COMPILER EXPLORER

```
class
{
public:
    std::string author = "Stefan Hammer";
    std::string date   = "November 10, 2017";

private:
    virtual void show_godbolt() noexcept;
    virtual void teach_asm() noexcept;
    virtual void live_demo(); //throws
}
```

shammer-linux.adtran.com:8001

WHAT IS COMPILER EXPLORER

CppCon2017 talk. Matt Godbolt.

WHAT IS COMPILER EXPLORER

THE GODBOLT

- Compiler
- Investigation tool
- Learning aid

http://godbolt.org

CE is a web based tool that wraps compiler, investigation tool, and learning aid into one package.

COMPILER

gcc, cppx, d, swift, haskell, go, ispc.
all available versions, historical, beta, nightly

INVESTIGATION TOOL

Ever wondered about the difference between two algorithms? Been concerned about the overhead in new C++ features? Curious about alternative compilers (CLang vs GCC)?

LEARNING

Learn ASM. Learn how compilers work. Learn C++ features.
More on this later.

QUICK ASSEMBLY OVERVIEW

ASSEMBLY TYPES

Intel

AT&T

I will use Intel. AT&T Syntax used by GNU Assembler

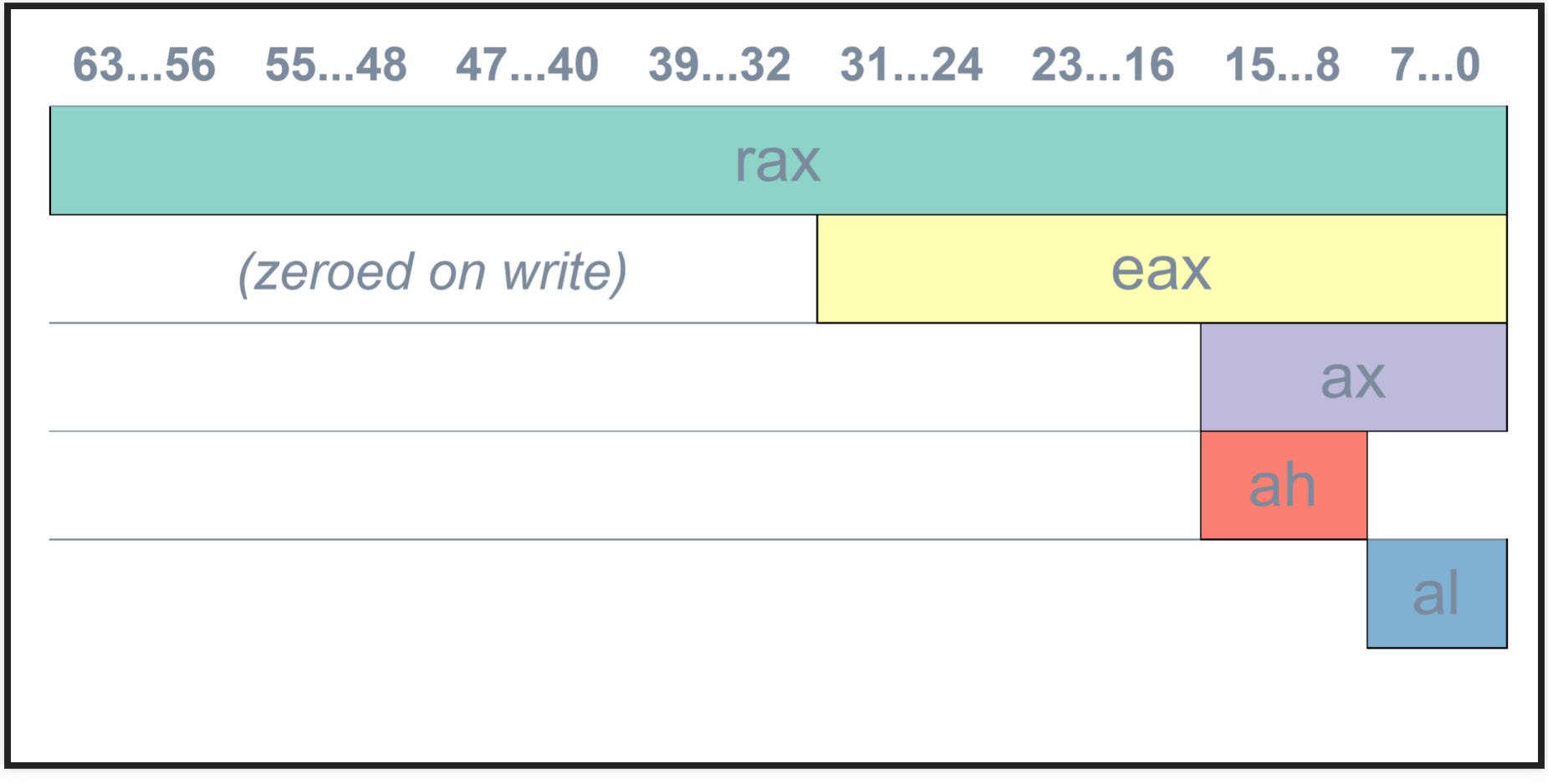
NAMES

- rax, rbx, rcx, rdx, rsp, rbp, rsi, rdi, r8-r15
- xmm1-xmm15
- rdi, rsi, rdx, rcx => arguments
- rax => return

