CHAPTER 13
USER-TIMER EVENTS AND INTERRUPTS

This chapter describes an architectural feature called **user-timer events**.

The feature defines a new 64-bit value called the **user deadline**. Software may read and write the user deadline. When the user deadline is not zero, a user-timer event becomes **pending** when the logical processor's timestamp counter (TSC) is greater than or equal to the user deadline.

A pending user-timer event is processed by the processor when CPL = 3 and certain other conditions apply. When processed, the event results in a user interrupt with the **user-timer vector**. (Software may read and write the user-timer vector). Specifically, the processor sets the bit in the UIRR (user interrupt request register) corresponding to the user timer vector. The processing also clears the user deadline, ensuring that there will be no subsequent user-timer events until software writes the user deadline again.

Section 13.1 discusses the enabling and enumeration of the new feature. Section 13.2 presents details of the user deadline, and Section 13.3 explains how it (together with the user-timer vector) is represented in a new MSR. Section 13.4 explains when and how a logical processor processes a pending user-timer event. Section 13.5 presents VMX support for virtualizing the new feature.

13.1 ENABLING AND ENUMERATION

Processor support for user-timer events is enumerated by CPUID.(EAX=07H, ECX=1H):EDX.UTMR[bit 13]. If this feature flag is set, the processor supports user-timer events, and software can access the IA32_UINTR_TIMER MSR (see Section 13.3).

13.2 USER DEADLINE

A logical processor that supports user-timer events supports a 64-bit value called the **user deadline**. If the user deadline is non-zero, the logical processor pends a user-timer event when the timestamp counter (TSC) reaches or exceeds the user deadline.

Software can write the user deadline using instructions specified later in this chapter (see Section 13.3). The processor enforces the following:

- Writing zero to the user deadline disables user-timer events and cancels any that were pending. As a result, no user-timer event is pending after zero is written to the user deadline.
- If software writes the user deadline with a non-zero value that is less than the TSC, a user-timer event will be pending upon completion of that write.
- If software writes the user deadline with a non-zero value that is greater than that of the TSC, no user-timer event will be pending after the write until the TSC reaches the new user deadline.
- A logical processor processes a pending user-timer event under certain conditions; see Section 13.4. The logical processor clears the user deadline after pending a user-timer event.

Races may occur if software writes a new user deadline when the value of the TSC is close to that of the original user deadline. In such a case, either of the following may occur:

- The TSC may reach the original deadline before the write to the deadline, causing a user-timer event to be pended. Either of the following may occur:
  - — If the user-timer event is processed before the write to the deadline, the logical processor will clear the deadline before the write. The write to the deadline may cause a second user-timer event to occur later.
  - — If the write to the deadline occurs before the user-timer event is processed, the original user-timer event is canceled, and any subsequent user-timer event will be based on the new value of the deadline.

  When writing to the deadline, it may not be possible for software to control with certainty which of these two situations occurs.

- The write to the deadline may occur before TSC reaches the original deadline. In this case, no user-timer event will occur based on the original deadline. Any subsequent user-timer event will be based on the new value of the deadline.

Software writes to the user deadline using a new MSR described in Section 13.3.

13.3 USER TIMER: ARCHITECTURAL STATE

The user-timer architecture defines a new MSR, IA32_UINTR_TIMER. This MSR can be accessed using MSR index 1B00H.

The IA32_UINTR_TIMER MSR has the following format:

- Bits 5:0 are the user-timer vector. Processing of a user-timer event results in the pending of a user interrupt with this vector (see Section 13.4).
- Bits 63:6 are the upper 56 bits of the user deadline (see Section 13.2).

Note that no bits are reserved in the MSR and that writes to the MSR will not fault due to the value of the instruction's source operand. The IA32_UINTR_TIMER MSR can be accessed via the following instructions: RDMSR, RDMSRLIST, URDMSR, UWRMSR, WRMSR, WRMSRLIST, and WRMSRNS.

If the IA32_UINTR_TIMER MSR is written with value X, the user-timer vector gets value X & 3FH; the user deadline gets value X & ~3FH.

If the user-timer vector is V ($0 \leq V \leq 63$) and the user deadline is D, a read from the IA32_UINTR_TIMER MSR return value (D & ~3FH) | V.

13.4 PENDING AND PROCESSING OF USER-TIMER EVENTS

There is a user-timer event pending whenever the user deadline (Section 13.2) is non-zero and is less than or equal to the value of the timestamp counter (TSC).

