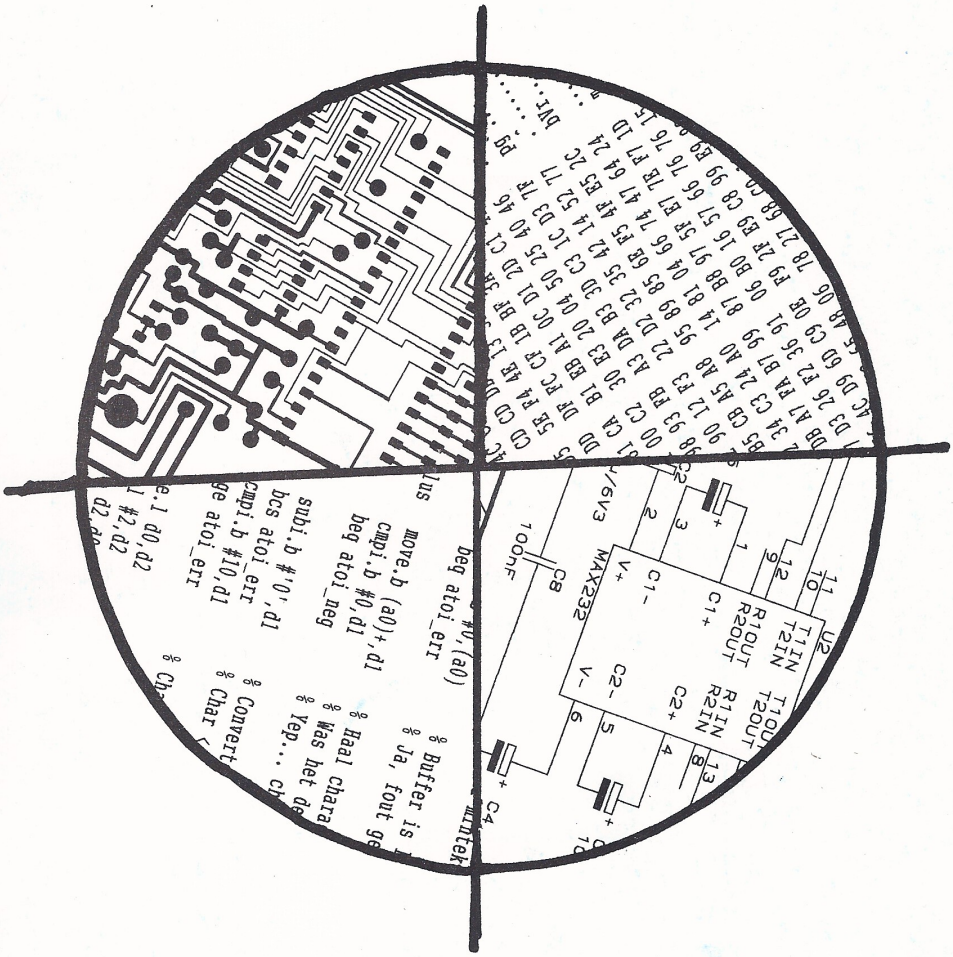
De µP Kenner

Zestiende jaargang nr. 2
April 1992
76



In dit nummer o.a.:

LINUX
Systeem informatie van 4DOS
Sorteren met behulp van chaos
Stereo D/A omzetter voor ModPlay
Object georiënteerd programmeren

Inhoudsopgave

De μP Kenner

Nummer 76, april 1992
Verschijnt 6 maal per jaar
Oplage: 250 stuks
Druk: FEBO Offset, Enschede

De redactie:

Joost Voorhaar
Gert van Opbroek
Geert Stappers
Nico de Vries

Eindredactie:

Joost Voorhaar

Vormgeving:

Joost Voorhaar
Nico de Vries

Redactieadres:

Joost Voorhaar
Jekerstraat 67
7523 VP Enschede

De μP Kenner nummer 77 verschijnt op 15 augustus 1992.

Kopijsluitingsdatum voor nummer 77 is vastgesteld op 1 augustus 1992.

Algemeen

Redactioneel 4
Moderne Schizofrenie 8
Nieuwtjes en andere wetenswaardigheden 39

Vereniging

Van de bestuurstafel 53

Talen/Software

To Share Or Not To Share, That's The Question... 10
Een snelle assembler met behulp van hash en chaos 13
Systeem informatie via 4DOS 25
COM-file maker 30
Object georiënteerd programmeren 31
De hereniging 40

Hardware

Stereo D/A omzetter voor ModPlay 5
Voortgang KGN-68k (deel 6) 38

De μP Kenner is het huisorgaan van de KIM gebruikersclub Nederland en wordt bij verschijnen gratis toegezonden aan alle leden van de club. De μP Kenner verschijnt vijf maal per jaar, in principe op de derde zaterdag van de maanden februari, april, augustus, oktober en december.

Kopij voor het blad dient bij voorkeur van de leden afkomstig te zijn. Deze kopij kan op papier, maar liever in machine-leesbare vorm opgestuurd worden aan het redactieadres. Kopij kan ook op het Bulletin Board van de vereniging gepost worden in de redactie area. Nadere informatie kan bij het redactieadres of via het bulletin board opgevraagd worden.

Redactioneel

Het rommelt wat af in computerland... Niet alleen op technisch gebied, maar vooral ook op organisatorisch en bedrijfstechnisch gebied. Zo las ik een aantal weken terug het bericht dat IBM het helemaal gehad heeft met de IBM/PC... zo erg zelfs dat ze hem zelf gaan klonen. Er zou een overeenkomst met een Aziatische klonenbakker zijn aangegaan die de nieuwe IBM-klonen gaat maken. Er komt alleen géén IBM-merkje op; dat gaat de hoge heren van IBM waarschijnlijk net te ver... Naast deze superklonerij heeft IBM vergaande plannen met Bull gesmeed; de IBM-dealers mogen van Bull portables van de dochteronderneming Zenith gaan verkopen.

Wat dichter bij huis vinden we de firma Zyztm. Deze firma heeft computers uitgeleverd met een illegale versie van MS-DOS 5.00. Niet helemaal duidelijk is of het hier gaat om originele versies die gepatched zijn of dat het om pure piraterij gaat. In ieder geval heeft MicroSoft de heren zo ver gekregen dat zij middels advertenties zijn overgegaan tot het oproepen van clienten die een dergelijke DOS-versie van Zyztm bij hun computer kregen. Als de klant de aankoopbon opstuurt, retourneert Zyztm een legale DOS-versie. MicroSoft houdt nauwkeurig in de gaten of er niet meer illegale pakketten verkocht zijn dan volgens Zyztm zou kunnen en heeft al gedreigd onmiddellijk naar de rechter te stappen als dat het geval blijkt te zijn!

Tenslotte lijkt er aan de slepende "look- and feel" zaak Apple versus MicroSoft en Hewlett Packard een einde te komen. Van de oorspronkelijke 189 claims heeft het Amerikaanse hof besloten dat er slechts enkele werkelijke geschilpunten bestaan tussen Apple en de twee beklaagden. Hierbij spitst de zaak zich toe op de fonts die MicroSoft gebruikt en het vuilnisbakje van HP's New Wave.

Ook op hardware gebied zijn er veel ontwikkelingen gaande. Zo heeft Intel een tipje van de '586-sluier opgelicht. Het ding zal in oktober waarschijnlijk het echte daglicht zien, maar het is al wel duidelijk dat het zal gaan om een RISC-achtige processor die óf-wel via een '486 emulator óf middels een speciale compiler de bestaande programmatuur kan draaien.

In clubverband loopt het echter allemaal niet zo fantastisch. De KGN68k/30 werkgroep schijnt druk bezig te zijn met de layout van de PCB. De schema's zullen dan neem ik aan wel klaar zijn, maar voorlopig krijg ik uit die hoek nog geen kopij. De enige conclusie die ik hieruit kan trekken is dat de werkgroep toch niet zo goed functioneert als we zouden wensen. Ook met het DOS-65 gebeuren gaat het niet voor de wind. Het wachten is daar op iemand die voldoende tijd in de lopende projecten kan en wil steken. Ik heb begrepen dat het voldoende zou zijn als iemand het voortouw daar op zou pakken... de wil is er wel, maar om het nu altijd op de zelfde personen terecht te laten komen... nee, dat werkt gewoonweg niet!

Gelukkig zijn er nog een aantal actieve leden die iedere keer weer op het laatste moment met iets aan komen dragen, maar op een zeker punt is dat geluk natuurlijk op.

De kopij-kist blijft zo natuurlijk beangstigend leeg. Dat er iedere keer weer wat kopij uit een verstoft hoekje opgeduikeld kan worden is meer geluk dan wijsheid lijkt het wel. Gelukkig zijn er nog een aantal actieve leden die iedere keer weer op het laatste moment met iets aan komen dragen, maar op een zeker punt is dat geluk natuurlijk op. En dan... sja, dan zullen we een μP Kenner moeten maken met een hoop witte pagina's. Ach ja, aantekeningenboekjes zijn nooit weg. Zeker niet als ze zo'n groot formaat hebben...

Joost Voorhaar

Stereo D/A omzetter voor ModPlay

In de uP-kenner nummer 75 publiceerde Joost Voorhaar een artikel over het shareware-programma ModPlay. Bij dat artikel werden een tweetal schema's afgedrukt van digitaal-naar- analoog omzetters. Echter: zelfs met een microscoop was ik niet in staat om te achterhalen wat er nu eigenlijk stond...

In het ModPlay pakket bevindt zich gelukkig een bestand HARDWARE.DOC. Daarin staan bovengenoemde schema's beschreven. En hoewel het geluid via de PC-speaker niet onaardig klinkt, blijft het wat krakkemikkig. Dus werd besloten om maar eens een echte D/A omzetter te bouwen.

De hardware

In HARDWARE.DOC gebruikt de auteur van ModPlay een D/A omzetter van Plessey, namelijk de ZN428. Deze omzetter bleek in Nederland moeilijk leverbaar. Gelukkig bestaan er nog meer fabrikanten van D/A omzetters, en de firma Analogue Devices bleek een prima alternatief te hebben, namelijk de AD558. Deze omzetter heeft zelfs iets betere specificaties dan de ZN428.

Het door de D/A omzetter geleverde signaal wil je natuurlijk ook graag hoorbaar maken. Daartoe werden twee geïntegreerde versterkertjes toegepast, namelijk de LM386. Voor de echte freaks is er ook nog voorzien in een lijn-uitgang, waarop een cassette-recorder kan worden aangesloten.

Het schema

Geheel links bevindt zich de connector K1 waarop de digitale informatie vanuit de printer-poort wordt aangeboden. De getallen links van de connector geven aan hoe elke lijn met de desbetreffende pen van de 25-polige D-connector van de printer-poort dienen te worden verbonden.

Na omzetting in de twee AD558 omzetters wordt het analoge signaal via twee instelpotmeters aangeboden aan de versterkertjes LM386. Het versterkte signaal is beschikbaar op de connectoren K2 en K4. Hierop kan direct een luidspreker worden aangesloten.

Het (geregelde) lijnsignaal is beschikbaar op de Cinch-connectoren K3 en K5.

De voeding-spanning (5 Volt) dient te worden aangeboden op connector K6. De voeding kan bijvoorbeeld worden afgetakt van een ongebruikte floppy-connector. Voor de eigenaren van een game-adapter is het nog eenvoudiger: op de D15-connector aan de achterzijde van de computer is +5 Volt beschikbaar op pen 1, en massa op pen 4.

De condensatoren C11 en C7..C10 zorgen voor de nodige ontkoppeling. Led D1 tenslotte verzorgt de indicatie van aanwezigheid van de voedings-spanning.

De print

Met behulp van UltiBoard is er een print ontworpen voor de schakeling. De opbouw ervan is eenvoudig: begin met de draadbruggen (6 x), daarna de rest. Voor de Cinch-connectoren zijn standaard inbouwdelen te gebruiken. Buig de massa-lip 90 graden om, en soldeer deze aan de sporenzijde van de print. Gebruik IC-voeten voor de IC's.

Het in gebruik nemen

Laat de IC's nog even uit de voeten. Sluit de 5 Volt aan, en controleer of LED D1 brandt. Als alles OK is, schakel dan de voeding weer af en plaats de IC's in hun voeten. Draai de potmeters VR1 en VR2 geheel linksom. Sluit K1 aan op de printerpoort. Het maakt niet uit of LPT1: of LPT2: gebruikt wordt. Nadat alles goed is aangesloten, kan ModPlay gestart worden. Type MP
. ModPlay zal vervolgens zelf 'ontdekken' dat er een stereo D/A converter is aangesloten, en dit op het scherm melden. Selecteer een .MOD file, en activeer die. Draai vervolgens VR1 en VR2 rechtsom tot het gewenste niveau. En dan....

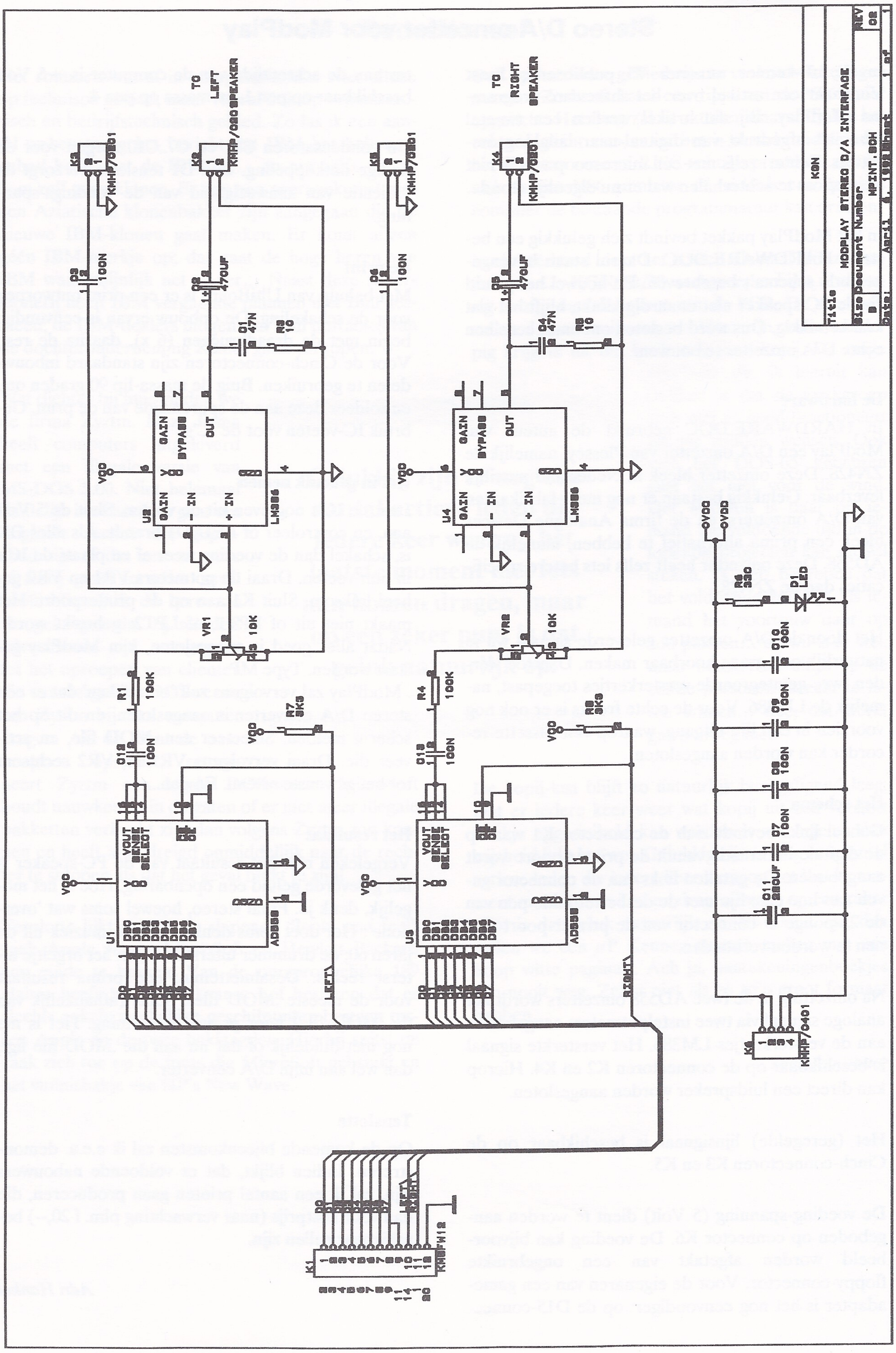
Het resultaat

Vergeleken met het resultaat van de PC-speaker is het geleverde geluid een openbaring. Hoe is het mogelijk, denk je. Fraai stereo, hoewel soms wat 'overdone'. Het doet soms denken aan de muziek uit de jaren 60; de drummer uiterst links, en het orgeltje uiterst rechts. Desalniettemin: een prima resultaat voor de meeste .MOD files. Soms (afhankelijk van de .MOD file) hoor je wat vervorming. Het is me nog niet duidelijk of dat nu aan die .MOD file ligt, dan wel aan mijn D/A converter.

Tenslotte

Op de komende bijeenkomsten zal ik e.e.a. demonstreren. Indien blijkt, dat er voldoende nabouwers zijn, zal ik een aantal printen gaan produceren, die dan voor kostprijs (naar verwachting plm. f 20,--) beschikbaar zullen zijn.

