ZIN EN ONZIN VAN BENCHMARK-TESTEN

H. J. C. OTTEN

Wellicht is het u opgevallen dat in Radio Bulletin tot nu toe bij de bespreking van computers het woord benchmark nooit is gevallen. In andere tijdschriften zijn bij besprekingen van computers, veelal „test" genoemd, bijna altijd indrukwekkende lijsten met cijfers opgenomen waarmee schijnbaar een vergelijking tussen computers mogelijk is. Ik vind het daarom nodig in te gaan op wat benchmark-testen zijn en meer wat ze zeer zeker niet zijn.
Hopelijk wordt het daarom duidelijk, waarom in Radio Bulletin geen benchmark-testen worden opgenomen.

■ Wat zijn benchmark-testen?

In de oorspronkelijke betekenis van het woord benchmark zit al begrepen wat een benchmark-test beoogt: het op een werkbank zetten van een apparaat en door middel van meten cijfers geven.

Bij computers worden benchmark-testen voornamelijk uitgevoerd om vast te stellen of een bepaalde combinatie van hard- en software in staat is een bepaalde opdracht binnen de vooraf vastgestelde tijd uit te voeren. Daartoe worden met zorg een aantal programma's gekozen die representatief zijn voor de applicatie èn de tijd, die nodig is voor de uitvoering daarvan, gemeten. Deze cijfers geven een redelijke indruk van wat de hard- en software kan presteren. De tijdrovende operaties worden op deze manier in getallen gevangen.

■ Tijdrovende operaties

Bij het gebruik van een computer vallen een paar tijdrovende operaties op. Daarbij is de snelheid van de centrale processor meestal niet van het grootste belang. Vaak is de oorzaak van een te traag verlopend programma terug te voeren op òf een verkeerde keuze van de software òf een te traag verlopende communicatie met de randapparatuur.
Het eerst heb ik „de verkeerde keuze van de software" genoemd. Als een Basic-interpreter wordt gebruikt voor intensieve berekeningen is dat een duidelijk voorbeeld van een verkeerde keuze. Als de computer alleen de Basic-interpreter als programmeertaal biedt, is ook duidelijk voor deze applicatie de verkeerde computer gekozen. Ik durf zelfs te stellen dat een Basic-interpreter op de snelste microcomputer nog een verkeerde keuze is voor veel rekenwerk.
Gelukkig kan voor alle serieuze computers een programmeertaal worden gekozen, die zonodig ook voldoet aan de snelheidseisen.
De benchmarks vertellen echter alleen maar iets over een Basic-interpreter.
Communicatie met de randapparatuur is een nog zwaarder wegende factor in het gebruik van tijd. Het voorbeeld van een supersnelle microcomputer met een audio-cassetterecorder voor data-opslag mag voor zich spreken. Het gebruik van bestanden is bij alle zinvolle programmatuur een vanzelfsprekende zaak. De benchmark-testen in lijst 1 vertellen echter niets over de communicatie met randapparatuur.

■ Microcomputers en benchmark-testen

Het grote aanbod van microcomputers maakt het moeilijk een keuze te maken. De harde cijfers van een benchmark-test lijken hierbij een onpartijdig en betrouwbaar hulpmiddel.

■ Zin van benchmarks

Benchmark-testen beloven, bij een enorm aanbod van machines met ieder zijn eigen toeters en bellen, een middel te zijn om de juiste computer voor de juiste applicatie te kiezen. Benchmark-testen leveren getallen op die kunnen worden vergeleken; beslissingen behoren gebaseerd te zijn op harde feiten.

