MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM

REKENAFDELING

Programmering voor de ARMAC

Deel I a

Supplement op Deel I

door

E.W. Dijkstra

MR 25 a

1957

The Mathematical Centre at Amsterdam, founded the 11th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications, and is sponsored by the Netherlands Government through the Netherlands Organization for Pure Research (Z.W.O.) and the Central National Council for Applied Scientific Research in the Netherlands (T.N.O.), by the Municipality of Amsterdam and by several industries.

Inhoud.

	pag.
Voorwoord	1
Het bedieningspaneel van de ARMAC	2
De kwarttracktransporten.	12
Het invoerprogramma	13
Herstartbaarheid.	16
Programma in het snelle kanaal.	21
Voorbeeld: Subroutine matrix maal vector. . .	22

Voorwoord

Nu het rapport MR 25, met welks inhoud de lezer verondersteld wordt op de hoogte te zijn, een jaar in gebruik is, zijn de tekortkomingen ervan voldoende duidelijk aan het licht gekomen, om een supplement te rechtvaardigen.

Hierin is ook opgenomen de beschrijving van de opdrachten - de kwarttracktransporten n.l. - die sinds het verschijnen van MR 25 aan de code van de ARMAC zijn toegevoegd.

De oorspronkelijke opzet van MR 25 blijft ongewijzigd. Hoewel dit supplement met een voorbeeld wordt afgesloten, blijft MR 25 meer "handboek" dan "leerboek".

Het bedieningspaneel van de ARMAC

De inhoud van vijf registers is permanent op lampjes van het bedieningspaneel van de ARMAC zichtbaar.

De opdrachtteller (zie MR 25, blz. 15)

Onder controle van de inhoud van de opdrachtteller worden door de opdrachtselectie de opdrachten stuk voor stuk uit het geheugen aangehaald. De opdrachtteller geeft steeds de (halve) plaats aan, waar de eerstvolgende opdracht vandaan gehaald moet worden; na het aanhalen van deze opdracht wordt tijdens de uitvoer ervan de inhoud van de opdrachtteller met $\frac{1}{2}$ verhoogd.

De inhoud van de opdrachtteller is zichtbaar op een rij van dertien lampjes, links bovenaan op het centrale gedeelte van het bedieningspaneel, n.l. van links naar rechts:

1. de zeven meest significante binalen van het adres, die tesamen het kanaal specificeren.
2. de vijf minst significante binalen van het adres, die tesamen de plaats in het kanaal specificeren.
3. het lampje, dat aangeeft of de a- dan wel de b-opdracht uit dit adres aan de beurt is. (Het lampje is uit - "de a-b-binaal is = 0" - voor de a-opdracht, het brandt - "de a-b-binaal is = 1" - voor de b-opdracht; in deze cijferplaats wordt de opdrachtteller opgehoogd!)

Vlak boven de linkse zeven lampjes van de opdrachtteller bevindt zich nog een zevental, dat het nummer aangeeft van het kanaal, dat zich op dat moment in de buffer bevindt. Door het vergelijken van deze twee zeventallen wordt het automatisch inlezen van de buffer bestuurd.

Het opdrachtregister

De onder controle van de inhoud van de opdrachtteller uit het geheugen aangehaalde opdracht wordt opgevangen in het opdrachtregister, waarvan de inhoud zichtbaar is op een rij van zeventien lampjes, rechts bovenaan op het centrale gedeelte van het bedieningspaneel. (Links van deze lampjes bevindt zich nog een extra lampje voor de "parity-digit": van de achttien lampjes moet er een oneven aantal branden.)

In overeenstemming met de wijze, waarop opdrachten in het geheugen opgeborgen worden, komen de vijf meest linkse lampjes overeen met de functiecijfers van de opdracht, de resterende twaalf - weer gesplitst in zeven plus vijf - met het adresgedeelte van de opdracht.

Opm. 1: Als de machine gestopt is, bevat de opdrachtteller de plaats van de volgende opdracht, die aan de beurt is; deze moet nog aangehaald worden en het opdrachtregister bevat dan ook nog de laatst uitgevoerde opdracht.

2: Bij de opdrachten, die langer dan één getaltijd (= 0.417 msec) duren (d.w.z. vermenigvuldigingen en tracktransporten) worden tijdens de uitvoering van de opdracht de laagste vijf cijfers van het opdrachtregister gewijzigd.

3: Boven de rechterkant van het opdrachtregister bevindt zich nog een rij van negen lampjes. Deze vermelding is volledigheidshalve: de interpretatie van het al of niet branden van deze lampjes is n.l. een privilege van het technisch personeel.

Het M-register

Alle transport van informatie van of naar het geheugen gaat via het M-register, waarvan de inhoud zichtbaar is op een rij van vierendertig lampjes onderaan (de verticale plaat van) het centrale gedeelte van het bedieningspaneel. Links van deze rij bevinden zich nog twee extra lampjes voor de parity digits (het linkse vult het aantal enen in het linker zeventiental aan tot een oneven totaal, het rechter lampje evenzo het rechter zeventiental.)

Na de uitvoering van een schrijfopdracht bevat M het weggeschreven getal, na een lezende opdracht het gelezene. In het bijzonder wordt tijdens een vermenigvuldiging de uit het geheugen aangehaalde factor in het M-register bewaard en is deze na afloop nog in het M-register zichtbaar.

Het A- en het S-register

Geheel rechts bevinden zich twee rijen van vierendertig lampjes, waarop ten alle tijde de momentane inhoud van de registers A en S (bovenste, resp. onderste rij) zichtbaar is. Deze registers

hebben geen parity digits. Aan de minst significante kant zit voor beide registers nog een extra lampje, dat de end-around-carry zichtbaar maakt.

Als de ARMAC werkt, wisselt de inhoud van de boven opgenoemde registers veel te snel, dan dat men er iets aan zien kan. Een uitzondering moet gemaakt worden voor de meest significante kant van de opdrachtteller. Deze bevat het kanaalnummer, dat vaak lang genoeg constant is, om herkend te worden: in dat geval ziet men, aan welk stuk van het programma de machine bezig is. In exceptionele gevallen kan men aan de "globale" inhoud van het A- of S-register een punt van de berekening herkennen, of althans het verloop volgen.

Behalve deze vijf rijen lampjes bevinden zich op het bedieningspaneel van de ARMAC vijf rijen schakelaars: een schakelaar naar boven stelt een één, naar beneden een nul voor.

