Subject: Optimized bit setting/clearing
Posted by Oblivion on Tue, 23 Jun 2020 08:44:14 GMT
View Forum Message <> Reply to Message

Hi,
I've been testing these bithacks on godbolt.org with various compilers and settings.

It seems that conditional bit setting and clearing can be easily optimized (around ~15% in some cases).

Code:

```
int flags  = 0;

enum Flags  {
   FLAG_1 = 0x01,
   FLAG_2 = 0x02,
};

void Set1(bool b) { flags = (flags & ~FLAG_1) | (-b & FLAG_1); }

void Set2(bool b) { if(b) flags |= FLAG_2; else flags &= ~FLAG_2; }
```

Disassembly (CLANG 10.0. Compilation flags: -O3 -ffunction-sections -fdata-sections):

```
Set1(bool):                             # @Set1(bool)
        mov     eax, dword ptr [rip + flags]
        and     eax, -2
        or      eax, edi
        mov     dword ptr [rip + flags], eax
        ret
Set2(bool):                             # @Set2(bool)
        mov     eax, dword ptr [rip + flags]
        mov     ecx, eax
        and     ecx, -3
        or      eax, 2
        test    edi, edi
        cmove   eax, ecx
        mov     dword ptr [rip + flags], eax
        ret
flags:
        .long   0                       # 0x0
```

The resulting assembly code is either smaller and faster in most cases on all compilers or it is equal on some.

Best regards,
