Zeitmessungen an PEARL-Systemen

von L. Frevert, Fachhochschule Bielefeld

Zusammenfassung

Es werden die Methoden kritisch betrachtet, mit denen die Ausführungszeiten von Anweisungsteilen ermittelt werden können. In der verwendeten Methode wurde gezählt, wie oft Schleifen während fest vorgegebener Zeiten durchlaufen werden. Das führt bei kleinen Ausführungszeiten zu größeren Genauigkeiten, als wenn man die Anzahl der Schleifendurchläufe vorgibt und die Zeiten abliest. Auf der Basis von Mustermoduln sind Module für Messungen an einer große Anzahl verschiedener Elementaroperationen relativ rasch erzeugbar; die Meßprogramme benötigen keine Parametrisierung. Bisher wurden je etwa 130 verschiedene Messungen auf zwei Rechnertypen durchgeführt. Die Meßergebnisse ermöglichen es, Programmierern Hinweise auf Optimierungsmöglichkeiten zu geben, Compilerbauer auf bisher unerkannte Schwachstellen hinzuweisen und detaillierte Leistungsvergleiche zwischen verschiedenen PEARL-Systemen aufzustellen.

1. Einleitung

Gute Assemblerprogrammierer kennen normalerweise die Ausführungszeiten der Maschinenbefehle aufgrund von Angaben des jeweiligen Prozessor-Herstellers und können deshalb zeitkritische Programmteile optimieren. Einigermaßen genaue Leistungsvergleiche verschiedener Prozessortypen allein anhand der Befehlsausführungszeiten sind jedoch schwierig, weil unterschiedliche Befehlssätze unterschiedliche Optimierungsstrategien erfordern. Deshalb werden solche Leistungsvergleiche üblicherweise anhand von sogenannten Benchmark-Programmen vorgenommen, deren Laufzeit mit der Stoppuhr gemessen wird. Jedes Benchmark-Programm entspricht jedoch einem ganz bestimmten Benutzerprofil und ist nur dann von praktischem Wert, wenn dieses Benutzerprofil und die geplante Anwendung sich

nicht zu sehr unterscheiden.

Einem Anwender einer höheren Programmiersprache ist es in der Regel nicht möglich, Programme gezielt zu optimieren, weil er nicht weiß, wieviel Rechenzeiten bestimmte Sprachkonstrukte erfordern, da hier zusätzlich die Güte des verwendeten Compilers eingeht und weil bei konventionellen Sprachen detaillierte Messungen derartiger Zeiten relativ aufwendig sind. Üblicherweise macht man auch hier nur pauschale Leistungsvergleiche mit Benchmark-Programmen. Die Ergebnisse sind dabei oft überraschend: Ein kurzes Programm mit FLOAT-Rechnungen und Benutzung trigonometrischer Funktionen benötigte auf demselben Rechner (einem ATARI 1040 ST+) bei Programmierung in PEARL (RTOS-UH-Compiler) 25 Sekunden Laufzeit, bei Verwendung von PASCAL (MCC-PASCAL) 77 Sekunden, bei C (Megamax C bzw. Lattice C) 85 bzw. 318 Sekunden. (Alle Rechnungen mit einfacher Genauigkeit; bei doppelter Genauigkeit benötigte PEARL 55 Sekunden.)

Bei Computern in Rechenzentren, auf denen Programme mit vielen verschiedenen Anwendungsprofilen laufen, mögen pauschale Leistungsvergleiche mit Benchmark-Programmen ausreichend sein. Bei Prozeßrechnern, die für eine bestimmte Aufgabe eingesetzt werden sollen, können solche Messungen jedoch nur ungefähren Anhalt für deren Eignung geben. Deshalb gibt DIN 19242 Meßvorschriften und Beispiele für "synthetische Benchmarks", mit denen gezieltere Leistungsvergleiche auf Rechnern mit Echtzeitbetriebssystemen möglich sind. Insbesondere beim Einsatz für zeitkritische Aufgaben sollte der Programmierer jedoch auch im einzelnen wissen, welche Ausführungszeiten zum Beispiel Prozeduraufrufe kosten und wie sich Rechnungen mit einfacher oder doppelter Genauigkeit zueinander verhalten.

