Welcher Typ ist der Co...?

Typ-Identifizierung von CPU und Coprozessor in MS-DOS-PCs

Das Ziel wohl jeden Softwareentwicklers ist es, die Anwender seines Programms rundum zufriedenzustellen. Eines der vielen Kriterien, das der Programmierer dabei beherzigen muß, ist die optimale Ausnutzung der CPU-(und Coprozessor-)Leistung im Anwender-PC, denn so etwas bringt Geschwindigkeit. Wegen der vielen unterschiedlichen Prozessortypen muß ein gutes Programm also vor dem Start selbst herausfinden, von welchem „Rechenchef“ es demnächst bearbeitet werden wird. Wie es das tut, das lesen Sie hier.

Im Abstand von nur wenigen Jahren brachte Intel immer neue Generationen der 80x86-CPU-Reihe heraus, jede mit einem spürbaren Leistungs-Plus im Vergleich zu den Vorgängern. Dazu kam eine ständige Erhöhung der Taktfrequenzen von ursprünglich 5 MHz beim 8086 bis zu fast 50 MHz heutzutage. Auch die Befehlssätze wurden ständig erweitert, und beim Übergang vom 80286 zum 80386 wechselte man sogar von einer 16- auf eine 32-Bit-Architektur. Alle denkbaren Hardware-Konfigurationen, dazu die Coprozessoren, findet man heute bei den PC-Anwendern vor, weshalb der Softwareentwickler vor der schwierigen Frage steht: „Auf welche CPU und auf welchen Coprozessor soll ich mein Programm zurechtschneidern?“

Der Programmierer kann sich natürlich auf die vom 8086 gebotene Power beschränken. Eine solche Software verschenkt dann auf 386/486-Rechnern einen Großteil der potentiellen Leistung. Er kann andererseits die speziellen Möglichkeiten der modernen Hochleistungschips ausnutzen und nimmt dabei in Kauf, daß seine Software auf Millionen noch existierender 8086- oder 80286-Computer nicht lauffähig ist. Die beste Lösung ist aber doch letztlich, wenn ein Programm die Hardware-Ressourcen optimal nutzt. Ein Beispiel für dieses Verhalten gibt Windows 3.0, das drei verschiedene Modi bietet, die grob den Prozessorklassen 8086, 80286 und 386/486 entsprechen. Jeder Modus nützt die speziellen Eigenschaften der zugeordneten CPU und bietet somit für diese eine optimale Leistung.

Keine Leistung verschenken

Die Coprozessor-Emulatoren, die fast jede Compilerbibliothek anbietet, verdeutlichen ebenfalls dieses Konzept. Programme mit eingebundenem Emulator benutzen für Fließkommaberechnungen den Coprozessor (*Numerical Data Processor*, NDP), wenn dieser beim Programmstart festgestellt wird. Sonst greifen sie auf Softwareroutinen zurück. Für die Erstellung professioneller Software ist daher eine Routine, die zuverlässig die verschiedenen CPUs und NDPs identifiziert, von großem Interesse. Ein solches Unterprogramm muß dann aber sehr hardwarenah ausgelegt sein, weshalb als Programmiersprache nur Assembler in Frage kommt. Dies gewährleistet gleichzeitig, daß die Routine nicht zu umfangreich wird und sich in jede höhere Programmiersprache einbinden läßt. Um eine

universelle Verwendbarkeit zu ermöglichen, zieht das Programm CPUTEST im *Listing* nur solche Eigenschaften der CPUs zur Unterscheidung heran, die unabhängig vom übrigen System sind. Das Programm CPUTEST2 hingegen benutzt das Zeitverhalten verschiedener Instruktionen, um den Typ noch genauer zu identifizieren. Aus Platzmangel finden Sie dieses Programm jedoch nur als Paperdisk in der mc und auf der Softedition. Zur Unterscheidung zwischen den 8- und 16-Bit-Buskonzepten siehe auch die Hinweise im *Kasten*.

Datenbus mit 8 und 16 Bit: der kleine, große Unterschied

Zu den Prozessoren 8086, 80186 und V30, deren interne und externe Datenpfade jeweils 16 Bit breit sind, existieren Pendants, die ihre Daten außerhalb des Chips nur mit acht Bit Breite bewegen, nämlich der 8088, der 80188 und der V20. Dadurch kommt man auf der Platine wegen der schmaleren Busse mit weniger Leitungen aus und kann preiswerte 8-Bit-Peripheriebausteine einsetzen. Das bringt gegenüber reinrassigen 16-Bit-Systemen eine gewisse Kostenersparnis, aber auch eine auf zwei Drittel reduzierte Systemleistung. Ähnliche wirtschaftliche Überlegungen spielten auch bei der Entwicklung der 80386-Schmalspurausführung 80386sx eine Rolle. Auf den ersten Blick aus der Softwareperspektive sehen die jeweiligen 8- und 16-Bit-Chips identisch aus, so daß eine weitere Differenzierung unmöglich erscheint. Mit etwas Einblick in den Prozessoraufbau findet sich aber doch eine Lösung.

Zur beschleunigten Ausführung von Befehlen verfügen die Intel-und NEC-Prozessoren, ebenso wie andere moderne Mikroprozessoren, über eine sogenannte „Prefetch Queue“. Hierbei handelt es sich um einige schnelle Speicherplätze in der CPU, die als Warteschlange verwaltet werden. Während die CPU einen Befehl ausführt, werden die linear folgenden Befehlsbyte aus dem Speicher in die Warteschlange geladen. Dadurch stehen sie sofort nach Abarbeitung des aktuellen Befehls zur Verfügung. Das Holen der Befehle und ihre Ausführung findet also mit Hilfe der Prefetch Queue überlappend statt. Ohne diesen Mechanismus ginge nach der Ausführung eines Befehls kostbare Zeit verloren, um die nächste Instruktion in die CPU zu befördern. Nun wird der Befehlsfluß immer wieder durch Sprünge und Unterprogrammaufrufe aus seiner linearen Bahn geworfen. Deshalb darf die Warteschlange auch nicht zu lang sein, da sie bei einem solchen Ereignis gelöscht und neu geladen werden muß, was seinerseits Zeit in Anspruch nimmt. Die Länge der Warteschlange muß daher an die Leistungsfähigkeit des Datenbusses angepaßt sein. Die 8-Bit-Typen der obengenannten CPUs verfügen über eine vier Byte lange Prefetch Queue, während die Warteschlange der 16-Bit-Typen sechs Byte umfaßt [1, 2, 4]. Diese Tatsache kann man nun mit einem sich selbst modifizierenden Programm ausnutzen. Verändert man einen Befehl im Speicher, der sich fünf oder sechs Byte vor der gerade ausgeführten Instruktion befindet, so wird diese Änderung von einer CPU mit einer sechs Byte langen Prefetch Queue nicht mehr berücksichtigt. Bei einer Länge der Warteschlange von vier Byte liegt das geänderte Kommando zum Zeitpunkt der Modifikation außerhalb der CPU, und es wird schließlich die geänderte Instruktion ausgeführt. Die beschriebene Methode ist außer von der CPU aber auch vom restlichen System abhängig. Damit sie funktioniert, muß die Prefetch Queue im Moment des Speicherzugriffes maximal gefüllt sein. Weist der Speicher Wartezyklen beim Lesezugriff auf, so kann die Füllung der Warteschlange soviel Zeit in Anspruch nehmen, daß diese Forderung nicht ohne weiteres erfüllt ist. Durch die Ausführung eines Befehls mit hohem Zeitbedarf direkt vor der Codemodifikation läßt sich das Problem lösen. Während der Abarbeitung dieses zeitintensiven Befehls kann die Prefetch Queue auch bei einem langsamen Speicherzugriff gefüllt werden.

