticC();

int& GetStaticC()
{
    static int s = InitStaticC();
    return s;
}
```

PowerPC

```
stwu            r1,-24(r1)
mflr            r0
stw             r30,16(r1)
lis             r30,6
stw             r31,20(r1)
addi            r31,r30,-27072
mr              r3,r31
stw             r29,12(r1)
stw             r0,28(r1)
bl
                <__cxa_guard_acquir
e>
cmpwi   cr7,r3,0
lis             r29,6
bne-            cr7,@1
lwz             r0,28(r1)
addi            r3,r29,-27064
lwz             r30,16(r1)
lwz             r29,12(r1)
lwz             r31,20(r1)
mtlr            r0
addi            r1,r1,24
blr
@1:
bl
                <test_6::InitStatic
C()>
stw             r3,-27064(r29)
mr              r3,r31
bl
                <__cxa_guard_releas
e>
lwz             r0,28(r1)
addi            r3,r29,-27064
lwz             r30,16(r1)
lwz             r29,12(r1)
lwz             r31,20(r1)
mtlr            r0
addi            r1,r1,24
blr
mr              r29,r3
addi            r3,r30,-27072
bl              <__cxa_guard_abort>
mr              r3,r29
bl              <_Unwind_Resume>
```

Статические переменные функции

C++

```
BaseClass* GetStaticE()
{
 static char obj[sizeof(DerivedClass)];
 static BaseClass* target;

 if (!target)
 {
  target = new(obj)DerivedClass();
 }

 return target;
}
```

x86

```
mov     eax,dword ptr ds:[0040758Ch]
test    eax,eax
jne     @1

mov     dword ptr [ebp-10h],407570h
mov     ecx,407570h
mov     dword ptr [ebp-4],eax
call    ??0DerivedClass@@QAE@XZ
mov     dword ptr ds:[0040758Ch],eax
@1:
ret
```

Статические переменные функции

C++

```
BaseClass* GetStaticE()
{
 static char obj[sizeof(DerivedClass)];
 static BaseClass* target;

 if (!target)
 {
  target = new(obj)DerivedClass();
 }

 return target;
}
```

x64

```
mov     rax,qword ptr [1400089A0h]
test    rax,rax
jne     @1
lea     rcx,[140008968h]
mov     qword ptr [rsp+40h],rcx
call    ??0DerivedClass@@QEAA@XZ
nop

mov     qword ptr [1400089A0h],rax
@1:
ret
```

Статические переменные функции

C++

```
BaseClass* GetStaticE()
{
 static char obj[sizeof(DerivedClass)];
 static BaseClass* target;

 if (!target)
 {
  target = new(obj)DerivedClass();
 }

 return target;
}
```

ARM

```
push    {r3, r4, r5, lr}
ldr     r4, [pc, #20] ; @d1
ldr     r0, [r4, #32]
cbz     r0, @1
pop     {r3, r4, r5, pc}

@1:
adds    r5, r4, #4
mov     r0, r5
bl      <DerivedClass::DerivedClass()>
mov     r0, r5
str     r5, [r4, #32]
pop     {r3, r4, r5, pc}

@d1:            .word 0x00012d08
```

Статические переменные функции

C++

```
BaseClass* GetStaticE()
{
 static char obj[sizeof(DerivedClass)];
 static BaseClass* target;

 if (!target)
 {
  target = new(obj)DerivedClass();
 }

 return target;
}
```

AVR

```
lds     r24, 0x017E
lds     r25, 0x017F
or      r24, r25
brne    @1

sts     0x0173, r1
sts     0x0172, r1
sts     0x0175, r1
sts     0x0174, r1
sts     0x0177, r1
sts     0x0176, r1
ldi     r24, 0x3A
ldi     r25, 0x01

sts     0x0171, r25
sts     0x0170, r24

sts     0x0179, r1
sts     0x0178, r1
sts     0x017B, r1
sts     0x017A, r1
sts     0x017D, r1
sts     0x017C, r1

ldi     r24, 0x70
ldi     r25, 0x01
sts     0x017F, r25
sts     0x017E, r24

@1:
lds     r24, 0x017E
lds     r25, 0x017F
ret
```

Статические переменные функции

C++

