Model Specific Registers and Functions 26

This chapter introduces the model specific registers (MSRs) as they are implemented on the embedded Pentium® processor family. Model specific registers are used to provide access to features that are generally tied to implementation dependent aspects of a particular processor. For example, testability features that provide test access to physical structures such as caches, and branch target buffers are inherently model specific. Features to measure the performance of the processor or particular components within the processor are also model specific.

The features provided by the model specific registers are expected to change from processor generation to processor generation and may even change from model to model within the same generation. Because these features are implementation dependent, they are not recommended for use in portable software. Specifically, software developers should not expect that the features implemented within the MSRs will be supported in an upward or downward compatible manner across generations or even across different models within the same generation.

The embedded Pentium processor with MMX™ technology MSRs are different than the embedded Pentium processor MSRs. When possible, fields were preserved between the two processors. Differences between the MSRs are noted throughout this chapter.

26.1 Model Specific Registers

The embedded Pentium processor processor family implements the RDMSR and WRMSR instructions to read and write the MSR's respectively. A feature bit in EDX (bit 5), reported by the CPUID instruction, indicates whether the processor supports the RDMSR and WRMSR instructions. The Pentium processor with MMX technology implements a new instruction called RDPMC (Read Performance Monitoring Counter). This instruction enables the user to read the performance monitoring counters in "Current Privilege Level = 3" given bit 8 is set in CR4 (CR4.PCE).

26.1.1 Model Specific Register Usage Restrictions

Proper use of the MSR features described in this chapter requires that the CPUID instruction be used not only to validate that the FAMILY reported in the EAX register is equal to "5", but also to validate the specific MODEL number within that FAMILY. Note that this requirement is significantly more restrictive than is required for new architectural features where it is sufficient to validate that the FAMILY is equal to or greater than that of the first family to implement the new feature. For more information regarding the use of the CPUID instruction, refer to the *Intel Architecture Software Developer's Manual*.

26.1.2 Model Specific Register Access

Access to the model specific registers is provided through the RDMSR and WRMSR instructions. Access to a particular MSR is achieved by loading the ECX register with the appropriate ECX value from Table 26-1 below, and then executing either RDMSR or WRMSR. For more information regarding the use of these instructions, refer to the *Intel Architecture Software Developer's Manual*.

Table 26-1. Model Specific Register Descriptions

ECX Value (in Hex)	Register Name	Description
00	Machine Check Address(1)	Stores address of cycle causing the exception
01	Machine Check Type(1)	Stores cycle type of cycle causing the exception
02	Test Register 1	Parity Reversal Register
03	RESERVED	
04	Test Register 2(2)	Instruction Cache End Bit
05	Test Register 3	Cache Test Data
06	Test Register 4	Cache Test Tag
07	Test Register 5	Cache Test Control
08	Test Register 6	TLB Test Linear Address
09	Test Register 7	TLB Test Control & Physical Address 31–12
0A	RESERVED	
0B	Test Register 9	BTB Test Tag
0C	Test Register 10	BTB Test Target
0D	Test Register 11	BTB Test Control
0E	Test Register 12	New Feature Control
0F	RESERVED	
10	Time Stamp Counter	Performance Monitor
11	Control and Event Select	Performance Monitor
12	Counter 0	Performance Monitor
13	Counter 1	Performance Monitor
14+	RESERVED	

NOTES:
1. CR4.MCE must be 1 in order to utilize the machine check exception feature.
2. Reserved on the embedded Pentium® processor with MMX™ technology.

26.2 Testability And Test Registers

The processor provides testability access to the on-chip caches, TLBs, BTB and internal parity checking features through model specific test registers. The RDMSR/WRMSR instructions may be utilized by the processor to access the test registers.

26.2.1 Cache, TLB and BTB Test Registers

The processor contains several test registers. The purpose of these test registers is to provide direct access to the processor's caches, Translation Look-aside Buffers (TLB), and Branch Target Buffer (BTB) so test programs can easily exercise these structures. Because the architecture of the caches, TLBs, and BTB is different, a different set of test registers (along with a different test mechanism) is required for each processor family member. Most test registers are shared between the code and data caches.

The test registers should be written to for testability purposes only. Writing to the test registers during normal operation causes unpredictable behavior. Note that when the test registers are used to read or write lines directly to or from the cache, external inquire cycles must be inhibited to guarantee predictable results when testing. This is done by setting both CR0.CD and CR0.NW to "1". In addition, the INVD, WBINVD and INVLPG instructions may be executed before and after but not during testing.

Caution: Writing to the test registers during normal operation causes unpredictable behavior.

Since the on-board caches, TLBs, and BTB implemented in embedded Pentium processor with MMX technology differ than those in embedded Pentium processor, the test register interface differs.

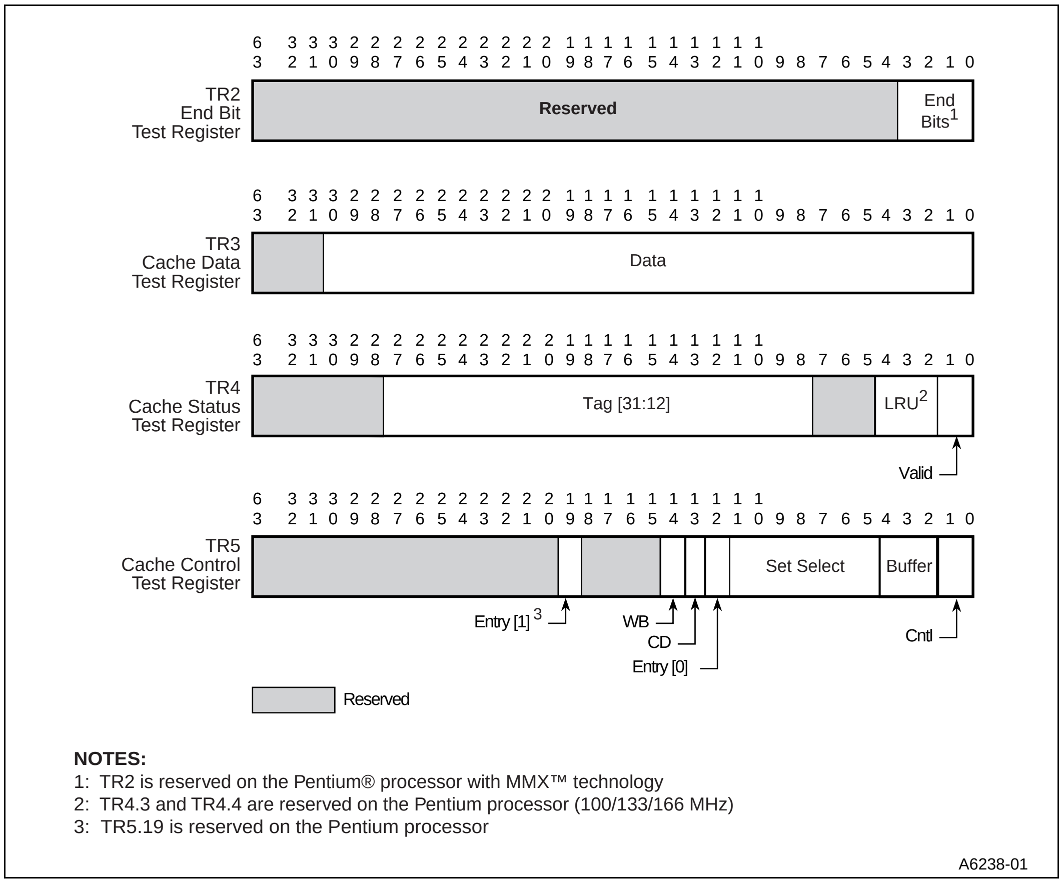
If a memory data access occurs during a code cache testability operation using the test registers, the data cache is checked before the external memory operation in initiated. If the access is a miss in the data cache, then if the accessed line is valid in the code cache, it is invalidated through the internal snooping mechanism. In addition, the same cache line fill buffer is used for cache testability writes and to temporarily store data from memory data reads. For this reason, memory data reads should be done with care or avoided to ensure data from the memory read does not overwrite data from the testability write in the cache line fill buffer.

Similarly, if a code access occurs during a data cache testability operation using the test registers, the code cache is checked before the external memory operation is initiated. If the access is a miss in the code cache, then the accessed line if valid in the data cache is invalidated (or written back and then invalidated if in the M state) through the internal snooping mechanism.

26.2.1.1 Cache Test Registers

The registers in Figure 26-1 provide direct access to the Pentium processor's code and data caches.

Figure 26-1. Cache Test Registers



On the embedded Pentium processor, TR2 is the End Bit Test Register for the code cache. It contains four end bits. Each end bit corresponds to one byte of instruction in TR3 during code cache testability access. Since a cache line has 32 bytes, eight accesses are needed to read or write the end bits for the entire cache line. TR2 is used for accesses to the code cache only. TR2 is reserved on the embedded Pentium processor with MMX technology.

End bits are used to indicate instruction boundaries on the embedded Pentium processor. The end bit mechanism aids the decode of two variable length instructions per clock by providing information on where the boundary between instruction is. If a given byte is the last byte in an instruction, the corresponding end bit is set to one. When a line is written into the code cache after a miss, all end bits corresponding to the line are initialized to one. As instructions are decoded, the end bits are checked for correctness and modified if incorrect. In order for two instructions to be issued in a single clock, the end bits of the u-pipe instruction must have the correct values, otherwise only one instruction will be issued. This does have the effect that instructions are usually not paired the first time that they are put in the code cache.

