STICHTING
MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

CR 8

Capita uit de numerieke wiskunde.

Colloquim 1955/56.

incompl.

A.van Wijngaarden en J.Berghuis en E.W.Dijkstra.

1956

Colloquium "Capita uit de Numerieke Wiskunde"

Multi-lengte programmering.

door

E.W. Dijkstra

De wijze, waarop multi-lengte rekenwerk door een automatische rekenmachine het meest efficiënt verricht kan worden, zal in hoge mate beinvloed worden door de specifieke eigenschappen van de betroffen machine.

Zonder te streven naar volledigheid laten wij ter inleiding enkele mogelijkheden volgen.

Wij zullen ons beperken tot machines met vaste komma. (Bij een machine met ingebouwde drijvende komma zal men van de drijvende komma waarschijnlijk geen enkel voordeel hebben, mogelijkerwijs zelfs nadeel, b.v. als er in de code niet voorzien is in een soepele methode ter isolatie van het minst significante gedeelte van een product.)

Het multi-lengte getal wordt opgesplitst in "blokken"; voor elk blok wordt een woord van de standaardlengte ter beschikking gesteld.

Bij een binaire machine rijst allereerst de vraag, of de eenheid van blok 10^n of 2^m maal zo significant moet zijn als de eenheid van het volgende blok.

Om het lange getal in decimale blokken te splitsen, biedt eigenlijk alleen voordeel bij de invoer en de uitvoer; bij administratieve processen via ponskaarten valt het te overwegen, mits de deling door de machine voldoende snel kan worden uitgevoerd. Dit laatste is minder urgent, als het administratieve proces hoofdzakelijk uit additieve bewerkingen bestaat; zij worden vergemakkelijkt als de woordcapaciteit meer dan twee maal zo groot is dan de betroffen tienmacht, die met de bloklengte overeenkomt.

Voor de ARMAC, zonder ingebouwde deling, en voor wetenschappelijke berekeningen, was het opsplitsen in decimale blokken kennelijk niet het overwegen waard.

Van vrij veel invloed is de in de machine gangbare representatie van negatieve getallen (annex de werking van het arithmetisch orgaan).

1. Het "complementen-systeem".

Is de woordlengte m+1 bits, (tekencijfer, gevolgd door m binaire cijfers) en denken wij de komma onmiddellijk achter het tekencijfer, dan zijn in dit systeem de capaciteitsgrenzen

$$-1 \leq x \leq 1-2^{-m}$$

0 is hier "het kleinste positieve getal". Een getal wordt van teken gewisseld, door alle cijfers te inverteren (0 door 1 vervangen en omgekeerd) en bovendien 2^{-m} (een 1 op de minst significante plaats) op te tellen. Tekenwisseling geschiedt dan wel m.b.v. de opteller; additieve bewerkingen worden zonder end around carry uitgevoerd.

(Een andere versie van het complementensysteem zou zijn met de capaciteitsgrenzen

$$-(1-2^{-m}) \leq x \leq 1 \ ;$$

0 zou dan "het grootste negatieve getal" zijn. De voorstelling van positieve getallen wordt dan minder vanzelfsprekend; m.i. wegen de - aanvechtbare! - bezwaren van een negatieve nul ruimschoots op tegen het bezit van de +1.)

2. Het inversen-systeem.

Hier zijn de capaciteitsgrenzen

$$-(1-2^{m}) \leq x \leq 1-2^{-m}$$

Er zijn twee representaties voor het getal nul: +0 (allemaal nullen) en -0 (allemaal enen). Een getal wordt van teken gewisseld door alle cijfers te inverteren; dit kan dus geschieden zonder gebruik te maken van de opteller. Additieve bewerkingen moeten echter uitgevoerd worden met end around carry.

Voor de representatie van een getal van q-voudige lengte nemen we aan dat het tekencijfer van elk woord als tekencijfer van het blok fungeert, dat dus met een woordlengte van m+1 bits de opeenvolgende blokken een factor 2^{m} in significantie verschillen.
Als we de blokken als geheel getal opgevat met B_1 aanduiden, is het multi-lengte getal gelijk aan

$$B_1.2^{-m} + B_2.2^{-2m} + \ldots + B_q 2^{-qm}$$

In het complementensysteem nu is het aantrekkelijkste om het meest-significante blok het teken van het getal mee te geven, en alle volgende blokken het +-teken (resp. het teken van de nul) te laten hebben.
(Analoog aan de voorstelling: log 0.2 = -1 + .30103).

Dat dit voordelen heeft wordt het beste geïllustreerd met het feit, dat met deze conventies de capaciteitsrestricties (voor positieve nul) luiden:

$$-1 \leqslant x \leqslant 1-2^{-mq} \ .$$

In het inversensysteem verdient het de voorkeur, om elk blok het teken van het lange getal te geven. Dan luiden n.l. de capaciteitsrestricties analoog aan het enkele woord