http://www.swansontec.com/sregisters.html

- RAX - Accumulator
- RBX - Base
- RCX - Counter
- RDX - Data
- RSI - Source Index
- RDI - Destination Index
- RBP - Base Pointer
- RSP - Stack Pointer
- XMM - Streaming SIMD Ext (SSE) - performing same op on mult data objs

SIZES



From "What Has My Compiler Done for Me Lately", M. Godbolt

OPERATIONS

```
op
op dest
op dest, src
op dest, src1, src2
```

- dest, src is register or memory reference
- [base + $reg1_{opt}$ + reg2*$(1, 2, 4, 8)_{opt}$]

BACK TO THE GODBOLT

GODBOLT ON THE INTERNET

- On AWS - godbolt.org
- On GitHub - mattgodbolt/compiler-explorer
- On DockerHub - mattgodbolt/compiler-explorer

AWS hosted solution supports almost any compiler imaginable. Should be restrited to non-proprietary code.

GitHub to contribute, or to clone and run locally (node.js on Linux, scans for installed compilers). Good solution for testing things with sensetive IP.

DockerHub has multiple personal created Docker images with a variety of use cases. As always, look at what the Dockerimage is doing before blindly running it. Godbolt's image requires you to mount your own compilers.

USES

- Education
- Optimization
- Comparing compilers

EDUCATION

- Flag "-
  O0"

Want to disable optimization to understand basic compiler operation and assembly.

EDUCATION (...WE'LL DO IT LIVE)

```
1   int testFunction(int* input, int length) {
2     int sum = 0;
3     for (int i = 0; i < length; ++i) {
4       sum += input[i];
5     }
6     return sum;
7   }
8
```

x86-64 gcc 7.2 | -O0

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1  testFunction(int*, int):
2    push rbp
3    mov rbp, rsp
4    mov QWORD PTR [rbp-24], rdi
5    mov DWORD PTR [rbp-28], esi
6    mov DWORD PTR [rbp-4], 0
7    mov DWORD PTR [rbp-8], 0
8  .L3:
9    mov eax, DWORD PTR [rbp-8]
10   cmp eax, DWORD PTR [rbp-28]
11   jge .L2
12   mov eax, DWORD PTR [rbp-8]
13   cdqe
14   lea rdx, [0+rax*4]
15   mov rax, QWORD PTR [rbp-24]
16   add rax, rdx
17   mov eax, DWORD PTR [rax]
18   add DWORD PTR [rbp-4], eax
19   add DWORD PTR [rbp-8], 1
20   jmp .L3
21 .L2:
22   mov eax, DWORD PTR [rbp-4]
23   pop rbp
24   ret
```

Edit on C++ Compiler Explorer

(/#g:!((g:!((g:!((h:codeEditor,i:(j:1,source:'int+testFunction(int*+input,+int+length)+%7B%0A++int+sum+%3D+0%3B%0A++for+
2Bi)+%7B%0A+++sum+%2B%3D+input%5Bi%5D%3B%0A++%7D%0A++return+sum%3B%0A%7D%0A'),l:'5',n:'0',o:'C%2B%2B+source+%231',t:'0')),k:50,l:'4',m:100,n:'0',o:'',s:0,t:'0'),
,filters:(b:'0',binary:'1',commentOnly:'0',demangle:'0',directives:'0',execute:'1',intel:'0',trim:'0',undefined:'1'),libs:!(),options:'-O0',source:1),l:'5',n:'0',o:'x86-64+gcc+7.2+
(Editor+%231,+Compiler+%231)',t:'0')),header:(),k:50,l:'4',n:'0',o:'',s:0,t:'0')),l:'2',n:'0',o:'',t:'0')),version:4)

UNDERSTAND YOUR COMPILER

- What does your compiler do for you?
- What can you do for your compiler?

COMPILER OPTIMIZATION FLAGS