Am Anfang: 8086/8088

Der 8086, im Jahre 1978 auf den Markt gebracht, steht am Anfang der bis heute erfolgreichen Intel 80x86-Baureihe. Ihr Erfolg beruht vor allem auf der Aufwärtskompatiblität der immer leistungfähigeren Prozessoren. Kurze Zeit später kam der 8088 hinzu, der sich vom 8086 durch seine externen 8-Bit-Busse unterscheidet. Im Flagregister des 8086/8088 haben die höchstwertigen vier Bit keinerlei Funktion. Sie sind daher durch die Hardware permanent

gesetzt und können nicht gelöscht werden, im Gegensatz zu den anderen unbenutzten Bits des Flagregisters [1]. Dies liefert ein wichtiges und leicht überprüfbares Erkennungsmerkmal für diese Prozessoren. Der Inhalt des Flagregisters wird dazu durch ein PUSHF zuerst auf den Stapel und durch ein anschließendes POP AX in das Akkumulatorregister AX gebracht. Dort werden die höchstwertigen vier Bit gelöscht und der neue Wert mittels PUSH AX, POPF in das Flagregister geladen. Ein weiteres Mal überführt man die Flags auf dem Umweg über den Stapel in den Akkumulator. Hier wird nun geprüft, ob die obersten vier Bit gelöscht werden konnten. Zu beachten ist, daß der direktere Zugriff auf die Flags mittels der Befehle LAHF und SAHF für diese Anwendung ungeeignet ist. Diese Kommandos berücksichtigen bei Lese- und Schreibzugriffen auf das Flagregister lediglich bestimmte definierte Bits.

Japanische Verfolger: V20/V30

Der V20 und der V30 wurden von der japanischen Firma NEC als Nachbauten des 8088 bzw. 8086 entwickelt. Die Chips von NEC sind aber nicht eineiige Zwillinge der Intel-CPUs. Vielmehr weisen sie gegenüber den Originalen einige Verbesserungen der internen Architektur auf, wie beispielsweise eine Verdoppelung der CPU-internen Datenpfade [2]. Daraus resultiert für den V20 bei gleicher Taktfrequenz eine um durchschnittlich 15 Prozent höhere Leistung als beim 8088; beim V30 liegt der Gewinn sogar bei 25 Prozent gegenüber dem 8086. Der Befehlssatz der NEC-CPUs ist eine Obermenge der auf dem 8086 zur Verfügung stehenden Instruktionen. So kennen sie die zusätzlichen Befehle der 80186/80188-Prozessoren und darüber hinaus Anweisungen zur Verarbeitung von BCD-Zahlen und Befehle zur Bitmanipulation. Letztere sind allerdings mit keinem Intel-Prozessor kompatibel.

Zur Unterscheidung zwischen dem 8088/8086 und dem V20/V30 eignet sich der POPA-Befehl. Dieser Befehl, der bei Intel erst ab dem 80186 implementiert ist, dient dem Wiederherstellen der zuvor mit einem PUSHA gesicherten acht allgemeinen Register der CPU. Eine Ausführung dieser auf dem 8086 nicht vorhandenen Instruktion auf einem 8086 oder 8088 führt aber nicht zum Absturz, sondern wird als Vorwärtssprung interpretiert. Da der Sprung genau das dem POPA-Befehl folgende Byte überspringt, bleibt Platz genug, um hier ein STC unterzubringen. Wenn vor dem Ausführen der Sequenz aus POPA und STC das Übertragsflag gelöscht war, so wird es anschließend genau dann gesetzt sein, wenn ein V20 oder V30 diese Befehlsfolge abarbeitete. Auf einem 8086/8088 hingegen wird die das Übertragflag setzende STC-Anweisung übersprungen.

Peripherie integriert: 80186/80188

Die seit 1982 verfügbaren Intel-Chips 80186/80188 sind mehr als nur um zehn Befehle erweiterte 8086/8088-Prozessoren. Vielmehr sind hier außer der CPU viele in einem PC übliche Peripheriebausteine, wie Clock Generator, Timer, Interrupt-Controller und die DMA-Kanäle auf einem Chip vereint [3]. Die neuere Ausführung 80C186 enthält sogar noch mehr Funktionseinheiten, beispielsweise einen DRAM-Refresh-Controller [4]. Bei gleicher Taktfrequenz bieten diese CPUs überdies eine rund 20 Prozent höhere Leistung als ein 8088 oder 8086. Bei der Ausführung von Schiebe- und Rotieroperationen auf 80188/80186-Prozessoren gibt es eine Abweichung von dem Verhalten der 8086/8088- und V20/V30-CPUs: Wird durch einen Befehl eine Schiebe- oder Rotationsoperation wiederholt ausgeführt, so führen letztere soviele Wiederholungen aus, wie im 8-Bit Zähler festgelegt ist. Im Extremfall sind das 255 Operationen. Eine derartig häufige Ausführung ist aber unsinnig, da bei der Anwendung auf 16-Bit-Register spätestens nach sechzehnmaligem Schieben entweder 0 oder FFFFh (bei SAR) im Register steht. Beim Rotieren ergibt sich ebenso nach sechzehn- (ROL, ROR) bzw. siebzehnmaliger (RCL, RCR) Ausführung der Operation der ursprüngliche Registerinhalt. Ein höherer Schiebefaktor verbraucht nur unnötig Zeit. Daher wird ab dem 80186/80188 aufwärts bei allen Intel-Prozessoren der Zähler für Schiebe- und Rotieroperationen auf fünf Bits maskiert. Es kann also zwischen 0 und 31 mal geschoben werden, höhere Schiebefaktoren werden modulo 32 auf diesen Bereich abgebildet. Diese Eigenschaft kann zur Unterscheidung des 80186/80188 vom 8086/8088 und V20/V30 herangezogen werden. Wird ein Register mit 1 geladen und darauf ein 33-faches Schieben angewandt, so ergibt sich auf einem 80186/80188 der Wert 2, bei den übrigen Prozessoren aber Null.