De getalschakelaars

Een volledig woord van vierendertig binaire cijfers kan opgezet worden op de zg. getalschakelaars: deze vierendertig schakelaars bevinden zich vlak onder de rij lampjes van het M-register. Omdat de getalschakelaars veel gebruikt worden voor het "met de hand" - d.w.z. niet via de ponsband - inbrengen van opdrachten-(paren) zijn deze schakelaars verdeeld in 2 maal 5 + 7 + 5. Om met de hand de inhoud van de getalschakelaars in S te krijgen kan men gebruik maken van "autostart 7". Om in een programma de stand der getalschakelaars te lezen, maakt men gebruik van de communicatieopdrachten, beschreven in MR 25, blz. 25 en 26.

Het beginadres

Boven de dertien lampjes van de opdrachtteller bevinden zich dertien schakelaars met het opschrift "Beginadres". Op deze schakelaars is dezelfde indeling van toepassing als op de lampjes, ook zij hebben de functie, de plaats van een opdracht - de z.g. "gekozen opdracht" - vast te leggen.

Vier druktoetsen van het toetsenpaneel hebben op het beginadres betrekking. Het zijn:

1. "Begin Gekozen Opdracht". De machine neemt de inhoud van

de schakelaars van het beginadres over in de opdrachtteller en gaat werken (begint met een opdracht aan te halen etc.)
Dit is de normale starttoets; de schakelaars van het beginadres stellen ons dus in staat de machine op een willekeurig punt met de uitvoering van programma te laten beginnen.

2. "Doe Gekozen Opdracht". De machine begint als onder 1, doch stopt na de uitvoering van de eerste opdracht (inclusief ophoging van de opdrachtteller).

3. "Lees Gekozen Opdracht". De machine begint als onder 1 en 2, maar stopt nog eerder, n.l. na het inlezen van de opdracht in het opdrachtregister en het ophogen van de opdrachtteller, doch vóór de uitvoering van de opdracht.

4. "Lees Gekozen Getal". Hier doet de stand van de a-b-schakelaar niet ter zake, het woord op het adres, dat op de overige twaalf schakelaars is opgezet, verschijnt in M. Deze faciliteit wordt gebruikt als men tijdens onderbreking van de berekening de inhoud van een of meer geheugenplaatsen wil inspecteren. De belangrijkste eigenschap van "Lees Gekozen Getal" is, dat noch de inhoud van A- of S-register, noch van de opdrachtteller, noch de conditie erdoor gewijzigd wordt.

Vlak boven deze druktoetsen bevinden zich vier toetsen met analoge functie: het overnemen van de inhoud van de schakelaars van het beginadres blijft alleen achterwege, zodat zij opereren onder controle van de momentele inhoud van de opdrachtteller. Het zijn

1. "Begin Volgende Opdracht". De machine begint te werken bij de opdracht, waarvan de plaats in de opdrachtteller is aangegeven. Aangezien de opdrachtteller al is opgehoogd, als de machine na de uitvoering van een opdracht is gestopt, zet men op deze manier correct de onderbroken berekening voort. Het drukken op deze toets heet "doorstarten".

2. "Doe Volgende Opdracht". De machine begint als onder 1, doch stopt na de uitvoering van de eerste opdracht (inclusief de ophoging van de opdrachtteller). Deze toets wordt gebruikt om tijdens het testen een stuk programma opdracht voor opdracht na te lopen. Na het beginadres te hebben ingevuld, drukt men een maal op "Doe Gekozen Opdracht", vervolgens steeds op "Doe Volgende Opdracht". Steeds kan men in het opdrachtregister in-

specteren of de bedoelde opdracht inderdaad is uitgevoerd, en in de registers, of het resultaat van de bewerking er vertrouwenwekkend uitziet.

3. "Lees Volgende Opdracht". De machine begint als onder 1 en 2, doch stopt nog eerder, n.l. na het inlezen van de opdracht in het opdrachtregister en de ophoging van de opdrachtteller, doch vóór de uitvoering van de opdracht. Deze toets - eventueel na één maal "Lees Gekozen Opdracht" - wordt gebruikt, om visueel te inspecteren of een stukje programma goed in het geheugen staat.

4. "Lees Volgend Getal". In M verschijnt het getal dat staat op het adres, dat door de twaalf hoogste binalen van de opdrachtteller wordt gespecificeerd. Aangezien de opdrachttellerinhoud hierbij onveranderd blijft, en het dus niet mogelijk is, met enkel deze toets een rijtje getallen in het geheugen te inspecteren, wordt hij zo goed als nooit gebruikt.

In de rij van de toetsen voor de "volgende" opdracht bevindt zich tenslotte de stopknop. Drukt men deze in, dan voltooit de machine nog de laatst aangehaalde opdracht. (De stopknop is bij werkende machine de enige niet geblokkeerde druktoets. Alle andere toetsen zijn geblokkeerd, d.w.z. men kan ze wel indrukken, maar het heeft geen effect.) Hieronder bevindt zich in de rij voor de "gekozen" opdracht de z.g. stopschakelaar. Staat deze naar de operateur toe, dan heeft hij geen effect, staat hij van de operateur af, dan stopt de machine steeds na uitvoering van de opdracht waarvan de plaats is aangegeven in het stopadres (zie onder).

Het Stopadres

Onder de dertien lampjes van de opdrachtteller bevinden zich in dezelfde indeling dertien schakelaars met het onderschrift "Stopadres". Als de stopschakelaar (zie boven) naar de operateur toe staat, doet de stand van de stopadresschakelaars niet ter zake. Staat de stopschakelaar echter van de operateur af, dan wordt steeds de inhoud van de opdrachtteller met het stopadres vergeleken: in geval van gelijkheid stopt de machine na de uitvoering van de betroffen opdracht. De inhoud van de opdrachtteller is dan dus "$\frac{1}{2}$ groter" dan de inhoud van het stopadres,

mits de laatste opdracht geen gehoorzaamde sprongopdracht was.

