PROGRAMMEREN VOOR DE A.R.R.A.

door

A. van Wijngaarden,

Rekenafdeling Mathematisch Centrum.

MR 7.

1951.

INHOUD

1. Eerste kennismaking.

De ARRA is een ADM (automatische digitale machine). In het volgende wordt de theorie van de programmering behandeld voor het stadium, waarin de machine het eerst gebracht zal worden na de ombouw volgend op het eerste stadium van gebruik in 1950. Zoveel mogelijk zal getracht worden daarbij alle technische kwesties te omzeilen. Pas gaandeweg zal zoveel geintroduceerd worden als strict noodzakelijk is.

De machine bezit een geheugen G. In dit geheugen bevinden zich een aantal zg. " woorden " op zg. " adressen ". Een woord is een getal, maar kan ook op andere wijzen door de machine worden geinterpreteerd, bijv. als een zg. " opdracht ", dat is een hoeveelheid informatie, die de machine vertelt welke handeling zij moet uitvoeren. Een adres is een plaatsaanduiding in G. De adressen zijn genummerd 0, 1, 2... enz., voorlopig nog beperkt maar later tot en met 2047. De inhoud van adres n geven wij aan met (n).

Het tweede orgaan van de machine is de besturing B. Deze haalt een woord uit G en interpreteert dit als een opdracht. Een gedeelte van het woord wordt eenvoudig genegeerd, een tweede gedeelte stelt de zg. " operatie " voor, bijv. "tel op", "trek af", "berg op", etc., en het derde gedeelte tenslotte is meestal een adres. Meestal betekent deze opdracht dan, dat die operatie moet worden uitgevoerd met het getal, dat zich bevindt op dat adres. De besturing leest dan het getal op dat adres, voert de operatie uit en leest dan de volgende opdracht, dat is normalerwijze de opdracht op het adres volgende op dat waarop de zo juist gehoorzaamde opdracht zich bevond. Op deze wijze wordt een "programma" van opdrachten uitgevoerd.

Het derde orgaan van de machine is het rekenkundig orgaan. Dit bestaat uit twee registers A en S, het best te vergelijken met het resultaatregister en omwentelingsregister van een gewone rekenmachine. Alleen hebben zij beide dezelfde lengte. Het derde noodzakelijke register van een rekenmachine, het instelregister ofwel toetsenbord is er wel, het heeft zelfs een naam, nl. M, maar de programmeur heeft er niets mee te maken, omdat de besturing de getallen er op inslaat!

Na deze minimuminformatie kunnen wij al overgaan tot het maken van enkele eenvoudige programma's. Daartoe maken wij een lijstje van enkele eenvoudige opdrachten, die wij schrijven als "operatie, adres". Er zijn totaal 16 operaties, genummerd 0, 1, 2...,15.

0/n : Tel bij A op het getal op adres n
8/n : Trek van A af het getal op adres n.
10/n : Schrijf het getal in A op adres n en maak A schoon.
1/n : Schrijf in S het getal op adres n.
11/n : Schrijf het getal in S op adres n en maak A schoon.

Bevindt zich het getal a op adres 200, b op 240 en wordt gevraagd a+b te schrijven op 275 en 0 te schrijven op adres 280. A zij schoon om te beginnen. Het benodigde programma omvat vier opdrachten. Die moeten natuurlijk ergens in G staan en wij zullen aannemen, dat de eerste opdracht zich bevindt op adres 100. Dan is het programma:

```
100   0/200
101   0/240
102  10/275
103  10/280
```

Om 2a-b neer te schrijven op 275 dient het volgende programma:

```
100   0/200
101   0/200
102   8/240
103  10/275
```

Soms is het adresgedeelte van de opdracht geen eigenlijk adres, maar een getal. Dit is het geval bij de volgende twee opdrachten:

6/n : Vermenigvuldig de inhoud van A met 2^n.
14/n : Vermenigvuldig de inhoud van A met 2^{-n}.

Het laatste voorbeeld had dus bijv. ook zo kunnen worden geprogrammeerd:

```
100   0/200
101   6/1
102   8/240
103  10/275
```

Om $2a-\frac{1}{2}b$ neer te schrijven op 275 dient dit programma:

```
100   0/200          of      100   8/240
101   6/2                    101  14/1
102   8/240                  102   0/200
103  14/1                    103   0/200
104  10/275                  104  10/275
```

Men ziet, dat vele wegen naar Rome leiden.

Het operatiegedeelte van de opdracht behoeft niet een rekenkundige bewerking te zijn. Dit is bijv. het geval bij de volgende twee opdrachten:

7/n : Als de inhoud van A $\geq$ 0 is, voer dan de opdracht op adres n uit, zo niet ga dan gewoon door.

15/n : Als de inhoud van A $<$ 0 is, voer dan de opdracht op adres n uit, zo niet ga dan gewoon door.

Gewoon doorgaan betekent: de volgende instructie lezen en uitvoeren. Als de "besturing verspringt" naar adres n, dan is de volgende instructie natuurlijk n+1 enz.

Deze twee opdrachten, de verschuivingen van de besturing zijn zeer fundamenteel. Een belangrijke toepassing vinden zij overal daar, waar een cyclus van opdrachten een aantal malen herhaald dient te worden. Is dit aantal van te voren voorgeschreven, dan kan een zg. "telroutine" toegepast worden. Als voorbeeld mag dienen het geval, dat een zekere cyclus van opdrachten n keer uitgevoerd moet worden en n moge zich bevinden op adres a. Verder zij b een hulpadres (werkruimte). Tenslotte bevat adres 47 altijd de constante 1. Het geheel ziet er dan als volgt uit:

```
          c     8/a
d+2 →     c+1   10/b
          c+2   ....  ┐
          ...   ....  ├  cyclus van opdrachten, die A schoon
          ...   ....  ┘  achterlaat.
          d     0/b
          d+1   0/47
          d+2   15/c+1 →
```

Bij het begin van dit programma wordt -n in b opgeborgen, een cyclus van operaties uitgevoerd en vervolgens de inhoud van b, die wij in het vervolg met (b) zullen aanduiden in A opgeteld en met 1 vermeerderd. Is het resultaat negatief, dan verspringt de besturing weer naar adres c+1, d.w.z. (A), d.i. -n+1 wordt in b opgeborgen en de cyclus wordt herhaald. Als de cyclus n maal is doorlopen is de inhoud van A voor de opdracht 15/c+1 gelijk aan -n+n=0 en de besturing springt niet naar c+1, maar loopt gewoon door. Zoals men ziet, laat deze telroutine A schoon achter, wat normaliter vereist is en staat ze klaar om op ieder ogenblik opnieuw gebruikt te worden. De pijltjes dienen om het overzicht van het programma te vergemakkelijken.

Uit het feit, dat opdrachten eigenlijk ook getallen zijn, volgt, dat de machine ook haar opdrachten aan rekenkundige bewerkingen kan onderwerpen. Om te zien, hoe dat gaat, moeten wij eerst weten, op

welke wijze een opdracht is samengesteld. Dit is heel eenvoudig in principe. Een opdracht zij f/a. Hierin is f het operatiegedeelte, dus $0 \leqslant f \leqslant 15$ en a is het adresgedeelte, dus $0 \leqslant a \leqslant 2047$. In G staat deze opdracht als het getal 2048 f+a. Dit kan dus nooit boven zekere grenzen uitkomen. In werkelijkheid is de capaciteit van de registers veel groter, maar als een getal in G als opdracht wordt geinterpreteerd neemt de besturing automatisch de kleinste positieve rest van het getal modulo 32768 en deze rest, de eigenlijke opdracht kan op één en slechts één wijze geschreven worden in de vorm 2048 f+a. Het getal 1233333 in G als opdracht geinterpreteerd is aequivalent met 1233333 - 37 x 32768 = 20917 = 10 x 2048 + 437 = 10/437. Alleen als het getal negatief is, moet om redenen, die later duidelijk zullen worden, het verkregen adres met 1 worden verminderd. Dus het getal -1234833 in G als opdracht geinterpreteerd is aequivalent met -1234833 + 38 x 32768-1 = 10350 = 5 x 2048 + 110 = 5/110. In verreweg de meeste gevallen zal deze opdracht zich in G reeds als de kleinste positieve rest modulo 32768 bevinden, in ieder geval tenminste als geen speciale tegenmaatregelen zijn genomen.

Wil men nu bij het adresgedeelte van een opdracht een constante optellen, dan behoeft men blijkbaar slechts bij de opdracht die constante op te tellen. Als voorbeeld berekenen wij de som van de getallen op adressen 100, 101 ... 149 en bergen het antwoord op op adres 150. De routine moet meerdere malen bruikbaar zijn en A schoon achterlaten. Een mogelijk programma is het volgende:

```
          c      10/150
          c+1     0/c+11
c+10 →    c+2    10/c+3
          c+3    (10/0)
          c+4    0/150
          c+5    10/150
          c+6    0/c+3
          c+7    8/c+12
          c+8    7/c+13 →
          c+9    0/c+4
          c+10   7/c+2 ⇒
          c+11   0/100
          c+12   0/149
c+8 →     c+13   .....
```

In dit programma zijn verschillende nieuwtjes ingevoerd. Opdracht c maakt adres 150 schoon. Opdracht c+1 telt in A opdracht c+11 op, dat is 0/100, ofwel eenvoudig het getal 100. Opdracht c+2 plaatst die op adres c+3 neer en deze wordt nu dus gehoorzaamd,d.w.z.

het getal op adres 100 wordt in A opgeteld. Opdracht c+4 telt de inhoud van adres 150, d.w.z. nul, erbij op en de "som" wordt door opdracht c+5 weer op adres 150 opgeborgen. Opdracht c+6 telt nu opdracht c+3, dus 0/100 op in A en opdracht c+7 trekt daar de inhoud van adres c+12, dus 0/49 ofwel eenvoudig het getal 149, van af. Het resultaat is natuurlijk negatief en de opdracht c+8 heeft dus geen effect. Opdracht c+9 telt nu opdracht c+4, d.i. 0/150 ofwel eenvoudig het getal 150 er weer bij op en het resultaat is nu, dat in A staat 0/101 i.p.v. 0/100. Dit is positief en opdracht c+10 verplaatst de besturing terug naar c+2. Er staat hier een pijl met een dubbele schacht, wat betekent, dat deze verplaatsing onvoorwaardelijk is, omdat als de besturing daar komt (A) noodzakelijkerwijze positief is.

Opdracht c+2 plaatst nu (A) weer op c+3 neer en nu wordt dus het getal op adres 101 in A opgeteld. Daarbij wordt weer de vorige som geteld en de som weer opgeborgen op adres 150. Dit gaat zo door tot het getal op adres 149 onder handen is genomen. Dan is (c+3) 0/149 geworden en opdracht c+7 laat nul achter in A. Opdracht c+8 verplaatst de besturing nu uit de cyclus, d.w.z. naar adres c+13. Inderdaad is A nu schoon en de gehele cyclus, mits binnengekomen aan het begin natuurlijk, d.w.z. bij c met schone A staat altijd voor gebruik gereed. Merk op, dat de eigenlijke telling nu is uitgevoerd aan de variabele opdracht zelf. Voorts is een van de opdrachten, nl. c+4 eenmaal als opdracht en eenmaalals getal gebruikt, nl. in opdracht c+9. Optelling is een zeer veel voorkomende opdracht en zoals uit dit voorbeeld blijkt, verschaft de volkomen willekeurige toekenning van het nummer 0 aan de optelopdracht aan een programma de merkwaardige eigenschap soms voor zijn eigen constanten te zorgen.

