Basic-Programm auf TRAB

Haben Sie schon einmal daran gedacht, sich eventuell einen Compiler zuzulegen, um Basic-Programme schneller zu machen? Sie wissen aber vielleicht noch zuwenig über Compiler, um sich den richtigen auszusuchen. Wir stellen Ihnen deshalb vier Typen vor.

Wir haben vier der bekanntesten Compiler getestet: Petspeed, Austro-Speed, BASS- und Ex-Basic Level II-Compiler. Bevor wir uns jedoch mit den »Prüflingen« genauer beschäftigen, wollen wir erst ein wenig auf die praktischen Grundlagen der Compiler eingehen.

Wenn man Compiler hört, denkt man unwillkürlich immer zuerst an den Geschwindigkeitsgewinn, den diese Programme bringen sollen. In der Tat ist der Hauptzweck von Compilern in der zeitlichen Optimierung eines Programmlaufs zu sehen, damit die Ausführungszeit verkürzt wird. So gibt es demnach viele Compiler, deren Hauptkonzept in der Geschwindigkeitserhöhung liegt.

Über diese Tatsachen darf man aber auch andere Eigenschaften von Compilern nicht vergessen. Compiler erhöhen zwar in der Regel die Geschwindigkeit der Basic-Programme, sie vertragen aber unter Umständen gewisse syntaktische Konstruktionen nicht, die beim Interpreter ohne weiteres funktionieren. Außerdem werden kleinere Programme in der Regel durch das Compilieren um einiges länger als sie ursprünglich waren.

Da wir gerade bei der Programmlänge sind, soll an dieser Stelle auch gleich auf die zwei verschiedenen Arten von Compilern eingegangen werden. Die erste Art ist der Assemblercode-Compiler, der echten Maschinencode erzeugt und damit maximalen Geschwindigkeitsgewinn bringt. Der Nachteil dieser Methode ist, daß das compilierte Programm (Compilat) in der Regel

Hersteller	Austro-Speed	BASS	Exbasic Level II-Compiler	Petspeed
Preis (ca.)	298,—	198,—	298,—	149,—
Lieferumfang	1 Diskette 1 Handbuch	3 Disketten 2 dicke Handbücher	1 Diskette 1 Handbuch	1 Diskette 3 Seiten Einweisung
Dokumentation	gut	ausgezeichnet	befriedigend	mangelhaft
ungefähre Länge des Programms	63 Blocks	263 Blocks	290 Blocks	300 Blocks
Anzahl PASSES	2	2 + 2 Assembler (wird nicht mitgeliefert)	2 + 2 Assembler (integriert)	4
Programmschutz möglich?	ja	nein	nein	nein
Integer Arithmetik	ja	ja	ja	ja
Erweiterungen möglich?	ja	ja	ja	nein
variables DIM möglich?	ja	nein	nein	nein
Automatisches DIM auf 11 Elemente	ja	nein	nein	ja
Compilierdauer EDDI (14 Blocks)	3 min	7,10 min + 9 min ASSI	12 min	7,30 min
Diskettenwechsel beim Compilieren	nein	ja, 2mal	ja, 4mal	nein
Anzahl der erzeugten Files	2	10	1	2
Länge des Compilats (EDDI)	32 Blöcke	39 Blöcke	39 Blöcke	42 Blöcke
Compilertyp	—	Adreßcode + Assemblercode	Assemblercode	Assemblercode

Tabelle 1. Vier Compiler im Vergleich

			Basic	Ex- + BASS	Pet-speed	Austro-Speed
Test	1a	Einlesen (100 Werte)	1,53	0,67	0,50	0,46
Test	1b	Sortieren	64,16	24,85	8,23	16,25
Test	1c	Anzeigen	0,55	0,67	0,10	0,33
Test	1	Gesamt (a + b + c)	66,25	26,18	8,83	17,05
Benchmark	1	FOR NEXT	1,83	1,01	0,28	1,10
	2	430 K = K + 1 440 IF K <1000 THEN 430	13,22	2,16	0,72	1,38
	3	A = K/K x K + K — K	11,25	4,70	5,54	4,38
	4	A = K/2 x 3 + 4 — 5	12,00	6,51	6,42	5,32
	5	GOSUB RETURN	16,65	0,45	0,18	0,15
	6	FOR L = 1 TO 5:NEXT L	14,47	6,34	1,93	6,50
	7	FOR L = 1 TO 5:M(L) = A:NEXT	24,85	8,80	2,50	3,95
	8	A = K!2 B = LOG(K) C = SIN(K)	112,30	102,19	93,25	101,98
Gesamtzeit (Test 1 + Bench 1 bis 8)			390,86 ≙100%	198,53 ≙ 50,79%	145,52 ≙ 37,23%	172,96 ≙ 44,25%

