MATHEMATISCH CENTRUM
2e BOERHAAVESTRAAT 49
AMSTERDAM
REKENAFDELING

# HANDBOEK VOOR DE PROGRAMMEUR

## FERTA

## DEEL I

door

E.W. Dijkstra

MR 17

1955

INHOUD:



- - - - - -

Inleiding.

In dit rapport wordt beschreven het opstellen van rekenschema's - kortweg het maken van "programma's" of het "**programmeren**" genoemd - voor de electronische rekenmachine FERTA.

FERTA is te beschouwen als een tweede ARRA; verbeteringen, wijzigingen en uitbreidingen zijn echter zo ingrijpend, dat op FERTA niet van toepassing zijn de eerder van de hand van E.W. Dijkstra verschenen rapporten over het programmeren voor de ARRA, n.l.

MR 12 Functionele beschrijving van de ARRA.
MR 14 In- en uitvoer van de ARRA.
MR 16 "Drijvende-Komma" rekentechniek.

In principe hoeft de programmeur maar heel weinig te weten over de physische opbouw en werking van de machine, en heeft hij voldoende aan een volledig abstracte, formeel complete, programmeerhandleiding. De praktijk heeft echter uitgewezen, dat het nuttig is, van de diverse logische onderdelen van de machine een zo concrete voorstelling te hebben, dat **men**, aan de machine gezeten, althans iets van de gang van het rekenproces kan volgen. Aan de beschrijving van de machine is dit eerste deel, dat nu voor U ligt, in hoofdzaak gewijd.

Algemene beschrijving.

Uiterlijk bestaat de machine - afgezien van de voeding - uit twee gedeelten, het "rek" en de "console"; deze twee zijn door een kabel van vele draden, die onder de vloer loopt, met elkaar verbonden.

Functioneel bestaat de machine uit vijf grote onderdelen, n.l. Rekenorgaan, Geheugen, Besturing, Invoer en Uitvoer, waarvan de eerste drie zich in hoofdzaak op het rek, de laatste twee zich op de console bevinden. Op de console bevinden zich bovendien de bedieningspanelen, de belangrijkste controlelampjes en enige andere apparatuur, welke hoofdzakelijk dient ter localisatie van eventuele fouten in de machine.

Het rekenorgaan is in vele opzichten te vergelijken met een tafelrekenmachine. Een tafelrekenmachine heeft, zoals bekend, drie registers (het instelregister alias toetsenbord, het resultaatregister en het omwentelings- of quotiëntregister) en is in staat de vier rekenkundige bewerkingen (optelling, aftrekking, vermenigvuldiging en deling) uit te voeren. In het gebruik brengt dit met zich mee, dat men, bij de uitvoering van gecompliceerdere berekeningen met behulp van een tafelrekenmachine, veelvuldig getallen op het instelregister moet aanslaan en tussenresultaten op een vel papier moet overschrijven, om ze later in de berekening weer te kunnen gebruiken.

Bezien we nu het rekenorgaan van de automatische machine: dit heeft ook drie registers, die ruwweg dezelfde functies hebben als de drie registers van de tafelmachine. Van automatisme kan echter slechts sprake zijn, als ook voor het inslaan in het instelregister en het schrijven van tussenresultaten op papier (dus getallentransport naar, resp. van het rekenorgaan) een aequivalent aanwezig is. In feite is dit zo uitgevoerd, dat het inslaan in het instelregister en de daarop volgende rekenkundige bewerking in een adem geschiedt, terwijl het "schrijven op het vel papier" door een aparte operatie wordt uitgevoerd. Als vel papier fungeert het tweede onderdeel, het geheugen, in ons geval een z.g. magnetische trommel, waarop getallen gemagnetiseerd kunnen worden ("geschreven") en waarvan ook weer gelezen kan worden. Om de machine in staat te stellen, dit ordelijk te kunnen doen, is het geheugen verdeeld in vakjes, die juist elk een getal kunnen bevatten; deze vakjes zijn, om verwijzing naar een speciaal vakje mogelijk te maken, genummerd 0, 1, 2, 3, ..... en worden aangeduid "adressen" (precieser, met de "adressen" worden zowel de vakjes, als hun nummers aangeduid). Getallen hebben de vaste lengte van 30 tweetallige cijfers (0 of 1), n.l. een voor het teken, en de negenentwintig andere ter bepaling van de numerieke waarde. Ieder adres bevat steeds 30 tweetallige cijfers

en is dus juist in staat om een getal te bergen. Een getal kan onbeperkt vaak uit hetzelfde adres gelezen worden en blijft onveranderd in dit adres staan, totdat het door een ander getal wordt overschreven: men schrijft als het ware als met potlood op een stuk papier, het vlakgom pas hanterend, als men op dezelfde plaats iets nieuws wil noteren. Zo wordt een getal, dat als tussenresultaat gevormd wordt, op een of ander bepaald adres geschreven; iedere keer, dat later in de loop van de berekening dit getal weer gebruikt wordt, wordt verwezen naar "het getal op dat en dat adres". In overeenstemming hiermede pleegt de specificatie van de operaties naast de aard van de bewerking (b.v. aftrekking of vermenigvuldiging) naar adressen te refereren.

Het rekenorgaan omvat drie registers, M, A en S genaamd, alle drie met de capaciteit van 30 cijfers. Het register M is te vergelijken met het instelregister van de tafelrekenmachine en vormt de brug tussen het geheugen en de rest van het rekenorgaan; het is voor de programmeur niet toegankelijk en komt verder in deze handleiding niet voor: als in het vervolg van "de registers van het rekenorgaan" gesproken wordt, worden de registers A en S bedoeld. Het A-register en het S-register vervullen samen de taken van resultaat- en omwentelingsregister; zij zijn in vele opzichten gelijkwaardig.

De handelingen, waarbij transport van een getal tussen geheugen en rekenorgaan plaats vindt, vallen natuurlijkerwijs uiteen in twee groepen:

a. leeshandelingen. Hierbij wordt een getal van de trommel gelezen, en naar het rekenorgaan gevoerd, waar het op verschillende wijze verwerkt kan worden: het getal kan - al of niet met tekenwisseling - in registers A of S geplaatst worden, met vernietiging van de oude inhoud van het betroffen register. Dit noemt men "schoon inlezen" in tegenstelling tot het z.g. "additief inlezen" waarbij het van de trommel gehaalde getal - al of niet na tekenwisseling - bij de inhoud van A of S wordt opgeteld. De omschrijving van een dergelijke opdracht luidt b.v.:"Trek het getal, dat staat - "te vinden is" - op dat en dat adres af van de inhoud van S en plaats het resultaat weer in S." Tevens vallen onder de leeshandelingen de vermenigvuldigingen en delingen, waarbij het aangehaalde getal als factor, resp. deler fungeert. Zij laten alle de inhoud van het geheugen onveranderd.

b. schrijfhandelingen. Hierbij wordt de inhoud van A of S - al of niet met tekenwisseling - op een willekeurig adres in het geheugen geschreven. Hierbij verandert de inhoud van het rekenorgaan niet, de inhoud van het betroffen adres (i.h.a.!) wel. Wat voordien op

dit adres stond, wordt vernietigd: dit mag dus geen nog "vitale" informatie zijn!

Vaag is hiermede aangeduid, wat voor soort handelingen de machine al zo verricht; hoe dit gebeurt, is nu - zij het in even grote trekken - aan de orde. Daartoe bezien wij nu het volgende onderdeel van de machine: de besturing. De besturing heeft twee eigen registers tot haar beschikking, n.l. het opdrachtregister R en de opdrachtteller T. Het opdrachtregister R is in staat 15 tweetallige cijfers te bevatten, juist de 15 cijfers, die n.l. nodig zijn om de opdracht, die bezig is uitgevoerd te worden, volledig vast te leggen: alle operaties, waartoe de machine in staat is, zijn n.l. met behulp van een (15 cijfers beslaande) nummercode uitgecodeerd volgens conventies, die straks nader uiteengezet zullen worden. Het belangrijkste op dit moment is, dat twee opdrachten samen juist 30 cijfers beslaan, en dus samen, een z.g. "opdrachtenkoppel" vormend, op een adres geborgen kunnen worden. In dit adres staan zij achter elkaar, ter onderscheiding wordt de ene de a-opdracht, de andere de b-opdracht genoemd. Men realisere zich, dat aan de 30 cijfers in een adres niet te zien is, of zij bedoeld zijn als een getal, of als een opdrachtenkoppel. Beide kunnen op een willekeurig adres in het geheugen staan en beide zijn .... alleen maar een rij van 30 tweetallige cijfers! Om die reden is het dienstig voor een rij tweetallige cijfers van een of andere lengte de neutrale term "woord" in te voeren. Of een woord van 30 cijfers, zoals het op een of ander adres staat, een getal, dan wel een opdrachtenkoppel voorstelt, is een quaestie van interpretatie van het woord, en wordt in feite uitgemaakt door de omstandigheden, waaronder de machine van de zich op dit adres bevindende informatie gebruik maakt. (Als de besturing een getal zoekt, vindt zij een getal - interpreteert n.l. het 30-cijferwoord als getal; evenzo als de besturing een opdrachtenkoppel zoekt, vindt zij een opdrachtenkoppel).

De opdrachtteller T bevat een woord van 11 cijfers, de eerste 10 bepalen een adres (dat dus lopen kan van 0 t/m 1023 = $2^{10} - 1$), de laatste bevat de indicatie, of het gaat om de a-opdracht of de b-opdracht. De inhoud van T bepaalt de plaats op de trommel, waar juist een opdracht staat; om op de trommel een getal te localiseren kan men "2 maal zo grof" te werk gaan: omdat daar de a-b-indicatie vervalt, geschiedt dit met behulp van 10 cijfers. Een dergelijk tiental vormt een vast onderdeel van elke opdracht, n.l. het numerieke gedeelte; de resterende vijf cijfers - een opdracht besloeg immers 15 cijfers - vormen het functiegedeelte van de opdracht. In overeenstemming hiermee is het opdrachtregister R dan ook gesplitst in twee gedeelten: het functieregister F en het numerieke

register N, met capaciteiten van 5 resp. 10 cijfers.

Terwijl een opdracht uitgevoerd wordt, staat deze in R, precieser bevindt zich het functiegedeelte in F en het numerieke gedeelte in N. In de opdrachtteller T bevindt zich het woord, dat bepaalt, waarvandaan uit het geheugen - d.w.z. adres en a-b-indicatie - deze opdracht gelezen is. En hier heeft de besturing zich doen gelden in de z.g. opdrachtselectie. Als de machine, die slechts een opdracht uit kan voeren, mits deze zich in register R bevindt, niet in staat zou zijn, nieuwe opdrachten in R te plaatsen, zou er in totaal maar één handeling uitgevoerd kunnen worden! Als een opdracht uitgevoerd is, moet er dus een nieuwe opdracht - de volgende - in R geplaatst worden. Dit gebeurt dan ook: de nieuwe opdracht, die in R geplaatst wordt, komt uit het grote informatiereservoir van de machine: uit het geheugen. Hij wordt daartoe op de trommel opgezocht door de zojuist genoemde opdrachtselectie; de plaats op de trommel, waar de nieuwe opdracht vandaan wordt gehaald, ligt vast door de inhoud van T: de opdrachtteller "houdt de plaats in het geheugen bij", waar de op dat moment uit te voeren opdrachten vandaan moeten komen.

De tweede manier, waarop onder leiding van de besturing de verbinding met het geheugen tot stand gebracht kan worden, is via de getalselectie: deze legt, tijdens de uitvoering van een opdracht, het contact tussen een - heel - adres en een van de registers van het rekenorgaan, ter transport van een woord van 30 cijfers. Via de getalselectie worden de getallen uit het geheugen aangehaald, die in het rekenorgaan verwerkt worden, en worden zij eveneens teruggetransporteerd: het schrijven in een adres uit een van de registers van het rekenorgaan geschiedt n.l. ook via de getalselectie. Welk adres op de trommel, ter lezing of ter schrijving, door de getalselectie opgezocht wordt, is bepaald door wat op dat moment in N, het numerieke register van het opdrachtregister, staat.

We zien dus, dat opdrachtselectie en getalselectie in vele opzichten verschillen: kan via de eerste alleen gelezen worden, via de tweede kan tevens worden geschreven; zoekt de eerste slechts naar 15-cijfer-woorden, de tweede manipuleert uitsluitend met 30-cijfer-woorden; dienovereenkomstig werken ze dan ook onder controle van niet evenlange registers: wordt de opdrachtselectie geleid door het woord in T (11 cijfers), de getalselectie reageert op het woord in N (10 cijfers); is bij de eerste het contactpunt "aan de machinekant" te allen tijde het opdrachtregister R - en wordt dus het 15-cijfer-woord als opdracht geïnterpreteerd - bij de tweede kan het elk der registers van het rekenorgaan zijn: het 30-cijfer-woord wordt als getal gehanteerd.

wordt als getal gehanteerd.

Er is hier verschillende malen sprake geweest van de specificatie van een adres door een woord van 10 cijfers. In dit geval vat men dit 10-cijferwoord als getal in het tweetallig stelsel: dit getal nu komt overeen met het "nummer van het vakje", dat door dit 10-cijferwoord gelocaliseerd zij. Dit brengt met zich mee, dat de getalselectie $2^{10} = 1024$ adressen kan bestrijken, genummerd van 0 t/m 1023. De opdrachtselectie bestrijkt eveneens 1024 adressen, echter dank zij het feit, dat in een adres twee opdrachten staan en in de opdrachtteller T een extracijfer ter beschikking is, dat de keuze bepaalt - het is =0 voor de a-opdracht, =1 voor de b-opdracht - bestrijkt de opdrachtselectie 2048 opdrachten. Om in overeenstemming met de adresnummers te komen, doet men het beste het cijfer van de a-b-indicatie in T als eerste cijfer na de komma op te vatten: de opdrachtselectie zoekt dan waarlijk "halve adressen".

Ter illustratie: als een getal neergeschreven wordt op adres 235, dan is de inhoud van N gelijk aan 00111 01011 ; wordt echter b.v. de b-opdracht van adres 235 gelezen, dan is de inhoud van T gelijk aan 00111 01011,1 .

Bij de uitvoering van een opdracht, kunnen wij volstaan de activiteiten van de besturing te splitsen in twee fasen, waar de eerste fase voor alle opdrachten gelijk is.