TR3 is the Cache Data Test Register. This is where the data is held on its way into or out of the cache. Prior to a cache testability write, software must load an entire cache line into the 32-byte fill buffer using TR3, 4 bytes at a time. Similarly, during a cache testability read, the processor extracts a specified 4-byte data quantity from a cache line and places the data in TR3. A 32-byte cache line may be written to or read from TR3 as eight 4-byte accesses.

TR4 is the Cache Status Test Register. It contains the tag, LRU and valid bits to be written to or read from the cache. Like TR3, TR4 must be loaded with the tag/LRU/valid bits prior to a testability write, and gets updated with the tag/LRU/valid bits as a result of a testability read. Note that TR4[31:28] are reserved and always return a zero as a result of a testability read.The two valid bits are interpreted differently by the code and data caches, depending upon the setting of TR5.CD bit. The encodings for TR4.valid are shown in Table 26-2. The encodings for the LRU bits are shown in Table 26-3 for the embedded Pentium processor and the embedded Pentium processor with MMX technology.

Table 26-2. Encoding for Valid Bits in TR4

TR5.CD=1 (Data Cache)	**valid[1]**	**valid[0]**	**Meaning**
	0	0	Cache line in I state
	0	1	Cache line in S state
	1	0	Cache line in E state
	1	1	Cache line in M state
TR5.CD=0 (Code Cache)	**valid[1]**	**valid[0]**	**Meaning**
	X	0	Cache line invalid
	X	1	Cache line valid

Table 26-3. Encoding of the LRU Bit in TR4

Pentium® Processor (100/133/166)			
LRU[0]		**Points to WAY**	
0		0	
1		1	
Pentium Processor with MMX™ Technology			
LRU[2]	**LRU[1]**	**LRU[0]**	**Points to WAY**
X	0	0	0
X	1	0	1
0	X	1	2
1	X	1	3

Note: The LRU bits for the instruction cache change state when an entry is read using the test registers, with CR0.CD=1. The LRU bits for the data cache, however, do not change their state during testability reads with CR0.CD=1.

TR5 is the Cache Control Test Register. It contains the writeback bit, the CD bit, the cache entry, the set address, the buffer select, and a two-bit control field, cntl.

The writeback bit determines whether that particular line is configured for writethrough or allows the possibility of writeback. It is used by the data cache only (i.e., if the writeback bit is set and a flush occurs (TR5.cntl=11), then if the addressed line in the data cache is modified, it will be invalidated and written back to the bus). The CD bit distinguishes between the code and data cache. The entry field selects one of the four ways in the embedded Pentium processor with MMX technology (two ways in the embedded Pentium processor) in the cache. The set address field selects one of 128 sets within the cache to be accessed. The buffer field selects one of the eight portions of a cache line to be visible through TR3. The control field selects one of the four possible operation modes. The encodings for the TR5 fields are shown in Table 26-4, Table 26-5, Table 26-6 and Table 26-7.

Table 26-4. Encoding of the WB Bit in TR5

WB	Writeback or Writethrough
0	Writethrough
1	Writeback

Table 26-5. Encoding of the Code/Data Cache Bit in TR5

CD	Cache
0	Code cache
1	Data cache

Table 26-6. Encoding of the Entry Bit in TR5

Entry[1]	Entry[0]	Way
0	0	0
0	1	1
1	0	2
1	1	3

Note: The Entry[1] bit, Way 2 and Way 3 are specific to the embedded Pentium processor with MMX technology.

Table 26-7. Encoding of the Control Bits in TR5

Cntl1	Cntl0	Command
0	0	Normal operation
0	1	Testability write
1	0	Testability read
1	1	Flush

Direct Cache Access

To access the cache for testing, the programmer specifies a set address and entry and requests a testability read or write. No tag comparison is done; the programmer can directly read/write a particular entry in a particular set. Note that since TR2 is reserved for the embedded Pentium processor with MMX technology, there is no TR2 access when reading an entry from the cache.

To write down an entry into the cache:

- Disable replacements by setting CR0.CD=1.
- For each 4-byte access:
  - Write address into TR5.buffer. Here, TR5.cntl=00.
  - Write data into TR3.
  - Write end bits into TR2 (for instruction cache only).
- Write the desired tag, LRU and valid bits into TR4. Note that the contents of TR4 completely overwrites the previous tag, LRU and valid bits in the cache.
- Perform a testability write by loading TR5 with the appropriate CD, entry, set address, and cntl fields. Here, TR5.cntl=01.

To read an entry from the cache:

- For each 4-byte access:
  - Write the appropriate CD, entry, set address, buffer and cntl fields into TR5. Here, TR5.cntl=10.
  - Read data from TR3.
  - Read end bits from TR2[3:0] (for instruction cache only).
  - Read the tag, LRU, and valid bits from TR4. No hit/comparison is performed. Whatever was in that entry in that set is read into TR4, TR3, and TR2.

To invalidate the cache or invalidate an entry:

- When TR5.cntl=11 (flush), and CD=0 (code cache), the entire code cache is invalidated. However, if TR5.cntl=11 and CD=1 (data cache), the user can specify through the TR5.WB bit whether to invalidate the entire data cache, or invalidate and writeback only the cache line specified by TR5 (see Figure 26-8).

Table 26-8. Definition of the WB Bit in TR5

TR5.cntl=11	TR5.WB	Meaning
CD=0	X	Invalidate the entire code cache.
CD=1	0	Invalidate entire data cache. Modified lines are not written back.
CD=1	1	Invalidate line. Writeback if modified.

Note that TR2, TR3, and TR4 permit both reads and writes, whereas TR5 is a write-only register. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation may cause unpredictable behavior. For example, inadvertent cache hits can be created.

Note: During cache testability operations, the internal snooping mechanism functions similar to that described in “Internal Snooping” on page 19-40. If a memory data access occurs during a code cache testability operation using the test registers, the date cache is checked before the external memory operation is initiated. If the access is a miss in the data cache, then the accessed line if valid in the code cache is invalidated through the internal snooping mechanism. In addition, the same cache line fill buffer is used for cache testability writes and to temporarily store data from memory data reads. For this reason, memory data reads should be done with care or avoided to ensure data from the memory read does not overwrite data from the testability write in the cache line fill buffer.

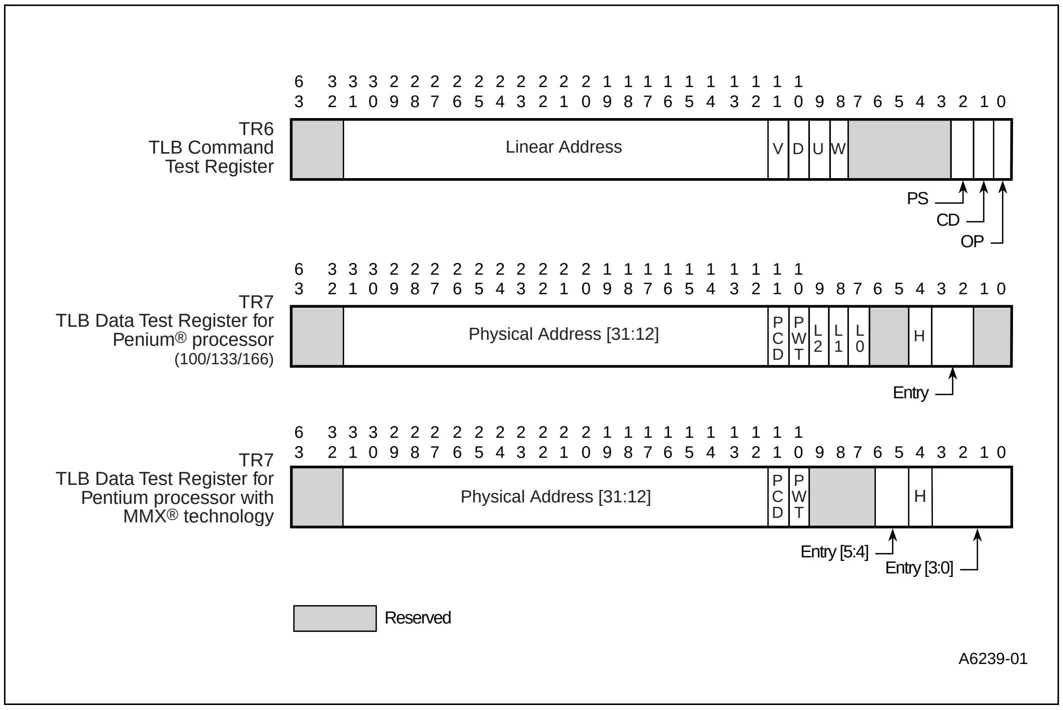
Similarly, if a code access occurs during a data cache testability operation using the test registers, the code cache is checked before the external memory operation is initiated. If the access is a miss in the code cache, then the accessed line if valid in the data cache is invalidated (or written back and then invalidated if in the M-state) through the internal snooping mechanism.

When the FLUSH# pin is asserted, it is treated as an interrupt, and when serviced at the next instruction boundary, it causes a writeback of the data cache and then invalidation of the internal caches. The cache test registers TR2, TR3, TR4 and TR5 are used in this process, and thus their values after FLUSH# has been serviced are unpredictable. Therefore FLUSH# should not be asserted while code is being executed which uses these test registers.

