Quick, but close look at

UNDEFINED BEHAVIOR

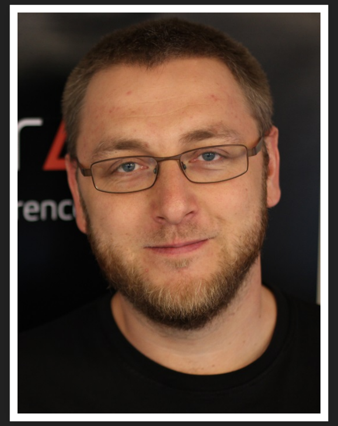
and

COMPILER OPTIMIZATIONS

Maciek Gajewski

ABOUT ME

- Maciek Gajewski <maciej.gajewski0@gmail.com>
- I work at Optiver, Amsterdam
- Role: C++ Developer and teacher



optiver

UNDEFINED BEHAVIOR

The popular definition:

“When the compiler encounters [a given undefined construct] it is legal for it to make demons fly out of your nose”

comp.std.c, 1992

UNDEFINED BEHAVIOR

More useful definition:

- Machine-dependent behavior that would be too costly to define
- Something, that compiler can assume you would never do

UNDEFINED BEHAVIOR

- Pretty much specific to C and C++
- Essential for some compiler optimizations
- (Yet another thing that makes C++) hard to teach
- Compiler Explorer is a great help

COMPILER EXPLORER

- https://godbolt.org/
- By Matt Godbolt
- Great tool for teachers and tweakers
- Basic assembly required
- CppCon 2017 *“What Has My Compiler Done for Me Lately?”*

X64 ASSEMBLY: REGISTERS

- Registers: **a, b, c, d, si, di, sp, bp, r8-r15, xmm0-xmm15**
- Register widths: **8: ah/al, 16: ax, 32: eax, 64: rax**
- Function params: **rdi, rsi, rdx, rcx, r8, r9**
- Function return value in: **rax, rdx, xmm0**

X64 ASSEMBLY: INSTRUCTIONS

Intel syntax

```
oper
oper dest
oper dest src
oper dest [src ptr]
```

ACCESSING ARRAY OUT OF BOUNDS

ARRAY BOUNDS - OPTIMIZATION

C++

```
int fun(int i)
{
        int array[4];
        array[i] = 333;
        return array[i];
}
```

asm

```
fun(int):
mov eax, 333
ret
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

ARRAY BOUNDS - OPTIMIZATION

C++

```
int fun(int i, int x)
{
        int array[4];
        array[i] = x;
        return array[i];
}
```

asm

```
fun(int, int):
mov eax, esi
ret
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

ARRAY BOUNDS - THE 5TH ELEMENT

C++

```
bool is_in_arr(int val)
{
        int array[] = {1, 2, 3, 5};

        for (int i = 0; i <= 4; i++)
        {
                if (array[i] == val)
                        return true;
        }

        return false;
}
```

asm

```
is_in_arr(int):
        mov eax, 1
        ret
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

SIGNED INTEGER OVERFLOW

INTEGER OVERFLOW - OPTIMIZATION

C++

```cpp
int foo(int a, int b)
{
        for(int i = 0; i < 10; i++)
        {
                if (b+i > b)
                        return 6;
                a++;
        }
        return a;
}
```

asm

```asm
foo(int, int):
        mov eax, 6
        ret
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

INTEGER OVERFLOW - OPTIMIZATION

C++

```
int foo(int a, unsigned b)
{
        for(int i = 0; i < 10; i++)
        {
                if (b+i > b)
                        return 6;
                a++;
        }
        return a;
}
```

asm

```
foo(int, unsigned int):
        mov eax, edi
        lea ecx, [rdi+10]
        jmp .L2
.L4:
        lea edx, [rax+rsi]
        sub edx, edi
        cmp esi, edx
        jb .L5
.L2:
        add eax, 1
        cmp eax, ecx
        jne .L4
        rep ret
.L5:
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

INTEGER OVERFLOW - OPTIMIZATION

C++