Tabelle 2. Die vier Compiler im Zeitvergleich. Im Test 1 wurden mit der RND-Funktion 100 Zeichen ermittelt, sortiert und nebeneinander ausgegeben. Die Benchmark-Tests 1 bis 8 waren so aufgebaut, daß die Zeit für die angegebenen Befehle selbst ermittelt werden konnten. Deshalb ist die Gesamtzeit nicht identisch mit der Summe der einzelnen Testzeiten. Jeder Befehl (Benchmark 1 bis 8) wurde 1000mal durchgeführt (siehe Listing 1).

gebracht-Compiler im Test

um einiges länger wird als das ursprüngliche Basic-Programm.

Bei der zweiten Art von Compilern (sogenannte Adreßcode-Compilern) geht man deshalb einen anderen Weg. Hier wird der Basic-Text in eine Liste von Sprungadressen umgewandelt, was Speicherplatz spart, aber andererseits wieder auf Kosten der Geschwindigkeit geht.

Wie Sie sehen, gibt es den perfekten Compiler nicht. Entweder ist ein Programm sehr schnell, dann ist es länger, oder ein Programm ist nicht ganz so schnell, dafür wird es kürzer. Einen guten Compiler erkennt man also am richtigen Kompromiß zwischen Geschwindigkeit und Länge des Compilats. Ein weiteres Qualitätsmerkmal für Compiler ist die Dauer des Compilierens. Dieser Vorgang ist zwar in der Regel einmalig, da man nur ein wirklich fertiges Programm compilieren wird, er sollte sich dennoch in vernünftigen Grenzen bewegen.

Wenn wir uns jetzt gleich einmal mit den speziellen Eigenschaften der Testkandidaten vertraut machen, sollten Sie immer an diese Merkmale denken. Das interessante an den getesteten Compilern ist nämlich, daß sie fast alle nach unterschiedlichen Kriterien entwickelt wurden.

Der Austro-Speed-Compiler

Der erste Compiler, mit dem wir uns beschäftigen wollen, ist der Austro-Speed von Commodore. Austro-Speed ist eine verbesserte Version des Austro-Comp für die CBM-Systeme. Im Lieferumfang sind eine Diskette mit dem Programm, ein Handbuch mittleren Umfangs und ein Dongle enthalten. Ein Dongle ist ein programmschutztechnischer Hardwarezusatz, dessen Vorhandensein abgefragt wird und ohne das der Compiler nicht läuft, (in diesem Fall ein Stecker für den User-Port). Als ich mir das Inhaltsverzeichnis der Diskette ansehen wollte, erlebte ich sofort eine Überraschung. Der ganze Compiler besteht aus einem einzigen Programm mit einer Länge von 63 Blöcken (15,75 KByte).

Als Testprogramm für den Compiliervorgang diente der Disk-Monitor EDDI aus der 64'er, Ausgabe 10/1984. Dieses Programm hat den Vorteil, daß es nicht zu kurz ist. Außerdem ist der Programmierstil an vielen Stellen alles andere als sauber. Haben Sie übrigens bemerkt, daß sich bei EDDI ein Fehler eingeschlichen hat? In der Zeile 1070 ist die IF-Abfrage überflüssig und zeigt außerdem auf eine nicht vorhandene Zeile (1 090). Diese Abfrage stört den Programmablauf nicht im geringsten, wir können jedoch gespannt auf die Reaktionen der Compiler sein, wenn sie diese Zeile abarbeiten. Jetzt aber wieder zurück zu Austro-Speed.

Bevor man den Compiler in den Computer lädt, muß man darauf achten, daß das Dongle auf den User-Port des C 64 gesteckt ist. Für den Vorgang des Compilierens ist es in der Regel notwendig, daß viel Platz auf der Diskette mit dem Basic-Programm vorhanden ist, da die Compiler eine Menge Dateien erstellen, die jedoch nach dem Compilieren normalerweise wieder gelöscht werden. Das ist besonders bei langen Programmen zu beachten.

Nachdem wir auch diese letzte Vorbereitung ausgeführt haben, starten wir den Compiler. EDDI besteht aus 14 Blöcken auf Diskette. Nach genau drei Minuten ist Austro-Speed mit der Arbeit fertig, und das Compilat steht zur Verfügung.