Alt, aber bewährt: 80286

Mit dem 1982 herausgebrachten 80286 rückte Intel wieder von der beim 80186 begonnenen Integration von Peripheriebausteinen in die CPU ab. Stattdessen wurde ein enormer Aufwand getrieben, um Hochsprachen und vor allem Multitasking-Betriebssysteme hardwaremäßig zu unterstützen. Ein Unterschied des 80286 zu allen früheren Intel-Prozessoren und dem V20/V30 ist die Behandlung des Befehls PUSH SP. Die Implementierung dieser Anweisung ist nicht völlig unproblematisch, da generell ein PUSH den Stapelzeiger SP natürlich verändert. Bei

einem PUSH-Befehl wird im allgemeinen zuerst der Stapelzeiger um zwei vermindert und dann der gewünschte 16-Bit-Wert an die Adresse geschrieben, auf die SP zeigt. Bezogen auf den Stapelzeiger selbst bedeutet dies, daß auf dem Stapel der Wert von SP nach dem PUSH abgelegt wird. So ist PUSH SP auf allen Prozessoren vor dem 80286 auch realisiert.

Ein PUSH-Befehl wird im Normalfall zum Sichern eines Registers auf dem Stapel verwendet, wobei das Register natürlich nicht verändert werden soll. Entsprechend wurde der Befehl PUSH SP beim 80286 und allen nachfolgenden Intel-CPUs realisiert. Dies führt dazu, daß nach Abarbeitung der Befehlssequenz PUSH SP, POP AX auf dem 80286 die Register AX und SP den gleichen Wert enthalten, auf den älteren Prozessoren aber zu diesem Zeitpunkt AX = SP-2 gilt. Diese einfache Anweisungsfolge ermöglicht also sehr effektiv, die Prozessorengruppe 8086, 80186, V30 inklusive der entsprechenden 8-Bit-Typen von der Gruppe 80286, 80386, 80486 zu unterscheiden [5]. Der 80386, seit 1986 im Einsatz, stellt gegenüber seinem Vorgänger einen gewaltigen Fortschritt dar. Zwar werden die meisten CPU-Kommandos nur unbedeutend schneller als beim 80286 ausgeführt.

Der Hit mit 32 Bit: 80386

Der Übergang auf eine 32-Bit-Architektur ermöglicht durch breitere Register und neue, leistungsfähige Adressierungsarten aber eine im Vergleich zum 80286 um 50 Prozent höhere Leistung. Darüber hinaus macht der 80386 Schluß mit der Unterteilung des Speichers in Segmente von maximal 64 KByte. Diese können nun bis zu 4 GByte groß sein. Zusätzlich verfügt dieser Prozessor über eine integrierte Speicherverwaltungseinheit (MMU, Memory Management Unit), die sogenanntes Paging mit 4 KByte großen Seiten gestattet [6]. An zusätzlichen Befehlen bietet der 80386 eine umfassende Auswahl von Kommandos zur Bitmanipulation. Dazu kommen einige sehr nützliche Schiebebefehle, die vor allem die Implementierung von schnellen BitBlt-Routinen für grafische Software unterstützen [7]. Eine interessante Neuerung stellen die Debugregister des 80386 dar. Sie unterstützen die modernen symbolischen Debugger wie CodeView oder den Turbo-Debugger und ermöglichen auch das Setzen von Breakpoints bei der Untersuchung von Code in einem ROM. Der 80386 enthält wie der 80286 im Flagregister zusätzliche Flags für die Verwaltung des Protected-Mode. Es handelt sich dabei um das Zweibit-Flag IOPL (Input/Output Privilege Level) und das Flag NT (Nested Task). Diese Flags können im Real-Modus des 80286 nicht gesetzt werden, während dies beim 80386 möglich ist. Zur Unterscheidung der Prozessoren wird mittels POPF ein Wert in das Flagregister befördert, bei dem die Bits für NT und IOPL gesetzt sind. Anschließend wird der nun im Flagregister befindliche Wert mit PUSHF ausgelesen und untersucht, ob die Bits für NT/IOPL gesetzt sind. Ist dies der Fall, so liegt ein 80386, sonst ein 80286 vor.

Doppelwort-Schiebungen: 80386DX/SX

Die Prozessoren 386DX und 386sx verfügen beide über eine Prefetch Queue von 16 Byte Länge. Eine Unterscheidung zwischen diesen Chips analog der beim 8086/8088 angewendeten Methode scheidet also aus. Nach einem Reset teilen die beiden CPUs zwar ihre Identität im DH-Register mit, wobei der 386DX hier eine 3 abspeichert und der 386SX eine 23h [8], aber diese Information kann wegen ihrer Flüchtigkeit nur vom BIOS ausgewertet werden. Eine systemabhängige Lösung zur Differenzierung der beiden Typen besteht darin, die unterschiedliche Leistung beim Datentransfer mittels der MOVSW-und MOVSD-Instruktionen auszunutzen. Hierzu wird ein Datenblock, der auf einer durch vier teilbaren Adresse beginnt, auf sich selbst verschoben. Mit REP MOVSW werden zuerst n Worte transportiert und die Zeit mit dem Timerbaustein des PCs gemessen. Anschließend bewegt man mit REP MOVSD n Doppelworte und bestimmt auch hierfür die Zeit. Auf einem 386DX werden die Werte annähernd gleich sein, da er in jeweils vier Taktzyklen ein Wort oder Doppelwort bewegt, wogegen der 386sx beim Transport der Doppelworte jeweils zwei 16-Bit-Buszugriffe benötigt und somit für n Doppelworte ungefähr doppelt so lange benötigt wie für n Worte. Diese Methode wurde mit verschiedenen 386- und 386sx-Computern erfolgreich getestet.