De opdrachtschakelaars

Boven het opdrachtregister bevinden zich zeventien schakelaars in dezelfde indeling als de lampjes eronder. De opdracht kan in het opdrachtregister worden overgenomen door te drukken op de toets "Schrijf" in de onderste van de drie rijen druktoetsen. De toets "Doe" voert de opdracht in het opdrachtregister uit. Deze knoppen stellen ons in staat een opdracht uit te voeren, die niet in het geheugen staat. Dit is van vrij essentieel belang als mogelijkheid iets in de machine te krijgen, als het invoerprogramma nog niet in het geheugen staat - een toestand, die zich voordoet als b.v. de trommel voor onderhoud uit de lagers is genomen.
De programmeur gebruikt deze toetsen hoofdzakelijk om tijdens het testen een wijziging in zijn programma zonder band even aan te brengen. De nieuwe opdrachten kan hij op de getalschakelaars zetten, met de "autostart 7" kan hij dit woord in S krijgen. Als in de opdrachtschakelaars een 12-opdracht met het adres van de te wijzigen plaats staat, kan hij vervolgens door op de knoppen "Schrijf" en "Doe" te drukken het nieuwe opdrachtenpaar in het geheugen schrijven.
Sommige programmeurs en operateurs prefereren, om bij correctie van programma niet altijd het nieuwe opdrachtenpaar in de getalschakelaarste zetten, maar als dat makkelijker uitkomt het verschil tussen de oude en de nieuwe. Met de "autostart 7" plaatst men dit verschil in S Met een 8-opdracht en daarna een 12-opdracht (of 9- gevolgd door 13-opdracht), beide met het adres van de te wijzigen plaats, corrigeren zij het programma.
Dan moeten ze twee maal op de toetsen "Schrijf" en "Doe" drukken; daartegenover staat, dat vaak het verschil veel sneller in de getalschakelaarswordt gezet.
Als derde druktoets in de onderste rij van het toetsenbord bevindt zich de druktoets "Herhaal". Deze combineert de operaties "Schrijf Doe" en gaat hiermee tot nader order door, d.w.z. tot het indrukken van de stopknop; dit ten behoeve van het technisch personeel.

De "Stop op opdracht"

De bovenbeschreven "stop-op-adres"-schakelaar bepaalde, of men de machine op een bepaald punt in het programma wenste te stoppen. Het is eveneens mogelijk de machine te stoppen op een bepaald type opdracht. Dit wordt beheerst door de "stop-op-opdracht"-schakelaar, die zich vlak onder de bovengenoemde "Stop-op-adres"-schakelaar bevindt; weer wordt deze stopmogelijkheid buiten werking gezet, door de schakelaar naar de operateur toe te zetten. Het type opdracht wordt in zeventien schakelaars opgezet, die zich gelijkelijk ingedeeld, onder de lampjes van het opdrachtregister bevinden. Dit zijn driestanden schakelaars: naar boven betekent één, naar beneden nul, en in de middenstand betekent "indifferent". Als nu de "stop-op-opdracht"-schakelaar van de operateur af staat, wordt elke opdracht in het opdrachtregister vergeleken met de overeenkomstige schakelaar: zolang met minstens een van de in een uiterste stand staande schakelaar verschil geconstateerd wordt, gaat de machine gewoon door, anders stopt de machine.
(Als alle zeventien schakelaars van de stopopdracht op indifferent staan, en de "stop-op-opdracht" staat in, dan stopt de machine dus na elke opdracht: begin gekozen resp. volgende opdracht is dan identiek aan doe gekozen resp. volgende opdracht.)
Men kan op deze wijze de ARMAC b.v. stoppen op elke opdracht, die naar een bepaald adres refereert: in het adresgedeelte zet men alle schakelaars op nul of een, de vijf schakelaars voor de functie zet men indifferent. Men kan de machine ook stoppen b.v. op alle schrijfopdrachten op een bepaald adres: dan zet men in de vijf functieschakelaars 0 X 1 0 X (met X indifferent aanduidend). Dit is vooral van belang wanneer door een fout in het programma een of ander adres abusievelijk wordt overschreven: men vindt zo onmiddellijk de boosdoener. (Dit is een van de belangrijkste toepassingen van deze stopfaciliteit).
Een achttiende schakelaar stelt de operateur in de gelegenheid deze stop op te heffen voor het geval van een communicatieopdracht, iets wat, omdat de adrescijfers hier heel andere betekenis hebben, vaak gewenst is.

De trommel-buffer-schakelaar

Normaal staat de z.g. trommel-buffer-schakelaar altijd op "buffer"; als hij op "trommel" staat, heeft dit tot gevolg dat bij starten altijd het trommelkanaal eerst in de buffer gelezen wordt, ook als het betroffen kanaal al in de buffer staat. (Starten is hier in de ruimste zin gebruikt: het betreft "Begin, Doe of Lees" en zowel "Gekozen" als "Volgende Opdracht".)

De schakelaar staat normaal op "buffer": dan is het mogelijk om zonder schadelijke gevolgen de machine te stoppen, eventueel eerst een paar opdrachten stuk voor stuk te doen en dan de machine weer door te starten.

Als tijdens een berekening de machine door het falen van de parity-check stopt, doet zich - als het een goed opgezet programma is - vaak de situatie voor, dat men een eindje terug beginnen kan. Dikwijls is "dit eindje terug" nog in hetzelfde kanaal. Tengevolge van bufferschrijfopdrachten na dit punt kan de buffer heel best geen getrouwe copie meer zijn van het betroffen trommelkanaal. In dat geval moet men de machine op dat bewuste punt starten met de schakelaar op "trommel". Onmiddellijk daarna de schakelaar weer terug zetten op "buffer"!

Dezelfde situatie kan zich voordoen bij het begin van een programma, als het startadres zich in een kanaal bevindt, waarin onmiddellijk na het starten - d.w.z. voordat andere kanalen in de buffer zijn gelezen - door bufferschrijfopdrachten de bufferinhoud van die van het trommelkanaal gaat afwijken. Een dergelijk programma kan slechts herstart worden, met de bijconditie dat de trommel-buffer-schakelaar op "trommel" staat. Omdat dit vraagt om bedieningsfouten, dient men een dergelijke opzette vermijden: men maakt het startadres in een ander trommelkanaal en springt onmiddellijk naar het bedoelde startadres en de onsierlijkheid is verwijderd.

Als de machine de buffer met de inhoud van een nieuw trommelkanaal vult, wordt deze informatie-transport aan de parity-check onderworpen: als hier een één teveel of te weinig doorkomt, stopt de machine. In dit geval kan men, met de schakelaar op "trommel" de machine doorstarten: de machine probeert

het dan opnieuw. Vaak lukt het dan wel en de berekening kan met verlies van enkele seconden voortgezet worden. Pas als een bepaald trommelkanaal bij herhaling moeite heeft, zijn inhoud correct aan de buffer af te leveren, gaat men tot forsere maatregelen over.