26.2.1.2 TLB Test Registers

The registers in Figure 26-2 provide access to the Pentium processor's code and data cache translation lookaside buffers (TLBs). Note that the data cache has two TLBs: a 64-entry TLB for 4-Kbyte data pages and an 8-entry TLB for 4-Mbyte data pages. The code cache contains only one 32-entry TLB for both 4-Kbyte code pages and 4-Mbyte code pages. The 4-Mbyte code pages are cached in 4-Kbyte increments (the PS bit in TR6 is ignored). The code cache contains one fully associative 32-entry TLB which is also integrated for both 4-Kbyte and 4-Mbyte pages. Note that, unlike the embedded Pentium processor, the embedded Pentium processor with MMX technology data cache contains one fully associative 64-entry TLB which is integrated for both 4-Kbyte and 4-Mbyte pages.

Figure 26-2. TLB Test Registers



TR6 is the TLB Command Test Register. It contains the linear address, code/data TLB select (CD), operation (Op) bits and the following status bits: valid (V), dirty (D), user (U), writeable (W), and page size (PS) bits.

The status bits are inputs to the TLB entry during testability writes, and outputs from the TLB entry during testability reads. The V bit indicates whether a TLB entry is valid or invalid during testability writes. The D bit indicates whether or not a write access was made to the page. The U bit indicates the privilege level that the processor must be in to access the page. The W bit is one of the factors in determining the read/write protection of the page. The PS (page size) bit specifies the page size for the TLB entry. The CD bit determines if the code or data TLB is being accessed. The Op bit distinguished between a read and write cycle.

The W-bit, D-bit, and PS-bit are defined only for the data TLB.

Tables 26-9 through 26-16 list the encodings for the fields in the TR6 register.

Table 26-9. Encoding for the Valid Bit in TR6

Valid	Valid/Invalid TLB Entry
0	Invalid
1	Valid

Table 26-10. Encoding for the Dirty Bit in TR6

D-bit	Write access made to page?
0	Write access was not made
1	Write access was made

Table 26-11. Encoding for the User Bit in TR6

U-bit	Privilege Level Access Allowed
0	PL=0,1,2,3
1	PL=0

Table 26-12. Encoding for the Writeable Bit in TR6

W-bit	Writes Allowed?
0	No writes, read only
1	Allows writes

Table 26-13. Encoding for the Page Size Bit in TR6

PS-bit	Page Size
0	4 KByte
1	4 MByte

Note: Normally the user should not allocate a page entry in both the TLBs; during testability however if a match is found in both, then the processor reports that it found it for the 4-Mbyte page size (PS=1).

Table 26-14. Encoding for the Operation Bit TR6

Op	Command
0	TLB write
1	TLB read

Table 26-15. Encoding for the Code/Data TLB in TR6

CD	Cache
0	Code TLB
1	Data TLB

TR7 is the TLB Data Test Register. In the embedded Pentium processor it contains bits 31:12 of the physical address, the hit indicator H, a two-bit entry pointer, and the status bits. The status bits of the Pentium processor include the two paging attribute bits PCD and PWT, and three LRU bits (L0, L1, and L2). PCD is the page level cache disable bit. PWT is the page level write through bit. The LRU bits determine which entry is to be replaced according to the pseudo-LRU algorithm. TLB reads which result in hits and TLB writes can change the LRU bits. The LRU bits reported for a test read are the value before the TLB read. The LRU bits are then changed according to the pseudo-LRU replacement algorithm. The two entry bits determine which one of the four ways to write to in the code or data TLB during testability writes.

In the embedded Pentium processor with MMX technology, the entry pointer has been extended from two bits to six bits. The six entry bits determine which one of the 64 entries to write to in the data TLB during testability writes. The lower five entry bits determine which one of the 32 entries to write to in the code TLB during testability writes. Also, the L0, L1 and L2 bits are reserved in the Pentium processor with MMX technology.

The H is the hit indicator. This bit needs to be set to 1 during testability writes. During testability reads, if the input linear address matches a valid entry in the TLB, the H bit is set to 1. The two entry bits determine in which one of the four ways to write to the TLB during testability writes. During testability reads, they indicate the way that resulted in a read bit.

TR6, and TR7 are read/write registers. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation causes unpredictable behavior.

When reading from the code cache TLB (TR5.CD = 0), the TR6 register zeros out bits [31:12] (corresponding to the linear address) at the end of the TLB testability read cycle. This does not mean that an incorrect linear address was used. All operations happen normally (with whatever linear address was written into TR6 before the testability read operation).

TLB Access

Unlike the caches, the TLB is structured as a CAM cell and, thus, can only be searched (rather than directly read). In other words, the programmer can directly read/write a particular entry in a particular set of the code or data caches, however the TLB only reports a hit or a miss in the Hit bit in TR7. Dumping the TLB requires the programmer to step through the entire linear address space one page at a time. Also, please note the following changes which apply to the Pentium processor with MMX technology:

- LRU bits of TR7 (bits 9:7) are reserved on the embedded Pentium processor with MMX technology.
- The entry pointer in TR7 has been extended from two bits to six bits in the embedded Pentium processor with MMX technology.
- To assure correct functioning, software MUST flush the TLB after testability writes and prior to return to normal operation mode by writing to CR3.
- It is recommended that users do not use testability reads to load the TLB with overlapping 4 Kbyte and 4 Mbyte pages.

To write an entry into the TLB:

- Write the physical address bits [31:12], attribute bits, LRU bits and replacement entry into TR7, setting TR7.H=1.
- Write the linear address, protection bits, and page size bit into TR6, setting TR6.Op=0.

To read an entry from the TLB:

- Write the linear address, CD, and OP bits into TR6, setting TR6.Op=1.
- If TR7.H is set to 1, the read resulted in a hit. Read the translated physical address, attribute bits, and entry from TR7. Read the V, D, U, and W bits from TR6. If TR7.H is cleared to 0, the read was a miss and the physical address is undefined.

Note that when reading from the TLB, the PS bit in the TR6 register does not have to be set; the PS bit is actually written by the processor at the end of the TLB (testability) lookup. Based on the PS bit the user is supposed to infer whether the linear address found in the TLB corresponds to the 4-Kbyte or 4-Mbbyte page size. Normally the user should not allocate a page entry in both the TLBs; during testability however if a match is found in both, then the processor reports that it found it for the 4-Mbyte page size (PS=1).

Also note that when reading from the code cache TLB (TR5.CD=0), the TR6 register zeros out bits 12–31 (corresponding to the linear address) at the end of the TLB testability read cycle. This does not mean that an incorrect linear address was used. All operations happen normally (with whatever linear address was written into TR6 before the testability read operation).

26.2.1.3 Branch Target Buffer (BTB) Test Registers

The test registers in Figure 26-3 provide direct access to the branch target buffer. Note that the branch prediction mechanism should be disabled through test register 12 before doing any BTB testability access.

TR9 is the BTB Tag Test Register. Before writing any entry into the BTB, software must first load TR9 with the appropriate information. After reading any entry in the BTB, the processor places the retrieved information in TR9.