Adri Hankel


Title
MODPLAY STEREO D/A INTERFACE
Size Document Number
B
MPINT.SCH
REV
02
KON
TO LEFT SPEAKER
TO RIGHT SPEAKER
AD558
LM386
KHHP/0401

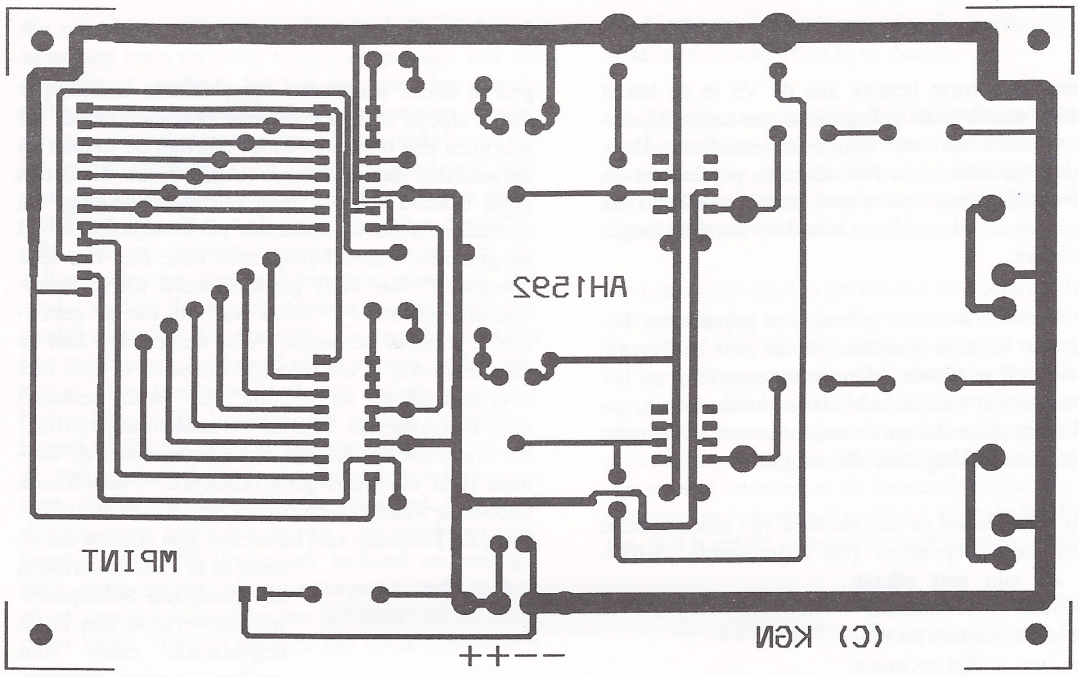


Fig. 1: De koperzijde van de print

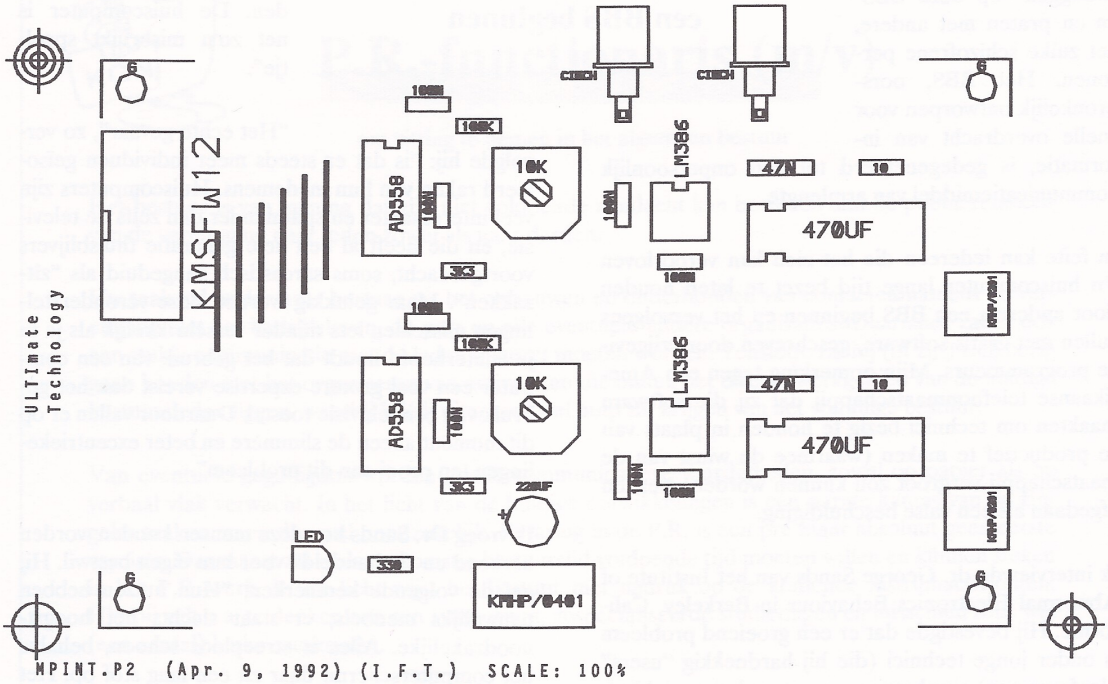


Fig. 2: De componentenopstelling

Moderne schizofrenie

Tijdens mijn korte bezoek aan de VS in de herfst van 1987 was ik in de gelegenheid een aantal bijzondere gevallen van schizofrenie te bestuderen. Hoewel deze gevallen door Amerikaanse psychiaters als ongevaarlijk wordt beschouwd, wordt er erkend dat het een groeiend probleem is onder vooral de jongere technici.

Gefrustreerd door een gebrek aan populariteit beginnen de technici waarover we het hier hebben tegen zichzelf te praten. Maar in tegenstelling tot het gemompel wat we hier in Moskou kunnen horen, gebruiken de Amerikanen de moderne technologie die hen ter beschikking staat: de computer.

In de V.S. bestaat er een netwerk van electronische bulletin-board systemen (de zogenaamde "BBS"-en). Ze zijn met elkaar verbonden door commerciële telefoonlijnen en modems; een stukje techniek om de computers met het telefoonnet te kunnen verbinden. Individuele gebruikers kunnen met hun eigen computer en modem "inloggen" op deze BBS-en en praten met andere, net zulke schizofrene personen. Het BBS, oorspronkelijk ontworpen voor snelle overdracht van informatie, is gedegenereerd tot een onpersoonlijk communicatiemiddel van armlengte.

In feite kan iedereen die het zich kan veroorloven z'n huiscomputer lange tijd bezet te laten houden door anderen, een BBS beginnen

In feite kan iedereen die het zich kan veroorloven z'n huiscomputer lange tijd bezet te laten houden door anderen, een BBS beginnen en het vervolgens vullen met gratis software, geschreven door vrijgevige programmeurs. Mijn opmerking tegen een Amerikaanse telefoonmaatschappij dat zij die software maakten om technici bezig te houden in plaats van ze productief te maken (waarmee de winst van de maatschappij vergroot zou kunnen worden) werden afgedaan als een valse beschuldiging.

Ik interviewde dr. George Sands van het Institute of Abnormal Electronics Behaviour in Berkeley, Californië. Hij bevestigde dat er een groeiend probleem is onder jonge technici (die hij hardnekkig "users" bleef noemen) met het toenemen van het aantal bulletin-board systemen. "Er zijn in feite meer BBS-en dan er users zijn in de "Bay-area" (San Fransisco en omgeving) en die users communiceren met steeds dezelfde personen.Sommigen tonen duidelijk symptomen van verveling. Een aantal slimmerikken loggen in onder een aantal verschillende namen, per naam steeds met een nieuwe persoonlijkheid. Ze schrijven een berichtje onder één van de namen en antwoorden dan onder een andere naam. Er is een geval bekend van een man van midden-vijftig, die zichzelf zo'n zes verschillende persoonlijkheden had aangemeten, waarschijnlijk zelfs acht. Eén van deze "personen" was zelfs gepromoveerd tot assistent-systeem-operator.!". "Maar hoe kan dat nu gebeuren?", vroeg ik dr. Sands. "Wel, de operator had de man nog nooit in levende lijve ontmoet, of zelfs met hem gepraat via de telefoon". Dr. Sands bestreed mijn bewering dat BBS-en verontmenselijkten met het argument dat dat ook werd gezegd over de telefoon toen die werd geïntroduceerd. "Amerikanen hebben te weinig geschiedenis om iets serieus te nemen. Ze Fröbelen veel liever met hun spullen dat ze maar al te vaak misbruiken als speelgoed. Scheepsontwerpen veranderden in de negentiende eeuw voor snellere en meer commerciële reizen over de gehele wereld. Al snel daarna werden schepen door de Amerikanen gebruikt om race-wedstrijden te houden. De huiscomputer is net zo'n misbruikt speeltje".

"Het echte gevaar", zo vervolgde hij: "is dat er steeds meer individuen geïsoleerd raken van hun medemens. Huiscomputers zijn veel interessanter en spannender dan zelfs de televisie, en die heeft al een hele generatie thuisblijvers voortgebracht, soms sarcastisch aangeduid als "zitzakken". Maar gelukkig worden deze veronderstellingen misschien iets minder verschrikkelijk als je in je achterhoofd houdt dat het gebruik van een computer een veel grotere expertise vereist dan het gebruik van een televisie toestel. Daardoor vallen er op dit moment alleen de slimmere en beter excentriekelingen ten prooi aan dit probleem".

Ik vroeg Dr. Sands hoe deze mensen konden worden herkend en "behandeld" voor hun eigen bestwil. Hij gaf de volgende kenmerken: "Hun huizen hebben nauwelijks meubels; er staat slechts het hoogstnoodzakelijke. Alles is streeploos schoon, behalve het computerscherm: daar zit een laag stof op. Het bed is nooit opgemaakt. Het huis heeft zes of zeven telefoonlijnen. Er zijn alleen computer-handleidingen te vinden, geen andere boeken. De mannen om wie het gaat zijn al gescheiden, of staan op het punt om dat te doen. Ze hebben geen relatie of hebben

die misschien zelfs nog nooit gehad. Er zijn trouwens nog geen vrouwen ten prooi gevallen aan dit verschijnsel, ondanks het feit dat sommige van de gefingeerde persoonlijkheden vrouwen zijn. Kinderen zijn weggelopen van huis. Geen enkele van de jongere slachtoffers heeft overigens kinderen. Sexueel zijn ze inactief; ze planten zich in ieder geval niet voort. Net als met alcoholisme, zullen de slachtoffers zich zeer punctueel op hun werk melden. Ze zullen nagenoeg niet worden gepromoveerd; niet omdat ze daar de capaciteiten niet toe bezitten maar omdat ze anders de vrije tijde, die ze gebruiken om hun verslavende bezigheid uit te voeren, zullen verliezen. Hun koelkast zal slechts enkele geopende blikjes bier of fris bevatten. De meeste slachtoffers beperken zich tot chips en frisdrank vanwege het hoge suikergehalte". Er is een geval bekend van een man, die in zes dagen niet had gegeten. Hij werd door agenten gevonden in een supermarkt na sluitingstijd, waar hij, hysterisch lachend en omringd door open blikjes cola en opengescheurde zakken chips, probeerde naar buiten te bellen met de computer-kassa. Toen de agenten zijn dikke brilleglazen en de plastic pennenhouder in zijn borstzakje ontdekten, waarschuwden ze dr. Sands.

De Amerikaanse regering heeft geprobeerd electronische bulletin board systemen te introduceren in Moskou, zodat onze kinderen op dezelfde wijze zouden worden beziggehouden met nutteloze arbeid. Zonder succes overigens.

De grote geleerde Pavlov maakte ooit eens duidelijk, dat om een kit te hypnotiseren je slechts een witte lijn op de stoep hoeft te trekken, waar je vervolgens de kip bovenop zet met zijn poten aan weerskanten van de lijn. De kip zal zich niet meer durven te bewegen. Mensen hebben complexer hypnotiserend gereedschap nodig; de televisie heeft wat dat betreft zijn rol wel bewezen in de laatste decennia. Nu is er dus een hypnotiserend werktuig voor de intelligenten. Het is gelukt ze tegen zichzelf te laten praten.

Rangott Spliekin, columnist voor de Pravda
Vertaling: Yves Barbero en Maarten Festen

To Share Or Not To Share, That's The Question...

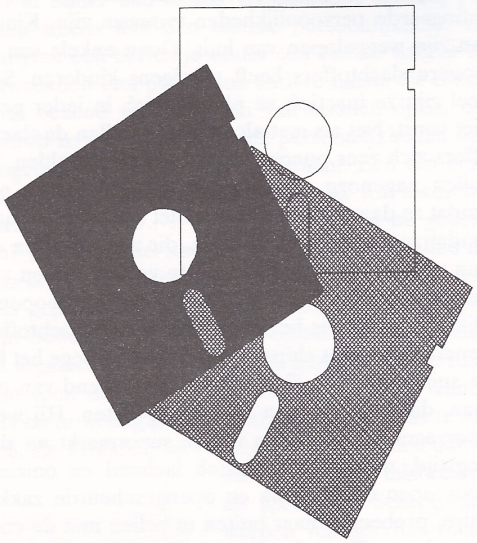
Al anderhalf jaar wordt er nu binnen de club gesproken over Minix en het KGN68k/30 project. Onlangs heeft Minix er echter een geduchte concurrent bijgekregen in de vorm van het vrij verspreidbare Linux. Linux is een UNIX-like operating system dat specifiek voor de '386 lijn geschreven is. In deze aflevering van "To Share..." werpen we een blik op deze veelbelovende UNIX-kloon en zal ik proberen de voor- en nadelen van Linux ten opzichte van Minix uit de doeken te doen.

Linux is geschreven door Linus Torvalds, voor zover ik heb kunnen nagaan een docent aan de universiteit te Helsinki. Het systeem is geheel volgens het zogenaamde "copyleft" agreement van de Free Software Foundation (beter bekend door het GNU project!) verkrijgbaar. Dat betekent dat het zaakje weliswaar copyrighted is, maar dat de executables vrij beschikbaar zijn. De sources van GNU-software moeten altijd tegen nominale kosten (d.w.z.: tegen materiaal- en verzendkosten) beschikbaar gesteld worden. In tegenstelling tot veel shareware mag GNU-spul wel voor commerciële doeleinden gebruikt worden, zolang de software zélf maar niet commercieel aan de man gebracht wordt. Zo wordt de GNU C-compiler (GCC) meegevelerd met de NeXT, het 68030-systeem uit de stal van Steve Jobs.

Systeemeisen

Voor het draaien van Linux is, zoals gezegd, minimaal een 80386-PC nodig. Dit mag in de praktijk natuurlijk ook best wel een 386SX zijn, maar dat merk je onmiddelijk in de performance van het systeem. Verder moet de machine minimaal beschikken over 2 MByte aan RAM en een 1.2 of 1.44 MByte floppydisk. Wil je écht gebruik gaan maken van Linux, dan heb je (uiteraard) ook een harddisk nodig en eigenlijk toch wel 4 MByte aan RAM. Die harde schijf moet in principe AT-compatible zijn; dat wil dus zeggen: MFM of IDE techniek. Of RLL ondersteund wordt is mij nog niet geheel duidelijk, maar ik verwacht eigenlijk van wel. Vanaf versie 0.95c+ wordt ook SCSI ondersteund volgens zowel de SCSI/1 als de SCSI/2 standaard. Aan diskruimte wordt tenslotte 20-40 MByte als minimum genoemd. Dit lijkt me wel wat veel; 10 MByte zou voldoende moeten zijn om een simpel systeem op te zetten. Echt lekker draait het systeem echter pas met zo'n 40 tot 60 MByte aan harddisk ruimte. Als console voldoet een hercules, CGA, EGA of (S)VGA compatible display. Van een aantal SVGA kaarten worden zelfs extended modes ondersteund (d.w.z.: meer dan de obligate 25 regels van 80 kolommen!).

De eisen die Minix aan een systeem stelt zijn, zoals bekend verondersteld mag worden, een stuk lager.



Zo is een PC/XT met twee floppen en 512 kByte RAM al voldoende. Verder draait Minix op de IBM-PC, Amiga, atari ST, McIntosh en Sun Sparc, om maar eens een aantal systemen te noemen. Daar staat natuurlijk wel tegenover dat Minix door zijn flexibiliteit minder bruikbaar is als dagelijks werkpaard dan Linux.

Installatie

Voor het installeren van Linux is volgens de verschillende bronnen (bijna) geen kennis van MeSsy DOS en/of UNIX nodig, maar naar mijn idee kun je zonder die kennis voor behoorlijke problemen komen te staan. Verder heb je een aantal tools nodig die lang niet iedereen direct beschikbaar heeft om Linux aan de praat te krijgen.

De eerste kennismaking met het systeem begint natuurlijk bij de distributie. Op een aantal bulletin boards en op een groeiend aantal Internet sites is Linux beschikbaar. Vooral als de originele distributie van het Internet naar MS-DOS gehaald wordt, is er wat niet-standaard programmatuur nodig om het zaakje aan de praat te krijgen. Zo worden de originele boot- en rootimages gecompressed aangeleverd. Wil je die dus op flop zetten, dan heb je sowieso al een implementatie van compress voor DOS nodig. Verder heb je programmatuur nodig om de diskimages naar flop te copiëren. Dit gaat uitstekend met de Norton Utilities, maar er is ook een apart programmaatje ("Rawrite") beschikbaar voor onder DOS. Met enkel de root- en bootimage heb je een minimaal systeem dat van flop draait.