Mit dem Co im Verein: 80486

Der neueste Chip in der 80x86-Prozessorfamilie ist der 80486. Vereinfacht ausgedrückt, handelt es sich hierbei um einen 80386, der zusammen mit einem mathematischen Coprozessor 80387 und einem 8 KByte großen Cache auf einem Chip integriert ist [9]. Durch die Anwendung von Prinzipien aus RISC-Architekturen, vor allem das Instruction-Pipelining, konnte aber die Ausführungszeit der einzelnen Befehle beträchtlich verbessert werden. Bei gleicher Taktfrequenz ist daher ein 80486 mehr als doppelt so schnell wie der Vorgänger 80386 [10]. Durch die Integration der Fließkommaeinheit (FPU = floating point unit) auf dem Chip entfällt die sonst übliche Kommunika-

tion zwischen CPU und NDP komplett. Dadurch ist eine beträchtliche Leistungssteigerung möglich; mit ca. 1 MFLOPS bei 25 MHz erreicht der 80486 die numerische Verarbeitungskapazität von älteren Großrechern.

Der 80486 bietet nur wenige neue Befehle an, die alle Spezialzwecken dienen. Neben Anweisungen zur Steuerung des Cache gibt es einige, die für die Koordination in Mehrprozessorsystemen nützlich sind. Dabei wurden auch die Bedürfnisse von RISC-Prozessoren wie dem i860 von Intel berücksichtigt. Diese schreiben fast immer vor, daß Daten ausgerichtet im Speicher abgelegt werden. Das bedeutet, daß Worte nur auf geraden Adressen, Doppelworte nur auf durch vier teilbaren Adressen gespeichert werden dürfen. Wird diese Regel mißachtet, kommt es zu einer Fehlerunterbrechung (Exception). Für die Zusammenarbeit mit RISC-Prozessoren kann auf dem 80486 durch Setzen eines Flags dieses Verhalten erzwungen werden. Dieses Flag AC (Alignment Check) befindet sich im 32-Bit-Flagregister EFLAGS. Um den 80386 vom 80486 zu unterscheiden, versucht man, das AC-Bit in EFLAGS zu setzen [11]. Da dieses beim 80386 nicht existiert, läßt es sich auch nicht setzen, sondern wird hardwaremäßig im gelöschten Zustand festgehalten.

Ist der „Co“ da oder nicht?

Mitte 1991 brachte Intel überraschend den Prozessor 486sx auf den Markt. Hierbei handelt es sich um einen mit nur 20 MHz getakteten 80486 ohne Fließkommaeinheit (FPU) [29]. Bei der Markteinführung dieses Chips geht es Intel wohl vor allem darum, den Mitbewerber AMD mit dessen 40-MHz-Version des 80386 aus dem Markt zu drängen, nachdem der Versuch gescheitert ist, diese Konkurrenz per Gerichtsverfahren auszuschalten. Die Unterscheidung zwischen dem 80486 und dem 486sx kann einfach durch den Test auf die Anwesenheit eines Coprozessors getroffen werden.

Um zu testen, ob ein NDP im System ist, werden einige Coprozessorbefehle ausgeführt, die bestimmte Werte im Speicher ablegen. In Systemen ohne NDP werden diese Befehle vom Hauptprozessor völlig ignoriert und folglich diese Werte auch nicht gespeichert. Durch Überprüfen der Speicherwerte kann also sicher die Existenz eines NDP festgestellt werden. Zuerst wird der Coprozessor durch ein FNINIT in einen genau definierten Zustand gebracht. Das N vor dem INIT ist wichtig. Hier wird dem Assembler mitgeteilt, er möge kein WAIT vor FINIT einfügen, was er sonst automatisch vor jedem NDP-Befehl macht. Ein WAIT in einem Rechner ohne Coprozessor würde nämlich die CPU anhalten, da der NDP, auf den da gewartet wird, natürlich niemals antwortet. Durch FNINIT werden im Kontrollwort des NDP alle Interrupts maskiert, der Rundungsmodus wird auf 0 gesetzt (runde zur nächsten Zahl), und die interne Rechengenauigkeit wird auf 64 Mantissenbits festgelegt. Dieses entspricht dem Genauigkeitsmodus 3. Nach dem Abspeichern des Kontrollwortes werden die genannten Felder ausmaskiert und auf ihren Inhalt hin geprüft. Fällt das Ergebnis negativ aus, so ist kein NDP vorhanden. Durch die Initialisierung wird im Statuswort des Coprozessors der Stapelzeiger auf Null gesetzt; alle Ausnahmebits werden gelöscht. Das Statuswort wird ebenfalls im Speicher abgelegt und überprüft, ob die entsprechenden Bitfelder gelöscht sind. Wurde auch dieser Test bestanden, so ist noch zu überprüfen, ob etwa ein Emulator das Vorhandensein eines Coprozessors vorspiegelt.

Alle Intel-Prozessoren ab dem 80286 bieten nämlich die Möglichkeit, beim Auftauchen von NDP-Befehlen im Befehlsstrom einen Interrupt 7 auszulösen, wenn kein Coprozessor im System ist. In den Interrupt kann sich nun ein Programm einklinken, das den NDP-Befehl analysiert und die gewünschte Operation in Software emuliert. Ob ein Emulator aktiv ist, kann durch Abfragen des EM-Bits im MSW (machine Status word) herausgefunden werden.

Unterschiede der Coprozessoren

Es existieren im Grunde zwei verschiedene Generationen der Intel NDPs. Die erste besteht aus dem 8087 und dem 80287, zur zweiten gehören 80387, 80387sx, 80C187, 80287A und 80287XL. Der 8087 und der 80287 wurden entwickelt, als der IEEE-Fließkommastandard noch nicht in seiner endgültigen Form vorlag. Daher implementierte man den Standard anhand des damals vorliegenden Entwurfes, einer sogenannten Draft Version [13]. Dies führte dazu, daß diese Prozessoren geringfügig von dem im endgültigen Standard festgelegten Verhalten abweichen. Die zweite Generation der Intel-NDPs ist dagegen voll kompatibel mit dem Fließkommastandard. Außerdem wurden einige zusätzliche Befehle eingeführt, wovon die Instruktionen zur direkten Berechnung des Sinus und Cosinus die wichtigsten sind. Zur Unterscheidung der beiden Gruppen kann das unterschiedliche Verhalten bei der Behandlung des Wertes „Unendlich“ herangezogen werden. Gemäß der Vorabversion des IEEE-Standards sind beim 8087 und 80287 für die Behandlung dieses Werts zwei Modi vorgesehen. Im Modus „projective closure“, der nach der Initialisierung als Vorgabe eingestellt ist, gibt es nur einen Wert für Unendlich. Im Alternativmodus „affin closure“

dagegen werden +∞ und −∞ unterschieden. Dieser Modus ist gleichzeitig der einzig mögliche Modus für die neueren Coprozessoren.