Het doet er niets toe, wat oorspronkelijk op adres c+3 stond, omdat het programma zelf er de passende opdracht neerzet voor de besturing er aan toe is. Wij hebben er de opdracht 10/0 neergezet. Dit is een heel bijzondere opdracht, die wij later zullen behandelen, maar tot resultaat heeft de machine te stoppen. Het voordeel hiervan is, dat als er iets mankeert aan het programma of aan de machine en op adres c+3 niet de juiste opdracht is neergeschreven, de besturing de machine zal stoppen op dit punt.

Vaak kan economischer gehandeld worden dan tellen als in de bovengenoemde voorbeelden en wel door het aanbrengen van een zg. etiket. Stel bijv. dat in het bovengenoemde voorbeeld alle termen positief zijn. Door er een negatief getal, het etiket, direct achter te plaatsen, kan het tellen achterwege blijven. De som kan dan natuurlijk niet vlak achter de termen geplaatst worden. Laat ons daarom veronderstellen, dat het aantal termen kleiner dan 50 is. Het

programma kan er dan als volgt uitzien:

```
          c      10/150
          c+1     0/c+10
c+9 →     c+2    10/c+3
          c+3    (10/0)
          c+4    15/c+11 →
          c+5     0/150
          c+6    10/150
          c+7     0/c+3
          c+8     0/48
          c+9     7/c+2 ⇒
          c+10    0/100
c+4 →     c+11   10/c+3
```

Dit programma is één adres korter dan het vorige. De hoofdcyclus is 8 opdrachten lang i.p.v. 9 in het vorige voorbeeld. Ruwweg betekent dit een snelheidswinst van 10%. Adres c+3 is als vuilnisbak gebruikt door opdracht c+11, die alleen dient om A schoon te maken.

Niet alleen het adresgedeelte maar ook het operatiegedeelte van de opdracht kan men aan rekenkundige bewerking onderwerpen. Om het opdrachtnummer van een opdracht met a te verhogen resp. te verlagen, behoeft men slechts 2048 a er bij op te tellen resp. van af te trekken. Bedenkt men evenwel, dat de opdrachten door de besturing toch slechts modulo 32768 beschouwd worden, dan ziet men dat men het opdrachtnummer ook bijv. met a kan verlagen door bij de opdracht 2048(16-a) op te tellen. Een merkwaardige consequentie hiervan is dat men opdrachten, wier nummer 8 verschilt door gedurig optellen van 16384 elkaar kan laten afwisselen. De opdrachtennummers zijn zo gekozen, datdie opdrachten, die in hun karakter slechts in algebraisch teken verschillen, bijv. tel op (0) en trek af (8), vermenigvuldig met 2^n(6) en met 2^{-n}(14), spring als (A)$\geqslant$0 (7) en als A$<$0 (15), nummers hebben die 8 verschillen. Hierdoor kan men geraffineerde trucs uithalen. Als voorbeeld dient de opgave de reeks getallen 5, 7, 11, 13, 17, 19 ... te vormen, d.w.z. oneven getallen, die geen drievouden zijn. De differenties van deze reeks zijn blijkbaar de getallen 2, 4, 2, 4, 2,... ofwel 3-1, 3+1, 3-1, 3+1, 3-1,...

Het volgende programma presteert dit (adres 63 bevat het getal -3):

	c	0/48
	c+1	10/d+3
	c+2	0/c
c+d+2 →	c+3	10/c+6
	c+4	0/d+3
	c+5	8/63
	c+6	(1/0)
	c+7	10/d+3
	...	programma, dat de oneven
	...	niet-drievouden op d+3
	...	gebruikt en A schoon achterlaat.
	d	0/c+6
	d+1	0/d+4
	d+2	7/c+3 ⇒
	d+3	(10/0)
	d+4	16384

De opdracht c+6 die aan het begin op 0/48 wordt ingesteld, wordt nu één cyclus vervangen door 8/48, de tweede cyclus door 0/48, enz. Om later te verklaren redenen loopt de zaak in de war als men dit meer dan ca 65000 malen achtereen uitvoert.

Alvorens over te gaan tot het behandelen van een volgende opdracht moet iets meer gezegd worden over het rekenkundig orgaan. Tot nu toe hebben wij nl. de capaciteit van de registers in het geheel niet besproken. Deze nu is voor beide gelijk en wel kunnen zij gehele getallen g bevatten, welke voldoen aan:

$$-536870912 < g < 536870912$$

Over deze wonderbaarlijke getallen komen wij later te spreken. Voorlopig is deze informatie echter voldoende. Natuurlijk moeten wij bij optellen, aftrekken, enz. er zorg voor dragen deze capaciteit niet te overschrijden.

De eerstvolgende opdrachten zijn de vermenigvuldiging en deling:

4/n : Vorm P=(S)x(n)+(A). Splits het getal P in twee delen P_A en P_S met hetzelfde teken, zodat $P = 536870912\ P_A + P_S$ en plaats P_A in A en P_S in S.

12/n : Vorm $Q = \left[\{536870912\ (A) + (S)\} : (n)\right]$, daarbij de kleinst mogelijke rest R met hetzelfde teken als (A) en (S) in A en het quotiënt Q in S plaatsend.

Zoals men ziet, zijn deze twee opdrachten reciprook, vandaar, dat hun nummers 8 verschillen. Wanneer (A), (S) en (n) kleine getallen zijn, zodat $|(S) \times (n) + (A)| < 536870912$ is, staat het product alleen in S. Daarbij moet echter toch nog opgelet worden. Is

(S)×(n) + (A) immers negatief, dan bevat A toch altijd een minusteken, d.wz. -0. Op zichzelf is dat niet erg als er iets bij moet worden opgeteld, want arithmetisch is -0 = 0, maar als men vlak op zo'n vermenigvuldiging een besturingsverplaatsing zou willen laten volgen moet men uitkijken of men 7/n of 15/n nodig heeft. In geval van twijfel is overigens de combinatie 7/n, 15/n altijd safe, want dan wordt de besturing in ieder geval verplaatst.

Is het product (S)×(n) + (A)>536870912, dan moet men er op letten, dat zich zowel in A als in S een gedeelte van het product bevindt. Beide delen, nl. P_A en P_S dragen het teken van het product, wat gemakkelijk is bij latere bewerkingen.

Bij het uitvoeren van een deling op een "lang" getal, d.w.z. op een getal>536870912 moet men er voor zorgen, dat zowel A als S eerst met de delen van dit getal worden gevuld, alvorens de deling uit te voeren. Is het getal < 536870912, d.w.z. is het getal "kort", dan moet men er op letten, dat A schoon is gemaakt.

Over het teken van A in geval het getal negatief zou zijn behoeft men zich geen zorgen te maken, aangezien de besturing zelf als (A) en (S) verschillend teken hebben aan (A) eerst het teken van (S) toekent alvorens met de deling te beginnen.

Als eerste eenvoudig voorbeeld de berekening van (m)×(n)→ S als het product kort is. A zij schoon aan het begin.

c	1/m
c+1	4/n
c+2	11/s

Na afloop is A schoon, S daarentegen bevat nog (m)X(n). Willen wij dus ook nog (m)×(n)×(p)→t verricht hebben, aangenomen, dat ook dit nieuwe product kort is, dan behoeven slechts de twee volgende opdrachten te worden toegevoegd:

c+3	4/p
c+4	11/t

Wat aardiger is de berekening van een polynoom, bijv. ax^2+bx+c. Gemakshalve onderstellen wij, dat alle optredende getallen kort zijn en verder duiden wij met a', b', c', x' aan de adressen, waarop zich de getallen a, b, c, x bevinden.

A zij schoon aan het begin:

c	1/a'
c+1	0/b'
c+2	4/x'
c+3	0/c'
c+4	4/x' .

Nu bevat S de vorm $ax^2 + bx + c$. Eventueel mag dit een lang getal zijn, als ax + b maar kort was.

Moet een lang getal weggeborgen worden in G, nu natuurlijk op 2 adressen, zeg s en t, bijv. het lange product (m)x(n) → s,t, dan moet zorg gedragen worden voor de volgorde:

c	1/m
c+1	4/n
c+2	10/s
c+3	11/t

Als men de volgorde van de laatste twee opdrachten had verwisseld liep het mis, want de opdracht 11/t laat A schoon achter!

Als voorbeeld van de deling geven wij het volgende vraagstuk. Bereken het lange product (m)x(n) = P. Dit is te schrijven in de vorm: $P = p_1 \times 10^8 + p_o$. Bereken p_1 en p_o en berg ze op op adressen t en t+1. Het getal 10^8 bevindt zich op adres 50.

c	1/m
c+1	4/n
c+2	12/50
c+3	10/t+2
c+4	11/t+1

Dit proces is zeer belangrijk, want het geeft een methode om een lang getal in het honderdmillioentallig stelsel te schrijven. Dat is nl. juist, wat wij gedaan hebben. De "cijfers" van P in dit stelsel zijn p_1 en p_o. Het zal later blijken, waarom wij zo gebrand zijn op de basis 10^8. Het is overigens de grootste macht van 10, welke nog binnen de capaciteit van een register ligt.

2. Achter de schermen.

In het eerste hoofdstuk hebben wij al bijna heel het opdrachtenlijstje de revue laten passeren en het zou niet moeilijk zijn om in de gevolgde trant door te gaan. Er zijn echter bezwaren aan deze techniek verbonden. De cursist heeft nl. al gemerkt, dat gaandeweg meer en meer vreemde getallen en conventies opdoken en het zou de indruk kunnen wekken, dat de gehele organisatie tamelijk onlogisch was. Dit is echter geenszins het geval en het is nu een goed moment eens iets over de werkwijze van de machine zelf te leren omdat dan zal blijken hoe eenvoudig deze kwesties achteraf zijn.

Het bijzondere van het geval is nl. dat de machine niet in het tientallig, maar in het tweetallig stelsel werkt. Wij hebben al gezien, dat de programmeur zich daar niet druk om behoeft te maken, want in het eerste hoofdstuk hebben wij al heel wat geprogrammeerd zonder er iets van te weten. De - zeer kleine - moeilijkheden komen nu.

Ieder register, zowel de adressen in G als ook A en S bieden plaats aan 30 cijfers, die slechts de waarden nul of een kunnen hebben. Ieder woord heeft dus ook 30 cijfers, genoemd $d_0, d_1, d_2 \ldots d_{29}$.

Een getal N wordt per definitie voorgesteld door:

$$N = \sum_k 2^{29-k}(d_k - d_0) \ .$$

Is $d_0 = 0$ dan zijn de termen van de som hetzij nul hetzij positief, is daarentegen $d_0 = 1$, dan zijn de termen hetzij nul hetzij negatief. Dus als $d_0 = 0$, dan is $N \geqslant 0$ en als $d_0 = 1$, dan is $N \leqslant 0$. Het cijfer d_0 fungeert dus als teken met de bijzonderheid dan, dat 0 zowel een positieve vorm +0 heeft, nl. een reeks van 30 nullen, als ook een negatieve vorm -0, nl. een reeks van 30 enen. De machine heeft een grondige afkeer van -0 en produceert vrijwel altijd +0. Als men om een of andere reden een -0 in A wil krijgen kan dat alleen met behulp van een truc gebeuren. Als resultaat van optelling of aftrekking kan de nul in ieder geval nooit optreden en dus telt nul practisch mee met de positieve getallen. Vandaar, dat de opdrachten 7/n en 15/n de nul ook bij de positieve getallen rekenden. (Zij reageren alleen op d_0!).

Het getal -N ontstaat uit N door alle cijfers te "inverteren", d.w.z. door alle enen in nullen en alle nullen in enen te veranderen. Dit volgt onmiddellijk uit de definitie. Deze inversie is geen rekenkundige bewerking en kan zeer eenvoudig physisch verwezenlijkt worden. Zij is niet toegankelijk voor de programmeur maar geschiedt door de besturing.