```
void zero_array(float* P, int offset)
{
   for (int i = 0; i != 10000; ++i)
         P[i+offset] = 0.0f;
}
```

asm (clang)

```
zero_array(float*, int): # @zero_array(float*, int)
        push rax
        movsxd rax, esi
        lea rdi, [rdi + 4*rax]
        xor esi, esi
        mov edx, 40000
        call memset
        pop rax
        ret
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

INTEGER OVERFLOW - OPTIMIZATION

C++

```cpp
void zero_array(float* P, unsigned offset)
{
   for (int i = 0; i != 10000; ++i)
         P[i+offset] = 0.0f;
}
```

asm (clang)

```asm
zero_array(float*, unsigned int): # @zero_array(float*, unsigned int)
        mov eax, esi
        xor ecx, ecx
        cmp esi, -10000
        jbe .LBB0_1
.LBB0_4: # =>This Inner Loop Header: Depth=1
        lea edx, [rax + rcx]
        mov dword ptr [rdi + 4*rdx], 0
        inc rcx
        cmp rcx, 10000
        jne .LBB0_4
        jmp .LBB0_3
.LBB0_1:
        xorps xmm0, xmm0
.LBB0_2: # =>This Inner Loop Header: Depth=1
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

NULL POINTER DEREFERENCE

NULL POINTER DEREFERENCE

C++

```
int some_other_fun();

void set_val(int* buf, int val)
{
	int v = *buf;
	if (buf)
		*buf = val;
	else
		*buf = val + some_other_fun();
}
```

asm

```
set_val(int*, int):
        mov DWORD PTR [rdi], esi
        ret
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

READING UNINITIALIZED LOCAL VARIABLE

UNINITIALIZED VARIABLE

C++

```
int type_to_code(char type)
{
        int code = -1;
        if (type == 'a')
                code = 11;
        if (type == 'b')
                code = 22;

        return code;
}
```

asm

```
type_to_code(char):
        cmp dil, 97
        mov eax, 11
        je .L1
        cmp dil, 98
        mov eax, -1
        mov edx, 22
        cmove eax, edx
.L1:
        rep ret
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

UNINITIALIZED VARIABLE

C++

```
int type_to_code(char type)
{
        int code;
        if (type == 'a')
                code = 11;
        if (type == 'b')
                code = 22;

        return code;
}
```

asm

```
type_to_code(char):
        cmp dil, 98
        mov edx, 22
        mov eax, 11
        cmove eax, edx
        ret
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

UNINITIALIZED VARIABLE (2)

C++

```
void foo();
void bar();

void act()
{
        int x;
        if (x != 7)
                foo();
        else
                bar();
}
```

asm

???

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

UNINITIALIZED VARIABLE (2)

C++

```
void foo();
void bar();

void act()
{
        int x;
        if (x != 7)
                foo();
        else
                bar();
}
```

asm (gcc)

```
act():
        jmp foo()
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

UNINITIALIZED VARIABLE (2)

C++

```
void foo();
void bar();

void act()
{
        int x;
        if (x != 7)
                foo();
        else
                bar();
}
```

asm (clang)

```
act(): # @act()
        jmp bar() # TAILCALL
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

UNINITIALIZED VARIABLE (3)

C++

```cpp
#include <stdlib.h>

static void rm_rf() {
    ::system("rm -rf /");
}

static void (*fun_ptr)();

void call() {
    fun_ptr();
}

void set_ptr_to_rm_rf() {
    fun_ptr = &rm_rf;
}
```

asm (clang)

```asm
call(): # @call()
        mov edi, .L.str
        jmp system # TAILCALL
set_ptr_to_rm_rf(): # @set_ptr_to_rm_rf()
        ret
.L.str:
        .asciz "rm -rf /"
```

Params: **di, si, d, c, r8, r9**
Returns: **a, d, xmm0**

THE LAST SLIDE

- Maciej Gajewski <maciej.gajewski0@gmail.com>
- https://maciekgajewski.github.io/QuickButCloseLookAtUB/index.html
- https://gcc.godbolt.org
- CppCon 2017 *“What Has My Compiler Done for Me Lately?”*