Om bestanden van DOS naar Linux te krijgen is er een speciaal pakketje geschreven onder de naam "MTools". Hier doet zich echter een kip-ei probleem voor dat zich niet zo eenvoudig laat oplossen. Hoe kom je namelijk aan die tools? Jawel, downloaden van een BBS of van Internet. Die programma's zijn in de originele distributie (mtools.tar.Z) getar'd en vervolgens gecompressed. Die kunnen we onder Linux wel uitpakken, maar hoe krijg je die dingen Linux binnen? De meest voor de hand liggende oplossing is natuurlijk door ze in Linux te downloaden. Maar daar hebben we weer een terminal programma voor nodig... en hoe je daar dan aan zou moeten komen? Downloaden misschien? Een verhaal zonder eind dus. Een andere mogelijkheid is, om Mtools.tar naar flop te "TARren" onder DOS. Daar heb je dus een tar voor DOS systemen voor nodig. En gelukkig bestaat die wel... vervolgens Linux opstarten, en van flop "terug-TARren". Maar dat gaat niet als je Linux van flop draait... daar heb je meer ruimte voor nodig! Van harde schijf starten dus. En daarvoor moet je eerst het hele zaakje voor op harddisk zetten. Voorwaar, een onhandig klusje!

Hier doet zich echter een kip-ei probleem voor dat zich niet zo eenvoudig laat oplossen. Hoe kom je namelijk aan die tools?

Om die MTools aan de praat te krijgen hebben we wel wat meer nodig dan alleen een compress en tar... ook een C-compiler is onontbeerlijk, want in de normale distributie is MTools alleen in source geleverd! Nu is Linux zelf met behulp van de GNU-C compiler geschreven en GCC is dan ook als compiler voor Linux beschikbaar. Er is voorgesteld om GCC 2.1 tot standaard-compiler te benoemen. Deze C++ compiler is nog in Beta-versie beschikbaar, maar dat is Linux zelf ook.

Ofschoon het begin eenvoudig is, haal je je met de volledige installatie van Linux dus toch best wat werk op de hals. Leerzaam is het allemaal wel, vooral als je nog (bijna) geen kaas van UNIX (of UNIX-achtige) systemen hebt gegeten.

Documentatie

Ik heb het al even zijdelings genoemd: Linux is nog in een beta-stadium. Dat betekent o.a. dat de documentatie vrijwel nihil is. Gelukkig is er al wel een manual systeem beschikbaar en lijkt Linux behoorlijk op een "echte" UNIX. De documentatie bestaat nu nog uit een information sheet, een installatie-handleidinkje, een FAQ-list (ehh... een lijst met "Frequently Asked Questions"), een aantal losse verhaaltjes en de on-line manuals. Alles is (uiteraard) in het Engels (en waarschijnlijk ook in het Fins...) beschikbaar. Het verdient wel de aanbeveling om er een goed UNIX-boek bij te hebben, vooral als je geen of weinig ervaring met een Minix-achtig systeem hebt.

Tenslotte

Groot is het allemaal wel. Nou ja, dat wil zeggen... als je het met bijvoorbeeld OS/2 vergelijkt valt het allemaal hard mee. Je kunt het allemaal net zo uitgebreid en ingewikkeld maken als je zelf wilt; een minimum systeem beslaat ca. 1 Megabyte; een totaal uitgebreid systeem kan tientallen MegaBytes in beslag nemen.

De voor- en nadelen van Linux t.o.v. Minix liggen aardig voor de hand. Om mee te beginnen: Linux is gratis, terwijl je voor Minix een 400 gulden neer moet leggen. Minix is wel kleiner en overzichtelijker, kan met een veel kleiner systeem overweg en is voor een hele hoop machines beschikbaar. Verder is Linux meer gericht op het bouwen van een bruikbaar systeem dan het (in eerste instantie) voor onderwijs bedoelde Minix. Dat uit zich bijvoorbeeld in het gebruik van de GNU C-compiler (Minix heeft een eigen compiler, hoewel voor een aantal versies ook een GNU-implementatie beschikbaar is), de memory swapper, job control, de task-grootte (64 MByte per task waar bij Minix/PC slechts 128 kB per task mogelijk is), de virtual consoles en het multi-threaded filesystem.

De verspreiding van Linux via de verschillende netwerken heeft zowel voor- als nadelen. Heb je zelf een modem, dan kun je makkelijk bij je distributiepunt komen. Een high-speed modem (9600 bps en hoger) verdient zich dan denk ik na verloop van tijd wel terug. Aan de andere kant is niet iedereen gezegend met zo'n mooi apparaat en biedt lang niet ieder BBS Linux files aan. En als je het echt goed wilt doen heb je eigenlijk ook een Internet-aansluiting nodig... Minix daarentegen is beschikbaar bij de beter technische boekhandel. Verder kun je het zaakje natuurlijk ook bij de uitgever bestellen, hetgeen de continuïteit van het produkt wel beter waarborcht dan de verspreiding via netwerken.

Conclusie

Met Linux haal je je een hele aardige UNIX binnen, die voor de gevorderde UNIX-gebruiker een aantal

hele aardige features biedt. Goedkoop is het systeem natuurlijk ook, hoewel je wel rekening moet houden met hoge(re) telefoonkosten en de relatief dure hardware. Linux lijkt me een uitstekende basis voor kennismaking met UNIX, mits je zelf al de benodigde apparatuur hebt. Minix blijft natuurlijk voor de KGN belangrijker dan Linux omdat de KGN68k/30 onder dit OS zal gaan lopen. De concurrentie voor Minix is echter met de komst van Linux behoorlijk toegenomen. Waar de Minix-aanhangers van het Coherent baksel weinig te duchten hebben gehad zal deze nieuwe kloon een geduchte concurrent worden.

Joost Voorhaar

Software overzicht

Voor een minimaal Linux systeem zijn de volgende bestanden nodig (zie de Linux-area van "The Ultimate"):

File	Inhoud
Boot095a.Zip	Boot image (de kernel)
Root095a.Zip	Het standaard root systeem
Inst-011.Zip	Installatie instructies voor 0.11, ook nodig voor 0.95
Rawrite.Zip	Dos utility om image files naar schijf te schrijven
RelNo095a.Zip	Release notes voor versie 0.95

In totaal is dit ca. 600 kByte groot. Naast deze files zijn er nog behoorlijk wat andere programma's en bestanden beschikbaar. Zo kan ik de GNU C-compiler, assembler en debugger aanraden (enige megabytes helaas). En dan is er natuurlijk nog de overvloed aan GNU-implementaties van allerlei verschillende UNIX-utilities, de emacs editor, TeX, UUCP, vi, Roff en zelfs een prolog compiler ontbreekt niet!

Systeem eisen

- 80386SX of hoger ('386/16 aanbevolen)
- 2 MByte RAM (4 MByte aanbevolen)
- 1.2 of 1.4 MByte diskette station
- MFM/IDE of SCSI harde schijf, 20 MB (60 MB aanbevolen)
- Hercules, CGA, EGA of (S)VGA adapter

Pluspunten

- Gratis
- GNU ontwikkel omgeving
- Memory paging, multi threaded file system, job control etc.
- 64 MByte per task (X-Windows implementatie is mogelijk!)
- Source beschikbaar

Minpunten

- Dure hardware noodzakelijk
- Geen centraal geregeld distributie centrum
- Kernel ondersteunt geen message passing
- Nog niet zo "volwassen" als Minix
- (Vrijwel) geen documentatie

Een snelle assembler met behulp van hash en chaos

Inleiding

Ik hoop dat de titel die boven dit artikeltje staat op zijn minst uw aandacht getrokken heeft, tenslotte willen alle auteurs graag dat de produkten van hun huisvlijt gelezen worden. Hoewel de titel het tegendeel doet vermoeden, zal dit echt een serieus artikel worden. Hash slaat namelijk niet op de beroemde (beruchte?) soft drug uit de jaren 60 maar is een verwijzing naar een techniek waarbij orde omgezet wordt in chaos en die onder andere gebruikt kan worden voor het ontwikkelen van een database.

Zoals ik heb aangegeven, wilde ik mijn activiteiten wat gaan verleggen naar KGN-68k. Binnen deze werkgroep wil ik mij met name bezig gaan houden met de software-kant van het project. Dat is ondertussen ook gebeurd en ik ben momenteel bezig met het schrijven van een cross-assembler waarmee we 68000, -010, -020 en -030 code kunnen genereren voor bijvoorbeeld de monitor en de device drivers.

Hash slaat namelijk niet op de beroemde (beruchte?) soft drug uit de jaren 60.

Ik ben, in overleg met de werkgroep, begonnen aan een assembler in 'C' die in staat moet zijn code voor de hele 68000 reeks, inclusief de floating point co-processor en de memory management unit te genereren. Ik heb in deze assembler meteen een aantal ideeën verwerkt waardoor de assembler beter en sneller zou moeten worden dan andere assemblers en over één van deze ideeën gaat dit artikel.

De opcode-tabel en labellijst

Een assembler werkt meestal twee keer de hem aangeboden source door. Deze doorgangen worden in het vak-jargon "passes" genoemd. Gedurende de eerste doorgang tracht de assembler al zo veel mogelijk fouten op te sporen en aan te geven zodat de gebruiker (of de assembler zelf) kan beslissen of een tweede doorgang wel zinvol is. Verder is het hoofddoel van de eerste doorgang uit te zoeken wat de waarden zijn van de gebruikte labels. Bekijk bijvoorbeeld het afgedrukte stukje 68000 code.

```
 1   ffffffff                          ; Example program for HASH article
 2   ffffffff                          ;
 3   ffffffff                          ; G. van Opbroek/10-05-1992
 4   ffffffff                          ;
 5   ffffffff                  ORG     $10000      ; Origin of the program
 6   00010000          number  EQU     100         ; This is a constant
 7   00010000  6064    start   BRA.S   p_start     ; Branch to program start
 8   00010002          storage DS.B    number      ; Reserve 100 bytes of storage
 9   00010066                  ;
10   00010066  7064    p_start MOVE.L  #number,d0      ; Initialize counter
11   00010068  223cffffffff   MOVE.L  #$FFFFFFFF,D1  ; Initialize mask in D1
12   0001006e  41f900010002   LEA     storage,A0  ; A0 -- pointer to storage
13   00010074  21810800  loop  MOVE.L  D1,0(A0,D0.L)     ; Move mask to storage
14   00010078  51c8fffa        DBRA    D0,loop     ; Decr. D0 and branch while = 0
15   0001007c                  END

Symbol table :

number          $00000064   start     $00010000
storage         $00010002   p_start   $00010066
loop            $00010074
```

Fig. 1: listing van DEMO.ASM

In dit voorbeeld staat een definitie van een constante tijdens assembleren. Dit is de regel waar met woord "EQU" in staat. Het label "number" krijgt dan de waarde die uit de expressie komt die achter het woord "EQU" staat, in dit geval 100. Verder staat er een reservering voor geheugenruimte in. Dit wordt aangegeven door de aanduiding "DS" of te wel "Define Storage". Hier wordt een hoeveelheid ruimte gereserveerd waarbij achter DS een .B, .W of .L kan staan waarmee aangegeven wordt of we een hoeveelheid bytes, words (2 byte) of long words (4 bytes) willen reserveren. Tenslotte staan in het programma twee labels die weer terugkomen in sprongopdrachten BRA (Branch Always) en DBRA (Decrement and Branch Always). Nu is het de bedoeling van pass 1 van de assembler dat de waarde van alle labels bepaald wordt zodat gedurende pass 2 de juiste waarden ingevuld kunnen worden. Dit betekent dat er in pass 1 een zogenaamde labellijst opgebouwd wordt waarin voor elk label staat welke waarde dit label heeft. Gedurende pass 2 kan de assembler uit deze lijst dan de waarden van de labels halen. Aan het einde van de listing wordt deze labellijst als zogenaamde "Symbol Table" afgedrukt.

Iets dergelijks gebeurt ook met de instructies zelf. Een 68000 heeft vele instructies in diverse varianten. Al deze instructies worden over het algemeen ook in een tabel gezet met daarbij enkele kenmerken van die instructie. Dit noemen we de opcode-tabel.

Zo zou je ook (in een domme bui) kunnen zeggen dat je 260 pagina's reserveert voor mensen waarvan de naam begint met A, 260 voor B, ... 260 voor Q ... 260 voor X ... en 260 voor Z.

Het zoeken in tabellen

Als je een tabel hebt opgesteld met daarin bijvoorbeeld alle instructies van een 68000 kun je daar op diverse manieren in gaan zoeken. In de eerste plaats kun je natuurlijk altijd van boven naar beneden de tabel doorlopen net zo lang tot je de gezochte instructie gevonden hebt. In de praktijk is deze methode alleen handig als je een kort lijstje hebt. Gemiddeld moet je namelijk altijd de halve tabel doorwerken voordat je het gezochte gevonden hebt zodat je 0.5 * N vergelijkingen moet doen (waarbij N de lengte van de tabel is). Deze methode zullen we "Lineair zoeken" noemen.

Een tweede mogelijkheid die je toe kunt passen is het zogenaamde "Binaire zoeken". In dat geval moet de lijst gesorteerd zijn, in ons voorbeeld dus op alfabet. Als nu de lengte van de lijst weer N is, dan kijk je eerst naar de opcode die precies in het midden staat, dus 0.5 * N. Is de gevonden waarde groter dan de gezochte, dan zit de gezochte waarde in de eerste helft van de lijst. Vervolgens ga je kijken op een kwart etc. Je halveert dus iedere keer het gebied waar je in moet zoeken. Op deze manier vindt je het gezochte element na maximaal LOG(N) waarbij LOG de logaritme met grondtal 2 is. Uit een lijst met 1024 elementen vindt je zo na maximaal 10 vergelijkingen je gezochte opcode. Zeer belangrijk is in dit geval dat je lijst gesorteerd is.

Het "Binaire zoeken" is een hele handige methode in die gevallen waarin de lijst van tevoren vast ligt zodat ze eerst gesorteerd kan worden. In de praktijk is dit niet altijd het geval. Neem als voorbeeld een labellijst. Deze wordt opgebouwd aan de hand van de labels die in de source staan. Deze labels zijn uiteraard niet gesorteerd zodat je ervoor moet zorgen dat de lijst tijdens het opbouwen gesorteerd wordt of je sorteert hem na pass 1 van de assembler. Zowel voor het opbouwen en gesorteerd houden van een lijst als voor het naderhand sorteren van een lijst zijn een groot aantal technieken. Voor mensen die daar meer over willen weten verwijs ik naar de literatuur en dan met name naar literatuur over lineaire lijsten, binaire bomen en diverse in- en externe sorteertechnieken.

In dit artikel wil ik een derde methode bespreken die vooral erg interessant is bij grote tot zeer grote lijsten die niet gesorteerd zijn. De snelheid van deze methode is namelijk in het geheel niet afhankelijk van de grootte van de lijst. Gemiddeld zul je bij deze techniek na zo'n 1,1 vergelijkings-operaties het gezochte element in de lijst vinden. Dit is de techniek die aangeduid wordt met de term "Hashing" en die in de rest van dit artikel wordt behandeld.

Het telefoonboek

Hashing is een techniek die bijvoorbeeld heel goed toegepast zou kunnen worden voor het maken van een elektronisch telefoonboek. Zoals iedereen weet, begint het telefoonboek van bijvoorbeeld Amsterdam met iemand waarvan de naam begint met Aa... en eindigt ongetwijfeld met iemand die Zy... heet. Dit is dus een gesorteerde lijst. Als je nu meneer N. Negenvoet zoekt, dan slaan de meeste mensen het telefoonboek zo'n beetje in het midden open en kijken welke namen daar staan. Vervolgens sla je enkele tientallen pagina's voorruit of terug etc. tot je de

juiste pagina gevonden hebt. Dan zoek je op de pagina de juiste kolom en vervolgens de juiste regel. Dit is dus zo'n beetje het binaire zoeken.

Het nadeel van een dergelijk telefoonboek is dat als er iemand bijkomt (laten we aannemen Questodor), we alles wat na deze persoon komt moeten gaan verschuiven. Zo komt het dus dat je elk jaar ergens anders in het telefoonboek staat. Wat je beter zou kunnen doen, is dat je bijvoorbeeld in elke kolom wat vrije ruimte houdt om mensen toe te voegen. Je blijft dan toch meestal op dezelfde bladzijde staan en meestal ook altijd op dezelfde plaats. Zo zou je ook (in een domme bui) kunnen zeggen dat je 260 pagina's reserveert voor mensen waarvan de naam begint met A, 260 voor B, ... 260 voor Q ... 260 voor X ... en 260 voor Z. Het voordeel is dat je al veel sneller kunt zoeken. Zeker als je zegt 10 voor Aa, 10 voor Ab etc. Meneer Baal komt dan altijd terecht op één van de pagina's 261 t/m 270. Hier komt echter een groot nadeel van de wijze waarop namen vergeven worden naar voren. Er zijn veel meer mensen waarvan de naam begint met Aa dan waarvan de naam begint met Qq. Het telefoonboek zal dus veel lege pagina's hebben bij de Q en weinig bij de A. Kortom je verspilt heel veel ruimte. De indeling wordt als het ware opgelegd door de letter waarmee de naam van de meeste mensen begint. Op deze manier is het telefoonboek voor meer dan de helft leeg, zonde van het papier dus.