In de eerste fase leest de besturing een opdracht. Dat wil dus zeggen, dat via de opdrachtselectie een opdracht in het opdrachtregister R ingelezen wordt vanuit het adres, dat met inbegrip van a- of b- gedefinieerd is in de opdrachtteller T. De vijf cijfers in F vormen het functiegedeelte f van de opdracht, de tien cijfers in N vormen het numeriek gedeelte n van de opdracht. Beide woorden lezen wij als getallen in het tweetalligstelsel; in het geheugen staat f "links van" (of "aan de hoge kant van") van n: als een enkel getal van 15 cijfers is de opdracht f,n dus gelijk aan $1024f + n$ . Aangezien voorts i.h.a. het functiegedeelte veel karakteristieker is dan het numerieke gedeelte, noemen wij de opdrachten naar hun functiegedeelte: b.v. een "8-opdracht" is een opdracht met f=8 en een willekeurig numeriek gedeelte. In het geheugen luidt deze opdracht dus 01000 , gevolgd door nog 10 willekeurige cijfers.

Na het inlezen wordt $\frac{1}{2}$ bij de inhoud van T opgeteld, dit met het oog op de volgende opdracht: is zojuist een a-opdracht ingelezen, dan is straks de b-opdracht uit hetzelfde adres aan bod, is juist een b-opdracht ingelezen, dan is straks de a-opdracht uit het numeriek volgende adres aan de beurt. Voor het zover komt, moet eerst de zojuist in R gelezen opdracht worden uitgevoerd, wat gebeurt in de tweede

fase. Ter bespreking van de tweede fase onderscheiden wij enige gevallen, omdat het verloop van de tweede fase afhangt van de f en n, die zich op dit moment in F en N bevinden.

Ia. Als f = 0,1,2,3, 8,9,10,11,16,17,18,19,20 of 21 is de opdracht wat eerder als een leeshandeling omschreven is. De besturing neemt opnieuw contact op met het geheugen, maar nu via de getalselectie. Zij leest dan het getal op adres n, voert dit naar het rekenkundig orgaan, waar de door f gespecificeerde handeling met dit getal wordt uitgevoerd, b.v. als f=0 wordt het ingelezen getal bij de inhoud van A opgeteld. Dan komt de besturing weer in de eerste fase, d.w.z. de volgende opdracht is aan bod.

Ib. Als f = 4,5,12 of 13 is de opdracht wat eerder een schrijfhandeling is genoemd. De besturing neemt ook hier opnieuw via de getalselectie contact op met adres n, maar schrijft hier nu de inhoud van A of S, al of niet van teken gewisseld, neer. Tevens wordt door deze operaties het conditieregister C ingevuld. Het conditieregister bestaat uit een cijfer c; als c=0 noemen we "de conditie positief", als c=1, zeggen we, "dat de conditie negatief is". Bij de schrijfhandelingen wordt in C het tekencijfer overgenomen van het getal, zoals het t.g.v. de schrijfhandeling in het geheugen terecht komt. De conditie wordt dus positief, als een positieve registerinhoud zonder of een negatieve registerinhoud met tekenwisseling wordt weggeschreven. Na afloop komt de besturing weer in de eerste fase en de volgende opdracht is aan de beurt.

II. Als f = 6,7,14,15,22,23,24,25,26,27,28 of 29 neemt de besturing niet opnieuw contact met het geheugen op, de getalselectie treedt niet in werking.

IIa. Als f = 24,25,26,27,28 of 29 is de opdracht f,n een uit de grote gevarieerde groep, die met de vage benamingen "schuif- en communicatieopdrachten" of "snelle bewerkingen" worden aangeduid. Wat hierbij in elk onderscheiden geval precies gebeurt, hangt van n af en zal later nauwkeurig beschreven worden. Wel geldt hier als regel dat de 24- en 25-opdracht een "additief karakter" hebben, de 26- en 27-opdracht een "schoon karakter", de 28- en 29-opdracht een "schrijfkarakter": de laatste twee zetten n.l. de conditie C, hoewel geen getal in het geheugen geschreven wordt. Tevens geldt hier, evenals bij de opdrachten uit groep I, dat opdrachten met f= even geen, met f= oneven wel een tekenwisseling inlassen. Onder de "schuif- en communicatieopdrachten" vallen de opdrachten, die het contact met de Invoer en de Uitvoer bewerkstelligen, dus de communicatie met de buitenwereld; vandaar de naam.

De "snelle bewerkingen" zijn in staat optellingen, aftrekkingen en vermenigvuldigingen van diverse allooi: met vaste, kleine getallen te verrichten. Dit kleine, vaste getal wordt nu door een groepje cijfers van n bepaald, terwijl de aard van de bewerking met behulp van de resterende cijfers van n wordt vastgelegd. Zo bestaat er b.v. de opdracht "tel 7 op bij de inhoud van A"; deze 7 wordt nu niet via de getalselectie van een of ander adres gelezen, maar wordt uit speciale cijferpositie's van R (precieser N) overgenomen. Deze "snelle bewerkingen" doen operaties, die ook met behulp van opdrachten uit de groep Ia uit te voeren zouden zijn, maar zij sparen ruimte - het getal 7 hoeft in het bovenstaande voorbeeld niet meer elders in het geheugen te staan - en doorgaans tijd; vandaar deze naam.

De snelste opdracht is de z.g. "Skip", als n = 10001 00000 (17 x 32 = 544); deze "doet niets", d.w.z. de besturing gaat meteen weer over in de eerste fase, waarin de volgende opdracht onder handen wordt genomen.

IIb. Als f = 22 of 23 en n = 0 is de opdracht een conditionele stopopdracht, d.w.z. als f,n = 22,0 en c = 0 ("conditie positief"), dan gaat de machine niet meer in de eerste fase over, d.w.z. de machine stopt. Men realisere zich, dat de opdrachtteller wel reeds met $\frac{1}{2}$ is opgehoogd, en dus de machine "klaar staat" om met de volgende opdracht te beginnen. Is c =1 ("conditie negatief"), dan werkt de opdracht 22,0 als Skip. Uit dien hoofde wordt 22,0 "de + conditionele Stop" genoemd. 23,0 reageert invers op de inhoud van C en heet "de - conditionele Stop".

IIc. Als f = 7 of 15 is de opdracht een van de ongeconditioneerde besturingsverplaatsingen of sprongopdrachten. De werking van deze opdrachten bestaat slechts hieruit, dat, als f = 7 de inhoud van T door n, als f = 15 de inhoud van T door $n+\frac{1}{2}$ wordt vervangen, waarna de machine weer in de eerste fase overgaat. Welke opdracht dan uit het geheugen aangehaald wordt, hangt af van wat zojuist in de opdrachtteller geplaatst is, m.a.w. het "op het rijtje" uitvoeren van opdrachten wordt door de sprongopdrachten onderbroken. Op welk adres het nieuwe rijtje begint, hangt af van het numeriek gedeelte van de sprongopdracht, terwijl de a-b-indicatie door f = 7 resp. 15 is vastgelegd.

IId. Als f = 6 of 14 is de opdracht een conditionele besturingsverplaatsing of sprongopdracht; dan hangt van de conditie af wat er gebeurt: Als de conditie + is, werken de opdrachten 6 resp. 14 gehoord als 7 resp. 15; we zeggen dan, dat de besturingsverplaatsing gehoorzaamd is. Is daarentegen de conditie - , dan werken de opdrachten 6 en 14 beide als de Skip en wij zeggen dan, dat de besturingsverplaatsing niet gehoorzaamd is. Het numerieke gedeelte n is bij deze op-

drachten dus slechts potentieel van invloed, omdat het niet gebruikt wordt, als de conditie - is.

III. Als f = 30 of 31 ontmoet de machine een z.g. onbestaanbare opdracht. Er gebeurt het een en ander - onzin n.l. - en de machine stopt, d.w.z. gaat niet meer in de eerste fase over. Wat precies gebeurt is oninteressant, voldoende is te weten, dat niet in het geheugen geschreven wordt. Het is niet de bedoeling, dat deze opdrachten gebruikt worden.

Het vierde onderdeel is de invoer, bestaande uit de bedieningspanelen en de bandlezer, met behulp waarvan de machine de noodzakelijke informatie van de buitenwereld kan betrekken. Er zijn twee bedieningspanelen, n.l. het schakelaarspaneel en het toetsenpaneel; op de rechterhelft van het toetsenpaneel bevindt zich het handregister, dat evenals de bandlezer, slechts werkt bij gratie van een permanent in het geheugen aanwezig programma, het z.g. Invoerprogramma(dat o.a. de in te voeren getallen van het 10-tallig naar het 2-tallig stelsel converteert). Aangezien op dit moment de machine beschreven wordt, beperken wij ons nu tot het schakelaarspaneel en de linkerhelft van het toetsenpaneel.

Het schakelaarspaneel bevat vier rijen schakelaars, die alle twee standen hebben: schakelaar naar beneden als 0, naar boven als 1 interpreterend, kan men in deze rijen binale woorden "inzetten".

De onderste rij, met het onderschrift "GETAL" bestaat uit de 30 getalschakelaars G. Het 30-cijfer-woord, wat hierin gezet kan worden ("de inhoud van G") kan door speciale communicatieopdrachten, al of niet met tekenwisseling, in A of in S, hetzij opgeteld, hetzij overgenomen worden (zie later). De drie andere rijen dragen de namen "BEGINADRES", "STOPADRES" en "OPDRACHT". De rij van het "BEGINADRES" staat links boven, de rij van het "STOPADRES" rechts boven op het schakelaarspaneel; beide bevatten zij 11 schakelaars. In het midden bevindt zich de rij "OPDRACHT", bestaande uit 15 schakelaars. Zij dienen slechts ter nadere specificatie van de operaties, die worden uitgevoerd onder controle van de linkerhelft van het toetsenpaneel, waarvan de beschrijving nu volgt.

De linkerhelft van het toetsenpaneel bevat tien toetsen en een schakelaar, gerangschikt in twee rijen van vier, waaronder een rij van drie. De ene schakelaar - de stopschakelaar - bevindt zich op de meest rechtse plaats van de middelste rij. Als de machine werkt, zijn alle toetsen van het toetsenpaneel, met uitzondering van de stopknop, geblokkeerd; deze blokkade is electronisch: men kan de toetsen wel indrukken, het heeft echter geen effect.

Tussen de twee onderste rijen bevindt zich het opschrift "GEKOZEN

OPDRACHT". Voor de middelste rij betekent dit, dat de drie toetsen links betrekking hebben op het "BEGINADRES" op het schakelaarspaneel, en dat de stopschakelaar, die rechts in de middelste rij zit, op het STOPADRES" op het schakelaarspaneel, betrekking heeft. Boven de middelste rij - en dus onder de vier toetsen van de bovenste rij - bevinden zich van links naar rechts de opschriften "BEGIN", "DOE", "LEES" en "STOP".

Bezien wij thans de middelste rij. Bij het indrukken van een van de drie toetsen, wordt om te beginnen het "BEGINADRES" in de opdrachtteller T overgenomen, en de machine start in de eerste fase. Na "LEES" stopt de machine na afloop van de eerste fase: de "gekozen opdracht" is van de trommel in het opdrachtregister R gelezen, en de opdrachtteller T is vervolgens opgehoogd. Aangezien de inhoud van R (evenals van T) in lichtjes op de console zichtbaar is, kan men visueel controleren, of een bepaalde opdracht goed in het geheugen staat. Na "DOE" stopt de machine niet na de eerste fase, maar na de tweede fase: de opdracht wordt dus tevens uitgevoerd. "DOE" laat in R R en T hetzelfde achter als "LEES" - met uitzondering van T in het geval van een gehoorzaamde of ongeconditioneerde sprongopdracht. - Na "BEGIN" stopt de machine - in eerste instantie althans - niet. "BEGIN GEKOZEN OPDRACHT" is de normaalste startknop van de machine. De schakelaar "STOP" uit de middelste rij heeft drie standen, ten eerste een neutrale stand, waarin hij "zichzelf uitschakelt", ten tweede een stand, die reageert op het "STOPADRES": in dit geval stopt de machine, als de opdracht, waarvan de plaats op de trommel door het "STOPADRES" is aangegeven, is aangehaald en uitgevoerd. In de derde stand reageert het stoppen niet op een plaats op de trommel (dus de inhoud van T) maar op een paar bepaalde opdrachten (dus op de inhoud van R). Dit zijn enkele speciale opdrachten uit de groep der "schuif- en communicatieopdrachten" die op karakteristieke punten in elk programma altijd gebruikt worden. Dit ter vereenvoudiging van de localisatie van fouten of ontsporingen in een programma (zie later). De bovenste rij toetsen draagt het gemeenschappelijk opschrift "VOLGENDE OPDRACHT". Drukt men in deze rij op de toets met onderschrift "STOP" - de stopknop - dan stopt de machine aan het einde van de tweede fase, d.w.z. na voltooiing van de opdracht, die zojuist uitgevoerd werd. De functie van de drie linkse toetsen is gelijk aan die van het drietal er onder, met dat verschil, dat in de bovenste rij het kopieren van het "BEGINADRES" in T achterwege blijft. "LEES VOLGENDE OPDRACHT" wordt gebruikt om visueel te controleren, of een stukje programma goed in het geheugen staat. "DOE VOLGENDE OPDRACHT" wordt gebruikt om een stukje programma opdracht voor opdracht door te nemen. Kan daarna de machine weer gewoon doorrekenen, dan drukt men op de toets "BEGIN VOLGENDE OPDRACHT" dit laatste heet "doorstarten". Men realisere zich, dat bij "LEES VOLGENDE OPDRACHT"

de machine nimmer in de tweede fase komt, en dus steeds de numeriek volgende opdracht leest, met veronachtzaming van alle sprongopdrachten.

De drie toetsen van de onderste rij laten bij indrukken - wat alleen gebeurt, als de machine gestopt is - de opdrachtteller onveranderd. Slechts de meest rechtse beinvloedt R: bij het indrukken van deze toets - die het onderschrift "LEES" heeft - wordt in R overgenomen de opdracht, die gevormd wordt door het 15-cijfer-woord in de rij "OPDRACHT" op het schakelaarspaneel. De andere twee toetsen, met de onderschriften "DOE" en "HERHAAL" laten de besturing de tweede fase doorlopen. Drukt men op de toets "DOE" dan wordt de opdracht in R uitgevoerd en de machine stopt vervolgens; drukt men op de toets "HERHAAL" dan wordt deze opdracht herhaald uitgevoerd, totdat men op de stopknop drukt, waarna de machine aan het einde van de tweede fase stopt. T blijft uiteraard ongewijzigd.

De oplettende lezer zal gemerkt hebben, dat bij de beschrijving van de bedieningspanelen reeds in kleinere details is getreden. Ten aanzien van het handregister en de bandlezer voorlopig het volgende.