Nog enkele losse controlelampjes bevinden zich op het bedieningspaneel.

Als de machine door een parity-check gestopt is, brandt minstens één van de lichtjes "Fout Getal" of "Foute Opdracht", indicerend bij welk transport de check niet klopte. Het falen van de parity-check bij het inlezen van de buffer geldt als fout getal. Als de machine een foute opdracht aanhaalt of een fout getal, wordt deze onjuiste operatie wel uitgevoerd, daarna pas stopt de machine. Dit is een inconvenient. De overwegingen waren, dat men van dit inconvenient slechts weinig last mag hebben, omdat de parity-check hoort te kloppen, terwijl anderzijds elk ander arrangement de ARMAC aanzienlijk vertraagd zou hebben: het aanleggen van de parity-check is een langdurige bezigheid, op de voltooiing waarvan met de uitvoering van de opdracht niet gewacht wordt. Men had dit offer aan snelheid wel moeten brengen, als men van zins was een onbetrouwbare machine te construeren! Het aan of uit zijn van deze lampjes heeft alleen zin, als de machine gestopt is. Dat tijdens het werken deze lichtjes opflikkeren is niet verontrustend.

Hieronder bevindt zich het conditielichtje: is dit uit, dan is de conditie positief, brandt het, dan is de conditie negatief.

Hieronder bevinden zich nog twee grotere lampjes: een groen, dat brandt, zolang de machine werkt, en een rood, dat brandt als de machine gestopt is.

Behalve deze uitgebreide manieren, waarop men visueel informatie omtrent het verloop van de berekening krijgt, kan men ook auditief omtrent het doen en laten van de ARMAC ingelicht worden.

Een luidspreker is aangesloten op het tekencijfer van het A-register. Wisselingen van de inhoud van deze cijferplaats

worden als "schokje" naar de luidspreker gezonden. Vele programma's geven als geluid een ondetermineerbaar geruis. Soms echter klinken de verschillende fasen van de berekening duidelijk verschillend, en kan men aan het geluid horen, als de berekening hapert (b.v. in een cyclusje blijft hangen).

Een zoemer kan men laten zoemen, als het rode licht brandt, d.w.z. als de ARMAC gestopt is.

Beide indicatie-mogelijkheden zijn van belang, vooral als men niet achter het bedieningspaneel wil blijven zitten, of als het oog het flikkeren der lampjes moe is. Luidspreker en zoemer zijn echter beide nogal luid en met twee schakelaars kan men gelukkig elk het zwijgen opleggen.

De kwarttracktransporten.

Nadat de beslissing gevallen was, dat het snelle geheugen van de ARMAC beperkt zou blijven tot één kanaal, zijn de z.g. kwarttracktransporten ingebouwd, die de programmeur in staat stellen met grotere flexibiliteit informatie van en naar kanaal XO te transporteren.
Het functiegedeelte van deze opdrachten is = 20 voor transport van trommel naar matrix, = 21 voor transport van matrix naar trommel.
De zeven hoogste adrescijfers bepalen het trommelkanaal, waar het transport betrekking op heeft.
De laagste vijf adrescijfers, die oorspronkelijk zouden aangeven op welk snel kanaal het transport betrekking had, worden nu gebruikt om de "kwarten" aan te geven. Hiertoe denkt men de plaatsen van beide kanalen verdeeld in vier kwarten (van 0 t/m 7, van 8 t/m 15, van 16 t/m 23 en van 24 t/m 31), genummerd 0, 1, 2 en 3. Het is nu mogelijk informatie van een willekeurig kwart van het ene kanaal te transporteren naar een willekeurig kwart van het andere kanaal. Als t = nummer van het trommelkwart en m = nummer van het matrix-kwart, dan moet in de laagste vijf adrescijfers, het "plaatsgedeelte" van de opdracht dus, worden ingevuld p = 8t + 4 + m.

In tabel:

trommelkwart t = \ matrixkwart m =	0	1	2	3
0	4	5	6	7
1	12	13	14	15
2	20	21	22	23
3	28	29	30	31

De kwart-tracktransporten duren 35 getaltijden (14.6 msec) d.w.z. evenlang als de volledige tracktransporten.

Het invoerprogramma

In MR 25 is beschreven, hoe het invoerprogramma als zelfstandig programma functionneert. Hierbij worden een aantal subroutines gebruikt, die op zichzelf ook door de programmeur aangeroepen mogen worden. (Alleen de subroutines voor het lezen van binaire woorden zijn in MR 25 (blz. 42) genoemd.) Hier wordt beschreven, hoe de programmeur "vanuit zijn programma" het standaardinvoer-programma of gedeelten daarvan kan benutten.

Met de sprongopdracht 6 25 X 16 start men het invoerprogram-ma schrijvend ("autostart 0"), met 6 26 X 16 controlerend ("autostart 1"). De laatste sprong wordt zelden gebruikt.

Een programma dat steeds een rij getallen verwerkt kan hier met vrucht gebruik van maken. Het complete bandje, beginnend met blank tape, adres-indicatie en soortspecificatie, controle-wissel en duplicaat wordt afgesloten door RJ gevolgd door de sprongopdracht naar de eerst uit te voeren opdracht na het invoeren van de getallen; hierachter komt na een stukje blank weer zo'n bandje etc.
Het programma geeft, zodra een nieuwe rij constanten nodig is, met de sprong 6 25 X 16 de controle geheel aan het invoer-programma, dat nu als zelfstandig programma, en niet als sub-routine werkt. Dit impliceert, dat alle controlecombinaties op de band moeten voorkomen, en na afloop door een expliciete RJ combinatie de besturing naar het programma terug verwezen moet worden.

Opm.1: Als men, beginnend bij een bepaald adres (b.v. 0 A 0) een rij getallen inleest van variabele lengte, kan het program-ma de lengte van de rij detecteren, mits de RJ-sprong op de band onmiddellijk volgt op het laatste getal. D.w.z. na het laatste getal mag niet de controlecombinatie RC komen, waarna opnieuw adresindicatie nodig is. (zie MR 25, blz. 38-40)
Als de rij n getallen lang is - dan is dus (n - 1) A 0 het laatst ingevulde adres! - dan staat in 25 X 0, waar de plaats van wegbergen c.q. vergelijken onthouden wordt,

in geval van schrijvend lezen:

```
25 X O:    12    n    A    O
           29    O    X    O
```

in geval van controlerend lezen

```
25 X O:     9    n    A    O
           28    O    X    8
```

Opm.2: Als steeds wordt de inhoud van kanaal XO geheel door het invoerprogramma gebruikt.