Während des Compilierens hat Austro-Speed sogar den Programmfehler entdeckt und angezeigt, jedoch seine Arbeit nicht unterbrochen.

Wenn wir uns die Diskette betrachten, so sehen wir, daß unser Programm an Länge ganz erheblich zugenommen hat. Es besteht jetzt aus 32 Blöcken und ist damit mehr als doppelt so lang geworden.

Zur Erleichterung einer eventuell noch folgenden Korrektur bei einem Fehler, legt Austro-Speed noch ein weiteres File ab, das die neuen Speicheradressen sämtlicher Programmzeilen enthält.

Da ein Compiler ein Basic-Programm nicht auf einmal übersetzt, sondern dafür mehrere Durchläufe benötigt, kann man auch anhand der Anzahl dieser Durchläufe (Durchlauf = Pass) einen Compiler charakterisieren. Austro-Speed benötigt für seine Arbeit zwei dieser Durchläufe; man bezeichnet ihn deshalb auch als 2-Pass-Compiler.

Unser nächstes Programm heißt BASS und kommt von gmbsoft. Der Unterschied zu Austro-Speed wird sofort deutlich, wenn man sich den Lieferumfang betrachtet. Er besteht aus drei Disketten und zwei dicken Handbüchern. Bei einer der Disketten handelt es sich um eine Demodiskette, die unter anderem ein Sortierprogramm enthält, um die Geschwindigkeit eines Compilats zu demonstrieren.

Der BASS-Compiler

Wie bei Austro-Speed habe ich auch hier erst einmal das Directory gelistet. Hat man die Länge der Austro-Speed noch vor Augen, so trifft einen hier fast der Schlag. Bei BASS handelt es sich um den reinsten »Mammutcompiler«. Er arbeitet zwar auch nur mit zwei Durchläufen, jedoch besteht hier allein schon der Pass 1 aus einem über 100 Blöcken langen Programm, der von Pass 2 noch übertroffen wird.

Da es bei einer solchen Komplexität kaum möglich ist, einfach »drauflos« zu arbeiten, sollte man sich erst einmal eines der beiden Handbücher vornehmen (das dünnere, versteht sich). Und hier erlebt man auch gleich die erste Überraschung. Der Compiler erzeugt bei seiner Arbeit kein lauffähiges Programm, sondern nur eine stattliche Anzahl von Dateien (insgesamt 10), die editierfähig sind und noch einen Assemblierlauf benötigen, bevor ein fertiges Programm daraus entsteht.

Diese Konzipierung hat aber natürlich einen Sinn. Bei gmbsoft hat man versucht, einen Compiler herzustellen, der so offen wie möglich arbeitet; das heißt der Programmierer soll auch nach dem Compilieren noch die Möglichkeit haben, optimierende Eingriffe und Änderungen an seinem Programm vorzunehmen. Zu diesem Zweck eignet sich ein editierfähiger Code natürlich eher, als der »Spaghetticode« in einem fertig compilierten Programm.

Durch diese sehr positive Eigenschaft des Programms angeregt, geht man erneut an die Arbeit, aber — wo ist der Assembler?

Es stellt sich heraus, daß der Assembler nicht im Lieferumfang des BASS enthalten ist; er muß extra besorgt werden. Wie im Handbuch empfohlen, beschafft man sich also das Assemblerpaket ASSI von Dirk Zabel (siehe Assembler-Test in Aus-

gabe 1/85), um endlich ein fertiges Compilat zu erhalten.

Den gewohnten Richtlinien folgend kopiert man das Testprogramm EDDI auf eine leere Diskette. Aber es müssen noch einige Handgriffe ausgeführt werden, bis das Programm endlich fertig compiliert sein wird. Zuerst muß noch eine Bibliothek auf die Programmdiskette kopiert werden, die der Assembler benötigt, und dann kann es endlich losgehen. Da BASS, wie schon erwähnt, ziemliche Ausmaße besitzt, lag es natürlich nahe, einmal Hypra-Load heranzuziehen, und siehe da — BASS arbeitet mit Hypra-Load einwandfrei zusammen, was die Compilierzeit insgesamt erheblich verkürzt.

Trotz aller dieser »Vorabhandgriffe« entpuppt sich der BASS als Langweiler. Für das reine Compilieren von EDDI benötigte er 7,10 Minuten. Das nachfolgende Assemblieren benötigt noch einmal neun Minuten, so daß man insgesamt mindestens eine halbe Stunde beschäftigt ist (alles eingerechnet).