Voorbeeld:

00000 00000 00000 00000 00000 I0III = 23,
IIIII IIIII IIIII IIIII IIIII 0I000 = -23.

Het grootste positieve getal N dat wij voor kunnen stellen is blijkbaar een rij geopend door een nul en gevold door enkel enen, d.i. $2^{29} - 1 = 536870911$. Het grootste negatieve getal N is een rij geopend door een één en gevolgd door louter nullen, d.i. -536870911. Het getal $2^{29} - 1$ bevindt zich permanent op adres 56.

Uit de definitie van N volgt, dat een negatief getal ook geinterpreteerd kan worden als het binaire positieve geval voorgesteld door de cijfers $d_0, d_1, \ldots d_{29}$ minus $(2^{30} - 1)$. Dit betekent, dat de machine eigenlijk rekent modulo $2^{30} - 1$, daarbij steeds de absoluut kleinst mogelijke rest nemend. Dit verklaart direct het merkwaardige feit, dat door 1 op te tellen bij $2^{29} - 1$ ontstaat $-2^{29} + 1$. Dit is modulo $2^{30} - 1$ inderdaad correct. Dit impliceert echter nog iets. Twee negatieve getallen staan in de machine als N_1 en N_2 en worden slechts negatief door de interpretatie $N_1 - 2^{30}+1$ en $N_2 - 2^{30}+1$. Telt de machine ze op, dan zou ze in eerste instantie leveren $N_1 + N_2$ geinterpreteerd als $N_1 + N_2 - 2^{30} + 1$. Dit is niet correct, want er treedt nu onherroepelijk een tweetallenoverdracht op uit d_0 naar links. Deze gaat verloren, dus er ontstaat $N_1+N_2-2x2^{30}+1$ i.p.v. $N_1 + N_2 - 2x2^{30} + 2$. Hieraan wordt nu tegemoetgekomen, doordat de overdracht die links ontstond rechts weer ingevoerd wordt en dan klopt de zaak. Dit is de zg. "terugoverdracht" (end-around carry), die blijkbaar essentieel is voor dit soort rekenen. Nu ziet men ook, waarom het trucje om de opdracht 0 in 8 en die 8 weer in 0 te veranderen door gedurig optellen van 16384 uiteindelijk spaak loopt. Op de duur loopt nl. het register links over en de terugoverdracht verknoeit het adres van de opdracht.

De merkwaardige definitie van de vermenigvuldigings- en de delingsopdrachten wordt nu ook duidelijk. A en S worden hier aan elkaar gekoppeld met uitzondering van de tekencijferplaats van S die buiten schot blijft.

3. De breuk.

Tot nu toe hebben wij de getallen in de adressen en registers als gehele getallen beschouwd. Evenals het op een tafelrekenmachine echter mogelijk is kunstmatig kommaplaatsen vast te leggen is dit ook op de ARRA mogelijk. De meest aan de machine aangepaste plaats is die tussen d_o en d_1. Zo'n breuk zullen wij door n voorstellen en dan is blijkbaar

$$n = \sum_K 2^{-k}(d_k - d_o).$$

In sommige opzichten is het gemakkelijker met breuken te rekenen. Het is nu bijv. ineens duidelijk, dat

$$-1 < n < 1$$

en dit is gemakkelijk te onthouden. De precisie waarin breuken kunnen worden voorgesteld is nu natuurlijk eindig aangezien het kleinste positieve getal nu 2^{-29} is. Dit zij p genoemd (peuter); $p = 1.8 \times 10^{-9}$, dus de precisie is ca. 8,5 decimaal.

Als met breuken en gehele getallen door elkaar heen wordt gerekend is het verstandig consequent de gehele getallen van de letter p te voorzien. Dit vergemakkelijkt het overzicht waar de resultaten terecht komen bij vermenigvuldiging en deling. Zo stelt bij gekoppeld A en S register de laatste plaats van A,p en die van S,p^2 voor. Dus is de eenheid bij vermenigvuldigen van 2 breuken d_o van A, van een breuk en een geheel getal d_{29} van A; van twee gehele getallen d_{29} van S.

Meestal zal men, wanneer men met breuken werkt, ook met afgeronde getallen werken. Dit betekent, dat men na het vormen van een product zich niet interesseert voor wat zich in S bevindt, maar dat men daarentegen het gedeelte in A af wil ronden door 2^{-30} erbij op te tellen Dat gaat niet zo eenvoudig, omdat alleen in A opgeteld kan worden en 2^{-30} natuurlijk juist buiten het bereik van A valt. Evenzo zal bij een deling meestal geen staart van het deeltal in S geplaatst(kunnen) worden en ook de rest interesseert ons meestal niet, maar wel wil men meestal het quotient afronden, wat wederom niet zo heel eenvoudig is.

Aan deze moeilijkheden wordt tegemoet gekomen door de volgende twee opdrachten:

5/n : Vorm P = (S)X(n) en plaats het afgeronde product in A.
13/n : Vorm Q = (A) :(n) en plaats het afgeronde quotient in S.

Als reciproke opdrachten verschillen hun nummers weer 8. Let op, dat blijkens de definitie bij 5/n resp. 13/n niet geeist wordt, dat A resp. S van te voren schoon is.

Als voorbeeld de berekening van (m) Ax (n)→ s. A schoon aan het begin.

c	1/m
c+1	5/n
c+2	10/s

Als ingewikkelder voorbeeld de berekening van $e - 2 = \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \ldots$, waarbij wij geen enkele constante in ons programma mogen opnemen. Dit kan bijv. zo:

	c	0/48!
	c+1	0/48!
	c+2	10/c+2
	c+3	10/c+3
	c+4	0/48
	c+5	12/c+2
	c+6	11/c+1
	c+7	0/c+1
	c+8	8/48!
	c+9	15/c+18 →
	c+10	0/48!
	c+11	0/c+3
	c+12	10/c+3
	c+13	0/c+2
	c+14	0/48!
	c+15	10/c+2
	c+16	1/c+1
	c+17	7/c+5 ⇒
c+9 →	c+18	10/c+2

Er zijn hier weer enkele trucjes toegepast. Het programma kan maar een maal worden gebruikt, want opdracht c+6 vermoordt opdracht c+1 om plaatsruimte te maken voor 1/k!, terwijl opdrachten c+2 en c+3 zelfmoord plegen om plaatsruimte te maken voor k+1 en $\frac{1}{2!} + \frac{1}{3!} + \ldots \frac{1}{k!}$. Het antwoord e-2 bevindt zich na afloop op adres c+3. Let ook goed op de manipulaties aan het begin om de beginconstanten in orde te brengen en de eerste deling in te leiden.

4. Het negende gebod.

Wij missen nu nog de opdrachten 2/n, 3/n en 9/n. De eerste twee zijn voorlopig gereserveerd voor veranderingen in de opbergmethode van opdrachten en bestaan voor ons nu niet, maar opdracht 9 is van groot belang:

9/n : Doe iets, op grond van specificatie op adres n.

Dat zegt niet veel. De voornaamste functie van deze opdracht is, dat de machine getuigenis af kan leggen van haar resultaten door ze uit te typen op de schrijfmachine. Een nevenfunctie is, dat de machine gegevens die haar toegevoerd worden op een geperforeerde band kan lezen. In het algemeen verzorgt ze dus de communicatie van de machine met de buitenwereld.

Het uittypen van resultaten kan van verschillende aard zijn. Allereerst kan de machine 16 verschillende schrijfmachine handelingen verrichten, nl. typen van een 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, +, -, ., wagen terug brengen gecombineerd met papieropvoering, tabuleren en spatieren. Dit noemen wij "imperatief typen". Dit geschiedt zodra (n) een getal is, waarvoor $d_0 = d_4 = d_5 = I$ is, wonderlijk genoeg. Het getal gevormd door de tetrade $d_{24}\ d_{25}\ d_{26}\ d_{27}$ geeft dan aan welke handeling geschiedt, dus 0000 typt dan een 0; 000I typt een 1; ...; I00I typt een 9; I0I0 typt een +; I0II typt een -; II00 typt een .; II0I geeft terug wagen; III0 tabuleert en IIII geeft spatie. Gelukkig heeft de programmeur hier niets mee te maken, want zulke "codewoorden" bevinden zich al bij de 64 geblokkeerde adresplaatsen 0,1,...,63, dat zijn nl. adressen waar zich een bepaalde groep woorden van groot belang bevinden en waarin niet geschreven kan worden (wel uit gelezen). Probeert men hier in te schrijven, dan stopt de machine. Daarom is het voldoende te weten, dat

9/5 : T 0, (Typt 0),
9/3 : T 1, (Typt 1),
9/14 : T 2, (Typt 2),
9/57 : T 3, (Typt 3),
9/10 : T 4, (Typt 4),
9/15 : T 5, (Typt 5),
9/58 : T 6, (Typt 6),
9/25 : T 7, (Typt 7),
9/59 : T 8, (Typt 8),
9/62 : T 9, (Typt 9),
9/16 : T +, (Typt +),
9/42 : T -, (Typt -),
9/6 : T ., (Typt .),
9/61 : TW , (Terugwagen, nieuwe regel),
9/60 : Tab, (Tabuleert),

Verder verrichten deze opdrachten allemaal evenals trouwens elke 9/n opdracht iets heel merkwaardigs, nl. ze keren het teken van (A) om indien (A)<0 is, dus (A) wordt door |(A)| vervangen. Dit is, zoals zal blijken, in het kader van de gehele 9/n opdracht een nuttige eigenschap, maar kan ook arithmetisch gebruikt worden. Wil men nl. |(A)| in A hebben dan geeft men een onschuldige opdracht als 9/63!

Van de bovenstaande opdrachten zal men de eerste 10 waarschijnlijk slechts zelden gebruiken, immers de machine heeft de mogelijkheid hele getallen uit te typen i.p.v. alleen cijfers. Zo'n getal is |(A)| , maar dit kan nog worden geinterpreteerd op verschillende manieren.

Geldt voor het codewoord d_0 = I, maar niet, althans niet gelijktijdig, $d_4 = d_5$ = I, dan beschouwt de besturing |(A)| als een breuk en typt het uit, ondertussen het deconverterend in het tientallig stelsel door voortgezette vermenigvuldiging met tien. Zijn d_j en d_k de eerste cijfers van het codewoord na d_3 die I zijn, dan typt de machine eerst j-4 cijfers, daarna een punt en daarna nog k-j-1 decimalen en maakt tenslotte A schoon. Dit is niet in tegenspraak met het feit, dat de breuk in de machine absoluut kleiner dan 1 was, want dit is slechts een interpretatie van de machine Rekent men bijv. het getal e = 2.71828... uit, dan doet men dat misschien door e/10 = 0.271828... door de machine te laten berekenen en bij het uittypen de decimale punt een plaats naar rechts te laten verschuiven.