Het handregister bestaat uit 14 toetsen, te weten de 10 cijfertoetsen voor de decimale cijfers van 0 t/m 9 en de 4 tekentoetsen met de opschriften +, -, +. en -. . Met behulp van het handregister kan men decimaal gegeven getallen in tweetallige representatie inbrengen volgens conventies, die door het invoerprogramma bepaald worden en later uiteengezet zullen worden. Het indrukken van een van de toetsen van het handregister, terwijl de machine werkt, evenals het indrukken van een cijfertoets, voordat een tekentoets is aangeslagen, heeft dank zij enige blokkade geen effect. Het indrukken van een van de toetsen van het handregister heeft een dubbel gevolg: ten eerste wordt in een aan het handregister gekoppeld register H een getal geplaatst, dat bepaald wordt door de ingedrukte toets: voor de cijfertoetsen is dit gelijk aan het toetssymbool, voor de tekentoetsen +, -, +. en -. resp. 10, 11, 12 en 13. Dit betekent dat het register H bestaat uit 4 variabele cijfers, aan de hoge kant ("de linkerkant") aangevuld met 26 vaste nullen. Het tweede gevolg van het indrukken van een cijfertoets is dat de machine start op adres 0,a : dit is het eerste adres van het invoerprogramma en hier staat een communicatieopdracht, die de inhoud van H aanhaalt en in A plaatst (zie later).

Verreweg het belangrijkste onderdeel van de invoer is echter de bandlezer. Hierin ligt een geponste telexband: op commando van daartoe bestemde communicatieopdrachten leest de bandlezer op 5 posities in de dwarsrichting van de band, of zich daar al of niet een gaatje bevindt. Een gat wordt als 1, ongeponste positie als 0 geinterpreteerd.

Dit variabele 5-cijfer-woord beslaat de 5 laagste posities van het register L, dat aan de hoge kant aangevuld is met 25 vaste nullen. De inhoud van L kan dan op verschillende manieren naar het rekenorgaan toegevoerd worden (zie later). Alle bandlezende communicatie-opdrachten worden gevolgd door opschuiven van de band, totdat de volgende rij gaatjes onder de bandlezer ligt (zie eveneens later).

Van de uitvoer - die groep onderdelen, die informatie aan de buitenwereld doorgeven - zijn onderdehand reeds enige facetten vermeld: strict genomen vallen n.l. de controlelichtjes ook onder de uitvoer. Op de console is in lichtjes - brandend lichtje = 1, niet brandend lichtje = 0 - de inhoud van R en T immers zichtbaar. Van de andere controlelichtjes worde hier slechts het conditielichtje genoemd, dat c zichtbaar maakt: als het conditielichtje brandt, is de conditie negatief.

De belangrijkste uitvoeren zijn echter de electrische schrijfmachine en de bandponser. Hierbij wordt het uitgangsregister U gebruikt, dat beschikt over 5 variabele plaatsen. Met behulp van speciale communicatieopdrachten - de typopdrachten en de ponsopdrachten kan U uit de registers A of S gevuld worden, waarna uit U getypt, resp. geponst wordt. Tevens kan - voor controledoeleinden - uit U weer gelezen worden. Bij de typopdrachten doet het $5^{e}$ cijfer van U niet mee: de typemachine reageert uitsluitend op de laagste 4 cijferplaatsen van U. De zestien typsignalen, die naar de schrijfmachine gestuurd kunnen worden, hangen als volgt met de inhoud van U samen: is deze inhoud 0, 1, 2, ....., 9 , dan wordt het cijfer 0, 1, 2, ..9 getypt; is de inhoud van U 10, 11, 12, 13, 14 resp. 15, dan is het typsignaal TAB., TWNR., -, +, . resp. spatie. Speciale typroutines verzorgen de deconversie van het 2-tallig naar het 10-tallig stelsel. Deze routines vormen een onderdeel van de permanente geheugeninhoud; men spreekt dus correcter van het "In- en Uitvoerprogramma".

Het belang van de bandponser ligt daarin, dat de machine informatie - in vrij ongelimiteerde hoeveelheden - op een medium kan vastleggen op een zodanige wijze, dat zij voor de machine weer toegankelijk is. Als de hoeveelheid tussenresultaten de geheugencapaciteit overschrijdt, is men op dit uitvoerorgaan aangewezen.

De interpretatie der woorden.

Wij hebben ingevoerd het begrip "woord" voor een rij tweetallige cijfers: het woord in een van de registers A, S, R, T, G, H, L en U en adres n duiden we aan, door de betroffen letter tussen haakjes te plaatsen, dus (A), (S), ..., (U) en voor woorden in het geheugen (n). Deze notatie heeft ten aanzien van de interpretatie dezelfde neutraliteit als het begrip woord; willen we tevens de interpretaie in de indicatie van het woord betrekken, dan plegen, althans voor de meest voorkomende gevallen, andere symbolen gebruikt te worden. Zo wordt het opdrachtenkoppel op adres n aangeduid met <n>. Ter precisering van een en ander is het niet ondienstig, de dertig cijfers individueel te onderscheiden: laat het woord gevormd worden door de cijfers

$$d_{29},\ d_{28},\ d_{27},\ \ldots\ldots,\ d_2,\ d_1,\ d_0.$$

In het opdrachtenkoppel staat de a-opdracht aan de lage ("rechter") kant van de b-opdracht, d.w.z: de a-opdracht beslaat de cijfers $d_{14}$ t/m $d_0$, de b-opdracht loopt van $d_{29}$ t/m $d_{15}$. In elke opdracht staan de vijf functiecijfers aan de hoge kant van het numerieke gedeelte; voor de a-opdracht dus van $d_{14}$ t/m $d_{10}$, voor de b-opdracht van $d_{29}$ t/m $d_{25}$. Conform de interpretatie <n>, wordt de inhoud van A ook met <A> aangeduid, indien b.v. een opdrachtenkoppel in het A-register is aangehaald ter wijziging, b.v. om het adres (= het numerieke gedeelte) van de a- of b-opdracht (of van beide) te veranderen (zie later).

Willen wij (n) speciaal als geheel getal interpreteren, dan noteren wij [n]. De numerieke waarde, die dan aan (n) toegekend wordt, is gegeven door de betrekking $d = \sum_{k=0}^{28} (d_k - d_{29}).2^k$.

We doen goed de twee gevallen $d_{29}= 0$ en $d_{29}= 1$ nauwkeurig te bekijken. Als $d_{29}=0$ luidt de formule dus $d = \sum_{k=0}^{28} d_k.2^k$. We zien hieruit, dat d dan een nonnegatief getal is, dat minimaal =0 is (alle $d_k=0$) en maximaal $=2^{29}-1 = 536870911$ (alle $d_k=1$, met uitzondering van $d_{29}$). De numerieke waarde is gelijk aan het getal, dat ontstaat, als men $d_{29}=0$ als + teken, de cijfers $d_{28}\ldots d_0$ als tweetallige cijfers opvat, $d_0$ als eenhedencijfer, $d_1$ als twee-tallen cijfer etc.: de index slaat dus op de macht van het grondtal 2. Als $d_{29}=1$ luidt de formule $d = \sum_{k=0}^{28} (d_k-1).2^k = -\sum_{k=0}^{28} (1-d_k).2^k$.

Bedenkend, dat de getallen $d_k$ uitsluitend de waarde 0 of 1 aan kunnen nemen, zien we, dat als $d_{29}= 1$ het getal d nonpositief is. Aangezien van $d_k$ en $1-d_k$ de ene =0, de andere =1 is, zien we, dat tekenwisseling overeenkomt met "wisseling van pariteit". Als d nonpositief is, is het maximaal =0 (alle $d_k=1$) en minimaal $=-(2^{29}-1)= -536870911$ (alle $d_k= 0$, met uitzondering van $d_{29}$).

Hieruit volgt, dat de machine twee representaties kent van het getal nul, ter onderscheiding +0 (alle $d_k=0$) en -0 (alle $d_k=1$) genoemd. (Waar in ongelijkheden een nul met teken voorkomt, wordt +0 geacht "groter" dan -0 te zijn: $d > +0$ betekent $d_{29} = 0$; $d > -0$ voegt -0 hieraan toe).

Een derde belangrijke interpretatie is "als breuk"; deze wordt met accolades aangegeven: $\{n\} = [n] \cdot 2^{-29}$. Hier wordt de komma gedacht tussen $d_{29}$ en $d_{28}$; in deze interpretatie is $|\{n\}| \leqslant 1 - 2^{-29}$. In formule $d = \sum_{k=0}^{28} (d_k - d_{29}).2^{k-29}$.

De conventie aangaande de interpretatie van negatieve getallen houdt ten nauwste verband met de wijze, waarop ARRA en FERTA de optelling en aftrekking uitvoeren. Ten eerste kan gezegd worden, dat de machine uitsluitend optelt: in geval van een aftrekking wordt n.l. het getal, dat van de trommel gelezen wordt, van teken gewisseld, wat geschiedt door het getal door een invertor te sturen, die nullen door enen en enen door nullen vervangt.

De opteller telt slechts op een cijferpositie tegelijkertijd op: eerst op $d_0$, dan op $d_1$, dan op $d_2$, enz. Als op $d_0$ de ene term een 0, de andere een 1 heeft, is het somcijfer **een** 1; als ze beide een 0 hebben is het somcijfer een 0. Hebben echter beide een 1, dan is de som 2 (in het tweetallig stelsel als 10 geschreven) wat betekent, dat het somcijfer een 0 is, maar dat op de volgende cijferpositie ($d_1$) een extra 1 opgeteld moet worden (het "1 **onthouden**"). Deze extra 1 noemt men de <u>overdracht</u> (= "carry over"). Dienovereenkomstig telt de opteller steeds 3 tweetallige cijfers op: twee van de op te tellen getallen en bovendien de eventuele overdracht, die op de vorige cijferplaats is ontstaan; de opteller levert 2 antwoorden: het somcijfer op dezelfde cijferpositie, dat instantaan wordt afgeleverd, en de overdracht, die tot de volgende cijferplaats onthouden wordt. De representatie van de negatieve getallen brengt echter met zich mede, dat de overdracht, die bij de laatste optelling in positie $d_{29}$ kan ontstaan, en aan de hoge kant "er uit loopt", aan de lage kant, dus in positie $d_0$, er bij opgeteld moet worden. We zullen deze rondlopende overdracht, de <u>kringoverdracht</u> ("end-around carry") illustreren aan een register van 5 binale plaatsen. Als de overdracht aan de hoge kant er gewoon uit zou lopen en we interpreteren voor het gemak even elke registerinhoud positief, dan kan dit register een van de getallen 00000 (0) t/m 11111 (31) bevatten. Tellen we bij 11111 (31) 00001 (1) op, dan wordt het antwoord, omdat de overdracht doorgegeven wordt en er uitloopt, 00000 (0): deze opteller werkt congruent modulo 32. Willen we nu van de registerinhoud b.v. 3 aftrekken, dan kunnen we dit doen door optelling van 32 - 3 = 29, in het binaal stelsel geschreven als 11101.

| | | |
|---|---|---|
| 12 | 01100 | 12 |
| -3 | 11101 | 29 |
| 9 | 01001 | 9 |

Aan de hoge kant is er een overdracht uitgelopen. De overgang van 3 naar 29 = 32 - 3 is tweetallig niet overdreven geriefelijk: voor deze aftrekking (in binale vorm) moet "geleend" worden. Veel gemakkelijker is het, om de 3 van de 31 af te trekken: immers 31 bestaat uit louter enen, er hoeft nooit geleend te worden: de overgang is $d_k \rightarrow 1 - d_k$, de inversie! Echter tellen we dan, in plaats van 32 - 3 er 31 - 3, dus 1 te weinig bij op: dit wordt hersteld, door de overdracht, die aan de hoge kant ontstaat, er aan de lage kant bij op te tellen. Omdat 11111 dan gelijk is aan (-)0 - immers optellen van 11111 heeft geen effect - werkt deze opteller congruent modulo $31 = 2^5 - 1$ . Dat het in de lijn ligt, om getallen boven de 16 als negatieve getallen te interpreteren, ligt voor de hand: men trekke er 31 van af. Het tekencijfer doet dus in de optelling gewoon mee, de opteller is volslagen cyclisch!

Het bewijs, dat de optelling in ARRA en FERTA dankzij de kringoverdracht het antwoord congruent modulo $2^{30} - 1$ aflevert, kan als volgt geleverd worden.

Ten eerste realisere men zich, dat twee "treinen" van secundaire overdrachten elkaar nooit in kunnen halen, waarmee het volgende bedoeld is: een secundaire overdracht is een overdracht, die ontstaat door de overdracht, die aan de lage kant is afgeleverd, in tegenstelling tot de z.g. primaire overdracht, die op een bepaalde cijferplaats ontstaat, ongeacht de overdracht die van rechts binnenkomt; de laatste ontstaat dus, als beide cijfers in de termen der som een 1 waren, en wordt doorgegeven - er ontstaat een trein van secundaire overdrachten - zolang de primaire som op de volgende cijferplaatsen een 1 is. Opdat echter later een primaire overdracht ontstaat, moeten twee enen opgeteld zijn: deze hebben als primair somcijfer een nul afgeleverd: hier kan de secundaire overdrachtentrein niet voorbij komen. Dit rechtvaardigt het procede van de opteller, die steeds drie tweetallige cijfers kan hanteren, waarvan een de overdracht is; maar doordat de overdrachtentreinen niet kunnen samenvallen, is de overdracht nooit groter dan 1, en is dus inderdaad 1 overdrachtcijfer voldoende. Tevens is de kringoverdracht maximaal 1. Formuleren wij de kringoverdracht zo, dat de overdracht, ontstaan in de positie $d_{29}$, optellen in $d_0$, in plaats van op de - hypothetische - $d_{30}$, dan impliceert dat de kringoverdracht 1 in effect neerkomt op vermeerdering met $1 = 2^0$ en vermindering met $2^{30}$, totaal een vermindering van $2^{30} - 1$.

