CHAPTER 17
MONITORLESS MWAIT

This chapter documents the monitorless MWAIT feature.

Prior this feature, execution of the MWAIT instruction causes a logical processor to suspend execution and enter an implementation-dependent optimized state only if the MONITOR instruction was executed previously, specifying an address range to monitor, and there have been no stores to that address range since MONITOR executed. The logical processor leaves the optimized state and resumes execution when there is a write to the monitored address range.

This existing functionality supports software that seeks to suspend execution until an event associated with a write to the monitored address range. For example, that range may contain the head pointer of a work queue that is written when there is work for the suspended logical processor.

It is possible that software may wish to suspend execution with no requirement to resume execution in response to a memory write. Such software is not well served by the existing MWAIT instruction since it must incur the overhead of monitoring some (irrelevant) address range and may resume execution earlier than intended following a memory write.

Monitorless MWAIT enhances the MWAIT instruction by allowing suspension of execution without monitoring an address range.

The feature is defined with an enumeration independent of that of existing MONITOR/MWAIT. That allows a VMM to virtualize monitorless MWAIT without having to virtualize the address-range monitoring of the existing feature.

17.1 USING MONITORLESS MWAIT

The MWAIT instruction uses the value of ECX to determine which MWAIT extensions are requested by software.

If ECX[2] is 1, an execution of MWAIT is **monitorless**. The logical processor will suspend execution and enter an implementation-dependent optimized state regardless of any previous execution of the MONITOR instruction. There is no address range to which a write is guaranteed to cause the logical processor to leave the optimized state and resume execution.

Software may set ECX[2] while also setting other bits in ECX that control other aspects of MWAIT execution.

Execution of MWAIT with ECX[2] = 1 is allowed only if the processor enumerates support for monitorless MWAIT (see Section 17.2); if it not, such an execution causes a general-protection fault (#GP(0)).

See Section 17.5 for details on the operation of the MWAIT instruction.

17.2 ENUMERATION

Existing processors indicate support for the MONITOR and MWAIT instructions by enumerating CPUID.01H:ECX.MONITOR[bit 3] as 1. This enumeration also implies support for CPUID leaf 05H, the MONTOR/MWAIT leaf. CPUID leaf 05H enumerates details of the operation of the MONITOR instruction (e.g., the size of the monitored address range) and the capabilities of the MWAIT instruction (e.g., the extensions that can be specified in ECX).

CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3] enumerates support for monitorless use of MWAIT. If this bit is enumerated as 1, software can execute MWAIT with ECX[2] = 1 (see Section 17.1).

To allow virtualization of monitorless MWAIT (without the monitored form; see Section 17.4), CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] indicates support for the MWAIT instruction and for CPUID leaf 05H. The following items detail the implications of the value enumerated for this bit:

- If CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] is enumerated as 0, MWAIT is still supported if CPUID.01H:ECX.MONITOR[bit 3] is enumerated as 1. Monitorless MWAIT is supported only if CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3] is enumerated as 1.

- If CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] is enumerated as 1, MONITOR is supported as long as CPUID.01H:ECX.MONITOR[bit 3] is enumerated as 1. Monitorless MWAIT is supported only if CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3] is enumerated as 1.
- If CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] is enumerated as 1 and CPUID.01H:ECX.MONITOR[bit 3] is enumerated as 0, MWAIT is supported but MONITOR is not. Moreover, only the monitorless form of MWAIT is supported; execution of MWAIT with ECX[2] = 0 causes #GP(0).

Software seeking to use MONITOR and MWAIT together should continue to use CPUID.01H:ECX.MONITOR[bit 3] and CPUID leaf 05H; software seeking to use monitorless MWAIT should consult CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] and CPUID leaf 05H.

NOTE

Cores in hybrid CPU support MWAIT consistently. A core will support monitorless MWAIT only if all cores in the hybrid CPU do so.

17.3 ENABLING

The MONITOR and MWAIT instructions are available only when IA32_MISC_ENABLE.ENABLE_MONITOR_FSM[bit 18] = 1.

If IA32_MISC_ENABLE.ENABLE_MONITOR_FSM[bit 18] = 0, execution of MONITOR or MWAIT causes an invalid-opcode exception (#UD). In addition, CPUID.01H:ECX.MONITOR[bit 3] and CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] are each enumerated as 0, and CPUID leaf 05H is not supported.

17.4 VIRTUALIZATION

A virtual-machine monitor (VMM) may want to present the abstraction of a virtual machine that supports monitorless MWAIT but not the existing monitoring of address ranges.