Figure 26-3. BTB Test Registers

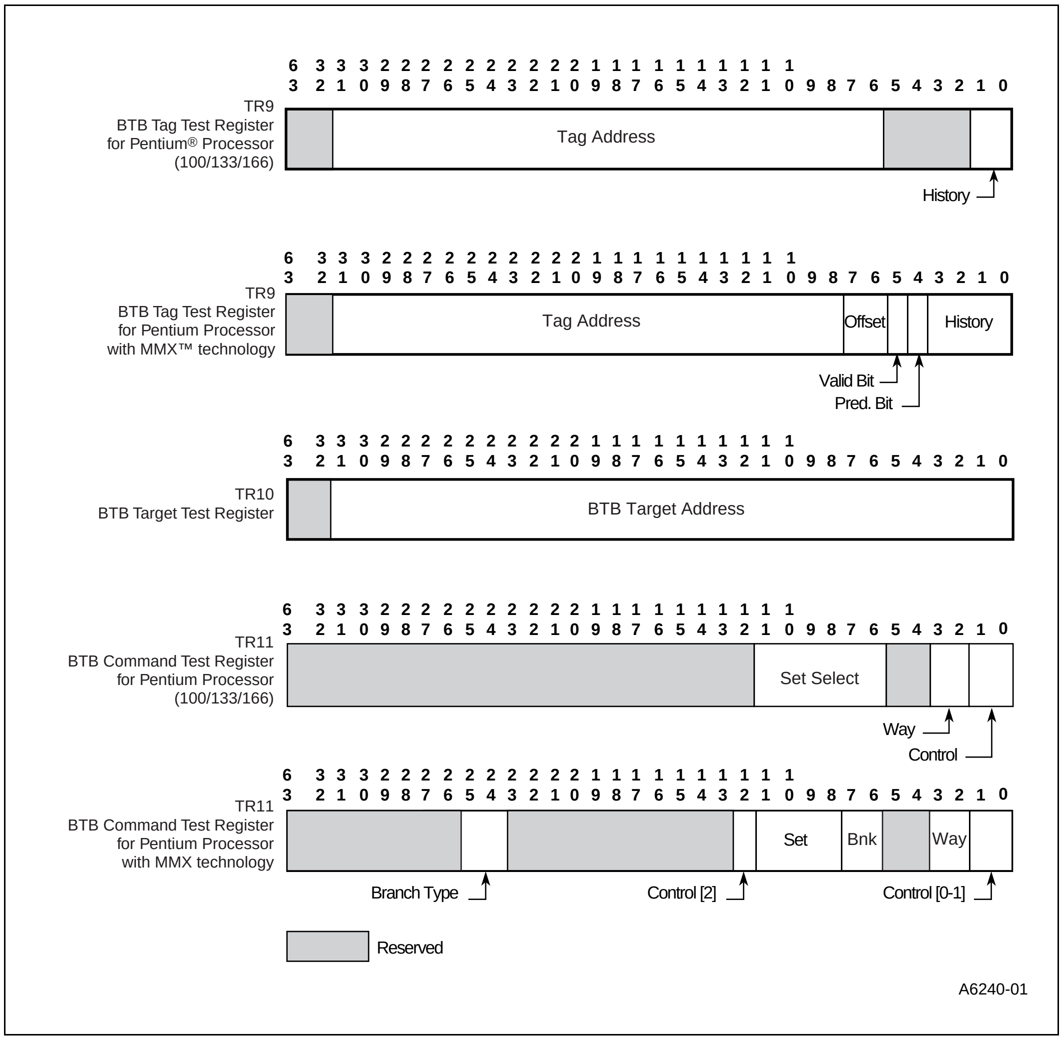


Table 26-16. TR9 Register Description (BTB Test Register)

Bits in the Embedded Pentium® Processor	Bits in the Embedded Pentium Processor with MMX™ Technology	TR9 Register Description (BTB Test Register)
63:32	63:32	Reserved
31:6	31:8	Tag Address: Bits 31:6 or 31:8 of the address of the last byte of the branch
N/A	7:6	Offset: Bits 1:0 of the address of the last byte of the branch
N/A	5	Valid bit: If set, the entry is allocated in the BTB
N/A	4	Prediction bit: Defines if this branch is predicted taken or not taken by the BTB
1:0	3:0	History: Contains the previous history for this branch

Table 26-17. TR10 Register Description (BTB Test Register)

Bits	TR10 Register Description (BTB Target Test Register)
63:32	Reserved
31:0	BTB Target Address: Linear address of the branch's target

Table 26-18. TR11 Register Description (BTB Command Test Register)

Bits in the Embedded Pentium® Processor	Bits in the Embedded Pentium Processor with MMX™ Technology	TR11 Register Description (BTB Command Test Register)
63:32	63:32	Reserved
31:12	31:26	Reserved
N/A	25:24	Branch type: 00 JCC (Jump if condition is met), 01 unconditional jump, 10 call, 11 return
N/A	23:13	Reserved
N/A	12	Control: Selects either Normal operation, or Testability Read/Write, Flush and Testability Read Tag
11:6	11:8	Set: Selects one of 64 sets to access in the embedded Pentium processor or 16 sets in the embedded Pentium processor with MMX technology
N/A	7:6	Bank: Selects one of the 4 banks per BTB cache line. The bank number corresponds to bits 3:2 of the branch address
5:4	5:4	Reserved
3:2	3:2	Way: Selects one of four ways within the Set (i.e., 00 = Way1, 01 = Way2, 10 = Way3 and 11 = Way4)
1:0	1:0	Control: Selects either Normal operation, or Testability Read/Write, Flush and Testability Read Tag

Note: The format for the control field is shown in Table 26-19.

TR10 is the BTB Target Test Register. Like TR9, TR10 must be loaded with the target address before a testability write. After a BTB testability read, the target address is placed in this register.

TR11 is the BTB Command Test Register. This register is used to issue read and write commands to the BTB. The set address field selects one of 16 sets (64 sets in the embedded Pentium processor) to access. The entry field selects one of four ways within the set on the embedded Pentium processor. A BTB testability cycle is initiated by loading TR11 controls bits with the appropriate values. The format for the control field is shown in Table 26-19.

Table 26-19. Format for TR11 Control Field

Cntl2(1)	Cntl1	Cntl0	Command
0	0	0	Normal operation
0	0	1	Testability write data
0	1	0	Testability read data
0	1	1	Testability BTB flush
1	0	1	Testability read TAG(2)

NOTES:
1. Applies to the embedded Pentium processor with MMX technology only.
2. Other combinations are reserved.

TR9, TR10 and TR11 are all read/write registers. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation causes unpredictable behavior.

The following BTB testability cycles exist:

1. Testability read data. Reads the Target, branch type, offset, history-prediction (according to spec bit)., and prediction bit of a BTB line defined by a set, way and bank into the corresponding testability register field.
2. Testability read TAG and valid bit (Pentium processor with MMX technology only). Reads the Tag defined by the testability registers set, way and bank into the corresponding testability register field.
3. Testability BTB flush. Clear all BTB valid bits.
4. Testability Write Data. Writes all the BTB fields from the corresponding test registers. If there is an entry on the same bank and set, with the same TAG, the write overwrites this entry even if the way chosen in TR11 is different from the existing entry's way. (This is done to avoid having two entries in the same bank and same set, but different ways, with the same TAG.)

TR9, TR10, TR11 are all read/write registers. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation causes unpredictable behavior.

Direct BTB Access

The BTB contents are directly accessible, in a manner similar to the code/data caches. Note that the branch prediction mechanism should be disabled before doing any BTB testability access.

To write an entry into the BTB for the embedded Pentium processor:

1. Disable BTB entry allocation by setting TR12.NBP=1 (see Feature Control section)
2. Write the tag address and history information in TR9

3. Write the target address in TR10
4. Write the appropriate set address, entry fields and control bits in TR11.

To write an entry into the BTB for the embedded Pentium processor with MMX technology:

1. Disable BTB entry allocation by setting TR12.NBP=1 (see Feature Control section)
2. Write the tag address and history, offset, valid and prediction information in TR9
3. Write the target address in TR10
4. Write the appropriate set address and entry fields, way, bank, branch type and control bits in TR11.

To read an entry from the BTB for the embedded Pentium processor:

1. Perform a testability read by writing to TR11 with the appropriate set address entry fields.
2. Read the tag address and history information from TR9.
3. Read the target address from TR10.

To read an entry from the BTB for the embedded Pentium processor with MMX technology:

1. Disable BTB entry allocation by setting TR12.NBP=1 (see Feature Control section)
2. Perform a testability read by writing to TR11 with the appropriate set address and entry fields, way, bank and control bits.
3. Read the tag address, history information, offset, prediction and valid bits from TR9.
4. Read the target address from TR10.
5. Read the branch type from TR11.
6. Perform a testability read tag by writing to TR11 with the appropriate set address, way, bank and control bits.
7. Read the branch tag from TR9

Note: Read Tag and Read data does not destroy the other's cycle fields in TR9. This means that the read from TR9 can be done only once after both cycles were executed.

26.2.1.4 Parity Reversal Register (TR1)

A model specific register, TR1, the Parity Reversal Register (PRR), allows the parity check mechanism to be tested. Figure 26-4 shows the format of the PRR.

Figure 26-4. Parity Reversal Register

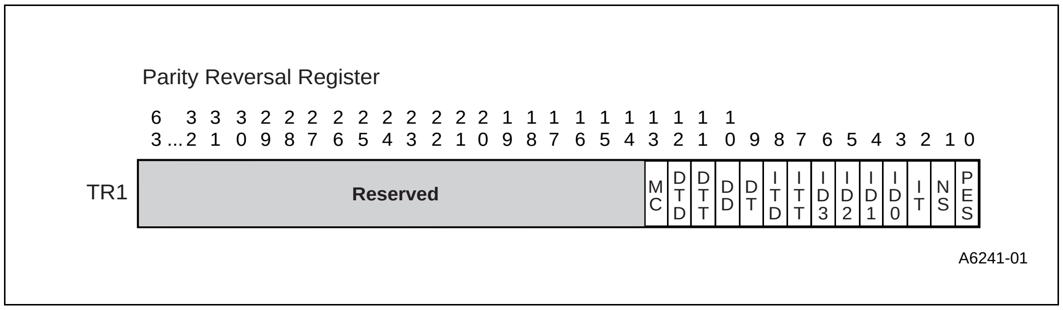


Table 26-20 lists each of the bits in the parity reversal register and their function.

Table 26-20. Parity Reversal Register Bit Definition

Bit Name	Description
PES	Parity Error Summary, set on any parity error
NS	0 = set PRR.PES, assert IERR#, and shutdown on parity error 1 = set PRR.PES, and assert IERR# on parity error
IT	code (instruction) cache tag
ID0	code cache data even bits 126, 124... 2,0
ID1	code cache data odd bits 127, 125... 3,1
ID2	code cache data even bits 254, 252... 130,128
ID3	code cache data odd bits 255, 253... 131, 129
ITT	code TLB tag
ITD	code TLB data
DT	data cache tag
DD	data cache data, use byte writes for individual access
DTT	data TLB tag
DTD	data TLB data
MC	microcode, reverse parity on read

Writing a one into bits 2-12 reverses the sense of the parity generation for any write into the corresponding array. This includes both normal cache replacements as well as testability writes and data writes. Parity is checked during both normal reads and testability reads.

To test parity error detection, software should write a one into the appropriate bit of the parity reversal register (PRR), perform a testability write into the array, and then perform a testability read. Upon successful detection of the parity error, the Pentium processor asserts the IERR# pin and may shutdown. Alternatively, after writing a one into the appropriate bit of the PRR, software may perform a normal write and read of the array by creating a cache miss and doing a read.