In een computer zou je hetzelfde kunnen doen. Je zegt dat je namen van maximaal 20 letters op kan slaan. Vervolgens zeg je dat je voor elke mogelijke naam 256 byte reserveert. Dit betekent dat je 26 tot de macht 20 maal 256 byte = heel veel schijfruimte nodig hebt. Zeer onpraktisch dus. Het voordeel is wel dat je precies weet waar iedereen staat en dat je dus niet hoeft te zoeken. Wat je echter ook zou kunnen doen, is dat je aan iedereen een nummer van bijvoorbeeld 6 cijfers toekent. Hoe je dat doet is nu even niet interessant. Deze 6 cijfers geven de index in een tabel van 10.000.000 records van 256 byte met daarin de informatie van de betreffende persoon. Op deze manier krijg je iets dat al hanteerbaar is, je houdt nu namelijk nog 2.500 MB aan informatie over, nog steeds niet niks maar wel voldoende om heel veel gegevens van alle bewoners van Amsterdam en de verre omtrek in op te slaan. Blijft natuurlijk de vraag hoe je aan dat getal van 7 cijfers komt.

Hashing

De vorige paragraaf eindigde met de vraag hoe je aan het getal van 7 cijfers komt. Dat is eigenlijk niet zo belangrijk mits het getal maar afgeleid kan worden uit de zogenaamde "key" of "sleutel" waarop je wilt zoeken, in dit geval dus de naam van de persoon. Je krijgt dan een zogenaamde Hash-functie waar je de naam in stopt en die het getal van 7 cijfers als uitkomst heeft. Wat wel belangrijk is, is de voorwaarde dat als je het hele telefoonboek van Amsterdam er doorgehaald hebt niet alles in de eerste helft van je tabel terecht komt. De verdeling van informatie over de tabel moet zo goed mogelijk homogeen zijn ondanks het feit dat de verdeling over de letters niet homogeen is. Hier komt dus de chaos om de hoek kijken. De functie zou er eigenlijk voor moeten zorgen dat een bepaald persoon op een willekeurige plaats in de tabel terecht komt, maar wel altijd op dezelfde plaats. De tabel zal dus over het algemeen niet meer op alfabet gesorteerd zijn.

Hier komt dus de chaos om de hoek kijken.

Waarom een homogene spreiding belangrijk is, blijkt uit de beschrijving van het begrip "collisions". Hoe goed een hash-functie ook is, hij kan nooit voorkomen dat twee verschillende personen op dezelfde plaats in de tabel terechtkomen. Dit zijn de zogenaamde "collisions" of botsingen. Zowel Jantje als Pietje krijgen index 1234567 toegewezen. Nadat Jantje in de tabel gezet is, is er voor Pietje dus geen plaats meer. De kans dat dit optreedt is bij een goede hash-functie alleen afhankelijk van de vullingsgraad van de tabel. In de praktijk zal deze kans tot een vulling van zo'n 85% onder de 10% liggen. Bij een slechte hash-functie gaan er meer factoren een rol spelen. Zo zal er bij een hash-functie die altijd de waarde 1 oplevert al na twee elementen een collision optreden.

Hashfuncties

Wat je als hashfunctie moet kiezen hangt een beetje van de situatie af. Vaak wordt er iets gedaan met de index van de letters in de ASCII-tabel. Je zou bijvoorbeeld de ASCII-waarde van alle letters bij elkaar op kunnen tellen. Vervolgens deel je het resultaat van de optelling door de lengte van de tabel waarna de rest van de deling de index in de tabel oplevert. Het nadeel van deze hash-functie dat hij alleen goed werkt als je een korte tabel hebt (tot zo'n 100 plaatsen) of hele lange namen. Je krijgt namelijk pas een goede verdeling als het resultaat voor de deling meerdere malen de lengte van de tabel oplevert.

In het programma-voorbeeld wordt dit probleem opgelost door in de hash-functie het tussenresultaat

steeds met een bepaald getal te vermenigvuldigen. Dit wordt uit snelheids-overwegingen gedaan d.m.v. een schuif-operatie maar kan natuurlijk ook d.m.v. een echte vermenigvuldiging gedaan worden.

Afhandeling van collisions

Zoals al is aangegeven kunnen er bij het vullen van de tabel collisions optreden. Het is natuurlijk niet de bedoeling dat het element waarbij de collision optreedt niet opgenomen wordt in de tabel. Er zijn een aantal manieren om deze situatie af te handelen. In de eerste plaats kun je bijvoorbeeld een tweede hash-functie aanroepen. Je zou bijvoorbeeld de vermenigvuldigingsfactor kunnen wijzigen. Ook kun je vast getal bij de index optellen en de deling opnieuw uitvoeren. Als je dat goed doet, zul je, zolang de tabel niet geheel gevuld is, altijd een lege plek vinden.

Een andere mogelijkheid is dat je op de plaats waar je de informatie kwijt wilt een lijstje aanlegt van informatie die daar staat. Deze lijst is dan uiteraard vele malen korter dan de oorspronkelijke lijst. Dit is gedaan in het programma-voorbeeld. Een plek in de tabel is niet meer dan een verwijzing naar een lijst. Is de verwijzing niet gevuld, dan is het een lege plek, is de verwijzing gevuld, dan wijst deze pointer naar de informatie. In het brokje informatie, het zogenaamde record, staat weer een dergelijke verwijzing.

Vullen van de tabel en terugzoeken

Bij hashing wordt de grootte van de tabel van tevoren vastgelegd. Een goede waarde is 1,1 à 1,2 maal het maximale aantal elementen dat opgeslagen moet worden. In alle elementen van deze tabel wordt aangegeven dat dit element leeg is. Vervolgens worden de elementen stuk voor stuk toegevoegd. Hierbij wordt de hash-functie op de sleutel losgelaten waarna de hash-functie de index in de tabel oplevert. Nu zal het in de meeste gevallen zo zijn dat het element op die plaats in de tabel nog leeg is zodat dit element zondermeer gevuld kan worden. Is het element al wel gevuld, dan kan het zijn dat je twee maal dezelfde key toe wilt voegen. Dat kun je controleren door de key van het element in de tabel te vergelijken met de key van het element dat je wilt toevoegen. Is dit niet het geval, dan heb je te maken met een collision en moet je op een andere manier een lege plek zoeken. Zo kun je bijvoorbeeld eens gaan kijken of het eerstvolgende element in de tabel gevuld is en indien dit niet het geval is, deze gebruiken.

Het terugzoeken gaat op precies dezelfde manier als het toevoegen. Je haalt de sleutel die je zoekt weer door de hash-functie die je een index oplevert waar het gezochte element zou moeten staan. Is dit element in de tabel niet gevuld, dan komt het gezochte element niet voor in de tabel; de gezochte persoon staat niet in het telefoonboek. Is het element gevuld, dan moet je de sleutel van het element in de tabel vergelijken met de gezochte sleutel. Zijn ze gelijk, dan heb je gevonden wat je zocht. Zijn ze niet gelijk, dan zoek je verder op dezelfde manier als die waarop je collisions afhandelt net zo lang tot je het gezochte element gevonden hebt of totdat je stuit op een leeg element. In dat laatste geval komt het gezochte element niet voor in de tabel.

Praktijkvoorbeeld

In de bijgevoegde listings staat een praktijkvoorbeeld van hashing. De sources zijn een klein deel van de KGN-68k assembler die in het kader van de projectgroep ontwikkeld wordt. Hoewel de assembler nog lang niet af is, zijn de getoonde programma-delen uitgebreid getest en zijn ze redelijk foutvrij.

De eerste listing bevat de zogenaamde "Header file" KGN68k.h. Hierin staan een deel van de globale declaraties van de assembler. Waar ik vooral de aandacht op wil vestigen zijn de datastructuren t_opcode_list en t_id_list. Dit zijn de elementen in respectievelijk de opcode-tabel en de labellijst. In een dergelijk element is de sleutel opgeslagen en wat extra gegevens zoals type cpu waarvoor de opcode geldt en de waarde van het label. Verder zijn er verwijzingen (pointers) naar het volgende en het vorige element en bij de labellijst een verwijzing waarmee de lijst opgebouwd wordt gesorteerd volgens de volgorde waarin de labels toegevoegd worden. Deze laatste lijst wordt later gebruikt om de symbol table in volgorde van definitie af te kunnen drukken.

De eigenlijke opcode-tabel en labellijst bestaan uit de array's van pointers die resp. opcode_table en ident_table genoemd zijn. Bij de start van het programma worden alle pointers op NULL geïnitialiseerd.

De verdere afhandeling van de tabellen staat in de source TABLES.C. Hierin staan een aantal routines die van buiten aangeroepen kunnen worden te weten: initialize_opcode_table waarmee van een externe file de opcode-tabel ingelezen wordt, get_opcode waarmee een opcode opgevraagd kan worden, add_identifier waarmee een label toegevoegd kan worden en uiteraard get_identifier waarmee een label teruggezocht wordt.

Uiteraard staat ook de gebruikte hash-functie in de source die eigenlijk zeer eenvoudig is.

In het programma-voorbeeld worden collisions afgehandeld met behulp van een lineaire linked list. De opcode-tabel bestaat niet uit een array van records

maar uit een array van pointers naar die records. Als een element in de tabel niet gevuld is, dan bevat deze de pointer de waarde NULL. Is het element wel gevuld, dan bevat de pointer uiteraard het adres van dit record. In het record zelf zijn de pointers prev en next opgenomen die bij het aanmaken van het record met NULL gevuld worden. Mochten we nu te maken krijgen met een collision, dan lopen we de keten van records langs de pointer next af totdat we aangeland zijn in een record waarvan next de waarde NULL heeft. Vervolgens schrijven we in deze next het adres van het toegevoegde record en in de prev pointer in het toegevoegde record het adres van het tot dan laatste record in de keten. Deze techniek heeft een aantal voordelen. In de eerste plaats kunnen we nu meerdere malen dezelfde key gebruiken wat in de opcode-tabel ook zeker voor zal komen. Verder kunnen er, ondanks de beperkte grootte van de array van pointers, toch zoveel records in de tabel opgenomen worden als waarvoor we geheugen hebben. Het zoeken zal echter wat langer gaan duren omdat we in de lineaire lijsten, die niet gesorteerd zijn, alleen lineair kunnen zoeken.

Afsluiting

De techniek van hashing is in het programma-voorbeeld ingezet bij de ontwikkeling van een assembler. Dit is uiteraard niet de enige toepassing van hashing. Zo heb ik bijvoorbeeld in het verleden eens een soort kaartenbak-programma in Forth geschreven. De insiders weten dat in Forth een schijf opgedeeld is in blokken die aangesproken worden met behulp van hun bloknummer. Ik had een record-definitie gemaakt die precies zo groot was als een Forth blok op schijf. Hierin stonden o.a. naam, adres en woonplaats en nog wat andere gegevens. Vervolgens werd het bloknummer bepaald met behulp van een hashfunctie die toegepast werd op de naam. Op deze manier was bekend in welk blok de gegevens geschreven moesten worden of konden worden opgehaald. Ik zal voor de geïnteresseerden dit programma, als ik het terug kan vinden, uploaden naar The Ultimate.

Verder heb ik iedereen even in de keuken van de software groep van KGN-68k laten kijken. Mochten er mensen zijn die zich graag bij deze groep aan willen sluiten, laten die dan contact opnemen met Geert Stappers. Een beetje ervaring in het programmeren in C of 68000 assembler is dan wel zeer gewenst.

Gert van Opbroek

```
/* KGN68k_AS.H:
      Header file with global declarations.

      KGN68k assembler.

      Created by : G. van Opbroek/KIM Gebruikersclub Nederland
      Date       : March 1992


      Copyright (c) 1992: KIM Gebruikersclub Nederland
*/

#define C_OPCODE_TABLE      1023      /* Number of entries in the opcode table */
#define C_IDENT_TABLE       2047      /* Number of entries in the identifier table */
#define C_MAX_MNEMONIC      12
#define C_MAX_ARGS          12
#define C_MAX_IDENT         20
#define C_MAX_OPERANDS      5     /* Maximum number of operands in an instruction */
#define OPCODE_TABLE        "KGN68K.TBL"
#define FALSE               0;
#define TRUE                1;
#define LINES_PAGE          50        /* Maximun number of lines per page */
```

Fig. 2: Listing van KGN68k.h

```
/* Define constants for the options */

#define OPTIMIZE    1       /* Optimize code          */
#define  LIST       2       /* List                   */
#define  QUIET      4       /* Don't print warnings   */
#define  SYMBOL     8       /* Print symbol table     */
#define EXACT       16      /* Don't translate to MOVEA etc.*/

/* Define the separators/terminators for an operand */

#define separator(a) (a==' '||a=='\t'||a==','||a==';'||a=='\0'||a=='\n')

typedef struct t_oc_list
    { int index;
        char                mnemonic [C_MAX_MNEMONIC + 1];
        short unsigned      cputype;
        long unsigned       opcode;
        char                args [C_MAX_ARGS + 1];
        struct t_oc_list    *prev;
        struct t_oc_list    *next;    /* Used in case of collisions */
    } t_opcode_list ;

typedef struct t_id_list
    { int           index;
        char        identifier  [C_MAX_IDENT + 1];
        short       valid;
        int         line_number;        /* Definition line number */
        long unsigned       value;
        struct t_id_list    *prev;
        struct t_id_list    *next;    /* Used in case of collisions */
        struct t_id_list    *sort;    /* Link for sorted identifiers */
    } t_ident_list ;

typedef struct t_ex_result
    { long    result;
        int   valid;            /* Result is valid          */
        int   is_signed;        /* Result is signed/unsigned */
        int   highest_line;     /* Used for forward references */
    } t_valid ;

/* Global data structures from source file KGN68k      */

extern    int   error_count,
                line_count,
                assembler_fase;
extern unsigned     CPU_type,
                    options;
extern unsigned long    location_counter;

extern char    *assembler_fases[];

extern t_opcode_list        *opcode_table    [C_OPCODE_TABLE];
extern t_ident_list         *ident_table  [C_IDENT_TABLE];
extern t_ident_list         *first;
extern t_ident_list         **sort;
```

```
extern FILE    *source ,
               *list ,
               *bin ;
extern char    bin_out[255];

/* Routines from source file TABLES */

short int hash(char *string, int number);
int initialize_opcode_table(void);
int get_opcode(char *mnemonic, t_opcode_list   **element) ;
int add_identifier(char *identifier, unsigned long value, short valid);
int get_identifier(char *identifier, t_ident_list **element) ;

/* Routines from source file EXPRESS  */

int expression(char **string, long *partial_result, t_valid *valid);

/* Routines from source file ASSEMBLE  */

int assembler(int pass);

/* Routines from source file PSEUDO */

int pseudo_opcode(char **string,char *result);

/* Routines from source file OPCODE              */

int opcode(char **string, char *result);

/* Routines from source file ADDRESS             */

int operand_type(char **string, int operand,
        int *mode, int *reg, t_valid *number);

/* Routines from source file S_records                    */

int s_format(int type, char *bin_out, char *string, FILE *bin);
int s_print (char *string, FILE *bin);
```

```
/* TABLE.C:
      Handle opcode and identifier tables.

      KGN68k assembler.

      Created by : G. van Opbroek/KIM Gebruikersclub Nederland
      Date : March 1992

      Copyright (c) 1992: KIM Gebruikersclub Nederland
*/

#include <stddef.h>
#include <stdio.h>
#include <alloc.h>
#include <string.h>
#include "kgn68k.h"
#include "error.h"

static short int hash(char *string, int number)

{ short unsigned result = 0,
                 index  = 0;

     while (string[index]!='\0')
     result = ((result << 3 ) + (int) string[index++]) % number;
     return result ;
}

int initialize_opcode_table(void)

/* initialize the opcode table from OPCODE_TABLE
 * parameters : none
 * return  : error = TRUE/FALSE
 */

{ unsigned long    ul;
     FILE       *table;
     char       string[132],          /* should be enough */
                 *p,*s;
     t_opcode_list  *element,*q;
     int        i,
          error = FALSE;

     /*
     * Initialisation of the global tables
     */

     p = OPCODE_TABLE;
     table = fopen(p,"r");
     if (table == NULL)
     error = process_error(F_OPC_NF,&p);

     if (!error)
```