Ook nu behoeft de programmeur zich weer geen zorgen te maken. Alle codewoorden van dit type die hij practisch nodig kan hebben, bevinden zich reeds in de machine in het geblokkeerde gebied. Geven wij kortheidshalve het typen van een breuk met a cijfers voor de punt en b er achter aan met het symbool Ba.b, dan zijn de volgende opdrachten zonder meer mogelijk:

9/2 : B 0.8	9/13 : B 1.7	9/23 : B 2.6	9/31 : B 3.5
9/4 : B 0.7	9/17 : B 1.6	9/24 : B 2.5	9/32 : B 3.4
9/7 : B 0.6	9/18 : B 1.5	9/26 : B 2.4	9/33 : B 3.3
9/8 : B 0.5	9/19 : B 1.4	9/27 : B 2.3	9/34 : B 3.2
9/9 : B 0.4	9/20 : B 1.3	9/28 : B 2.2	9/35 : B 3.1
9/11 : B 0.3	9/21 : B 1.2	9/29 : B 2.1	9/36 : B 3.0
9/12 : B 0.2	9/22 : B 1.1	9/30 : B 2.0	
9/37 : B 4.4	9/43 : B 5.3	9/50 : B 6.2	9/53 : B 7.1
9/38 : B 4.3	9/44 : B 5.2	9/51 : B 6.1	9/54 : B 7.0
9/39 : B 4.2	9/45 : B 5.1	9/52 : B 6.0	
9/40 : B 4.1	9/46 : B 5.0		
9/41 : B 4.0			

Er zijn nog enkele bijzonderheden op te merken. Bij B 0.8, B 0.7,...,B 0.2 typt de machine eerst een punt zonder spatie of nul ervoor. Bij B 1.7, B 1.6,..., B 1.1 typt de machine altijd één cijfer voor de punt, ook wanneer dit een nul is. Bij de andere opdrachten echter vervangt de machine alle eventuele cijfers nul voor aan het getal door spaties (zg. "facultatief typen"). Als alle cijfers voor de punt nul zijn typt zij echter de nul voor de punt wel en zodra eenmaal iets getypt is (anders dan spaties !) gaat het typen imperatief verder, d.w.z. nullen worden niet meer onderdrukt. Als voorbeeld geven wij machten van 10 zoals gedrukt zouden worden door B 3.3 en B 0.3:

	B 3.3	B 0.3
10^{-3}	0.001	.001
10^{-2}	0.010	.010
10^{-1}	0.100	.100
10^{0}	1.000	
10^{1}	10.000	
10^{2}	100.000	

Alvorens een breuk uit te typen moet zij eerst afgerond worden door $5 \times 10^{-k-1}$ op te tellen bij de modulus, als k het aantal getypte cijfers is, d.w.z. de som van het aantal cijfers voor en dat van die achter de punt. In de machine staan de cijfers immers alle achter de punt en de plaatsing van de punt is iets zuiver kunstmatigs. Bij B 0.3 moet dus worden afgerond met 5×10^{-4} en bij B 3.3 met 5×10^{-7}. Bij de bovengenoemde B a.b's loopt het totaal aantal getypte cijfers van 2 tot en met 8 en de corresponderende afrondingen nl. 5×10^{-3} tot en met 5×10^{-9} bevinden zich ook al in het geblokkeerde gebied, overigens in negatieve vorm en wel niet geheel nauwkeurig, wat met het oog op de beperkte precisie van een woord ook niet goed mogelijk is. Voor alle practische doeleinden zal de precisie echter volkomen voldoende zijn. Wil men nog nauwkeuriger waarden hebben, wat soms kan, omdat de getallen ook als codewoorden dienen en daarom wel eens een weinigje veranderd zijn, dan moet men ze in het programma opnemen. De getallen zijn:

Adres:	Waarde ongeveer:	Waarde nauwkeurig:
57	-5×10^{-3}	$-4.999971017 \times 10^{-3}$
58	-5×10^{-4}	$-5.000308156 \times 10^{-4}$
59	-5×10^{-5}	$-5.000084638 \times 10^{-5}$
60	-5×10^{-6}	$-5.014240742 \times 10^{-6}$
61	-5×10^{-7}	$-4.973262548 \times 10^{-7}$
62	-5×10^{-8}	$-5.029141903 \times 10^{-8}$
63	-5×10^{-9}	$-5.587935448 \times 10^{-9}$

Als voorbeeld typen wij het getal e in 6 decimalen, dat wij berekend hadden in het laatste voorbeeld van het vorige hoofdstuk. Wij plakken er eenvoudig enige opdrachten achter:

	c+19	0/c+3	2-e in A
	c+20	8/61	Rondt af op 6 dec.
	c+21	9/14	Typt 2
	c+22	9/7	Typt ·718282 en maakt A schoon.

Als tweede voorbeeld zij te typen (A) als B 2.4 voorafgegaan door een teken en een spatie:

	c	7/c+3 →	
	c+1	9/42	Typt - en keert teken van (A) om!
	c+2	7/c+4 →	
c →	c+3	9/16	Typt +
c+2 →	c+4	9/63	Spatieert
	c+5	8/61	Rondt af op 6 dec.
	c+6	9/26	Typt B 2.4 en maakt A schoon

Geldt voor het codewoord $d_0 = 0$ en $d_5 = 0$, dan beschouwt de besturing (A) als een geheel getal van 8 cijfers en typt |(A)| uit. Is $d_4 = 0$, dan worden eventuele nullen aan het begin van het getal door spaties vervangen en wel hoogstens 7 maal (facultatief typen). Is $d_4 = I$, dan worden alle nullen getypt. (Dit laatste treedt op bij het typen van lange getallen, die in blokken van 8 cijfers uitgetypt worden). In beide gevallen wordt A schoon achtergelaten.

Codewoorden die hieraan voldoen zijn weer aanwezig, en wel luiden de opdrachten:

9/0 : GGF (Typt geheel getal facultatief),
9/1 : GGI (Typt geheel getal imperatief).

Uit het bovenstaande blijkt, dat een geheel getal, dat getypt moet worden hoogstens 8 cijfers mag hebben. Is het getal, dat aan 9/0 of 9/1 wordt onderworpen groter, dan wordt de positieve rest modulo 10^8 getypt, d.w.z. de laatste acht cijfers van het getal.

Geldt voor het codewoord $d_0 = 0$, $d_4 = 0$, $d_5 = I$, dan leest de machine een rij gaatjes (pentade) van de ponsband, plaatst de pentade achter in A en voert de band één plaats op. Hierbij worden de pentaden 00000 en IIIII genegeerd; bij lezing hiervan wordt de volgende rij gelezen. Een passend codewoord is aanwezig, zodat een opdracht luidt:

9/49 : LB (Leest nieuwe pentade op de band).

5. De band.

Er is reeds enkele malen gesproken over de ponsband, waarop de machine een berekeningsprogramma voorgelegd krijgt. De organisatie hiervan zal hier nu besproken worden. De machine bevat in het geblokkeerde gebied een aantal opdrachten, te beginnen op adres 0, het zg. invoerprogramma. Bij de aanvang wordt nu de band in de bandlezer gelegd, de machine wordt schoongemaakt door drukken op een knop en de besturing start op opdracht 0. Dit heeft tot effect, dat de machine de band gaat lezen en alle opdrachten en getallen, die er op staan, op de gewenste plaatsen in het geheugen gaat opbergen en tenslotte begint dit programma uit te voeren, daarbij startende op de gewenste plaats. Op de band moet dus meer informatie aanwezig zijn dan alleen de opdrachten en getallen. Bovendien heeft het invoerprogramma nog wat ruimte buiten het geblokkeerde gebied nodig als werkruimte, omdat het immers in het geblokkeerde gebied niet schrijven kan. Hoeveel ruimte dit is, hangt af van een nog nader te bespreken kwestie, doch in elk geval moeten de adressen 64 tot en met 70 vrijgehouden worden en hoogstens ook nog de adressen 71 tot en met 83. Als men niet dringend om ruimte verlegen zit, doet men er dus verstandig aan niets voor adres 84 te schrijven en anders moet men aan de hand van nog nader te bespreken zaken nagaan hoe veel eerder men kan beginnen.

Op het toetsenbord van de ponsmachine, waarop men de band ponst bevinden zich 30 toetsen en wel:

16 toetsen, genummerd: 0(+), 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15.
14 toetsen, genummerd: 70,71,72,73(+.),74,75(-.) 76, 77, 78(-)
79, 80, 81(A), 82, 83.
en voorts nog een spatietoets en een correctietoets.

Men ziet, dat 5 toetsen gemakshalve een dubbele benaming hebben. Iedere molecuul informatie wordt op de band geponst in drieen, nl. eerst een "beginletter", dan een "getal" en tenslotte een "eindletter".

De beginletter is in het geval dat het molecuul een opdracht is,eenvoudig het opdrachtnummer, in te slaan op één enkele toets van de eerste serie. In het geval, dat het een getal betreft, is zij -., -, +., +, al naar gelang het getal een negatieve breuk, een negatief geheel getal, een positieve breuk, dan wel een positief geheel getal is, in te slaan op één enkele toets. In het geval, dat het een opbergadres is, is de beginletter een A, in te slaan op één enkele toets.

Het getal is eenvoudig het adres of getal zelf, cijfer voor cijfer in te slaan op toetsen van de eerste serie. Daarbij behoeven alleen significante cijfers te worden ingeslagen, d.w.z. nullen voor een getal behoeven niet te worden aangeslagen. Bij een breuk moeten echter precies acht decimalen worden ingevoerd, dus eventueel moeten nullen achteraan gesuppleerd worden. Een geheel getal mag nooit groter zijn in absolute waarde dan 536870911. Begint het getal met 10, 11, 12, 13, 14 of 15 dan mogen deze twee begincijfers desgewenst ook ineens op de corresponderende toetsen worden ingeslagen maar dit geldt alleen voor de twee begincijfers. Enige voorbeelden mogen dit verduidelijken:

Getal	Te ponsen
348	+,3,4,8,
34800	+,3,4,8,0,0,
- 126	-,1,2,6, of -,12,6,
0.12345678	+·,1,2,3,4,5,6,7,8, of +·,12,3,4,5,6,7,8
0.1234	+·,1,2,3,4,0,0,0,0, of +·,12,3,4,0,0,0,0
- 0.003456	-·,3,4,5,6,0,0,
0.00005678	+·,5,6,7,8,
0	+,
Opdracht	
0/79	0,7,9,
12/123	12,1,2,3, of 12,12,3,

De eindletter tenslotte heeft een heel speciale functie. Zij telt nl. bij het getal op de inhoud van het adres, dat zij voorstelt. Het invoerprogramma maakt adres 70 schoon en de eindletter 70 is dus altijd onschuldig. Wil men echter in een reeks opdrachten zekere adressen met een constante vermeerderd zien voor de machine ze opbergt, dan plaatst men op de band de eindletter 71 i.p.v. 70 en zorgt van te voren, dat zich in 71 de gewenste constante bevindt. Eventueel kan men verschillende constanten toevoegen, in totaal 13, want er zijn 14 toetsen van de tweede serie, maar aan 70 mag niet geknoeid worden! Het belang hiervan zal in het volgende hoofdstuk blijken. Het zal slechts zelden voorkomen, dat men achter getallen een andere sluitletter plaatst dan 70, aangezien hoofdzakelijk de wijziging van adressen van opdrachten het doelmerk is, maar men zou het kunnen doen. In dat geval moet men weten, dat een breuk als natuurlijk getal in de machine wordt ingevoerd, dán de eventuele correctie wordt aangebracht en tenslotte pas door 10^8 gedeeld en het teken in orde gebracht wordt.

De correctiebedragen in de registers 71, 72 etc. mogen alleen positieve getallen zijn.

Voor het gemak is op de speciale programmaformulieren van de ARRA het getal 70 altijd al in donkergrijze inkt achter alle woorden gedrukt. Aleen als men 71 of 72 enz. wenst te hebben, behoeft men de 0 van 70 slechts door 1 of 2 enz. te vervangen.

Ieder programma begint dus met

A / n / 70

waarin n het beginadres is van een serie op te bergen woorden. Zolang geen nieuwe adresindicatie volgt, blijft de machine de woorden op opeenvolgende adressen opbergen, dus op adres n, n+1, n+2, enz.