Zij $f = e+d$, zij $d = \sum_{k=0}^{28} (d_k - d_{29}).2^k = \sum_{k=0}^{29} (d_k - d_{29}).2^k$ en $e = \sum_{k=0}^{29} (e_k - e_{29}).2^k$. Dan is de algebraische som f gelijk aan:

$$f = \sum_{k=0}^{29} (d_k + e_k - d_{29} - e_{29}).2^k = \sum_{k=0}^{29} (d_k + e_k)2^k - (d_{29} + e_{29}) \sum_{k=0}^{29} 2^k$$

$$= \sum_{k=0}^{29} (d_k + e_k).2^k - (d_{29} + e_{29})(2^{30} - 1)$$

$\sum_{k=0}^{29} (d_k + e_k).2^k$ is de som van de twee 30-cijferwoorden, indien $d_{29}$ en $e_{29}$ normaal op de macht $2^{29}$ worden betrokken. Maar deze som vormen ARRA en FERIA, onder voorbehoud van een eventuele overdracht naar $d_{30}$: deze resulteert in een vermindering met $2^{30} - 1$. De tweede term uit de algebraische som is eveneens een veelvoud van $2^{30} - 1$, waarmee het gestelde bewezen is.

En passant volgt hieruit, dat in een som van vele termen een tussenresultaat best de grenzen $\pm (2^{29} - 1)$ mag overschrijden: als bekend is, dat het eindantwoord in de capaciteit kan, wordt het correcte antwoord afgeleverd.

Hoewel mathematisch gezien de kringoverdracht een volslagen cyclische symmetrie in de 30 cijferposities bewerkstelligt, is dit technisch niet het geval. In de machine zit n.l. maar één opteller voor de drie tweetallige cijfers: deze schuift niet van de lage kant naar de hoge langs het register, maar het register schuift van de hoge naar de lage kant rond: de cijfers, die aan de lage kant het register verlaten, worden er via de opteller aan de hoge kant weer ingestuurd. Tegelijkertijd komen de successievelijke cijfers van het addendum (eveneens $d_0$ voorop) van de trommel: de som wordt "van links" het register "ingepompt". Na afloop blijft de kringoverdracht over! Omdat deze op $d_0$ een secundaire overdrachtenterm lanceren kan, zou voor de verrekening van de kringoverdracht het register nog een keer helemaal rond moeten gaan. Dit is echter overbodig, indien men aan A en S een overdrachtsregister toevoegt (OA, resp. OS), elk bestaand uit 1 cijfer. De kringoverdracht wordt tot de volgende operatie onthouden: is dit weer een optelling, dan wordt de onthouden kringoverdracht van de vorige keer bij de vorming van het "nulde" somcijfer in rekening gebracht. Schrijven naar het geheugen geschiedt eveneens via de opteller: na een schrijfopdracht uit A of S is de (OA) resp. (OS) in rekening gebracht, (en dan dus = 0). Adressen in het geheugen bestaan immer uit 30 cijfers, de registers A en S eigenlijk uit 31, met een individuele inhoud, die niet eenduidig bepaald is door de "orithmetische" inhoud. Vermenigvuldiging en deling beginnen met de registers A en S een keer rond te sturen, om de kringoverdrachten effectief te doen zijn. De programmeur hoeft dit echter allemaal slechts te weten in verband met de schuifopdrachten: deze kunnen zich op deze wijzen gedragen t.o.v. de kringoverdrachten:

0 effectief: de overdracht wordt in rekening gebracht, en gelijk aan nul, als de overdrachtentrein stopt.

O kapot : de overdracht wordt niet in rekening gebracht, en door nul vervangen.

O intact: de overdracht wordt niet in rekening gebracht, blijft onveranderd staan en kan zich in de volgende opdracht doen gelden.

Na de deling OA = O en OS = O.
Na de vermenigvuldiging is OS = O; OA = O, behalve als bij een **additieve** vermenigvuldiging, **terwijl** (A) van te voren negatief was, na afloop een positief resultaat overblijft, dat in (S) louter nullen achter laat.

Voor optellingen (en daarmede aftrekkingen) kan men het volgende staatje controleren. Als (n) bij (A) wordt opgeteld geldt:

| | | | | |
|---|---|---|---|---|
| $(A)\gtrless+0$ | $(n)\gtrless+0$ | $(A)+(n)\gtrless+0$ | OA=O | |
| $(A)\geqslant+0$ | $(n)\geqslant+0$ | $(A)+(n)\leqslant-0$ | OA=O | (capaciteitsoverschrijding) |
| $(A)\geqslant+0$ | $(n)\leqslant-0$ | $(A)+(n)>0$ | OA=1 | |
| $(A)\geqslant+0$ | $(n)\leqslant-0$ | $(A)+(n)\leqslant-0$ | OA=0 | |
| $(A)\leqslant-0$ | $(n)\geqslant+0$ | $(A)+(n)>0$ | OA=1 | |
| $(A)\leqslant-0$ | $(n)\geqslant+0$ | $(A)+(n)\leqslant-0$ | OA=0 | |
| $(A)\leqslant-0$ | $(n)\leqslant-0$ | $(A)+(n)>0$ | OA=1 | (capaciteitsoverschrijding) |
| $(A)\leqslant-0$ | $(n)\leqslant-0$ | $(A)+(n)\leqslant-0$ | OA=1 | |

Opm.1 In de derde kolom is meestal $>0$ en niet $\geqslant+0$ gezet: additieve bewerkingen, waar exact nul uitkomt laten n.l. -0 in het register achter, met uitzondering van (A)= +0, (n)= +0, die het antwoord +0 levert. Vandaar, dat in de bovenste regel $\gtrless+0$ is gezet.

Opm. 2 Men **kan** de regels samenvatten:
OA=1, als of beide addenda $\leqslant-0$, of de addenda van **verschillend** teken, maar **zodanig**, dat hun som positief is.
OA=0, als of beide addenda $\geqslant+0$, of de addenda van verschillend teken, maar zodanig, dat hun som non-positief is.

Opm.3 Voor S gelden dezelfde regels.

Wanneer A en S hetzelfde teken hebben, is er nog een belangrijk begrip, n.l. (AS), n.l. het getal het de twee identieke tekencijfers als tekencijfer gevolgd door de 58 gewone cijfers uit A en S. Hierbij wordt (A) aan de meest significante zijde van (S) gedacht, het tekencijfer (S), slechts een herhaling van het tekencijfer (A), wordt overgeslagen. Al naar gelang men de komma aan de lage kant van S, tussen A en S of aan de hoge kant van A denkt, krijgt men de volgende drie notaties en interpretaties:

$[AS] = [A] \cdot 2^{29} + [S]$ "een lang geheel getal"
$[AS\} = [A] + \{S\}$ "een lang gemengd getal"
$\{AS\} = \{A\} + \{S\} \cdot 2^{-29}$ "een lange breuk".

De opdrachten.

Thans volgt de volledige lijst der opdrachten. Alleen de registers of adressen die gewijzigd worden, worden vermeld; de gewijzigde inhoud wordt met een accent aangegeven. (T)' echter is alleen aangegeven, als niet (T)'= (T) + ½.

Een bepaald cijfer uit het register wordt door een index rechts onderaan aangegeven: b.v. $(A)_{29}$ is het tekencijfer van A. De inversie van een cijfer is door aftrekking van de 1 aangegeven. De vijf functiecijfers en de 10 van het adres worden (voorlopig) door een komma gescheiden.

Eerst worden de opdrachten met f=0 t/m 23 beschreven.

| | | | |
|---|---|---|---|
| 0,n | (A)'= (A)+(n) | | |
| 1,n | (A)'= (A)-(n) | | |
| 2,n | (A)'= (n) | | |
| 3,n | (A)'=-(n) | | |
| 4,n | (n)'= (A) | (C)'= $(A)_{29}'$ | |
| 5,n | (n)'=-(A) | (C)'= 1-$(A)_{29}'$ | |
| 6,n | | | als (C)= 0, dan (T)'= n |
| 7,n | | | (T)'= n |
| 8,n | (S)'= (S)+(n) | | |
| 9,n | (S)'= (S)-(n) | | |
| 10,n | (S)'= (n) | | |
| 11,n | (S)'=-(n) | | |
| 12,n | (n)'= (S) | (C)'= $(S)_{29}'$ | |
| 13,n | (n)'=-(S) | (C)'= 1-$(S)_{29}'$ | |
| 14,n | | | als (C)= 0, dan (T)'= n+½ |
| 15,n | | | (T)'= n+½ |
| 16,n | [AS]'= [n].[S]+[A] | | |
| 17,n | [AS]'=-[n].[S]+[A] | | |
| 18,n | [AS]'= [n].[S] | | |
| 19,n | [AS]'=-[n].[S] | | |
| 20,n | [n].[S]'+[A]'= [AS] | | |
| 21,n | -[n].[S]'+[A]'= [AS] | | |
| 22,0 | | als (C)=0, stop | |
| 23 0 | | als (C)=1, stop | |

Thans slechts tot zover. Enig commentaar ter verduidelijking en complementering van de korte omschrijving, volge.

0,n. +Additief in A.

(A)'= (A)+(n) is een verkorte schrijfwijze voor [A]'= [A]+[n] of {A}'= {A}+{n}, m.a.w: in beide termen en de som dient men de komma op dezelfde plaats te interpreteren.
OA effectief, OS intact. Alleen als (n)= (A)= +0, is (A)'= +0.

1,n -Additief in A.
OA effectief OS intact. Alleen als (n)= -0 en (A)= +0 is (A)'= +0.

2,n + Schoon in A
OA kapot, OS intact. Alleen als (n)= +0 is (A)'= +0.

3,n - Schoon in A.
OA kapot, OS intact. Alleen als (n)= -0 is (A)'= +0

4,n + Vuil uit A.
OA effectief, OS intact. De conditie wordt ingesteld op het teken van (A)'. (Als men -0 optelt bij $-(2^{29}-1)$ is aanvankelijk $(A)_{29}= 0$; OA = 1, de kringoverdracht loopt echter door tot in het tekencijfer van (A), dat weer een 1 wordt: zo wordt het getal naar het geheugen gestuurd en in A achtergelaten. "Vuil" slaat er op, dat, in tegenstelling tot "Schoon" de inhoud van A niet vernietigd wordt. Integendeel!)

5,n - Vuil uit A.
OA effectief, OS intact. De conditie wordt ingesteld op het omgekeerde teken van (A)'.

6,n Conditionele sprong naar de a-plaats.
OA en OS intact. Als de conditie positief is, is de volgende opdracht die op n a; anders fungeert de opdracht als Skip.

7,n Ongeconditioneerde sprong naar de a-plaats.
OA en OS intact. De volgende opdracht is die op adres n a, ongeacht de conditie, die intact blijft.

8,n + Additief in S.
OA intact OS effectief. Alleen als (n) = (S) = +0, is (S)'=+0.

9,n - Additief in S.
OA intact OS effectief. Alleen als (n)= -0 en (S) = +0, is (S)'= +0.

10,n + Schoon in S.
OA intact, OS kapot. Alleen als (n)= +0, is (S)= +0.

11,n - Schoon in S.
OA intact, OS kapot. Alleen als (n)= -0 in (S)'= +0.

12,n + Vuil uit S.
OA intact, OS effectief. De conditie wordt ingesteld op het teken van (S)'.

13,n - Vuil uit S.
OA intact, OS effectief. De conditie wordt ingesteld op het omgekeerde teken van (S)'.

14,n Conditionele sprong naar de b-plaats.
OA en OS intact. Als de conditie positief is, is de volgende opdracht die op n b anders fungeert de opdracht als Skip.

15,n Ongeconditioneerde sprong naar de b-plaats.
OA en OS intact. De volgende opdracht is die op n b, ongeacht de conditie, die intact blijft.

16,n + Additieve vermenigvuldiging.
Bij de vermenigvuldiging wordt (S) als vermenigvuldiger, (n) als vermenigvuldigtal opgevat: de successievelijke cijfers van (S), te beginnen bij $d_0$, worden "afgetast" en afhankelijk hiervan wordt op de betroffen positie (n) al of niet opgeteld. De optelling geschiedt steeds in (A), waarna het partiele product in A en S een plaats naar rechts schuift. De vermenigvuldiger in (S) schuift mee - aan de hoge kant voor het opkomende product ruimte makend - zodat de te testen cijfers van de vermenigvuldiger in successie aan de lage kant S verlaten. Een en ander brengt met zich mee, dat de aanvankelijke inhoud van A juist naar S toeschuift. Door linker- en rechterlid van de omschrijving van de vermenigvuldiging al of niet met 1 of 2 factoren $2^{-29}$ te vermenigvuldigen verkrijgt men de volgende omschrijvingen:

Als (A), (S) en (n) als gehele getallen worden beschouwd, is

$$[AS]' = [n].[S]+[A]$$

Als (A) en (S) als gehele getallen, (n) als breuk wordt beschouwd, is

$$[AS\}' = \{n\}.[S]+[A].2^{-29}$$

Als (A) en (n) als gehele getallen, (S) als breuk wordt beschouwd, is

$$[AS\}' = [n].\{S\}+[A].2^{-29}$$

Als (S) en (n) als gehele getallen, (A) als breuk wordt beschouwd, is

$$[AS]' = [n].[S]+\{A\}.2^{+29}$$

Als (A) als geheel getal, (S) en (n) als breuken worden beschouwd, is

$$\{AS\}' = \{n\}.\{S\}+[A].2^{-58}$$

Als (S) als geheel getal, (A) en (n) als breuken worden beschouwd, is

$$[AS]' = \{n\}.[S]+\{A\}$$

Als (n) als geheel getal, (A) en (S) als breuken worden beschouwd, is

$$[AS]' = [n].\{S\}+\{A\}$$

Als (A), (S) en (n) als breuken worden beschouwd, is

$$\{AS\} = \{n\}.\{S\}+\{A\}.2^{-29}$$

Aan de tekens van (A), (n) en (S) zijn geen restricties opgelegd; de tekens van (A)' en (S)' zijn gelijk.
De enige mogelijkheid, dat (A)' = (S)' = +0, is, wanneer (A)=+0 en het product (n).(S) = +0, met inbegrip van teken, d.w.z. minstens een van beide factoren = $\pm$0, en beide factoren hebben hetzelfde teken. Is het antwoord $\rangle$ 0, doch is $[AS]$ resp. $\langle 2^{29}$ of een $2^{29}$-voud, dan zijn (A) resp. (S) = +0.
Steeds is (OS)'=0; slechts is (OA)' = 1 in het speciale geval, dat (A) $\leqslant$ -0, (AS)' $\rangle$ 0 en (S)' = +0; is niet tegelijkertijd aan deze drie voorwaarden voldaan, dan is (OA)'=0.

17,n -Additieve vermenigvuldiging.