EDDI wird von BASS einwandfrei verarbeitet; der Fehler in Zeile 1070 wurde jedoch während der Compilation nicht entdeckt. Er wurde erst vom Assembler registriert und äußerte sich in einem »UNDEFINED SYMBOL ERROR«. Auch in diesem Fall wurde die Arbeit jedoch ordnungsgemäß zu Ende geführt.

Der Exbasic Level II-Compiler

Der Exbasic Level II-Compiler von Interface Age machte bei Erhalt der Lieferung wieder einen ganz anderen Eindruck als der BASS. Diese beiden Compiler sind dabei fast identisch. Was den Namen dieses Compilers betrifft, so erscheint er vielleicht etwas irreführend. Der Exbasic Level II-Compiler hat mit Exbasic Level II ebensoviel oder ebensowenig zu tun, wie fast alle anderen Compiler dieses Tests auch.

Der Name soll eine Eigenschaft dieses Compilers verdeutlichen, die Austro-Speed und BASS jedoch ebenso besitzen: die Verarbeitung von Erweiterungen (sogenannte Extensions).

Das heißt nichts weiter, als daß diese Programme in der Lage sind, auch Befehle, die im Standard-Basic V 2.0 von Commodore nicht vorkommen, zu verarbeiten. Wenn diese Compiler zum Beispiel auf einen Befehl des Exbasic Level II stoßen, so wird dieser Befehl nicht compiliert, sondern im non-compiled-Code angelegt. Wird ein so compiliertes Programm jetzt zum Beispiel unter Exbasic gestartet, so übergibt das Steuerprogramm, das jedes Compilat enthält, den entsprechenden Befehl einfach dem Interpreter zur Ausführung und macht anschließend weiter.

Der Unterschied zwischen dem Exbasic Level II-Compiler und BASS besteht lediglich in der Dicke des Handbuchs, im Preis und in der Tatsache, daß der Exbasic-Compiler um den notwendigen Assembler erweitert wurde. Auf der Diskette erkennt man das an Pass 3 und Pass 4, die der BASS nicht besitzt.

Im Test zeigte der Exbasic-Compiler demzufolge auch die gleichen Eigenschaften wie der BASS, auf die ich gleich noch zu sprechen komme.

Das Compilieren und Assemblieren wird vom Exbasic Level II-Compiler um einiges schneller erledigt, als von BASS. Außerdem spart man sich das Kopieren der Bibliothek. Für EDDI wurde eine Zeit von 12 Minuten gemessen, was jedoch immer noch viermal so lang ist, wie beim Austro-Speed. Durch das jeweilige Nachladen der einzelnen Programmteile ergibt sich außerdem noch zusätzlich ein viermaliger Diskettenwechsel.

Da der BASS- und Exbasic Level II-Compiler nahezu identisch sind, soll auch gleich einmal auf die negativen Seiten der beiden Programme eingegangen werden.

Wie Sie vielleicht wissen, kann man im Commodore-Basic sowohl mit Fließkomma- als auch mit Integerwerten rechnen. Der Unterschied zeigt sich in den Variablennamen, wobei die Integervariablen durch ein »%« am Ende gekennzeichnet sind. Die Integerarithmetik läßt nur Zahlenbereiche von —32768 bis 32767 zu; hat aber dadurch den Vorteil, daß weniger Speicherplatz und geringerer Zeitaufwand beim Rechnen mit diesen Werten erforderlich ist. Im Gegensatz zu Fließkommawerten benötigen Integerzahlen normalerweise nur 2 Byte Speicherplatz pro Wert; das sind 3 Byte weniger als bei Fließkommaberechnungen.

Das Commodore-Basic hat jetzt den Nachteil, daß keine Integerroutinen existieren, die die Berechnungen durchführen. Alle Zahlen werden deshalb zuerst ins Fließkommaformat umgewandelt und dann verrechnet. Anschließend konvertiert der Interpreter diese Werte wieder zurück.

Alle Vorteile, die die Integerzahlen also haben, werden durch den Interpreter zunichte gemacht. Die Entwickler von Compilern haben dieses Manko sehr wohl erkannt, und deshalb sind alle getesteten Produkte auch mit eigenen Integerroutinen ausgestattet, die einen enormen Zeitgewinn versprechen. Es ist somit möglich, auch beim Interpreter verbotene Konstruktionen, wie eine Integerschleife, zu verwenden. FOR X% = 0 TO 1000 : NEXT X% ist zum Beispiel beim Interpreter nicht gestattet und wird mit einem »SYNTAX ERROR« quittiert.

