Fachhochschule Bonn / Rhein-Sieg
Fachbereich Angewandte Informatik
Verteilte und Parallele Systeme

Performance Counter im Pentium 4

Seminararbeit

von
Michael Pietz

Bonn, 03. Januar 2002

Inhaltsverzeichnis

Seite

1 Abbildungsverzeichnis

Seite

2 Tabellenverzeichnis

Seite

3 Einleitung

Die Performance Counter eines Prozessors wurden zur Leistungsüberwachung durch Mitlesen und/oder Mitschreiben während der Programmausführung entwickelt und implementiert. Heutzutage besitzt nahezu jeder moderne Prozessor eine Vielzahl unterschiedlicher Counter, mit denen sich Ereignisse während des Programmablaufs protokollieren lassen. Dem Softwaredesigner wird hiermit die Möglichkeit gegeben, sein Programm während der Ausführung im Hinblick auf Programmablauf, Programmperformance oder auch Programmfehlverhalten zu beobachten. Die Informationen, die man aus den verschiedenen Countern – die an vielen verschiedenen Positionen Ereignisse mitzählen – erhält, können mit spezieller Software weiter verarbeitet und in aufbereiteter Form, z.B. durch Diagramme, Tabellen usw., analysiert und ausgewertet werden. Die Informationen lassen sich dann zur Abstimmung und Verbesserung der System-, Programm- und/oder Compiler - Leistung gezielt einsetzen.

Mit Countern kann man z.B. die Anzahl der Instruktionen, die der Prozessor ausgeführt hat, wie oft er Treffer im L1-Cache landete, wie viele Sprünge er richtig oder falsch vorhersagte, wie oft die zweite Pipeline zum Einsatz kam, die Anzahl von Fließkomma-Rechenoperationen, die ausgeführt wurden, und vieles mehr vorhersagen.
Die Software kann hierdurch auf die Hardware abgestimmt und entwickelt werden und nicht wie bei einem namenhaften Betriebssystemhersteller in entgegengesetzter Reihenfolge. Im Folgenden wird auf die Möglichkeiten der Leistungsüberwachung des Intel Pentium 4 (P4) Prozessors eingegangen.

4 Register im Pentium 4

Zur kurzzeitigen Speicherung von Angaben und Informationen, die zur weiteren Verarbeitung wieder zur Verfügung stehen müssen, sind die Register als feste Bestandteile der Prozessoren eingeführt worden. Sie haben in der Regel eine beschränkte Kapazität von wenigen Bytes und werden über einen Namen angesprochen.

4.1 Model Specific Registers (MSRs)

Die MSRs wurden entwickelt, um eine Vielzahl unterschiedlicher Hard- und Softwarefunktionen zu kontrollieren und zu überwachen (siehe Anhang Tabelle A-2), einschließlich der Performance Counter Ereignisse (siehe Anhang Tabelle A-1). Die Implementierung der MSRs ändert sich von Prozessorgeneration zu Prozessorgeneration. Funktionen, die gerade noch unterstützt wurden, sind in der folgenden Version möglicherweise nicht mehr implementiert oder an eine andere Position verlegt worden. Die MSRs müssen für ihre jeweiligen Einsatzgebiete wie z.B. das Perfomance Counting oder auch Hardwarechecks/Systemchecks konfiguriert und initialisiert werden, bevor sie diese Aufgaben ausführen können. Das Auslesen und Beschreiben der MSRs funktioniert mit dem Befehl Read MSR (RMSR) bzw. Write MSR (WMSR).

4.2 Event Selection Contol Register (ESCR) MSRs

Mit der entsprechenden Software werden über die ESCR MSRs die speziellen Ereignisse (Aktionen/Zugriffe) ausgewählt, die von Countern gezählt werden sollen. Jedes ESCR ist mit einem Paar von Performance Countern (i.d.R zwei Counter) verbunden und jedem Performance Counter stehen verschiedene ESCRs zur Verfügung (siehe folgenden Auszug aus der Tabelle mit den Assoziationen zwischen Performance Countern und ESCRs).

Counter			CCCR		ESCR		
Name	Nr.	Adresse	Name	Adresse	Name	Nr.	Adresse
MSR_BPU_COUNTER0	0	300H	MSR_BPU_CCCR0	360H	MSR_BSU_ESCR0	7	3A0H
					MSR_FSB_ESCR0	6	3A2H
					MSR_MOB_ESCR0	2	3AAH
					MSR_PMH_ESCR0	4	3ACH
					MSR_BPU_ESCR0	0	3B2H
					MSR_IS_ESCR0	1	3B4H
					MSR_ITLB_ESCR0	3	3B6H
					MSR_IX_ESCR0	5	3C8H
MSR_BPU_COUNTER1	1	301H	MSR_BPU_CCCR1	361H	MSR_BSU_ESCR0	7	3A0H
					MSR_FSB_ESCR0	6	3A2H
					MSR_MOB_ESCR0	2	3AAH
					MSR_PMH_ESCR0	4	3ACH
					MSR_BPU_ESCR0	0	3B2H
					MSR_IS_ESCR0	1	3B4H
					MSR_ITLB_ESCR0	3	3B6H
					MSR_IX_ESCR0	5	3C8H

Tabelle 1: Performance Counter MSRs und zugehörige CCCR und ESCR MSRs

(Quelle: IA32 Intel Architecture Software Developer's Manual , Volume 3, Kapitel 14.9, Tabelle: 14-4)

4.2.1 Aufbau eines ESCR MSR

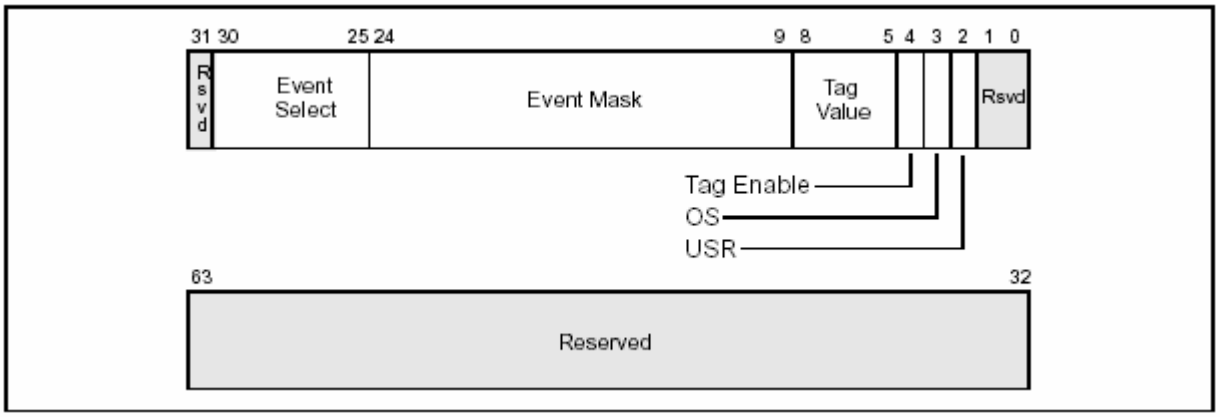

Abbildung 1: Event Selection Control Register

(Quelle: IA32 Intel Architecture Software Developer's Manual , Volume 3, Kapitel 14.9.1, Abbildung: 14-7)

USR flag, Bit 2

Wird dieses Bit gesetzt, dann werden Ereignisse gezählt, die vom Prozessor mit den Systemrechten der Level 1,2 oder 3 ausgeführt werden (current privilege level – CPL). Normalerweise laufen alle Anwendungen und harmlose Betriebssystembefehle mit diesen Leveln.

OS flag, Bit 3

Wird dieses Bit gesetzt, dann werden nur Ereignisse gezählt, die mit den Systenrechten des Level 0 ausgestattet sind; zu diesen zählen eigentlich nur Anweisungen von geschützten Betriebssystembefehlen (Kernelinstruktionen).

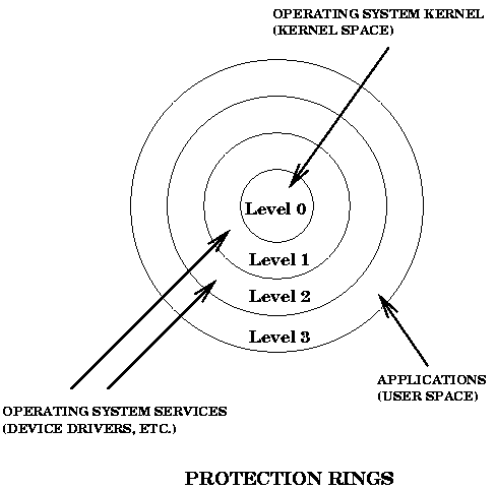

Abbildung 2: Current Privilege Level (CPL)

(*Quelle: http://qso.lanl.gov/~mpg/ames/cpl.gif*

Hinweis: Sind die Bits des USR und des OS Flag beide gesetzt, dann werden Ereignisse aller Level gezählt. Ist keines der beiden gesetzt, dann werden keine Ereignisse gezählt.