Zodra men een serie woorden op wil bergen op adressen m,m+1, m+2, enz. geeft men eenvoudig een nieuwe indicatie A / m / 70.

Wil men van een serie getallen eventuele adressen wijzigen door optellen van de constante a, dan laat men de combinatie voorafgaan

A / 71 / 70
+ / a / 70
A / m / 70

en van alle opdrachten die nu van de eindletter 71 zijn voorzien, wordt het adres met a vermeerderd. In het algemeen luidt het recept:

A / 71 / 70
+ / a / 70
+ / b / 70
+ / c / 70
. . . . . .
A / m / 70

en van alle opdrachten, die van de eindletter 71, 72, 73... zijn voorzien wordt het adres met a, b, c... vermeerderd.

Tenslotte wil men het door de machine opgeborgen programma beginnen op opdracht m. Dit geschiedt door de combinatie:

A / 69 / 70
7 / m / 70

achter aan de band. Dit is natuurlijk onbegrijpelijk, maar zal in het volgende hoofdstuk, dat het invoerprogramma behandelt, verklaard worden.

Tenslotte nog de mededeling, dat spatie , d.w.z. ongeponste band, en correcties, d.w.z. volgeponste band, volkomen genegeerd worden. Naar willekeur en hartelust mag dus spatie en correctie toegepast worden.

Als voorbeeld van een volledig programma van een eenvoudige berekening geven wij nog eens de berekening en typing van het getal

e in zes decimalen, als behandeld in hoofdstuk 3 en 4. Let op, dat de programmeur niet alle 70's en 71's behoeft in te vullen, doch alleen een 1 voor een 71 op het voorgedrukte papier. Wij bergen het programma willekeurig op, zeg op adres 90 te beginnen:

		A	71	70	
		+	90	70	
		A		71	
	0	0	48	70	
	1	0	48	70	
	2	10	2	71	
	3	10	3	71	
	4	0	48	70	
17/71 →	5	12	2	71	
	6	11	1	71	
	7	0	1	71	
	8	8	48	70	
	9	15	18	71	→
	10	0	48	70	
	11	0	3	71	
	12	10	3	71	
	13	0	2	71	
	14	0	48	70	
	15	10	2	71	
	16	1	1	71	
	17	7	5	71	⟹
9/71 →	18	10	2	71	
	19	0	3	71	
	20	8	61	70	
	21	9	14	70	
	22	9	7	70	
	23	1		70	
		A	69	70	
		7		71	

Alleen het gedeelte tussen de stippellijnen moet worden geponst. De rest is slechts hulpmiddel voor de programmeur. Let op, dat de eigenlijke plaatsaanduiding in het geheugen, nl. 90 slechts eenmaal is geponst. Men kan daar, binnen zekere grenzen natuurlijk, elk getal neerzetten, dat men wenst, zonder aan de werking van het geheel iets te veranderen. De gehele berekening inclusief lezen van de band geschiedt binnen één minuut. Het is dus een geschikte demonstratieberekening of controle van de machine. Het geponste gedeelte van het bandje is ruim 24 cm lang. Om het begin en het eind van het bandje

te onderscheiden is het verstandig na het programma en wat spatie een aantal correctietekens aan te brengen. Deze markeren dan duidelijk de achterzijde. Voor hen, die alles erg mooi willen hebben bestaat ook het recept achter aan de band te ponsen:

Correctie, 6,6,6,6, spatie, spatie, correctie, spatie, spatie, correctie 71,73,77, correctie, spatie, spatie, correctie, 2,2,2, 83, spatie, spatie, correctie, 6,6,6,6, spatie, spatie, 1.

Dan staat nl. netjes in de band geponst: EINDE.

6. Het prille begin.

Wij zullen nu het invoerprogramma bezien. Dit is strikt gesproken niet nodig voor de programmeur, omdat de direct nodige gegevens verstrekt zijn in het vorige hoofdstuk. Evenwel verschaft de kennis van het invoerprogramma beter inzicht in dat hoofdstuk en bovendien is het een goed voorbeeld van wat ingewikkelder programmering.

Allereerst moet verteld worden met welke pentaden op de band de aanduidingen op de toetsen overeenkomen. Dit zijn:

Op toets	Op band	Op toets	Op band
0,+	15	70	1
1	16	71	2
2	17	72	3
3	18	73,+.	4
4	19	74	5
5	20	75,-.	6
6	21	76	7
7	22	77	8
8	23	78,-	9
9	24	79	10
10	25	80	11
11	26	81,A	12
12	27	82	13
13	28	83	14
14	29	Spatie	0
15	30	Correctie	31

Blijkbaar zijn de bandwaarden 15 groter dan de toetswaarden in de eerste serie en 69 kleiner dan de toetswaarden in de tweede serie. De bandlezer zelf negeert de spaties en correcties. De 0 en de 31 worden dus nooit aan de machine toegevoerd.

Het invoerprogramma start bij 0. De opdrachten 0 en 1 plaatsen de opdracht 7/21 op adres 67. Opdrachten 2 en 3 plaatsen de opdracht 7/43 op adres 69 en opdracht 4 maakt adres 70 schoon. Dit is alles voorbereiding, welke slechts eenmaal plaats heeft. Nu begint het verwerken van een molecuul informatie. Opdracht 5 maakt adres 64, dat het getal herbergt, schoon om het in gereedheid te brengen een nieuw getal te helpen opbouwen. Opdracht 6 leest de beginletter en opdracht 7 bergt ze op op adres 65. Opdracht 8 leest de volgende pentade en opdracht 9 trekt er 15 vanaf om te zien of het soms de sluitletter is. (Opdracht 0 is hier als constante, nl. 15, gebruikt. De opdracht 7/21 die zich op adres 15 bevindt is juist daar neerge-

zet, midden tussen het programma in, om opdracht 0 die vorm te geven!). Immers alle cijfers zijn $\geqslant 15$ en alle sluitletters <15. Valt het antwoord niet-negatief uit, dan heeft het cijfer juist zijn getalwaarde herkregen en opdrachten 11 en 12 tellen er tienmaal het getal in 64 (nu nog 0) bij op. Opdracht 13 stuurt het getal, dat nu een stadium verder in zijn opbouw is naar 64 terug en opdracht 14 verschuift de besturing weer naar 8, zodat opnieuw een cijfer wordt gelezen. Dit gaat zo een poosje door totdat de eindletter wordt gelezen. Het getal is dan voltooid in 64 en opdracht 10 verschuift de besturing naar opdracht 18. Register A bevat nu de eindletter -15. Opdracht 18 telt hier de opdracht 0/84 bij op, zodat A nu bevat 0/eindletter +69. Opdracht 19 stuurt deze opdracht naar adres 66 en vervolgens verschuift de besturing naar dit adres. Nu wordt dus de inhoud van adres 69+ eindletter in A opgeteld. Is de eindletter 1 (toets 70), dan is die inhoud nul, want adres 70 was schoongemaakt. Opdracht 67 schuift de besturing terug naar 21, want er was afgesproken, dat de adrescorrecties, gezet op adressen 71 enz. positief zouden zijn. Opdracht 21 telt nu het getal bij de correctie op en opdracht 22 stuurt het gecorrigeerde getal weer naar 64 terug. Nu gaat opdracht 23 de beginletter in A neerzetten en opdracht 24 trekt er weer 15 vanaf om te zien of het een opdracht dan wel een getal of adresindicatie is. Het + teken, dat dus een niet-negatief geheel getal inleidt is wel hetzelfde als het opdrachtnummer 0, maar dit betekent eenvoudig, dat wij het getal n invoeren als een pseudoopdracht 0/n, wat toch precies hetzelfde is!

Valt het resultaat negatief uit, dan verschuift opdracht 25 de besturing naar opdracht 29, die de opdracht 7/43 bij A optelt. A bevat dus nu de opdracht 7/beginletter +28. Opdracht 30 bergt deze opdracht op op adres 65 en opdracht 31 verschuift de besturing naar dit adres. Opdracht 65 verschuift de besturing nu dus naar het adres "beginletter +28", dat is dus naar adres 32, 34, 37 of 40 al naar gelang de beginletter 4, 6, 9 of 12, d.wz. +., -., - of A was. In het laatste geval (dat is altijd het geval bij het eerste molecuul op de band!) telt opdracht 40 het adres op in A, opdracht 41 telt er de opdracht 10/0 bij op, zodat de inhoud van A is de opdracht 10/adres. Adres 50 bevat in werkelijkheid een negatief getal, omdat links van de eigenlijke opdracht ook nog wat staat. Daarom is $(A)<0$ en opdracht 42 verschuift de besturing naar opdracht 45, die de opdracht 10/adres wegbergt op adres 68, waarna opdracht 46 de besturing terugzendt naar 5, zodat een volgend molecuul informatie gelezen wordt.

Was de beginletter 4, d.w.z. +., dan werd de besturing naar 32 gestuurd. Opdrachten 32 en 33 draaien het teken van 64 om en de be-

sturing komt bij 34, waar zij in het geval van de letter -. ineens heen gezonden was. Opdrachten 34, 35 en 36 delen de inhoud van 64 door 10^8, waardoor het getal (dat 8 cijfers zou hebben volgens afspraak) omgezet wordt in een breuk. De besturing komt dan op adres 37, waar ze ineens heengezonden was in het geval van de letter -, en opdracht 37 trekt het getal in 64 af in A. A bevat nu in alle die gevallen precies het getal zoals het bedoeld was (met twee tekenwisselingen in het geval +. 1), maar dit kan nu zowel positief als negatief zijn, zodat twee opdrachten 38 en 39 nodig zijn om in elk geval de besturing te verleggen naar adres 68.

Wij verlaten nu het getal even om te zien, wat er met een opdracht gebeurt. De aftrekking door opdracht 24 valt dan positief uit en de besturing gaat recht door naar 26. Door de aftrekking van 15 is de beginletter net tot zijn juiste getalwaarde herleid en opdracht 26 schuift dit opdrachtnummer 11 plaatsen naar links, terwijl opdracht 27 er het adres bij optelt, zodat de opdracht gereed in A staat. Opdracht 28 verschuift ook nu de besturing naar adres 68.

Zowel in het geval van een getal als van een opdracht arriveert de besturing dus op adres 68 met register A in het bezit van het op te bergen woord. Adres 68 bevat nu de opdracht 10/adres en het woord wordt op dit adres opgeborgen. Opdracht 69 zendt de besturing nu terug naar opdracht 43, die opdracht 68 optelt in A, terwijl opdracht 44 het adres erin bevat met 1 verhoogt. Evenals zoeven bergt opdracht 45 deze vernieuwde opdracht terug op adres 68 en, tenzij het volgende molecuul een adresindicatie is, wordt het opgeborgen op het volgende adres.

Zo leest de machine het gehele programma van de band af, totdat het uiteindelijk de slotcombinatie A/69/70, 7/m/70 ontmoet. De adresindicatie zet opdracht 68 op 10/69, zodat de volgende opdracht 7/m opgeborgen wordt op adres 69, daarmede de opdracht 7/43, die er zich bevond vermoordend. Direct na dit opbergen loopt de besturing er dan ook zelf in en springt naar m in plaats van naar 43, zoals anders.