In het bovenstaande schema is het verwerken van de geassembleerde moleculen aan het invoerprogramma gedelegeerd. Een andere mogelijkheid is, dat men alleen van het invoerprogramma vraagt, de woorden te assembleren: in dit geval mogen adresindicaties en controlewissels niet meer op de band voorkomen, alleen de moleculen en hun soortspecificatie. (Hieronder valt RE: skip blank, X tot R.)

Men moet hier met enige omzichtigheid te werk gaan in verband met de informatie, die tijdens het bandlezen zich in XO bevindt. Dit is gedeeltelijk vaste informatie - o.a. een stuk programma voor de decimaal-binaire conversie, gedeeltelijk variabele informatie. In eerste instantie is dit de (laatste) soortspecificatie, maar bij het lezen van getallen ook het eerste symbool van de volgende informatie-eenheid. (Het ARMAC-leesprogramma kan het einde van een getal slechts detecteren, door een pentade teveel te lezen en dan te ontdekken, dat dit géén decimaal cijfer is; deze pentade, die dus deel uitmaakt van de volgende informatie-eenheid, moet onthouden worden; ook dit geschiedt in XO.)

Met de subroutine aanroep

```
22  30  X   O  =) "lees molecuul van de band ≠ (S)"
```

worden automatisch soortspecificaties in rekening gebracht. Na de soortspecificatie RE skipt het programma blank tape totdat na een nieuwe soortspecificatie een echt molecuul geassembleerd wordt. Met dit woord in S komt de besturing terug.

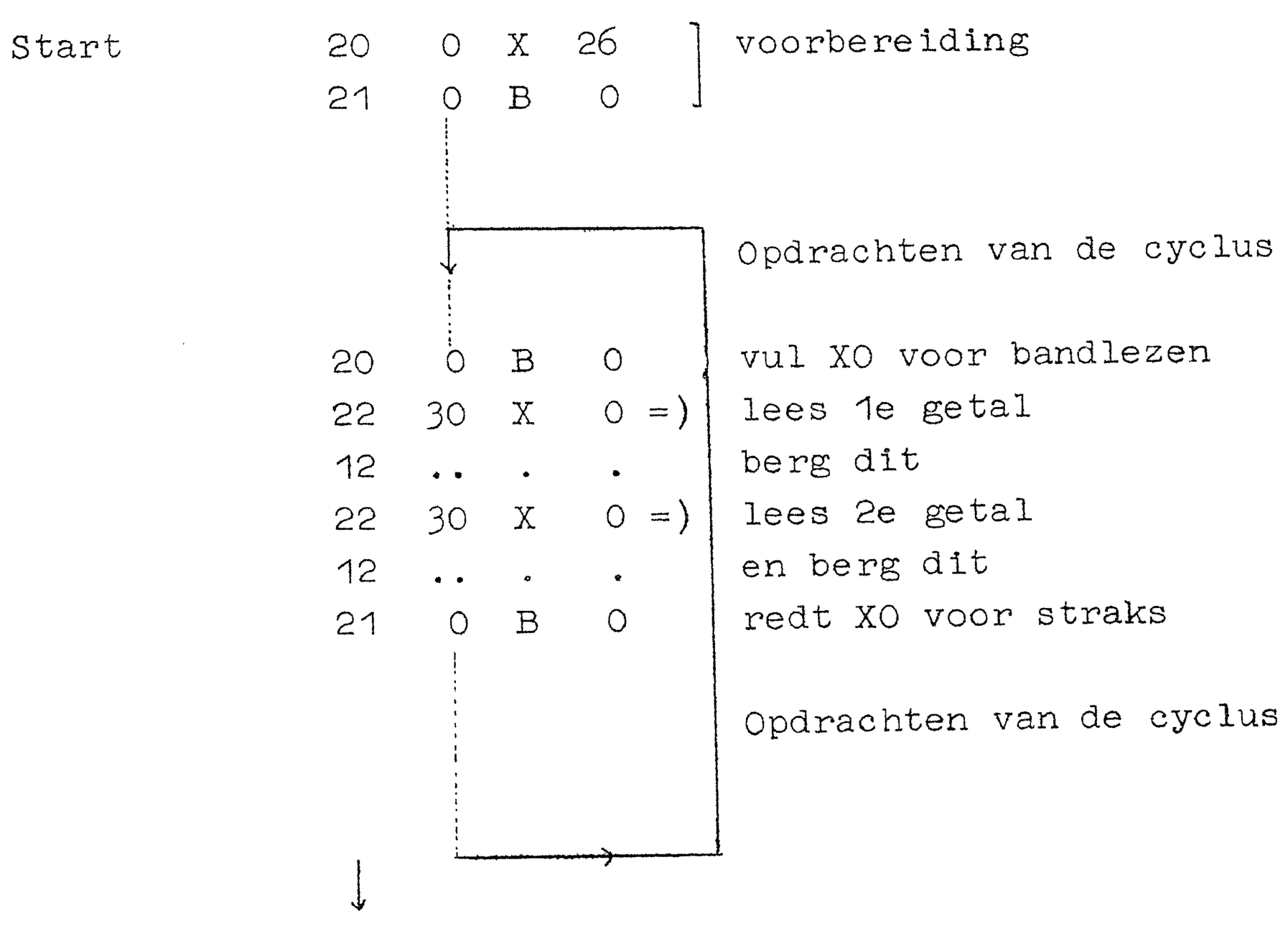
De aanvankelijke vulling van kanaal XO geschiedt door het tracktransport

20 0 X 26 Voorbereiding, stel soort in op RE.

Hierdoor wordt de noodzakelijke vaste informatie op XO ingevuld, voorts wordt de soortspecificatie ingesteld op "skip blank X tot R". Dit tracktransport mag dus alleen ter inleiding van de aanroep 22 30 X 0 gegeven worden, als de band nog met de pentades blank onder de lezer ligt, d.w.z. als regel aan het begin van een programma.

Vaak zal het verwerkingsprogramma kanaal XO nodig hebben; in dat geval moet men na het bandlezen kanaal XO op een of ander trommel-kanaal "redden".