Tag Enable flag, Bit 4
Wenn dieses Bit gesetzt ist, lassen sich µops identifizieren, die beim at-retirement Ereigniszählen auftreten. Ist dieses Bit nicht gesetzt, dann können keine µops identifiziert werden.

Tag Value Feld, Bits 5 bis 8
Mit diesen Bits kann ein tag Wert gesetzt werden, der beim at-retirement Ereigniszählen mit einer µop assoziiert ist.

Event Mask Feld, Bits 9 bis 24
Mit diesen Bits wählt man Ereignisse aus, die im event select Feld stehen.

Event Select Feld, Bits 25 bis 30
Mit diesen Bits bestimmt man eine Gruppe von Ereignissen, die gezählt werden sollen, z.B. Sprunganweisungen jeglicher Art (z.B. Sprungvorhersage durchlaufen, Sprungvorhersage nicht durchlaufen, Sprungvorhersage gemacht, aber nicht durchlaufen, Sprungvorhersage gemacht und durchlaufen). Die Ereignisse, die zu dieser Gruppe gehören, werden über das event mask Feld festgelegt.

Die vorgestellten Flags und Felder können mit den WRMSR Befehlen gesetzt bzw. beschrieben werden. Die Adressen der ESCRs bekommt man aus der vollständigen Tabelle, die unter *Punkt 2.2* erwähnt wurde.

4.3 Counter Configuration Control Registers (CCCR)s

Da sich über die ESCR Flags und Felder nur die Ereignisse auswählen lassen, die gezählt werden sollen, benötigt man noch die Counter Configuration Control Register (CCCR), um den Counter auch zu aktivieren bzw. den Counter zu initialisieren. Die Überwachung des Zählvorgangs übernehmen nun die CCCR automatisch. Die folgende Tabelle zeigt einen Ausschnitt zum MSR_ITLB ESCR und gibt eine Beschreibung zu den Ereignissen, die sich überwachen lassen:

Ereignisname	Ereignis Parameter	Parameter Wert	Beschreibung
ITLB_reference			This event counts translations using the Instruction Translation Lookaside Buffer (ITLB).
	ESCR restrictions	MSR_ITLB_ESCR0 MSR_ITLB_ESCR1	
	Counter numbers per ESCR	ESCR0: 0, 1 ESCR1: 2, 3	
	ESCR Event Select	18H	ESCR[31:25]
	ESCR Event Mask	Bit 0: HIT Bit 1: MISS Bit 2: HIT_UC	ESCR[24:9], ITLB hit, ITLB miss, Uncacheable ITLB hit.
	CCCR Select	03H	CCCR[15:13]
	Event Specific Notes		All page references regardless of the page size are looked up as actual 4- KByte pages. Use the page_walk_type event with the ITMISS mask for a more conservative count.

Tabelle 2: Performance Monitoring Ereignisse für no-retirement Zählung

(Quelle: IA32 Intel Architecture Software Developer's Manual , Volume 3, Anhang, Tabelle: A.1.)

ESCR Restrictions
Zeigt eine Liste von ESCRs, die benutzt werden können, um das Ereignis zu programmieren. Normalerweise wird je Ereignis ein ESCR benötigt.

Counter numbers per ESCR
Zeigt eine Liste von Performance Countern, die mit einem ESCR verbunden sind. Normalerweise wird nur ein Counter benötigt, um die Ereignisse zu zählen.

ESCR Event Select
Gibt den Wert an, der im event select Feld des ESCR stehen muss, um das Ereignis auszuwählen.

ESCR Event Mask
Gibt den Wert an, der im event select Feld des ESCR stehen muss, um Unterereignisse auszuwählen. Die Werte in der „Parameter Wert“ Spalte definieren die dokumentierten Bits mit ihrer relativen Bit Position, wobei der offset bei 0 startet.

CCCR Select
Gibt den Wert an, der im ESCR select Feld stehen muss und ist verbunden mit dem Counter, um das ESCR auszuwählen, dessen Ereignis definiert werden soll.

Event Specific Notes
Enthält Informationen darüber, ob das gleiche oder ein ähnliches Ereignis auch für die P6 Prozessorfamilie definiert wurde.

Can Support PEBS *(Hier nicht aufgeführt)*
Zeigt an, wenn PEBS für dieses Ereignis unterstützt werden.

Requires Additional MSR for Tagging *(Hier nicht aufgeführt)*
Zeigt an, wenn zusätzliche MSRs programmiert werden müssen, um das Ereignis zu zählen.

Jeder Performance Counter ist einer festgelegte Gruppe von Ereignissen und daraus resultierend auch bestimmten ESC-Registern zugeordnet!

4.3.1 Aufbau eines CCCR

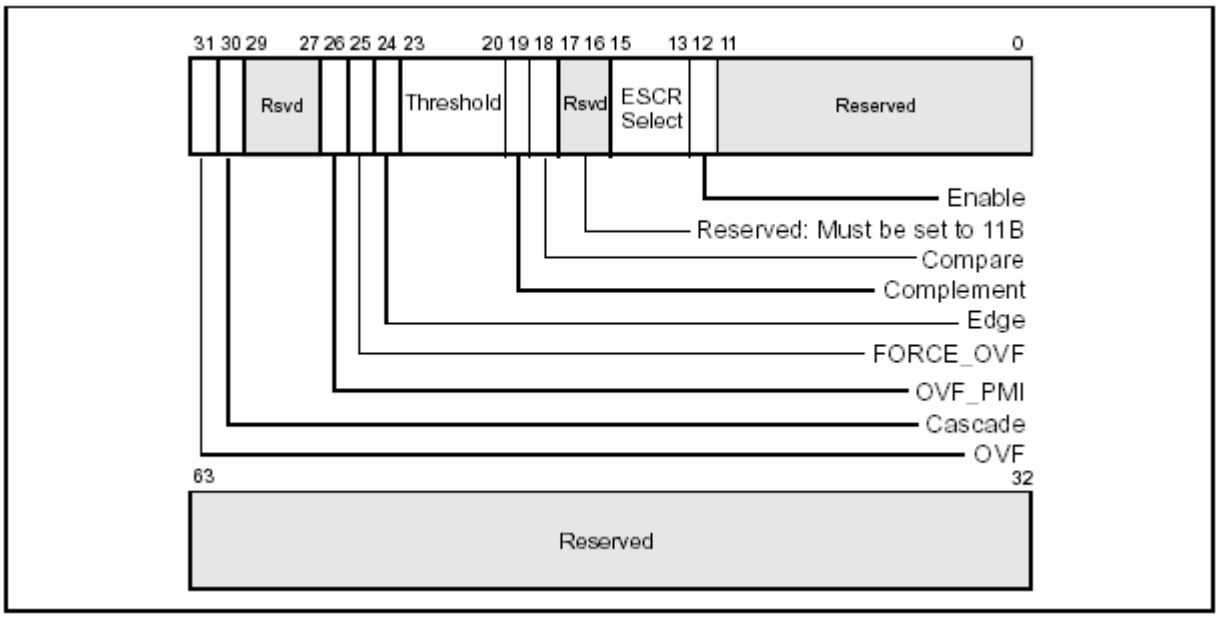

Abbildung 3: Counter Configuration Control Register (CCCR)

(Quelle: IA32 Intel Architecture Software Developer's Manual , Volume 3, Kapitel 14.9.3, Abbildung: 14-9)

Enable flag, Bit 12
Mit diesem Bit kann das Zählen ein- und ausgeschaltet werden. Bei gesetztem Bit wird gezählt. Dieses Bit wird bei reset auf 0 gesetzt.

ESCR Select Feld, Bits 13 bis 15
Kennzeichnet das ESCR in Verbindung mit dessen CCCR, das benutzt wird, um die ausgewählten Ereignisse zu zählen.

Compare flag, Bit 18 – aktiviert Filterfunktionen bei der Ereigniszählung
Ist dieses Bit gesetzt, dann ist die Filterfunktion für das Ereigniszählen aktiv. Die Art der Filtermethode kann durch Setzen der threshold, complement und edge Bits gewählt werden.

Complement flag, Bit 19 – unterstützt Filterfunktionen
Mit diesem Bit kann ausgewählt werden, wie das ankommende Ereignis mit dem Wert des threshold Bit verglichen wird. Ist es gesetzt, dann werden nur Ereignisse zum Performance Counter durchgereicht, die einen größeren Wert haben als das threshold Flag.

Threshold Feld, Bits 20 bis 23 – unterstützt Filterfunktionen
Enthält den threshold Wert, der für den Vergleich benötigt wird. Der Prozessor fragt diese Feld nur ab, wenn das compare flag (Bit 18) gesetzt ist. Ist es gesetzt, dann wird im complement flag (Bit 19) nachgesehen, auf welche Art gefiltert werden soll.

