XBEGIN — Transactional Begin

Opcode/Instruction	Op/ En	64/32bit Mode Support	CPUID Feature Flag	Description
C7 F8 XBEGIN rel16	A	V/V	RTM	Specifies the start of an RTM region. Provides a 16-bit relative offset to compute the address of the fallback instruction address at which execution resumes following an RTM abort.
C7 F8 XBEGIN rel32	A	V/V	RTM	Specifies the start of an RTM region. Provides a 32-bit relative offset to compute the address of the fallback instruction address at which execution resumes following an RTM abort.

Instruction Operand Encoding

Op/En	Operand 1	Operand2	Operand3	Operand4
A	Offset	NA	NA	NA

Description

The XBEGIN instruction specifies the start of an RTM code region. If the logical processor was not already in transactional execution, then the XBEGIN instruction causes the logical processor to transition into transactional execution. The XBEGIN instruction that transitions the logical processor into transactional execution is referred to as the outermost XBEGIN instruction. The instruction also specifies a relative offset to compute the address of the fallback code path following a transactional abort.

On an RTM abort, the logical processor discards all architectural register and memory updates performed during the RTM execution and restores architectural state to that corresponding to the outermost XBEGIN instruction. The fallback address following an abort is computed from the outermost XBEGIN instruction.

Operation

XBEGIN

```
IF RTM_NEST_COUNT < MAX_RTM_NEST_COUNT
    THEN
        RTM_NEST_COUNT++
        IF RTM_NEST_COUNT = 1 THEN
            IF 64-bit Mode
                THEN
                        fallbackRIP ← RIP + SignExtend64(IMM)
                                (* RIP is instruction following XBEGIN instruction *)
                    ELSE
                        fallbackEIP ← EIP + SignExtend32(IMM)
                                (* EIP is instruction following XBEGIN instruction *)
            FI;

            IF (64-bit mode)
                THEN IF (fallbackRIP is not canonical)
                    THEN #GP(0)
                FI;
                ELSE IF (fallbackEIP outside code segment limit)
                    THEN #GP(0)
                FI;
            FI;

            RTM_ACTIVE ← 1
            Enter RTM Execution (* record register state, start tracking memory state*)
        FI; (* RTM_NEST_COUNT = 1 *)
```

```
    ELSE (* RTM_NEST_COUNT = MAX_RTM_NEST_COUNT *)
        GOTO RTM_ABORT_PROCESSING
FI;

(* For any RTM abort condition encountered during RTM execution *)
RTM_ABORT_PROCESSING:
    Restore architectural register state
    Discard memory updates performed in transaction
    Update EAX with status
    RTM_NEST_COUNT ← 0
    RTM_ACTIVE ← 0
    IF 64-bit mode
        THEN
            RIP ← fallbackRIP
        ELSE
            EIP ← fallbackEIP
    FI;
END
```

Flags Affected

None

Intel C/C++ Compiler Intrinsic Equivalent

XBEGIN: unsigned int _xbegin(void);

SIMD Floating-Point Exceptions

None

Protected Mode Exceptions

#UD	CPUID.(EAX=7, ECX=0):EBX.RTM[bit 11]=0.
	If LOCK prefix is used.
#GP(0)	If the fallback address is outside the CS segment.

Real-Address Mode Exceptions

#GP(0)	If the fallback address is outside the address space 0000H and FFFFH.
#UD	CPUID.(EAX=7, ECX=0):EBX.RTM[bit 11]=0.
	If LOCK prefix is used.

Virtual-8086 Mode Exceptions

#GP(0)	If the fallback address is outside the address space 0000H and FFFFH.
#UD	CPUID.(EAX=7, ECX=0):EBX.RTM[bit 11]=0.
	If LOCK prefix is used.

Compatibility Mode Exceptions

Same exceptions as in protected mode.

64-bit Mode Exceptions

#UD	CPUID.(EAX=7, ECX=0):EBX.RTM[bit 11] = 0. If LOCK prefix is used.
#GP(0)	If the fallback address is non-canonical.