Oben wurde schon auf Nachteile vom BASS- und Exbasic Level II-Compiler hingewiesen. Bei diesen beiden Produkten gibt es eine solche Konstruktion ebenfalls nicht. Hier muß man alle Variablen, die man als Integer verwenden möchte, mit direkten Befehlen an den Compiler als solche vordefinieren.

Als weiterer Minuspunkt zeigte sich bei diesen beiden Produkten die »Intoleranz« gegenüber der Syntax von Programmen.

Bei Basic-Programmen ist es üblich, eine Dimensionierung von Variablen, sofern das Feld nicht mehr als elf Elemente benötigt, zu unterlassen. Der Interpreter übernimmt diese Dimensionierung automatisch. Bei besagten beiden Compilern ist dies jedoch nicht der Fall und führt während des Compilierens zu einer Fehlermeldung in Form einer Nummer. Da diese beiden Compiler jedoch mit einer Fülle an Fehlermeldungen ausgestattet sind, erwies es sich bei dem Exbasic-Compiler als äußerst nachteilig, daß er keinen Fehlertext, sondern nur die Nummer der Meldung ausgibt. Wie es sich zeigte, enthält das Handbuch zwar eine Aufstellung aller Fehlermeldungen; diese aber wiederum ohne Nummer (im Gegensatz zum BASS), so daß man spekulativ schon sehr auf Zack sein muß, um zu erfahren, was für ein Fehler denn nun beanstandet wurde.

Auch das oft übliche Belegen einer Zeile mit einem Doppelpunkt »:«, um ein Programm lesbarer zu gestalten, wurde nicht toleriert und führte zu einer Fehlermeldung.