Edge flag, Bit 24 – unterstützt Filterfunktionen
Wenn es gesetzt ist, ist die Funktion rising edge detection aktiviert. Die Funktion kann nur genutzt werden, wenn das compare flag (Bit 18) gesetzt ist.

FORCE_OVF flag, Bit 25
Wenn es gesetzt ist, wird ein Counterüberlauf bei jeder Countererhöhung erzwungen.

OVF_PMI flag, Bit 26
Wenn es gesetzt ist, löst es einen Performance Monitor Interrupt (PMI) aus, wenn der Counter überläuft. Ist es nicht gesetzt, dann wird kein PMI erzeugt.

Hinweis: Ein PMI wird in jedem Fall ausgelöst, wenn nach einem Überlauf der Counter wieder mit Zählen beginnt, d.h. einen Schritt später.

Cascade flag, Bit 30
Wenn es gesetzt wird, zählt ein Counter eines Counterpaares weiter, wenn sein zugehöriger Partner Counter - der in einer anderen Gruppe liegt – überläuft.

OVF flag, Bit 31
Wenn es gesetzt ist, zeigt es einen Counterüberlauf an. Diese Bit ist ein sticky Bit und muss explizit durch die Software zurückgesetzt werden.

4.4 Debug Store (DS) Mechanismus

Der DS geschützte Bereich ist ein durch Software festgelegter Bereich im Speicher, der einzig dazu dient, Informationen folgenden Typs in ihm abzulegen.

a) Branch Records (Sprungdaten): Ist das Branch Trace Store (BTS) Flag Bit in der IA32_DEBUGCTL MSR gesetzt, so werden die Sprungdaten, die im BTS Zwischenspeicher auflaufen, in den dafür reservierten Bereich des DS verschoben bzw. abgelegt; sobald ein Sprung ausgeführt wird, ein Interrupt

oder eine Exzeption auftritt, werden diese Ereignisse im DS Bereich abgespeichert.

b) PEBS Records: Ist ein Performance Counter so konfiguriert, dass er Precise Event-Based Sampling (PEBS) Daten überwacht und sammelt, dann werden diese auch im DS Bereich abgespeichert. PEBS - Datensätze entstehen, wenn ein Counterüberlauf stattfindet. In dem Datensatz sind die Zustände der acht Allzweck Register (general purpose registers), des EIP Registers und des EFLAGS Registers zum Zeitpunkt des Überlaufs enthalten. Diese Funktion ist jedoch nur für einen kleinen Teil der Perfomance Event Aufzeichnung im P4 Prozessor verfügbar.

5 Performance Counter

Jeder der 18 Performance Counter ist 40 Bit groß. Für den P4 Prozessor wurde der Read Performance Monitoring Counter (RDPMC) Befehl eingeführt. Über diesen kann man auswählen, ob die ganzen 40 Bit ausgelesen werden sollen oder nur die ersten 32 Bit. Es ist selbstverständlich nur sinnvoll, die ersten 32 Bit auszulesen, wenn man sich sicher ist, dass im Counter keine Zahl größer 2^{32} steht.

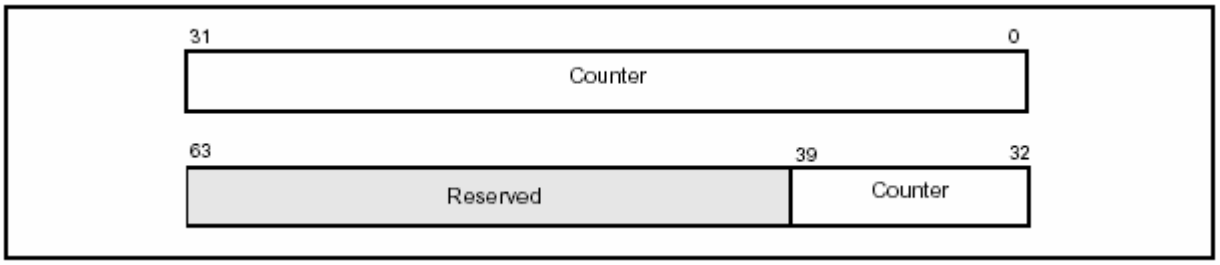

Abbildung 4: Performance Counter

(Quelle: IA32 Intel Architecture Software Developer's Manual , Volume 3, Kapitel 14.9.2, Abbildung: 14-8)

Um die verschiedenen Ereignisse, die Einfluss auf die Leistung haben, zu zählen, gibt es folgende Einrichtungen im P4 Prozessor:

- Die IA32_MISC_ENABLE MSR; diese Einrichtung zeigt die Verfügbarkeit des performance monitoring und der precise event-based sampling (PEBS) in einem IA-32 Prozessor an.
- Die 45 ESCR MSRs, um die Ereignisse auszuwählen, die von den speziellen performance countern gezählt werden sollen.
- Die 18 performance counter model specific registers, um die Ereignisse zu zählen, die ausgewählt wurden.
- Der Debug Store (DS), der in einem geschützten Bereich des Speichers liegt und die PEBS Datensätze und Sprungdaten (branch records) speichert.
- Die IA32_DS_AREA MSR; diese Einrichtung stellt die Verbindung zum DS geschützten Bereich her.
- Das DS Funktionsflag (Bit 21), das durch die CPUID Anweisung ausgegeben wird. Dieses zeigt die Verfügbarkeit des DS Mechanismus in einem IA-32 Prozessor an.
- Das IA32_PEBS_ENABLE MSR; dieses stellt die PEBS Einrichtung und das wiederholte Identifizieren, das bei der at-retirement Ereigniszählung genutzt wird, zur Verfügung.

- Ein Satz vordefinierter Ereignisse und Ereignismetriken, die zur Vereinfachung bei der Aktivierung und Bedienung spezieller Performance Counter Ereignisse vorgesehen sind.

Die 18 Performance Counter lassen sich zu neun Paaren zusammenlegen, die dann in vier Gruppen eingeteilt werden können:

I. Die BPU Gruppe:
   1. MSR_BPU_COUNTER0 und MSR_BPU_COUNTER1
   2. MSR_BPU_COUNTER2 und MSR_BPU_COUNTER3.

II. Die MS Gruppe:
   1. MSR_MS_COUNTER0 und MSR_MS_COUNTER1.
   2. MSR_MS_COUNTER2 und MSR_MS_COUNTER3.

III. Die FLAME Gruppe:
   1. MSR_FLAME_COUNTER0 und MSR_FLAME_COUNTER1.
   2. MSR_FLAME_COUNTER2 und MSR_FLAME_COUNTER3.

IV. Die IQ Gruppe:
   1. MSR_IQ_COUNTER0 und MSR_IQ_COUNTER1.
   2. MSR_IQ_COUNTER2 und MSR_IQ_COUNTER3.
   3. MSR_IQ_COUNTER4 und MSR_IQ_COUNTER5.

Die Counter der MSR_IQ_COUNTER4 bieten die Unterstützung zur Bearbeitung der PEBS an.

6 Benutzermodelle

Im P4 sind drei verschiedene Benutzermodelle für die Counter Programmierung vorgesehen. Die ersten beiden Modelle werden eingesetzt, um no-retirement und at-retirement Ereignisse zu zählen. Das dritte Modell ist nur zum Zählen von at-retirement Ereignissen vorgesehen.

6.1 Event Counting

Die Leistungsüberwachung durch einen Performance Counter ist so konfiguriert, dass er ein oder mehrere unterschiedliche Ereignisse überwacht und zählt. Der Zähler wird in festgelegten Zeitabständen von einer Software ausgelesen, jetzt kann die Anzahl der Ereignisse, die zwischen zwei Auslesevorgängen liegt, angezeigt und bewertet werden.

6.2 Non-precise event –based sampling

Die Leistungsüberwachung durch einen Performance Counter ist so konfiguriert, dass er ein oder mehrere unterschiedliche Ereignisse überwacht und bei einem Überlauf ein Interrupt generiert. Um den Interrupt auszulösen, ist der Counter auf einen bestimmten Modulus Wert eingestellt, bei dem der Counter nach einer definierten Anzahl von aufgetretenen Ereignissen überläuft. Wenn der Counter nun überläuft, wird vom Prozessor ein Performance Monitoring Interrupt (PMI) erzeugt. Die Interrupt Service Routine für den PMI läßt die Ereignisse durch den Return Instruktion Pointer (RIP) aufzeichnen, setzt den Zähler zurück und alles beginnt von neuem. Von Intel gibt es eine Software (VTune), mit der sich RIPs analysieren lassen.

6.3 Precise event-based sampling (PEBS)

Diese Art der Leistungsüberwachung ist vergleichbar mit dem Non-precise event – based sampling. Der Unterschied besteht in der Art der Speicherung bzw. Behandlung beim Überlauf eines Counters. Bei diesem Modell wird nicht mit RIPs gearbeitet, sondern ein Datensatz mit dem Zustand des Prozessors im Arbeitsspeicher abgelegt, wenn ein Counter überläuft. Diese Art der Leistungsüberwachung lässt sich nur für eine kleine Gruppe von at-retirement Ereignissen einsetzen.