```
1   int testFunction(int* input, int length) {
2     int sum = 0;
3     for (int i = 0; i < length; ++i) {
4       sum += input[i];
5     }
6     return sum;
7   }
8
```

Edit on C++ Compiler Explorer

(/#g:!((g:!((g:!((h:codeEditor,i:(j:1,source:'int+testFunction(int*+input,+int+length)+%7B%0A++int+sum+%3D+0%3B%0A++for+
2Bi)+%7B%0A+++sum+%2B%3D+input%5Bi%5D%3B%0A++%7D%0A++return+sum%3B%0A%7D%0A'),l:'5',n:'0',o:'C%2B%2B+source+%231',t:'0')),k:50,l:'4',m:100,n:'0',o:'',s:0,t:'0'),
,filters:(b:'0',binary:'1',commentOnly:'0',demangle:'0',directives:'0',execute:'1',intel:'0',trim:'0',undefined:'1'),libs:!(),options:'-O0',source:1),l:'5',n:'0',o:'x86-64+gcc+7.2+
(Editor+%231,+Compiler+%231)',t:'0')),k:50,l:'4',n:'0',o:'',s:0,t:'0')),l:'2',n:'0',o:'',t:'0')),version:4)

x86-64 gcc 7.2 | -O0

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1  testFunction(int*, int):
2    push rbp
3    mov rbp, rsp
4    mov QWORD PTR [rbp-24], rdi
5    mov DWORD PTR [rbp-28], esi
6    mov DWORD PTR [rbp-4], 0
7    mov DWORD PTR [rbp-8], 0
8  .L3:
9    mov eax, DWORD PTR [rbp-8]
10   cmp eax, DWORD PTR [rbp-28]
11   jge .L2
12   mov eax, DWORD PTR [rbp-8]
13   cdqe
14   lea rdx, [0+rax*4]
15   mov rax, QWORD PTR [rbp-24]
16   add rax, rdx
17   mov eax, DWORD PTR [rax]
18   add DWORD PTR [rbp-4], eax
19   add DWORD PTR [rbp-8], 1
20   jmp .L3
21 .L2:
22   mov eax, DWORD PTR [rbp-4]
23   pop rbp
24   ret
```

[https://goo.gl/85Rw3J] Optimization level, O1, O2. From 0 to 1, to 2 tries to reduce number of operations (redundancy, alternate ops, etc). O3 for the real optimizations. Number of instructions != execution speed. In modern CPUs, many instructions will be prefetched in L1 cache. CPU architecture and type become very important.

OPTIMIZATIONS

Let's take a look at some optimization the compiler does for us from source.

AVOID PREMATURE OPTIMIZATION!!!

Don't sacrifice readability, maintainability, robustnes for performance. Don't spend time optimizing code that may not be in the critical path. In the following slides we will look at how the compiler handles and optimizes different use cases.

PROFILE YOUR CODE!!!

(Assembly does not tell the whole story)

DEMO

EFFECTS OF VIRTUAL, OWNERSHIP, AND <MEMORY>

SPOILER:

Trust your compiler.

I'M A LEAF ON THE WIND

```
#include <memory>
using namespace std;

using Time = int;
using Speed = int;
using Distance = int;

struct Ship {
  virtual Distance fly(Speed, Time) = 0;
};

struct Firefly : public Ship {
  static constexpr Speed max_speed = 7;
  virtual Distance fly(Speed speed, Time time) {
    return speed * time;
  }
};

struct Captain {
  Distance run(Speed speed, Time time) {
    return ship.fly(speed, time);
  }
private:
  Firefly ship;
};

Distance flee_from_feds(Time time) {
  Captain mal;
  return mal.run(Firefly::max_speed, time);
}
```

in+mal%3B%0A++return+mal.run(Firefly::max_speed,+time)%3B%0A%7D'),l:'5',n:'0',o:'C%2B%2B+source+%231',t:'0')),j:__glMaximised,k:100,l:'4',m:100,n:'0',o:'',s:0,t:'0')),version:4)

https://goo.gl/uDciFf
This is the code that I will be playing around with.

- `Ship` interface
- `Firefly` implementation
- `Captain` that owns a `Firefly`
- `flee_from_feds` is our test function

WATCH HOW I SOAR

```
#include <memory>
using namespace std;

using Time = int;
using Speed = int;
using Distance = int;

struct Ship {
  virtual Distance fly(Speed, Time) = 0;
};

struct Firefly : public Ship {
  static constexpr Speed max_speed = 7;
  virtual Distance fly(Speed speed, Time time) {
    return speed * time;
  }
};

struct Captain {
  Distance run(Speed speed, Time time) {
    return ship.fly(speed, time);
  }
private:
  Firefly ship;
};

Distance flee_from_feds(Time time) {
  Captain mal;
  return mal.run(Firefly::max_speed, time);
}
```

x86-64 gcc 4.9.0

-O3 -mtune=atom -std=c++14

11010 .LX0: .text // \s+ Intel Demangle

```
flee_from_feds(int):
  lea eax, [0+rdi*8]
  sub eax, edi
  nop
  nop
  nop
  nop
  ret
```

ine+time)+%7B%0A++Captain+mal%3B%0A++return+mal.run(Firefly::max_speed,+time)%3B%0A%7D'),l:'5',n:'0',o:'C%2B%2B+source+%231',t:'0')),k:50,l:'4',m:100,n:'0',o:'',s:0,t:'0'),

https://goo.gl/9jEZWR
Here is the assembly. Simple, efficient, shiny.