Rechenzeitmessungen an Echtzeitsystemen sind einerseits leicht automatisierbar, weil derartige Systeme per Programm ablesbare Uhren haben, andererseits aber auch schwieriger, weil man hier nicht nur die Ausführungszeit arithmetischer Anweisungen, sondern auch diejenige für Betriebssystem-Funktionen wie Taskwechsel und

Reaktion auf Interrupts kennen möchte. Bei Ein/Ausgabe-Operationen interessiert hier nicht nur die gesamte Ausführungszeit, sondern auch die Zeit, die der Rechnerkern dabei benötigt, der bei modernen Systemen aus mehreren Prozessoren bestehen kann.

Der PEARL-Anwender-Ausschuß beschäftigt sich seit einiger Zeit mit dem Leistungsvergleich von PEARL-Systemen. Auf seine Anregung sind an der Fachhochschule Bielefeld über hundert PEARL-Module entstanden, von denen jeder die Prozessorzeit einer einzelnen Anweisung oder eines Anweisungsteiles mißt. Die Meßresultate sollen dem Anwender dabei helfen, die Eignung eines PEARL-Systems für seine spezielle Prozeßsteuerungs-Aufgabe zu beurteilen und ihm bei der Programmentwicklung helfen, zeitaufwendige Anweisungsfolgen durch bessere zu ersetzen. Um ein Beispiel zu nennen: beim RTOS-UH auf dem ATARI 520 ST+ dauert "IF BITGESETZT=='1'B THEN...." mehr als zwanzigmal länger als das gleichwertige "IF BITGESETZT THEN...." Für den Compilerbauer resultiert aus solchen Meßresultaten der Anreiz, sein Produkt zu überarbeiten.

2. Zeitmessung bei vorgewählter Wiederhol-Häufigkeit

Bei Echtzeitsprachen wie PEARL scheint es relativ einfach zu sein, die Rechenzeiten einzelner Anweisungen oder Anweisungsteile zu messen, weil die interne Uhr des Rechners von Meßprogrammen abgefragt werden kann. Um einigermaßen genaue Ergebnisse zu bekommen, muß man sich jedoch die möglichen Fehlerquellen überlegen.

Die einfachste und naheliegendste Meßmethode besteht daraus, die zu untersuchende Operation (z. B. "C:=A*B;") in einer Schleife n-mal zu wiederholen und die dafür benötigte Zeit T_a zu messen, indem die Uhrzeiten vor und nach der Schleifenausführung per Programm abgelesen werden; eine zweite Messung mit einer Vergleichsanweisung "C:=A;" im "Testrahmen" ergibt dann die Vergleichszeit T_v. Die Ausführungszeit t_a einer einzelnen Operation (hier der Multiplikation) ist dann $t_a=(T_a-T_v)/n$.

3. Betrachtungen über Meßgenauigkeiten

Die Genauigkeit einer derartigen Messung wird in erster Linie von der Genauigkeit bestimmt, mit der die Uhrzeiten abgelesen werden können. Die interne Uhr des Rechners wird mit einem Taktabstand T_{takt} weitergesetzt. (Er kann durch mehrmaliges Ablesen der Uhr in sehr kurzen Abständen bestimmt werden.) Bild 1 zeigt, daß bei einer gemessenen Zeitdifferenz von 3 Takten die wirkliche Zeitdifferenz knapp über 2 Takten oder knapp unter 4 Takten liegen kann. Deshalb sind Messungen von Uhrzeitdifferenzen mit einer Ungenauigkeit von T_{takt} behaftet.

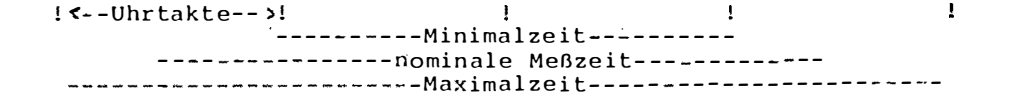

Bild 1: Taktabstand der Uhrtakte und daraus resultierende Meßzeitfehler.