Het kan zijn, dat het in te voeren programma te groot is, om ineens in de machine te worden ingevoerd. Dit kan bijv. het geval zijn, als het een groot aantal getallen bevat. Als het zo ingericht kan worden, dat alvast een deel van de berekening kan worden uitgevoerd met een kleinere hoeveelheid materiaal, dan kan deze berekening vast beginnen en wanneer het programma nieuw materiaal van de band nodig heeft, kan ze dit aan laten rukken. Dit kan eenvoudig geschieden, door de besturing naar adres 0 te laten verspringen. Op

de band moeten dan echter weer merkers aangebracht zijn, die het eigenlijke programma aan het werk zetten. Een soepeler methode is, het programma zo in te richten, dat het zelf de strategische punten 67 en 69 van passende opdrachten voorziet en dan het invoerprogramma aan te laten roepen door verschuiving van de besturing naar adres 4 i.p.v. 0. Het invoerprogramma loopt dan aan de leiband van het werkprogramma en geen speciale merkers zijn meer op de band nodig. Dit zal i.h.a. verreweg de beste methode zijn.

Het invoerprogramma:

	0	0/15
	1	10/67
	2	0/16
	3	10/69
	4	10/70
46 →	5	10/64
	6	9/49
	7	10/65
14 →	8	9/49
	9	8/0
	10	15/18 →
	11	1/48
	12	4/64
	13	11/64
	14	7/8 ⟹
	15	7/21
	16	7/43
	17	0/84
10 →	18	0/17
	19	10/66
	20	7/66 ⟹
67 →	21	0/64
	22	10/64
	23	0/65
	24	8/0
	25	15/29 →
	26	6/11
	27	0/64
	28	7/68 ⟹
25 →	29	0/16
	30	10/65
	31	7/65 ⟹
(+.) 65 →	32	8/64
	33	10/64
(-.) 65 →	34	0/64
	35	13/49
	36	11/64
(-) 65 →	37	8/64
	38	7/68 →
	39	15/68 ⟹
(A) 65 →	40	0/64
	41	0/50
	42	15/45 ⟹
69 →	43	0/68
	44	0/47
42 →	45	10/68
	46	7/5 ⟹
	47	1
	48	10
	49	10^8
	50	10/0

7. De sleur.

Het zou hoogst onpractisch zijn als ieder programma geheel van de grond af zou moeten worden opgebouwd. Immers zou men steeds stukken tegenkomen, die men in andere programma's ook had toegepast. Het ligt dus voor de hand standaardprogramma's, zg. subroutines, te maken, waar men veel tijd aan kan besteden om ze zo handig mogelijk in te richten, en deze op banden te ponsen, die men naar willekeur kan copieren op de band, die men ponst voor zijn eigen programma. Men is er dan bovendien verzekerd van, dat de kans op fouten in deze delen van het programma tot het minimum is gereduceerd.

Het simpelste soort subroutine is de zg. "open subroutine", welke zonder meer in het programma ingelast wordt. Zo'n subroutine kan dan alleen maar op die plaats van het programma gebruikt worden. Wil men haar op verschillende plaatsen van het programma toepassen, dan moet men haar ook verschillende malen copieren, wat veel geheugenruimte en inleestijd kost.

Een meer geraffineerde vorm is de "gesloten subroutine". Deze vormt een eenheid op zichzelf. Zij kan op ieder punt van het programma worden ingeroepen met behulp van twee bepaalde opdrachten en is zo ingericht dat zij na beeindiging van haar taak de besturing terugzendt naar het adres direct volgend op die twee aanroepopdrachten. Dit werkt als volgt: In het geblokkeerde gebied bevindt zich op adres 46 de opdracht 7/5 in de vorm van een negatief getal. Stel, dat de subroutine zich bevindt op adres d en dat het hoofdprogramma haar op adres c wil gebruiken. Dan luiden de aanroepopdrachten:

	c	0/c-3	Telt adres -3 op in A
	c+1	7/d ⟹	Besturing naar subroutine.
d+e ⟶	c+2	.....	

en de subroutine luidt als volgt:

d	0/46	Vormt 7/c+2
d+1	10/d+e	
d+2	.....	e-2 opdrachten, die de eigenlijke subroutine vormen en die A schoon achterlaten
.....	.....	
....	.....	
d+e	(10/0)⟹	Wordt 7/c+2, besturing naar hoofdprogramma.

Dit is het eenvoudigste geval.

Op de band ziet de subroutine er als volgt uit:

	A	71	70
	+	d	70
	A		71
0	0	46	70
1	10	e	71
2	.	. . . . .	
..	.	. . . . .	
e	(10	0	70)

en iedere opdracht in de subroutine, welke verwijst naar een adres in de subroutine zelf, heeft een eindletter 71, terwijl die opdrachten, die naar een absoluut adres verwijzen de eindletter 70 hebben. Hier ziet men duidelijk het grote voordeel van de adrescorrectie door middel van de eindletter. De subroutine, zoals zij als geponst bandje van speciale kleur in de bibliotheek aanwezig is, bevat nl. alles onder de streep. De programmeur ponst op de band A/71/70, +/d/70. Dit kan natuurlijk niet op de band geponst staan, want het beginadres d, waarop de subroutine wordt geplaatst, is ter beschikking van de programmeur. Voorts hoeft hij de band van de subroutine slechts te copieren op zijn band, hetgeen automatisch door een reproduceerpons geschiedt. De eindletters zorgen er dan voor, dat de adressen van alle opdrachten in de machine de juiste waarden krijgen.

Stel nu het geval, dat de subroutine een of meer andere subroutines nodig heeft. Bijv. zal de bibliotheek een subroutine bevatten, die een reëel nulpunt bepaalt van een functie, zodra twee argumenten gegeven worden, waarvoor de functiewaarden van verschillend teken bezit. Zo'n subroutine moet natuurlijk weten hoe ze de waarde van de functie moet berekenen en heeft daarvoor dus een speciale subroutine als knechtje nodig. Misschien bevat de functie een wortelteken en haar subroutine zal dan gebruik maken van de subroutine, die een vierkantswortel van een getal bepaalt. Het kan dus zijn, dat een subroutine aanroepopdrachten bevat van andere subroutines. De adressen hiervan zijn natuurlijk nog niet bekend, maar allen van verschillende eindletters, dus 72, 73,... voorzien en op de band moeten deze gegevens vooraf naar adressen 72,73,... gestuurd worden. Een subroutine, die zelf starten moet op adres d en twee subroutines nodig heeft, aangeduid door 72 en 73, die starten moeten op e en f, moet dus voorafgegaan worden op de band door de indicaties A/71/70, +/d/70, +/e/70, +/f/70 en op de subroutineband staat vooraf weer A//71. Natuurlijk staat in de beschrijving van elke subroutine nauwkeurig aangegeven, wat de aan-

roepcombinatie is en wat op de band voorafgeponst moet worden.

De aanroepcombinatie kan nl. ook anders zijn, dan in het voorbeeld om een andere reden. Het kan nl. voorkomen, dat de subroutine een of meer parameters bevat, bijv. wanneer zij de n-de machtswortel van een getal bepaalt. In het eenvoudigste geval wordt het getal n als zg. "vaste parameter" bijv. via adres 72 toegevoegd. Overal, waar de subroutine dan n wil optellen bij een getal of adres bevat zij de eindletter 72. Gedurende de gehele berekening kan de subroutine dan slechts voor die enkele waarde van n gebruikt worden. Algemener van toepassing is de subroutine met zg. "programmaparameters". Dit zijn parameters, die het hoofdprogramma iedere keer, dat het de subroutine inroept, aangeeft en dus gedurende de berekening gedurig kunnen veranderen. Deze parameters worden nu niet geïncorporeerd in de subroutine gedurende de band-invoer. Er zijn vele mogelijkheden om met deze programmaparameters te manipuleren. Stel bijv. weer het geval van een subroutine, die de n-de machtswortel bepaalt, maar nu voor variabele n. De aanroepcombinatie zou nu kunnen luiden:

	c	0/c-2
	c+1	7/d ⟹
	c+2	n
d+e ⟶	c+3	

en dezelfde organisatie van de subroutine, als zoeven stuurt de besturing nu terug naar c+3 i.p.v. naar c+2, op welk adres nu de programmaparameter gespecificeerd is. De subroutine kan deze parameter vinden, omdat ze immers zodra ze aangeroepen is het adres c+2 kan vormen. Dit kan op vele wijzen gebeuren en kan van geval tot geval verschillen. Voor de programmeur is alleen van belang de aanroepcombinatie en de benodigde voorponsing te kennen. Hij hoeft de subroutine zelf niet te bestuderen!

Als voorbeeld van een subroutine met vaste parameters geven wij een zg. regelindelingssubroutine voor gelijksoortige positieve getallen, alle te typen volgens het voorschrift 9/t, bijv. voor gewone gehele getallen: t = 0 en voor gewone breuken met zes decimalen achter de punt: t = 7. Het te typen getal, zo nodig al afgerond, bevindt zich op adres m. De subroutine typt n zulke getallen op een regel en gaat dan op een nieuwe regel over. Daartoe moeten de tabulatorstoppen van de schrijfmachine ingesteld zijn volgens de kolomindeling, die men wenst te hebben. Tenslotte moet de subroutine geplaatst worden op adres d. Er moeten nu dus op de band vier getallen gegeven worden, nl. d, m, n en t.

De subroutine luidt:

	A	71	70	
	+	d	70	d in 71
	+	m	70	m in 72
	+	n	70	n in 73
	+	t	70	t in 74
	A		71	
0	0	46	70	} vormt
1	10	9	71	} koppelopdracht
2	0		72	getal in A
3	9		74	typt getal
4	0	14	71	} trekt 1 af van (14),
5	8	47	70	} oorspronkelijk n-1
6	15	10	71	⟶ springt bij eindregel
7	10	14	71	n-1 opbergen
8	9	60	70	tabelleert
9	(10		70)	⟹ koppelopdracht
10	0	15	71	} n-1 opbergen
11	10	14	71	} op (14)
12	9	61	70	nieuwe regel
13	7	9	71	⟹ naar koppelopdracht
14	-	1	73	oorspronkelijk n-1
15	0		73	bevat n

De subroutineband bevat dus alles onder de streep. Voorgeponst moet worden wat boven de streep staat. De aanroepcombinatie is de normale:

```
        c       0/c-3
        c+1     7/d⟹
d+9 ⟶   c+2
```

Een ander voorbeeld van een subroutine is de volgende, die de vierkantswortel van de breuk a op adres a' ($0 \leqslant a < 1$) berekent en het antwoord $x = \sqrt{a}$ op adres x' plaatst. De adressen a', x' en d (plaats van de subroutine) worden als vaste parameters meegegeven.

De berekening geschiedt door iteratie volgens Newton:

$$x_{n+1} = \tfrac{1}{2}(x_n + a/x_n) \quad , x = \lim x_n .$$

Deze vorm is voor het rekenen echter niet erg prettig, want $x_n + a/x_n$ kan allicht groter dan 1 zijn. Daarom schrijven wij

$$x_{n+1} - x_n = \tfrac{1}{2}(- x_n + a/x_n).$$

Wanneer men er nu zorg voor draagt, dat de beginschatting $x_o > x$, dan geldt verder algemeen, dat $x_n > x$, omdat, zoals men gemakkelijk ziet, $x_{n+1} - x_n < 0$ is en de limiet x dus van de bovenzijde wordt benaderd. Zowel x_n als a/x_n zijn kleiner dan 1 en hun verschil ook. Bij de berekening kunnen dus geen ongelukken optreden doordat getallen groter dan 1 optreden. Hoever moet men doorgaan met itereren? Doorgaans is a natuurlijk niet exact gegeven, maar is slechts bekend een grootheid $a' = a + \theta p$, met in het gunstigste geval $|\theta| < \frac{1}{2}$. Men kan dus stoppen, zodra $x_{n+1}^2 - a < p/2$. Nu is $2 x_n x_{n+1} = x_n^2 + a$, dus $(x_n - x_{n+1})^2 = x_{n+1}^2 - a$, dus kan worden gestopt zodra $x_n - x_{n+1} < \sqrt{p/2} = 2^{-15}$ bij de ARRA. Het kan natuurlijk geen kwaad een wat kleiner getal als criterium te kiezen, bijv. (10)= 2029 $p < 2^{-18}$.