Unendlich ist nicht unendlich

Die Unterscheidung der beiden Coprozessorgruppen erfolgt, indem nach der Initialisierung zuerst der Wert ∞ erzeugt wird. Dieser wird dann erneut geladen und negiert. Ein anschließender Vergleich ergibt dann beim 8087/80287 die Gleichheit beider Werte, da beim projektiven Modell nur ein Wert für Unendlich existiert. Auf den neueren NDPs dagegen bescheinigt ein Vergleich die Ungleichheit von +∞ und −∞, da hier das affine Modell verwendet wird. Da der CPU-Typ schon identifiziert wurde, kann innerhalb der einzelnen Gruppen weiter differenziert werden. Der 8087 arbeitet mit dem 8086/8088 zusammen und mittels eines weiteren Bausteins auch mit dem 80186/80188. Der 80287 arbeitet mit dem 80286 und dem 80386 zusammen. Ist die CPU ein 80286 und der Coprozessor einer der neueren Typen, so kommen nur 80287A und 80287XL in Frage. Bei einem NDP der zweiten Generation, der mit dem 80186/ 80188 eingesetzt wird, kann es sich nur um einem 80C187 handeln.

Die Kombination 80386/80387 und 80386/80287XL kann mit solchen Überlegungen nicht unterschieden werden, da beide NDPs zur zweiten Generation der Coprozessoren zählen. Allerdings dürfte dieser Fall extrem selten auftreten. Über einen Sockel für den 80287 verfügen nämlich nur solche Systeme, die vor dem Erscheinen des 80387 konzipiert wurden, während der 80287XL erst im letzten Jahr auf den Markt kam. Übrigens existiert der 80387 in zwei verschiedenen Versionen. Der Anfang 1990 als Verbesserung der Erstversion herausgekommene 80387DX bietet eine um durchschnittlich 20 Prozent höhere Leistung [19][21]. Eine Unterscheidung beider Chips hat zwar wenig praktischen Nutzen, kann aber vorgenommen werden, wenn die Ausführungszeiten der FBLD und FBSTP Befehle ausgewertet werden. Diese betragen beim 80387DX nur noch ein Drittel derer des ursprünglichen 80387.

Es gibt nicht nur Intel...

Seit einiger Zeit beschäftigen sich auch andere Anbieter mit der Fertigung von Coprozessoren. Diese NDPs sind im allgemeinen voll kompatibel zu den entsprechenden Intel-Chips. Die IIT-Coprozessoren 2C87 und 3C87 [22] beispielsweise verfügen über vier Registerbänke mit je acht Fließkommaregistern, von denen drei dem Programmierer zugänglich sind. Die Ausführung einer 4×4-Matrizenmultiplikation auf dem Chip wird somit durch einen zusätzlichen Befehl möglich gemacht. Diese Operation dient vor allem den Transformationen in 3D-Grafik-Applikationen, was gegenüber einem „normalen“ Coprozessor ein Geschwindigkeits-Plus bringt [23].

Zur Verwaltung der drei Registerbänke gibt es ebenfalls zusätzliche Befehle. Mit ihrer Hilfe ist eine Erkennung der IIT-Prozessoren möglich. IIT selbst verwendet allerdings eine noch elegantere Methode zur Erkennung ihrer NDPs, die ohne die Verwendung der speziellen Instruktionen der IIT-Coprozessoren auskommt [24]. Hierbei wird das unterschiedliche Verhalten der Intel- und IIT-Chips bei der Verarbeitung einer speziellen Operandenklasse, der denormalisierten Zahlen, ausgenutzt.

Die Firma Cyrix bietet als zum 80387 und 80387sx kompatible Coprozessoren den 83D87 bzw. den 83S87 an. Dieses Unternehmen hat großen Wert auf absolute Kompatibilität mit Intels Originalen gelegt [25]. Allerdings weisen sie eine beträchtlich größere Leistung auf, vor allem bei den höheren Funktionen wie z. B. Quadratwurzel oder Sinus [26]. Dies liegt daran, daß die Cyrix-Prozessoren ihre Funktionen durch Polynom-Approximationen mit einem schnellen Arraymultiplizierer bilden. Dagegen wird in allen Intel-NDPs das sogenannte CORDIC-Verfahren zur Berechnung der transzendenten Funktionen verwendet [14]. Die Cyrix-Chips berechnen diese Funktionen nicht nur schneller, sondern auch genauer als alle anderen Coprozessoren für die 80x86-Baureihe [27]. Wenn eine Identifizierung von 83D87/83S87 erforderlich ist, können die Cyrix-NDPs durch die enorme Geschwindigkeit beispielsweise der Quadratwurzelfunktion von den Intel-Chips unterschieden werden. Dies wurde erfolgreich ausprobiert, ist jedoch vom Vorhandensein eines Timers abhängig. Kürzlich hat Cyrix auch noch den 82S87 als Coprozessor für Systeme auf Basis des 80286 nachgeschoben [30]. Er entspricht in seinem internen Aufbau den 387-Modellen von Cyrix. Als Besonderheit ist anzumerken, daß eine einzige Version alle Taktfrequenzen zwischen zwölf und zwanzig Megahertz abdeckt.

Der 80C287 der Firma AMD verhält sich vollkommen identisch mit dem 80287, auch die Ausführungszeiten der Befehle sind gleich [28]. Es läßt sich daher kein Kriterium angeben, wie sich beide unterscheiden lassen. Durch die 100-Prozent-Kompatiblität hätte eine solche Differenzierung auch keinerlei praktische Bedeutung. Jüngster Mitbewerber auf dem Markt der Coprozessoren ist die amerikanische Firma ULSI mit ihrem 83C87/83C87sx.

