AMD Athlon™ Processor Model 10 Revision Guide

Publication # **27532** Rev: **C**
Issue Date: **October 2003**

Revision History

Date	Rev	Description
October 2003	C	Revised erratum #27.
June 2003	B	Initial public release.

AMD Athlon™ Processor Model 10 Revision Guide

The purpose of the *AMD Athlon™ XP Processor Model 10 Revision Guide* is to communicate updated product information on the AMD Athlon™ XP processor model 10 to designers of computer systems and software developers. This revision guide applies to the AMD Athlon XP processor model 10, mobile AMD Athlon XP processor model 10, AMD Athlon XP processor model 10 low-power desktop, and AMD Athlon MP processor model 10. This guide consists of three sections:

- **Product Errata:** This section, which starts on page 5, provides a detailed description of product errata, including potential effects on system operation and suggested workarounds. An erratum is defined as a deviation from the product's specification. A product errata may cause the behavior of the AMD Athlon processor model 10 to deviate from the published specifications.
- **Revision Determination:** This section, which starts on page 14, shows the AMD Athlon processor model 10 identification numbers returned by the CPUID instruction for each revision of the processor.
- **Technical and Documentation Support:** This section, which starts on page 15, provides a listing of available technical support resources. It also lists corrections, modifications, and clarifications to listed documents. Please refer to the data sheets listed in this section for product marking information.

Revision Guide Policy

Occasionally AMD identifies deviations from or changes to the specification of the AMD Athlon processor model 10. These changes are documented in the *AMD Athlon™ XP Processor Model 10 Revision Guide* as errata. Descriptions are written to assist system and software designers in using the AMD Athlon processor model 10 and corrections to AMD's documentation on the AMD Athlon processor model 10 are included. This release documents currently characterized product errata.

1 Product Errata

This section documents AMD Athlon XP processor model 10 product errata. The errata are divided into categories to assist referencing particular errata. A unique tracking number for each erratum has been assigned within this document for user convenience in tracking the errata within specific revision levels. Table 1 cross-references the revisions of the processor to each erratum. An "X" indicates that the erratum applies to the stepping. The absence of an "X" indicates that the erratum does not apply to the stepping. Table 2 on page 6 cross-references erratum to each processor segment. An "X" indicates that the erratum applies to the processor segment.

Note: *There can be missing errata numbers. Errata that have been resolved from early revisions of the processor have been deleted, and errata that have been reconsidered may have been deleted or renumbered.*

Table 1. Cross-Reference of Product Revision to Errata

Errata Numbers and Description	Revision Numbers
	A2
17 Deadlock May Occur in a Two-Processor System in the Presence of Probe to Memory- Mapped I/O	X
18 Processor Performance Counters Do Not Count Some x86 Instructions	X
20 A Speculative SMC Store Followed by an Actual SMC Store May Cause One-Time Stale Execution	X
22 Processor Does Not Support Reliable Microcode Patch Mechanism	X
26 Single Step Across I/O SMI Skips One Debug Trap	X
27 Software Prefetches May Report A Page Fault	X

Table 2. Cross-Reference of Erratum to Processor Segments

Errata Number	Workstation/Server[1]	Desktop[2]	Low-Power Desktop[3]	Mobile[4]
17	X			
18	X	X	X	X
20	X	X	X	X
22	X	X	X	X
26	X	X	X	X
27	X	X	X	X

Notes:

1. The workstation/server segment currently includes the AMD Athlon™ MP processor.

2. The desktop segment currently includes the AMD Athlon XP processor.

3. The low-power desktop segment currently includes the AMD Athlon XP processor.

4. The mobile segment currently includes the mobile AMD Athlon XP processor.

17 Deadlock May Occur in a Two-Processor System in the Presence of Probe to Memory-Mapped I/O

Products Affected. A2

Normal Specified Operation. Processor should not hang.

Non-conformance. In a multiprocessor system, if one processor (A) is continuously writing to a cacheable memory-mapped I/O block while the other processor (B) is trying to read the same cacheable I/O block, and at the same time both processors are also trying to write a different memory-based cache block, then processor B may hang. Should this occur and processor A fields an interrupt, the deadlock is resolved.

Potential Effect on System. System will hang or exhibit performance degradation.

Suggested Workaround. The current processor design assumes that memory-mapped I/O is incoherent and does not handle all deadlock cases. System logic should not generate probes for memory-mapped I/O addresses.