If CR4.UINTR = 1, a logical processor processes a pending user-timer event at an instruction boundary at which the following conditions all hold[1]: (1) IA32_EFER.LMA = CS.L = 1 (the logical processor is in 64-bit mode); (2) CPL = 3; (3) UIF = 1; and (4) the logical processor is not in the shutdown state or in the wait-for-SIPI state.[2]

When the conditions just identified hold, the logical processor processes a user-timer event. User-timer events have priority just above that of user-interrupt delivery. If the logical processor was in a state entered using the TPAUSE and UMWAIT instructions, it first wakes up from that state and becomes active. If the logical processor was in enclave mode, it exits the enclave (via AEX) before processing the user-timer event.

The following pseudocode details the processing of a user-timer event:

```
UIRR[UserTimerVector] := 1;
recognize a pending user interrupt;// may be delivered immediately after processing
IA32_UINTR_TIMER := 0;// clears the deadline and the vector
```

Processing of a user-timer event aborts transactional execution and results in a transition to a non-transactional execution. The transactional abort loads EAX as it would have had it been due to an ordinary interrupt.

Processing of a user-timer event cannot cause a fault or a VM exit.

13.5 VMX SUPPORT

The VMX architecture supports virtualization of the instruction set and its system architecture. Certain extensions are needed to support virtualization of user-timer events. This section describes these extensions.

1. Execution of MOV SS, POP SS, or STI may block the processing of user-timer events for one instruction.
2. A logical processor processes a user-timer event only if CPL = 3. Since the HLT and MWAIT instructions can be executed only if CPL = 0, a user-timer event is never processed when a logical processor is an activity state that was entered using one of those instructions.

13.5.1 VMCS Changes

One new 64-bit VM-execution control field is defined called the **virtual user-timer control**. It can be accessed with the encoding pair 2050H/2051H. See Section 13.5.2 for its use in VMX non-root operation. This field exists only on processors that enumerate CPUID.(EAX=07H, ECX=1H):EDX[13] as 1 (see Section 13.1).

13.5.2 Changes to VMX Non-Root Operation

This section describes changes to VMX non-root operation for user-timer events.

13.5.2.1 Treatment of Accesses to the IA32_UINTR_TIMER MSR

As noted in Section 13.3, software can read and write the IA32_UINTR_TIMER MSR using certain instructions. The operation of those instructions is changed when they are executed in VMX non-root operation:

- Any read from the IA32_UINTR_TIMER MSR (e.g., by RDMSR) returns the value of the virtual user-timer control.
- Any write to the IA32_UINTR_TIMER MSR (e.g., by WRMSR) is treated as follows:
  - The source operand is written to the virtual user-timer control (updating the VMCS).
  - Bits 5:0 of the source operand are written to the user-timer vector.
  - If bits 63:6 of the source operand are zero, the user deadline (the value that actually controls when hardware generates a user time event) is cleared to 0. Section 13.2 identifies the consequences of this clearing.
  - If bits 63:6 of the source operand are not all zero, the user deadline is computed as follows. The source operand (with the low 6 bits cleared) is interpreted as a virtual user deadline. The processor converts that value to the actual user deadline based on the current configuration of TSC offsetting and TSC scaling.[1]
  - Following such a write, the value of the IA32_UINTR_TIMER MSR (e.g., as would be observed following a subsequent VM exit) is such that bits 63:6 contain the actual user deadline (not the virtual user deadline), while bits 5:0 contain the user-timer vector.

13.5.2.2 Treatment of User-Timer Events

The processor's treatment of user-timer events is described in Section 13.4. These events occur in VMX non-root operation under the same conditions described in that section.

The processing of user-timer events differs in VMX non-root operation only in that, in addition to clearing the IA32_UINTR_TIMER MSR, the processing also clears the virtual user-timer control (updating the VMCS).

13.5.3 Changes to VM Entries

A VM entry results in a pending user-timer event if and only if the VM entry completes with the user deadline non-zero and less than or equal to the (non-virtualized) TSC. The processor will process such an event only if indicated by the conditions identified in Section 13.4.

1. This conversion may not be meaningful if "RDTSC exiting" is 1. Software setting "RDTSC exiting" to 1 should ensure that any write to the IA32_UINTR_TIMER MSR causes a VM exit.