■ Onzin van benchmarks

Veel van de gepubliceerde benchmark-resultaten zijn in de praktijk van weinig waarde. Om deze uitspraak duidelijk te maken wil ik aan de hand van twee

Lijst 1

```
Benchmark 1

 10 PRINT "S"
 20 FOR K=1 TO 1000
 30 NEXT K
 40 PRINT "E"
 50 END


Benchmark 2

 10 PRINT "S"
 20 K=0
 30 K+K+1
 40 IF K<1000 THEN 30
 50 PRINT "E"
 70 END


Benchmark 3

 10 PRINT "S"
 20 K=0
 30 K+K+1
 40 A+K/K*K-K
 50 IF K<1000 THEN 30
 60 PRINT "E"
 70 END


Benchmark 4

 10 PRINT "S"
 20 K=0
 30 K+K+1
 40 A=K/2*3+4-5
 50 IF K<1000 THEN 30
 60 PRINT "E"
 70 END
```

```
Benchmark 5

 10 PRINT "S"
 20 K=0
 30 K+K+1
 40 A=K/2*3+4-5
 45 GOSUB 80
 50 IF K<1000 THEN 30
 60 PRINT "E"
 70 END
 80 RETURN

Benchmark 6

 10 PRINT "S"
 20 K=0
 30 DIM M(5)
 40 K+K+1
 50 A=K/2*3+4-5
 60 GOSUB 120
 70 FOR L=1 TO 5
 80 NEXT L
 90 IF K<1000 THEN 30
100 PRINT "E"
110 END
120 RETURN

Benchmark 7

 10 PRINT "S"
 20 K=0
 30 DIM M(5)
 40 K+K+1
 50 A=K/2*3+4-5
 60 GOSUB 120
 70 FOR L=1 TO 5
 80 M(L)=A
 90 NEXT L
100 IF K<1000 THEN 30
110 PRINT "E"
120 END
130 RETURN
```

populaire benchmark-testen voor microcomputers, die voorzien zijn van de programmeertaal Basic, laten zien hoe gering de waarde van deze benchmarks zijn.

■ Benchmark van Kilobaud

Het Amerikaanse tijdschrift Kilobaud heeft in de beginjaren van de op microprocessoren gebaseerde hobbycomputers een aantal in Basic geschreven programma's als benchmark voorgesteld. Deze korte programma's zijn in lijst 1, als benchmark 1 tot en met 7, weergegeven. Alle programma's bestaan uit één programmalus, waarbij telkens één operatie in de lus wordt uitgevoerd, zoals een berekening of het aanroepen van een subroutine. De tijd die nodig is voor elk programma wordt gemeten en als benchmark opgegeven ter vergelijking. De test is zo ontworpen dat elke volgende test iets toevoegt, zoals een aanroep van een subroutine of een berekening naar aanleiding van de handelingen in de vorige test. Om een vergelijking tussen de implementaties van de programmeertaal Basic te krijgen, levert deze benchmark best interessante getallen op. Maar, als we zinvolle conclusies willen trekken uit een benchmark voor de implementatie van de programmeertaal Basic, is wel meer nodig dan de programma's uit te voeren en de resultaten in een lijst ter vergelijking aan te bieden.

Ten eerste geven de benchmarktesten 1 tot en met 7 een zeer beperkte indruk. Het aantal gebruikte verschillende Basic-statements is in het algemeen maar een kleine greep uit het repertoire van de meeste implementatie van Basic.

Ten tweede is het gebruik van variabelen zonder aan te geven wat voor type (integer of real) en welke precisie ze bezitten niet zonder gevaar.

De meeste Basic-interpreters bieden een keuze uit enkele of dubbele precisievariabelen van het type real en het veel snellere resultaten opleverend type integer. Bijna altijd wordt in vergelijkende benchmark-testen dit aspect niet vermeld.

Ten derde geven de benchmarktesten geen enkele indruk van de snelheid waarmee programma's kunnen communiceren met de randapparatuur. Een analyse van wat in programma's de meest tijdrovende handelingen zijn, duidt vrijwel altijd in de richting van de in/uit-faciliteiten.

■ Zeef van Eratosthenes

Vooral in het tijdschrift Byte wordt veel gebruik gemaakt van een enkel programma als benchmark.