Die in diesem Beitrag gezeigten Programme zur Prozessorerkennung wurden mit allen besprochenen CPUs mit Ausnahme von 80186/80188 getestet, da ein PC mit diesen Prozessoren nicht verfügbar war. Ebenso wurde die Coprozessor-Identifizierung anhand aller vorgestellten Typen außer dem 80187, dem 287XL, dem 82S87 und den ULSI 387ern überprüft. Zusätzlich erfolgte ein Test mit einem Public-Domain-Coprozessor-Emulator.

Norbert Juffa

Literatur

[1] *Rector, R. Alexy, G.:* Das 8086/8088 Buch, te-wi Verlag, München 1982
[2] NEC: Microprocessors & Peripherals. 1989
[3] Intel: iAPX 86, 88, 186 and 188 User's Manual, Programmer's Reference. 1983
[4] Intel: iAPX 86, 88, 186 and 188 User's Manual, Hardware Reference. 1985
[5] Intel: 80C186 CHMOS High Integration 16-Bit Microprocessor Data Sheet. 1989
[6] Intel: 80286 and 80287 Programmer's Reference Manual. 1987
[7] Intel: 386DX Microprocessor: High Performance 32-Bit CHMOS Microprocessor With Integrated Memory Management. Data Sheet. 1989
[8] Intel: 386DX Programmer's Reference Manual. 1990
[9] Intel: 386SX Microprocessor Data Sheet. 1990
[10] Intel: i486 Microprocessor Data Sheet. 1989
[11] Intel: i486 Microprocessor Performance Report. 1990
[12] Intel: i486 Microprocessor Programmer's Reference Manual. 1990
[13] *Thies, K.-D.:* Die 8087/80287 numerischen Prozessor-Erweiterungen für 8086/80286 Systeme. te-wi Verlag, München, 1985.
[14] *Palmer, J.F.; Morse, S.:* Die mathematischen Grundlagen der Numerik-Prozessoren 8087/80287. te-wi Verlag, München 1985
[15] Intel: 80C187 80-BitMath Coprocessor Data Sheet. 1989
[16] Intel: 80C287A CHMOS III Math Coprocessor Data Sheet. 1989
[17] Intel: 80287XL/XLT CHMOS III Math Coprocessor Data Sheet. 1990
[18] Intel: 387DX User's Manual. 1989
[19] Intel: 387 Numerics Coprocessor Extension Data Sheet. 1989
[20] Intel: 387SX Math Coprocessor Data Sheet. 1989
[21] Intel: 387DX Math Coprocessor Data Sheet. 1989
[22] Mikroprozessor Datenbuch 3. Elektor-Verlag, Aachen 1990.
[23] Integrated Information Technology: Engineering Note 4×4 Matrix Multiply Transformation. 1989
[24] *David Lewis*, IIT Europe: Persönliche Mitteilung vom 19.02.91
[25] Cyrix: FasMath 83D87 Compatibility Report. 1989
[26] Cyrix: FasMath 83D87 User's Manual. 1990
[27] Cyrix: FasMath 83D87 Accuracy Report. 1990
[28] Advanced Micro Devices: AMD 80C287 80 Bit CMOS Numeric Processor Data Sheet. 1989
[29] Intel: 486SX Microprocessor/487 Math Coprocessor Data Sheet. 1991
[30] Cyrix: FasMath 82S87 Math Coprocessor Data Sheet. 1991