De preciese interpretatie vindt men door in de lijst bij 16,n slechts -(n) i.p.v. (n) te lezen.
In overeenstemming hiermede vormt 17,n slechts dan (A)'=(S)' = = +0, als (A)= +0 en -(n).(S) = +0, d.w.z.: minstens een van beide factoren = $\pm$ 0 en de twee getallen (n) en (S) hebben verschillend teken.
Steeds is (OS)' = 0; slechts is (OA)' = 1 in het speciale geval, dat (A) $\leqslant$ -0, (AS)' $\rangle$ 0 en (S)' = +0; is niet tegelijkertijd aan deze drie voorwaarden voldaan, dan is (OA)' = 0.

18,n + Schone vermenigvuldiging.

De interpretaties luiden:

$$[AS]' = [n].[S]$$
$$[AS]' = \{n\}.[S]$$
$$[AS]' = [n].\{S\}$$
$$\{AS\}' = \{n\}.\{S\}$$

De oorspronkelijke registerinhoud (A) wordt geheel vernietigd; er zijn geen tekenrestricties voor (S) en (n). (A)' en (S)' hebben het juiste teken van het product (d.w.z: $(A)'_{29}=(S)'_{29}=$ $= 1 - (S)_{29}.(n)_{29} - (1 - (S)_{29})(1 - (n)_{29})$. (A)' = (S)' = +0 ontstaat dus, als van (n) en (S) minstens een = $\pm$0, terwijl beide hetzelfde teken hebben.
(OA)' = (OS)' = 0.

19,n - Schone vermenigvuldiging.
De precieze interpretaties vindt men, door in de lijst bij 18,n slechts -(n) i.p.v. (n) te lezen.

20,n + Deling.
Alleen dan ontstaat een zinvol resultaat, als aan de volgende twee voorwaarden voldaan is.
1e. $(A)_{29} = (S)_{29}$, dus van te voren moeten (A) en (S) hetzelfde teken hebben (Anders is ook het symbool (AS) niet van toepassing!).
2e. $|[n]| > |[A]|$ (of, wat hetzelfde is: $|\{n\}| > |\{A\}|$.)
De deling geschiedt op het lange getal (AS), het quotient q verschijnt in S, de rest r in A, waarbij de deling zo gedefinieerd is, dat r de in absolute waarde kleinst mogelijke rest is met het teken van het deeltal (AS). De precieze interpretatie luidt dus
$[AS] = q[n] + r$, teken $[AS]$ = teken r; $|r| < |[n]|$, $[A]' = r$, $[S]' = q$.
In de verschillende representaties heeft men:

$$[AS] = [n].[S]' + [A]'$$
$$[AS] = \{n\}.[S]' + \{A\}' = [n].\{S\}' + \{A\}'$$
$$\{AS\}' = \{n\}.\{S\}' + \{A\}'.2^{-29}$$

Slechts als $(AS) \geqslant +0$ en de deling opgaat, is $(A)' = +0$. Slechts als $|(AS)| < |(n)|$ en (AS) en (n) hetzelfde teken hebben, is $(S)' = +0$.

21,n - Deling.
Voor de beschrijving van de - deling vervange men in de explicatie bij 20,n , (n) door -(n).

22,0 + Conditionele stop.
Het numeriek gedeelte verwijst hier niet naar een adres, is onveranderlijk =0. Als de conditie + is, fungeert de opdracht als Stop, anders als Skip.

23,0 - Conditionele stop.
Als de conditie - is, fungeert de opdracht als Stop, anders als Skip.

De opdrachten f=24 t/m 29 worden in twee groepen besproken. Nu wordt de conventie ingevoerd, dat de 10 cijfers van het numerieke gedeelte in twee groepen van 5 gesplitst worden, elk een getal van 0 t/m 31 vormend. De opdracht wordt dus in het 32-talligstelsel geschreven: f v r. De drie "32-tallige cijfers" f v r worden door spaties gescheiden en elk door hun decimaal equivalent weergegeven: de skip opdracht wordt dus geschreven als 24 17 0 . De 17 staat op de plaats

van het z.g. velnummer , de 0 op de plaats van het regelnummer. (Dit met het oog op de indeling van het geheugen: ook waar het numeriek gedeelte niet naar een adres op de trommel verwijst, houden we deze nomenclatuur aan).

In de opdrachten f=24 t/m 29 is het numeriek gedeelte niet meer een adres op de trommel; dit specificeert de opdracht nu op andere wijze, voor v=0 t/m 17 in onderstaand overzicht weergegeven.

| Functiecijfers→ v | r ↓ | 24 | 25 | 26 | 27 | 28 | 29 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | (A)'=(A)+0 | (A)'=(A)-0 | (A)'=+0 | (A)'=-0 | (C)'=(A)'$_{29}$ | (C)'=1-(A)'$_{29}$ |
| 0 | 1 | (A)'=(A)+(H) | (A)'=(A)-(H) | (A)'=(H) | (A)'=-(H) | " | " |
| 0 | 2 | (A)'=(A)+(G) | (A)'=(A)-(G) | (A)'=(G) | (A)'=-(G) | " | " |
| 0 | 4 | (A)'=(A)+("T") | (A)'=(A)-("T") | (A)'=("T") | (A)'=-("T") | " | " |
| 0 | 8 | (A)'=(A)+(U) | (A)'=(A)-(U) | (A)'=(U) | (A)'=-(U) | (U)'=(A)<br>(C)'=(A)'$_{29}$ | (U)'=-(A)<br>(C)'=1-(A)'$_{29}$ |
| 0 | 16 | (A)'=(A)+0 | (A)'=(A)-0 | (A)'=+0 | (A)'=-0 | (A)≠-0? | (A)≠+0? |
| "4 | n" (n=0,1,...,127) | [A]'=[A]+n | [A]'=[A]-n | [A]'=+n | [A]'=-n | (C)'=(A)'$_{29}$ | (C)'=1-(A)'$_{29}$ |
| 8 | 0 | (S)'=(S)+0 | (S)'=(S)-0 | (S)'=+0 | (S)'=-0 | (C)'=(S)'$_{29}$ | (C)'=1-(S)'$_{29}$ |
| 8 | 1 | (S)'=(S)+(H) | (S)'=(S)-(H) | (S)'=(H) | (S)'=-(H) | " | " |
| 8 | 2 | (S)'=(S)+(G) | (S)'=(S)-(G) | (S)'=(G) | (S)'=-(G) | " | " |
| 8 | 4 | (S)'=(S)+("T") | (S)'=(S)-("T") | (S)'=("T") | (S)'=-("T") | " | " |
| 8 | 8 | (S)'=(S)+(U) | (S)'=(S)-(U) | (S)'=(U) | (S)'=-(U) | (U)'=(S)<br>(C)'=(S)'$_{29}$ | (U)'=-(S)<br>(C)'=1-(S)'$_{29}$ |
| 8 | 16 | (S)'=(S)+0 | (S)'=(S)-0 | (S)'=+0 | (S)'=-0 | (S) ≠ -0? | (S) ≠ +0? |
| "12 | n" (n=0,1,...,127) | [S]'=[S]+n | [S]'=[S]-n | [S]'=+n | [S]'=-n | (C)'=(S)'$_{29}$ | (C)'=1-(S)'$_{29}$ |
| 16 | n | [AS]'=[A]+n[S] | [AS]'=[A]-n[S] | [AS]=n[S] | [AS]=-n[S] | | |
| 17 | 0 | Skip<br>OA en OS intact | Skip<br>OA en OS intact | Skip<br>OA en OS intact | Skip<br>OA en OS intact | Skip<br>OA en OS intact | Skip<br>OA en OS intact |

Ter explicatie en voltooiing wordt deze lijst wederom doorgenomen.

v r = 0 0 Register A rond.

Als f=24 wordt +0 bij (A) opgeteld. Dit is tevens te omschrijven als OA eff. en na afloop =0 (OS uiteraard int.).
Als f=25 wordt -0 bij (A) opgeteld. Als (A)= +0, is (A)'= -0, in die representatie, waarin OA = 0. In alle andere gevallen wordt (A) zo "herschreven", dat OA = 1. f=24 en f=25 laten de arithmetrische inhoud van (A) ongewijzigd.
Als f=26 of 27 wordt OA = 0. Is f=26, dan is (A)'=+0, is f=27, dan is (A)'=-0.
Als f=28 of f=29, dan gaat het register A rond, en tevens wordt de conditie gezet. Dit impliceert, dat OA effectief is, om hieraan te herinneren is in de laatste kolom $(A)'_{29}$ gezet, met een accent, hoewel de arithmetische inhoud niet veranderd is - een verplichting, die ontstaat, doordat we naar een bepaald cijfer refereren, door de arithmetische inhoud immers niet eenduidig bepaald. De conditie wordt gezet op het teken, resp. het omgekeerde teken van (A)'. De omschrijving had ook kunnen luiden: 28 00 (OA eff.) (A) $\geqslant$ +0?
29 00 ( " ) (A) $\leqslant$ -0? , met de afspraak, dat bevestigend antwoord overeenkomt met conditie +.

v r = 0 1 Handregister.

Dit zijn de opdrachten, die uit het handregister - rechterhelft van het toetsenpaneel - lezen. H wordt aan de hoge kant aangevuld gedacht door 26 nullen. Als b.v. f=27 komen aan de hoge kant 26 enen in A. Ontmoet de besturing een van deze opdrachten, zonder dat de machine door indrukken van een van de toetsen van het handregister gestart is, dan wordt de inhoud [H] = 15 gelezen. Buiten het invoerprogramma zullen deze opdrachten niet gebruikt worden. Als f=28 of 29 is de functie als bij v r = 0 0 .

v r = 0 2 Getalschakelaars.

Als f=24, 25, 26 of 27 wordt + add., - add., + schoon, - schoon in A gelezen uit de 30 getalschakelaars G van het schakelaarspaneel.
Als f=28 of 29 wordt uitsluitend de conditie gezet.

v r = 0 4 Vorming van koppelopdracht.

("T") wordt bepaald door de inhoud van de opdrachtteller T. Als de f 0 4 opdracht stond op adres v r a, dan is < "T" > = 0 0 0 7 v (r+1); als de f 0 4 opdracht stond op adres v r b, dan is < "T" > =
= 0 0 0 15 v (r+1); in de a-positie van A wordt dus door f 0 4 op-

drachten ingelezen de sprongopdracht, die 2 opdrachten verder verwijst: de koppelopdracht voor subroutines (zie later). 26 0 4 is bij verre de belangrijkste van deze groep. De "Uit-versies" zetten weer de conditie.

v r = 0 8 Typopdracht.

Als f=28 of 29 wordt er een symbool getypt door de electrische schrijfmachine. Dit symbool wordt bepaald door de laagste 4 cijfers van A: de schrijfmachine is tot typen van 16 symbolen in staat. Dit gaat echter via register U, dat vijf variabele plaatsen heeft: de laagste 5 cijfers van A worden door 28 0 8 zonder inversie, door 29 0 8 met inversie overgenomen: de laagste 4 in U bepalen het getypte symbool. Als men echter met 24 t/m 27 0 8 uit U terug leest, dan worden alle 5 variabele plaatsen van U ingelezen, aan de hoge kant met 25 nullen aangevuld. (In geval van 25 0 8 of 27 0 8 zijn deze nullen in de tekenwisseling betrokken, en komen dus 25 enen het rekenorgaan binnen). Als b.v. $[A] = +15$ dan typt 28 0 8 een spatie, evenals bij $[A] = -0$. Na het eerste geval leest 26 0 8 $[A]' = 15$, na het tweede $[A]' = 31$ terug. f=28 en 29 stellen de conditie weer zonder, resp. met inversie in op het tekencijfer van $(A)'$.

v r = 0 16 Nultest op A.

Als f=24 t/m 27 is de functie gelijk aan f 0 0 voor f=24 t/m 27. Aan de gevallen f=28 en f=29 ontlenen deze opdrachten hun naam. 28 0 16 zet de conditie +, als $(A) \neq -0$, alleen als alle cijfers van (A) gelijk zijn aan 1, is de conditie -, in formule $(C)' = \prod_{k=0}^{29} (A)_k$. 29 0 16 zet de conditie +, als $(A) \neq +0$, alleen als alle cijfers van (A) gelijk zijn aan 0, is de conditie -, in formule $(C)' = \prod_{k=0}^{29} (1-(A)_k)$.

In een opzicht verschillen de eerste vier f 0 16 opdrachten van de overeenkomstige f 0 0 opdrachten: als de besturing een van de zes f 0 16 opdrachten ontmoet, stopt de machine na beeindiging van de opdracht, als de stopschakelaar op het toetsenpaneel in de derde stand staat. Dit is ingebouwd met het oog op de volgende tactiek: een programma wordt ten alle tijde ingedeeld in z.g. moten, d.w.z. stukken berekening die enige tussenresultaten in het geheugen afleveren, de voor deze berekeningen noodzakelijke informatie - in principe althans! - onaangetast laten, en de resultaten controleren. Deze controle wordt als regel herleid op een test op nul. Wil men een programma controleren, dan zet men de stopschakelaar in de derde stand: de machine stopt bij de nultest aan het einde van elke moot: aan het conditielichtje kan men zien, of de berekening in deze moot door de controle van het programma geaccepteerd wordt. Een en ander om het opsporen van fouten in een hetzij nieuw, hetzij ontspoord programma te vergemakkelijken.

v r = "4 n" Snelle ± n in A.

"4 n" is tussen aanhalingstekens gezet, omdat, n loopt van 0 t/m 127: voor elk 32-voud in n begrepen, krijgt men een overdracht in het velnummer. v r is minimaal = 4 0 (n=0), maximaal = 7 31 (n=127).

f=24 en 25 stellen ons instaat snel kleine vaste getallen in A op te tellen of af te trekken (± additief in A). Bij administratieve handelingen, speciaal ophogingen en tellingen, zijn deze opdrachten van grote waarde.

f=26 en 27 lezen deze kleine getallen pos., resp. neg. schoon in A.

f=28 en 29 zetten weer de conditie.

v r = 8 r en "12 n".

Met het verschil, dat deze opdrachten op (S) opereren, en niet op (A), is hun functie volledig analoog aan die, waarvan het velnummer 8 kleiner is.

v r = 16 n Snelle vermenigvuldiging.

n kan bij deze opdrachten lopen van 0 t/m 15.
Als f=24, 25, 26, 27, dan is de functie van f 16 n geheel gelijk aan die van f v r met resp. f=16, 17, 18, 19, als vr = n. Zij hebben het voordeel, sneller te zijn.

Opm. 1. 27 16 1 Wisselt S van teken, A wordt nul met hetzelfde teken.

Opm. 2. 24 16 1 telt in (A) bij (S) op. Eventuele overloop daarentegen vindt men op de eenhedenplaats van A! Opererend op z.g. lange getallen heeft 24 16 1 de functie van de "half adder". f=28 of 29 resulteert in dit geval in niet te omlijnen onzin.

v r = 17 0.