Fig. 3: sourcetekst van TABLES.C

```
while (fgets(string,132,table) != NULL)
{ if (string[0] == '"')          /* This is an opcode definition */
    { element = (t_opcode_list *) malloc(sizeof(t_opcode_list));
    if (element == NULL)
        error = process_error(F_OPC_MEM,&p);
    else
    { /* clear opcode element                    */

        element->index    =
        element->cputype  =
        element->opcode   =
    element->mnemonic[0]      =
    element->args          [0]  = '\0';
    element->prev          =
    element->next     = NULL;

        /* copy mnemonic                         */

        i                 = 0;
    p                 = string + 1;
    s                 = element->mnemonic;    /* For errors */
    while (*p != '"' &&  *p != '\0'&& i < C_MAX_MNEMONIC)
    element->mnemonic[i++] = *p++;
        element->mnemonic[i] = '\0';
        if ((i >= C_MAX_MNEMONIC && *p != '"') || *p == '\0')
        error = process_error(F_OPC_MNM,&s);

    /* calculate index with the hash function      */

        element->index = hash(element->mnemonic,C_OPCODE_TABLE);

    /* copy CPU type                               */

    while ((*p < '0' ||  *p > '9' ) && *p != '\0') p++;
    if (*p == '\0')
    error = process_error(F_OPC_CPU,&s);
        else
        if (*p == '0' && tolower(*(p + 1)) == 'x')
         sscanf(p,"%x",&(element->cputype));
        else
        if (*p == '0')
         sscanf(p,"%o",&(element->cputype));
        else
         sscanf(p,"%d",&(element->cputype));

    /* copy opcode                                 */

        for (i = 1 ; i <= 2 ; i++)
        { while (*p != ',' && *p != '\0') p++;
    while ((*p < '0' ||  *p > '9' ) && *p != '\0') p++;
    if (*p == '\0')
              error = process_error(F_OPC_OPC,&s);
        else
        if (*p == '0' && tolower(*(p + 1)) == 'x')
         sscanf(p,"%lx",&ul);
    else
```

```
        if (*p == '0')
            sscanf(p,"%lo",&ul);
        else
            sscanf(p,"%ld",&ul);
        if (!error) element->opcode = (element->opcode << 16 ) + ul;
        }

        /* copy arguments                          */

            i     = 0;
            while (*p != '"' && *p != '\0') p++;
            if (*p == '"') p++;
        while (*p != '"' &&  *p != '\0'&& i < C_MAX_ARGS)
        element->args[i++] = *p++;
            element->args[i] = '\0';
        if ((i >= C_MAX_ARGS && *p != '"') || *p == '\0')
            error = process_error(F_OPC_ARG,&s);

        /* insert opcode element in the data structure    */

            if (opcode_table[element->index] == NULL)
             opcode_table[element->index] = element;
            else
            { q = opcode_table[element->index];
        while (q->next != NULL) q = q->next;
            q->next = element;
            q->prev = q;
            }
    };

        }
    }
    return error == FALSE;
}

int get_opcode(char *mnemonic, t_opcode_list   **element)

/* Get an opcode from the opcode table
 * parameters : *mnemonic  : mnemonic
 *              *element      : pointer to the opcode element
 * return  : error = TRUE/FALSE
 */

{ t_opcode_list     *q;
    int             index,
                    error = FALSE ;

    *element = NULL;
    index    = hash(mnemonic,C_OPCODE_TABLE);
    q        = opcode_table[index] ;

    /* search opcode element in the data structure  */

    while( q != NULL                          &&
            (strcmp(q->mnemonic,mnemonic) != 0 ||
            (q->cputype & CPU_type) == 0 ||
```

```
            q->cputype & 1<<15 &&
            options   & EXACT))
        q = q->next;

        if (q == NULL || strcmp(q->mnemonic,mnemonic) != 0)
            error = process_error(E_OPC_NF,&mnemonic);

        *element = q;
        return error == FALSE;
}
int add_identifier(char *identifier, unsigned long value, short valid)

/* Add an identifier to the identifier table
 * parameters : *identifier     : identifier
 *              value           : value of the identifier (if valid)
 *              valid       : value = valid TRUE/FALSE
 * return  : error = TRUE/FALSE
 */

{ t_ident_list   *element,*q;
    int             error = FALSE;

    element = (t_ident_list *) malloc(sizeof(t_ident_list));
    if (element == NULL)
    error = process_error(F_IDN_MEM,&identifier);
    else
    { strcpy(element->identifier,identifier);
    element->index    = hash(identifier,C_IDENT_TABLE);
    element->valid    = valid;
    element->value    = value;
    element->line_number = line_count;
    element->prev          = element->next =element->sort = NULL;

    /* insert identifier element in the data structure   */

    if (ident_table[element->index] == NULL)
    { ident_table[element->index] = element;
        *sort      = element;
        sort  = &element->sort;
    }
    else
    { q = ident_table[element->index];
        while (q->next != NULL && strcmp(q->identifier,identifier) != 0)
    q = q->next;
        if (strcmp(q->identifier,identifier) == 0)
        { error = process_error(W_IDN_DUP,&identifier);
        q->valid = valid;
        q->value = value;
        free(element);
        }
        else
        { q->next = element;
        q->prev = q;
        *sort      = element;
        sort  = &element->sort;
        }
```

```
        }
        }
        return error = = FALSE;
}

int get_identifier(char *identifier, t_ident_list  **element)

/* Get an identifier from the identifier table
 * parameters : *identifier     : identifier
 *              *element        : pointer to the identifier element
 * return  : error = TRUE/FALSE
 */

{ t_ident_list   *q;
        int             index,
                        error = FALSE ;

        *element = NULL;
        index    = hash(identifier,C_IDENT_TABLE);
        q        = ident_table[index] ;

        /* search opcode element in the data structure   */

        while( q != NULL && strcmp(q->identifier,identifier) != 0)
        q = q->next;

        if (q = = NULL || strcmp(q->identifier,identifier) != 0)
        if (assembler_fase < = 1)
                error = process_error(W_IDN_NF,&identifier);
        else
                error = process_error(E_IDN_NF,&identifier);

        *element = q;
        return error = = FALSE;
}
```

Systeem informatie via 4DOS

De hier gepresenteerde 4DOS-4.0 batchfile kan gebruikt worden om informatie over uw computersysteem te verkrijgen. Daarnaast is het ook een aardig voorbeeld van de extra mogelijkheden die 4DOS biedt boven de nogal saaie command-interpretor van MS-DOS, Command.Com. De getoonde informatie kan op het scherm getoond worden, maar kan ook naar printer (hij controleert dan eerst even of de printer aanstaat) of logfile geschreven worden.

Het begon allemaal toen ik wat informatie over mijn systeem wilde hebben op de tijden dat mijn mailer begon met zijn dagelijkse werk. Dat is de reden dat de informatie naar logfile geschreven wordt als de printer niet aanstaat; het is natuurlijk mogelijk dat ik vergeet om 's avonds de printer aan te zetten of dat het papier op is... Daarnaast maakte ik een utility die de FrontDoor logfile even schoonmaakt voordat dat ding afgedrukt werd. Zo hoef ik de computer niet eens aan te zetten om te zien wat er 's nachts allemaal gebeurd is! Daarnaast is zo'n logboek verdraaid handig voor als ik eens niet thuis ben en iemand anders op m'n systeem past...

Ik sta natuurlijk altijd open voor suggesties en aanpassingen. Stuur uw commentaar naar Wim Janssen op The Ultimate (Fidonet point 2:512/32.9). Ik draag er dan zorg voor dat deze batchfile netjes up-to-date blijft en dat anderen ook nog gebruik kunnen maken van uw verbeteringen!

Wat algemene informatie:

Programma : SysInfo.Btm, versie 3.0
Auteur : Wim Janssen (2:512/32.9)
Benodigde programmatuur: 4DOS 4.0 B1 of 4OS2 0.95

Type "SysInfo /?" voor hulp over de parameters.

Wim Janssen

Noot van de redactie: SysInfo.Btm is geschreven door een point van het BBS "The Ultimate". Het programmaatje werkt heel aardig, maar je moet wel een hele bult extra environment ruimte definiëren om de zaak aan de praat te helpen. De originele archive is verkrijgbaar op The Ultimate onder de naam "4D-SI30.Zip".

```
@Rem ------------------------------------------------------------------------
@Rem          *** 4Dos/4Os2 System Information  ***
@Rem             By: Wim Janssen (2:512/32.9)
@Rem ------------------------------------------------------------------------
@Rem 11-4-92 1.0   Original [4DOS] ( Message Area:4Dos ).
@Rem 17-4 2.0  (*) Update to [4Os2 0.95]   By: Marcel Stikkelman (2:512/4)
@Rem           New Logging Routine
@Rem 22-4 2.1ß (*) Update @Select
@Rem 23-4 2.1  (*) Alias 'Display'                 By: Marcel Stikkelman
@Rem 25-4          No @Unique %InfoFile anymore!
@Rem           'Display' has become '#'
@Rem           New functions/variables added
@Rem 28-4          Asking for a FormFeed on the printer
@Rem           @ added that where missing in the end
@Rem 29-4          Inkey routine changed
@Rem 30-4          Alias '#' with printer corrected
@Rem           Coprocessor routine altered
@Rem 01-5          Added Echo, Log, Swapping, LoadBtm, Verify and Break
@Rem           LayOut Changed
@Rem 01-5 3.0      Upload: "The Ultimate"
@Rem                  ( Node1 2:512/32 - Node2 2:512/308)
@Rem         (*) = Not Released
@Rem  @Rem ..//..! = Edit next lines to one line
```

Fig. 1: de batch-file SYSINFO.BTM

```
@Rem ------------------------------------------------------------------

@SetLocal
@Set Version=SYSINFO (Version 3.0)
@Set Author=By: Wim Janssen (2:512/32.9)
@Rem = = = My Own _LastDisk is always a RamDisk! = = =
@Rem = = =  That's why, I can keep 4D-SIxxx.Btm! = = =
@Set TempFile=%@Unique[%_LastDisk:\]
@Set InfoFile=%_LastDisk:\4D-SI%DayNr.Btm

@Rem = = = No Piping Into Input Under OS/2 = = =
@Rem = = =   This Has To Be Redirection  = = =
@Echo > %TempFile
 @Input %%E_IO < %TempFile
 @If "%E_IO" = = "ECHO is ON" Echo Off
Log  > %TempFile
 Input %%L_IO < %TempFile
 If "%L_IO" = = "LOG is ON"  Log  Off

@Rem = = = Check Parameters.. = = =
If "%1" NE "" .AND. "%1" NE "Prn" .AND. "%1" NE "Log" (
  Gosub Help
  Goto Out
  )

@Rem = = = Check for DOS/Windows or OS/2 = = =
Iff "%_Win" EQ "" Then
 Set Escape=^ ^
 Set OsType=OS2
@Rem  Not Used!
@Rem  Set Compound='&'
Else
@Rem  Set Escape=[Ctrl-X][Ctrl-X] !!
 Set Escape=
 Set OsType=DOS
@Rem  Not Used!
@Rem  Set Compound='^'
Endiff

@Rem = = = Some other useful variables = = =
Set Spc='          '
Set Length=8
Set Fill=%%@SubStr[%%Spc,1,%%@Eval[%%Length-%%@Len

@Rem = = = These lines are in my AutoExec.Bat! = = =
Set Day=%@Eval[%@Date[%_Date]-%@date[1/1/%@Substr[%_Date,1,-2]]+1]
Set DayNr=%@Substr[000,%@Len[%Day]]%Day
Set WeekNr=%@Int[%@Eval[(%DayNr+2)/7+1]]
@Rem English
Set Day=SunMonTueWedThuFriSat
Set Day=%@Word[%@Eval[%@Index[%Day,%_Dow]/3],Sun Mon Tues Wednes Thurs Fri Sa-
tur]day
Set Month=%@Word[%@Substr[%_Date,3,2],Unknown January February March April May June
July August September October November December]
@Rem Dutch
@Rem Set Day=SunMonTueWedThuFriSat
```

```
@Rem Set Dag = %@Word[%@Eval[%@Index[%Day,%_Dow]/3],zon maan dins woens donder vrij
zater]dag
@Rem Set Maand = %@Word[%@Substr[%_Date,3,2],Onbekend januari februari maart april mei ju-
ni juli augustus september oktober november december]
@Rem
= = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = = =

If Exist %InfoFile (
  Set Ans =
  InKey /W5 /K"yn" Scanning System Again? [Y/N] %%Ans
  If "%Ans" EQ "n"  Goto Show
  Goto Begin
)
Goto Begin

:VolumeLabel
Set Tmp = %@Upper[%@Label[%Drv:]]
Set Length = 11
Set Output = %Drv: %Fill[%Tmp]]]%Tmp%
Iff %OsType = = DOS Then
 Set Output = %Output [%@Removable[%Drv:]]
Else
 Set Length = 4
 Set Output = %OutPut [%@Fstype[%Drv:]%Fill[%@Fstype[%Drv:]]]]]
Endiff
Set Length = 8
Echo # %OutPut Free: %Fill[%@DiskFree[%Drv:,K]]]]%@DiskFree[%Drv:,K] KB Total:
%Fill[%@DiskTotal[%Drv:,K]]]]%@DiskTotal[%Drv:,K] KB > > %InfoFile
Return

:ChkDrv
Iff %@Ready[%Drv:] = = 1 Then
 Gosub VolumeLabel
Else
 Iff %OsType = = DOS Then
  Echo # %Drv:          [%@Removable[%Drv:]] No Disk Found! > > %InfoFile
 Else
  Echo # %Drv:          [   ] No Disk Found! > > %InfoFile
 Endiff
EndIff
Return

:Begin
Echo # %Day, %Month %@SubStr[%_Date,0,2] 19%@Substr[%_Date,6,2]  > %InfoFile
Echo # DayNumber: %DayNr  -  WeekNumber: %WeekNr              > > %InfoFile
Echo # Booting from: %_Boot [%@Label[%_Boot:]]                 > > %InfoFile
Echo # LastDisk %_LastDisk: Current Directory: %@Upper[%_CWD]   > > %InfoFile

@Rem QUESTION:
@Rem Does anyone know another(better) solution for the next 15 lines?

Ver /R > %TempFile
Set Line = %@Line[%TempFile,0]
Iff "%Line" NE "" Then
 Echo # Operating System: %Line          > > %InfoFile
Else
```

```
  Echo # Operating System: %@Line[%TempFile,1] > > %InfoFile
Endiff

Set Line = %@Line[%TempFile,2]
Iff "%Line" NE "**EOF**" Then
  Echo # %Line                    > > %InfoFile
  Set Line = %@Line[%TempFile,3]
  Iff "%Line" NE "**EOF**" Then
    Echo # %Line                  > > %InfoFile
  Endiff
Endiff

Iff %_NDP = = 0 Then Set OutPut = 'No '
Else Set Output = '80%_NDP-'
EndIff
Echo # System CPU = 80%_CPU, %Output%CoProcessor Installed  > > %InfoFile

If %_Ansi = = 1 Set OutPut =
If %_Ansi = = 0 Set OutPut = 'No '
Echo # %@Upper[%_Video] Video Card, %_Rows Screen Rows, %Output%Ansi Driver Installed
> > %InfoFile
Echo # Color is %_FG on %_BG.                  > > %InfoFile

Echo # Current Code Page Number: %_CodePage        > > %InfoFile

If %_Mouse = = 1 Set OutPut =
If %_Mouse = = 0 Set OutPut = 'No '
Echo # %Output%Mouse Driver Installed          > > %InfoFile

Set Output = 'Current Batch Level: %_Batch.'

Iff %_Shell EQ 99 Then
Echo # No Swapping Mode 4Dos. %Output          > > %InfoFile
Elseiff %_Transient EQ 1 Then
Echo # Shell Level: %_Shell (Transient). %Output  > > %InfoFile
Else Echo # Shell Level: %_Shell (Not Transient). %Output  > > %InfoFile
EndIff

Echo # Variables: Temp4Dos = %Temp4Dos , Temp = %Temp > > %InfoFile

Echo # %E_IO , %L_IO  > > %InfoFile
Echos '# '   > > %InfoFile
Swapping     > > %InfoFile
Echos '# '   > > %InfoFile
LoadBtm      > > %InfoFile
Echos '# '   > > %InfoFile
Verify       > > %InfoFile
Echos '# '   > > %InfoFile
Break        > > %InfoFile

Set Length = 8
If %OsType = = DOS (
  Echo # Alias          Free: %Fill[%_Alias]]]%_Alias Bytes       > > %InfoFile
  Echo # Environment    Free: %Fill[%_Env]]]%_Env Bytes           > > %InfoFile
  Echo # Base Memory    Free: %Fill[%@DosMem[b]]]]%@DosMem[b] Bytes > > %InfoFile
  Echo # XMS  Memory    Free: %Fill[%@XMS[b]]]]%@XMS[b] Bytes     > > %InfoFile
```

```
  Echo # EMS  Memory       Free: %Fill[%@EMS[b]]]]%@EMS[b] Bytes      > > %InfoFile
)

If %OsType = = OS2 (
  Echo # Base Memory       Free: %Fill[%@DosMem[b]]]]%@DosMem[b] Bytes > > %InfoFile
)

Set Drv = %_LastDisk

:First
If "%Drv" GE "A" Gosub ChkDrv
Set Drv = %@Char[%@@Eval[%@Ascii[%Drv]-1]]
If "%Drv" GE "A" Goto First


:Show
If "%1" EQ "" Set Device = Con

Iff %Ostype = = DOS .AND. "%1" = = "Prn" Then
  Set Device = Log
  If %@Lpt[3] = = 1 Set Device = Lpt3
  If %@Lpt[2] = = 1 Set Device = Lpt2
  If %@Lpt[1] = = 1 Set Device = Lpt1
  If "%Device" = = "Log" Log ERROR [%Version]: Printer Not Ready!
Endiff

If "%1" = = "Log" Set Device = Log

Iff "%Device" NE "Log" Then
  Iff "%Device" EQ "Con" Then
  Keystack /W360 ! Left /W1 Right /W1 Left /W1 Right  /W90 ! 27
  Set Option = %@Select[%InfoFile, 0, 0, 25, 65,[ %Version ]]
  Keystack !
  Iff "%Option" NE "" Then Alias # = 'Echo'
    %Option
    Else Echo %Version %Author
    EndIff
  Else
  @Rem My Printer Initialisation ....
  Copy /Q C:\Elite.Rx8 %Device > & > Nul

  Alias # = 'Echo %& > %Device'
  Echo *** %Version ***   > %Device
  Call %InfoFile > & > Nul
  Set Ans =
  Inkey /K"yn" /W5 Do You Want A FormFeed? [Y/N] %%Ans
  If "%Ans" EQ "y" Echo %Escape%f > %Device
  EndIff
Else
  Alias # = 'Log'
  Echo %Version writing to 4%OsType LogFile ...
  Log Executing: %@Upper[%CmdLine]
  Call %InfoFile > & > Nul
Endiff

Del /Q %TempFile
```

```
If "%L_IO" = = "LOG is ON"  Log On
If "%E_IO" = = "ECHO is ON" Echo On
@Goto Out

:Help
Cls
Echo %Version
Echo %Author
Text
[4Dos 4.0 B1]
[4Os2 0.95  ]

Parameters :

    1. 'Nothing' (to Screen)
    2. Prn     (to Printer if ready, else to 4DOS/4OS2 LogFile)
    3. Log     (to 4DOS/4OS2 LogFile)
    4. ?/Help  (This Info)
EndText
Return

:Out
@EndLocal
@Quit
```

COM-file maker

In dit blad staat regelmatig een sourcetekst van MASM/TASM, die moet uitmonden in een programmaatje van het .COM-formaat. Om dat te maken moet je het eerst assembleren. Dan heb je een .OBJ-file. Dan moet er gelinkt worden: een .EXE-file is het resultaat. Tenslotte moet dat bestand nog worden omgezet naar een .COM-file. Resultaat: je typt je compleet klem, vooral als dat allemaal niet de eerste keer goed gaat. Vandaar dit handige batch-filetje (MC.BAT) dat het allemaal keurig voor je doet:

```
@ECHO OFF
ECHO Assembleren van %1.ASM...
MASM %1 /ML /B63; > NUL
IF EXIST %1.OBJ GOTO DOLINK
ECHO Fout bij het Assembleren.
GOTO END
:DOLINK
ECHO LINKen van %1.OBJ...
LINK %1; > NUL
IF EXIST %1.EXE GOTO EXETWO
ECHO Fout bij het LINKen.
GOTO END
:EXETWO
DEL %1.OBJ > NUL
ECHO Converteren van %1.EXE naar %1.COM...
EXE2BIN %1.EXE %1.COM > NUL
IF EXIST %1.COM GOTO DELEXE
ECHO Fout bij aanmaken %1.COM
GOTO END
:DELEXE
DEL %1.EXE > NUL
:END
```

Om bijvoorbeeld TEST.ASM te assembleren tot TEST.COM, wordt er ingetypt: MC TEST. Veel gemak van dit bestandje toegewenst.

