The ARM processor (Thumb-2), part 13: Trampolines

devblogs.microsoft.com/oldnewthing/20210616-00

June 16, 2021

Raymond Chen

As we noted last time, the relative branch instructions have a limited reach. In particular, the `bl` instruction, which is used for intra-module direct calls, has a reach of around ±16MB. But what happens if the call target is too far away? Or if the function is a naïvely-imported function?

In the case of a faraway call target, the linker injects a trampoline, called a *veneer* in the ARM documentation.

```
    bl      toofar_trampoline
...

toofar_trampoline:
    mov     r12, #lo(|toofar|+1)
    movt    r12, #hi(|toofar|+1)
    bx      r12             ; jump to r12
```

The *r12* register, known as the *intraprocedure call* register, is a register that the linker is permitted to use for the purpose of generating trampolines and function prologues. From the compiler's point of view, it is super-volatile: Any branch instruction could damage the *r12* register.

In practice, the compiler doesn't use *r12* for anything at all.

In the case of a naïvely-imported function, the actual call target is stored in the import address table, and the linker must generate a trampoline that jumps to the imported function:

```
    bl      imported_trampoline
...

imported_trampoline:
    mov     r12, #lo(iat_imported)
    movt    r12, #hi(iat_imported)
    ldr     pc, [r12]
```

Here, we take advantage of the overly-uniform *pc* register: Loading a value into it acts as a jump instruction. It saves an instruction, because we don't have to load the jump target into a register and then `BX` to it.

Next time, we'll look at a few miscellaneous instructions.

Bonus chatter: I don't know why the linker prefers to use a `MOV` + `MOVT` instruction pair instead of a single *pc*-relative `LDR`. My guess is that it avoids memory latency.

Bonus chatter 2: You might think that trampolines can never be deployed for jumps within a function. However, that's not true: Code motion due to profile-guided optimization can cause rarely-executed code blocks to be relocated to faraway locations in the module. The most likely case is that a relative short jump becomes long and has to be converted to a jump-to-a-jump. In rare cases, the destination could end up more than 16MB away, in which case you would need a full trampoline.

Raymond Chen

Follow