Der Meßfehler des Termes $T_a - T_v$ ist deshalb höchstens $2 * T_{takt}$. Für den relativen Fehler der Ausführungszeit ergibt sich daraus nach Umformung von $(T_a - T_v)$

$$\frac{\Delta t_a}{t_a} = \frac{2 * T_{takt}}{T_a} * \frac{1}{1 - \frac{T_v}{T_a}}$$

Die Genauigkeit, mit der t_a bestimmt werden kann, hängt deshalb empfindlich vom Verhältnis von T_v zu T_a ab. Liegt dieses nahe bei Eins, kann ausreichende Genauigkeit nur durch Vergrößerung von T_a gewonnen werden.

Eine weitere Fehlerquelle kann darin liegen, daß ein PEARL-Betriebssystem bei jedem Interrupt der Rechneruhr prüfen muß, ob eventuell eine eingeplante Task gestartet oder fortgesetzt werden muß. Diese Interrupt-Bearbeitungszeit geht von der Prozessorzeit

ab, die für Benutzerprogramme (hier die Zeitmeßprogramme) zur Verfügung steht und kann nicht mit Hilfe von Benutzerprogrammen gemessen werden. Bei Mehrbenutzer-Systemen sucht das Betriebssystem außerdem in festen Zeitabständen nach weiteren Benutzern; auch diese Rechenzeit geht von der Zeit für Benutzerprogramme ab. (Selbstverständlich dürfen Zeitmessungen nur gemacht werden, wenn keine weiteren Benutzer auf dem System sind.)

Auf den ersten Blick könnte man annehmen, daß diese Rechenzeiten für Betriebssystem-Prozesse den Rechner für Benutzerprogramme nur scheinbar langsamer machen. Man darf jedoch nicht davon ausgehen, daß die Rechenzeiten für diese Routinen konstant sind. Messungen auf einem Rechner mit Mehrbenutzer-Betriebssystem deuten darauf hin, daß dort die Rechenzeiten für die Bearbeitung der Uhr-Interrupts innerhalb von 10 Sekunden insgesamt um etwa 3 Millisekunden schwanken.

Einen weiteren, wenn auch geringen Beitrag zum Meßfehler kann auch die Ablesung und Aufbereitung des Standes der Rechneruhr und ihre Speicherung in Variablen des Meßprogrammes liefern. Die dazu benötigten Rechenzeiten heben sich zwar wegen der Differenzbildung $T_a - T_v$ in erster Näherung heraus; bei genauer Betrachtung darf man jedoch nicht voraussetzen, daß die zur Aufbereitung ausgeführten arithmetischen Operationen konstante Rechenzeiten haben.

4. Das Verfahren nach DIN 19242

Bei der bisher beschriebenen Meßmethode wird die gesamte Ausführungszeit einer Anweisung (in DIN 19242 "Verweilzeit" genannt) gemessen. Bei modernen Rechnern laufen E/A-Anweisungen größtenteils parallel zur Arbeit des Rechenprozessors; letzterer wird nur dazu benutzt, deren Ausführung anzustoßen. Um nur die dafür benötigte Zeit (in DIN 19242: "Belegungszeit des Zentralprozessors") zu messen, werden in DIN 19242 zusätzliche Messungen vorgeschrieben: während der Messung der Verweilzeit läuft im Hin-

tergrund eine Task (DIN 19242: "Hintergrundprogramm") mit geringerer Priorität. Wie Bild 2 zeigt, verlängert sich auf einer Einprozessormaschine die Ausführungszeit dieser Hintergrundtask, weil ihr zugunsten der Verweilzeitmessung Rechenzeit entzogen wird. Falls die Meßtask den Rechnerkern zwischenzeitlich freigibt, kann die Hintergrundtask derweil weiter arbeiten. Bei E/A-Anweisungen wird die Ausführungszeit der Hintergrundtask deshalb nur um die Zeit verlängert, die das Betriebssystem auf dem Zentralprozessor für den Anstoß der E/A und für die Taskwechsel benötigt.

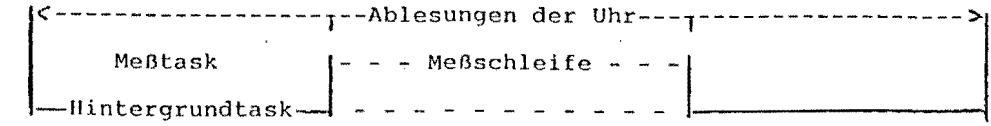

Bild 2: Zeitmessungen nach DIN 19242: Die Zeit zwischen Beginn und Ende einer Hintergrundtask verlängert sich dadurch, daß die Meßtask ihr den Prozessor entzieht.

