How did real-mode Windows implement its LRU algorithm without hardware assistance?

devblogs.microsoft.com/oldnewthing/20120810-00

August 10, 2012

Raymond Chen

I noted some time ago that real-mode Windows had to do all its memory management without any hardware assistance. And yet, along the way, they managed to implement an LRU-based discard algorithm. Gabe is really interested in how that was done.

As we saw a few months ago, inter-segment calls were redirected through a little stub which either jumped directly to the target (if it was in memory) or loaded the target (possibly discarding other memory to make room) before jumping to it. And we saw that the executable format had `INT 3Fh` instructions baked into it so that the Entry Table could be loaded directly into memory for execution.

As it happens, Windows didn't take advantage of that feature, because it wanted to do more.

When it came time to load the Entry Table, the loader did a little rewriting, converting each

```
db  flags
INT 3Fh
db  entry_segment
dw  entry_offset
```

sequence into

```
db  flags
sar byte ptr cs:[xxx], 1
INT 3Fh
db  entry_segment
dw  entry_offset
```

where the `xxx` refers to a table of bytes in the Entry Table preallocated for this purpose, initialized to 1's.

What is "this purpose"?

Whenever anybody needed the address of an inter-segment function, instead of return the address of the `int 3Fh`, the kernel returned the address of the `sar` instruction. The `sar` instruction stands for *shift arithmetic right*, For a byte value, this means to shift the bits right one place, but keep the high-order bit the same.

a	b	c	d	e	f	g	h
↓							
a	a	b	c	d	e	f	g

Okay, so what was the effect of sticking this little `sar` instruction at the start of every inter-segment call? Since the values in the table were initialized to 1, a right arithmetic shift changed the 1 to 0. Therefore, each time an inter-segment call was performed, the corresponding byte in the table was set to zero.

Hooray, a software-implemented Accessed bit!

Every 250 milliseconds, Windows scanned and reset the Access bits, using the data to maintain an LRU-list of all the segments in the system. That way, when it was time to discard some memory, it could discard the least recently used ones first.

Today, a timer that runs continuously at 250ms would incur the wrath of the power management team. But back in the days of real-mode Windows, there was no power management. Like Chuck Norris, PCs ran at only one power level: Awesome.

I continue to be amazed at how much Windows 1.0 accomplished with so little.

[Raymond is currently away; this message was pre-recorded.]

Raymond Chen

Follow