Voor de beginschatting van de wortel moet eigenlijk 1 worden gekozen, als men zeker wil zijn, dat ze groter is dan de wortel. De eerste iteratiestap levert dan $x_1 = \frac{1}{2}(1+a)$, wat een geschikt uitgangspunt levert. Als wij het geval a = 1-p uitsluiten, kunnen wij ook starten met $x_o = 1-p = -(56)$. Dit zullen wij hier doen. Het programma wordt dan iets korter, wat betreft het aantal opdrachten, maar wat langer in tijdsduur. Een mogelijk programma is dan

		A	71	70	
		+	d	70	d in 71
		+	a'	70	a' in 72
		+	x'	73	x' in 73
		A		71	
	0	0	46	70	} vormt
	1	10	19	71	} koppelopdracht
	2	0	56	70	} $x_0 = 1-p$
	3	10		73	} in x'
15 →	4	0		72	} $x_{n+1} - x_n =$
	5	13		73	} $\frac{1}{2}(-x_n + a/x_n)$
	6	11	64	70	} in A via 64,
	7	0	64	70	} dat als doorgangs-
	8	8		73	} huis gebruikt
	9	14	1	70	} wordt
	10	8	10	70	} controleert grootte
	11	7	16	71	→ } van $x_{n+1} - x_n$
	12	0	10	70	} vormt x_{n+1} en
	13	0		73	} bergt op in
	14	10		73	} x'.
	15	7	4	71	⇒ Herhaal iteratie.
11 →	16	0	10	70	} Vormt x en
	17	0		73	} bergt op in
	18	10		73	} x'.
	19	(10		70)	⇒ Koppelopdracht.

De aanroepcombinatie is de normale:

```
                c       0/c-3
                c+1     7/d ⇒
    d+19 ⟶      c+2
```

Natuurlijk zijn vele variaties mogelijk. Zo is het waarschijnlijk wel zo handig x af te leveren in A, i.p.v. op een speciaal adres x'. Dan vervalt eenvoudig opdracht 18. Voor x' kan dan een onschuldig adres als bijv. 65 gekozen worden.

8. De verandering.

In paragraaf 4 zeiden wij, dat de opdrachten 2 en 3 nog gereserveerd waren met het oog op een geprojecteerde wijziging in het systeem van opbergen van opdrachten. Intussen is echter besloten voorlopig aan deze 2 opdrachten een andere betekenis toe te kennen. Zij zullen nl. gelijk effect hebben als opdrachten 10 en 11 behalve dan, dat register A na afloop niet schoongemaakt wordt. Nu zou men menen, dat de inhoud van A dus ongewijzigd zou blijven, maar dit is alleen het geval met opdracht 2/n. Het systeem van opbergen is nl. zó, dat altijd uit A opgeborgen wordt naar G. Als uit S opgeborgen moet worden, plaatst de besturing dus eerst (S) in A en bergt het dan pas op op adres n. Niet schoonmaken van A na afloop betekent hier dus, dat (S) in A achterblijft. De opdrachten luiden dus:

2/n Schrijf (A) neer op adres n.
3/n Schrijf (S) neer op adres n en in A.

Opdracht 2/n is identiek met de twee opdracht 10/n, 0/n. Overal, waar deze twee opdrachten in het voorgaande dus achter elkaar voorkomen, kunnen ze dus worden vervangen door de enkele opdracht 2/n.

Opdracht 3/n is identiek met de twee opdrachten 11/n, 0/n. Waar deze twee achter elkaar voorkomen, kunnen zij dus ook hier weer door een opdracht 3/n worden vervangen. In dit geval kan het zelfs gebeuren, dat het opbergen op adres n ons eigenlijk helemaal niet interesseert, maar dat het enige doel is om (S) in A te krijgen. Voor n kan dan een of ander onschuldig adres worden gekozen.

9. Lijst van opdrachten.

0,n	Telt (n) op bij (A).
1,n	Plaatst (n) in S.
2,n	Schrijft (A) op adres n.
3,n	Schrijft (S) op adres n en in A.
4,n	Vormt (n)×(S) + p×(A) en plaatst het resultaat in A en S.
5,n	Vormt (n)×(S) + ½ p en plaatst het resultaat in A en S.
6,n	Vermenigvuldigt (A) met 2^n.
7,n	Verplaatst besturing naar n als (A)≥0.
8,n	Trekt (n) af van (A).
9,n	Communicatieopdracht.
10,n	Schrijft (A) op adres n en maakt A schoon.
11,n	Schrijft (S) op adres n en maakt A schoon.
12,n	Deelt (A) + p×(S) door (n), plaatst rest in A en quotient in S.
13,n	Deelt (A) door (n), plaatst afgerond quotient in S en maakt A schoon.
14,n	Vermenigvuldigt (A) met 2^{-n}.
15,n	Verplaatst besturing naar n als (A)<0.

10. Geblokkeerde geheugeninhoud.

Adres:	Octaal:	Geheel getal:	Breuk:	Op-dracht	Code-woord
0	00000 00017	15	+ 0,0000 00028	0/15	GGF
1	02000 50103	335 74979	+ 0,0625 38272	10/67	GGI
2	42002 00020	- 5032 50927	- 0,9373 77898	0/16	B 0.8
3	77777 50105	- 12218	- 0,0000 22758	10/69	T 1
4	42004 50106	- 5031 64857	- 0,9372 17580	10/70	B 0.7
5	77777 50100	- 12223	- 0,0000 22767	10/64	T 0
6	77777 44061	- 14286	- 0,0000 26610	9/49	T .
7	42010 50101	- 5030 33790	- 0,9369 73449	10/65	B 0.6
8	42020 44061	- 5027 73710	- 0,9364 89012	9/49	B 0.5
9	42040 40000	- 5022 51519	- 0,9355 16355	8/0	B 0.4
10	77777 74022	- 2029	- 0,0000 03779	15/18	T 4
11	42100 04060	- 5012 17231	- 0,9335 89844	1/48	B 0.3
12	42200 20100	- 4991 13919	- 0,9296 72120	4/64	B 0.2
13	41002 54100	- 5200 05567	- 0,9685 85847	11/64	B 1.7
14	77777 34010	- 18423	- 0,0000 34316	7/8	T 2
15	77777 34025	- 18410	- 0,0000 34291	7/21	T 5
16	77777 34053	- 18388	- 0,0000 34250	7/43	T +
17	41004 00124	- 5199 62539	- 0,9685 05701	0/84	B 1.6
18	41010 00021	- 5198 31534	- 0,9682 61685	0/17	B 1.5
19	41020 50102	- 5195 48861	- 0,9677 35166	10/66	B 1.4
20	41040 34102	- 5190 30717	- 0,9667 70047	7/66	B 1.3
21	41100 00100	- 5179 96479	- 0,9648 43629	0/64	B 1.2
22	41200 50100	- 5158 78847	- 0,9608 99232	10/64	B 1.1
23	40402 00101	- 5284 16702	- 0,9842 52807	0/65	B 2.6
24	40404 40000	- 5283 34847	- 0,9841 00340	8/0	B 2.5
25	77777 74035	- 2018	- 0,0000 03759	15/29	T 7
26	40410 30013	- 5282 07860	- 0,9838 63808	6/11	B 2.4
27	40420 00100	- 5279 57951	- 0,9833 98316	0/64	B 2.3
28	40440 34104	- 5274 19323	- 0,9823 95043	7/68	B 2.2
29	40500 00020	- 5263 85135	- 0,9804 68718	0/16	B 2.1
30	40600 50101	- 5242 67454	- 0,9765 24230	10/65	B 2.0
31	40202 34101	- 5325 96670	- 0,9920 38604	7/65	B 3.5
32	40204 40100	- 5326 60159	- 0,9921 56861	8/64	B 3.4
33	40210 50100	- 5323 93919	- 0,9916 60951	10/64	B 3.3
34	40220 00100	- 5321 52255	- 0,9912 10816	0/64	B 3.2
35	40240 64061	- 5316 01358	- 0,9901 84691	13/49	B 3.1
36	40300 54100	- 5305 56863	- 0,9882 39167	11/64	B 3.0
37	40102 40100	- 5346 91775	- 0,9959 41041	/64	B 4.4
38	40104 34104	- 5346 28283	- 0,9958 22778	7/68	B 4.3

Adres:	Octaal:	Geheel getal:	Breuk:	Opdracht:	Code-woord:
39	40110 74104	- 5344 80827	- 0,9955 48120	15/68	B 4.2
40	40120 00100	- 5342 49407	- 0,9951 17066	0/64	B 4.1
41	40140 00062	- 5337 25133	- 0,9941 40530	0/50	B 4.0
42	77777 74055	- 2002	- 0,0000 03729	15/45	T -
43	40042 00104	- 5357 56731	- 0,9979 24676	0/68	B 5.3
44	40044 00057	- 5356 91216	- 0,9978 02645	0/47	B 5.2
45	40050 50104	- 5355 39647	- 0,9975 20326	10/68	B 5.1
46	40060 34005	- 5352 83706	- 0,9970 43598	7/5	B 5.0
47	00000 00001	1	0,0000 00002	(0/1)	(GGF)
48	00000 00012	10	0,0000 00019	(0/10)	(GGF)
49	05753 60400	10^8	0,1862 64515	(12/256)	LB
50	40022 50000	- 5362 60607	- 0,9988 63218	10/0	B 6.2
51	40024				B 6.1
52	40030				B 6.0
53	40012				B 7.1
54	40014				B 7.0
55	00000 40000	16384	0,0000 30518	8/0	(GGF)
56	37777 77777	5368 70911	0,9999 99998		PB
57	77656 05114	- 26 84339	- 4,9999 71017 $x10^{-3}$		T 3
58	77767 63533	- 2 68452	- 5,0003 08156 $x10^{-4}$		T 6
59	77777 13443	- 26844	- 5,0000 84638 $x10^{-5}$		T 8
60	77777 72573	- 2692	- 5.0142 40742 $x10^{-6}$		Tab
61	77777 77364	- 267	- 4,9732 62548 $x10^{-7}$		TW
62	77777 77744	- 27	- 5,0291 41903 $x10^{-8}$		T 9
63	77777 77774	- 3	- 5.5879 35448 $x10^{-9}$		Spa

Controle door herhaling subroutine.

Doel : Een berekening wordt herhaald en aan de hand van de gelijkheid of ongelijkheid van een bepaald resultaat x in beide gevallen wordt de foutenvrijheid gecontroleerd. Als de twee resultaten gelijk zijn wordt overgegaan tot het volgende programma, zo niet dan wordt de berekening zo vaak herhaald tot twee maal achter elkaar hetzelfde resultaat is gevonden.

Details: De subroutine kan willekeurig vaak gebruikt worden. Als de subroutine gebruikt wordt om verschillende deelberekeningen van een groot programma te controleren, dan mag één dier deelberekeningen niet bevat zijn in een andere. Zij kan dus slechts op deelberekeningen in serie worden toegepast.
De te controleren berekening moet het resultaat x achterlaten in S en A schoon achterlaten. De subroutine laat x ook in S, en A schoon achter.
De berekening begint op adres b. De subroutine begint op adres d en omvat 26 adressen. In het normale geval van gelijke resultaten kost zij inclusief aanroepen 2 + 10 + 2 + 9 = 23 korte opdrachten plus natuurlijk een extra berekening.

Aanroep:

	c	0/c
	c+1	7/d ⇒ naar subroutine
	c+2	7/b
van subr →	c+3	....