WHO'S FLYING THIS THING?

```
struct Captain {
    ...
private:
  Firefly ship;
};

Distance flee_from_feds(Time time) {
  Captain mal;
  return mal.run(Firefly::max_speed, time);
}
```

```
struct Captain {
  Captain(Ship &ship_) : ship(ship_) {}
    ...
private:
  Ship &ship;
};

Distance flee_from_feds(Time time) {
  Firefly serenity;
  Captain mal(serenity);
  return mal.run(serenity.max_speed, time);
}
```

Change Captain to take a reference, `Ship&`, instead of a fixed member, `Firefly`.

WHO'S FLYING THIS THING? ...

```
#include <memory>
using namespace std;

```

```
#include <memory>
using namespace std;

using Time = int;
using Speed = int;
using Distance = int;

struct Ship {
  virtual Distance fly(Speed, Time) = 0;
};

struct Firefly : public Ship {
  static constexpr Speed max_speed = 7;
  virtual Distance fly(Speed speed, Time time) {
    return speed * time;
  }
};

struct Captain {
  Captain(Ship &ship_) : ship(ship_) {}
  Distance run(Speed speed, Time time) {
    return ship.fly(speed, time);
  }
private:
  Ship &ship;
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -std=c++14

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
flee_from_feds(int):
  lea eax, [0+rdi*8]
  sub eax, edi
  nop
  nop
  nop
  nop
  ret
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -std=c++14

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
flee_from_feds(int):
  lea eax, [0+rdi*8]
  sub eax, edi
  nop
  nop
  nop
  nop
  ret
```

6JycMnaQR53xcH4JnK62cia53zoXYupdpDl0rtXWuZA26vibqzFu9chAEM7ljcg31frr0gbjMGC5Vz8B1FWBGYjYaT2JmjWeXdsYgx4PjSRB4OEjlxm9JRqNSayPIC/QWvt6wziAA%3D%3D)

https://goo.gl/DtC3zV

Change Captain to take a reference instead of a fixed member.

ADD VIRTUAL MAXSPEED

```
struct Captain {
  Distance run(Speed speed, Time time) {
    return ship.fly(speed, time);
  }
};
```

```
struct Ship {
  virtual Speed maxSpeed() const = 0;
};

struct Firefly : public Ship {
  virtual Speed maxSpeed() const { return max_speed; }
};

struct Captain {
  Distance run(Time time) {
    return ship.fly(ship.maxSpeed(), time);
  }
};
```

Add `virtual maxSpeed` instead of passing it into run().
Expect this to be virtual function call overhead. Every call to `run` requires a virtual lookup for `maxSpeed`.

ADD VIRTUAL MAXSPEED ...

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
9     virtual Distance fly(Speed, Time) = 0;
10  };
11
12  struct Firefly : public Ship {
13    static constexpr Speed max_speed = 7;
14    virtual Distance fly(Speed speed, Time time) {
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -std=c++14

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 flee_from_feds(int):
2   lea eax, [0+rdi*8]
3   sub eax, edi
4   nop
5   nop
6   nop
7   nop
8   ret
```

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
9     virtual Distance fly(Speed, Time) = 0;
10    virtual Speed maxSpeed() const = 0;
11  };
12
13  struct Firefly : public Ship {
14    static constexpr Speed max_speed = 7;
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -std=c++14

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1  Firefly::fly(int, int):
2    lea eax, [rsi]
3    imul eax, edx
4    nop
5    nop
6    nop
7    nop
8    ret
9  Firefly::maxSpeed() const:
10   mov eax, 7
11   nop
12   nop
```

G6skCM3RQrcHDt07k4bu0glK4S4PwQeqNh5y1HhPKeM854LyXq6Fea9CAbwPthHehs96bwJkfRWJ9Sbk0ptTFGV9Pz80xgrSCj89Fn1MQzaWZj5bH3FijQmDj766Oceg62Zd1wviAA%3D)

https://goo.gl/j8vQaB

Add `virtual maxSpeed` instead of passing it into run().

Expect this to be virtual function call overhead. Every call to `run` requires a virtual lookup for `maxSpeed`.