Die Belegungszeit des Zentralprozessors durch die Meßschleife kann deshalb bestimmt werden, indem man die Zeiten miteinander vergleicht, um die sich die Hintergrundtask verlängert, wenn man sie mit und ohne die Meßschleife laufen läßt. Bei Mehrprozessormaschinen ist das Verfahren in dieser einfachen Form allerdings nicht verwendbar, sondern muß so erweitert werden, daß so viele Hintergrundtasks benutzt werden, wie Prozessoren vorhanden sind.

Das Verfahren von DIN 19242 ist entwickelt worden, um die Verweil- und Belegungszeiten von "synthetischen Benchmarks" zu bestimmen. Sie bestehen aus Anweisungsfolgen. Beim Vergleich ist der Testrahmen leer. Da zusätzlich vorgeschrieben wird, daß T_a im Minutenbereich liegen soll, ist bei Messungen nach DIN 19242 sowohl T_{takt} klein gegen T_a und die anderen erwähnten Fehlerzeiten, als auch das Verhältnis von T_v zu T_a klein gegen Eins, sodaß die dadurch verursachten Fehler vernachlässigt werden können.

Ziel der hier beschriebenen Messungen war jedoch, die Ausführungszeiten von Teilen einzelner Anweisungen genau zu messen. Wenn dabei kleine Unterschiede zwischen Meß-Operation und Vergleichsoperation bestehen, z.B. bei der Messung der für eine Parameterübergabe in eine Prozedur erforderliche Zeit, sind bei der Verwendung der DIN-Methode Meßzeiten von mehreren Minuten erforderlich. Man muß außerdem bei jedem Einzeltest anhand vorläufiger Ergebnisse die notwendige Laufzeit ermitteln und im Testprogramm die notwendige Anzahl der Schleifendurchläufe entsprechend festlegen. Diese Parameterwahl läßt sich zwar automatisieren, erfordert dann aber relativ aufwendige Programme.

5. Zählung der Wiederholungen innerhalb vorgewählter Meßzeit

Bei Verwendung von PEARL ist eine Verwendung kleiner, immer gleicher Meßzeiten dadurch möglich, daß man nicht die Laufzeiten für eine feste Anzahl von Schleifendurchläufen mißt, sondern bei fest vorgegebener Laufzeit T die Anzahl N_a der in dieser Zeit ausgeführten Schleifendurchläufe bestimmt und mit der Anzahl N_v der in derselben Zeit ausgeführten Vergleichsschleifen vergleicht. Die Laufzeit wird dabei durch eine mit höherer Priorität laufende Steuertask bestimmt, die sich mit der Rechneruhr synchronisiert (Bild 3).

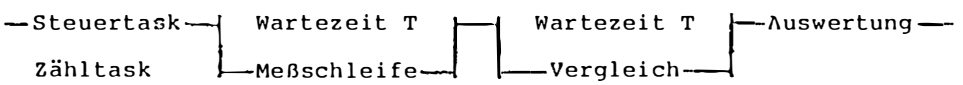

Bild 3: Die zur Zeitmessung benutzt Zähltask kann die Meßschleife nur ausführen, während die Steuertask wartet. Ausgewertet wird die Zahl der Schleifenwiederholungen während dieser Wartezeiten.

Die Zähltask besteht dabei aus einer Schleife, in der die Meßoperation und die (eventuell leere) Vergleichsoperation in bedingt durchlaufenen Programmteilen stehen (Bild 4). Diese Schleifenorganisation hat den Vorteil, daß sie auch bei optimierenden Compilern benutzt werden kann.

```
ZAEHLTASK: TASK PRIO 5;
   WHILE SCHLEIFE REPEAT
     IF MESSEN THEN
        /* Na hochzählen */
        /* zu messende Anweisung */
     FIN;
     IF VERGLEICHEN THEN
        /* Nv hochzählen    */
        /* Vergleichsanweisung */
     FIN;
   END;
END;
```

Bild 4: Wesentlicher Teil der Meßtask, welche die Schleifendurchläufe für Meß- und Vergleichsoperation zählt.