Een programma dat b.v. per cyclus twee getallen van de band nodig heeft, ziet er dan ongeveer als volgt uit (kanaal BO op de trommel is gereserveerd als bewaarplaats van "XO tijdens bandlezen".



Herstartbaarheid.

Wij noemen een programma herstartbaar, als de bij de invoer ingebrachte informatie tijdens de berekening onveranderd in het geheugen blijft staan. Aangezien het onmogelijk is tijdens de invoer plaatsen in kanaal XO of de buffer te vullen, betreft de herstartbaarheid in eerste instantie alleen de trommeladressen.

Bij een dergelijk programma is het mogelijk, de berekening opnieuw van voren af aan uit te voeren door enkel, zonder weer de banden in te lezen, op de startknop te drukken, als na enige tijd rekenen de machine door wat voor oorzaak dan ook gestopt is. Immers, alle benodigde informatie staat nog onaangetast in de machine.

De herstartbaarheid is een vereiste, waaraan elk programma moet voldoen, ook programma's, die in theorie slechts één keer op hun beginadres gestart hoeven te worden. De praktijk heeft uitgewezen, dat dit een idealisatie is, die zelden door de werkelijkheid gedekt wordt.

Het is b.v. mogelijk, dat de machine na een minuut stopt, op het falen van de parity-check. Een herstartbaar programma start men dan onmiddellijk weer. Blijkt deze fout permanent, dan zal de technische hulp van de operateur ingeroepen worden, die, door het kanaal schoon te maken, de fout waarschijnlijk in een oogwenk hersteld zal hebben. Of men ontdekt b.v. bij het tikken van de eerste regel, dat het papier scheef in de schrijfmachine zit. Men stopt de machine, doet het recht en wil opnieuw beginnen. In al deze situaties is het kennelijk hinderlijk, als men steeds genoodzaakt zou zijn, de banden opnieuw in te moeten lezen.

Nog sterker is het argument bij het testen van een programma. Stel dat men op hoop van zegen het programma gestart heeft, en aan de hand van de uitkomst - of uit het wegblijven daarvan! - tot de conclusie komt, dat er dan wel iets niet deugen zal aan het programma. (Onervaren programmeurs neigen vaak tot de haast altijd onjuiste veronderstelling, dat de ARMAC defect is!). In zo'n geval zal men opnieuw de ARMAC het programma willen laten uitvoeren, om de machine te stoppen bij het vormen van een

bekend tussenresultaat; als dit b.v. al fout is, start men de machine weer, maar stopt nu eerder, etc. op deze wijze de fout localiserend. Telkens inlezen zou het testen bovenmate vertragen; nog anders wordt de situatie, als de eerste fout gevonden en in de machine "met de hand" gerepareerd is: op dat moment zijn de banden niet meer correct, voordat een correctieband gemaakt is! Met het opstellen en ponsen van een correctieband wacht men echter altijd, totdat zo niet alle, dan toch wel een aanzienlijk aantal fouten gevonden en hersteld zijn.

De formele consequentie van de herstartbaarheid is dat elk gebruikt trommel-adres óf tijdens de invoer wordt ingevuld óf door het programma, maar niet door beide.

Laat ons het speciale geval beschouwen, dat een bepaalde cyclus steeds 10 keer doorlopen moet worden. Om dit te programmeren reserveert men een geheugenplaats voor de z.g. telling, waarin b.v. onthouden wordt, hoe vaak de cyclus nog doorlopen moet worden. Bij het niet herstartbare programma wordt bij de invoer op dit adres het getal 10 ingevuld. Steeds als de cyclus één maal is doorlopen wordt van de telling 1 afgetrokken en wordt getest, of nog niet de telling = (-)0 is; zodra de telling nul geworden is, verlaat het programma de cyclus. Om te zorgen, dat de volgende keer de cyclus weer tien maal wordt doorlopen, wordt dan onmiddellijk na het verlaten de telling hersteld, het programma wordt "weer goed gemaakt". Zou hier de machine midden in de cyclus gestopt zijn, dan was de telling al bedorven en nog niet hersteld: het programma is niet herstartbaar. In het wel herstartbare programma wordt bij de invoer de voor de telling gereserveerde plaats opengelaten: vlak voordat de besturing de cyclus ingaat, wordt door het programma "de telling gezet", d.w.z. het aantal nog uit te voeren doorgangen wordt = 10 gemaakt.

Het zojuist aangehaalde voorbeeld is representatief. Goldt het hier een telling, een administratieve grootheid dus, het is eveneens van toepassing voor arithmetische grootheden, zoals b.v. adressen, waarin men een som opbouwt; deze moeten eerst door het programma worden schoongemaakt (d.w.z. er moet nul

op geschreven worden.) Als in een cyclus variabele opdrachten voorkomen, is hierop hetzelfde van toepassing. Het geven van de initiele waarden aan adressen, wier inhoud in een cyclus gewijzigd worden, wordt wel "de voorbereiding" genoemd; als de voorbereiding alleen een telling betreft, wordt van "het zetten van de telling" gesproken.

We zien, dat in deze voorbeelden we het programma, door aan de eis van de herstartbaarheid te voldoen, niet of niet noemenswaard langer maken: door de voorbereiding vervalt het herstellen.

Het begrip herstartbaarheid is niet alleen van toepassing op het programma als geheel, maar ook op onderdelen van een programma.

Mocht de machine stoppen omdat b.v. de parity-check een fout detecteert, dan is het **dikwijls niet** noodzakelijk, de gehele berekening van voren af aan over te doen. In een verstandig opgezet programma is er bijna altijd wel gelegenheid, om de berekening een eindje terug op te vatten en van daar opnieuw uit te voeren. Dit impliceert b.v. dat de voorbereiding van een cyclus onmiddellijk aan de binnenkomst in de cyclus vooraf moet gaan. Zou n.l. de machine in de cyclus stoppen en zou de hele cyclus overgedaan moeten worden, dan moet dit inclusief de voorbereiding: zou na de voorbereiding eerst nog een stukje van de berekening komen, dan compliceert men het "een eindje terug beginnen" wel, omdat het niet gezegd is, dat het onschadelijk is, dit stukje berekening een tweede maal uit te voeren.