7 Programmierung der Performance Counter

Um einen Performance Counter so zu programmieren, dass dieser Ereignisse zählt, muss die Software die folgenden Operationen unterstützen:

- Ein oder mehrere Ereignisse, die gezählt werden sollen, müssen auswählbar sein.
- Der Performance Counter, mit dem die Ereignisse gezählt werden sollen, muss auszuwählen sein und der damit verbundene ESCR, über den die Ereignisse ausgewählt werden, muss zur Verfügung stehen.
- Der ESCR muss so eingestellt werden, dass er die Ereignisse mit den entsprechenden Systemrechte-Leveln zählt.
- Die CCCR für den Performance Counter müssen eingestellt werden, um die Ereignisse mittels des ausgewählten ESCR und der gewünschten Ereignis-Filter zu zählen.
- Der CCCR sollte optional so eingestellt sein, dass er weiterzählt, wenn der verantwortliche Performance Counter überläuft.
- Der CCCR sollte so eingestellt sein, dass dieser einen Performance Monitor Interrupt (PMI) auslöst, wenn der Counter überläuft.
- Der Counters muss aktiviert sein, um mit dem Zählen zu beginnen.

Die Performance Counter Programmierung kann einmal für die no-retirement Ereignisse und einmal für die at-retirement Ereignisse vorgenommen werden.

7.1 no-retirement Ereignisse

Diese Ereignisse treten zu jeder Zeit während der Befehlsausführung auf. Zu ihnen gehören z.B. Sprungbefehle, Bus Transaktionen oder Cache Transaktionen.
Um die Auswahl der Ereignisse zu vereinfachen, wurden für den P4 Prozessor einige Ereignisse vordefiniert. Ein Beispiel für die Art, in der diese Ereignisse dokumentiert sind und wie auf diese zugegriffen wird, findet man unter *Punkt 2.3.*
Mit den folgenden Schritten lässt sich ein Performance Counter mit seinen Basisfunktionen für das no-retirement Counting einstellen:

Schritt 1: Wähle das Ereignis aus, das gezählt werden soll.

Schritt 2: Wähle über das „ESCR Name“ Feld das ESCR aus, das benutzt wird, um die ausgewählten Ereignisse zu zählen.

Schritt 3: Wähle im „Counter Numbers Per ESCR“ Feld die Nummer des Counters aus, der für das Zählen benutzt werden soll.

Schritt 4: Bestimme damit den Namen des Counters und des damit verbundenen CCCR. Wähle mit den Informationen die MSR Adresse des Counters (siehe Tabelle unter Punkt 2.2).

Schritt 5: Benutze den WRMSR Befehl, um den „ESCR Event Select“ und den „ESCR Event Mask“ Wert in die dafür vorgesehenen Felder des ESCR zu schreiben (siehe

Tabelle unter Punkt 2.3). Zu diesem Zeitpunkt kann das USR und das OS Feld im ESCR gesetzt oder gelöscht werden.

Schritt 6: Benutze den WRMSR Befehl, um den „CCCR Select“ Wert in das dafür vorgesehene Feld des CCCR zu schreiben (siehe Tabelle unter Punkt 2.3).

Schritt 7: (Optional) Um den Counter für das threshold filtern einzustellen, müssen mit dem WRMSR Befehl Werte in die folgenden Bit-Felder geschrieben werden: CCCR compare, complement und in das threshold Feld.

Schritt 8: Um mit dem Ereigniszählen zu beginnen, muss mit dem WRMSR Befehl das „CCCR enable“ Bit für den Performance Counter gesetzt werden.

Schritt 9: Um den aktuellen Zählerstand eines Performance Counters auszulesen, muss dem entsprechenden Lesebefehl (RDPMC) bei Ausführung die auszulesende Counter Nummer als Parameter übergeben werden.

Schritt 10: Um das Ereigniszählen zu beenden, muss mit dem WRMSR Befehl das „CCCR enable“ Bit gelöscht werden.

7.2 at-retirement Ereignisse

Der at-retirement Mechanismus wertet nur tatsächlich durchgeführte Ereignisse aus. Der P4 Prozessor unterstützt z.B. Sprungvorhersagen, d.h. nach einem bestimmten Algorithmus und anhand einer Sprungvorhersagentabelle versucht der Prozessor vorherzusagen, ob ein bedingter Sprung ausgeführt wird oder nicht. Die Abarbeitung der Befehle kann dann schon spekulativ fortgesetzt werden. Bei der Ereigniszählung würde der verantwortliche Performance Counter auch Pfade mitzählen, die letztendlich gar nicht ausgeführt wurden, z.B. durch falsche Sprungvorhersagen, die dann wieder gelöscht wurden.
Der at-retirement Mechanismus verhindert solche Fehlinterpretationen und zählt nur die tatsächlich ausgeführten Instruktionen, d.h. die tatsächliche Arbeit, die vom Prozessor verrichtet wurde und nicht zusätzlich noch dessen spekulative Arbeit.

Um die Art der at-retirement Ereigniszählung zu beschreiben, gibt es vier verschiedene Fachbereiche. Diese werden zur Beschreibung der unterschiedlichen Arten der Ereigniszählung benötigt, sozusagen zur Feinabstimmung der Zählmechanismen:

- Bogus, Non-Bogus, Retire
  Der Begriff “bogus” bezeichnet diejenigen Instruktionen, die nicht mitgezählt werden dürfen, z.B. aufgrund falscher Vorhersagen. Die Begriffe „non-bogus“ und „retire“ bezeichnen Instuktionen oder µops, die während der Programmausführung auftreten und Einfluss auf den Zustand des Prozessors haben.
- Tagging
  Beim “tagging” werden µops identifiziert und dann gezählt, die auch wirklich vollständig ausgeführt wurden. Während der Instruktionsausführung kann das gleiche Ereignis mehr als einmal auftreten und damit auch die gleiche µops Instruktion. Das „tagging“ verhindert, dass beim gleichen Ereignis die µops nicht doppelt gezählt werden und somit ein verfälschtes Ergebnis erzeugt wird.
- Replay
  Um die Leistung zu erhöhen, werden die µops bereits durch die Intel NetBurst Architektur bereitgestellt, bevor überhaupt feststeht, dass alle Bedingungen erfüllt wurden, um diese Instruktionen auszuführen. Wenn nun diese µops zurückgezogen werden müssen, um neu zugeteilt zu werden, dann wird das „replay“ genannt.

- Assist
  Wenn die Hardware eine Unterstützung durch Mikrocode benötigt, um bestimmte Ereignisse auszuführen, dann benötigt die Maschine einen „Assist“.

Das at-retirement ermöglicht die Zählung von Ereignissen und der damit im Zusammenhang auftretenden µops.

8 Motivation

Die Firma Intel hat die MSRs eingeführt, um den Benutzern oder besser den Personen, die Pentium Prozessoren einsetzen, eine bessere Kontrolle bei der Feinabstimmung und Leistungsoptimierung ihrer Systeme zu geben. Das Betriebssystem Windows 2000 verfügt in der Standardinstallation bereits über das Program **perfmon,** mit dem sich eine Vielzahl der erwähnten Hardwarecounter auslesen und ihre Aufzeichnungsergebnisse in aufbereiteter Form darstellen lassen. Neben den Standardtools lassen sich im Internet noch einige Kommerzielle (z.B. **VTune** von Intel) und Nichtkommerzielle (z.B. **pperf** - http://qso.lanl.gov/~mpg/perfmon.html#what) Produkte finden.