Insgesamt also eine Reihe von Nachteilen, die einem die Arbeit mit einem Compiler sicherlich schwerer machen, zumal wir an den anderen Testkandidaten feststellen konnten, daß es auch anders geht. Austro-Speed ist syntaktisch sehr großzügig. Das einzige, was er und Petspeed nicht vertragen, sind ver-

```
10 INPUT"WIEVIEL WERTE";Q1                          <044>
20 DIM FF$(1000)                                    <092>
30 T1=TI                                            <242>
40 FOR I=1 TO Q1                                    <251>
50 FF$(I)=CHR$(INT(RND(0)*26+64))                   <176>
60 NEXT                                             <190>
70 T2=TI                                            <027>
80 GOSUB 930                                        <121>
90 T3=TI                                            <048>
100 PRINT:PRINT                                     <208>
110 FOR I=1 TO Q1                                   <065>
120 PRINT FF$(I);                                   <150>
130 NEXT                                            <004>
140 T4=TI                                           <099>
150 GOSUB 330                                       <185>
160 PRINT:PRINT                                     <012>
170 PRINT"ANZAHL WERTE= "Q1:PRINT                   <094>
180 PRINT"EINLESEZEIT={3SPACE}"(T2-T1)/60           <041>
190 PRINT"SORTIERZEIT={3SPACE}"(T3-T2)/60           <088>
200 PRINT"ANZEIGEZEIT={3SPACE}"(T4-T3)/60           <063>
210 PRINT"GESAMT ZEIT={3SPACE}"(T4-T1)/60           <005>
215 PRINT:PRINT                                     <067>
220 PRINT"BENCH1{5SPACE}={3SPACE}"(B1-B0)/60
                                                    <123>
230 PRINT"BENCH2{5SPACE}={3SPACE}"(B2-B1)/60
                                                    <136>
240 PRINT"BENCH3{5SPACE}={3SPACE}"(B3-B2)/60
                                                    <149>
250 PRINT"BENCH4{5SPACE}={3SPACE}"(B4-B3)/60
                                                    <162>
260 PRINT"BENCH5{5SPACE}={3SPACE}"(B5-B4)/60
                                                    <176>
270 PRINT"BENCH6{5SPACE}={3SPACE}"(B6-B5)/60
                                                    <189>
280 PRINT"BENCH7{5SPACE}={3SPACE}"(B7-B6)/60
                                                    <202>
290 PRINT"BENCH8{5SPACE}={3SPACE}"(B8-B7)/60
                                                    <215>
300 PRINT"GESAMT ZEIT={3SPACE}"(B8-T1)/60           <082>
310 IF F=1 THEN PRINT:PRINT:PRINT:PRINT:PRINT#1
   :CLOSE 1:END                                     <108>
320 OPEN 1,4:CMD 1:F=1:GOTO 170                     <055>
330 REM--------------------------                   <198>
340 REM     BENCHMARKS                              <194>
350 REM-------------------1-------                   <222>
360 REM                                             <248>
370 B0=TI                                           <052>
380 FOR K=1 TO 1000                                 <145>
390 NEXT K                                          <084>
400 B1=TI                                           <083>
410 REM-------------------2---------                <117>
420 K=0                                             <210>
430 K=K+1                                           <210>
440 IF K<1000 THEN 430                              <065>
450 B2=TI                                           <134>
451 REM-------------------3---------                <159>
452 K=0                                             <242>
453 K=K+1                                           <233>
454 A=K/K*K+K-K                                     <223>
455 IF K<1000 THEN 453                              <085>
456 B3=TI                                           <141>
460 REM-------------------4---------                <169>
470 K=0                                             <004>
480 K=K+1                                           <004>
490 A=K/2*3+4-5                                     <165>
500 IF K<1000 THEN 480                              <130>
510 B4=TI                                           <196>
520 REM-------------------5---------                <231>
530 K=0                                             <065>
540 K=K+1                                           <065>
550 A=K/2*3+4-5                                     <226>
560 GOSUB 600                                       <085>
570 IF K<1000 THEN 540                              <198>
580 B5=TI                                           <012>
590 GOTO 620                                        <113>
600 RETURN                                          <232>
610 REM-------------------6---------                <066>
620 K=0                                             <155>
630 K=K+1                                           <155>
640 A=K/2*3+4-5                                     <060>
650 GOSUB 710                                       <177>
660 FOR L=1 TO 5                                    <031>
670 NEXT L                                          <110>
680 IF K<1000 THEN 630                              <052>
690 B6=TI                                           <123>
700 GOTO 730                                        <225>
710 RETURN                                          <086>
720 REM-----;---------------7--------               <177>
725 DIM M(10)                                       <092>
730 K=0                                             <009>
740 K=K+1                                           <009>
750 A=K/2*3+4-5                                     <170>
760 GOSUB 830                                       <034>
770 FOR L=1 TO 5                                    <142>
780 M(L)=A                                          <236>
790 NEXT L                                          <231>
800 IF K<1000 THEN 740                              <143>
810 B7=TI                                           <245>
820 GOTO 850                                        <093>
830 RETURN                                          <207>
840 REM-------------------8--------                 <043>
850 K=0                                             <130>
860 K=K+1                                           <130>
870 A=K↑2                                           <135>
880 B=LOG(K)                                        <191>
890 C=SIN(K)                                        <205>
900 IF K<1000 THEN 860                              <022>
910 B8=TI                                           <090>
920 RETURN                                          <041>
930 REM ----------------------------                <077>
940 REM -         UP SORTIEREN         -            <248>
950 REM ----------------------------                <097>
960 REM -    Q1     = ANZAHL ELEMENTE -             <120>
970 REM -    FF$() = SORTIERFELD      -              <055>
980 REM ----------------------------                <127>
990 :                                               <027>
1000 REM JJ,LL = LAUFVARIABLE                       <125>
1010 REM Q2$   = ZWISCHENSPEICHER                   <038>
1020 :                                              <057>
1030 REM SORTIERT DAS FELD FF$ MIT                  <162>
1040 REM Q1 ELEMENTEN IN ALPHABE -                  <115>
1050 REM TISCHER FOLGE                              <044>
1060 :                                              <098>
1070 FOR JJ=1 TO Q1-1                               <044>
1080 FOR LL=JJ+1 TO Q1                              <156>
1090 IF FF$(JJ)<=FF$(LL)THEN 1130                   <208>
1100 Q2$=FF$(JJ)                                    <062>
1110 FF$(JJ)=FF$(LL)                                <058>
1120 FF$(LL)=Q2$                                    <086>
1130 NEXT LL                                        <136>
1140 NEXT JJ                                        <142>
1150 RETURN                                         <016>
```

Listing 1. Die in Tabelle 2 zusammengefaßten Ergebnisse wurden mit diesem Programm ermittelt. Die Zeit für zum Beispiel 1000 GOSUB-RETURN erhält man, indem die Zeit für Benchmark 4 von der Zeit für Benchmark 5 abgezogen wird.

schachtelte MID$-Statements, die man aber generell bei der Arbeit mit Compilern vermeiden sollte.

Was angenehm überrascht, ist die Tatsache, daß Austro-Speed sogar die variable Dimensionierung (zum Beispiel DIM A (B), zuläßt. Eine eigentlich gar nicht selbstverständliche Eigenschaft, da Compiler auf das feste Anlegen von Variablenfeldern angewiesen sind und somit deren Ausmaße beim Compilieren feststehen müssen.