Een uitzondering op de regel van de herstartbaarheid moet soms gemaakt worden voor numerieke informatie: als in de berekening van eigenvectoren van een matrix de eerste eigenvector gevonden is, moet de matrix gereduceerd worden, waarna op de gereduceerde matrix hetzelfde proces uitgevoerd moet worden, om de volgende eigenvector te vinden, etc. Omdat, zodra een eigenvector gevonden is, de matrix alleen nog maar elementsgewijs gebruikt wordt, om de overeenkomstige elementen van de gereduceerde matrix te berekenen, kan de gereduceerde matrix de oorspronkelijke in het geheugen vervangen. Om aan de herstartbaarheid

te voldoen, zouden we in het geheugen ruimte voor twee matrices moeten reserveren: omdat echter de per matrix beschikbare ruimte ons een bovengrens oplegt voor de orde van nog verwerkbare matrices, is hier de herstartbaarheid wat betreft de numerieke gegevens opgeofferd aan het "bereik" van het programma. Dit offer is echter alleen toelaatbaar voor de numerieke informatie, en alleen indien nodig.

Herstartbaarheid en de bufferschrijfopdracht

Dankzij de buffer en de bufferschrijfopdracht krijgt men de herstartbaarheid van "kleine cycli" vaak cadeau. De bufferschrijfopdracht stelt ons immers in staat op een plaats van de buffer te schrijven, zonder de inhoud van het overeenkomstige adres op de trommel te wijzigen.

Als nu een cyclus zich afspeelt in één trommelkanaal, dan kunnen alle wijzigingen uitsluitend door bufferschrijfopdrachten worden uitgevoerd. Als men bij de invoer alle initiele waarden op de trommel heeft gezet, blijven deze bij doorgang door de cyclus onveranderd op de trommel staan. De voorbereiding hoeft dan slechts te bestaan uit het inlezen van de buffer.

Wie dit wil toepassen, controlere terdege:
1. dat inderdaad het maagdelijk trommelkanaal in de buffer wordt gelezen, als de besturing weer de cyclus ingaat; dit is het geval als tussen het verlaten van de cyclus - in de buffer een "bedorven" copie achterlatend - en het opnieuw de cyclus ingaan opdrachten uit een ander trommelkanaal zijn uitgevoerd.
2. dat tijdens de cyclus geen andere kanalen in de buffer worden gelezen: in de cyclus mag b.v. niet de deelsubroutine of een typroutine worden aangeroepen.

Aan de onder 1 genoemde voorwaarde is meestal vanzelf voldaan, zo niet, dan is het soms wel mogelijk, opzettelijk even naar een ander trommelkanaal te springen en weer terug; soms kan dit niet; n.l. als in de buffer vitale informatie staat! Wie een programma met twee cycli om elkaar heen in één kanaal heeft staan - b.v. "vector maal vector" als cyclus in het programma "matrix maal vector" - kan alleen de buitenste cyclus voorbereiden door

het inlezen van de buffer, de voorbereiding van de binnenste cyclus moet expliciet geprogrammeerd worden.

Wie de onder 2 genoemde voorwaarde veronachtzaamt, maakt een fout, die niet zonder meer te herstellen is: de bufferschrijf-opdrachten moeten door normale schrijfopdrachten worden vervangen, tengevolge waarvan de voorbereiding nu expliciet geprogrammeerd moet worden ingelast.

Als het een kleine subroutine betreft, is er misschien nog wel ruimte in het kanaal zelf, misschien kan men de subroutine met een tracktransport in kanaal XO zetten. Als het een uitgebreide of vaste subroutine betreft, is er een andere techniek, die b.v. nogal eens gevolgd wordt bij het typen van een rij getallen: men plaatst de cyclus, die de te typen getallen in het geheugen aanhaalt, in maagdelijke toestand met een tracktransport in kanaal XO (In dit voorbeeld mag de cyclus vanwege de typroutine natuurlijk niet de adressen OXO t/m 5XO gebruiken.) Dezelfde techniek is toegepast in een programma waar af en toe onder controle van een cyclus met de nodige variabele opdrachten een groot aantal sinus- en cosinusfuncties berekend moesten worden. In het laatste geval waren snelheidsoverwegingen de drijfveer, in het eerste gemakzucht. (Zie onder)

Programma in het snelle kanaal

Kanaal XO omvat 32 volwaardige adressen, waarin men zowel getallen als opdrachten kan bergen; als opdrachten uit kanaal XO gehoorzaamd worden, wordt de buffer hierbij niet gebruikt. Het is dan ook niet om technische, maar om organisatorische redenen, dat wij aanraden het snelle kanaal niet al te veel voor opdrachten te gebruiken.

De "service" van een programma wordt n.l. aanmerkelijk gecompliceerder als kanaal XO achter elkaar verschillende stukken programma bevat, m.a.w. als men kanaal XO gebruikt als "geprogrammeerde buffer". In dat geval stuit men n.l. op moeilijkheden als men de machine met behulp van het stopadres wil stoppen op een bepaalde opdracht, die voor zijn uitvoering in kanaal XO is gezet. Wil men de machine stoppen op dit adres, dan stopt de machine óók, zodra een andere opdracht van deze zelfde plaats wordt aangehaald.

Voorbeeld: Subroutine matrix maal vector

Voorponsing RFR 0 s_0 k_0

RFB 0 s_1 k_1

RFC p_2 s_2 k_2

RFH p_3 s_3 k_3

RFT n X 0

als s_0 k_0 het kanaal is, dat voor de subroutine gereserveerd is,
als s_1 k_1 en (eventuele) volgende gereserveerd zijn voor de factorvector,
als p_2 s_2 k_2 het beginadres van de productfactor is,
als p_3 s_3 k_3 het beginadres van de matrix is,
als n de orde van de matrix is.

Opm.: Zowel subroutine als factorvector beginnen dus op het begin van een kanaal.

Matrix-elementen en vector-elementen staan aansluitend. In formule

$$\sum_{i=0}^{n-1} \{i + jn\ H\ 0\}.\{i\ B\ 0\} \neq \{j\ C\ 0\} \quad \text{voor} \quad 0 \leq j \leq n-1.$$

Aanroep: 22 0 s_0 k_0 =) De subroutine bederft X 0, de aanroep dient van af de trommel te geschieden.

Tot zover de "gebruiksaanwijzingen" voor de subroutine.