9 Literaturverzeichnis

[1] Microsoft, N.N.:
IA32 Intel Architecture Software Developer's Manual , Volume 3, Kapitel 14.7-14.11, Anhang A
ftp://download.intel.com/design/pentium4/manuals/24547204.pdf

[2] http://www.heise.de/ct/98/01/021/

[3] http://qso.lanl.gov/~mpg/ames/main.html

[4] http://www.sandpile.org/ia32/

[5] http://developer.intel.com/design/intarch/techinfo/pentium/mdelregs.htm

[6] http://linux-patches.rock-projects.com/v2.2-f/msr.html

10 Anhang

A-1: Performance Monitoring Ereignisse für Non-Retirement Counting

Event Name	Event Parameters	Parameter Value	Description
branch_retired			This event counts the retirement of a branch. Specify one or more mask bits to select any combination of taken, not-taken, predicted and mispredicted.
	ESCR restrictions	MSR_CRU_ESCR2 MSR_CRU_ESCR3	See Table 14-4 for the addresses of the ESCR MSRs
	Counter numbers per ESCR	ESCR2: 12, 13, 16 ESCR3: 14, 15, 17	The counter numbers associated with each ESCR are provided. The performance counters and corresponding CCCRs can be obtained from Table 14-4.
	ESCR Event Select	06H	ESCR[31:25]
	ESCR Event Mask	Bit 0: MMNP 1: MMNM 2: MMTP 3: MMTM	ESCR[24:9], Branch Not-taken Predicted, Branch Not-taken Mispredicted, Branch Taken Predicted, Branch Taken Mispredicted.
	CCCR Select	05H	CCCR[15:13]
	Event Specific Notes		P6: EMON_BR_INST_RETIRED
mispred_branch_ retired			This event represents the retirement of mispredicted IA-32 branch instructions.
	ESCR restrictions	MSR_CRU_ESCR0 MSR_CRU_ESCR1	
	Counter numbers per ESCR	ESCR0: 12, 13, 16 ESCR1: 14, 15, 17	
	ESCR Event Select	03H	ESCR[31:25]
	ESCR Event Mask	Bit 0: NBOGUS	ESCR[24:9] The retired instruction is not bogus
	CCCR Select	04H	CCCR[15:13]
	Event Specific Notes		
TC_deliver_mode			This event counts the duration (in clock cycles) of the trace cache operating modes. The mode is specified by one or more of the event mask bits.
	ESCR restrictions	MSR_TC_ESCR0 MSR_TC_ESCR1	
	Counter numbers per ESCR	ESCR0: 4, 5 ESCR1: 6, 7	
	ESCR Event Select	01H	ESCR[31:25]
	ESCR Event Mask	Bit 2: DELIVER 5: BUILD	ESCR[24:9], TC is delivering traces, TC is building traces while decoding instructions.

Event Name	Event Parameters	Parameter Value	Description
	CCCR Select	01H	CCCR[15:13]
	Event Specific Notes		
BPU_fetch_ request			This event counts instruction fetch requests of specified request type by the Branch Prediction unit. Specify one or more mask bits to qualify the request type(s).
	ESCR restrictions	MSR_BPU_ESCR0 MSR_BPU_ESCR1	
	Counter numbers per ESCR	ESCR0: 0, 1 ESCR1: 2, 3	
	ESCR Event Select	03H	ESCR[31:25]
	ESCR Event Mask	Bit 0: TCMISS	ESCR[24:9], Trace cache lookup miss.
	CCCR Select	00H	CCCR[15:13]
	Event Specific Notes		
ITLB_reference			This event counts translations using the Instruction Translation Look-aside Buffer (ITLB).
	ESCR restrictions	MSR_ITLB_ESCR0 MSR_ITLB_ESCR1	
	Counter numbers per ESCR	ESCR0: 0, 1 ESCR1: 2, 3	
	ESCR Event Select	18H	ESCR[31:25]
	ESCR Event Mask	Bit 0: HIT 1: MISS 2: HIT_UC	ESCR[24:9], ITLB hit, ITLB miss, Uncacheable ITLB hit.
	CCCR Select	03H	CCCR[15:13]
	Event Specific Notes		All page references regardless of the page size are looked up as actual 4-KByte pages. Use the page_walk_type event with the ITMISS mask for a more conservative count.
memory_cancel			This event counts the canceling of various type of request in the Data cache Address Control unit (DAC). Specify one or more mask bits to select the type of requests that are canceled.
	ESCR restrictions	MSR_DAC_ESCR0 MSR_DAC_ESCR1	
	Counter numbers per ESCR	ESCR0: 8, 9 ESCR1: 10, 11	
	ESCR Event Select	02H	ESCR[31:25]

Event Name	Event Parameters	Parameter Value	Description
	ESCR Event Mask	Bit 2: ST_RB_FULL 3: 64K_CONF	ESCR[24:9], Replayed because no store request buffer is available. Conflicts due to 64K aliasing.
	CCCR Select	05H	CCCR[15:13]
	Event Specific Notes		Note: All_CACHE_MISS will include uncacheable memory in its count.
memory_complete			This event counts the completion of a load split, store split, uncacheable (UC) split, or UC load. Specify one or more mask bits to select the operations to be counted.
	ESCR restrictions	MSR_SAAT_ESCR0 MSR_SAAT_ESCR1	
	Counter numbers per ESCR	ESCR0: 8, 9 ESCR1: 10, 11	
	ESCR Event Select	08H	ESCR[31:25]
	ESCR Event Mask	Bit 0: LSC 1: SSC	ESCR[24:9], Load split completed, excluding UC/WC loads Any split stores completed
	CCCR Select	02H	CCCR[15:13]
	Event Specific Notes		
load_port_replay			This event counts replayed events at the load port. Specify one or more mask bits to select the cause of the replay.
	ESCR restrictions	MSR_SAAT_ESCR0 MSR_SAAT_ESCR1	
	Counter numbers per ESCR	ESCR0: 8, 9 ESCR1: 10, 11	
	ESCR Event Select	04H	ESCR[31:25]
	ESCR Event Mask	Bit 1: SPLIT_LD	ESCR[24:9], Split load.
	CCCR Select	02H	CCCR[15:13]
	Event Specific Notes		Must use ESCR1 for at-retirement counting.
store_port_replay			This event counts replayed events at the store port. Specify one or more mask bits to select the cause of the replay.

Event Name	Event Parameters	Parameter Value	Description
	ESCR restrictions	MSR_SAAT_ESCR0 MSR_SAAT_ESCR1	
	Counter numbers per ESCR	ESCR0: 8, 9 ESCR1: 10, 11	
	ESCR Event Select	05H	ESCR[31:25]
	ESCR Event Mask	Bit 1: SPLIT_ST	ESCR[24:9], Split store
	CCCR Select	02H	CCCR[15:13]
	Event Specific Notes		Must use ESCR1 for at-retirement counting.
MOB_load_replay			This event triggers if the memory order buffer (MOB) caused a load operation to be replayed. Specify one or more mask bits to select the cause of the replay.
	ESCR restrictions	MSR_MOB_ESCR0 MSR_MOB_ESCR1	
	Counter numbers per ESCR	ESCR0: 0, 1 ESCR1: 2, 3	
	ESCR Event Select	03H	ESCR[31:25]
	ESCR Event Mask	Bit 1: NO_STA 3: NO_STD 4: PARTIAL_DATA 5: UNALGN_ADDR	ESCR[24:9], Replayed because of unknown store address, Replayed because of unknown store data, Replayed because of partially overlapped data access between the load and store operations, Replayed because the lower 4 bits of the linear address do not match between the load and store operations.
	CCCR Select	02H	CCCR[15:13]
	Event Specific Notes		
page_walk_type			This event counts various types of page walks that the page miss handler (PMH) performs.
	ESCR restrictions	PMH_CR_ESCR0 PMH_CR_ESCR1	
	Counter numbers per ESCR	ESCR0: 0, 1 ESCR1: 2, 3	
	ESCR Event Select	01H	ESCR[31:25]

Event Name	Event Parameters	Parameter Value	Description
	ESCR Event Mask	Bit 0: DTMISS 1: ITMISS	ESCR[24:9], Page walk for a data TLB miss (either load or store). Page walk for an instruction TLB miss.
	CCCR Select	04H	CCCR[15:13]
	Event Specific Notes		
BSQ_cache _reference			This event counts 2nd level cache references as seen by the bus unit. Specify one or more mask bits to select whether a reference is counted based on the access type (load, store) and the access result (hit, miss).
	ESCR restrictions	BSU_CR_ESCR0 BSU_CR_ESCR1	
	Counter numbers per ESCR	ESCR0: 0, 1 ESCR1: 2, 3	
	ESCR Event Select	0CH	ESCR[31:25]
	ESCR Event Mask	0: RD_2ndL_HITS 1: RD_2ndL_HITE 2: RD_2ndL_HITM 8: RD_2ndL_MISS 10: WR_2ndL_MISS	ESCR[24:9], Read 2nd level cache hit Shared Read 2nd level cache hit Exclusive Read 2nd level cache hit Modified Read 2nd level cache miss Write 2nd level cache miss
	CCCR Select	07H	CCCR[15:13]
	Event Specific Notes		Specify edge trigger in CCCR MSR to avoid double counting. This event is known to both overcount and undercount by as much as a factor of two.

