The Intel 80386, part 11: The TEB

devblogs.microsoft.com/oldnewthing/20190203-00

February 4, 2019

Raymond Chen

The 80386 does not have a lot of registers. But there needs to be a place to record per-thread information. For performance reasons, this should be something available in user mode, to avoid a kernel transition. But where do we keep it? We don't want to burn a precious general-purpose register to hold this value.

Ah, but there are some available registers: The segment registers! There are six segment registers on the 80386:

Segment	Mnemonic
ss	stack segment
cs	code segment
ds	data segment
es	extra segment
fs	
gs	

The previous versions of the processor had only *ss*, *cs*, *ds*, and *es*. The 80386 added two new segment registers, which were named *fs* and *gs* to continue the alphabetic pattern, but the letters *f* and *g* don't have any mnemonic significance.

The first four segments have architectural meaning. The stack segment is used by instructions that access the stack, either implicitly via instructions like `PUSH`, or explicitly by accessing memory with the *esp* or *ebp* registers. The code segment specifies which segment the instruction pointer is reading from. The data segment is used by most memory-access instructions, and the extra segment is used by the block operation instructions.

But the two bonus segment registers aren't architecturally significant. We can use them for anything!

On the 80386, Windows uses the *fs* segment register to access a small block of memory that is associated with each thread, known as the Thread Environment Block, or TEB.

To access memory relative to a specific segment register, you prefix the segment register and a colon to the memory reference.

```
    MOV     eax, fs:[0]        ; eax = memory at offset 0 in segment fs
```

The part of the TEB you're going to see most often is the memory at offset 0, which is the head of a linked list of structured exception handling records threaded through the stack. The 80386 is unusual in that it's the only architecture which executes instructions at runtime to manage exception handling state. All the other architectures use tables generated at compile time, so that there is no runtime penalty.

Windows on the 80386 does not use the *gs* register for anything as far as I can tell.

Next time, I'm going to break my promise and cover the instructions that you will never see.

Raymond Chen

Follow