Nico de Vries

Object georiënteerd programmeren

Zo af en toe verschijnt er in de computerwereld een nieuw modewoord. Zo kun je de laatste maanden geen computerblad meer openslaan of er valt wel iets te lezen over het zogenaamde "multi-media" gebeuren. Wat dat precies is schijnt niemand te weten, maar het heeft in ieder geval te maken met de integratie van (hoge) beeld- en geluidskwaliteit op de PC (So, what's new?)...

Een jaar terug stonden de bladen vol van "object georiënteerd programmeren" (afgekort tot "OOP"). Het maakte niets meer uit of je programma wel werkte, zolang het maar object georiënteerd was gebouwd, leek het wel. In dat ene jaar bleek toch wel dat OOP méér was dan een "officiële" methode van programmeren die in feite al jaren werd toegepast. In dit artikel zal ik aan de hand van een eenvoudige functie-evaluator proberen het OOP-gebeuren (of eigenlijk: het OOP-denken) nader toe te lichten. Dit stukje programmatuur leest van het toetsenbord een expressie in en evalueert deze. Een soort eenvoudige rekenmachine dus...

Notatie methoden

Vóór ik kan beginnen met mijn uitleg over onze object georiënteerde parser zal ik eerst wat uitleggen over de werking van parsers in het algemeen. De parser accepteert, net zoals iedere normale rekenmachine dat doet, uitdrukkingen in zogenaamde infix notatie. Dat betekent dat de binaire operators (dat wil zeggen: de rekenkundige basisfuncties die betrekking hebben op twee waarden zoals de optelling en vermenigvuldiging) tussen de beide operands staan. Voor computers is het veel eenvoudiger om te werken met behulp van de zogenaamde postfix notatie of RPN (Reversed Polish Notation). Hierbij staan de binaire operators ná de operands; in plaats van "a + b" schrijven we dan "a b +". Om dit uit te rekenen maken we dan gebruik van een rekenstack waar alle tussenresultaten op bewaard kunnen worden. De uitdrukking "a b +" kunnen we dan omschrijven als:

```
push a
push b
add
```

De add-operator popt de twee getallen van rekenstack, telt ze bij elkaar op en pusht het resultaat terug op rekenstack.

Naast deze twee notaties bestaat er ook de zogenaamde prefix notatie, waarbij de operators vóór de operands staan. Deze notatie methode is uitgevonden door de Poolse logicus Jan Ljukasiewicz (1878-1956). Zijn schrijfmethode had als grootste voordeel boven de normale infix notatie dat het gebruik van haakjes niet nodig was. Hij schreef bijvoorbeeld "* + 2 3 + 4 5" in plaats van "(2 + 3) * (4 + 5)". In RPN zouden we "2 3 + 4 5 + *" schrijven. Dat het allemaal veel eenvoudiger wordt door het probleem in OOP op te lossen zal verderop blijken...

De parser tree

Om onze rekenmachine te kunnen laten rekenen moeten we ingevoerde expressie van infix notatie omzetten naar de postfix notatie. Dit gaat het eenvoudigst met een binaire boom, de zogenaamde "parser tree". De knooppunten (nodes) van deze boom bevatten steeds een operator en de bladeren bevatten steeds variabelen of constante waarden. We kunnen nu een recursieve functie schrijven die de hele boom afloopt en globaal het volgende doet:

1. Loop eerst recursief de linker sub-boom door
2. Loop recursief de rechter sub-boom door
3. Voer de knooppunt berekening uit voor de twee verkregen waarden
4. Retourneer het resultaat van de knooppunt expressie

De stopvoorwaarde voor onze recursieve functie is bereikt als er geen sub-bomen zijn. We hebben dan dus te maken met een eindpunt (een blad) in plaats van een node. Voor bladeren geldt in principe hetzelfde als voor functies; ze retourneren alleen niet het resultaat van een berekening maar de waarde van de opgeslagen constante of variabele.

Ter illustratie de eerder genoemde expressie "(2 + 3) * (4 + 5)". Deze expressie zetten we voor het gemak eerst even met de hand om in postfix notatie: 2 3 + 4 5 + *. Hieruit kan op eenvoudige wijze een parsertree gedestilleerd worden (zie figuur 1).

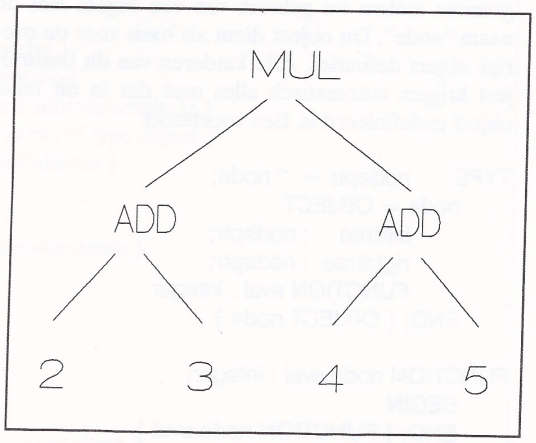


Fig. 1: De parsertree voor (2+3)(4+5)*

Object georiënteerdheid

Vroeger moest er moeizaam geknoeid worden met variant records ("unions" voor de C-freaks) die alleen herkenbaar waren aan een typeaanduiding in het record zelf. Voor het rekenwerk moest er dan een grote routine geschreven worden waarin alle bekende operators in één groot CASE-statement verwerkt konden worden. Een nieuwe operator toevoegen betekende dan een enorm geknoei in dat CASE statement. In OOP definiëren we geen variant records meer maar zogenaamde objecten. Op het eerste gezicht lijkt een object heel sterk op een record. Een object heeft echter meer mogelijkheden! Allereerst kan aan een object een stuk code gekoppeld worden. Deze code (in de vorm van een procedure of functie) is onlosmakelijk verbonden met het object. Deze functies die gekoppeld zijn aan een object worden ook wel "methods" genoemd. Het totale object zou er dan zo uit kunnen zien:

```
TYPE     add = OBJECT
           a,b  : integer;
           FUNCTION eval : integer;
   END; { OBJECT add }

FUNCTION add.eval : integer;
    BEGIN
         add.eval := a + b
    END; { FUNCTION eval }
```

Merk op dat de functiedefinitie bij de object definitie hoort! De parameters "a" en "b" hebben betrekking op de gelijknamige velden in het object.

Een ander aardig kenmerk van objecten is hun zogenaamde "overerfbaarheid". In het voorbeeld programma maken we gebruik van een object met de naam "node". Dit object dient als basis voor de overige object definities. Alle kinderen van dit basisobject krijgen automatisch alles mee dat in dit basis object gedefinieerd is. Een voorbeeld:

```
TYPE     nodeptr = ^node;
    node = OBJECT
           lefttree   : nodeptr;
           righttree  : nodeptr;
           FUNCTION eval : integer;
    END; { OBJECT node }

FUNCTION node.eval : integer;
    BEGIN
    END; { FUNCTION node.eval }
```

```
TYPE     add = OBJECT(node)
           FUNCTION eval : integer;
    END; { OBJECT add }

FUNCTION add.eval : integer;
    BEGIN
         add.eval := lefttree^.eval + righttree^.eval
    END; { FUNCTION add.eval }
```

Fig. 2: kind van het basis-object

De functie node.eval heeft geen code. Hij is er slechts om aan te geven dat alle kinderen van "node" een gelijksoortige evaluatie-functie hebben. We breiden ons add-object uit tot een kind van dit basisobject: zie figuur 2.

Door het gereserveerde woord "OBJECT" te laten volgen door de naam van het basis-object geven we aan dat "add" een directe afstammeling is van node. De herdefinitie van de method "eval" geeft aan dat "add" zijn eigen interpretatie heeft van deze functie. In dit geval dus de som van een evaluatie van een linkerboom en een evaluatie van de rechterboom. Helemaal volledig zijn deze objectdefinities nog niet. De extra functies in het voorbeeld programma zijn sterk taal-afhankelijk en een gedetailleerde omschrijving van hun werking kunt u eenvoudig vinden in de OOP-guide van Borland.

Onze parsertree krijgt nu een iets ander gezicht. Voor ons voorbeeld hebben we drie objecten nodig: een value-object, een add-object en een mul-object. De definitie van het value-object is afwijkend van de overige objecten in die zin dat er geen sub-tree's aan het geheel vastzitten. Het resultaat van zijn eval-method is niet afhankelijk van een tweetal sub-vertakkingen, maar van een opgeslagen waarde. Deze waarde wordt dan ook in het object opgenomen:

```
TYPE value = OBJECT(node)
           fresult    : integer;
           FUNCTION eval : integer;
     END; { OBJECT value }

FUNCTION value.eval : integer;
     BEGIN
          value.eval := fresult
     END; { FUNCTION value.eval }
```

Nieuwe operators zijn nu eenvoudig toe te voegen. De scanner moet uitgebreid worden zodat hij ook de

nieuwe tokens herkent en er moet een object met bijbehorende routines bijgeschreven worden. Het is nu eenvoudig in te zien dat we het hele postfix gebeuren met een ingewikkelde extra rekenstack overbodig gemaakt hebben. Voor het bouwen van een compiler waarbij werkende code gegenereerd moet worden gaat ons verhaal helaas maar heel ten dele op. De OOP-parsertree is nog best bruikbaar, alleen kunnen de functies van zo'n tree nu geen directe resultaten leveren maar zullen code moeten gaan genereren. En die resulterende code kan het eenvoudigst de "ouderwetse" postfix-manier gebruiken.

Object oriënted programming is dus niet zo maar een loze kreet. We kunnen er met wat soepeler denkwerk hele leuke dingen mee doen! Nu weet ik dat de "echte programmeur" nogal conservatief ingesteld is en weinig of niet geneigd zal zijn nieuwe programmeer-methoden aan te leren. OOP heeft de originele interessante programmeertechniek die voor het parser-probleem nodig was gedeeltelijk overbodig gemaakt. In de praktijk betekent zoiets dat je een stukje parate kennis wel overboord kunt gooien... Toch zijn dit soort innovaties nodig om (steeds ingewikkelder wordende!) computers efficiënter te kunnen programmeren. Geen overbodige nieuwerwetsheid dus!

Joost Voorhaar

```
PROGRAM parser;

{ Copyright (c) 1992, J.Voorhaar - Alle rechten voorbehouden          }
{ Gelicenseerd voor publicatie aan de KIM Gebruikersclub Nederland    }
{                                                                     }
{ Dit programma demonstreert OOP aan de hand van een eenvoudige parser.  }
{ Compiler: TurboPascal 5.5                                           }
{                                                                     }
{ De volgende prioriteiten en operators worden gehanteerd:            }
{ Prio.    Operator                                                   }
{ $80      * / %      (mul, div, mod)                                 }
{ $70      + -        (add, sub)                                      }
{                                                                     }
{ De unaire - (negatieve getallen!) wordt niet ondersteund. Wil men toch  }
{ negatieve getallen invoeren, dan kan dat door de expressie 0 - getal te  }
{ gebruiken. Implementatie van deze unaire min, haakjes en andere operators }
{ wordt aan de lezer overgelaten "ter leering ende vermaeck".         }

USES crt;

TYPE restype = longint;
  nodeptr = ^node;
  node = OBJECT                          { De oer-ouder }
   left,right : nodeptr;            { Linker- en rechter subtree }
   FUNCTION eval : restype; VIRTUAL; { Retourneert subtree resultaat }
   PROCEDURE print; VIRTUAL;          { Geeft waarde of type object }
   CONSTRUCTOR init;                { Nodig voor initialisatie }
  END; { OBJECT node }

  valobj = OBJECT(node)             { Definitie van een constante }
   result : restype;
   FUNCTION eval : restype; VIRTUAL;
   PROCEDURE print; VIRTUAL;
   CONSTRUCTOR init(strg : string);
  END; { OBJECT valobj }

  addobj = OBJECT(node)             { Definitie van een optelling }
```

```
    FUNCTION eval : restype; VIRTUAL;
    PROCEDURE print; VIRTUAL;
    CONSTRUCTOR init;
   END; { OBJECT addobj }

   subobj = OBJECT(node)              { Definitie van een aftrekking }
    FUNCTION eval : restype; VIRTUAL;
    PROCEDURE print; VIRTUAL;
    CONSTRUCTOR init;
   END; { OBJECT subobj }

   mulobj = OBJECT(node)          { Definitie van een vermenigvuldiging }
    FUNCTION eval : restype; VIRTUAL;
    PROCEDURE print; VIRTUAL;
    CONSTRUCTOR init;
   END; { OBJECT mulobj }

   divobj = OBJECT(node)                { Definitie van een deling }
    FUNCTION eval : restype; VIRTUAL;
    PROCEDURE print; VIRTUAL;
    CONSTRUCTOR init;
   END; { OBJECT divobj }

   modobj = OBJECT(node)              { Definitie van een rest deling }
    FUNCTION eval : restype; VIRTUAL;
    PROCEDURE print; VIRTUAL;
    CONSTRUCTOR init;
   END; { OBJECT mod }

VAR errorcnt : word;      { Aantal errors }

PROCEDURE error(msg : string);   { Geeft een foutmelding }
 BEGIN
  writeln('Error: ',msg);
  inc(errorcnt)
 END; { PROCEDURE error }

{ De volgende constructors zijn slechts gedefinieerd omdat TP ze nodig heeft }
{ bij het linken van de virtual methode table.                               }

CONSTRUCTOR node.init; BEGIN END;
CONSTRUCTOR addobj.init; BEGIN END;
CONSTRUCTOR subobj.init; BEGIN END;
CONSTRUCTOR mulobj.init; BEGIN END;
CONSTRUCTOR divobj.init; BEGIN END;
CONSTRUCTOR modobj.init; BEGIN END;

{ Een uitzondering op de bovenstaande constructors is de constructor van   }
{ valobj; die heeft namelijk een parameter nodig!                          }

CONSTRUCTOR valobj.init(strg : string);
 VAR errpos : integer;
 BEGIN
  WHILE (strg '') AND (strg[1] IN [' ',#9]) DO
   delete(strg,1,1);    { Delete preceeding whitespaces }
  errpos := length(strg);
```

```
  WHILE (errpos  0) AND (strg[errpos] IN [' ',#9]) DO
    dec(errpos);
  strg := copy(strg,1,errpos);  { And delete terminating whitespaces }
  val(strg,result,errpos);
  IF errpos  0 THEN
    error('illegal character in value "' + strg + '"');
 END; { CONSTRUCTOR valopbj.init }

{ De evaluatie functie voor het parent object. is slechts noodzakelijk om }
{ TurboPascal tevreden te stellen...                                       }

FUNCTION node.eval : restype; BEGIN END;
FUNCTION valobj.eval : restype; BEGIN eval := result END;
FUNCTION addobj.eval : restype; BEGIN eval := left^.eval + right^.eval; END;
FUNCTION subobj.eval : restype; BEGIN eval := left^.eval - right^.eval; END;
FUNCTION mulobj.eval : restype; BEGIN eval := left^.eval * right^.eval; END;
FUNCTION divobj.eval : restype; BEGIN eval := left^.eval DIV right^.eval; END;
FUNCTION modobj.eval : restype; BEGIN eval := left^.eval MOD right^.eval; END;

{ De print-procedure van node is wederom slechts noodzakelijk om TP niet }
{ teleur te stellen...                                                    }

PROCEDURE node.print; BEGIN END;
PROCEDURE valobj.print; BEGIN write(result,' '); END;
PROCEDURE addobj.print; BEGIN write('+ '); END;
PROCEDURE subobj.print; BEGIN write('- '); END;
PROCEDURE mulobj.print; BEGIN write('* '); END;
PROCEDURE divobj.print; BEGIN write('/ '); END;
PROCEDURE modobj.print; BEGIN write('% '); END;

{ Hier volgt de eigenlijke scanner voor het opzetten van de parsertree    }
{ De scanner zoekt steeds een breekpunt in de string; dit is altijd de    }
{ operator met de laagste prioriteit. Vervolgens roept de scanner zichzelf }
{ aan om zijn linker- en rechter subtree te bepalen. Als de parameter uit  }
{ een reeks digits bestaat hebben we een blad bereikt. De string bevat dan }
{ de waarde van de constante. Als er ergens een fout gedetecteerd wordt   }
{ geeft de scanner de fout weer en increment de error-counter. De tree    }
{ retourneert dan een NIL.                                                }


FUNCTION scan(expstr : string) : nodeptr;
 VAR index  : integer;
    break  : byte;
    prio   : byte;     { Minimale prioriteit }
    help   : nodeptr;
    valhelp : ^valobj ABSOLUTE help; { Deze ABSOLUTE statements maken het  }
    addhelp : ^addobj ABSOLUTE help; { mogelijk om op eenvoudige wijze het }
    subhelp : ^subobj ABSOLUTE help; { object te kunnen initialiseren.     }
    mulhelp : ^mulobj ABSOLUTE help; { Door slechts 1 object te alloceren  }
    divhelp : ^divobj ABSOLUTE help; { met "new" worden al deze variabelen }
    modhelp : ^modobj ABSOLUTE help; { tegelijk gevuld!                    }
 BEGIN
  help := NIL;
  break := 0;
  prio := $FF;        { Begin met maximale prioriteit }
  FOR index := 1 TO length(expstr) DO
```

```
  IF errorcnt = 0 THEN
   CASE expstr[index] OF
       ' ',#9 : ; { Spaties overslaan a.u.b.! }
     '0'..'9' : ; { Voorlopig ook geen digits... }
     '+','-' : IF prio  $70 THEN  { Nieuw breekpunt! }
               BEGIN
                break := index;
                prio := $70
               END;
   '*','/','%' : IF prio  $80 THEN  { Nieuw breekpunt gevonden! }
               BEGIN
                break := index;
                prio := $80
               END;
         ELSE error('unknown symbol "' + expstr[index] + '"');
   END; { CASE }
  IF errorcnt = 0 THEN
   IF break = 0 THEN
    BEGIN
     new(valhelp,init(expstr));
     valhelp^.left := NIL;
     valhelp^.right := NIL;
    END { IF }
   ELSE
    BEGIN
     CASE expstr[break] OF
      '*' : new(mulhelp,init);
      '/' : new(divhelp,init);
      '%' : new(modhelp,init);
      '+' : new(addhelp,init);
      '-' : new(subhelp,init);
     END; { CASE }
     help^.left := scan(copy(expstr,1,break - 1));
     help^.right := scan(copy(expstr,break + 1,255));
     IF (help^.left = NIL) OR (help^.right = NIL) THEN
      help := NIL;
    END; { ELSE }
  scan := help;
 END; { FUNCTION scan }

{ Printpost print de boom uit in postfix notatie }

PROCEDURE printpost(nptr : nodeptr);
 BEGIN
  IF nptr^.left  NIL THEN
   printpost(nptr^.left);
  IF nptr^.right  NIL THEN
   printpost(nptr^.right);
  nptr^.print;
 END; { PROCEDURE printtree }

VAR strg   : string;
  tree   : nodeptr;
  garbage : pointer;
BEGIN
 textattr := lightgray;       { Grijze letters op een zwarte achtergrond }
```

```
clrscr;                          { Begin met een leeg scherm }
writeln('Parser OOP demo - Copyright (c) 1992, J.Voorhaar');
writeln('Licensed to the KIM Usergroup Holland (KGN) for publication purposes only');
writeln;
writeln('Enter integer expressions on the prompt, or an empty line to quit.');
window(1,6,80,25);
write(': ');
readln(strg);
WHILE strg '' DO
 BEGIN
  mark(garbage);
  errorcnt := 0;
  tree := scan(strg);
  IF (tree  NIL) AND (errorcnt = 0) THEN
   BEGIN
    write('Postfix: ');
    printpost(tree);
    writeln;
    writeln('Result : ',tree^.eval);
   END; { IF }
  release(garbage);        { De tree mag nu in z'n geheel weg! }
  writeln;
  write(': ');
  readln(strg);
 END; { WHILE }
writeln;
writeln('Goodbye!');
window(1,1,80,25);
gotoxy(1,24);
END.
```