Eratosthenes was het hoofd van de bibliotheek van Alexandrië ongeveer 200 voor Christus. Hij was één van de veelzijdigste geleerden van zijn tijd. Hij heeft bijvoorbeeld de omtrek van de aarde vrij nauwkeurig berekend en een methode ontwikkeld om priemgetallen te vinden. Deze methode heeft de naam „Zeef van Eratosthenes" gekregen en laat zich gemakkelijk in een computerprogramma implementeren.

In lijst 2 zijn twee implementaties van de zeef van Eratosthenes getoond, de eerste in Basic en de tweede in Pascal. In de literatuurverwijzingen naar het tijdschrift Byte zijn vele implementaties en resultaten voor een

Lijst 2 Zeef van Eratosthenos in Basic

```
  5 DEFINT A-Z
 10 DIM FLAGS(8191)
 20 PRINT "10 ITERATIES"
 30 FOR M=1 TO 10
 40   COUNT = 0
 50   FOR I=1 TO 8191
 60     FLAGS[I] = 1
 70   NEXT I
 80   FOR I=1 TO 8191
 90     IF FLAGS(I) = 0 GOTO 170
100     PRIME = I + I + 3
110     K = I + PRIME
120     IF K > 8190 GOTO 160
130       FLAGS(K) = 0
140       K = K + PRIME
150       GOTO 120
160     COUNT = COUNT + 1
170   NEXT I
180 NEXT M
190 PRINT COUNT, " PRIMES"
200 END
```

Zeef van Eratosthenos in Pascal

```
Program Priem ;

CONST
        size = 8190 ;

VAR
        flags : array [ 0 .. size ] of boolean ;
        i, k, prime, count, iter : integer ;

BEGIN

  WRITELN ('10 iteraties') ;
  FOR iter := 1 TO 10 DO
    BEGIN
        count := 0 ;
        FOR i := 0 TO size DO
          flags[i] := true ;
        FOR i := 0 TO size DO
          IF flags[i]
            THEN
              BEGIN
                prime := i + i + 3 ;
                k := i + prime ;
                WHILE k <= size DO
                  BEGIN
                    flags[k] := false ;
                    k := k + 1
                  END ;
                count := count + 1
              END
    END ;

  WRITELN ( count, ' primes' )

END.
```

indrukwekkend aantal computers te vinden.
Deze benchmark-test is overigens in vrijwel elke beschikbare programmeertaal beschikbaar.
Ook op deze benchmark-test zijn de meeste hierboven aangevoerde bezwaren tegen de cijfers zonder meer als zinvol resultaat van toepassing.
Verder is het de vraag of het vinden van priemgetallen gezien mag worden als representatief voor het gebruik van een computer.

■ Zinvolle benchmarks

Benchmarks vind ik zeer zeker niet zonder zin.
Goed uitgevoerde benchmarks kunnen een gefundeerde keuze van hard- en software mogelijk maken voor een goed omschreven applicatie.
Het samenstellen van een dergelijke benchmark is echter geen kleinigheid en vereist veel ervaring en produktkennis. Het resultaat moet altijd worden geëvalueerd samen met alle andere aspecten, die van belang zijn bij de aanschaf van hard- en software voor een door automatisering op te lossen probleem. Een algemene benchmark voor de keuze van een personal computer is een illusie en moet met wantrouwen worden bekeken.
Daarom vindt u in Radio Bulletin geen benchmarkresultaten. Ik presenteer ook geen testen, maar probeer een totaalindruk van een produkt, een computer, te geven.

■ Literatuur over benchmarks-testen

1. „A high-level Language Benchmark" van Jim Gilbreath in Byte van september 1981 op pagina 180 e.v.
2. „Eratosthenes Revisited" van Jim en Gary Gilbreath in Byte van januari 1983 op pagina 283 e.v.
3. „Benchmarks on test" van Mike Lewis in Practical Computing van januari 1984 op pagina 102 e.v.
4. „Don't Bench Me In" van Jerry Houston in Byte van januari 1984 op pagina 160 e.v.