```
BaseClass* GetStaticE()
{
 static char obj[sizeof(DerivedClass)];
 static BaseClass* target;

 if (!target)
 {
  target = new(obj)DerivedClass();
 }

 return target;
}
```

PowerPC

```
stwu        r1,-16(r1)
mflr        r0
stw         r0,20(r1)
stw         r31,12(r1)
lis         r31,6
stw         r30,8(r1)
lwz         r0,-
27056(r31)
cmpwi  cr7,r0,0
beq-        cr7,@1
lwz         r0,20(r1)
lwz         r3,-
27056(r31)
lwz         r30,8(r1)
lwz         r31,12(r1)
mtlr        r0
addi        r1,r1,16
blr
```

```
lis         r4,6
li          r3,28
addi        r4,r4,-25724
bl          <operator new(unsigned int,
void*)>
mr          r30,r3
bl
            <DerivedClass::DerivedClass(
)>
stw         r30,-27056(r31)
lwz         r0,20(r1)
lwz         r30,8(r1)
@1:
lwz         r3,-27056(r31)
lwz         r31,12(r1)
mtlr        r0
addi        r1,r1,16
blr
```

Статические переменные функции

C++

```
BaseClass* GetStaticE()
{
 static char obj[sizeof(DerivedClass)];
 static BaseClass* target;

 if (!target)
 {
  target = new(obj)DerivedClass();
 }

 return target;
}
```

MicroBlaze

```
imm     -28663
lwi     r3, r0, 28640
addik   r1, r1, -28
beqid   r3, @1
swi     r15, r1, 0
lwi     r15, r1, 0
rtsd    r15, 8
addik   r1, r1, 28
@1:
imm     -28663
addik   r5, r0, 28644
imm     -1
brlid   r15, <DerivedClass::DerivedClass()>
imm     -28663
addik   r3, r0, 28644
lwi     r15, r1, 0
imm     -28663
swi     r3, r0, 28640
```

Статические переменные функции

C++

```
template<int X>
Point GetStaticF()
{
    static Point staticFPoint;
    return staticFPoint;
}
```

x86

```
??$GetStaticF@$00@test_6@@YA?AUPoint@@XZ:
mov             eax,
[?staticFPoint@?1???$GetStaticF@$00@test_6@@YA?AUPoint@@XZ@4U2@A]
mov             edx,dword ptr ds:[407594h]
ret

??$GetStaticF@$01@test_6@@YA?AUPoint@@XZ:
mov    eax, [?staticFPoint@?1???$GetStaticF@$01@test_6@@YA?AUPoint@@XZ@4U2@A]
mov    edx,dword ptr ds:[40759Ch]
ret

??$GetStaticF@$02@test_6@@YA?AUPoint@@XZ:
mov    eax, [?staticFPoint@?1???$GetStaticF@$02@test_6@@YA?AUPoint@@XZ@4U2@A]
mov    edx,dword ptr ds:[4075A4h]
ret
```

Статические переменные функции

C++

```
template<int X>
Point GetStaticF()
{
    static Point staticFPoint;
    return staticFPoint;
}
```

x64

```
??$GetStaticF@$00@test_6@@YA?AUPoint@@XZ:
movups  xmm0,xmmword ptr \
            [?staticFPoint@?1???$GetStaticF@$00@test_6@@YA?AUPoint@@XZ@4U2@A]
mov     rax,rcx
movups  xmmword ptr [rcx],xmm0
ret

??$GetStaticF@$01@test_6@@YA?AUPoint@@XZ:
movups  xmm0,xmmword ptr \
            [?staticFPoint@?1???$GetStaticF@$01@test_6@@YA?AUPoint@@XZ@4U2@A]
mov     rax,rcx
movups  xmmword ptr [rcx],xmm0
ret