Resolution Status. No fix planned.

18 Processor Performance Counters Do Not Count Some x86 Instructions

Products Affected. A2

Normal Specified Operation. The processor should count all x86 instructions when programmed to do so.

Non-conformance. There are two types of uncounted instructions. One set of instructions is always uncounted. Another set of instructions are uncounted only if a certain data dependency exists.

Instructions never counted are: RDMSR, WRMSR, FSTENV, FSAVE, FLDENV, FPTAN, FYL2XP1, FCLEX, LLDT, LTR, MOV CRx, LGDT, LIDT, INVLPG, INVD, WBINVD, MOV DRx, CPUID, and SFENCE.

Instructions that are uncounted only when certain data dependencies exist are:

- LAR, LSL, VERR, VERW if they clear the Zero Flag
- FXSAVE, FXRSTOR if FERR is changed
- FPU instructions with exceptional data conditions
- IO instructions that detect an interrupt
- POPF with the trap flag =1
- POPFD and PUSHFD with IOPL not equal 3 and Virtual Mode enabled
- POPFD when Alignment Check is being enabled
- MOV SS with the trap flag =1
- Segment Loads that generate accessed bit exceptions
- STI with the trap flag or the interrupt flag already a 1
- CLTS with the CR0.TS flag =1
- LMSW that changes any bit

Potential Effect on System. Performance counter may under count the actual number of x86 instructions.

Suggested Workaround. Versions of the AMD Athlon™ processor not affected by this erratum may be used to gather instruction counts.

Resolution Status. No fix planned.

20 A Speculative SMC Store Followed by an Actual SMC Store May Cause One-Time Stale Execution

Products Affected. A2

Normal Specified Operation. Self-modifying code sequences should be correctly detected and handled in a manner consistent with canonical results; stale code should not be executed.

Non-conformance. The following scenario can result in a one-time execution of stale instructions:

1. A speculative store instruction initiates a request (R) to modify a 64-byte cache line with address A, which currently resides within the L1 instruction cache.
2. The speculative store instruction is ultimately not executed because of a branch misprediction. However, the store R is still in flight attempting to bring the line into the data cache in the modified state.
3. The instruction cache, which fetches instructions 16 bytes at a time, is redirected by the branch into the cache line with address A and fetches a portion of the line into the instruction buffer.
4. R then invalidates the instruction cache line with address A and brings the line into the L1 data cache, marking it as modified. However, the instruction buffer, which also contains some bytes from address A, is not invalidated.
5. The instruction fetch mechanism attempts to read the next 16-byte chunk of code and must issue a request to bring the 64-byte line back into the instruction cache.
6. This instruction cache request for address A hits on the modified line now in the L1 cache, and evicts it from the data cache to the L2.
7. A second store instruction (S) from the instruction buffer is issued into the execution units. S is a self-modifying code reference to another instruction that currently exists in the 64-byte cache block at address A and is also in the instruction buffer.
8. The execution of S detects that an instruction request to fetch address A is in flight. However, the store request is given priority. Since it now hits in the L2 and the L2 state is modified, it assumes that the line cannot be in the instruction cache or the instruction buffer.

Potential Effect on System. The processor will execute stale code instructions.

Suggested Workaround. None. This failure has only been observed in internally generated synthetic code.

Resolution Status. No fix planned.

22 Processor Does Not Support Reliable Microcode Patch Mechanism

Products Affected. A2

Normal Specified Operation. The processor should function properly after a microcode patch is loaded.

Non-conformance. The processor has the patch RAM BIST function disabled. Since BIST is not run on the patch RAM, reliable operation of the patch RAM cannot be guaranteed. Therefore it should not be used.

Potential Effect on System. When a microcode patch is loaded, the system may not behave properly.

Suggested Workaround. Do not load a microcode patch.

Resolution Status. No fix planned.

26 Single Step Across I/O SMI Skips One Debug Trap

Products Affected. A2

Normal Specified Operation. When single stepping (with EFLAGS.TF) across an IN or OUT instruction that detects an SMI, the processor correctly defers taking the debug trap and instead enters SMM. Upon RSM (without I/O restart), the processor should immediately enter the debug trap handler.

Non-conformance. Under this scenario, the processor does not enter the debug trap handler but instead returns to the instruction following the I/O instruction.