As an option, software may mask the shutdown by setting PRR.NS to 1 if the system is unable to recover from a shutdown. To determine if a parity error has occurred, software may read the parity error summary bit, PRR.PES. Hardware sets this bit on any parity error, and it remains set until cleared by software.

For the microcode, bad parity may be forced on a read by a transition of the PRR.MC bit from 0 to 1. No bad parity will be forced by setting the PRR.MC bit if the bit was already set.

Bit 0 of TR1 is read/write. The remaining bits are write only. The test registers should be written to for testability accesses only. Writing to the test registers during normal operation causes unpredictable behavior.

26.3 New Feature Control (TR12)

The new features of branch prediction, execution tracing, and instruction pairing in the Pentium processor can be selectively enabled or disabled through individual bits in test register TR12 (Figure 26-5). The branch prediction, execution tracing, and instruction pairing features of the Pentium processor family can be selectively enabled or disabled through individual bits in test register TR12. In addition, level 1 caching can be disabled without affecting the PCD output to allow testing of a second level cache.

Figure 26-5. Test Register (TR12)

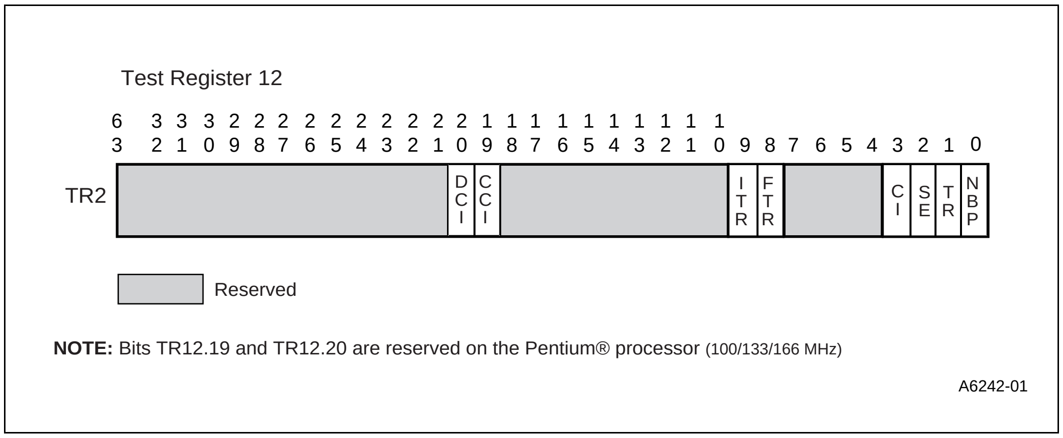


Table 26-21. New Feature Controls

Name	Position	Function
NBP	0	No Branch Prediction controls the allocation of new entries in the BTB. When TR12.NBP is clear, the code cache allocates entries in the BTB. When TR12.NBP is set, no new entry is allocated in the BTB, however, entries already in the BTB may continue to cause a BTB hit and result in the pipeline being reloaded from the predicted branch target. To completely disable branch prediction, first set TR12.NBP to 1 and then flush the entire BTB by loading CR3.
TR	1	Execution Tracing controls the Branch Trace message Special Cycle. When the TR12.TR bit is set to 1, a branch trace message special cycle is generated whenever a taken branch is executed. Two cycles are produced: one for the linear address of the instruction causing the taken branch, and one for the branch target linear address.
SE	2	Single Pipe Execution controls instruction pairing. When TR12.SE is cleared to zero, instructions are issued to both the u and v pipes contingent on pairing restrictions. When TR12.SE is set to one, the v pipe is disabled and instructions are issued only to the u pipe. Microcoded instructions are designed to utilize both pipes concurrently, independent of the state of TR12.SE. Note that all instructions requiring microcode are not pairable.
CI	3	Cache Inhibit controls line fill behavior. When TR12.CI is reset to 0, the on-chip data and instruction caches operate normally. When TR12.CI is set to 1, all cache line fills are inhibited and all bus cycles due to cache misses are run as single transfer cycles (CACHE# is not asserted). Unlike CR0.CD, TR12.CI does not affect the state of the PCD output pin. This allows the first level cache to be disabled while the second level cache is still active and can be tested. Note that the contents of the instruction and data caches are not affected by the state of TR12.CI, e.g., they are not flushed. The second level cache test sequence should be: set TR12.CI to 1, flush the internal caches, run the second level cache tests.
	4-7	Reserved
FTR	8	Fast Execution Tracing is similar to Execution Tracing (TR12.TR). If TR12.FTR is set to 1 while execution tracing is enabled (TR12.TR = 1), only one branch trace message special cycle is produced containing the linear address of the instruction causing the taken branch.
ITR	9	IO Trap Restart enables proper interrupt prioritization to support restarting IO accesses trapped by System Management Mode.
	10–18	Reserved
CCI	19[†]	Code Cache Inhibit is the same instruction as Cache Inhibit (CI), but only applies to the code cache.
DCI	20[†]	Data Cache Inhibit is the same instruction as Cache Inhibit (CI), but only applies to the data cache.
	21–63	Reserved

† These bits are reserved on the Pentium processor (100/133/166).

TR12.NBP, TR12.TR, TR12.SE, and TR12.CI are initialized to zero on reset. This register is write only and the reserved bits should be written with zeros.

26.4 Performance Monitoring

The processor includes features to measure and monitor various parameters that contribute to the performance of the processor. This information can be then used for compiler and memory system tuning. For memory system tuning, it is possible to measure data and instruction cache hit rates, and time spent waiting for the external bus. The performance monitor allows compiler writers to gauge the effectiveness of instruction scheduling algorithms by measuring address generation interlocks and parallelism.

While the performance monitoring features that are provided by the Pentium processor are generally model specific and available only to privileged software, the Pentium processor also provides an *architectural* Time Stamp Counter that is available to the user. With this notable exception, the performance monitor features and the events they monitor are otherwise implementation dependent, and consequently, they are not considered part of the Pentium processor *architecture*. The performance monitor features are expected to change in future implementations.

Note: It is essential that software abide by the usage restrictions for accessing model specific registers as discussed in section “Model Specific Register Usage Restrictions” on page 26-1.

26.4.1 Performance Monitoring Feature Overview

Processor performance monitoring features include:

Table 26-22. Architectural Performance Monitoring Features

RDTSC	Read Time Stamp Counter - a user level instruction to provide read access to a 64-bit free-running counter
RDPMC	Read Performance Monitoring Counter - this instruction enables reading of the performance monitoring counters (in CPL = 3) provided bit 8 of CR4 (CR4.PCE) is set. Note: The RDPMC instruction is only defined on the embedded Pentium processor with MMX technology. Execution of the RDPMC instruction in a embedded Pentium processor will result in an invalid opcode exception.
CPUID (EDX.TSC)	Time Stamp Counter Feature Bit - Bit 4 of EDX is set to 1 to indicate that the processor implements the TSC and RDTSC instruction
CR4.TSD	Time Stamp Disable - A method for a supervisor program to disable user access to the time stamp counter in secure systems. When bit 2 of CR4 is set to 1, an attempt to execute the RDTSC instruction generates an general protection exception (#GP).

Table 26-23. Model Specific Performance Monitoring Features

CTR0, CTR1	Counter 0, Counter 1 - two programmable counters
CESR	Control and Event Select Register - programs CTR0, CTR1
TSC	Time Stamp Counter - provides read and write access to the architectural 64-bit counter in a manner that is model specific.
PM0/BP0, PM1/BP1	Event Monitoring Pins - These pins allow external hardware to monitor the activity in CTR0 and CTR1.

26.4.2 Time Stamp Counter (TSC)

A dedicated, free-running, 64-bit time stamp counter is provided on chip. Note that on the Pentium processor, this counter increments on every clock cycle, although it is not guaranteed that this will be true on future processors. As a time stamp counter, the RDTSC instruction reports values that are guaranteed to be unique and monotonically increasing. Portable software should not expect that the counter reports absolute time or clock counts. The user level RDTSC (Read Time Stamp Counter) instruction is provided to allow a program of any privilege level to sample its value. A bit in CR4, TSD (Time Stamp Disable) is provided to disable this instruction in secure environments. Supervisor mode programs may sample this counter using the RDMSR instruction or reset/preset this counter with a WRMSR instruction. The counter is cleared after reset.

While the user level RDTSC instruction and a corresponding 64-bit time stamp counter will be provided in all future Pentium processor compatible processors, access to this counter via the RDMSR/WRMSR instructions is dependent upon the particular implementation.

26.4.3 Programmable Event Counters (CTR0, CTR1)

Two programmable 40-bit counters CTR0 and CTR1 are provided. The implementation of these two counters is slightly different between the embedded Pentium processor with MMX technology and the embedded Pentium processor. In the embedded Pentium processor each counter may be programmed to count any event from a pre-determined list of events. These events, which are described in the *Events* section of this chapter, are selected by programming the Control and Event Select Register (CESR). In the embedded Pentium processor with MMX technology some additional events were added and cannot be assigned to either of the two counters independently. These new events are paired, so when one event is assigned to counter 0, a second related event is automatically assigned to counter 1. The counters are not affected by writes to CESR and must be cleared or pre-set when switching to a new event. The counters are undefined after RESET.