Die Steuertask startet die Zähltask, synchronisiert sich mit der Rechneruhr, setzt die Steuervariablen und wertet die Messungen aus (Bild 5).

```
STEUERTASK:TASK PRIO 1;
   SCHLEIFE='1'B; MESSEN:='0'B; VERGLEICHEN:='0'B;
   /* Zählvariable nullsetzen */
   ACTIVATE ZAEHLTASK;
   AFTER SYNCZEIT RESUME;  /* Synchronisierung mit Uhr */
   VERGLEICHEN:='1'B;     /* zuerst Vergleichsmessung */
   AFTER T RESUME;        /* Messzeit abwarten */
   VERGLEICHEN:='0'B;
   AFTER SYNCZEIT RESUME;  /* Synchronisierung mit Uhr */
   MESSEN:='1'B;
   AFTER T RESUME;  /* gleiche Zeit für eigentliche Messung */
   MESSEN:='0'B;
   SCHLEIFE:='0'B;       /* Zähltask beendet sich später */
   /* Zählungen auswerten */
END;
```

Bild 5: Die Steuertask synchronisiert sich mit der Rechneruhr, setzt Steuerbits, die durch die Meßtask abgefragt werden, und wartet während der Meßzeit.

Das Verfahren mißt in der bisher beschriebenen Form auch bei E/A-Anweisungen die gesamte Ausführungszeit; man kann jedoch auch die Rechnerkern-Belegungszeit leicht bestimmen, indem man noch eine weitere (bei Mehrprozessormaschinen mehrere) Hintergrund-Task mit geringerer Priorität als die Zähltask laufen läßt, die immer dann zählt, wenn die Zähltask den Rechnerkern freigibt (Bild 6). Eine

dritte Vergleichsmessung bei terminierter Zähltask und zählender Hintergrundtask ermöglicht dann, die Auslastung des Rechnerkerns durch die E/A-Operation zu bestimmen.

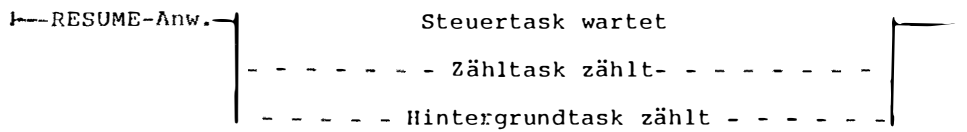

Bild 6: Messung der Prozessorzeit bei E/A-Anweisungen: während die Zähltasks den Prozessor nicht benötigt, kann die Hintergrundtask Schleifendurchläufe zählen.

Seien

- T die Meßzeiten (AFTER T RESUME)
- t_a die Ausführungszeit der zu messenden Operation,
- t_k die Rechnerkernzeit der zu messenden Operation,
- t_v die Rechnerkernzeit der Vergleichsoperation,
- t_s die Rechnerkernzeit für Schleifenorganisation und Zählen der Schleifendurchläufe,
- N_a die Zahl der Schleifendurchläufe, in denen die zu messenden Operation und die Vergleichsoperation beide von der Meßtask aus geführt werden,
- N_v die Zahl der Schleifendurchläufe, in denen nur die Vergleichs-Operation ausgeführt wird,
- N_h die Zahl der Schleifendurchläufe der Hintergrund-Task während der Messung von N_a,
- N_l die Zahl der Schleifendurchläufe der Hintergrund-Task, während die Meßtask nicht läuft

so gelten folgende Gleichungen, wobei sich die ersten beiden auf dieselbe Messung beziehen:

$$T=N_a*(t_a+t_v+t_s)$$
$$T=N_a*(t_k+t_v+t_s)+N_h*t_s$$
$$T=N_v*(t_v+t_s)$$
$$T=N_l*t_s$$

aus denen sich

$$t_a = \frac{T}{N_a} * \left(1 - \frac{N_a}{N_v}\right)$$

und

$$t_k = \frac{T}{N_a} * \left(1 - \frac{N_a}{N_v} - \frac{N_h}{N_l}\right)$$

ergeben.