SHIP REF TO CAPTAIN CTOR

```
Distance flee_from_feds(Time time) {
  Firefly serenity;
  Captain mal(serenity);
}
```

```
Distance flee_from_feds(Time time) {
  Firefly serenity;
  Ship &ship = serenity;
  Captain mal(ship);
}
```

SKIP IF NEED TIME.
Just to make sure that it's not infering data from the function call, let's pass in a `Ship&`

SHIP REF TO CAPTAIN CTOR ...

```
#include <memory>
using namespace std;

```

```
#include <memory>
using namespace std;

using Time = int;
using Speed = int;
using Distance = int;

struct Ship {
  virtual Distance fly(Speed, Time) = 0;
  virtual Speed maxSpeed() const = 0;
};

struct Firefly : public Ship {
  static constexpr Speed max_speed = 7;
  virtual Distance fly(Speed speed, Time time) {
    return speed * time;
  }
  virtual Speed maxSpeed() const {
    return max_speed;
  }
};

struct Captain {
  Captain(Ship &ship_) : ship(ship_) {}
  Distance run(Time time) {
```

x86-64 gcc 4.9.0 | -O3 -mtune=

11010 | .LX0: | .text | // | \s+ | Intel

Demangle

```
Firefly::fly(int, int):
  lea eax, [rsi]
  imul eax, edx
  nop
  nop
  nop
  nop
  ret
Firefly::maxSpeed() const:
```

x86-64 gcc 4.9.0 | -O2 -mtune=

11010 | .LX0: | .text | // | \s+ | Intel

Demangle

```
Firefly::fly(int, int):
  lea eax, [rsi]
  imul eax, edx
  nop
  nop
  nop
  nop
  ret
Firefly::maxSpeed() const:
```

s40So/PWz9X7v0fiX%2B/9AHALIDAniEC46QNASLOBxtxaS2IrLeWyD0IISQug/WWCaLhPNiAQQWFGgUPVvwFOT5BBPifOrQQOsOae11j7TBYT/YONFuUjBBscFm0EZnFe74UJAA%3D)

https://goo.gl/uCd35H
SKIP IF NEED TIME.

STD::UNIQUE_PTR<T>

PASS OWNERSHIP USING UNIQUE_PTR

```
struct Captain {
  Captain(Ship &ship_) : ship(ship_) {}
}
```

```
struct Captain {
  Captain(unique_ptr<Firefly> ship_) : ship(move(ship_)) {}
}
Distance flee_from_feds(Time time) {
  auto ship = make_unique<Firefly>();
  Captain mal(move(ship));
}
```

Want to pass ownership. We'll use unique_ptr's.
The traditional way it to use raw pointers. :(We'll look at raw pointers if we have time.

PASS OWNERSHIP USING UNIQUE_PTR...

```
1   #include <memory>
2   using namespace std;
3
```

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
9     virtual Distance fly(Speed, Time) = 0;
10    virtual Speed maxSpeed() const = 0;
11  };
12
13  struct Firefly : public Ship {
14    static constexpr Speed max_speed = 7;
15    virtual Distance fly(Speed speed, Time time) {
16      return speed * time;
17    }
18    virtual Speed maxSpeed() const {
19      return max_speed;
20    }
21  };
22
23  struct Captain {
24    Captain(unique_ptr<Firefly> ship_) : ship(move(ship_)) {}
25    Distance run(Time time) {
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 Firefly::fly(int, int):
2   lea eax, [rsi]
3   imul eax, edx
4   nop
5   nop
6   nop
7   nop
8   ret
9 Firefly::maxSpeed() const:
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 Firefly::fly(int, int):
2   lea eax, [rsi]
3   imul eax, edx
4   nop
5   nop
6   nop
7   nop
8   ret
9 Firefly::maxSpeed() const:
```

4pmGsPYZw7HvD%2BGGLIFkSYjp4SOEQHGR%2Bdg7vjDhHYxBYXZKlwIhVRXdc4sVGeQdCIcMz0A9CRL8X4m7aCELXESLs26Z07jwWRKF46BxOfQnOmiQ7vTXog/8OEgA%3D%3D)

https://goo.gl/ptX29z

ABSTRACT UNIQUE_PTR

```
Captain(unique_ptr<Firefly> ship_) : ship(move(ship_)) {}
private:
  unique_ptr<Firefly> ship;
```

```
Captain(unique_ptr<Ship> ship_) : ship(move(ship_)) {}
private:
  unique_ptr<Ship> ship;
```

Want to use the interface abstraction, so we use the `Ship` interface pointer.

ABSTRACT UNIQUE_PTR