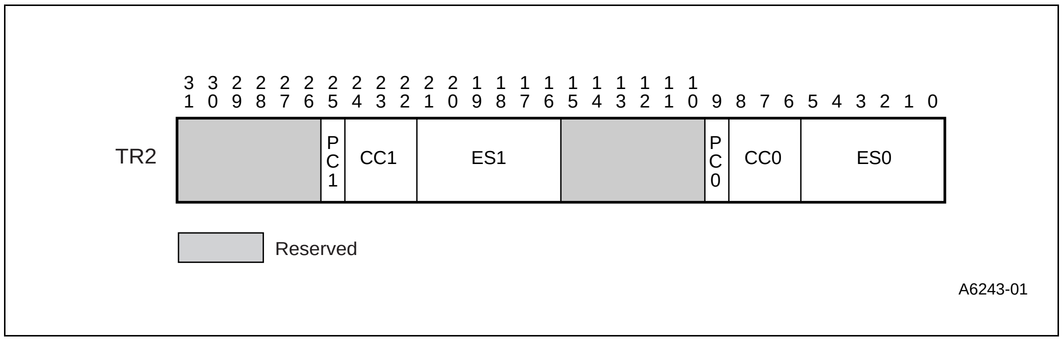
Associated with each counter is an event pin (PM1/BP1, PM0/BP0) which externally signals the occurrence of the selected event.

Note that neither the CTR0/CTR1 nor CESR are part of the processor state that is automatically saved and restored during a context switch. If it is desired to coordinate the use of the programmable counters in a multiprocessing system, it is the software's responsibility to share or restrict the use of these counters through a semaphore or other appropriate mechanism.

26.4.4 Control and Event Select Register (CESR)

A 32-bit Control and Event Select Register (CESR) is used to control operation of the programmable counters and their associated pins. Figure 26-6 depicts the CESR. For each counter, the CESR contains a 6-bit Event Select field (ES), a Pin Control bit (PC), and a three bit control field (CC). It is not possible to selectively write a subset of the CESR. If only one event needs to be changed, the CESR must first be read, the appropriate bits modified, and all bits must be written back. At reset, all bits in the Control and Event Select Register are cleared.

Figure 26-6. Control and Event Select Register



26.4.4.1 Event Select (ES0, ES1)

Up to two events may be monitored by placing the appropriate event code in the Event Select field. The events and codes are listed in the Events section of this chapter.

26.4.4.2 Counter Control (CC0, CC1)

A three bit field is used to control the operation of the counter. the highest order bit selects between counting events and counting clocks. The middle bit enables counting when the CPL=3. The low order bit enables counting when the CPL=0, 1 or 2.

CC	Meaning
000	Count Nothing (Disable Counter)
001	Count the selected Event while the CPL=0, 1 or 2
010	Count the selected Event while the CPL=3
011	Count the selected Event regardless of the CPL
100	Count Nothing (Disable Counter)
101	Count Clocks while the CPL=0, 1 or 2
110	Count Clocks while the CPL=3
111	Count Clocks regardless of the CPL

While a counter need not be stopped to sample its contents, it must be stopped and cleared or preset before switching to a new event.

26.4.4.3 Pin Control (PC0, PC1)

Associated with CTR0 and CTR1 are two pins, PM0 and PM1 (PM0/BP0, PM1/BP1), and two bits which control their operation, PC0 and PC1. These pins may be programmed by the PC0/PC1 bits in the CESR to indicate either that the associated counter has incremented or that it has overflowed. Note that the external signalling of the event on the pins will lag the internal event by a “few” clocks as the signals are latched and buffered.

PC	PM pin signals when the corresponding counter:
0	has incremented
1	has overflowed

When the pins are configured to signal that a counter has incremented, it should be noted that although the counters may increment by 1 or 2 in a single clock, the pins can only indicate that the event occurred. Moreover, since the internal clock frequency may be higher than the external clock frequency, a single external clock may correspond to multiple internal clocks.

A “count up to” function may be provided when the event pin is programmed to signal an overflow of the counter. Because the counters are 40 bits, a carry out of bit 39 indicates an overflow. A counter may be preset to a specific value less than 2^{40} - 1. After the counter has been enabled and the prescribed number of events has transpired, the counter will overflow. Approximately 5 clocks later, the overflow is indicated externally and appropriate action, such as signaling an interrupt, may then be taken.

When the performance monitor pins are configured to indicate when the performance monitor counter has incremented and an “occurrence event” is being counted, the associated PM pin is asserted (high) each time the event occurs. When a “duration event” is being counted the associated PM pin is asserted for the entire duration of the event. When the performance monitor pins are configured to indicate when the counter has overflowed, the associated PM pin is not asserted until the counter has overflowed.

The PM0/BP0, PM1/BP1 pins also serve to indicate breakpoint matches during in Circuit Emulation, during which time the counter increment or overflow function of these pins is not available. After RESET, the PM0/BP0, PM1/BP1 pins are configured for performance monitoring, however a hardware debugger may re-configure these pins to indicate breakpoint matches.

26.4.5 Performance Monitoring Events

Events may be considered to be of two types: those that count OCCURRENCES, and those that count DURATION. Each of the events listed below is classified accordingly.

Occurrences events are counted each time the event takes place. If the PM0 or PM1 pins are configured to indicate when a counter increments, they are asserted each clock the counter increments. Note that if an event can happen twice in one clock the counter increments by 2, however the PM0/1 pins are asserted only once.

For Duration events, the counter counts the total number of clocks that the condition is true. When configured to indicate when a counter increments, the PM0 and PM1 pins are asserted for the duration of the event.

Table 26-24 lists the events that can be counted, and their encodings for the Control and Event Select Register.

The performance monitoring features present in the embedded Pentium processor have been extended in the embedded Pentium processor with MMX technology. The event list is longer, and there is a new instruction defined to facilitate use of the instruction monitoring. To leave room for future additions all new embedded Pentium processor with MMX technology events are assigned to just one of the two events counters (CTR0, CTR1). It is not possible to assign these events to any of the two counters at will. “Twin events” (such as “D1 starvation and FIFO is empty”) are assigned to different counters to allow their concurrent measurement.

The Read Performance Monitoring Counter (RDPMC) is implemented in the embedded Pentium processor with MMX technology. See the *Intel Architecture Software Developer’s Manual* for more information about the RDPMC instruction.

Table 26-24. Performance Monitoring Events (Sheet 1 of 4)

Decimal Encoding	Binary Encoding	Counter 0	Counter 1	Performance Monitoring Event	Occurrence or Duration?
0	000000	Yes	Yes	Data Read	Occurrence
1	000001	Yes	Yes	Data Write	Occurrence
2	000010	Yes	Yes	Data TLB Miss	Occurrence
3	000011	Yes	Yes	Data Read Miss	Occurrence
4	000100	Yes	Yes	Data Write Miss	Occurrence
5	000101	Yes	Yes	Write (hit) to M- or E-state lines	Occurrence
6	000110	Yes	Yes	Data Cache Lines Written Back	Occurrence
7	000111	Yes	Yes	External Snoops	Occurrence
8	001000	Yes	Yes	External Data Cache Snoop Hits	Occurrence
9	001001	Yes	Yes	Memory Accesses in Both Pipes	Occurrence
10	001010	Yes	Yes	Bank Conflicts	Occurrence

NOTE: Shaded areas only apply to the embedded Pentium® processor with MMX™ technology.

Table 26-24. Performance Monitoring Events (Sheet 2 of 4)

Decimal Encoding	Binary Encoding	Counter 0	Counter 1	Performance Monitoring Event	Occurrence or Duration?
11	001011	Yes	Yes	Misaligned Data memory or I/O References	Occurrence
12	001100	Yes	Yes	Code Read	Occurrence
13	001101	Yes	Yes	Code TLB Miss	Occurrence
14	001110	Yes	Yes	Code Cache Miss	Occurrence
15	001111	Yes	Yes	Any Segment Register Loaded	Occurrence
16	010000	Yes	Yes	Reserved	
17	010001	Yes	Yes	Reserved	
18	010010	Yes	Yes	Branches	Occurrence
19	010011	Yes	Yes	BTB Hits	Occurrence
20	010100	Yes	Yes	Taken Branch or BTB hit	Occurrence
21	010101	Yes	Yes	Pipeline Flushes	Occurrence
22	010110	Yes	Yes	Instructions Executed	Occurrence
23	010111	Yes	Yes	Instructions Executed in the v pipe e.g. parallelism/pairing	Occurrence
24	011000	Yes	Yes	Clocks while a bus cycle is in progress (bus utilization)	Duration
25	011001	Yes	Yes	Number of clocks stalled due to full write buffers	Duration
26	011010	Yes	Yes	Pipeline stalled waiting for data memory read	Duration
27	011011	Yes	Yes	Stall on write to an E- or M-state line	Duration
28	011100	Yes	Yes	Locked Bus Cycle	Occurrence
29	011101	Yes	Yes	I/O Read or Write Cycle	Occurrence
30	011110	Yes	Yes	Non-Cacheable memory reads	Occurrence
31	011111	Yes	Yes	Pipeline stalled because of an address generation interlock	Duration
32	100000	Yes	Yes	Reserved	
33	100001	Yes	Yes	Reserved	
34	100010	Yes	Yes	FLOPs	Occurrence
35	100011	Yes	Yes	Breakpoint match on DR10 Register	Occurrence
36	100100	Yes	Yes	Breakpoint match on DR1 Register	Occurrence
37	100101	Yes	Yes	Breakpoint match on DR2 Register	Occurrence
38	100110	Yes	Yes	Breakpoint match on DR3 Register	Occurrence
39	100111	Yes	Yes	Hardware Interrupts	Occurrence
40	101000	Yes	Yes	Data Read or Data Write	Occurrence
41	101001	Yes	Yes	Data Read Miss or Data Write Miss	Occurrence

NOTE: Shaded areas only apply to the embedded Pentium® processor with MMX™ technology.