De opdrachten zijn de Skip-opdrachten. De tweede fase bestaat slechts uit het onmiddellijk weer teruggaan naar de eerste fase, waarin de volgende opdracht gehaald wordt.

In de opdrachten f=24 t/m 29 en v=0 t/m 15 schuift elk register 30 plaatsen naar rechts: als f=24, 25, 28 of 29, worden de cijfers, die aan de lage kant het register verlaten, via de opteller aan de hoge kant weer het register ingestuurd. Als f=28 of 29 is in de opteller slechts de overdracht in rekening gebracht, als f=24 of 25, wordt in de opteller behalve de kringoverdracht het addendum er bij opgeteld. Bij f=26 en 27 gaat de overdracht kapot - wordt door nul vervangen - het substituut wordt aan de hoge kant het register ingestuurd. Dit biedt de mogelijkheid tot diverse uitbreidingen.

Ten eerste kunnen registers minder dan de volle 30 plaatsen rond-

geschoven worden. Dit bestaat weer in verschillende versies.

In de additieve versie verlaten dus enige cijfers aan de lage kant het register, en komen via de opteller aan de hoge kant weer binnen. De kringoverdracht is hier wel effectief, maar dit impliceert nu niet meer, dat na afloop de kringoverdracht =0 is: de optelling is "halverwege blijven hangen". Wat als addendum fungeert, hangt af van f: als f=24 worden in de opteller nullen bij geteld - en zal meestal na afloop OA = 0; de optelling moet net een te lange secundaire overdrachtentrein ten gevolge hebben, om OA = 1 over te houden - als f=25 worden enen bij geteld: zodra aan de lage kant maar een 1 het register verlaten heeft, of OA oorspronkelijk =1 was, is dus na afloop OA = 1. Met de additieve schuiven draait men een register n plaatsen rond, bij de "- additieve schuif" worden op de n hoogste plaatsen enen bij geteld (vermindering met $2^{30-n}-1$).

Bij de schone schuiven (f=26 en 27) komt de opteller niet in werking: OA wordt vernietigd (=0 gemaakt) en niet in rekening gebracht. De registerinhoud wordt n plaatsen naar de minst significante plaats geschoven. Wat aldaar het register verlaat, gaat verloren, wat aan de hoge kant aangevuld wordt, hangt af van f en het oorspronkelijke tekencijfer. Als f=26 (+ schone schuif) wordt het register met het tekencijfer aangevuld; als f=27 is (-schone schuif) wordt het register met de inverse van het momentane (alternerende) tekencijfer aangevuld.

De derde mogelijkheid ligt in de "uit-versies" f=28 (+ uit) en f=29 (- uit). De functie van f=28 is gelijk aan f=24, de + add. schuif, bovendien wordt de conditie gezet: de conditie wordt gelijk aan het nieuwe tekencijfer: is dus na afloop de registerinhoud pos., dan is (C)'=0, anders (C)'=1. Hier moet bij gezegd worden, dat als men nul plaatsen schuift, onveranderlijk (C)=1 gezet wordt: er is dan geen <u>nieuw</u> tekencijfer: wat zich dan na afloop op de plaats van het tekencijfer bevindt, is het register niet uitgeweest: het kan het zetten van de conditie niet beinvloed hebben. Als f=29, wordt (C) ingesteld op het inverse van het nieuwe tekencijfer. (f=29 is <u>niet</u> analoog aan f=25! Er worden <u>geen</u> enen bijgevoegd; maar de negatieve schrijfopdrachten wisselen toch geen registerinhouden van teken!) Als nul plaatsen geschoven wordt, wordt weer (C)'=1 gezet.

Een volgende uitbreiding ligt in de koppeling tussen de beide registers. De eerste wijze, waarop dit uitgevoerd is, is door achter de cijfers van A, dus aan de lage kant, de 29 laagste cijfers van S te denken en dit 59 tal als een register te beschouwen. (Schuif A naar S). Het teken van S doet niet mee, evenals OA, die (meestal) intact blijft; OA kan niet effectief zijn; omdat de opteller aan het einde van het 59-cijfer-woord in werking treedt, kan OS wel effectief zijn.

Bij de + additieve schuif A→S is OS effectief: wat S, dus met inbegrip van de OS, aan de lage kant verlaat, komt A aan de hoge kant binnen.
Bij de - additieve schuif worden er aan de hoge kant weer n enen bijgeteld. Bij de + schone schuif A→S wordt A met het oorspronkelijke tekencijfer aangevuld, bij de - additieve schuif A→S met de alternering van het momentane tekencijfer. Wat A aan de lage kant verlaat, gaat nu dus niet verloren, maar verschijnt in S achter het tekencijfer, pas wat S aan de lage kant verlaat, verdwijnt. Bij de schone schuifopdrachten, waarin beide registers betrokken zijn, gaan beide kringoverdrachten kapot. In dit geval dus ook OA.
Bij de uit-versie wordt de conditie op de al of niet inverse van het tekencijfer gezet, dat via de opteller in het register is gekomen. Bij de schuif S A is dit dus, omdat de opteller aan het lage einde van S staat - OS effectief - het nieuwe tekencijfer van A. Bij de negatieve uit-versie wordt C ingesteld op de inverse van het nieuwe tekencijfer van A, er worden weer geen enen aan de hoge kant van A bijgeteld. Bij "schuif A→ S" nul plaatsen, wordt als steeds (C)'=1 gezet. Het tekencijfer van (S) blijft dus bij al deze schuiven ongewijzigd!

De andere wijze van koppeling vindt men door de 30 cijfers van A aan de lage kant van S te denken. Dit is de schuif van S→A; nu zijn beide tekencijfers in de schuif betrokken. De opteller kan nu geschakeld worden aan de lage kant van A: OS is nu intact, OA effectief, behalve bij de schone schuiven, waarbij beide kringoverdrachten vernietigd worden.

Bij de schone schuiven S→A wordt dus S met het oorspronkelijke tekencijfer, of de inverse daarvan, aangevuld; wat aan de lage kant A verlaat, gaat nu verloren. Bij de uit-versies wordt nu de conditie gezet op het nieuwe tekencijfer van S.

Dit geeft een totaal van vier "rondlopen" n.l. "A naar A", "S naar S", "A naar S , onder teken S door" en "S naar A". Deze vier mogelijkheden hebben alle een variant met de band, d.w.z., als f=24 t/m 27 kan het lezen van de telexband er in betrokken worden, als f=28 of 29 het ponsen van een band.

Bij het bandlezen komt het register L in werking, bestaande uit de vijf variabele plaatsen, die door de pentade onder de bandlezer (gaatje =1, ongeponste plaats =0), worden ingevuld, aan de hoge kant door 25 vaste nullen aangevuld. Van deze nullen maakt men alleen maar gebruik, als de bandlezende schuifopdracht meer dan vijf plaatsen schuift: in deze opdrachten worden n.l. de successievelijke cijfers van L betrokken - meegenomen - te beginnen bij het laagste cijfer (Een bandlezende schuifopdracht, die minder dan vijf plaatsen inschuift, haalt niet alle informatie van de band. Na de bandleesop-

dracht wordt n.l. onmiddellijk de band opgeschoven, totdat de volgende rij gaatjes onder de bandlezer ligt). Deze successievelijke cijfers van L komen altijd, met de laagste voorop, een register aan de hoge kant binnen. Bij de schuiven "A naar A" en "S naar S" is dit dus A, resp. S; bij de gekoppelde registers komt L na de opteller binnen. Bij de schuif "A naar S, onder teken S door", waar OS effectief is, komen de cijfers van L dus A aan de hoge kant binnen; bij de schuif "S naar A", waar OA effectief is, komen ze S aan de hoge kant binnen.

Bij de additieve bandleesopdrachten (f=24, resp. 25) wordt L, resp. -L in de opteller opgeteld: L, resp. -L nemen hier dus de functie van de +0, resp. -0 over, die bij de gewone additieve schuifopdrachten toegevoegd worden. Onder -L wordt verstaan inversie van de vijf variabele plaatsen en van de eventueel aangehaalde nullen: bij een additieve bandleesopdracht, die meer dan vijf plaatsen schuift, worden via de opteller na de eerste vijf, dus enen opgeteld.

Bij de schone bandleesopdrachten (f=26, resp. 27) gaan ten eerste de overdrachten kapot, als bij de overeenkomstige schone schuif. In plaats van aanvullen met het oorspronkelijke tekencijfer, alias het inverse daarvan, worden voor zover nodig nu de cijfers van L, resp. -L het register ingepompt.

Tenslotte de bandponsende opdrachten: (f=28 resp. 29) de inhoud van de registers A en S, evenals de conditie, blijven achter, net als bij de overeenkomstige niet ponsende schuiven met f=28, resp. 29. Tevens gaan de eerste vijf cijfers, die "door de opteller" een register verlaten, al of niet geinverteerd, naar het vijf plaatsen omvattende register U, en de inhoud van U wordt als pentade geponst. De inversie bij f=29 betreft weer niet, wat aan de hoge kant in een register terugkomt. Bij de bandponsende schuif "A naar S, onder teken S door" worden dus de vijf laagste plaatsen van S geponst, bij de schuif "S naar A" de laagste vijf van A (Het zijn de cijfers, die bij de overeenkomstige schone schuif verloren zouden gaan). Als men een minder dan vijf plaatsen schuivende bandponsopdracht geeft, dan worden de uitgeschoven cijfers op hun "eigen" plaats in U ingevuld, de resterende van het vijftal worden gelijk gemaakt aan het hoogste wel ingevulde cijfer. Zo kan men de pentade 30 (= 11110) ponsen, door de registerinhoud 5 (101) negatief, 2 plaatsen rondschuivend te ponsen. Bij een nul plaatsen schuivende ponsopdracht wordt mirabele dictu, iets getypt, in de geprobeerde gevallen was dit steeds een nul. Men gebruike deze opdracht niet.
Opm. Het ponsen geschiedt via hetzelfde register U, dat bij het typen gebruikt wordt.

Thans volgt een volledig overzicht van de schuifopdrachten: de vijftien cijfers van de opdracht vallen zeer duidelijk uiteen in de vijftallen f v r . f=24, .... 29 specificeert ± add. in (f=24,25),

± schoon in (f=26,27) en ± uit (f=28,29); v specificeert over welk(e) register(s) de schuifslag zich uitstrekt, en of de band er in betrokken is; in het laatste geval is v steeds 1 groter: v=20 schuift A naar A, v=21 betrekt L of U hierin; v=22 en 23 "A naar S", v=28,29: "S naar A" en v=30,31 "S naar S". Het aantal plaatsen dat geschoven worde (0,1, .....,30) is gelijk aan r.

Opm. Links van elk pijltje staat het register, waar de cijfers aan de lage kant uitkomen; rechts het register, waar de cijfers aan de hoge kant in komen. Het schuiven van A naar S, onder het tekencijfer van S door, is met A ⌒ S aangegeven. Het register U is steeds onder het register A of S geplaatst, dat de cijfers voor U levert; dit, omdat U min of meer parallel gevuld wordt. TA en TS zijn afkortingen van het tekencijfer van A, resp. S; van deze notatie is ook bij de beschrijving van de conditie gebruik gemaakt.

| | f→ | 24,25 | 26,27 | 28,29 |
|---|---|---|---|---|
| v ↓ | r ↓ | r (r=0,1,...,30) | | |
| 20 | r | A ± 0→A<br>OA eff. OS int. | ± IA → A<br>OA kap. OS int. | A→A; (C)'=±IA'<br>OA eff. OS int. |
| 21 | r | A ± L→A | ± L→A | A→A; (C)'=±IA'<br>↓<br>±U |
| 22 | r | S ± 0→A⌒»S<br>OA int. OS eff. | ± IA→A⌒»S<br>OA kap. OS kap. | S→A⌒»S; (C)'=±IA'<br>OA int. OS eff. |
| 23 | r | S ± L→A⌒»S | ± L→A⌒»S | S→A⌒»S; (C)'=±IA'<br>↓<br>±U |
| 28 | r | A ± 0→S→A<br>OA eff. OS int. | ± TS→S→A<br>OA kap. OS kap. | A→S→A; (C)'=±TS'<br>OA eff. OS int. |
| 29 | r | A ± L→S→A | ± L→S→A | A→S→A; (C)'=±TS'<br>↓<br>±U |
| 30 | r | S ± 0→S<br>OA int. OS eff. | ± TS→S<br>OA int. OS kap. | S→S; (C)'= ± TS'<br>OA int. OS eff. |
| 31 | r | S ± L→S | ± L→S | S→S; (C)'= ± TS'<br>↓<br>±U |

De negatief schone schuiven zonder band (27 20 r, 27 22 r, 27 28 r en 27 30 r) moet men liever **niet** gebruiken, omdat wat gebeurt nogal **zinloos** is. Als tekencijfer wordt elke slag ingevuld de inversie, van wat zojuist tekencijfer was: het register wordt alternerend met nullen en enen aangevuld, te beginnen bij de inversie van het oorspronkelijke tekencijfer. Is **r** even (oneven) dan wisselt betroffen register niet (wel) van teken.

De tijdsduur der opdrachten.

De tijdsduur der opdrachten hangt ten nauwste samen met de organisatie van het geheugen. Hiervan is de bespreking nu eerst aan de orde.

Het geheugen bestaat uit een z.g. magnetische trommel, d.w.z. een draaiende cylinder met een magnetiseerbaar oppervlak, langs een beschrijvende waarvan een serie lees- en schrijfkoppen zijn opgesteld. Deze koppen vervullen beide functies: bij het schrijven wordt door de wikkelingen van de kop een "stroombeeld" gestuurd, tengevolge waarvan bij het trommeloppervlak een wisselend magnetisch veld ontstaat, karakteristiek voor het te schrijven getal; omdat in die tijd de trommel onder de kop doordraait wordt een magnetisch patroon langs een streepje van het oppervlak vastgelegd. Draait later dit stukje van de trommel weer onder de kop door, dan ontstaat bij de kop een wisselend magnetisch veld, dat in de wikkelingen van de kop een wisselende spanning induceert, karakteristiek voor het op de trommel geschreven patroon.