Tabelle 3: Performance Monitoring Ereignisse der Pentium 4 Prozessoren

(Quelle: IA32 Intel Architecture Software Developer's Manual , Volume 3, Anhang A)

A-2: Beschreibung der Model Specific Register

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
0H	0	IA32_P5_MC_ADDR	See Section B.3., "MSRs in Pentium Processors".
1H	1	IA32_P5_MC_TYPE	See Section B.3., "MSRs in Pentium Processors".
10H	16	IA32_TIME_STAMP_ COUNTER 63:0	Time Stamp Counter. See Section 14.7., "Time-Stamp Counter" **Timestamp Count Value.** (R/W) Returns the current time stamp count value. All 64 bits are readable; only the lower 32 bits are writable. On any write to the lower 32 bits, the upper 32 bits are cleared.
17H	23	IA32_PLATFORM_ID 49:0	Platform ID. (R) The operating system can use this MSR to determine "slot" information for the processor and the proper microcode update to load. **Reserved.**

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
		52:50	**Platform Id.** (R) Contains information concerning the intended platform for the processor. 52 51 50 0 0 0 Processor Flag 0 0 0 1 Processor Flag 1 0 1 0 Processor Flag 2 0 1 1 Processor Flag 3 1 0 0 Processor Flag 4 1 0 1 Processor Flag 5 1 1 0 Processor Flag 6 1 1 1 Processor Flag 7
		63:53	**Reserved.**
1BH	27	IA32_APIC_BASE	APIC Location and Status. (R/W) Contains location and status information about the APIC (see Section 7.6.7., "Local APIC Status and Location")
		7:0	**Reserved.**
		8	**Bootstrap Processor (BSP).** Set if the processor is the BSP.
		10:9	**Reserved.**
		11	**APIC Global Enable.** Set if enabled; cleared if disabled.
		31:12	**APIC Base Address.** The base address of the xAPIC memory map.
		63:32	**Reserved.**
2AH	42	MSR_EBC_HARD_ POWERON	Processor Hard Power-On Configuration. (R/W) Enables and disables processor features; (R) indicates current processor configuration.
		0	**Output Tri-state Enabled.** (R) Indicates whether tri-state output is enabled (1) or disabled (0) as set by the strapping of SMI#. The value in this bit is written on the deassertion of RESET#; the bit is set to 1 when the address bus signal is asserted.
		1	**Execute BIST.** (R) Indicates whether the execution of the BIST is enabled (1) or disabled (0) as set by the strapping of INIT#. The value in this bit is written on the deassertion of RESET#; the bit is set to 1 when the address bus signal is asserted.
		2	**In Order Queue Depth.** (R) Indicates whether the in order queue depth for the system bus is 1 (1) or up to 12 (0) as set by the strapping of A7#. The value in this bit is written on the deassertion of RESET#; the bit is set to 1 when the address bus signal is asserted.

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
		3	**MCERR# Observation Disabled.** (R) Indicates whether MCERR# observation is enabled (0) or disabled (1) as set by the strapping of A9#. The value in this bit is written on the deassertion of RESET#; the bit is set to 1 when the address bus signal is asserted.
		4	**BINIT# Observation Enabled.** (R) Indicates whether BINIT# observation is enabled (0) or disabled (1) as set by the strapping of A10#. The value in this bit is written on the deassertion of RESET#; the bit is set to 1 when the address bus signal is asserted.
		6:5	**APIC Cluster ID.** (R) Contains the logical APIC cluster ID value as set by the strapping of A12# and A11#. The logical cluster ID value is written into the field on the deassertion of RESET#; the field is set to 1 when the address bus signal is asserted.
		7	**Bus Park Disable.** (R) Indicates whether bus park is enabled (0) or disabled (1) as set by the strapping of A15#. The value in this bit is written on the deassertion of RESET#; the bit is set to 1 when the address bus signal is asserted.
		11:8	**Reserved.**
		13:12	**Agent ID.** (R) Contains the logical agent ID value as set by the strapping of BR[3:0]. The logical ID value is written into the field on the deassertion of RESET#; the field is set to 1 when the address bus signal is asserted.
		63:14	**Reserved.**
2BH	432	MSR_EBC_SOFT_ POWERON	Processor Soft Power-On Configuration. (R/W) Enables and disables processor features.
		0	**RCNT/SCNT On Request Encoding Enable.** (R/W) Controls the driving of RCNT/SCNT on the request encoding. Set to enabled; clear to disabled (default).
		1	**Data Error Checking Disable.** (R/W) Set to disable system data bus parity checking (default); clear to enable parity checking.
		2	**Response Error Checking Disable.** (R/W) Set to disable (default); clear to enable.
		3	**Address/Request Error Checking Disable.** (R/W) Set to disable (default); clear to enable.
		4	**Initiator MCERR# Disable.** (R/W) Set to disable MCERR# driving for initiator bus requests (default); clear to disable.
		5	**Internal MCERR# Disable.** (R/W) Set to disable MCERR# driving for initiator internal errors (default); clear to enable.

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
		6	**BINIT# Driver Disable.** (R/W) Set to disable BINIT# driver (default); clear to enable driver.
		63:7	**Reserved.**
2CH	44	MSR_EBC_ FREQUENCY_ID	Processor Frequency Configuration. (R) Indicates current processor frequency configuration.
		20:0	**Reserved.**
		23:21	**Scaleable Bus Speed.** (R/W) Indicates the intended scaleable bus speed: Encoding — Scaleable Bus Speed 000B — 100 MHz All Others — Reserved
		63:24	**Reserved.**
79H	121	IA32_BIOS_UPDT_ TRIG	**BIOS Update Trigger Register.** (R/W Triggers the loading of a microcode update. Executing a WRMSR instruction to this MSR causes a microcode update to be loaded into the processor (see Section 8.11.2.1., "Update Loading Procedure").
8BH	139	IA32_BIOS_SIGN_ID	**BIOS Update Signature ID.** (R/W) Returns the microcode update signature, following the execution of a CPUID instruction with EAX set to 1.
		31:0	**Reserved.**
		63:32	**Microcode Update Signature.** (R/W) It is recommended that this field be pre-loaded with a 0 prior to executing the CPUID instruction. If the field remains 0 following the execution of the CPUID instruction, then there is no microcode update loaded. Any other non-0 value is the microcode update signature.
FEH	254	IA32_MTRRCAP	MTRR Information. See Section 9.11.1., "MTRR Feature Identification".
119H	281	IA32_MISC_CTL	**Processor Serial Number.** (R/W) Disables the processor's serial number feature.
		20:0	**Reserved.**
		21	**Processor Serial Number Disable.** Disables processor serial number feature when set; enables it when clear (default). This bit is a set-once bit, meaning that once set it cannot be cleared except by the assertion of the RESET# signal or the removal of processor power. In processors that support the processor serial number feature (see the feature information returned from the CPUID instruction), this bit is allows the BIOS to disable the processor serial number feature.
		63:22	**Reserved.**

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
174H	372	IA32_SYSENTER_CS	**CS register target for CPL 0 code.** (R/W) Used by SYSENTER and SYSEXIT instructions (see Section 4.8.7., "Performing Fast Calls to System Procedures with the SYSENTER and SYSEXIT Instructions").
175H	373	IA32_SYSENTER_ESP	**Stack pointer for CPL 0 stack.** (R/W) Used by SYSENTER and SYSEXIT instructions (see Section 4.8.7., "Performing Fast Calls to System Procedures with the SYSENTER and SYSEXIT Instructions").
176H	374	IA32_SYSENTER_EIP	**CPL 0 code entry point.** (R/W) Used by SYSENTER and SYSEXIT instructions (see Section 4.8.7., "Performing Fast Calls to System Procedures with the SYSENTER and SYSEXIT Instructions").
179H	377	IA32_MCG_CAP	Machine Check Capabilities. (R) Returns the capabilities of the machine check architecture for the processor (see Section 13.3.1.1., "IA32_MCG_CAP MSR (Pentium 4 and Intel Xeon Processors)").
17AH	378	IA32_STATUS	Machine Check Status. (R) Returns machine check state following the generation of a machine check exception (see Section 13.3.1.3., "IA32_MCG_STATUS MSR").
17BH	379	IA32_CTL	Machine Check Feature Enable. (R/W) Enables machine check capability (see Section 13.3.1.4., "IA32_MCG_CTL MSR").
180H	384	IA32_MCG_EAX	Machine Check EAX Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**EAX Register Contents.** (R/W) Contains the state of the EAX register at the time of the last machine check error.
		63:32	**Reserved.**
181H	385	IA32_MCG_EBX	Machine Check EBX Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**EBX Register Contents.** (R/W) Contains the state of the EBX register at the time of the last machine check error.
		63:32	**Reserved.**
182H	386	IA32_MCG_ECX	Machine Check ECX Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**ECX Register Contents.** (R/W) Contains the state of the ECX register at the time of the last machine check error.
		63:32	Reserved.
183H	387	IA32_MCG_EDX	Machine Check EDX Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**EDX Register Contents.** (R/W) Contains the state of the EDX register at the time of the last machine check error.