```
1   #include <memory>
2   using namespace std;
3
```

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
9     virtual Distance fly(Speed, Time) = 0;
10    virtual Speed maxSpeed() const = 0;
11  };
12
13  struct Firefly : public Ship {
14    static constexpr Speed max_speed = 7;
15    virtual Distance fly(Speed speed, Time time) {
16      return speed * time;
17    }
18    virtual Speed maxSpeed() const {
19      return max_speed;
20    }
21  };
22
23  struct Captain {
24    Captain(unique_ptr<Ship> ship_) : ship(move(ship_)) {}
25    Distance run(Time time) {
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 Firefly::fly(int, int):
2   lea eax, [rsi]
3   imul eax, edx
4   nop
5   nop
6   nop
7   nop
8   ret
9 Firefly::maxSpeed() const:
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 Firefly::fly(int, int):
2   lea eax, [rsi]
3   imul eax, edx
4   nop
5   nop
6   nop
7   nop
8   ret
9 Firefly::maxSpeed() const:
```

iQshFCqE0OkHQnh7OWGED7Y6lV2sx6DqL4YiQpgFGJex9n7AOoiuLMVYplhOMiFle0DnnSC9A%2BICTQmhcu2ghBFwMpbaucc67SJThMkObsNlSMTnlz28Zp5gOwuxIAA%3D%3D%3D)

https://goo.gl/qJgf1A

RAW POINTERS?

Change `unique_ptr` to `Ship*`

Who has ownership?

RAW POINTER IS NOT BETTER

```cpp
#include <memory>
using namespace std;

using Time = int;
using Speed = int;
using Distance = int;
```

```cpp
#include <memory>
using namespace std;

using Time = int;
using Speed = int;
using Distance = int;

struct Ship {
  virtual Distance fly(Speed, Time) = 0;
  virtual Speed maxSpeed() const = 0;
};

struct Firefly : public Ship {
  static constexpr Speed max_speed = 7;
  virtual Distance fly(Speed speed, Time time) {
    return speed * time;
  }
  virtual Speed maxSpeed() const {
    return max_speed;
  }
};
```

x86-64 gcc 4.9.0 -O3 -mtune=atom

11010 .LX0: .text // \s+ Intel

Demangle

```
Firefly::fly(int, int):
  lea eax, [rsi]
  imul eax, edx
  nop
  nop
  nop
  nop
  ret
Firefly::maxSpeed() const:
```

x86-64 gcc 4.9.0 -O3 -mtune=atom

11010 .LX0: .text // \s+ Intel

Demangle

```
Firefly::fly(int, int):
  lea eax, [rsi]
  imul eax, edx
  nop
  nop
  nop
  nop
  ret
Firefly::maxSpeed() const:
```

W3jZVR6jNHaN0a3AxRjCAmJsd5CxD8rGmNznYg%2BDiGLF1LmE5stdXEKSUh4veg8fFnPIPJQQb4yLMUEMxN89A3z8G0IILS/AW7eVrlvXunj7Gj0TnnSFjzTkwrjN/NhHEVJAA%3D%3D)

https://goo.gl/1kQ6ML

STD::SHARED_PTR<T>

SHARED OWNERSHIP

SHARED SHIP

```
struct Captain {
  Captain(unique_ptr<Firefly> ship_) : ship(move(ship_)) {}
private:
  unique_ptr<Firefly> ship;
}

Distance flee_from_feds(Time time) {
  auto ship = make_unique<Firefly>();
}
```

```
struct Captain {
  Captain(shared_ptr<Ship> ship_) : ship(move(ship_)) {}
private:
  shared_ptr<Ship> ship;
}

Distance flee_from_feds(Time time) {
  auto ship = make_shared<Firefly>();
}
```

Change from `unique_ptr` to `shared_ptr`. GCC was able to optimize `unique_ptrs` well, but `shared_ptrs`? Ouch!

SHARED SHIP ...

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
```

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
9     virtual Distance fly(Speed, Time) = 0;
10    virtual Speed maxSpeed() const = 0;
11  };
12
13  struct Firefly : public Ship {
14    static constexpr Speed max_speed = 7;
15    virtual Distance fly(Speed speed, Time time) {
16      return speed * time;
17    }
18    virtual Speed maxSpeed() const {
19      return max_speed;
20    }
```

x86-64 gcc 4.9.0 -O3

11010 .LX0: .text // \s+

Intel Demangle

```
1 Firefly::fly(int, int)
2   lea eax, [rsi]
3   imul eax, edx
4   nop
5   nop
6   nop
7   nop
8   ret
9 Firefly::maxSpeed() co
```

x86-64 gcc 4.9.0 -O3

11010 .LX0: .text // \s+

Intel Demangle

```
1 Firefly::fly(int, int
2   lea eax, [rsi]
3   imul eax, edx
4   nop
5   nop
6   nop
7   nop
8   ret
9 Firefly::maxSpeed() c
```

)8HoY7RhRdmGsPYexTh3DpC8P4YI4RZBZA%2B1mPlMxMiW4LHkZXQOwdQ7hxUWo5irFNG91LrohREdc6QUEIlUSvE9naHoFnPOaFm4qM7s7HuSlJkMRUb7C5WjrkB3jJvVB2F2JAA)

https://goo.gl/RZdJqs

OWNERSHIP VIA STACK

C++ developers should love the stack.