Voortgang KGN-68k (deel 6)

De tijd tussen dit nummer van de μP Kenner en de vorige is kort geweest. En in een korte tijd kun je niet zoveel vooruitgang boeken. Uit de winterstop is de werkgroep nog niet helemaal, de vaart zit er niet meer zo goed in als in het begin.

Printed Circuit Board

Wat er nu ondertussen wel vast staan zijn de maten van de printplaat, het bord zal 12 bij 13,2 inch groot worden. Niet de insteekkaart waar we in het begin van uit wilden gaan, maar zoiets heet ontwikkeling. Je het kunt ook overschrijding van specificaties noemen, maar de werkgroep vindt dit toch nog steeds de meeste ideale oplossing.

Software

De Software zaken zijn nog steeds niet lekker aan het rollen, er zit wel beweging in maar het gaat met horten en stoten. Bedenk wel dat je een programma achteraf makkelijker hebt aangepast dan een printed circuit board.

Geld & Auto's

Prijstechnisch gezien zal de strijd met de concurrentie steeds moeilijker worden. De i80386DX machines worden namelijk steeds goedkoper en daar kun je ook MINIX 32-bits breed opdraaien. De prijs is gelukkig niet altijd de beslissende factor, anders zouden er toch veel meer Oost-Europese auto's in Nederland verkocht worden. Automobielen zijn eigenlijk bedoeld om te rijden en dat doen de meeste mensen dan ook. Maar voor auto-monteurs is het meer dan een transportmiddel. Kenners kijken nou eenmaal toch anders tegen (gebruiks)voorwerpen aan.

Tendens

In een oogopslag kunt je zien dat het verslag van de voortgang weer korter is als de vorige keer. Pessimisten & doemdenkers zullen zeggen: "Zie je wel, een aflopende zaak." Nee, er is nog steeds sprake van voortgang. U/Je weet het, reacties zijn altijd welkom.

Geert Stappers

Voortgang KGN-68k

Het is niet overtuigend om te vertellen dat het KGN-68k project nog bestaat en dat er dan niets van in De μP Kenner staat. Als het goed gaat staan er dan in dit nummer dan ook twee afleveringen in. Beter laat dan nooit, maar beter nooit te laat.

Printed Circuit Board

Tijdens het ontwerp van de printplaat zijn we al een keer tegen de grens aangelopen van het printontwerppakket. Nu is het weer gebeurd, we zijn echt met een gigantische brok techniek bezig. De lay-outer kwam met de opmerking dat het pakket veel minder geheugen rapporteert dan er in de machine zit. Navraag bij Ultimate technology leverde de bevestiging op dat we boven de limiet van Ultiboard Advanced Level zaten. Met de huidige versie zullen we het board niet af kunnen maken. Op dit moment werken we met een tijdelijke oplossing. De connectors waar alle MC68030 signalen op zitten, zijn allemaal op één na vervangen door een component met maar een pin om zo onder de 2800 pinnen grens van het pakket te komen. Zonder dat dit belangrijk stuk werk stil valt, wordt er onder tussen gezocht naar een oplossing om op het einde die laatste konnectors aan te sluiten en de plotfiles te maken. Als bij U op de zaak deze handelingen verricht kunnen worden, iets wat niet langer dan een halve dag duurt, zouden wij dit graag vernemen.

Software

In de software hoek worden ondertussen ook bergen verzet. Er wordt ijverig gebouwd aan een 680X0 assembler. Op het eerste gezicht iet wat overbodig werk, want achtenzestigduizend (cross-)assemblers zijn er toch al lang. De truuk zit hem in het feit dat hij ook geschikt is voor de MC68020 & MC68030. Hierbij kwam de documentatie die we van EBV Elektronik gekregen goed van pas. Verder nog het voordeel dat we hem in (C-)source hebben. Dit betekent dat hij naar believen (binnen redelijke grenzen) aangepast kan worden en dat we hem straks mee kunnen nemen naar MINIX.

Afsluiting

Dit is eigenlijk standaard werk: Laat de op- en aanmerkingen maar komen!

Geert Stappers

Nieuwtjes en andere wetenswaardigheden, soms met een knipoog

Het is mijn bedoeling om een vaste column toe te voegen aan ons lijfblad. De inhoud hiervan is min of meer in de kop al omschreven. Waar haal je die nieuwtjes weg kun je je afvragen. Dat lijkt moeilijker dan het is. Ik zal in eerste instantie materiaal uit andere computerbladen halen, maar ook de dagelijkse krant geeft soms interessante info. Verder zal ik oren en ogen open houden. Je weet maar nooit, soms ligt de beste informatie gewoon op straat of op je deurmat.

Denk je dat je zelf iets leuks of interessants hebt laat het me dan even weten, mijn telefoonnummer en adres kun je op de laatste pagina van ons lijfblad vinden. Mensen met een modem kunnen natuurlijk ook gebruik maken van ons bulletinboard. Schroom ook niet om kritiek te uiten. Het is uiteindelijk jullie blad. De statuten staan ons echter niet toe om het principe van 'niet goed geld terug' toe te passen. Dus als het niet goed gaat, laat dan wat van je horen!

CHECKIT-LAN Versie 2.0

ToachStone heeft een nieuwe versie van het netwerk diagnoseprogramma uitgebracht. CheckIt-LAN 2.0 is een programma waarmee de netwerkbeheerder een overzicht krijgt van de aangesloten hardware. Tevens is hij in staat om te kijken welke software er op de verschillende stations in gebruik zijn. Performance en viruscontrole is ook aan deze versie toegevoegd. T.o.v. de vorige versie zijn er verbeteringen aangebracht m.b.t. het aantal voorgedefinieerde programma's waarop getest kan worden, automatische alarmering en je kan belangrijke boodschappen doorschakelen naar een willekeurige terminal. Het prijskaartje hangt tussen de $ 249,= = voor 5 gebruikers en $ 1955,= = voor 250 gebruikers.

Microsoft Informatie

Microsoft heeft in Nederland een BBS. Ze hebben daar de originele naam MICROSOFT BBS voor bedacht. Het is bereikbaar onder telefoonnummer: 02503-34221. De vereiste instellingen zijn 1200 of 2400 baud, 8 bits, 1 stopbitje en non-parity. Er is een database met meer dan 30.000 informatieblokken over MicroSoft-produkten (ik heb ze niet geteld, als het niet waar is lieg ik in commissie).

Travelpilot

Handig voor verdwaalde bestuursleden is de Travelpilot. Het Amerikaanse blad 'Popular Sience' heeft de Travelpilot uitgekozen als één van de honderd beste nieuwe produkten van 1991. Met behulp van een boordcomputer, CD-ROM met een compleet wegennet, beeldscherm, elektronisch kompas en wielsensoren word(en)t de bestuurder(s) de weg gewezen. Met een dergelijk pilootje in de auto heeft uw bestuur nooit meer het excuus om te laat te komen omdat ze verdwaald zijn.

De PTT doet aan klantenbinding, maar laat particulieren betalen

Wij mogen ons verheugen dat wij de laagste binnenlandse telefoontarieven hebben binnen Europa. De PTT verwacht echter zware concurrentie uit het buitenland. Omdat op te vangen denkt de PTT erover om telefoonverkeer met het buitenland goedkoper te maken. Ze sluiten niet uit dat dan de binnenlandse tarieven duurder zullen worden. Omdat bedrijven meer gebruik maken van internationaal telefoon verkeer dan particulieren kun je op je vingers natellen dat de laatste groep dadelijk voor de kosten op mag draaien.

De hereniging

In μP Kenner 75 heeft u een programma kunnen aantreffen, dat van een file twee nieuwe maakte, een met de even, en een met de oneven bytes erin. Ook werd daarbij beloofd, dat het omgekeerde er ook was, namelijk twee files met elk even en oneven bytes weer samenvoegen tot de oorspronkelijke file. Wel, hier is 'ie dan. SHUFFLE.EXE heet het.

Dit programma is absoluut niet wereldschokkend als u reeds SPLIT.ASM heeft bestudeerd: de opbouw en stijl van programmeren zijn vrijwel identiek. Ook hier wordt gebruik gemaakt van handles, terwijl een aantal subroutines zelfs precies hetzelfde zijn. Dat ligt ook wel voor de hand: want in plaats van lezen wordt er geschreven in files en omgekeerd ten opzichte van SPLIT.ASM.

Misschien moet ik een paar kleine excuses aanbieden wegens het plegen van moedwillige bladvulling, maar liever compleet dan half...

Nico de Vries

```
NAME    SHUFFLE
PAGE    66,131
TITLE   SHUFFLE ---- Merge odd and even input files into one 16-bit file

;****************************************
;*
;* SHUFFLE Version 1.0
;*
;* Merges two input files into one
;* output file. The first input file
;* must have extension .LO and will
;* supply all even bytes in the output
;* file, the second input file must
;* have extension .HI and will supply
;* all odd bytes.
;* In case the input files have dif-
;* ferent lengths, the output file will
;* have a length of twice the length
;* of the longest input file; the
;* missing bytes wil be filled with
;* a zero value.
;* The input files inherit their path
;* and file name from the output file,
;* but their extensions will be .LO and
;* .HI by default.
;*
;* Note that the file naming system is
;* compatible with that of SPLIT.EXE.
;*
;* Syntax:
;* SHUFFLE [d:][path]filename[.ext]
;*
;* To create SHUFFLE.EXE:
;* MASM SHUFFLE;
;* LINK SHUFFLE,TEMP;
;* EXEPACK TEMP.EXE SHUFFLE.EXE
;*
;****************************************
```