??$GetStaticF@$02@test_6@@YA?AUPoint@@XZ:
movups  xmm0,xmmword ptr \
            [?staticFPoint@?1???$GetStaticF@$02@test_6@@YA?AUPoint@@XZ@4U2@A]
mov     rax,rcx
movups  xmmword ptr [rcx],xmm0
ret
```

Статические переменные функции

C++

```
template<int X>
Point GetStaticF()
{
    static Point staticFPoint;
    return staticFPoint;
}
```

ARM

```
<Point test_6::GetStaticF<1>()>:
movw      r2, #11788
movt      r2, #1
mov       r3, r0
ldmia.w   r2, {r0, r1}
stmia.w   r3, {r0, r1}
mov       r0, r3
bx        lr
nop
<Point test_6::GetStaticF<2>()>:
movw      r2, #11780
movt      r2, #1
mov       r3, r0
ldmia.w   r2, {r0, r1}
stmia.w   r3, {r0, r1}
mov       r0, r3
bx        lr
nop
```

```
<Point test_6::GetStaticF<3>()>:
movw      r2, #11772
movt      r2, #1
mov       r3, r0
ldmia.w   r2, {r0, r1}
stmia.w   r3, {r0, r1}
mov       r0, r3
bx        lr
nop
```

Статические переменные функции

C++

```
template<int X>
Point GetStaticF()
{
    static Point staticFPoint;
    return staticFPoint;
}
```

AVR

```
<Point test_6::GetStaticF<1>()>:
lds     r22, 0x0186
lds     r23, 0x0187
lds     r24, 0x0188
lds     r25, 0x0189
ret
<Point test_6::GetStaticF<2>()>:
lds     r22, 0x018A
lds     r23, 0x018B
lds     r24, 0x018C
lds     r25, 0x018D
ret
<Point test_6::GetStaticF<3>()>:
lds     r22, 0x018E
lds     r23, 0x018F
lds     r24, 0x0190
lds     r25, 0x0191
ret
```

Статические переменные функции

C++

```
template<int X>
Point GetStaticF()
{
    static Point staticFPoint;
    return staticFPoint;
}
```

PowerPC

```
<Point test_6::GetStaticF<1>()>:
lis             r9,6
addi            r9,r9,-26872
lwz             r3,0(r9)
lwz             r4,4(r9)
blr
<Point test_6::GetStaticF<2>()>:
lis             r9,6
addi            r9,r9,-26880
lwz             r3,0(r9)
lwz             r4,4(r9)
blr
<Point test_6::GetStaticF<3>()>:
lis             r9,6
addi            r9,r9,-26888
lwz             r3,0(r9)
lwz             r4,4(r9)
blr
```

Статические переменные функции

C++

```
template<int X>
Point GetStaticF()
{
    static Point staticFPoint;
    return staticFPoint;
}
```

MicroBlaze

```
<Point test_6::GetStaticF<1>()>:
addk    r3, r5, r0
imm     -28663
lwi     r4, r0, 29140
imm     -28663
lwi     r5, r0, 29144
swi     r4, r3, 0
swi     r5, r3, 4
rtsd    r15, 8
<Point test_6::GetStaticF<2>()>:
addk    r3, r5, r0
imm     -28663
lwi     r4, r0, 29132
imm     -28663
lwi     r5, r0, 29136
swi     r4, r3, 0
swi     r5, r3, 4
rtsd    r15, 8
```

```
<Point test_6::GetStaticF<3>()>:
addk    r3, r5, r0
imm     -28663
lwi     r4, r0, 29124
imm     -28663
lwi     r5, r0, 29128
swi     r4, r3, 0
swi     r5, r3, 4
rtsd    r15, 8
```

Исключения

Реализация сильно зависит от компилятора и от операционной системы

В основе лежат нетривиальные концепции

Очень большой объём ассемблерного кода

Оставим для самостоятельного изучения

Итоги

1. Магии нет
2. Многое из сказанного - ложь
    - примеры кода сильно упрощены
    - приходилось принимать меры против оптимизаций
1. C++ - быстрый язык, потери производительности возникают в предсказуемых местах
2. Не бойтесь, но запаситесь терпением

Спасибо за внимание!

А теперь - вопросы!

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