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
		63:32	**Reserved.**
184H	388	IA32_MCG_ESI	Machine Check ESI Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**ESI Register Contents.** (R/W) Contains the state of the EDI register at the time of the last machine check error.
		63:32	**Reserved.**
185H	389	IA32_MCG_EDI	Machine Check EDI Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**EDI Register Contents.** (R/W) Contains the state of the EDI register at the time of the last machine check error.
		63:32	**Reserved.**
186H	390	IA32_MCG_EBP	Machine Check EBP Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**EBP Register Contents.** (R/W) Contains the state of the EBP register at the time of the last machine check error.
		63:32	**Reserved.**
187H	391	IA32_MCG_ESP	Machine Check ESP Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**ESP Register Contents.** (R/W) Contains the state of the ESP register at the time of the last machine check error.
		63:32	**Reserved.**
188H	392	IA32_MCG_EFLAGS	Machine Check EFLAGS Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**EFLAGS Register Contents.** (R/W) Contains the state of the EFLAGS register at the time of the last machine check error.
		63:32	**Reserved.**
189H	393	IA32_MCG_EIP	Machine Check EIP Save State. See Section 13.3.2.5., "IA32_MCG Extended Machine Check State MSRs".
		31:0	**EIP Register Contents.** (R/W) Contains the state of the EIP register at the time of the last machine check error.
		63:32	**Reserved.**

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
18AH	394	IA32_MCG_MISC	Machine Check Miscellaneous. See Section 13.3.1.5., "IA32_MCG_MISC MSR".
		0	**DS.** When set, indicates that a page assist or page fault occurred during DS normal operation. The processor will shutdown under these circumstance. This bit is set as an aid for debugging the DS handling code. It is the responsibility of the user (BIOS or operating system) to clear this bit for normal operation.
		63:1	**Reserved.**
19AH	410	IA32_TERM_ CONTROL	Thermal Monitor Control. (R/W) Enables and disables on-demand clock modulation and allows selection of the on-demand clock modulation duty cycle. (See Section 12.14.3., "Software Controlled Clock Modulation".)
19BH	411	IA32_TERM_ INTERRUPT	Thermal Interrupt Control. (R/W) Enables and disables the generation of an interrupt on temperature transitions detected with the processor's thermal sensor and thermal monitor. (See Section 12.14.2., "Automatic Thermal Monitor".)
19CH	412	IA32_TERM_STATUS	Thermal Monitor Status. (R/W) Contains status information about the processor's thermal sensor and automatic thermal monitoring facilities. (See Section 12.14.2., "Automatic Thermal Monitor".)
1A0H	416	IA32_MISC_ENABLE	Enable Miscellaneous Processor Features. (R/W) Allows a variety of processor functions to be enabled and disabled.
		0	**Fast-Strings Enable.** When set, the fast-strings feature on the Pentium 4 processor is enabled; when clear (default), fast-strings are disabled.
		1	**Reserved.**
		2	**x87 FPU Fopcode Compatibility Mode Enable.** When set, fopcode compatibility mode is enabled; when clear (default), mode is enabled. See "Fopcode Compatibility Mode" in Chapter 8 of the *IA-32 Intel Architecture Software Developer's Manual, Volume 1*.
		3	**Thermal Monitor Enable.** When set, clock modulation controlled by the processor's internal thermal sensor is enabled; when clear (default), automatic clock modulation is disabled. (See Section 12.14.2., "Automatic Thermal Monitor".)
		4	**Split-Lock Disable.** When set, the split-lock feature of the Pentium 4 processor is disabled; when clear (default), split-lock feature is enabled.
		6:5	**Reserved.**

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
		7	**Performance Monitoring Available.** (R) When set, performance monitoring is enabled; when clear, performance monitoring is disabled.
		10:8	**Reserved.**
		11	**Branch Trace Storage Unavailable (BTS_UNAVILABLE).** (R) When set, the processor does not support branch trace storage (BTS); when clear, BTS is supported.
		12	**Precise Event Based Sampling Unavailable (PEBS_UNAVILABLE).** (R) When set, the processor does not support precise event-based sampling (PEBS); when clear, PEBS is supported.
		63:13	**Reserved.**
1D7H	471	MSR_LER_FROM_LIP	Last Exception Record From Linear IP. (R) Contains a pointer to the last branch instruction that the processor executed prior to the last exception that was generated or the last interrupt that was handled. (See Section 14.5.6., "Last Exception Records (Pentium 4 and Intel Xeon Processors)".)
		31:0	**From Linear IP**: Linear address last branch instruction.
		63:32	**Reserved.**
1D8H	472	MSR_LER_TO_LIP	Last Exception Record To Linear IP. (R) Contains a pointer to the target of the last branch instruction that the processor executed prior to the last exception that was generated or the last interrupt that was handled. (See Section 14.5.6., "Last Exception Records (Pentium 4 and Intel Xeon Processors)".)
		31:0	**From Linear IP**: Linear address of the target of the last branch instruction.
		63:32	**Reserved.**
1D9H	473	IA32_DEBUGCTL	Debug Control. (R/W) Controls how several debug features are used. See Section 14.5.1., "IA32_DEBUGCTL MSR (Pentium 4 and Intel Xeon Processors)".
1DAH	474	MSR_LASTBRANCH _TOS	Last Branch Record Stack TOS. (R) Contains an index (0, 1, 2, or 3) that points to the top of the last branch record stack (that is, that points the index of the MSR containing the most recent branch record. See Section 14.5.2., "LBR Stack (Pentium 4 and Intel Xeon Processors)".

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
1DBH	475	MSR_LASTBRANCH _0	Last Branch Record 0. (R/W) One of four last branch record registers on the last branch record stack. It contains pointers to the source and destination instruction for one of the last four branches, exceptions, or interrupts that the processor took. See Section 14.5.2., "LBR Stack (Pentium 4 and Intel Xeon Processors)".
1DCH	476	MSR_LASTBRANCH _1	Last Branch Record 1. See description of the MSR_LASTBRANCH_0 MSR.
1DDH	477	MSR_LASTBRANCH _2	Last Branch Record 1. See description of the MSR_LASTBRANCH_0 MSR.
1DEH	478	MSR_LASTBRANCH _3	Last Branch Record 1. See description of the MSR_LASTBRANCH_0 MSR.
200H	512	IA32_MTRR_PHYS BASE0	Variable Range Base MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
201H	513	IA32_MTRR_ PHYSMASK0	Variable Range Mask MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
202H	514	IA32_MTRR_ PHYSBASE1	Variable Range Base MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
203H	515	IA32_MTRR_ PHYSMASK1	Variable Range Mask MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
204H	516	IA32_MTRR_ PHYSBASE2	Variable Range Base MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
205H	517	IA32_MTRR_ PHYSMASK2	Variable Range Mask MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
206H	518	IA32_MTRR_ PHYSBASE3	Variable Range Base MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
207H	519	IA32_MTRR_ PHYSMASK3	Variable Range Mask MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
208H	520	IA32_MTRR_ PHYSBASE4	Variable Range Base MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
209H	521	IA32_MTRR_ PHYSMASK4	Variable Range Mask MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
20AH	522	IA32_MTRR_ PHYSBASE5	Variable Range Base MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
20BH	523	IA32_MTRR_ PHYSMASK5	Variable Range Mask MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
20CH	524	IA32_MTRR_ PHYSBASE6	Variable Range Base MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
20DH	525	IA32_MTRR_ PHYSMASK6	Variable Range Mask MTRR. See Section 9.11.2.3., "Variable Range MTRRs".

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
20EH	526	IA32_MTRR_ PHYSBASE7	Variable Range Base MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
20FH	527	IA32_MTRR_ PHYSMASK7	Variable Range Mask MTRR. See Section 9.11.2.3., "Variable Range MTRRs".
250H	592	IA32_MTRR_FIX64K_ 00000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
258H	600	IA32_MTRR_FIX16K_ 80000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
259H	601	IA32_MTRR_FIX16K_ A0000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
268H	616	IA32_MTRR_FIX4K_ C0000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
269H	617	IA32_MTRR_FIX4K_ C8000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
26AH	618	IA32_MTRR_FIX4K_ D0000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
26BH	619	IA32_MTRR_FIX4K_ D8000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
26CH	620	IA32_MTRR_FIX4K_ E0000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
26DH	621	IA32_MTRR_FIX4K_ E8000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
26EH	622	IA32_MTRR_FIX4K_ F0000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
26FH	623	IA32_MTRR_FIX4K_ F8000	Fixed Range MTRR. See Section 9.11.2.2., "Fixed Range MTRRs".
277H	631	IA32_CR_PAT	Page Attribute Table. See Section 9.12.2., "IA32_CR_PAT MSR", for further information about this MSR.
2FFH	767	IA32_MTRR_DEF_ TYPE	Default Memory Types. (R/W) Sets the memory type for the regions of physical memory that are not mapped by the MTRRs. See Section 9.11.2.1., "IA32_MTRR_DEF_TYPE MSR".
300H	768	MSR_BPU_COUNTER0	See Section 14.9.2., "Performance Counters".
301H	769	MSR_BPU_COUNTER1	See Section 14.9.2., "Performance Counters".
302H	770	MSR_BPU_COUNTER2	See Section 14.9.2., "Performance Counters".
303H	771	MSR_BPU_COUNTER3	See Section 14.9.2., "Performance Counters".
304H	772	MSR_MS_COUNTER0	See Section 14.9.2., "Performance Counters".
305H	773	MSR_MS_COUNTER1	See Section 14.9.2., "Performance Counters".