Elke kop bestrijkt 32 opeenvolgende adressen: deze 32 adressen vormen samen een kanaal. De localisatie van een bepaald adres op de trommel geschiedt nu in twee "richtingen": ten eerste wordt het goede kanaal, d.w.z. de goede kop, geselecteerd; om in het kanaal het juiste adres te vinden, moet van de goede plaats op de trommel gelezen worden, d.w.z. juist op die momenten, dat de cijfers van het betroffen adres onder de kop doordraaien. Daarom is het rythme van de machine afgestemd op de trommelomwentelingen: aan de zijkant van de trommel zit een schijf met 1024 kleine kerfjes - de klokpulsring - waardoor bij draaiende trommel een sinusvormig signaal, de z.g. klokpuls, naar de machine gestuurd wordt. De klokpuls bepaalt, hoe snel de machine werkt. Draait door een of andere oorzaak de trommel iets te langzaam, dan zal de machine evenredig langzamer werken.

Dit brengt met zich mede, dat de natuurlijke eenheden, waarin de tijdsduur der opdrachten wordt uitgedrukt, omwentelingstijden (of onderdelen daarvan) van de trommel zijn.

Voor ons doel kunnen we volstaan met

de omwenteling - 25 msec.
de getaltijd - ±3 msec.

De trommel hoort te draaien met een snelheid van 40 omwentelingen per seconde, een getaltijd is een achtste gedeelte van een omwenteling. De getaltijd is ingevoerd, omdat dit de tijd is, die verstrijkt tussen het passeren van het laagste en hoogste cijfer van een adres, dat dus een achtste van de omtrek beslaat. (Het laagste cijfer wordt het eerste geschreven en gelezen: de a-opdracht van een koppel komt dus het eerste binnen). Uit het feit, dat de getaltijd een achtste van de omwenteling is, zien we, dat de 32 adressen niet

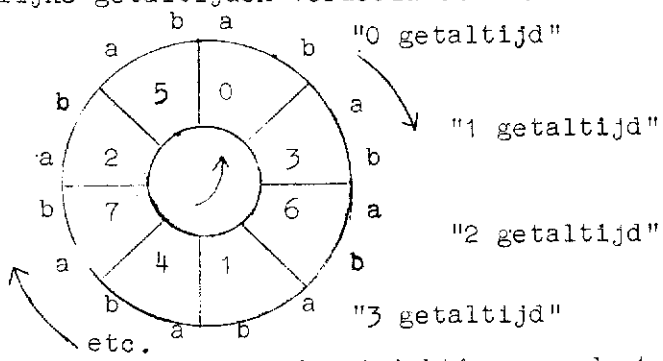
"achter elkaar" geschreven zijn. In dat geval zou n.l. een getal sneller afgeleverd moeten worden en sneller binnenkomen, dan de rest van de apparatuur verwerken kan: tussen het eerste en tweede cijfer van een woord staan op de trommel nog drie andere (eerste of tweede) cijfers van drie andere woorden: elke getaltijd is viervoudig beschreven. Omdat er acht getaltijden zijn kan een kanaal juist 32 adressen bevatten. De tijdslocalisatie geschiedt dus op zijn beurt ook nog in tweeen: ten eerste wordt de betroffen getaltijd opgezocht: van de fase, waarin om de vier cijfers er een gelezen wordt, hangt af, welke van de vier adressen gelezen wordt.

Voor ons doel is de getaltijd een voldoend fijne onderscheiding; wij zullen de vier adressen, die tijdens een getaltijd onder de kop doordraaien, beschouwen als "tegelijkertijd bereikbaar".

De omwenteling is verdeeld in 8 getaltijden, genummerd van 0 t/m 7; de adressen, die gedurende een bepaalde getaltijd langs de koppen draaien, hebben nummers, die een 8-voud verschillen. Schrijven wij dus het adres in het 8-tallig stelsel, dan is de bijbehorende getaltijd bepaald door het laagste cijfer. Aangezien de tijdsduur van de opdrachten op de trommelomtrek, die als lees- en schrijfhandeling bencemd zijn, afhangt van de relatieve positie van de twee er in betrokken adressen - d.w.z. de plaats van de opdracht en de plaats van het door de getalselectie opgezochte adres - kunnen deze adressen voor tijdsberekeningen dus modulo 8 gereduceerd worden. In de tijdstafel zal de opdracht geacht worden te staan op 0 a of 0 b het aangehaalde adres zal lopen van 0 t/m 7.

Als de optelopdracht op 0 b snel is, als het getal van adres 6 komt dan is dit ook het geval, als het getal van 14, 22 of 246, algemeen 8m+6 komt. Stond de opdracht op 1 b, dan geldt hetzelfde voor de getallen op adres 15, 23 of 247, algemeen 8m+7.

Voordat de volledige tijdentabel gegeven wordt, volgt nu de specificatie, hoe de adressen 8m+n (n=0 t/m 7) voor de 8n waarden over de 8 successievelijke getaltijden verdeeld staan. Ze staan "om de drie"




De binnenste pijl geeft aan de draairichting van de trommel; de buitenste geeft aan de richting, waarin dus de koppen langs het oppervlak gaan.

Na de 0 a (8 a, 16 a, 240 a) opdracht komt de 0 b (8 b etc.) op-

dracht langs de koppen. De volgende twee getaltijden komen adressen n=3 en n=6 beschikbaar; adres 3 komt dus 3 getaltijden na adres 0 binnen. Staat een optelopdracht op adres 0 b, dan is aan het einde van de nulde getaltijd de opdracht gelezen, gedurende de eerste getaltijd - als adressen met n=3 de koppen passeren - stelt de besturing zich op de optelling in, aan het einde van die getaltijd kan het getal van de trommel gelezen worden. Staat het getal nu in de positie "2-getaltijd" (adres b.v. 6, 14, 22 of 246) dan wordt het getal gelezen, en in diezelfde getaltijd (instantaan) opgeteld - registers draaien juist in een getaltijd rond -; aan het einde van "2-getaltijd" is de optelling voltooid, en juist komt de opdracht op 1 a onder de koppen. Deze wordt onmiddellijk gelezen. Had het addendum op een van de zeven andere posities gestaan, dan had de optelling onherroepelijk een omwenteling langer geduurd.

Een dergelijke winst is bij de a-opdracht nooit te behalen, hoewel hier de tweede helft van dezelfde getaltijd al gebruikt wordt om de machine op de optelling in te stellen; staat de opdracht op 0 a, dan kan het getal op 1 getaltijd (n=3) al gelezen worden, maar ... de volgende opdracht is immers 0 b, dus de trommel moet helemaal rond, voordat de 0 b opdracht gelezen kan worden. Erger, de a-opdracht duurt een omwenteling extra, als het getal op adres met n=0 staat: als de opdracht gelezen is, is het getal juist voorbij, als na een hele omwenteling het getal gelezen is, is de 0 b-opdracht juist voorbij, en moet weer een hele omwenteling gewacht worden: we hebben een omwenteling verloren!

Met uitzondering van de sprongopdrachten duren de a-opdrachten een geheel aantal omwentelingen + $\frac{1}{2}$ getaltijd, de b-opdracht een geheel aantal omwentelingen + $2\frac{1}{2}$ getaltijd. Deze $\frac{1}{2}$, resp. $2\frac{1}{2}$ getaltijd bengevolge van de "staggering" rond de omtrek zijn <u>niet</u> in onderstaande tabel opgenomen. In de tabel staan omwentelingen.

De opdracht sta op 0 a of 0 b ($\equiv$ mod 8), het functiegedeelte is in de linkerkolom uitgezet, het adres zij gereduceerd modulo 8 gelijk aan n.

| Opdracht | n= | a-opdracht op v r = 8m | | | | | | | | b-opdracht op v r = 8m | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 t/m 3/n | | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 4,5/n | | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 8 t/m 11/n | | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 12,13/n | | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 16 t/m 21/n | | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 5 | 5 | 5 | 5 | 4 | 5 |
| 22/0,23/0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 17 0 | | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 24 t/m 29/16r (snelle verm.) | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 24 t/m 29 (overige) | | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Opm: Bij de schrijfhandelingen kan niet de winst geboekt worden, die bij de leeshandelingen als b-opdracht te oogsten is. De versterkers kunnen nl. niet zo snel overschakelen van schrijven op lezen.

De tijdsduur der sprongopdrachten is bepaald door "de snelste sprong". De snelste sprong van 0 a is naar 6 a, van 0 b naar 1 b. De langzaamste sprong van 0 a is naar 3 b: de eerste keer dat deze positie onder de kop doordraait, is de machine nog niet ingesteld, er moet nog een omwenteling gewacht worden. Van de 0 b opdracht is de sprong naar 6 b het langzaamst. De niet gehoorzaamde conditionele sprongen gedragen zich ook wat de tijdsduur betreft als de skip. Zij kosten dan "geen tijd".

De indeling van het geheugen.

Een kanaal bevat 32 opeenvolgende adressen: dit 32-tal wordt ook wel een vel genoemd: bij de laatste benaming valt het accent anders: is het kanaal een concrete eenheid, aanwijsbaar op de trommel, het vel is de (papieren) eenheid, waarin de opdrachten door de opdrachten door de programmeur groepsgewijs worden ingedeeld.

Elk programma valt natuurlijkerwijze uiteen in enige "losse" stukken, b.v. de opdrachten, een rij constante getallen, een rij, van geval tot geval, veranderende parameters of algemene numerieke gegevens, en een rij z.g. werkruimtes (adressen, waar tussenresultaten zolang onthouden worden). Maar ook de opdrachten zijn in groepen in te delen: "dit stuk doet dit" en "dat stuk doet dat" etc.

Als een programma opgezet wordt, denkt men aanvankelijk uitsluitend in dergelijke termen: men bouwt eerst het programma op in grove trekken (In de trant van "Deze tabel moet in het geheugen staan, er moeten verschillende grootheden - grootheden, die onderling verschillend verwerkt moeten worden - uit deze tabel geinterpoleerd worden. Interpoleren in de tabel zondert zich dus af als "groepje opdrachten". We moeten ettelijke malen een sinus berekenen. Dat en dat getal wordt steeds door iteratie berekend: het stuk programma, dat dit verzorgt, zondert zich eveneens als entiteit te beschouwen groepopdrachten af, etc.).

Nu maakt de programmeur deze afzonderlijke brokken. Waar deze in het geheugen komen te staan, kan hij nog niet bepalen, omdat hij van te voren niet weet, hoelang de diverse brokken zijn. Zolang voorziet hij elke brok van een parameter, de sluitletter s=10 t/m 27 (eventueel 0 t/m 9, zie het invoerprogramma). Elk adres, dat verwijst naar een van deze brokken, krijgt de voor dit brok karakteristieke sluitletter aan zich toegevoegd, terwijl het adresgedeelte nummert van af het begin van betroffen brok (dus 0 0 15= het nulde adres (het beginadres) van de brok, die met s=15 gekarakteriseerd wordt; 2 17 15 is het $81^{ste}$ adres (z.32+17=81) voorbij het beginadres van het brok s=15; de beginadressen zijn als regel de nulde adressen van kanalen; dit vergemakkelijkt de service in hoge mate).

Het vel is de eenheid, waarin de programmeur het geheugen indeelt. Hij karakteriseert de vellen, door de karakteristieke sluitletter er achter, en het velnummer in de brok, ervoor te plaatsen: als de programmabrok, die s=15 als label gekregen heeft, drie vellen beslaat, spreekt hij over 0 vel 15, 1 vel 15 en 2 vel 15 (adres 17 van 2 vel 15 etc.).

Als in deze terminologie het programma voltooid is, wordt de

"absolute" plaats van deze brokken beslist. Immer wordt een vel precies in een kanaal, en niet half in het ene, en half in het andere geplaatst. Zo kan men beslissen, dat 0 vel 15 in kanaal 8 komt; dan komt 1 vel 15 in kanaal 9, 2 vel 15 in kanaal 10; kanaal 11 kan dan het nulde vel van een volgend brok bevatten.

Opm. Programma's worden ingebracht via de geponste telexband. Adressen worden op deze band met drie pentades geponst (opdracht, door de functieletter, dus met vier; in de volgorde f (functie) v (vel) r (regel) s (sluitletter)). Het invoerprogramma (zie aldaar) verzorgt de assemblage van de pentades tot de opdrachtenkoppels. De functie van de sluitletter is, dat het adres met een addendum vermeerderd wordt, n.l. het beginadres van de brok, die door deze sluitletter gelabeld is. Voordat deze banden ingelezen kunnen worden, wordt op daartoe gereserveerde werkruimtes van het invoerprogramma ingevuld, waar de betroffen brokken beginnen, dus welk increment bij welke sluitletter hoort. Dit geschiedt door de z.g. voorponsing, een bandje, dat eerst wordt ingelezen. De machine rekent dus met behulp van het invoerprogramma uit, hoe de opdrachten "er echt uitzien". De programmeur kan dus het programma meteen opschrijven in een code, zodat van deze papieren de banden onmiddellijk geponst kunnen worden. Een additioneel voordeel is, dat men een (standaard) programma door wijziging van de voorponsing, verder met dezelfde banden elders in het geheugen plaatsen kan. Alle wijzigingen in de adressen worden dan automatisch gemaakt.

De verbindingen tussen de machine en de trommel zijn bij FERTA uiterst flexibel: 64 schakelaars zijn bij tweeen gegroepeerd, deze tweetallen zijn genummerd van 0 t/m 31 en zijn door verbonden met de kanaalselecties van de machine. In elk tweetal heeft de bovenste schakelaar betrekking op de opdrachtselectie, de onderste op de getalselectie. Aan dit tweetal beantwoorden twee boven elkaar geplaatste contactbussen, waarin met coaxiale kabels aansluitingen van de koppen geplugd kunnen worden. Staat een schakelaar naar boven dan is de betroffen selectie doorverbonden aan de bovenste contactbus van het bijbehorend tweetal, staat hij naar beneden, dan met de onderste. Als alle schakelaars naar boven staan, zijn dus alleen de 32 koppen, die met de bovenste contactbussen zijn verbonden aangesloten: de machine bestrijkt 1024 adressen met beide selecties. Het zal echter vaak voorkomen, - men kan het hier op aanleggen! - dat vele kanalen gevuld zijn met opdrachten, die niet tijdens de berekening veranderen. Als in deze kanalen tevens geen constanten staan die als getal aangehaald moeten worden, kan men de onderste schakelaars (van de getalselectie) van betroffen kanalen naar beneden zetten, als het programma eenmaal is ingelezen.

Dit heeft twee voordelen

1° de opdrachten in dit kanaal kunnen door deraillement van het programma of machinefouten niet meer overschreven worden; we hebben "een dood kanaal geblokkeerd".

2° de getalselectie is aan een nieuwe kop aangesloten; in dit kanaal kunnen we getallen schrijven, we kunnen het niet met de opdrachtselectie bestrijken. Het geheugen is min of meer met een kanaal uitgebreid.

