The Intel 80386, part 7: Conditional instructions and control transfer

devblogs.microsoft.com/oldnewthing/20190128-00

January 29, 2019

Raymond Chen

Finally we get to use the conditions that we defined way back in Part 2.

```
SETcc    r/m8        ; d = 1 if cc is true, d = 0 if false
```

The `SETcc` family of instructions sets an 8-bit value to 0 or 1 based on a condition code. For example, `SETE al` sets the *al* register to 1 if the *ZF* flag is set (equal), or to 0 if it is clear.

If the destination of the `SETcc` instruction is an 8-bit register, it is typically preceded by `XOR r32, r32` or followed by `MOVZX r32, r8` to convert the 8-bit value to a 32-bit value.

And of course we have control transfer.

```
Jcc      dest        ; relative branch if cc is true
JMP      dest        ; relative branch unconditionally
```

There are two encodings for each relative branch, a short one if the branch destination is within 128 bytes, and a longer one if it is not. Back in the old days, you had to tell the assembler which kind of relative branch you wanted (short or long), but nowadays the assembler will figure it out for you. This is trickier than it sounds, because when a branch needs to be upgraded from short to long, that causes it to become a bigger instruction, which can have a cascade effect on other branches which also need to be upgraded.

```
CALL     dest        ; subroutine call
                     ; esp = esp - 4
                     ; *esp = address of instruction following CALL
                     ; continue execution at dest
```

The subroutine call instruction pushes the address of the next instruction (the return address) onto the stack and then transfers control to the destination.

Of course, when your subroutine is done, you probably want to return.

```
RET     i16         ; subroutine return
                    ; temp = *esp
                    ; esp = esp + 4 + s
                    ; continue execution at temp
```

The `RET` instruction pops the return address from the stack and resumes execution at the return address. (It is sometimes written as `RETD` to emphasize that it operates on doubleword registers.) The immediate parameter specifies an additional amount to be added to the stack pointer (in bytes) after the return address is popped. If omitted, the value is assumed to be zero.

Whereas on most other processors, subroutine linkage is done in registers, the 80386 requires a memory access on a subroutine call and on a return.

Call and unconditional jump instructions also support indirect transfer.

```
CALL    r/m         ; indirect subroutine call
JMP     r/m         ; indirect unconditional branch
```

The destination of the transfer can be specified by a value in a register or a value in memory.

The last type of control transfer instruction is the software interrupt.

```
INT     i8          ; software interrupt
```

Software interrupts trap to kernel mode. They are a common way to trigger a system call. Interrupt numbers 0 through 31 (`1fh`) are reserved by the processor; software-defined interrupts start at 32 (`20h`).

There are a number of control transfer instructions that you are not going to see in compiler-generated code, so I'm not going to cover them here.

Next time, we'll look at the block instructions.

Raymond Chen

Follow