6. Fehlerbetrachtung

Eine der Fehlerquellen des zuletzt beschriebenen Verfahrens besteht daraus, daß die Zähltask am Ende der Messung während der Ausführung der Meßschleife durch die Steuertask unterbrochen wird. Dadurch ist es ungewiß, ob die gezählte Operation schon durchgeführt ist: die Zählwerte N sind mit einer Unsicherheit von Eins behaftet. Aus diesem Grunde ist das Verfahren (bei vergleichbaren Meßzeiten) ungenauer als DIN 19242, wenn die Ausführungszeit der gezählten Operation größer ist als T_{takt}, während es im umgekehrten Falle genauer ist.

Auf den ersten Blick könnte man annehmen, daß die Werte für N_a und N_v mit dieser Unsicherheit von höchstens Eins gezählt werden können. Bei wiederholten Messungen mit Anweisungen von sehr kurzer Ausführungszeit ergeben sich jedoch Schwankungen von N_a oder N_v, die größer als Eins sind. Sie werden vermutlich dadurch verursacht, daß die Zeiten für diejenigen Betriebssystem-Prozesse (Bearbeitung der Uhr-Interrupts), die das Meßprogramm unterbrechen, nicht konstant sind. Beim ATARI 510 ST+ betragen diese Schwankungen bei reinen Zählschleifen höchstens 4 in den Zählungen der Schleifendurchläufe (insgesamt ca. 500000) bei einer Meßzeit von insgesamt 10 Sekunden, beim einem Prozeßrechner mit Multiuser-Betriebssystem etwa 70 (bei insgesamt etwa 250000). Man kann diese Zahlen umrechnen in einen relativen Fehler $\Delta T/T$ von rund 0.001% bzw. 0.03%.

Ein weiterer Fehler kann bei den verwendeten Meßzeiten T von 10 Sekunden vernachlässigt werden: die wirkliche Meßzeit ist kleiner als die nominelle Meßzeit T, weil T ab dem letzten Uhrtakt gerechnet wird, der die Fortsetzung der Steuertask verursacht hat, die Zähltask aber während des Neusetzens der Steuerbits und der Einplanungs-Anweisung "AFTER T RESUME;" noch nicht laufen kann. Für Einplanungen werden bei den untersuchten PEARL-Systemen höchstens 3 Millisekunden benötigt: der ATARI 520 ST+ bzw. der Vergleichsrechner benötigen für eine RESUME-Einplanung inklusive Wiederstart der Task 0.5 bzw. 2.5 Millisekunden.

Der relative Fehler des Meßergebnisses t_a ergibt sich als Summe der relativen Fehler von T, N und dem Klammerterm. Beim Klammerterm ist der relative Fehler

$$\left(\frac{1}{N_a} + \frac{1}{N_v} + 2 * \frac{\Delta T}{T}\right) * \frac{\frac{N_a}{N_v}}{1 - \frac{N_a}{N_v}}$$

und wird stark von der Größe des Quotienten N_a/N_v beeinflußt. Wenn N_a und N_v ungefähr gleich sind, geht er gegen $2/(N_v-N_a)$, kann also untragbar groß werden.

Die für die Messungen entwickelten Programme berechnen auch die Meßunsicherheiten. Sie lagen bei allen Messungen unter 10%, bei den meisten sogar unter 1%.

7. Messung von Tasking-Anweisungen

Auch die Messung von Tasking-Anweisungen ist mit dem beschriebenen Verfahren möglich. Um z. B. zu bestimmen, wieviel Zeit die Ausführung einer ACTIVATE-Anweisung, Start und Beendigung einer Task erfordern, wurde in der Zähltask eine leere Task aktiviert,

deren Priorität unterhalb der Steuertask und über der Zähltask lag (Bild 7).

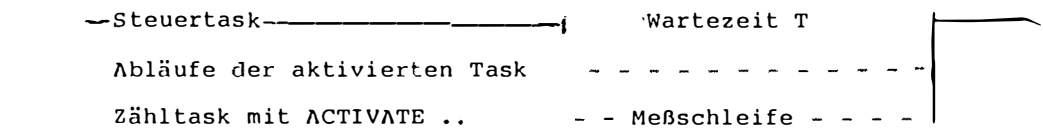

Bild 7: Messung der Zeit für Taskaktivierung und -start: eine Task höherer Priorität wird in der Meßschleife immer wieder aktiviert und läuft entsprechend oft.