Doch nun zum Petspeed, dessen Lieferumfang aus drei Blättern Druckerpapier und einer Diskette bestand. Das »3-Blatt-Handbuch« strotzt nur so von Fehlern und macht einen gleich einmal auf eine nette Überraschung gefaßt. Das Directory der beigefügten Diskette ist nur über Spezialprogramme zu listen. Dieser (unnötige) »Scherz« hätte unter normalen Umständen sicher nichts ausgemacht. Das Sonderbare an Petspeed ist nur, daß man sein Basic-Programm auf die Systemdiskette kopieren muß, damit der Compiler arbeiten kann. Aus diesem Grund hat mich dieser »Gag« ziemlich verärgert, da er die Möglichkeit eines Bedienungsfehlers geradezu herausfordert.

Hat man auch hier EDDI in die compilierfähige Form gebracht, so kann es losgehen. Der Petspeed ist ein 4-Pass-Compiler, was schon einmal gewisse Erwartungen bezüglich der Leistung weckt.

Nach 7 ½ Minuten ist die Arbeit an EDDI abgeschlossen, und wir erhalten, wie schon bei Austro-Speed, zwei Programm-Files zurück. Eines der beiden Programme ist dabei wieder eine Korrekturerleichterung, die alle vorhandenen Zeilennummern mit deren neuen Adressen enthält.

Der Petspeed-Compiler

Petspeed ist also beim Compilieren hinter Austro-Speed der zweitschnellste Compiler dieses Tests. Der einzige Nachteil besteht in der Tatsache, daß das zu compilierende Programm auf die Systemdiskette kopiert werden muß. Erstens ist damit die Wahrscheinlichkeit einer Panne mit der Originaldiskette größer, und zweitens wird der Platz zum Compilieren ganz erheblich eingeschränkt, da Petspeed schon über 200 Blöcke für sich beansprucht. So darf das zu übersetzende Basic-Programm auch nicht länger als 80 Blocks sein.

Sieht man sich die Gesamtlänge der vier Compilate aller Compiler an, so erkennt man, daß sich BASS-, Exbasic Level II-Compiler und Austro-Speed in etwa entsprechen. Petspeed hat mit Abstand das längste File hinterlassen, was sich auch bei der weiteren Arbeit mit diesem Compiler immer wieder zeigen wird.

Sie werden jetzt natürlich gespannt auf die Ergebnisse des Compilierens sein. Was ist eigentlich aus dem einstmals so langsamen Basic-Programm geworden?

Nun, ich will Sie nicht länger auf die Folter spannen. Allerdings habe ich zum Zeitvergleich nicht EDDI herangezogen, obwohl sich das Ergebnis (bei rückblickender Betrachtung) nicht verändert hätte. Es wurde ein Programm erstellt, in dem systematisch ein paar Befehlsgruppen abgefragt wurden, um die Geschwindigkeit in verschiedenen Bereichen vergleichen zu können.

Das Ergebnis der Messungen sehen Sie in der Zusammenfassung in Tabelle 1 und 2. Es zeigt sich ganz deutlich, daß Petspeed (obwohl der »Oldtimer« dieses Tests) der eigentliche Testsieger ist. Er hat in fast allen Bereichen die Nase vorne und erreicht Geschwindigkeiten, von denen seine Konkurrenten nur träumen können.

Unser Testprogramm absolvierte er beispielsweise in 145 Sekunden. Das Original unter dem Interpreter benötigt noch 391 Sekunden. Mit großem Abstand folgt dann erst einmal das Compilat von Austro-Speed. Es erreicht immerhin eine Zeit von 173 Sekunden und ist damit um fast 20 Prozent langsamer als Petspeed.

Enttäuscht hat in diesem Test der Exbasic Level II-Compiler. Einmal davon abgesehen, daß er schon eine Reihe anderer Schwächen aufzuweisen hatte, bildete er noch zusätzlich in diesem Geschwindigkeitstest das Schlußlicht mit einer Zeit von 198 Sekunden.

Der BASS-Compiler ist zwar genauso schnell oder langsam wie der Exbasic-Compiler (198 Sekunden), er hat aber immer noch den Vorteil seines positiven Konzepts, des bedienerfreundlichen Compilats, was zumindest Maschinensprach-Spezialisten zu schätzen wissen dürften.