Men spreekt van een gemengd kanaal, als beide selecties aan dezelfde kop zijn aangesloten. Voor een gemengd kanaal kan men een getallenkanaal en een opdrachtenkanaal in de plaats krijgen.

Men vergroot met deze procedure het aantal adressen, dat door de machine - zij het gedeeltelijk - tegelijkertijd "bespeeld" kan worden. Volslagen verdubbeling tot 2048 adressen is niet mogelijk, omdat een gemengd kanaal nodig is om variabele opdrachten uit te kunnen voeren.

Met het oog op bovenstaande zijn eerder getalselectie en opdrachtselectie afzonderlijk genoemd en beschreven.

Een laatste 32 tal contactbussen is niet aangesloten. Zij dienen uitsluitend als kapstokken voor de resterende 32 coaxiale kabels: er zijn n.l. 96 koppen langs de trommel opgesteld. Dit is geen overdreven luxe: dit maakt onderbreking van een berekening, om plaats te maken voor een urgenter probleem mogelijk: na afloop kan men na terugpluggen gewoon doorgaan.

Het gebruik van subroutines.

Deel I van dit Handboek wordt afgesloten met een beschrijving van het gebruik van subroutines. Strict genomen valt dit niet onder de beschrijving van de machine, het concrete gegeven, dat de programmeur ter beschikking staat. Toch is het in dit eerste deel geplaatst, omdat subroutines een zo essentieel aspect van elk programma vertegenwoordigen, dat de FERTA zonder hen niet denkbaar zou zijn.

Onder een subroutine verstaat men een reeks opdrachten, die tezamen een duidelijk omlijnde operatie uitvoeren.

Het is immers de taak van de programmeur om een bepaald rekenproces op te bouwen uit de opdrachten, die in het voorafgaande beschreven zijn. Het zou voor hem nodeloos veel werk zijn, als hij elk programma moest opbouwen uit deze minuscule bouwstenen, omdat elke berekening gesplitst kan worden in grotere onderdelen, waarvan sommige een zo algemene functie hebben, dat soortgelijke rijen opdrachten zeker in andere programma's voorkomen. Als voorbeelden noemen we het berekenen van de sinus, het trekken van de vierkantswortel, het decimaal uittypen van een getal (d.w.z. berekenen en typen van de successievelijke decimale cijfers, die in het 10-tallig stelsel het intern binair gegeven getal representeren), en het vermenigvuldigen van twee complexe getallen (b.v. berekening van reeel en imaginair deel van het product, als de reele en imaginaire delen van de factoren gegeven zijn).

Als een dergelijk proces in verschillende problemen voorkomt, is het gewenst, dat in beide programma's de opdrachten van de betroffen subroutine met hetzelfde bandje - of een mechanisch vervaardigde copie daarvan - kunnen worden ingebracht. Dit bespaart de staf veel ponswerk en vermindert daardoor de kans op fouten op de band. Echter is het niet gezegd, dat de subroutine in verschillende programma's op dezelfde geheugenplaatsen komt te staan. Aangezien verwijzingen naar plaatsen in de subroutine gespecificeerd worden door adressen, die afhankelijk zijn van de plaats, waar de subroutine in het geheugen staat, stelt de voorwaarde van "identieke banden" (zij het voorafgegaan door een parameter, die de plaats bepaalt) een scherp omlijnde eis aan het invoerprogramma; door de sluitletter **faciliteit** wordt aan deze eis voldaan (zie "De indeling van het geheugen" (Deel I) en het Invoerprogramma (Deel II) van dit Handboek).

Ook in een ander opzicht is de flexibiliteit van een subroutine groot. Laat ons het (klassieke) voorbeeld van een vierkantswortel beschouwen; deze kan met een iteratief proces berekend worden. Laat de subroutine zo zijn uitgevoerd, dat in S achtergelaten wordt de tweedemachtswortel van het getal, dat zich in het begin ("bij binnenkomst in de subroutine") in S bevond.

Nu is het helemaal niet ondenkbaar,dat op verschillende punten in de berekening de vierkantswortel van een getal getrokken moet worden. Nu zou men, iedere keer, dat een tweedemachtswortel berekend moet worden, de opdrachten van de wortelsubroutine kunnen copieren, maar het is duidelijk verkwistend met geheugenruimte, om dit iteratieproces ettelijke malen gecodeerd in het geheugen te hebben staan. (Bij een berekening met complexe getallen, waarin b.v. vijftig verschillende vermenigvuldigingen van complexe getallen voorkomen - geen irreeel geval! - zou waarschijnlijk meer dan het halve geheugen gevuld worden met replica's van de vermenigvuldigsubroutine). Er moet dus gezocht worden naar een methode, om al deze worteltrekkingen (c.q. complexe vermenigvuldigingen etc.) door dezelfde subroutine te laten uitvoeren.

Na de eerste worteltrekking moet echter de berekening op een ander punt worden voortgezet, dan na de tweede, en de derde b.v. Dit betekent, dat als de tweedemachtswortel in S is geplaatst, de besturing een variabele sprongopdracht moet ontmoeten; deze heet de koppelopdracht (= link of linkorder).

Men geeft de subroutine in de a-helft van een van de registers - doorgaans het A-register - de koppelopdracht mee: deze wordt door een schrijfopdracht in het begin van de subroutine aan het einde, achter de laatste opdracht, die de worteltrekking voltooit, geplaatst. Behalve het getal, waaruit de wortel getrokken moet worden, wordt dus een tweede gegeven, parameter, aan de subroutine meegedeeld; de koppelopdracht fungeert als label, ter onderscheiding van de verschillende keren, dat er gebruik van gemaakt wordt. Ten behoeve van de vorming van de koppelopdracht zijn de opdrachten 24 0(8) 4 t/m 27 0(8) 4 ingebouwd. Hiervan is 26 0 4 bij verre de belangrijkste: deze leest de opdrachtteller uit en plaatst in de a-helft van A de sprongopdracht, die twee opdrachten verder verwijst: staat de opdracht 26 0 4 op de a-helft van adres v r, dan wordt ⟨A⟩ = 0 0 0 7 v r+1; stond 26 0 4 op de b-helft van v r, dan ⟨A⟩ = 0 0 0 15 v r+1; in beide gevallen is er ruimte voor een opdracht: hier wordt de sprongopdracht naar het beginadres van de subroutine geplaatst. Als de subroutine voor de wortelsubroutine gelijk is aan V R, wordt die subroutine als volgt gebruikt:

```
                ..... {S} = a
                26 0 4      koppelopdracht in A gevormd.
                 7 V R =)   De besturing springt naar de subroutine
{S} = a^½       =) .....           De besturing komt terug.
```

De besturing gaat heen met $\{S\} = a$, en komt "er onder" terug met $\{S\} = a^{\frac{1}{2}}$ Dank zij de opdracht 26 0 4 is bovenstaand arrangement, de aanroep, onafhankelijk van a-b-positie. Ter onderscheiding van de subroutines zegt men, dat de aanroep staat in het hoofdprogramma. Als de subrou-

tine voor de worteltrekking ergens in het geheugen staat, fungeert de aanroep van de subroutine in het hoofdprogramma als uitbreiding van de opdrachtencode (Bij andere dan sprongopdrachten wordt de opdrachtteller met $\frac{1}{2}$ verhoogd en verder ongemoeid gelaten. Nu de uitvoering van de operatie "trek wortel" geprogrammeerd moet worden, moet daarom de oude (T)even"bewaard blijven". De inhoud van de opdrachtteller wordt door 26 0 4 (inclusief verhoging) in het register aangehaald, bij de sprong naar V R wordt (T) weer "hersteld": we gaan na een intermezzo door, waar we gebleven waren).

Indien een subroutine maar een keer in het programma aangeroepen wordt, betekent de normale aanroep overbodig tijdverlies: steeds wordt n.l. aan het einde dezelfde koppelopdracht ingevuld, wat na de eerste keer kennelijk niet meer nodig is. Om derwille van de overzichtelijkheid van het programma en de uniformiteit in het gebruik van de subroutine legge men zich hierbij neer. Sterker, het is gewoonte, om ook stukken programma, specifiek voor het betroffen probleem als subroutine uit te voeren, met het oog op de overzichtelijkheid en de flexibiliteit.

Een voorbeeld moge dit toelichten. Een programma tikt de resultaten in genummerde regels uit; als het regelnummer een vijfvoud is, worde van te voren een blanke regel (extra TWNR) ingelast, als het regelnummer een vijftigvoud is, moet na afloop van het typen van die regel de machine stoppen (zodat een nieuw vel in de schrijfmachine gedaan kan worden) en bij doorstarten moet deze regel opnieuw uitgetypt worden.

Het stukje programma, dat een regel uittypt is gemaakt in de veronderstelling, dat aanvankelijk de wagen aan het begin van de regel staat, terwijl na afloop door het signaal TWNR, de wagen in dezelfde positie afgeleverd wordt. Voor dit stukje programma sta de 4 opdracht, die de koppelopdracht aan het einde van het programma "Typ regel" plaatst: we voeren met in- en uitloop het programma als subroutine uit. Als de resultaten van een regel berekend en gecontroleerd zijn, en het typen kan beginnen, luidde het programma, dat sta op $v_0\ r_0$ en volgende, b.v. als volgt

| | | | | | |
|---|---|---|---|---|---|
| $v_0\ r_0$ | ,a | 10 | $v_1$ | $r_1$ | $[v_1\ r_1]$ = regelnummer. |
| | ,b | 26 | 4 | 0 | plaats +0 om A (voor deling). |
| $v_0\ r_0+1$ | ,a | 20 | $v_2$ | $r_2$ | $[v_2\ r_2] = 5$ |
| | ,b | 29 | 0 | 16 | is A $\neq +0$, n.l. de rest. |
| $v_0\ r_0+2$ | ,a | 14 | $v_0$ | $r_0+9$ -+ | $\neq 0$, midden in een blok. |
| | ,b | 26 | 4 | 11 | 11 in A. |
| $v_0\ r_0+3$ | | 28 | 0 | 8 | TWNR |
| | | 26 | 4 | 10 | 10 in A. |
| $v_0\ r_0+4$ | | 28 | 0 | 8 | TAB |
| | | 26 | 4 | 0 | 0 in A (om door te delen) |

```
                  v0 r0+5     20   v3   r3        [v3 r3] = 10
                              29   0    16        is de rest ≠ 10
                  v0 r0+6     14   v0   r0+9  -+ niet laatste pagina
                              26   0    4
                  v0 r0+7      7   v4   r4   =)  naar subr. "Typ regel"
terug         =)              23   0    0        stop
                  v0 r0+8     22   0    0
                              26   0    4    }   standaard TWNR, wacht,
                                                    TAB
                  v0 r0+9      7   3    31   }   (In- en uitvoerprogramma)
2'a,6'a       =)              26   0    4
                  v0 r0+10     7   v4   r4   =)  naar subr. "Typ regel"
terug         =)              ............
```

Ter explicatie:

De opdrachten zijn van boven naar beneden opgeschreven in de volgorde, waarin ze - onder voorbehoud van sprongopdrachten - worden uitgevoerd. Voor elk tweetal staat het adres, dat dit opdrachtenkoppel herbergt. De a-b-indicatie, als in de bovenste regels, laat men verder weg. $v_1\ r_1$ , $v_2\ r_2$ en $v_3\ r_3$ zijn de benamingen voor de drie adressen, waaruit regelnummer en constanten 5 en 10 staan. $v_4\ r_4$ a is de beginopdracht van de subroutine "Typ regel".

De eerste twee opdrachten plaatsen het regelnummer in [AS] ; omdat de deling geschiedt op het lange getal [AS] is het plaatsen van 0 in A absoluut noodzakelijk. Na de deling door 5 staat de rest in A en het quotient in S; dit laatste zullen we nog gebruiken, als de rest = +0 is. Als de rest ≠ +0 is, is de komende regel niet de eerste van een blokje van vijf, en kan deze dus zonder meer getypt worden. In dit geval wordt de sprong naar $v_0\ r_0$+9,b wel gehoorzaamd, daarna ontmoet de besturing de aanroep voor de subroutine "Typ regel"; dit gebeurt, de besturing komt na afloop terug in $v_0\ r_0$+10,b , waar het programma doorgaat. Als echter wel het regelnummer een vijfvoud is, is de rest = +0, door 29 0 16 (in $v_0\ r_0$+1,b) wordt de conditie - , de sprong wordt niet gehoorzaamd. De komende regel is of de eerste van een blokje van vijf, of de allerlaatste - eenzame - onderaan de pagina. In elk geval moet het papier een regel opgeschoven worden, wat gebeurt door de twee op de sprong volgende opdrachten. Vervolgens wordt een TAB gegeven: het begin van de regel wordt n.l. niet gekozen bij de kantlijn, maar bij de eerst volgende tabulatorstop - dit, omdat bij het teruglopen van de wagen bij de kantlijn door de schok de wagen wel eens een plaats terugschiet. TWNR en TAB volgen elkander niet te snel op: de wagen stond immers aan het begin van de regel - bij de eerste tabulatorstop. In S bevindt zich nog het quotient: als het oorspronkelijke regelnummer een vijftigvoud is geweest, is dit quotient een 10-voud. Nadat +0 in A geplaatst is, wordt [AS] door 10 gedeeld, en getest wordt, of de rest ≠ +0. Zo ja, dan is het einde

van de pagina nog niet bereikt, en de sprong naar $v_0$ $r_0$+9,b verwijst de besturing naar de aanroep, die een regel "in een blokje" typt, waarna etc. Laat de deling door 10 echter een rest =0 achter, dan is de laatste regel aan de beurt: deze wordt getypt via de eerste aanroep - de aanroep van de eenzame regel onderaan - waarna de machine op een van beide stopopdrachten zeker stopt. De operateur kan een nieuw papier in de schrijfmachine doen, en doorstarten. Eerst wordt met een aanroep van het In- en Uitvoerprogramma (zie aldaar) TWNR en TAB gegeven, nu echter gescheiden door een vertraging om de wagen terug te laten lopen (De operateur heeft aan de schrijfmachine gezeten). Dan wordt de eerste regel van de nieuwe pagina via de tweede aanroep getypt.

Als het invoerprogramma en de ponsconventies beschreven zijn, kunnen wij de programma's opschrijven "zoals ze op de band komen". Aangezien dit gewoonte is, zal tot dat moment gewacht worden met aan de hand van voorbeelden enkele subroutines, ter verduidelijking van 't bovenstaande, te bespreken.