Such a VMM can virtualize CPUID to enumerate CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] as 1, CPUID.01H:ECX.MONITOR[bit 3] as 0, and CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3] as 1. The VMM can intercept executions of MONITOR and deliver a #UD to the guest; it can intercept executions of MWAIT and either (1) deliver a #GP(0) to the guest if ECX[2] = 0; or (2) virtualize monitorless MWAIT if ECX[2] = 1.

17.5 MWAIT INSTRUCTION DETAILS

All changes to existing MWAIT operation are highlighted in violet with change bars.

MWAIT—Monitor Wait

Opcode	Instruction	Op/ En	64-Bit Mode	Compat/ Leg Mode	Description
0F 01 C9	MWAIT	ZO	Valid	Valid	A hint that allows the processor to stop instruction execution and enter an implementation-dependent optimized state until occurrence of a class of events.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
ZO	N/A	N/A	N/A	N/A

Description

MWAIT instruction provides hints to allow the processor to enter an implementation-dependent optimized state. There are two principal targeted usages: address-range monitor and advanced power management.

CPUID.01H:ECX.MONITOR[bit 3] and CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] both indicate the availability of MWAIT in the processor; the instruction is supported if either is enumerated as 1. When set, MWAIT may be executed only at privilege level 0 (use at any other privilege level results in an invalid-opcode exception). CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3] indicates the availability of MWAIT that does not use a monitored address range (with ECX[2] set; "monitorless MWAIT") but does not indicate availability of MONITOR or of non-monitorless MWAIT (MWAIT with ECX[2] cleared).

The operating system or system BIOS may disable this instruction by using the IA32_MISC_ENABLE MSR; disabling MWAIT clears the CPUID feature flag and causes execution to generate an invalid-opcode exception.

This instruction's operation is the same in non-64-bit modes and 64-bit mode.

ECX specifies optional extensions for the MWAIT instruction. EAX may contain hints such as the preferred optimized state the processor should enter. The first processors to implement MWAIT supported only the zero value for EAX and ECX. Later processors allowed setting ECX[0] to enable masked interrupts as break events for MWAIT or setting ECX[2] to enable monitorless MWAIT (see below). Software can use the CPUID instruction to determine the extensions and hints supported by the processor.

MWAIT for Address Range Monitoring

For address-range monitoring, the MWAIT instruction operates with the MONITOR instruction. The two instructions allow the definition of an address at which to wait (MONITOR) and a implementation-dependent-optimized operation to commence at the wait address (MWAIT). The execution of MWAIT is a hint to the processor that it can enter an implementation-dependent-optimized state while waiting for an event or a store operation to the address range armed by MONITOR.

The following cause the processor to exit the implementation-dependent-optimized state: a store to the address range armed by the MONITOR instruction, an NMI or SMI, a debug exception, a machine check exception, the BINIT# signal, the INIT# signal, and the RESET# signal. Other implementation-dependent events may also cause the processor to exit the implementation-dependent-optimized state.

In addition, an external interrupt causes the processor to exit the implementation-dependent-optimized state either (1) if the interrupt would be delivered to software (e.g., as it would be if HLT had been executed instead of MWAIT); or (2) if ECX[0] = 1. Software can execute MWAIT with ECX[0] = 1 only if CPUID.05H:ECX[bit 1] = 1. (Implementation-specific conditions may result in an interrupt causing the processor to exit the implementation-dependent-optimized state even if interrupts are masked and ECX[0] = 0.)

Following exit from the implementation-dependent-optimized state, control passes to the instruction following the MWAIT instruction. A pending interrupt that is not masked (including an NMI or an SMI) may be delivered before execution of that instruction. Unlike the HLT instruction, the MWAIT instruction does not support a restart at the MWAIT instruction following the handling of an SMI.

If the preceding MONITOR instruction did not successfully arm an address range or if the MONITOR instruction has not been executed prior to executing MWAIT, then the processor will not enter the implementation-dependent-optimized state. Execution will resume at the instruction following the MWAIT.

MWAIT for Power Management

MWAIT accepts a hint and optional extension to the processor that it can enter a specified target C state while waiting for an event or a store operation to the address range armed by MONITOR. Support for MWAIT extensions for power management is indicated by CPUID.05H:ECX[bit 0] reporting 1.

EAX and ECX are used to communicate the additional information to the MWAIT instruction, such as the kind of optimized state the processor should enter. ECX specifies optional extensions for the MWAIT instruction. EAX may contain hints such as the preferred optimized state the processor should enter. Implementation-specific conditions may cause a processor to ignore the hint and enter a different optimized state. Future processor implementations may implement several optimized "waiting" states and will select among those states based on the hint argument.

Table 17-1 describes the meaning of ECX and EAX registers for MWAIT extensions.