Potential Effect on System. When using the single step debug mode, following an I/O operation that detects an SMI, one instruction may appear to be skipped.

Suggested Workaround. None required as this is a debug limitation only. If a workaround is desired, modify the SMM handler to detect this case and enter the debug handler directly.

Resolution Status. No fix planned.

27 Software Prefetches May Report A Page Fault

Products Affected. A2

Normal Specified Operation. Software prefetches should not report page faults if they encounter them.

Non-conformance. Software prefetch instructions are defined to ignore page faults. Under highly specific and detailed internal circumstances, a prefetch instruction may report a page fault if both of the following conditions are true:

- The target address of the prefetch would cause a page fault if the address was accessed by an actual memory load or store instruction under the current privilege mode;
- The prefetch instruction is followed in execution-order by an actual or speculative byte-sized memory access of the same modify-intent to the same address.

PREFETCH and PREFETCHNTA/0/1/2 have the same modify-intent as a memory load access. PREFETCHW has the same modify-intent as a memory store access.

The page fault exception error code bits for the faulting prefetch will be identical to that for a byte-sized memory access of the same-modify intent to the same address.

Note that some misaligned accesses can be broken up by the processor into multiple accesses where at least one of the accesses is a byte-sized access.

If the target address of the subsequent memory access of the same modify-intent is aligned and not byte-sized, this errata does not occur and no workaround is needed.

Potential Effect on System. An unexpected page fault may occur infrequently on a prefetch instruction.

Suggested Workaround. Two workarounds are described for this erratum.

Kernel Workaround

The Operating System kernel can work around the erratum by allowing the page fault handler to satisfy the page fault to an "accessible" page regardless of whether the fault was due to a load, store, or prefetch operation. If the faulting instruction is permitted access to the page, return to it as usual. (An "accessible" page is one for which memory accesses are allowed under the current privilege mode once the page is resident in memory).

If the faulting instruction is trying to access an "inaccessible" page, scan the instruction stream bytes at the faulting Instruction Pointer to determine if the instruction is a prefetch. (An "inaccessible" page is one for which memory accesses are not allowed under the current privilege mode.) If the faulting instruction is a prefetch instruction, simply return back to it; the internal hardware conditions that caused the prefetch to fault should be removed and operation should continue normally. If it is not a prefetch instruction, generate the appropriate memory access control violation as appropriate. The performance impact of doing the scan is small because the actual errata is infrequent and does not produce an excessive number of page faults that affect system performance.

General Workaround

If the page-fault handler for a kernel can be patched as described in the preceding kernel workaround, no further action by software is required. The following general workarounds should only be considered for kernels where the page-fault handler can not be patched and a prefetch instruction could end up targeting an address in an "inaccessible" page.

Because the actual errata is infrequent, it does not produce an excessive number of page faults that affect system performance. Therefore a page fault from a prefetch instruction for an address within an "accessible" page does not require any general workaround.

Software can minimize the occurrence of the errata by issuing only one prefetch instruction per cache-line (a naturally-aligned 64-byte quantity) and ensuring one of the following:

- In many cases, if a particular target address of a prefetch is known to encounter this errata, simply change the prefetch to target the next byte.
- Avoid prefetching inaccessible memory locations, when possible.
- In the general case, ensure that the address used by the prefetch is offset into the middle of an aligned quadword near the end of the cache-line. For example, if the address desired to be prefetched is "ADDR", use an offset of 0x33 to compute the address used by the actual prefetch instruction as: "(ADDR & ~0x3f) + 0x33".

Resolution Status. No fix planned.

2 Revision Determination

Table 3 shows the AMD Athlon XP processor model 10 identification number returned by the CPUID instruction for each revision of the processor.

Table 3. CPUID Value for the Revision of the AMD Athlon™ XP Processor Model 10

Revision	CPUID
A2	6A0

3 Technical and Documentation Support

The following documents provide additional information regarding the operation of the AMD Athlon™ XP processor model 10. Please refer to the data sheets listed in this section for product marking information.

- *AMD Athlon™ XP Processor Model 10 Data Sheet,* order# 26237
- *Mobile AMD Athlon™ XP Processor Model 10 Data Sheet*, order# 26200
- *AMD Athlon™ and AMD Duron™ Processors BIOS, Software, and Debug Developers Guide*, order# 21656

For the latest updates, refer to *www.amd.com* and download the appropriate files. For documents under NDA, please contact your local sales representative for updates.