PASS FIREFLY OWNERSHIP ON STACK

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
```

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
9     virtual Distance fly(Speed, Time) = 0;
10    virtual Speed maxSpeed() const = 0;
11  };
12
13  struct Firefly : public Ship {
14    static constexpr Speed max_speed = 7;
15    virtual Distance fly(Speed speed, Time time) {
16      return speed * time;
17    }
18    virtual Speed maxSpeed() const {
19      return max_speed;
20    }
21  };
22
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 Firefly::fly(int, int):
2   lea eax, [rsi]
3   imul eax, edx
4   nop
5   nop
6   nop
7   nop
8   ret
9 Firefly::maxSpeed() const:
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 flee_from_feds(int):
2   lea eax, [0+rdi*8]
3   sub eax, edi
4   nop
5   nop
6   nop
7   nop
8   ret
```

LJRQqhNCg70MYYQZh3C9LsN7pwlh1teG134UhR2ztVHNm9pxJiLFxEmxrjJaR7TyCMUEHQ%2BggltCoVQkXbQQhc56X1uXKOQy%2BFJ1LjbdZ1dNn0XlHGCelCsJsSAA%3D%3D%3D)

CAN'T PASS SHIP OWNERSHIP ON STACK

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
```

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
9     virtual Distance fly(Speed, Time) = 0;
10    virtual Speed maxSpeed() const = 0;
11  };
12
13  struct Firefly : public Ship {
14    static constexpr Speed max_speed = 7;
15    virtual Distance fly(Speed speed, Time time) {
16      return speed * time;
17    }
18    virtual Speed maxSpeed() const {
19      return max_speed;
20    }
21  };
22
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 Firefly::fly(int, int):
2   lea eax, [rsi]
3   imul eax, edx
4   nop
5   nop
6   nop
7   nop
8   ret
9 Firefly::maxSpeed() const:
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 <Compilation failed>
```

hLEe4mTYRwrhPC%2BECPTsl0RhBxHyO8tlxesiJH%2B0UYPZRDFQ7h0sc2eOGiFJKW0f3Yu%2BipnkHkolIR9A9LaGYsxdu2ghBN28vHbu%2BcdFKPLs7AOlzVmTJuXGHe6COlqSAA)

https://goo.gl/V4p4s9F

TEMPLATED CAPTAIN FOR GENERIC SOLUTION

```
1  #include <memory>
2  using namespace std;
3
4  using Time = int;
5  using Speed = int;
6  using Distance = int;
```

```
1  #include <memory>
2  using namespace std;
3
4  using Time = int;
5  using Speed = int;
6  using Distance = int;
7
8  struct Ship {
9    virtual Distance fly(Speed, Time) = 0;
10   virtual Speed maxSpeed() const = 0;
11 };
12
13 struct Firefly : public Ship {
14   static constexpr Speed max_speed = 7;
15   virtual Distance fly(Speed speed, Time time) {
16     return speed * time;
17   }
18   virtual Speed maxSpeed() const {
19     return max_speed;
20   }
21 };
22
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 Firefly::fly(int, int):
2   lea eax, [rsi]
3   imul eax, edx
4   nop
5   nop
6   nop
7   nop
8   ret
9 Firefly::maxSpeed() const:
```

x86-64 gcc 4.9.0 | -O3 -mtune=atom -s

11010 | .LX0: | .text | // | \s+ | Intel | Demangle

```
1 flee_from_feds(int):
2   lea eax, [0+rdi*8]
3   sub eax, edi
4   nop
5   nop
6   nop
7   nop
8   ret
```