$$-(1-2^{-mq}) \leqslant x \leqslant 1-2^{-mq}$$

Met betrekking tot de additieve bewerkingen valt - ongeacht of de machine uitgerust is met een echte dubbel-lengte accumulator - de vergelijking tussen beide systemen uit ten gunste van het complementen-systeem. Weinig verschil maakt het bij additie van twee getallen van hetzelfde teken: de optelling wordt uitgevoerd te beginnen bij het minst significante blok, de overdracht, die ontstaat wordt bij de optelling der volgende blokken in rekening gebracht. Bij het complementen-systeem is deze overdracht 0 of 1, bij het inversen-systeem is deze overdracht +1 of +0 bij additie van positieve getallen, -1 of -0 bij die van negatieve getallen. Bij het complementen-systeem gaat de additie van twee getallen van verschillend teken even gemakkelijk; bij het inversen-systeem echter is de inhoud van het minst significante blok van het antwoord afhankelijk van het uiteindelijke teken van de som, dus van iets wat pas na afloop van de optelling bekend is. De optelling van twee getallen van verschillend teken geschiedt nu in twee fasen: eerst worden de getallen bloksgewijs opgeteld (overdracht is uitgesloten!); als aan het meest significante blok $\neq 0$ het teken van het antwoord gedetecteerd is, wordt het "teken consistent" gemaakt.

De ARMAC heeft geen dubbel-lengte accumulator; evenwel is dankzij de schuifopdrachten, welker verrichtingen, zoals in een machine werkend met het inversen-systeem voor de hand

ligt, volslagen symmetrisch zijn in de enen en de nullen, een simpele mogelijkheid geboden om de "getekende overdracht" te isoleren, zoals deze voorkomt bij de additie van getallen van hetzelfde teken.

Bij een vermenigvuldiging van twee teken consistente getallen kan hiervan uiteraard alle vruchten geplukt worden.

De multi-lengte routines van de ARMAC zijn bestemd voor het rekenen met breuken, en niet net heel grote gehele getallen. Wat gewoonlijk als deling wordt aangeduid is hier dus vanwege de veronachtzaming van de rest een quotientberekening. Hiervoor is een proces gekozen, waarbij multi-lengte optellingen slechts hoeven worden toegepast op getallen van hetzelfde teken. Er is gebruik gemaakt van een variant van de relatie:

$$\frac{a}{1-c} = (1+c)(1+c^2)(1+c^4)(1+c^8) \ldots\ldots \qquad |c|<1$$

Evenals bij toepassing van bovenstaande formule nodig, resp. gewenst is, worden teller en noemer van teken gewisseld, als de noemer negatief is en worden ze beide met 2^n vermenigvuldigd, n zo gekozen, dat de nieuwe noemer b voldoet aan de ongelijkheden

$$\tfrac{1}{2} \leq b < 1$$

Als we de zo verkregen teller en noemer met a resp. b aanduiden zou het boven gegeven proces aanleiding geven tot het rekenschema:

Inleiding: $(q_0 = a)$ en $c_0 = 1-b$
Repetitie: $q_{i+1} = q_i + q_i c_i$ en $c_{i+1} = c_i^2$
Resultaat: " als $c_i = 0$, dan geldt $q_i = \frac{a}{b}$ ".

Omdat dit proces per stap twee vermenigvuldigingen van multi-lengte getallen vergt is de volgende variant gekozen:

Inleiding: $(q_0 = a$ en $b_0 = b)$
Repetitie: $c_i \approx 1-b_i$ $q_{i+1} = q_i + q_i c_i$ en
$b_{i+1} = b_i + b_i c_i$

Resultaat: "als $b_i = c$, dan geldt $q_i = \frac{a}{b}$ "

Als in de repetitie exact $c_i = 1-b_i$ gekozen wordt, zijn de processen identiek; we kiezen nu voor c_i een getal, dat slechts in één blok van nul afwijkt (en wel zó, dat

$c_i \leq 1-b_i$, m.a.w. de q_i blijft het quotient van onderen naderen, de c_i blijven dus positief).

De twee vermenigvuldigingen per stap zijn nu teruggebracht tot vermenigvuldigingen van een multi-lengte getal met het enkele-lengte getal c_i. Een en ander gaat - tenslotte - ten koste van de quadratische convergentie. Desondanks is het proces in deze vorm in tijd, en zeker in programmaruimte, aanmerkelijk voordeliger.

Hoewel we het niet hebben toegepast, zie ik geen enkel bezwaar om de quadraat-wortel op een analoge wijze te berekenen: nl.

Inleiding: $w_0 = b.2^{\frac{1}{2}n}$ en $b_0 = b.2^n$ zodat $\frac{1}{2} \leq b_0 < 1$
(Ter versnelling van de convergentie):

Repetitie: $c_i \approx \frac{1}{2}(1-b_i)$ $w_{i+1} = w_i(1+c_i)$ en $b_{i+1} = b_i(1+c_i)^2$

Resultaat: als $b_i = 1$, dan is $w_i = b^{\frac{1}{2}}$.

De dubbel-lengte worteltrekking in de ARMAC b.v. maakt er expliciet gebruik van, dat het getal maar in dubbele lengte gegeven is: het argument wordt tussen $\frac{1}{4}$ en 1 genormeerd: dan wordt de (enkele lengte) wortel met de normale wortel-subroutine getrokken, deze wordt met één naslag m.b.v. Newton's iteratie tot tweevoudige lengte verbeterd, en over het halve aantal binaire plaatsen teruggeschreven.

Resumerend moet worden opgemerkt, dat de multi-lengte routines voor de ARMAC nog al ernstig zijn beinvloed door de beperkingen van de machine. Het inversen-systeem is hiervan de minste. Meer sporen hebben achtergelaten de afwezigheid van de ingebouwde deling van het grotere snelle geheugen en wellicht van de "volle" dubbel-lengte accumulator.