BASS überholt Austro-Speed lediglich bei der Übersetzung von POKEs und PEEKs. Dieser Unterschied ist jedoch sehr gering und kann an dem Gesamtergebnis nichts ändern.

Fazit

Fangen wir mit dem Exbasic Level II-Compiler an. Dieses Produkt konnte in wesentlichen Punkten nicht überzeugen. Einige dieser Gründe sind mit Sicherheit darin zu sehen, daß dieser Compiler nur die um einen Assembler erweiterte Version des BASS ist, wobei jedoch die relativ schwache Leistung des BASS noch durch die verlorengegangenen positiven Eigenschaften dieses Programms verstärkt wurde. Der BASS ist zwar langsam, aber sein Vorteil liegt im offenen Konzept des erzeugten Codes. Damit hat der Benutzer die Möglichkeit, auch beim Compilat noch leicht Änderungen vorzunehmen. Zu diesem Zweck ist BASS sehr umfassend dokumentiert, im Gegensatz zum Exbasic Level II-Compiler.

Die Nachteile beider Compiler waren aber die »pingeligen« Ansprüche an die Syntax des zu compilierenden Programms. Ein Anwender, der mit Compilern arbeitet, möchte in der Regel ein älteres Programm ohne große Änderungen am Original schnell compilieren können. Das ist aber ohne große Änderungen bei diesen beiden Compilern fast nicht möglich.

Als Kaufempfehlung kann hier also höchstens BASS gelten (198 Mark). Dieser Compiler ist jedoch nichts für Anfänger und für Anwender, die sich einen Compiler nur zum bequemen »Hochpuschen« von Basic-Programmen zulegen. Der Exbasic Level II-Compiler ist allein schon seines hohen Preises wegen (298 Mark) gemessen an den Leistungen, nicht ohne Vorbehalt zu empfehlen. Er besitzt keine herausragenden Vorzüge und könnte bei weniger erfahrenen Programmierern schnell ein falsches Bild von Compilern hervorrufen.

Das Nachladen von Programmen, sogenanntes Overlay, machte bei allen Compilern dieses Tests (infolge der Variablenorganisation) Schwierigkeiten. Das im Interpretermodus mögliche Nachladen von Programmen mit Variablenübergabe (Warm-Overlay) ist in der Regel mit den Compilern nicht möglich. Hier muß man normalerweise eine Speicherstelle als Flag (beim Nachladen mehrere Programme) benutzen, da die Variablen durch das Nachladen gelöscht werden (Kalt-Overlay).

Am großzügigsten bei der Analyse von Programmen zeigte sich der Austro-Speed. Er war beim Compilieren mit Abstand der Schnellste. Die Endgeschwindigkeit des Compilats ist zwar nicht mit der des Petspeed vergleichbar; es zeigt sich jedoch deutlich, daß dieser Compiler die wenigsten Probleme aufwirft, zumal seine Bedienung ein wahres Kinderspiel ist. Er »verdaut« die meisten Programme ohne Schwierigkeiten und zeigt sich auch in der Anwendung sehr vielseitig, da er sowohl mit einem, als auch mit mehreren Floppy-Laufwerken zusammenarbeitet. Der Preis vom Austro-Speed ist mit dem von Exbasic Level II-Compiler identisch (298 Mark); hätte man also die Wahl zwischen beiden, so dürfte die Entscheidung nicht allzu schwer fallen.

Sehr viel fürs Geld bekommt man mit Petspeed ins Haus geschickt (149 Mark). Dieser Compiler übertraf alle Erwartungen und zeigte sich auch in der Bedienung recht einfach. Die zwei negativen Aspekte dürften hier wohl die unzureichende Literatur und das notwendige Kopieren des Basic-Programms auf die Systemdiskette sein. Wenn man aber einmal die 149 Mark sieht, die der Petspeed kostet, so zeigt sich dennoch ein hervorragendes Preis/Leistungsverhältnis, das von keinem Compiler des Tests erreicht wurde. In den Tabellen finden Sie zur besseren Orientierung noch einmal alle Test-Ergebnisse zusammengefaßt.

(Karsten Schramm/gk)

Bezugsadressen und Info:
BASS: gmbsoft, Kaiser-Friedrich Ring 55, 6200 Wiesbaden
Austro-Speed (Austro-Comp): Commodore, Lyoner Str. 38, 6000 Frankfurt/Main und für Österreich: Digimat, Arbeitergasse 48, 1050 Wien
Ex-Basic-Compiler II: Interface Age, Vohburger Str. 1, 8000 München 21
Petspeed: Infotronik, Birkenstr. 40, 4100 Duisburg