Table 26-24. Performance Monitoring Events (Sheet 3 of 4)

Decimal Encoding	Binary Encoding	Counter 0	Counter 1	Performance Monitoring Event	Occurrence or Duration?
42	101010	Yes	No	Bus Ownership Latency	Duration
42	101010	No	Yes	Bus Ownership Transfers	Occurrence
43	101011	Yes	No	MMX instructions executed in u pipe	Occurrence
43	101011	No	Yes	MMX instructions executed in v pipe	Occurrence
44	101100	Yes	No	Cache M-Sate line Sharing	Occurrence
44	101100	No	Yes	Cache Line Sharing	Occurrence
45	101101	Yes	No	EMMS instructions executed	Occurrence
45	101101	No	yes	Transition between MMX™ instructions and FP instructions	Occurrence
46	101110	Yes	No	Bus Utilization Due to processor Activity	Duration
46	101110	No	Yes	Writes to Non-Cacheable Memory	Occurrence
47	101111	Yes	No	Saturating MMX instructions executed	Occurrence
47	101111	No	Yes	Saturations performed	Occurrence
48	110000	Yes	No	Number of Cycles Not in HLT State	Duration
48	110000	No	Yes	Number of Cycles Not in HLT State	Duration
49	110001	Yes	No	MMX instruction data reads	Occurrence
49	110001	No	Yes	MMX instructions data read misses	Occurrence
50	110010	Yes	No	Floating Point Stalls	Duration
50	110010	No	Yes	Taken Branches	Occurrence
51	110011	Yes	No	D 1 Starvation and FIFO is empty	Occurrence
51	110011	No	Yes	D1 Starvation and only one instruction in FIFO	Occurrence
52	110100	Yes	No	MMX instruction data writes	Occurrence
52	110100	No	Yes	MMX instruction data write misses	Occurrence
53	110101	Yes	No	Pipeline flushed due to wrong branch prediction	Occurrence
53	110101	No	Yes	Pipeline flushes due to wrong branch predictions resolved in WB-stage	Occurrence
54	110110	Yes	No	Misaligned data memory references on MMX instruction	Occurrence
54	110110	No	Yes	Pipeline stalled waiting for MMX instruction data memory read	Duration
55	110111	Yes	No	Returns Predicted Incorrectly or not predicted at all	Occurrence
55	110111	No	Yes	Returns Predicted (Correctly and Incorrectly)	Occurrence
56	111000	Yes	No	MMX multiply unit interlock	Duration

NOTE: Shaded areas only apply to the embedded Pentium® processor with MMX™ technology.

Table 26-24. Performance Monitoring Events (Sheet 4 of 4)

Decimal Encoding	Binary Encoding	Counter 0	Counter 1	Performance Monitoring Event	Occurrence or Duration?
56	111000	No	Yes	MOVD/MOVQ state stall due to previous operation	Duration
57	111001	Yes	No	Returns	Occurrence
57	111001	No	Yes	Reserved	
58	111010	Yes	No	BTB false entries	Occurrence
58	111010	No	Yes	BTB miss prediction on a Not-Taken branch	Occurrence
59	111011	Yes	No	Number of clocks stalled due to full write buffers while executing MMX instructions	Duration
59	111011	No	Yes	Stall on MMX instruction write to E- or M-state line	Duration

NOTE: Shaded areas only apply to the embedded Pentium® processor with MMX™ technology.

26.4.6 Description of Events

The following descriptions clarify the events. The event codes are provided in parenthesis.

Data Read (0, 000000), Data Write (1, 000001), Data Read or Data Write (40, 101000):

These are memory data reads and/or writes (internal data cache hit and miss combined), I/O is not included. The individual component reads and writes for split cycles are counted individually. Data Memory Reads that are part of TLB miss processing are not included. These events may occur at a maximum of two per clock.

Data TLB Miss (2, 000010):

This event counts the number of misses to the data cache translation look-aside buffer.

Data Read Miss (3, 000011), Data Write Miss (4, 000100), Data Read Miss or Data Write Miss (41, 101001):

These are memory read and/or write accesses that miss the internal data cache whether or not the access is cacheable or non-cacheable. Additional reads to the same cache line after the first BRDY# of the burst linefill is returned but before the final (fourth) BRDY# has been returned, will not cause the Data Read Miss counter to be incremented additional times. Data accesses that are part of TLB miss processing are not included. Accesses directed to I/O space are not included.

Write (hit) to M- or E-state lines (5, 000101):

This measures the number of write hits to exclusive or modified lines in the data cache. (These are the writes which may be held up if EWBE# is inactive.) This event may occur at a maximum of two per clock.

Data Cache Lines Written Back (6, 000110):

This counts ALL Dirty lines that are written back, regardless of the cause. Replacements and internal and external snoops can all cause writeback and are counted.

External Snoops (7, 000111), Data Cache Snoop Hits (8, 001000):

The first event counts accepted external snoops whether they hit in the code cache or data cache or neither. Assertions of EADS# outside of the sampling interval are not counted. No internal snoops are counted. The second event applies to the data cache only. Snoop hits to a valid line in either the data cache, the data line fill buffer, or one of the write back buffers are all counted as hits.

Memory Accesses in Both Pipes (9, 001001):

Data memory reads or writes which are paired in the pipeline. Note that these accesses are not necessarily run in parallel due to cache misses, bank conflicts, etc.

Bank Conflicts (10, 001010):

These are the number of actual bank conflicts.

Misaligned Data Memory or I/O References (11, 001011):

Memory or I/O reads or writes that are misaligned. A two or four byte access is misaligned when it crosses a four byte boundary; an eight byte access is misaligned when it crosses an eight byte boundary. Ten byte accesses are treated as two separate accesses of eight and two bytes each.

Code Read (12, 001100), Code TLB Miss (13, 001101), Code Cache Miss (14, 001110):

Total instruction reads and reads that miss the code TLB or miss the internal code cache whether or not the read is cacheable or non-cacheable. Individual eight byte non-cacheable instruction reads are counted.

Any Segment Register Loaded (15, 001111):

Writes into any segment register in real or protected mode including the LDTR, GDTR, IDTR, and TR. Segment loads are caused by explicit segment register load instructions, far control transfers, and task switches. Far control transfers and task switches causing a privilege level change will signal this event twice. Note that interrupts and exceptions may initiate a far control transfer.

Branches (18, 010010):

In addition to taken conditional branches, jumps, calls, returns, software interrupts, and interrupt returns, the Pentium processor treats the following operations as causing taken branches: serializing instructions, VERR and VERW instructions, some segment descriptor loads, hardware interrupts (including FLUSH#), and programmatic exceptions that invoke a trap or fault handler. Both Taken and Not Taken Branches are counted. The pipe is not necessarily flushed. The number of branches actually executed is measured, not the number of predicted branches.

BTB Hits (19, 010011):

Hits are counted only for those instructions that are actually executed.

Taken Branch or BTB Hit (20, 010100):

This is a logical OR of taken branches and BTB hits (defined above). It represents an event that may cause a hit in the BTB. Specifically, it is either a candidate for a space in the BTB, or it is already in the BTB.

Pipeline Flushes (21, 010101):

BTB Misses on taken branches, mis-predictions, exceptions, interrupts, and some segment descriptor loads all cause pipeline flushes. This event counter will not be incremented for serializing instructions (serializing instructions cause the prefetch queue to be flushed but will not trigger the Pipeline Flushed event counter) and software interrups (software interrupts do not flush the pipeline).

Instructions Executed (22, 010110):

Up to two per clock. Invocations of a fault handler are considered instructions. All hardware and software interrupts and exceptions will also cause the count to be incremented. Repeat prefixed string instructions will only increment this counter once despite the fact that the repeat loop executes the same instruction multiple times until the loop criteria is satisfied. This applies to all the Repeat string instruction prefixes (i.e., REP, REPE, REPZ, REPNE, and REPNZ). This counter will also only increment once per each HLT instruction executed regardless of how many cycles the processor remains in the HALT state.

Instructions Executed in the v pipe e.g. parallelism/pairing (23, 010111):

Same as the Instructions executed counter except it only counts the number of instructions actually executed in the v pipe. It indicates the number of instructions that were paired.

Clocks while a bus is in progress (bus utilization) (24, 011000):

Including HLDA, AHOLD, BOFF# clocks.

Number of clocks stalled due to full write buffers (25, 011001):

This event counts the number of clocks that the internal pipeline is stalled due to full write buffers. Full write buffers stall data memory read misses, data memory write misses, and data memory write hits to S state lines. Stalls on I/O accesses are not included.

Pipeline stalled waiting for data memory read (26, 011010):

Data TLB Miss processing is also included. The pipeline stalls while a data memory read is in progress including attempts to read that are not bypassed while a line is being filled.