Table 17-1. MWAIT Extension Register (ECX)

Bits	Description
0	Treat interrupts as break events even if masked (e.g., even if EFLAGS.IF=0). May be set only if CPUID.05H:ECX[bit 1] = 1.
1	Reserved.
2	Allows MWAIT to enter and stay in an implementation-dependent-optimized state regardless of whether an address range armed by MONITOR exists or has been stored to. May be set only if CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3] = 1. Must be set if CPUID.01H:ECX.MONITOR[bit 3] = 0.
31:3	Reserved.

Table 17-2. MWAIT Hints Register (EAX)

Bits	Description
3:0	Sub C-state within a C-state, indicated by bits [7:4]
7:4	Target C-state* Value of 0 means C1; 1 means C2, etc. Value of 01111B means C0. Note: Target C states for MWAIT extensions are processor-specific C-states, not ACPI C-states
31:8	Reserved.

Note that if MWAIT is used to enter any of the C-states that are numerically higher than C1, a store to the address range armed by the MONITOR instruction will cause the processor to exit MWAIT only if the store was originated by other processor agents. A store from non-processor agent might not cause the processor to exit MWAIT in such cases.

If MWAIT is used with ECX[2] set, it will ignore any preceding MONITOR instruction and will ignore stores to any address range that may have been monitored. Support for this is enumerated by CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3].

For additional details of MWAIT extensions, see Chapter 15, "Power and Thermal Management," of Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 3B.

Operation

```
(* MWAIT takes the argument in EAX as a hint extension and is architected to take the argument in ECX as an instruction extension
MWAIT EAX, ECX *)
{
WHILE ( (“Monitor Hardware is in armed state”)) {
    implementation_dependent_optimized_state(EAX, ECX); }
Set the state of Monitor Hardware as triggered;
}
```

Intel C/C++ Compiler Intrinsic Equivalent

MWAIT void _mm_mwait(unsigned extensions, unsigned hints)

Example Using a Monitored Address

MONITOR/MWAIT instruction pair must be coded in the same loop because execution of the MWAIT instruction will trigger the monitor hardware. It is not a proper usage to execute MONITOR once and then execute MWAIT in a loop. Setting up MONITOR without executing MWAIT has no adverse effects.

Typically the MONITOR/MWAIT pair is used in a sequence, such as:

```
EAX = Logical Address(Trigger)
ECX = 0 (*Hints *)
EDX = 0 (* Hints *)

IF ( !trigger_store_happened) {
    MONITOR EAX, ECX, EDX
    IF ( !trigger_store_happened ) {
        MWAIT EAX, ECX
    }
}
```

The above code sequence makes sure that a triggering store does not happen between the first check of the trigger and the execution of the monitor instruction. Without the second check that triggering store would go unnoticed. Typical usage of MONITOR and MWAIT would have the above code sequence within a loop.

Numeric Exceptions

None.

Protected Mode Exceptions

#GP(0)	If ECX[31:3] ≠ 0 or ECX[1] = 1. If ECX[0] = 1 and CPUID.05H:ECX[bit 1] = 0. If ECX[2] = 1 and CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3] = 0. If ECX[2] = 0 and CPUID.01H:ECX.MONITOR[bit 3] = 0.
#UD	If CPUID.01H:ECX.MONITOR[bit 3] = 0 and CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] = 0. If current privilege level is not 0.

Real Address Mode Exceptions

#GP	If ECX[31:3] ≠ 0 or ECX[1] = 1. If ECX[0] = 1 and CPUID.05H:ECX[bit 1] = 0. If ECX[2] = 1 and CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3] = 0. If ECX[2] = 0 and CPUID.01H:ECX.MONITOR[bit 3] = 0.
#UD	If CPUID.01H:ECX.MONITOR[bit 3] = 0 and CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] = 0.

Virtual 8086 Mode Exceptions

#UD	The MWAIT instruction is not recognized in virtual-8086 mode (even if CPUID.01H:ECX.MONITOR[bit 3] = 1 or CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] = 1).

Compatibility Mode Exceptions

Same exceptions as in protected mode.

64-Bit Mode Exceptions

#GP(0)	If RCX[63:3] ≠ 0 or RCX[1] = 1. If RCX[0] = 1 and CPUID.05H:ECX[bit 1] = 0. If RCX[2] = 1 and CPUID.05H:ECX.MONITORLESS_MWAIT[bit 3] = 0. If RCX[2] = 0 and CPUID.01H:ECX.MONITOR[bit 3] = 0.
#UD	If the current privilege level is not 0. If CPUID.01H:ECX.MONITOR[bit 3] = 0 and CPUID.(EAX=07H, ECX=01H):EDX.MWAIT[bit 23] = 0.