Listing. Das CPU- und Coprozessor-Identifikations-Programms CPUTEST. Die Assemblerroutine eignet sich zur Einbindung in alle Hochsprachen. Sie nutzt nur solche Unterscheidungskriterien, die unabhängig von übrigen Systemparametern sind. Zu Demonstrationszwecken gibt das Programm die CPU- und NDP-Namen als Meldung aus. Dieser Teil kann natürlich entsprechend modifiziert oder weggelassen werden.

```
            PAGE    ,120

;--------------------------------------------------------------------------
;
; This is a test program to demonstrate the capabilities of subroutine
; CPU_NDP_TST. To produce an executable COM file, proceed as follows:
;
; MASM CPUTEST;
; LINK CPUTEST;
; EXE2BIN CPUTEST.EXE CPUTEST.COM
;
; use MASM 5.1 or TASM 2.01 to assemble this file
;
; CPU_NDP_TST is (c) 1988-1991 Norbert Juffa, this version dated 09-14-1991
;
; Code to check for IIT NDPs courtesy of IIT Corporation Europe 02-19-1991
;
;--------------------------------------------------------------------------

            .386p                       ; 386 and 387 code
            .387                        ;  will be used

JMPS        EQU     <JMP SHORT>         ; jumps have to be
JES         EQU     <JE  SHORT>         ;  declared as short,
JBS         EQU     <JB  SHORT>         ;   since the default
JNZS        EQU     <JNZ SHORT>         ;    distance with the .386
JCS         EQU     <JC  SHORT>         ;     directive is near
JZS         EQU     <JZ  SHORT>         ;      jumps, which can
JNES        EQU     <JNE SHORT>         ;       only be executed on
JAES        EQU     <JAE SHORT>         ;        386 and 486 machines


_TEXT       SEGMENT BYTE PUBLIC USE16 'CODE'
            ASSUME  CS:_TEXT, DS:_TEXT  ; DS=CS=ES=SS for COM program

            ORG     100h                ; to make COM program

MainStart:  MOV     DX, OFFSET IntroMsg ; pointer to introductory message
            MOV     AH, 9               ; DOS #9, print string
            INT     21h                 ; call DOS
            CALL    CPU_NDP_TST         ; determine installed CPU and NDP
            MOV     BX, AX              ; save processor types
            MOV     DX, OFFSET CPUMsg   ; pointer to CPU message
            MOV     AH, 9               ; DOS #9, print string
            INT     21h                 ; call DOS
            MOV     AX, BX              ; get processor types (AL = CPU)
            MOV     AH, 22              ; compute pointer
            MUL     AH                  ;  to message for
            ADD     AX, OFFSET Processor ;   this CPU type
            XCHG    AX, DX              ; pointer to correct CPU type message
```

```
            MOV     AH, 9                   ; DOS #9, print string
            INT     21h                     ; call DOS
            MOV     DX, OFFSET EOL          ; get pointer to CR-LF
            MOV     AH, 9                   ; DOS #9, print string
            INT     21h                     ; call DOS
            MOV     DX, OFFSET NDPMsg       ; get pointer to NDP message
            MOV     AH, 9                   ; DOS #9, print string
            INT     21h                     ; call DOS
            XCHG    AX, BX                  ; get processor types (AH = NDP)
            MOV     AL, 23                  ; compute pointer
            MUL     AH                      ;  to message for
            ADD     AX,OFFSET Coprocessor;   this NDP type
            XCHG    AX, DX                  ; pointer to correct CPU type message
            MOV     AH, 9                   ; DOS #9, print string
            INT     21h                     ; call DOS
            MOV     DX, OFFSET EOL          ; get pointer to CR-LF
            MOV     AH, 9                   ; DOS #9, print string
            INT     21h                     ; call DOS
            MOV     AX, 4C00h               ; DOS #4C, terminate program, errlvl 0
MainEnd:    INT     21h                     ; call DOS


;-------------------------------------------------------------------------------
; CPU_NDP_TST is a subroutine which returns the currently installed CPU and
; numerical coprocessor. To distinguish between the different types of CPUs
; and NDPs, it relies only on the incompatiblities of the processors themselves
; and does not require other system ressources such as timers. This routine
; must be executed in the real mode of i286, i386(sx), and i486 processors!
;
; INPUT:       NONE                    CPU          NDP
; OUTPUT:      AH   - NDP type         ----------   ------------
;              AL   - CPU type         n.a.   - 0       none - 0
;                                      8088   - 1   Emulator - 1
; DESTROYED:   AX,BX,CX,DX,Flags       8086   - 2       8087 - 2
;              NDP control word        V20    - 3      80187 - 3
;              (user configuration)    V30    - 4      80287 - 4
;              preserved               80188  - 5    80287XL - 5
;                                      80186  - 6      80387 - 6
;                                      80286  - 7     IIT2C87 - 7
;                                      80386  - 8     IIT2C87 - 8
;                                      80486  - 9     IIT3C87 - 9
;                                      80486sx-10      80486 -10
;
;-------------------------------------------------------------------------------

            PUBLIC  CPU_NDP_TST
            ASSUME  CS: _TEXT, DS: _TEXT

OriginlCtrl EQU     [BP-2]              ; caller's NDP control word
Ctrl        EQU     [BP-4]              ; saved NDP control word for tests
Stat        EQU     [BP-6]              ; saved NDP status word for tests

CPU_NDP_TST PROC    NEAR
            PUSH    BP                  ; save caller's frame pointer
            MOV     BP, SP              ; make new frame pointer
            SUB     SP, 6               ; create local variables
            PUSH    SI                  ; save register
```

```
            PUSH    DI                  ;  variables (C, Modula)
            PUSH    DS                  ; save caller's data segment
            PUSH    CS                  ; make code segment
            POP     DS                  ;  also data segment
            PUSH    SP                  ; test updating
            POP     AX                  ;  of stackpointer
            CMP     AX, SP              ; stackpointer updated before push ?
            JES     $286_386            ; no, must be 286, 386 or 486
            MOV     AX, 1               ; try to shift
            MOV     CL, 33              ;  accu 33 times
            SHL     AX, CL              ; shift count masked off ?
            JNZS    $186_188            ; yes, must be 186 or 188
            PUSHA                       ; PUSHA executed on 88/86 as JMP --+
            STC                         ; carry set if V20 or V30          |
            JCS     $V20_V30            ; yes, must be V20 or V30  <-------+
            PUSHF                       ; save flags
            POP     AX                  ; pop flags into AX
            AND     AX, 0FFFH           ; clear bits 12-15 of flag register
            PUSH    AX                  ; put new flags in stack
            POPF                        ; pop into flag register
            PUSHF                       ; put flags on stack
            POP     AX                  ; get flags
            AND     AH, 0F0H            ; test if all bits
            CMP     AH, 0F0H            ;  in highest nibble set
            JES     $88_86              ; all bits in highest nibble set
            XOR     DL, DL              ; failed all tests, unknown CPU
            JMPS    $ndp_test           ; go and test ndp
$88_86:     MOV     DL, 1               ; else its an 88 or 86
            JMPS    $queue_test         ; decide wether 88 or 86
$V20_V30:   POPA                        ; remove pushed bytes
            MOV     DL, 3               ; its an V20 or V30
            JMPS    $queue_test         ; decide wether V20 or V30
$186_188:   MOV     DL, 5               ; 188/186
            JMPS    $queue_test         ; decide wether 188 or 186
$286_386:   MOV     DL, 7               ; 286, 386 or 486
            PUSH    7000H               ;  try to set
            POPF                        ;   IOPL and NT fields
            PUSHF                       ;    in bit 12-14
            POP     AX                  ;     of flag register
            TEST    AX, 7000H           ; bits cannot be set in 286 real mode
            JZS     $ndp_test           ; bits not set --> 286
            INC     DX                  ; CPU is an 386 (DL = 8) or 486