Voorponsing:

	A	71	70
	+	d	70

Subroutine:

		A		71	(A)=0/c+2 (S)=x
c+1 →	0'	15	19	71	wordt 7/10' door 5'
	1'	2	7	71	(7')=0/c+2
	2'	0	24	71	(A)=7/c+3
	3'	10	22	71	(A)=0 (22')=7/c+3
	4'	0	25	71	(A)=7/10'
	5'	10	0	71	(A)=0 (0')=7/10'
	6'	11	23	71	(23')=x_1
	7'	(10		70)	wordt 0/c door 1' (A)=7/b
	8'	10	9	71	(A)=0 (9')+7/b

15'→ 9'	(10		70)	wordt 7/b⇒door 8'	Herhaal berekening
0'→10'	3	64	70		(A)=x_2
11'	8	23	71		(A)=x_2-x_1
12'	7	16	71	→	Halfgoed
19'→13	0	23	71		(A)=$x_2 \neq x_1$
14'	10	23	71		(A)=0 (23')=x_2
15'	7	9	71	⇒	Fout
12'→16'	8	47	70		(A)=x_2-x_1-p
17'	15	20	71	→	Goed
18'	0	47	70		(A)=x_2-x_1>0
19'	7	13	71	⇒	Fout
17'→20'	0	17	71		(A)= 15/19'
21'	10	0	71		(A)=0 (0')=15/19'
22'	(10	0	70)	wordt 7/c+3⇒door 3'	Volgende berekening
23'	(10	0	70)	wordt x_1	
24'	7	3	70	constanten.	
25'	7	10	71		

Controle door uittypen van het gevolgde programma. (Kort,langzaam).

Doel: Dit is een routine, geen subroutine. Het te bestuderen programma komt zelf helemaal niet aan bod. De routine leest een opdracht van het te controleren programma, typt het functiegedeelte als getal, dus 0 of 1 of 2... of 15 met een streepje (-teken) erachter ter betere onderscheiding en voert vervolgens de opdracht op dezelfde wijze uit als het programma het zelf zou hebben gedaan. Daarna leest de routine de volgende opdracht van het programma, enz. Is de opdracht een besturingsverplaatsing, waaraan gehoorzaamd wordt, dus 7/x met $(A) \geq 0$ of 15/x met $(A) < 0$, dan wordt i.p.v. het streepje een nieuwe regel getypt. Op deze wijze is enerzijds tegemoet gekomen aan de eindige wagenbreedte van de schrijfmachine, maar bovendien kan men zien of aan een besturingsverplaatsing gehoorzaamd is of niet.

Details: Twee geheugenplaatsen, nl. 41' resp. 42' dienen als vervangers van de registers A resp. S, die natuurlijk voor de routine zelf gebruikt moeten worden. Erin worden opgeborgen de inhouden van A resp. S, zoals die zouden zijn na het uitvoeren van de betreffende opdracht door het programma zelf.
Op de band wordt eerst het gehele te controleren programma ingevoerd, maar in plaats van nu de besturing te verwijzen naar het begin ervan, wordt erna de controleroutine ingevoerd en hiernaar wordt de besturing verwezen. Er is dus geen aanroep. Het te controleren programma beginne op adres n, de controleroutine op adres d. De band eindigt dus met: A, 69, 70; 7,, 71.
Het gebruik van de routine betekent een geweldige vermindering van het arbeidstempo van de machine, omdat i.p.v. 1 opdracht F/x, 24 opdrachten als $0 \leq F \leq 7$, 28 opdrachten als $8 \leq F \leq 14$ en 25 opdrachten als F = 15 uitgevoerd moeten worden.
Het uittypen van een opdracht kost 2 of 3 plaatsen op de schrijfmachine. Er zijn er op één regel 180, maar behalve de opdrachtnummers zelf, typt de routine ook uit, wat het programma zelf zou hebben uitgetypt, dus er moet niet al te zelden een gehoorzaamde opdracht 7 of 15 voorkomen. Dit zal echter doorgaans zeker wel het geval zijn.

De opdrachten van het te controleren programma mogen geen kunstopdrachten zijn zoals ze in het invoerprogramma voorkomen, maar normale opdrachten, d.w.z. positieve getallen kleiner dan 2^{15}p. De routine neemt 46 adressen in beslag.

Voorponsing:

A	71	70	
+	d	70	(71)= d
+	n	70	(72)= n.

Routine:

	A		71			
40 →0	0		72			(A)= F/x
1	2	33	71			(33')= F/x
2	14	9	70			(A)=(4F+0 à 3)p
3	8	43	71			(A)={4(F-10)+0 à 3}p
4	7	7	71	→	als F≥10	
5	0	43	71			(A)=(4 F+0 à 3)p
6	7	8	71	⟹		
4 →7	9	3	70		typt 1	
6 →8	0	5	70			(A)=code voor te typen cijfer
9	10	64	70			(A)=0 (64)=code
10	9	64	70		typt eenheden van F.	
11	0	33	71			(A)= F/x
12	8	44	71			(A)= F-7/x
13	15	30	71	→	als 0≤F≤6	
14	8	45	71			(A)= F - 8/x
15	15	25	71	→	als F = 7	
16	8	44	71			(A)= F-15/x
17	15	30	71	→	als 8≤F≤14	
18	10	33	71		F = 15	(A) = 0 (33')= 0/x
19	0	41	71			(A)= (Acc)
20	7	28	71	→	als F=15,A>0	
27 →21	10	64	70		Verplaatsing geaccepteerd	(A)=0
22	9	61	70		typt NR	
23	0	33	71			(A)= 0/x
24	7	39	71	⟹		
15 →25	10	33	71			(A)=0 (33')= 0/x
26	0	41	71			(A)= (Acc)
27	7	21	71	→	als F=7,A>0	
20 →28	10	64	70		Verplaatsing geweigerd	(A)= 0
29	7	37	71	⟹		
13',17' →30	10	64	70			(A)= 0

	31'	0	41	71		(A)=(41')
	32'	1	42	71		(S)=(42')
	33'	(10	0	70)	wordt F/x	
	34'	10	41	71		(A)=0 (41')=(A)
	35'	11	42	71		(42')=(S)
	36'	9	42	70	typt -	
29' →	37'	0	0	71		(A)= 0/n
	38'	0	47	70		(A)= 0/n+1
24' →	39'	10	0	71		(A)=0 (0)=0/n+1 of 0/x
	40'	7	0	71 ⇒		
	41'	0	0	70	wordt (A)	
	42'	0	0	70	wordt (S)	
	43'	+	40	70	} con-	
	44'	7	0	70	} stan-	
	45'	1	0	70	} ten.	

Controle door uittypen van het gevolgde programma (Lang,snel).

Doel: Dit is een variant van de voorgaande routine. De routine beslaat meer geheugenplaatsen maar is sneller. De routine typt voor de opdrachten 0,1,...,6 en 8 en 9 de corresponderende cijfers. Voor opdracht 10 een +, voor 11 een -, voor 12 een · (punt), voor 13 een 7 (!), voor 14 een spatie, voor 7 een 07 en voor 15 een 15, terwijl in geval van gehoorzaamde besturingsverspringing (dus 07 met A$\geqslant$0 of 15 met A$<$0) bovendien een nieuwe regel gegeven wordt. Tussen de opdrachtnummers wordt een · (punt) getypt.

Details: De twee geheugenplaatsen 65' en 66' dienen als vervangers van de registers A en S. De lengte van het programma is totaal 67 adressen. In plaats van één opdracht van het te controleren programma komen nu 19 opdrachten, behalve bij opdracht 7 met A$\geqslant$0, waar het er 20 zijn en bij opdracht 15, waar het er 18 zijn.
Voor de rest zie beschrijving van de voorgaande routine.

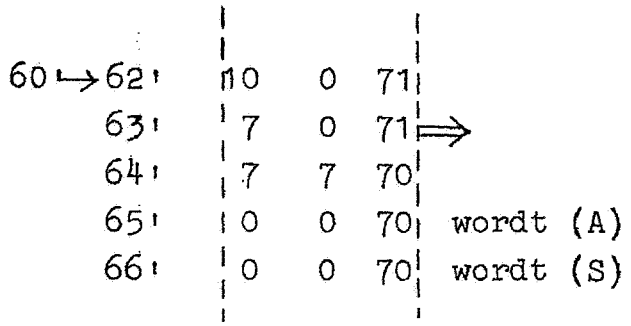
Voorponsing:

	A	71	70	
	+	d	70	(71)= d
	+	n	70	(72)= n

Routine:

		A		71	
63'→	0'	0		72	(A)= F/x
	1'	2	45	71	(45')= F/x
	2'	14	11	70	(A)= 0/F
	3'	6	1	70	(A)= 0/2F
	4'	0	64	71	(A)= 7/7+2F
	5'	10	6	71	(A)= 0 (6')=7/7+2F
	6'	10	0	70	wordt 7/7+2F ⟹
F=0→	7'	9	5	70	typt 0
	8'	7	43	71	⟹
F=1→	9'	9	3	70	typt 1
	10'	7	43	71	⟹
F=2→	11'	9	14	70	typt 2
	12'	7	43	71	⟹
F=3→	13'	9	57	70	typt 3
	14'	7	43	71	⟹
F=4→	15'	9	10	70	typt 4
	16'	7	43	71	⟹
F=5→	17'	9	15	70	typt 5
	18'	7	43	71	⟹
F=6→	19'	9	58	70	typt 6

```
        20'|  7  43  71 ⟹
F=7 ⟶   21'|  9   5  70 | typt 0
        22'|  7  53  71 ⟹
F=8 ⟶   23'|  9  59  70 | typt 8
        24'|  7  43  71 ⟹
F=9 ⟶   25'|  9  62  70 | typt 9
        26'|  7  43  71 ⟹
F=10⟶   27'|  9  16  70 | typt +
        28'|  7  43  71 ⟹
F=11⟶   29'|  9  42  70 | typt -
        30'|  7  43  71 ⟹
F=12⟶   31'|  9   6  70 | typt .
        32'|  7  43  71 ⟹
F=13⟶   33'|  9  25  70 | typt 7
        34'|  7  43  71 ⟹
F=14⟶   35'|  9  63  70 | typt spatie
        36'|  7  43  71 ⟹
F=15⟶   37'|  9   3  70 | typt 1
        38'|  9  15  70 | typt 5
        39'|  0  65  71 |                    (A)= (Acc)
        40'| 15  56  71 ⟶geaccepteerd
55'⟶    41'| 10  64  70 | geweigerd          (A)= 0
        42'|  7  48  71 ⟹
....⟶   43'|  0  65  71 |                    (A)= (Acc)
        44'|  1  66  71 |                    (S)=(SR)
        45'| (10   0  70) wordt F/x
        46'| 10  65  71 |                    (A)=0   (65')=(A)
        47'| 11  66  71 |                            (66')=(S)
42'⟶    48'|  0   0  71 |                    (A)= 0/n
        49'|  0  47  70 |                    (A)= 0/n+1
        50'| 10   0  71 |                            (0')=0/n+1
        51'|  9   6  70 | typt .
        52'|  7   0  71 ⟹
22'⟶    53'|  9  25  70 | typt 7
        54'|  0  65  71 |                    (A)=(Acc)
        55'| 15  41  71 ⟶geweigerd
40'⟶    56'|  9  61  70 | geaccepteerd typt
                                 NR
        57'| 10  64  70 |                    (A)= 0
        58'|  0  45  71 |                    (A)= F/x
        59'|  8  ..  71 |                    (A)= F-15/x
        60'|  7  62  71 ⟶als F=15
        61'|  0  55  71 |                    (A)=F-7/x= 0/x
```



(A)=0 (0')=0/x