Locked Bus Cycle (28, 011100):

LOCK prefix or LOCK instruction, Page Table Updates, and Descriptor Table Updates. Only the Read portion of the Locked Read-Modify-Write is counted. Split Locked cycles (SCYC active) count as two separate accesses. Cycles restarted due to BOFF# are not recounted.

I/O Read or Write Cycle (29, 011101):

Bus cycles directed to I/O space. Misaligned I/O accesses will generate two bus cycles. Bus cycles restarted due to BOFF# are not re-counted.

Non-cacheable memory reads (30, 011110):

Non-cacheable instruction or data memory read bus cycles. Includes read cycles caused by TLB misses; does not include read cycles to I/O space. Cycles restarted due to BOFF# are not re-counted.

Pipeline stalled because of an address generation interlock (31, 011111):

Number of address generation interlocks (AGIs). An AGI occurring in both the u- and v- pipelines in the same clock signals this event twice. An AGI occurs when the instruction in the execute stage of either of u- or v-pipelines is writing to either the index or base address register of an instruction in the D2 (address generation) stage of either the u- or v- pipelines.

FLOPs (34, 100010);

Number of floating point adds, subtracts, multiplies, divides, remainders, and square roots. The transcendental instructions consist of multiple adds and multiplies and will signal this event multiple times. Instructions generating the divide by zero, negative square root, special operand, or stack exceptions will not be counted. Instructions generating all other floating point exceptions will be counted. The integer multiply instructions and other instructions which use the floating-point arithmetic circuitry will be counted.

Breakpoint match on DR0 Register (35, 100011),

Breakpoint match on DR1 Register (36, 100100),

Breakpoint match on DR2 Register (37, 100101),

Breakpoint match on DR3 Register (38, 100110):

If programmed for one of these breakpoint match events, the performance monitor counters will be incremented in the event of a breakpoint match whether or not breakpoints are enabled. However, if breakpoints are not enabled, code breakpoint matches will not be checked for instructions executed in the v-pipe and will not cause this counter to be incremented (they are checked on instruction executed in the u-pipe only when breakpoints are not enabled). These events correspond to the signals driven on the BP[3:0] pins. Please refer to the Debugging chapter of this volume for more information.

Hardware Interrupts (39, 100111):

Number of taken INTR and NMI only.

Bus ownership latency (42, 101010/0), Bus ownership transfers (42, 101010/1):

The first event measures the time from LRM bus ownership request to bus ownership granted, the time from the earlier of PBREQ (0), PHITM# or HITM# to PBGNT. The second event is count of the number of PBREQ (0). The ratio of these two events is the average stall time due to bus ownership conflict.

MMX instructions executed in U pipe (43, 101011/0):

Total number of MMX instructions executed in U-pipe.

MMX instructions executed in V pipe (43, 101011/1):

Total number of MMX instructions executed in V-pipe.

Cache M-state line sharing (44, 101100/0):

Counts the number of times a processor identified a hit to a modified line due to a memory access in the other processor (PHITM (O)). If the average memory latencies of the system are known, this event enables the user to count the **Write Backs on PHITM(O)** penalty and the **Latency on Hit Modified(I)** penalty.

Cache line sharing (44, 101100/1):

Counts the number of shared data lines in the L1 cache (PHIT (O)).

EMMS instructions executed (45, 101101/0):

Counts number of EMMS instructions executed.

Transition between MMX instructions and FP instructions (45, 101101/1):

Counts first floating point instruction following any MMX instruction or first MMX instruction following a floating point instruction. May be used to estimate the penalty in transitions between FP state and MMX state. An even count indicates the processor is in MMX state. an odd count indicates it is in FP state.

Bus utilization due to processor activity (46, 101110/0):

Counts the number of clocks the bus is busy due to the processor’s own activity, i.e., the bus activity which is caused by the processor.

Writes to non-cacheable memory (46, 101110/1):

Counts the number of write accesses to non-cacheable memory. It includes write cycles caused by TLB misses and I/O write cycles. Cycles restarted due to BOFF# are not recounted.

Saturating MMX instructions executed (47, 101111/0):

Counts saturating MMX instructions executed, independently of whether or not they actually saturated. Saturating MMX instructions may perform either add, subtract or pack operations.

Saturations performed (47, 101111/1):

Counts number of MMX instructions that used saturating arithmetic and that at least one of its results actually saturated; i.e., if an MMX instruction operating on four dwords saturated in three out of the four results, the counter will be incremented by one only.

Number of cycles not in HLT state (48, 110000/0):

This event counts the number of cycles the processor is not idle due to HLT instruction. This event will enable the user to calculate “net CPI”. Note that during the time that the processor is executing the HLT instruction, the Time Stamp Counter is not disabled. Since this event is controlled by the Counter Controls CC0, CC1 it can be used to calculate the CPI at CPL=3 which the TSC cannot provide.

Clocks stalled on Data cache TLB miss (48, 110000/1):

Counts the number of clocks the pipeline is stalled due to a data cache translation look-aside buffer (TLB) miss. This is the same as the event with encoding 011010 (pipeline stalled waiting for data memory read), but only for TLB miss.

MMX instruction data reads (49, 110001/0):

Analogous to “Data reads,” counting only MMX instruction accesses.

MMX instruction data read misses (49, 110001/1):

Analogous to “Data read misses,” counting only MMX instruction accesses.

Floating Point stalls (50, 110010/0):

This event counts the number of clocks while pipe is stalled due to a floating-point freeze.

Taken Branches (50, 110010/1):

This event counts the number of taken branches.

D1 starvation and FIFO is empty (51, 110011/0), D1 starvation and only one instruction in FIFO (51, 110011/1):

The D1 stage can issue 0, 1, or 2 instructions per clock if those are available in an instructions FIFO buffer. The first event counts how many times D1 cannot issue ANY instructions since the FIFO buffer is empty. The second event counts how many times the D1-stage issues just a single instruction since the FIFO buffer had just one instruction ready. Combined with previously defined events, Instruction Executed (010110) and Instruction Executed in the V-pipe (010110), the second event enables the user to calculate the numbers of time pairing rules prevented issuing of two instructions.

MMX instruction data writes (52, 110001/1):

Analogous to “Data writes,” counting only MMX instruction accesses.

MMX instruction data write misses (52, 110100/1):

Analogous to “Data write misses,” counting only MMX instruction accesses.

Pipeline flushes due to wrong branch prediction (53, 110101/0), Pipeline flushes due to wrong branch prediction resolved in WB-stage(53, 110101/1):

Counts any pipeline flush due to a branch which the pipeline did not follow correctly. It includes cases where a branch was not in the BTB, cases where a branch was in the BTB but was mispredicted, and cases where a branch was correctly predicted but to the wrong address. Branches are resolved in either the Execute stage (E-stage) or the Writeback stage (WB-stage). In the later case, the misprediction penalty is larger by one clock. The two events count the number of pipeline flushes due to wrong branch predictions. The first event counts the number of wrong branch predictions resolved in either the E-stage or the WB-stage. The second event counts the number of wrong branch prediction resolved in the WB-stage. The difference between these two counts is the number of E-stage resolved branches.

Misaligned data memory reference on MMX instruction (54, 110110/0):

Analogous to “Misaligned data memory reference,” counting only MMX instruction accesses.

Pipeline stalled waiting for MMX instruction data memory read (54, 110110/1):

Analogous to “Pipeline stalled waiting for data memory read,” counting only MMX instruction accesses.

Returns predicted incorrectly or not predicted at all (55, 110111/0):

These are the actual number of Returns that were either incorrectly predicted or were not predicted at all. It is the difference between the total number of executed returns and the number of returns that were correctly predicted. Only RET instructions are counted (e.g., IRET instructions are not counted.).

Returns predicted (correctly and incorrectly) (55, 110111/1):

This is the actual number of Returns for which a prediction was made. Only RET instructions are counted (e.g. IRET instructions are not counted).

MMX multiply unit interlock (56, 111000/0):

This is the number of clocks the pipe is stalled since the destination of previous MMX multiply instruction is not ready yet. The counter will not be incremented if there is another cause for a stall. For each occurrence of a multiply interlock this event will be counted twice (if the stalled instruction comes on the next clock after the multiply) or by one (if the stalled instruction comes two clocks after the multiply).

MOVD/MOVQ store stall due to previous operation (56, 111000/1):

Number of clocks a MOVD/MOVQ store is stalled in D2 stage due to a previous MMX operation with a destination to be used in the store instruction.

Returns (57, 111001/0):

This is the actual number of Returns executed. Only RET instructions are counted (e.g., IRET instructions are not counted). Any exception taken on a RET instruction and any interrupt recognized by the processor on the instruction boundary prior to the execution of the RET instruction will also cause this counter to be incremented.

BTB false entries (58, 111010/0):

Counts the number of false entries in the Branch Target Buffer. False entries are causes for misprediction other than a wrong prediction.

BTB miss prediction on a Not-Taken Branch (58, 111010/1):

Counts the number of times the BTB predicted a Not-Taken branch as Taken.

Number of clocks stalled due to full write buffers while executing MMX instructions (59, 111011/0):

Analogous to “Number of clocks stalled due to full write buffers,” counting only MMX instruction accesses.

Stall on MMX instruction write to an E- or M-state line (59, 111011/1):

Analogous to “Stall on write to an E- or M-state line,” counting only MMX instruction accesses.