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
306H	774	MSR_MS_COUNTER2	See Section 14.9.2., “Performance Counters”.
307H	775	MSR_MS_COUNTER3	See Section 14.9.2., “Performance Counters”.
308H	776	MSR_FLAME_ COUNTER0	See Section 14.9.2., “Performance Counters”.
309H	777	MSR_FLAME_ COUNTER1	See Section 14.9.2., “Performance Counters”.
30AH	778	MSR_FLAME_ COUNTER2	See Section 14.9.2., “Performance Counters”.
30BH	779	MSR_FLAME_ COUNTER3	See Section 14.9.2., “Performance Counters”.
3OCH	780	MSR_IQ_COUNTER0	See Section 14.9.2., “Performance Counters”.
3ODH	781	MSR_IQ_COUNTER1	See Section 14.9.2., “Performance Counters”.
3OEH	782	MSR_IQ_COUNTER2	See Section 14.9.2., “Performance Counters”.
3OFH	783	MSR_IQ_COUNTER3	See Section 14.9.2., “Performance Counters”.
310H	784	MSR_IQ_COUNTER4	See Section 14.9.2., “Performance Counters”.
311H	785	MSR_IQ_COUNTER5	See Section 14.9.2., “Performance Counters”.
360H	864	MSR_BPU_CCCR0	See Section 14.9.3., “CCCR MSRs”.
361H	865	MSR_BPU_CCCR1	See Section 14.9.3., “CCCR MSRs”.
362H	866	MSR_BPU_CCCR2	See Section 14.9.3., “CCCR MSRs”.
363H	867	MSR_BPU_CCCR3	See Section 14.9.3., “CCCR MSRs”.
364H	868	MSR_MS_CCCR0	See Section 14.9.3., “CCCR MSRs”.
365H	869	MSR_MS_CCCR1	See Section 14.9.3., “CCCR MSRs”.
366H	870	MSR_MS_CCCR2	See Section 14.9.3., “CCCR MSRs”.
367H	871	MSR_MS_CCCR3	See Section 14.9.3., “CCCR MSRs”.
368H	872	MSR_FLAME_CCCR0	See Section 14.9.3., “CCCR MSRs”.
369H	873	MSR_FLAME_CCCR1	See Section 14.9.3., “CCCR MSRs”.
36AH	874	MSR_FLAME_CCCR2	See Section 14.9.3., “CCCR MSRs”.
36BH	875	MSR_FLAME_CCCR3	See Section 14.9.3., “CCCR MSRs”.
36CH	876	MSR_IQ_CCCR0	See Section 14.9.3., “CCCR MSRs”.
36DH	877	MSR_IQ_CCCR1	See Section 14.9.3., “CCCR MSRs”.
36EH	878	MSR_IQ_CCCR2	See Section 14.9.3., “CCCR MSRs”.
36FH	879	MSR_IQ_CCCR3	See Section 14.9.3., “CCCR MSRs”.
370H	880	MSR_IQ_CCCR4	See Section 14.9.3., “CCCR MSRs”.

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
371H	881	MSR_IQ_CCCR5	See Section 14.9.3., “CCCR MSRs”.
3A0H	928	MSR_BSU_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3A1H	929	MSR_BSU_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3A2H	930	MSR_FSB_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3A3H	931	MSR_FSB_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3A4H	932	MSR_FIRM_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3A5H	933	MSR_FIRM_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3A6H	934	MSR_FLAME_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3A7H	935	MSR_FLAME_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3A8H	936	MSR_DAC_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3A9H	937	MSR_DAC_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3AAH	938	MSR_MOB_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3ABH	939	MSR_MOB_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3ACH	940	MSR_PMH_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3ADH	941	MSR_PMH_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3AEH	942	MSR_SAAT_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3AFH	943	MSR_SAAT_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3B0H	944	MSR_U2L_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3B1H	945	MSR_U2L_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3B2H	946	MSR_BPU_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3B3H	947	MSR_BPU_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3B4H	948	MSR_IS_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3B5H	949	MSR_IS_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3B6H	950	MSR_ITLB_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3B7H	951	MSR_ITLB_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3B8H	952	MSR_CRU_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3B9H	953	MSR_CRU_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3BAH	954	MSR_IQ_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3BBH	955	MSR_IQ_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3BCH	956	MSR_RAT_ESCR0	See Section 14.9.1., “ESCR MSRs”.
3BDH	957	MSR_RAT_ESCR1	See Section 14.9.1., “ESCR MSRs”.
3BEH	958	MSR_SSU_ESCR0	See Section 14.9.1., “ESCR MSRs”.

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
3C0H	960	MSR_MS_ESCR0	See Section 14.9.1., "ESCR MSRs".
3C1H	961	MSR_MS_ESCR1	See Section 14.9.1., "ESCR MSRs".
3C2H	962	MSR_TBPU_ESCR0	See Section 14.9.1., "ESCR MSRs".
3C3H	963	MSR_TBPU_ESCR1	See Section 14.9.1., "ESCR MSRs".
3C4H	964	MSR_TC_ESCR0	See Section 14.9.1., "ESCR MSRs".
3C5H	965	MSR_TC_ESCR1	See Section 14.9.1., "ESCR MSRs".
3C8H	968	MSR_IX_ESCR0	See Section 14.9.1., "ESCR MSRs".
3C9H	969	MSR_IX_ESCR0	See Section 14.9.1., "ESCR MSRs".
3CAH	970	MSR_ALF_ESCR0	See Section 14.9.1., "ESCR MSRs".
3CBH	971	MSR_ALF_ESCR1	See Section 14.9.1., "ESCR MSRs".
3CCH	972	MSR_CRU_ESCR2	See Section 14.9.1., "ESCR MSRs".
3CDH	973	MSR_CRU_ESCR3	See Section 14.9.1., "ESCR MSRs".
3E0H	992	MSR_CRU_ESCR4	See Section 14.9.1., "ESCR MSRs".
3E1H	993	MSR_CRU_ESCR5	See Section 14.9.1., "ESCR MSRs".
3FOH	1008	MSR_TC_PRECISE _EVENT	
3F1H	1009	IA32_PEBS_ENABLE	Precise Event-Based Sampling (PEBS). (R/W) Controls the enabling of precise event sampling and replay tagging.
		12:0	See Table A-5.
		23:13	**Reserved.**
		24	**UOP Tag.** Enables replay tagging when set.
		25	**Enable PEBS.** (R/W) Enables PEBS when set; disables PEBS when clear (default).
		63:26	**Reserved.**
3F2H	1010	MSR_PEBS_MATRIX _VERT	See Table A-5.
400H	1024	IA32_MC0_CTL	See Section 13.3.2.1., "IA32_MCi_CTL MSRs".
401H	1025	IA32_MC0_STATUS	See Section 13.3.2.2., "IA32_MCi_STATUS MSRs".
402H	1026	IA32_MC0_ADDR	See Section 13.3.2.3., "IA32_MCi_ADDR MSRs".
403H	1027	IA32_MC0_MISC	See Section 13.3.2.4., "IA32_MCi_MISC MSRs".
404H	1028	IA32_MC1_CTL	See Section 13.3.2.1., "IA32_MCi_CTL MSRs".
405H	1029	IA32_MC1_STATUS	See Section 13.3.2.2., "IA32_MCi_STATUS MSRs".
406H	1030	IA32_MC1_ADDR	See Section 13.3.2.3., "IA32_MCi_ADDR MSRs".

Register Address		Register Name Fields and Flags	Bit Description
Hex	Dec		
407H	1031	IA32_MC1_MISC	See Section 13.3.2.4., "IA32_MCi_MISC MSRs".
408H	1032	IA32_MC2_CTL	See Section 13.3.2.1., "IA32_MCi_CTL MSRs".
409H	1033	IA32_MC2_STATUS	See Section 13.3.2.2., "IA32_MCi_STATUS MSRs".
40AH	1034	IA32_MC2_ADDR	See Section 13.3.2.3., "IA32_MCi_ADDR MSRs".
40BH	1035	IA32_MC2_MISC	See Section 13.3.2.4., "IA32_MCi_MISC MSRs".
40CH	1036	IA32_MC3_CTL	See Section 13.3.2.1., "IA32_MCi_CTL MSRs".
40DH	1037	IA32_MC3_STATUS	See Section 13.3.2.2., "IA32_MCi_STATUS MSRs".
40EH	1038	IA32_MC3_ADDR	See Section 13.3.2.3., "IA32_MCi_ADDR MSRs".
40FH	1039	IA32_MC3_MISC	See Section 13.3.2.4., "IA32_MCi_MISC MSRs".
600H	1536	IA32_DS_AREA	DS Save Area. (R/W) Points to the DS buffer management area, which is used to manage the BTS and PEBS buffers (see Section 14.9.4., "Debug Store (DS) Mechanism").
		31:0	**DS Buffer Management Area.** Linear address of the first byte of the DS buffer management area.
		63:32	**Reserved.**

Tabelle 4: Liste der Model Specific Register

(Quelle: IA32 Intel Architecture Software Developer's Manual , Volume 3, Anhang B)