Het programma haalt de elementen van de factorvector kanaalsgewijs in X 0; zodra deze in X 0 staan wordt het tempo der vermenigvuldiging bepaald door de wachttijd op de matrixelementen, die op de trommel staan. Voor elke sommatie heeft men matrixelementen in opeenvolgende adressen nodig: zij worden van achteren naar voren afgewerkt, zodat elke term 31 getaltijden duurt, en niet 33, wat het geval geweest zou zijn als men ze in volgorde van opklimmend adres had verwerkt.
Voorts test het programma of de factorvector in één kanaal staat. Zo ja, dan wordt eens en vooral aan het begin van de subroutine de factorvector in X 0 geplaatst, verder worden dan geen tracktransporten uitgevoerd; als de factorvector meer kanalen beslaat, alterneren de tracktransporten aanhoudend.

Deze subroutine is speciaal geconstrueerd meer met de bedoeling een ingewikkeld, dan een "verstandig" programma te maken. In een kort bestek zijn hier een aanzienlijk aantal trucjes geconcentreerd, en wij hopen hiermee een instructief voorbeeld gegeven te hebben met betrekking tot de eis der herstartbaarheid - in dit geval de voorbereiding van het interne cyclusje - en de mogelijkheden van de bufferschrijfopdracht.

Wij willen dit commentaar op deze subroutine besluiten met enige aanmerkingen.

Ten eerste heeft het programma er geen gebruik van gemaakt, dat als de orde > 32 is, hij in elk geval < 64: de matrix overschrijdt tegen die tijd n.l. allang de capaciteit van het geheugen.
Voorts is het aanvechtbaar of het de moeite loont om als de orde $\leq$ 32 is, de factorvector maar een keer in XO te transporteren, en niet, wat een aanmerkelijk eenvoudiger programma had gegeven, n keer, n.l. aan het begin van elke vermenigvuldiging van vector maal vector.
Dit is een besparing $\sim$ n, terwijl het proces $\sim n^2$ duurt.
Men doet er meestal het beste aan, al zijn vernuft te spenderen aan het "meest intensieve gedeelte" van het programma; voor de stukken programma die een orde minder vaak doorlopen worden, kan men dan met een gerust geweten een minder geraffineerd programma maken.
En tenslotte: dit programma is <u>te</u> "getruct".
Wie zijn hele programma in deze trant maakt, zal merken, dat het waarschijnlijk heel veel fouten bevat, en dat deze fouten niet met kleine wijzigingen te herstellen zijn. Het anderszins wijzigen van het programma (b.v. het uittypen van nog een tussenresultaat) kan dan ook wel eens tot ingrijpende veranderingen aanleiding geven.

Subroutine matrix maal vector

		RD / RA	O	R	O	
=)	(0	28	30	X	2	zet link
		10	31	R	0	) n ≠ S
	(1	18	31	R	0	n^2 ≠ S
		8	23	R	0	)] zet opdr. voor matrixelement
	2	28	23	X	10	] op 1 voorbij het laatste
		26	31	T	11	= 26 (n-1)X 12 : (n-1) ≠ S
	3	26	5	X	22	] plaats in kanaal van laatste
		24	29	X	20	] vectorelement wordt in A gezet
	4	0	22	R	0	] vorm aanhaalopdr. achterste vectorelement - 32
		24	29	X	30	] in (n-1) begrepen 32-voud
	5	28	0	X	8	] in S; is dit ≠ 0?
		15	7	R	0	→ vector langer dan 1 kanaal
	6	20	0	B	0	lees het ene kanaal in XO
		24	32	X	4	tel de ontbrekende 32 op
	7	25	32	X	12	verminder de (nooit gehoor-zaamde) transportopdracht
b5 →		8	6	R	0	voltooi tracktransport
	8	28	1	X	10	plaats initiele tracktransport
		28	2	X	2	zet initiele aanhaalopdr. (-32)
	9	26	31	T	3	= 26 n-1 X 4 : n-1 ≠ A
		6	27	R	0	⇒ vorm 1e. wegbergopdracht
b29 ⇒	10	26	0	X	12	0 ≠ S (staart partiele som)
		28	0	X	10	0 ≠ kop partiele som
	11	2	1	R	0	] zet initiele
		28	18	X	2	] tracktransport
	12	2	2	R	0	initiele aanhaalopdracht (-32)
		6	14	R	0	⇒
b24 ⇒	13	27	1	X	4	] aflaging aanhaal-
a20 →		0	22	R	0	] opdracht voor
b12 →	14	28	22	X	2	] vectorelement
		28	6	X	20	geen nieuw kanaal naar XO?
	15	15	20	R	0	→
		2	6	R	0	] is de kanaalgrens

Subroutine matrix maal vector

```
                16 |  1 : 18 : R :  0 |   tevens het einde
                   | 28 : 34 : X : 20 |   van de vector?
                17 | 14 : 25 : R :  0 | → einde vector x vector
                   |  3 : 18 : R :  0 |   tracktransport  doen
             (18   |    : RX : 1 :    |   20 0 Bj
                   |    :    :   :    | ) 24 32 X 4
                19 | 29 : 18 : X :  2 |   en aflagen
                   | 26 : 32 : X :  4 |   32 ≠ A
                20 |  7 : 13 : R :  0 | ⇒ herstel aanhaalopdracht
a15 ⇒              | 27 :  1 : X :  4 |   aflaging van vermenig-
                21 |  0 : 23 : R :  0 |   vuldigopdracht
                   | 28 : 23 : X :  2 |   met matrix element
             (22   |  1 : 4064 : X : 0 |  2 i X 0
                   | 24 : 34 : X : 22 | ) A↔S
             (23   | 16 :  0 : H :  0 |   x matrix element
                   |  0 :  0 : R :  0 | ) tel kop product bij kop
                24 | 28 :  0 : X :  2 |   partiele som
                   |  6 : 13 : R :  0 |   ⇒ volgende vermenigvuldiging
a17 ⇒ 25           | 26 : 32 : X : 30 |   plaats afgerond
                   |  8 :  0 : R :  0 |   product in S
             (26   | 12 :  0 : C :  0 |   berg element product vector
                   | 27 :  1 : X :  4 |   aflaging van
b9 →  27           |  0 : 26 : R :  0 |   de wegbergopdracht
                   | 28 : 26 : X :  2 |   element productvector
                28 |  3 : 31 : R :  0 |   telling aantal
                   | 24 :  1 : X :  4 |   vector x vector
                29 | 29 : 31 : X :  2 |   vermenigvuldigingen
                   | 14 : 10 : R :  0 | → nog niet klaar
             (30   |    : RX : 1 :    | ⇒ link
                   |    :    :   :    | )
             (31   |  0 :  0 : T :  0 |   = n
                   |  0 :  0 : X :  0 | )
```