=Ilis8TLiMkdI2R8jFEVxUWowgGjXyCG8joo%2Bu8lIDKMR3VO6dM7Z3MepBSSlrELxbvY4xOdBDKPoHpbQzFmIT20ElYe3kC4zwbjYrZskQ6Jw%2BacpeFzyBxlvhQjiKkgA%3D%3D%3D)

https://goo.gl/keL4Vi

COMPARING COMPILERS

GCC 4.9 VS 7.2 (UNIQUE_PTR)

```cpp
#include <memory>
using namespace std;

using Time = int;
using Speed = int;
using Distance = int;

struct Ship {
  virtual Distance fly(Speed, Time) = 0;
  virtual Speed maxSpeed() const = 0;
};

struct Firefly : public Ship {
  static constexpr Speed max_speed = 7;
  virtual Distance fly(Speed speed, Time
    return speed * time;
  }
  virtual Speed maxSpeed() const {
    return max_speed;
  }
};

struct Captain {
  Captain(Ship &ship_) : ship(ship_) {}
  Distance run(Time time) {
    return ship.fly(ship.maxSpeed(), time
  }
private:
  Ship &ship;
};
```

x86-64 gcc 7.2 -O3 -mtune=a...

```
Firefly::fly(int, int):
  lea eax, [rsi]
  imul eax, edx
  nop
  nop
  nop
  nop
  ret
Firefly::maxSpeed() con
  mov eax, 7
  nop
  nop
  nop
  nop
  nop
  nop
  ret
flee_from_feds(int):
  push rbx
  lea ebx, [rdi]
  mov edi, 8
  call operator new(uns
  mov QWORD PTR [rax],
  lea rdi, [rax]
  mov esi, 8
  call operator delete(
  lea eax, [0+rbx*8]
  sub eax, ebx
  pop rbx
```

x86-64 gcc 4.9.0 -O3 -n

```
Firefly::fly(int, int):
  lea eax, [rsi]
  imul eax, edx
  nop
  nop
  nop
  nop
  ret
Firefly::maxSpeed() con
  mov eax, 7
  nop
  nop
  nop
  nop
  nop
  nop
  ret
flee_from_feds(int):
  push rbx
  lea ebx, [rdi]
  mov edi, 8
  call operator new(uns
  mov QWORD PTR [rax],
  lea rdi, [rax]
  call operator delete(
  mov eax, ebx
  sal eax, 3
  sub eax, ebx
  pop rbx
```

ake_unique%3Cship%3E()%3B%0A++Captain+mal(*ship)%3B%0A++Distance+d+%3D+mal.run(time)%3B%0A++return+d%3B%0A%7D'),l:'5',n:'0',o:'C%2B%2B+source+%231',t:'0'),

https://goo.gl/mTisX1

CLASS TEMPLATE DEDUCTION

Previous template class: https://godbolt.org/g/t96hhN

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
9     virtual Distance fly(Speed, Time) = 0;
10    virtual Speed maxSpeed() const = 0;
11  };
12
13  struct Firefly : public Ship {
14    static constexpr Speed max_speed = 7;
15    virtual Distance fly(Speed speed, Time time) {
16      return speed * time;
17    }
18    virtual Speed maxSpeed() const {
19      return max_speed;
20    }
21  };
22
23  template <typename T>
24  struct Captain {
25    Captain(T && ship_) : ship(move(ship_)) {}
26    Distance run(Time time) {
27      return ship.fly(ship.maxSpeed(), time);
```

x86-64 gcc 7.1 -O3 -mtu

11010 .LX0: .text // \s+ Intel

Demangle

```
1 flee_from_feds(int):
2   lea eax, [0+rdi*8]
3   sub eax, edi
4   nop
5   nop
6   nop
7   nop
8   ret
```

s(Time+time)+%7B%0A++Firefly+serenity%3B%0A++Captain+mal(move(serenity))%3B%0A++return+mal.run(time)%3B%0A%7D'),l:'5',n:'0',o:'C%2B%2B+source+%232',t:'0')),header:

https://goo.gl/GTgRkW

GCC VS CLANG

```
1   #include <memory>
2   using namespace std;
3
4   using Time = int;
5   using Speed = int;
6   using Distance = int;
7
8   struct Ship {
9     virtual Distance fly(Speed, Time) = 0;
10    virtual Speed maxSpeed() const = 0;
11  };
12
13  struct Firefly : public Ship {
14    static constexpr Speed max_speed = 7;
15    virtual Distance fly(Speed speed, Time time) {
16      return speed * time;
17    }
18    virtual Speed maxSpeed() const {
19      return max_speed;
20    }
21  };
22
23  struct Captain {
24    Captain(unique_ptr<Ship> ship_) : ship(move(ship_)) {}
25    Distance run(Time time) {
26      return ship->fly(ship->maxSpeed(), time);
27    }
28  private:
29    unique_ptr<Ship> ship;
30  };
```

Edit on C++ Compiler Explorer

(/)

https://goo.gl/nmoxu3

QUESTIONS?

Check out Matt Godbolt's
"What Has My Compiler Done for Me Lately? Unbolting the Compiler's Lid"
https://www.youtube.com/watch?v=bSkpMdDe4g4