Etwas schwieriger ist die Messung der Zeiten für SUSPEND - CONTINUE. Auf den ersten Blick scheint die Sache einfach: in die zusätzliche Task A wird "IF MESSUNG THEN SUSPEND; FIN;" geschrieben, in die Meßschleife "IF MESSUNG THEN .... ACTIVATE A; CONTINUE A; FIN;". Um die Zeit für das "ACTIVATE A;" zu eliminieren, liegt es nahe, beim Vergleichen eine leere Task B mit "ACTIVATE B;" zu aktivieren. Das kann jedoch zu Meßfehlern führen, weil die Dauer von Tasking-Anweisungen von der Länge der Tasklisten des Betriebssystems und dem Eintragungsort abhängen kann (beim RTOS-UH beispielsweise auch von der Speicherbelegung). Es darf deshalb nur eine zusätzliche Task für diese Messung geben; sie muß "IF MESSUNG THEN SUSPEND; FIN; IF VERGLEICHEN THEN ; FIN;" enthalten, so daß sie die SUSPEND-Anweisung nur ausführt, wenn das Bit MESSUNG von der Steuertask gesetzt ist. Das wieder führt zu der Komplikation, daß Meßschleife und A die Steuerbits nacheinander abfragen und diese zwischenzeitlich von der Steuertask geändert werden können, sodaß SUSPEND und CONTINUE nicht mehr paarig ausgeführt werden. Steuertask, Meßtask und Task A mußten deshalb über Semaphore sauber koordiniert werden. Ohne diese Koordination kann es insbesondere bei der Messung von Semaphoranweisungen zu Verklemmungen kommen.

8. Anwendungserfahrungen

Es wurde zunächst versucht, bei dem Verfahren nach DIN 19242 die Festlegung der Meßzeit dadurch zu automatisieren, daß das Programm zuerst die Ausführungszeiten der zu testenden Operationen grob bestimmte und daraus die Anzahl der notwendigen Wiederholungen berechnete. Wegen der sich ergebenden großen Laufzeiten wurde dieser Weg dann nicht weiter verfolgt, sondern die zuletzt beschriebene Methode entwickelt. Einige Anweisungen wurden mit beiden Methoden untersucht; die Ergebnisse stimmten erwartungsgemäß gut überein.

Mit der neuen Methode wurde dann eine große Anzahl Messungen für verschiedene Anweisungen und Anweisungsteile auf einem Prozeßrechner und an einem RTOS-UH-PEARL-System auf einem ATARI 520 ST+ durchgeführt. Die Meßzeiten T betrugen dabei durchweg 10 Sekunden. Auf beiden Rechnern liefen selbstverständlich während der Messungen keine weiteren Programme. Für einige Anweisungen wurden mehrere Messungen durchgeführt, um ihre Reproduzierbarkeit untersuchen zu können. Diese war bei arithmetischen Anweisungen sehr gut. Wenn das Betriebssystem involviert war, ergaben sich in einigen Fällen größere Schwankungen, was wohl darauf zurückzuführen ist, daß die fraglichen Betriebssystem-Routinen je nach Speicherbelegung und Vorhandensein ruhender Tasks unterschiedliche Laufzeiten haben. Beim ACTIVATE auf dem ATARI wurden z. B. Werte zwischen 0.9 und 2.25 Millisekunden gemessen,

Praktisch wurde so vorgegangen, daß alle rechnerspezifischen Programmteile, insbesondere der Systemteil, in einem Modul INIT zusammengefaßt wurden, der auch die Unterprogramme für Initialisierung, Auswertung und Protokollierung der Messungen enthält. Außerdem wurde ein Mustermodul entwickelt, in den die zu testenden Anweisungen und die Texte für die Protokollierung, sowie Task- und Prozedurnamen eingesetzt wurden, um die Testmodule zu gewinnen.

Je neun der so entstandenen Testmodule wurden mit dem schon er-

wähnten Modul INIT und einem Modul für die Starttask zu einem PEARL-Programm gebunden. Mit Hilfe einiger Unterstützungsprogramme dauerte die Erzeugung inklusive Probelauf eines derartigen PEARL-Programmes für neun Einzelmessungen etwa eine Stunde.