            MOV     EBX, ESP            ; save current stackpointer to align it
            AND     ESP, 0FFFFFFFCh     ; align stack to avoid AC fault
            PUSHFD                      ; save EFLAGS
            POP     EAX                 ; get EFLAGS from stack
            MOV     ECX, EAX            ; original value of EFLAGS
            XOR     EAX, 40000H         ; toggle AC bit in EFLAGS
            PUSH    EAX                 ; copy new value
            POPFD                       ;  to EFLAGS
            PUSHFD                      ; get new EFLAGS value
            POP     EAX                 ; put into EAX
            XOR     EAX, ECX            ; test if AC bit could be changed
            SHR     EAX, 18             ; EAX = 0 if 386, 1 if 486
            PUSH    ECX                 ; restore original
            POPFD                       ;  value of EFLAGS
            MOV     ESP, EBX            ; restore original stack pointer
```

```
            AND     EAX, 1              ; check CPU code in EAX
            JZS     $ndp_test           ; if 0, must be 386
            INC     DX                  ; it's a 486 (CPU = 9)
            JMPS    $ndp_test           ; check for NDP (486 <-> 486sx)
$queue_test:CLI                         ; int service disturbs prefetch queue
            MOV     BX, OFFSET $patch   ; load patch adress into BX
            MOV     BYTE PTR [BX], 42h  ; preset with opcode for INC DX
            MOV     AL, 90H             ; patch in a NOP (opcode 90h)
            MOV     CL, 31              ; rotate register 31 times to use up
            ROL     AH, CL              ;  time so prefetch queue can be filled
            MOV     BYTE PTR [BX], AL   ; insert NOP at label $patch
            NOP                         ; fill
            NOP                         ;  prefetch
            NOP                         ;   queue
            NOP                         ;    with NOPs
$patch:     INC     DX                  ; patched to nop on i88, i188, and V20
            STI                         ; test complete, interrupts OK now
$ndp_test:  FNSTCW  [OriginlCtrl]       ; save caller's NDP configuration
            XOR     DH, DH              ; assume no coprocessor
            XOR     AX, AX              ; clear register
            OUT     0F0h, AL            ; clear error signal
            FNINIT                      ; initialize coprocessor
            MOV     WORD PTR [Ctrl], AX ; clear status variable
            NOT     AX                  ; load all 1's
            MOV     WORD PTR [Stat], AX ; initialize status variable to all 1's
            FNSTCW  [Ctrl]              ; store NDP control word
            MOV     AX, [Ctrl]          ; get control word
            AND     AX, 0F3Fh           ; extract RC, PC and exception masks
            CMP     AX, 033Fh           ; RC=0, PC=3, masks=3F ?
            JES     $cont_test          ; NDP seems to be present, more tests
            CMP     DL, 9               ; no NDP. CPU = 486 ?
            SBB     DL, -1              ; yes: 486 w/o NDP = 486sx
            JMPS    $end_test           ; no further NDP testing required
$cont_test: FNSTSW  [Stat]              ; store NDP status
            TEST    WORD PTR [Stat],383Fh; stack top & exceptions must be clear
            JNZS    $end_test           ; ST & exceptions not clear -> no NDP
            MOV     DH, 1               ; coprocessor is at least emulator (=1)
            CMP     DL, 7               ; is CPU 80286 or higher ?
            JBS     $no_emulat          ; no, emulation impossible
            SMSW    AX                  ; get machine status word
            TEST    AL, 4               ; test if EM bit of MSW set
            JNZS    $ndp_exit           ; yes, NDP only emulated
$no_emulat: MOV     DH, 2               ; coprocessor is at least 8087 (=2)
            FLD1                        ; load 1.0
            WAIT                        ; needed for 8087
            FLDZ                        ; load 0.0
            WAIT                        ; needed for 8087
            FDIV                        ; 1.0 / 0.0 = +infinity
            WAIT                        ; needed for 8087
            FLD     ST(0)               ; duplicate +infinity
            WAIT                        ; needed for 8087
            FCHS                        ; generate -inifinity
            WAIT                        ; needed for 8087
            FCOMPP                      ; compare infinities and clear NDP stk
            WAIT                        ; needed for 8087
            FSTSW   WORD PTR [Stat]     ; save condition codes
            MOV     AX, [Stat]          ; load condition codes
```

```
            SAHF                        ; transfer CC into CPU flags
            JNES    $187_387            ; 187, C287, 287XL, 387, 486 if n. eq.
            CMP     DL, 7               ; is CPU >= 286 ?
            JBS     $ndp_exit           ; no, coprocessor is 8087
            MOV     DH, 4               ; coprocessor is 287
            JMPS    $chk_iit            ; check for IIT coprocessors
$187_387:   CMP     DL, 7               ; is CPU >= 286 ?
            JAES    $C287_387           ; yes, NDP is 287A,287XL,387,486,487sx
            MOV     DH, 3               ; coprocessor is 187
            JMPS    $ndp_exit           ; store CPU and NDP code
$C287_387:  CMP     DL, 8               ; is CPU >= 386 ?
            JAES    $387_486            ; yes, NDP is 387,486,487sx
            MOV     DH, 5               ; coprocessor is 287A or 287XL
            JMPS    $chk_iit            ; check for IIT coprocessors
$387_486:   CMP     DL, 9               ; is CPU >= 486 ?
            JBS     $387                ; no, NDP is 387 / 387sx
            MOV     DH, 10              ; NDP = 486
            JMPS    $ndp_exit           ; test done
$387:       MOV     DH, 6               ; coprocessor is 387 or 387sx
$chk_iit:   FINIT                       ; initialize coprocessor
            FWAIT                       ; wait until operation complete
            FLD     TBYTE PTR [TstVal]  ; load denormal number
            FWAIT                       ; wait until operation complete
            FADD    ST(0), ST           ; result is a normal
            FWAIT                       ; wait until operation complete
            FSTSW   AX                  ; get status word of NDP
            AND     AX, 2               ; flag for denormal exception set ?
            JNZS    $ndp_exit           ; Intel types set denormal exception
            ADD     DH, 3               ; set IIT coprocessor types
$ndp_exit:  FLDCW   [OriginlCtrl]       ; restore caller's NDP configuration
$end_test:  XCHG    AX, DX              ; get result into AX
            POP     DS                  ; restore callers data segment
            POP     DI                  ; restore register
            POP     SI                  ;  variables (C, Modula)
            MOV     SP, BP              ; remove local variables
            POP     BP                  ; restore caller's frame pointer
            RET                         ; done

TstVal      DT      1                   ; denormal number for IIT test

CPU_NDP_TST ENDP


IntroMsg    DB      79 DUP ('-'), 10, 13
            DB      20 DUP (' '), 'C P U - T e s t     (c) 1988-1991 N.J.',10,13
            DB      79 DUP ('-'), 10, 13, 10, 13, '$'

CPUMsg      DB      'CPU ist $'
NDPMsg      DB      'NDP ist $'
EOL         DB      10, 13, '$'

Processor   DB      'nicht identifizierbar$'
            DB      'Intel 8088           $'
            DB      'Intel 8086           $'
            DB      'NEC V20              $'
            DB      'NEC V30              $'
            DB      'Intel 80188          $'
```

```
            DB      'Intel 80186          $'
            DB      'Intel 80286          $'
            DB      'Intel 80386 / 80386sx$'
            DB      'Intel 80486          $'
            DB      'Intel 80486SX        $'

Coprocessor DB      'nicht vorhanden       $'
            DB      'Emulator via INT 7    $'
            DB      'Intel 8087            $'
            DB      'Intel 80187           $'
            DB      'Intel 80287           $'
            DB      'Intel 80287A / 80287XL$'
            DB      'Intel 80387 / 80387sx $'
            DB      'IIT 2C87              $'
            DB      'IIT 2C87              $'
            DB      'IIT 3C87              $'
            DB      'eingebaut             $'

_TEXT       ENDS
            END     MainStart
```

aus mc 11/91 – Seite 152-161