9. Meßergebnisse

Insgesamt sind bisher etwa 130 verschiedene Operationen auf einem ATARI 520 ST+ (mit RTOS-UH Version 2.0) und auf einem Prozeßrechner untersucht worden. Dabei wurden einige "Ausreißer" entdeckt: EXOR dauert auf dem ATARI etwa zwanzigmal länger als AND oder OR; eine Anweisungsfolge REQUEST S; RELEASE S; (ohne Warten) benötigt auf dem Prozeßrechner etwa etwa zehnmal länger als auf dem ATARI, der nur 32 Mikrosekunden braucht. Überraschend war auch, daß die Zeiten auf dem Prozeßrechner etwa doppelt so hoch waren, wenn statt auf tasklokale auf modulglobale FIXED-Variable zugewiesen wurde.

Bei dem Prozeßrechner handelte es sich um einen KRUPP ATLAS ELEKTRONIK EPR 1300, der 1979 ausgeliefert worden ist. Auch das PEARL-System ist alt (Stand von 1980), weil sich eine Fachhochschule den Kauf neuerer Versionen nicht leisten kann. Beim RTOS-UH ist die CHAR-Verarbeitung gegenüber der ersten Version bedeutend schneller geworden; es wäre interessant, auch auf einem neuen EPR 1300 Messungen vorzunehmen.

Pauschal kann man über einen Vergleich beider Rechner folgendes aussagen: bei Operationen zur CHAR-Verarbeitung ist der ATARI mindestens zehnmal schneller als der andere Rechner (beim RTOS-UH kann man jedoch nicht auf Teilketten, sondern nur auf Einzelzeichen aus CHAR-Ketten zugreifen), bei Zuweisungen, Schleifenorganisation und einfachen IF-Verzweigungen etwa doppelt so schnell. Dafür sind beim Prozeßrechner Bitshift-Operationen und FLOAT(23)-Rechnungen etwa fünfmal schneller. Aktivierungen mit Taskstart führt der Prozeßrechner schneller aus (0.77 ms), dafür braucht er für Taskstarts per Einplanung (1.23 ms) und für RESUME (2.5 ms)

etwa drei- bzw. viermal so lange wie der ATARI.

Programmstrukturierung durch Verwendung von Prozeduren kostet Zeit: Aufrufe von RESIDENT REENT-Prozeduren verbrauchen beim Prozeßrechner 1.2 Millisekunden, beim ATARI dagegen nur 0.25; RESIDENT-Prozeduren hingegen werden vom Prozeßrechner in nur 0.11 Millisekunden aufgerufen und wieder verlassen (RTOS-UH kennt nur RESIDENT REENT). Die Übernahme eines FIXED-Parameters erledigt der Prozeßrechner in 6 Mikrosekunden, der ATARI hingegen benötigt 43, weil beim RTOS-UH-PEARL die Parameter-Verträglichkeit erst zur Laufzeit geprüft wird.

Die Meßergebnisse zeigen, daß Resultate aus einem Vergleich mit Benchmark-Programmen mit großer Vorsicht zu bewerten sind, wenn die Eignung eines Rechners für eine bestimmte Anwendung zur Debatte steht. Der ATARI dürfte bei Textverarbeitung in der Regel merklich schneller sein als der Prozeßrechner. Wenn jedoch in einem Programm Teilketten häufig umgespeichert oder auf Gleichheit geprüft werden sollen, muß man beim ATARI eigens für diesen Zweck geschriebene Prozeduren aufrufen, wodurch sich dessen Vorsprung womöglich ins Gegenteil verkehrt. Der Prozeßrechner wird seine größere Schnelligkeit bei FLOAT-Rechnungen nur in speziellen Programmen ganz ausspielen können: jede Zuweisung, jede Schleife vermindert seinen Vorsprung.

Die Ergebnisse sind auch lehrreich für Fälle, bei denen die Rechenleistung bei Verwendung verschiedener Programmiersprachen verglichen werden soll: es muß dann beispielsweise darauf geachtet werden, daß sich die Programme auch in Hinblick darauf entsprechen, daß Variable prozedur- bzw. tasklokal oder modulglobal (bei FORTRAN im COMMON) deklariert werden.

Den Mitgliedern des PEARL-Anwender-Ausschusses gebührt Dank für kritische Diskussionen über die Meßmethoden und deren Auswertungen; die Meßergebnisse für C- und PASCAL-Programme stammen von meinen Kollegen Dr. Ehrich und Dr. Eisenack.