Fig. 1: sourcetekst van SHUFFLE.ASM

```
CR              EQU     0DH             ;ASCII carriage return
LF              EQU     0AH             ;ASCII line feed
TAB             EQU     09H             ;ASCII TAB code
BLANK           EQU     20H             ;ASCII space code

COMMAND         EQU     80H             ;buffer for command tail

B_SIZ           EQU     16384           ;size of records

OUTPUT_HANDLE   EQU     1               ;handle of standard output device
ERROR_HANDLE    EQU     2               ;handle of standard error device

CSEG            SEGMENT PARA PUBLIC 'CODE'

                ASSUME  CS:CSEG,DS:DATA,ES:DATA,SS:STACK

                ;****************************************
                ;*
                ;* Start of program. Test for correct
                ;* DOS version, because SHUFFLE uses
                ;* 'handle calls' not available in
                ;* DOS 1.X.
                ;*
                ;****************************************
SHUFFLE         PROC    FAR
                PUSH    DS              ;save DS:0000 for final
                XOR     AX,AX           ;return to DOS in case
                PUSH    AX              ;function 04CH cannot be used
                MOV     AX,Seg DATA     ;make our own data segment
                MOV     ES,AX           ;accesible via ES register
                MOV     AH,030H         ;check version
                INT     21H             ;of DOS
                CMP     AL,2            ;if version 1.X
                JAE     SHUFFLE1
                MOV     DX,Offset DOS_ERR ;point to DOS 1.X error message
                PUSH    ES              ;point DS
                POP     DS              ;to data segment
                MOV     AH,9            ;cannot use modern function here
                INT     21H             ;print message
                RET                     ;and exit to caller
                ;****************************************
                ;*
                ;* Read output file name from input
                ;* buffer.
                ;*
                ;****************************************
SHUFFLE1:       CALL    GET_FILENAME    ;get path and file name spec
                PUSH    ES              ;point DS
                POP     DS              ;to own data segment
                JNC     SHUFFLE2        ;if no error, proceed
                MOV     DX,Offset SIGN_ON ;point to description message
                MOV     CX,SIGN_ON_LENGTH ;get length
                JMP     SHUFFLE16       ;and print message
                ;****************************************
                ;*
                ;* Create the input file names.
```

```
                ;*
                ;***************************************
SHUFFLE2:       CLD                             ;set direction forward
                MOV     DI,Offset INL_NAME      ;point DI to output name low
                CALL    MAKE_INNAME             ;create input file name low
                JC      SHUFFLE3                ;if no usable file name, exit with error
                MOV     AX,'OL'                 ;else get extention
                STOSW                           ;move too
                XOR     AL,AL                   ;get a zero
                STOSB                           ;and terminate inputname low
                MOV     DI,Offset INH_NAME      ;and DI to output name high
                CALL    MAKE_INNAME             ;create input file name hi
                MOV     AX,'IH'                 ;get extention
                STOSW                           ;move too
                XOR     AL,AL                   ;get a zero
                STOSB                           ;and terminate inputname high
                ;***************************************
                ;*
                ;* Try to open the .LO input file.
                ;*
                ;***************************************
                MOV     DX,Offset INL_NAME      ;point DX to filename, even
                CALL    OPEN_INPUT              ;try to open input file, even
                JNC     SHUFFLE4                ;if OK, proceed
SHUFFLE3:       MOV     DX,Offset FILE_NT_FND       ;point to message
                MOV     CX,FILE_NT_FND_L        ;get length
                JMP     SHUFFLE16               ;and print message
                ;***************************************
                ;*
                ;* Try to open the .HI input file.
                ;*
                ;***************************************
SHUFFLE4:       MOV     INL_HANDLE,AX           ;store handle number
                MOV     DX,Offset INH_NAME      ;point DX to filename, odd
                CALL    OPEN_INPUT              ;try to open input file, odd
                JC      SHUFFLE3                ;if error, print it
                MOV     INH_HANDLE,AX           ;store handle number
                ;***************************************
                ;*
                ;* Get length of input files. Print a
                ;* warning if the files hold different
                ;* numbers of bytes.
                ;*
                ;***************************************
                MOV     BX,INL_HANDLE           ;get handle
                CALL    GET_FILE_LENGTH         ;get input file length, even file
                JC      SHUFFLE5                ;if error exit with message
                MOV     INL_SIZEH,DX            ;else save MSW
                MOV     INL_SIZEL,AX            ;and LSW
                MOV     IN_SIZEH,DX             ;and in longest length
                MOV     IN_SIZEL,AX             ;do high too
                MOV     BX,INH_HANDLE           ;get handle
                CALL    GET_FILE_LENGTH         ;get input file length, odd file
                JC      SHUFFLE5                ;if error exit with message
                MOV     INH_SIZEH,DX            ;else save MSW high
                MOV     INH_SIZEL,AX            ;and LSW high
                CMP     INL_SIZEH,DX            ;if high lengths differ
```

```
                JNE     LENGTH_DIFF     ;get highest as count
                CMP     INL_SIZEL,AX    ;if low lengths same
                JE      SHUFFLE6        ;proceed
LENGTH_DIFF:    JA      PR_DIFF_LENGTH  ;if longest stored now, print message
                MOV     IN_SIZEH,DX     ;else store new longest length
                MOV     IN_SIZEL,AX     ;do high too
PR_DIFF_LENGTH: MOV     DX,Offset DIFF   ;point to 'File lengths differ
                MOV     CX,DIFF_LENGTH  ;get length
                CALL    WRITE_STD       ;print message
                JMP     Short SHUFFLE6  ;and proceed
SHUFFLE5:       MOV     DX,Offset MSG4  ;point to 'Input file(s) is/are empty'
                MOV     CX,MSG4_LENGTH  ;get length
                JMP     SHUFFLE15       ;and print message
                ;****************************************
                ;*
                ;* Create the output file.
                ;*
                ;****************************************
SHUFFLE6:       MOV     DX,Offset OUT_NAME  ;get file name address
                MOV     AL,1            ;access is write
                MOV     CX,0            ;normal attribute
                MOV     AH,03CH         ;create file
                INT     21H             ;via DOS
                JNC     SHUFFLE7        ;if no error, proceed
SHUFFLE6A:      JMP     SHUFFLE14       ;and print message
SHUFFLE7:       MOV     OUT_HANDLE,AX   ;else save handle
                ;****************************************
                ;*
                ;* Read 1st block of input file .LO
                ;* into      even input buffer. Check if DOS
                ;* returns an error. If so, report
                ;* empty input file.
                ;*
                ;****************************************
                CALL    READ_BLOCK_L    ;read 1st block of data
                JNC     SHUFFLE7B       ;if OK, proceed
SHUFFLE7A:      MOV     DX,Offset MSG4  ;point to 'Input file is empty'
                MOV     CX,MSG4_LENGTH  ;get length
                JMP     SHUFFLE14       ;and print message
                ;****************************************
                ;*
                ;* Read 1st block of input file .HI
                ;* into      odd input buffer. Check if DOS
                ;* returns an error. If so, report
                ;* empty input file.
                ;*
                ;****************************************
SHUFFLE7B:      CALL    READ_BLOCK_H    ;read 1st block of data
                JC      SHUFFLE7A       ;if error, report 'Input file is empty'
                ;****************************************
                ;*
                ;* Perform the actual shuffle.
                ;*
                ;****************************************
SHUFFLE8:       MOV     Word Ptr OUT_PTR,0  ;reset output byte counter
SHUFFLE9:       MOV     AX,INL_SIZEH    ;get byte count even high
                OR      AX,INL_SIZEL    ;if through, pad a zero
```

```
                JE      FILL_LO         ;and put in buffer
                CALL    GET_CHAR_L      ;else read a byte from even input
                JC      PAD_LO          ;if error, pad buffer
                DEC     INL_SIZEL       ;else count input bytes
                JNE     FILL_LO         ;if even low zero,
                MOV     DI,INL_SIZEH    ;get even hi
                OR      DI,DI           ;if zero too
                JE      FILL_LO         ;write last even byte
                DEC     INL_SIZEH       ;else adapt count hi
                JMP     Short FILL_LO   ;and put even byte in buffer
PAD_LO:         MOV     AL,0            ;pad entry, get a zero
FILL_LO:        MOV     DI,Offset OUT_BUF ;get buffer address
                MOV     BX,OUT_PTR      ;get buffer offset
                MOV     [DI+BX],AL      ;put byte in buffer
                INC     OUT_PTR         ;adapt output byte counter
SHUFFLE10:      MOV     AX,INH_SIZEH    ;get byte count high
                OR      AX,INH_SIZEL    ;if through, pad a zero
                JE      PAD_HI          ;and put in buffer
                CALL    GET_CHAR_H      ;else read 1 byte from odd input
                JC      PAD_HI          ;if error, pad buffer
                DEC     INH_SIZEL       ;else count input bytes
                JNE     FILL_HI         ;if odd low zero,
                MOV     DI,INH_SIZEH    ;get odd hi
                OR      DI,DI           ;if zero too
                JE      FILL_HI         ;write last odd byte
                DEC     INH_SIZEH       ;else adapt count hi
                JMP     Short FILL_HI   ;and put odd byte in buffer
PAD_HI:         MOV     AL,0            ;pad entry, get a zero
FILL_HI:        MOV     DI,Offset OUT_BUF ;get buffer address
                MOV     BX,OUT_PTR      ;get buffer offset
                MOV     [DI+BX],AL      ;put byte in buffer
                INC     OUT_PTR         ;adapt output byte counter
                INC     BX              ;and BX
                DEC     IN_SIZEL        ;count input bytes low (longest)
                JNE     SHUFFLE11       ;if longest low zero,
                MOV     DI,INH_SIZEH    ;get longest hi
                OR      DI,DI           ;if zero too
                JE      SHUFFLE12       ;write remaining bytes
                DEC     INH_SIZEH       ;else adapt count hi
SHUFFLE11:      CMP     BX,B_SIZ        ;if buffer not full yet
                JNE     SHUFFLE9        ;continue to fill
                CALL    WRITE_BLOCK     ;else write buffer to file
                JMP     SHUFFLE8        ;and loop
                ;***************************************
                ;*
                ;* Longest input file complety read.
                ;* Flush data remaining in output
                ;* buffer in output file.
                ;*
                ;***************************************
SHUFFLE12:      CALL    WRITE_BLOCK     ;else write output file
                MOV     BX,INL_HANDLE   ;get file handle
                CALL    CLOSE_INPUT     ;close input file, even
                MOV     BX,INH_HANDLE   ;get file handle
                CALL    CLOSE_INPUT     ;close input file, odd
                CALL    CLOSE_OUTPUT    ;close output file
                MOV     DX,Offset MSG5  ;point to 'SHUFFLE succesful.'
```

```
                MOV     CX,MSG5_LENGTH      ;get length
                CALL    WRITE_STD           ;print message
                MOV     AX,04C00H           ;command is exit DOS
                INT     21H                 ;with error level 0
                ;************************************
                ;*
                ;* Error exit: problem with output
                ;* file. Report problem through error
                ;* output handle and close all files.
                ;*
                ;************************************
SHUFFLE14:      CALL    CLOSE_OUTPUT        ;close output file
                MOV     DX,Offset WR_ERR    ;point to 'Error writing output file'
                MOV     CX,WR_ERR_LENGTH    ;get length
                ;************************************
                ;*
                ;* Error exit: problem with input file.
                ;* Report problem through error output
                ;* handle and close the file.
                ;*
                ;************************************
SHUFFLE15:      PUSH    BX
                PUSH    CX
                PUSH    DX
                MOV     BX,INL_HANDLE       ;get file handle
                CALL    CLOSE_INPUT         ;close input file, even
                MOV     BX,INH_HANDLE       ;get file handle
                CALL    CLOSE_INPUT         ;close input file, odd
                POP     DX
                POP     CX
                POP     BX
                ;************************************
                ;*
                ;* Error exit: problem with input file
                ;* name. Report problem through error
                ;* output handle.
                ;*
                ;************************************
SHUFFLE16:      CALL    WRITE_ERROR         ;write error message
                MOV     AX,04C01H           ;exit to DOS
                INT     21H                 ;with error level 1

SHUFFLE         ENDP

                ;************************************
                ;*
                ;* Move output file description to
                ;* internal buffers, stripping it from
                ;* spaces and TAB characters.
                ;* If no filename is found, an error is
                ;* flagged.
                ;*
                ;************************************
GET_FILENAME    PROC    NEAR
                MOV     SI,Offset COMMAND   ;point SI to command line
                MOV     DI,Offset OUT_NAME  ;point DI to buffer for name
                CLD                         ;set direction forward
```

```
                LODSB                   ;get length byte
                OR      AL,AL           ;if no name
                JE      GET_FILENAME5   ;exit with error
GET_FILENAME1:  LODSB                   ;else get command line byte
                CMP     AL,CR           ;if carriage return
                JE      GET_FILENAME5   ;exit with error
                CMP     AL,BLANK        ;if space
                JE      GET_FILENAME1   ;get next character
                CMP     AL,TAB          ;if TAB
                JE      GET_FILENAME1   ;get next character
GET_FILENAME2:  STOSB                   ;else store file name character
GET_FILENAME3:  LODSB                   ;get next
                CMP     AL,CR           ;if last
                JE      GET_FILENAME4   ;exit without error
                CMP     AL,BLANK        ;if space
                JE      GET_FILENAME3   ;skip it
                CMP     AL,TAB          ;if not TAB
                JNE     GET_FILENAME2   ;store and get next character
GET_FILENAME4:  CLC                     ;flag no error
                RET                     ;and exit
GET_FILENAME5:  STC                     ;flag error
                RET                     ;and exit
GET_FILENAME    ENDP

                ;****************************************
                ;*
                ;* Open input file. If an error occurs,
                ;* carry will be set. DX points to file
                ;* name to open. AX holds handle if
                ;* the file open was succesful.
                ;*
                ;****************************************
OPEN_INPUT      PROC    NEAR
                MOV     AL,0            ;file is read only
                MOV     AH,03DH         ;open handle
                INT     21H             ;call DOS
                RET                     ;and exit
OPEN_INPUT      ENDP

                ;****************************************
                ;*
                ;* Read file length of input file. If
                ;* an error occurs carry will be set.
                ;*
                ;****************************************
GET_FILE_LENGTH PROC    NEAR
                PUSH    BX              ;save it on stack
                MOV     AL,2            ;get end file position
                MOV     AH,042H         ;move file pointer
                XOR     DX,DX           ;get zero
                XOR     CX,CX           ;offset
                INT     21H             ;get file length
                POP     BX              ;get handle back
                JC      LENGTH_ERROR    ;if error exit
                PUSH    DX              ;else save MSW
                PUSH    AX              ;and LSW
                MOV     AL,0            ;get begin file position
```

```
                        MOV     AH,042H         ;move file pointer
                        XOR     DX,DX           ;get zero
                        XOR     CX,CX           ;offset
                        INT     21H
                        POP     AX              ;get length low in AX
                        POP     DX              ;and length high in DX
LENGTH_ERROR:           RET                     ;and exit
GET_FILE_LENGTH         ENDP

                        ;***************************************
                        ;*
                        ;* Create output file name. The input
                        ;* file name string is copied to a
                        ;* buffer upto the end, or the period
                        ;* at the start of extension. The new
                        ;* name string is terminated with a
                        ;* period.
                        ;*
                        ;***************************************
MAKE_INNAME             PROC    NEAR
                        CLD                     ;set direction forward
                        MOV     SI,Offset OUT_NAME ;point SI to output name buffer
                        LODSB                   ;get character
                        CMP     AL,'.'          ;if 1st is not a period
                        JNE     MAKE_NAME3      ;parse input string
                        STOSB                   ;else store character
MAKE_NAME1:             LODSB                   ;get next
                        STOSB                   ;move to buffer
                        OR      AL,AL           ;if at end now
                        JE      MAKE_NAME5      ;exit
                        CMP     AL,'\'          ;if not backslash
                        JNE     MAKE_NAME1      ;loop
MAKE_NAME2:             LODSB                   ;get next character
                        CMP     AL,'.'          ;if at period of extension
                        JE      MAKE_NAME4      ;stop now
MAKE_NAME3:             STOSB                   ;else move character
                        OR      AL,AL           ;if not at end now
                        JNE     MAKE_NAME2      ;loop
                        MOV     AL,'.'          ;else get period
MAKE_NAME4:             STOSB                   ;in output name
                        CLC                     ;flag no error
                        RET                     ;and exit
MAKE_NAME5:             STC                     ;flag error
                        RET                     ;and exit
MAKE_INNAME             ENDP

                        ;***************************************
                        ;*
                        ;* Close input file. If an error occurs
                        ;* carry will be set.
                        ;*
                        ;***************************************
CLOSE_INPUT             PROC    NEAR
                        PUSH    AX
                        PUSH    BX
                        MOV     AH,03EH         ;close handle
                        INT     21H
```

```
                POP     BX
                POP     AX
                RET                     ;and exit
CLOSE_INPUT     ENDP

                ;***************************************
                ;*
                ;* Close output file. If an error
                ;* occurs, carry will be set.
                ;*
                ;***************************************
CLOSE_OUTPUT    PROC    NEAR
                PUSH    AX
                PUSH    BX
                MOV     BX,OUT_HANDLE   ;get file handle
                MOV     AH,03EH         ;close handle
                INT     21H
                POP     BX
                POP     AX
                RET                     ;and exit
CLOSE_OUTPUT    ENDP

                ;***************************************
                ;*
                ;* Get character from the even input
                ;* file buffer. If the buffer is empty,
                ;* it will be filled by reading from
                ;* disk.
                ;* If the input file is completely read
                ;* carry will be set.
                ;*
                ;***************************************
GET_CHAR_L      PROC    NEAR
                PUSH    BX              ;save BX
GET_CHARL1:     MOV     BX,INL_PTR      ;get current buffer address
                CMP     BX,B_SIZ/2      ;if not through
                JNE     GET_CHARL2      ;get character from buffer
                MOV     INL_PTR,0       ;else zero buffer address
                CALL    READ_BLOCK_L    ;read in next block
                JNC     GET_CHARL1      ;if no error, loop
                POP     BX              ;restore BX
                RET                     ;else exit
GET_CHARL2:     MOV     AL,[INL_BUFFER+BX] ;get buffer character
                INC     INL_PTR         ;adapt buffer address
                POP     BX              ;restore BX
                CLC                     ;flag no error
                RET                     ;and exit
GET_CHAR_L      ENDP

                ;***************************************
                ;*
                ;* Get character from the odd input
                ;* file buffer. If the buffer is empty,
                ;* it will be filled by reading from
                ;* disk.
                ;* If the input file is completely read
                ;* carry will be set.
```

```
                ;*
                ;****************************************
GET_CHAR_H      PROC    NEAR
                PUSH    BX                      ;save BX
GET_CHARH1:     MOV     BX,INH_PTR              ;get current buffer address
                CMP     BX,B_SIZ/2              ;if not through
                JNE     GET_CHARH2              ;get character from buffer
                MOV     INH_PTR,0               ;else zero buffer address
                CALL    READ_BLOCK_H            ;read in next block
                JNC     GET_CHARH1              ;if no error, loop
                POP     BX                      ;restore BX
                RET                             ;else exit
GET_CHARH2:     MOV     AL,[INH_BUFFER + BX]    ;get buffer character
                INC     INH_PTR                 ;adapt buffer address
                POP     BX                      ;restore BX
                CLC                             ;flag no error
                RET                             ;and exit
GET_CHAR_H      ENDP

                ;****************************************
                ;*
                ;* Fill the even input buffer by
                ;* reading from the .LO input file from
                ;* disk. If the file is read up to the
                ;* end, carry will be set.
                ;*
                ;****************************************
READ_BLOCK_L    PROC    NEAR
                MOV     BX,INL_HANDLE           ;get file handle
                MOV     DX,Offset INL_BUFFER    ;get buffer address
                CALL    READ_BLOCK              ;read block of data
                MOV     INL_PTR,0               ;zero buffer pointer
                RET                             ;and exit
READ_BLOCK_L    ENDP

                ;****************************************
                ;*
                ;* Fill the odd input buffer by reading
                ;* from the .HI input file from disk.
                ;* If the file is read up to the end,
                ;* carry will be set.
                ;*
                ;****************************************
READ_BLOCK_H    PROC    NEAR
                MOV     BX,INH_HANDLE           ;get file handle
                MOV     DX,Offset INH_BUFFER    ;get buffer address
                CALL    READ_BLOCK              ;read block of data
                MOV     INH_PTR,0               ;zero buffer pointer
                RET                             ;and exit
READ_BLOCK_H    ENDP

                ;****************************************
                ;*
                ;* Fill an input buffer by reading an
                ;* input file from disk. If the file is
                ;* read up to the end, carry will be
                ;* set.
```

```
                ;*
                ;**************************************
READ_BLOCK      PROC    NEAR
                MOV     CX,B_SIZ/2          ;get byte count
                MOV     AH,03FH             ;read from handle
                INT     21H
                OR      AX,AX               ;if any read in
                JNZ     READ_BLOCK1
                STC                         ;else flag error
READ_BLOCK1:    RET                         ;and exit
READ_BLOCK      ENDP

                ;**************************************
                ;*
                ;* Write current output buffer to the
                ;* output file. If an error occurs
                ;* carry will be set.
                ;*
                ;**************************************
WRITE_BLOCK     PROC    NEAR
                MOV     BX,OUT_HANDLE       ;get file handle
                MOV     CX,OUT_PTR          ;and byte count
                MOV     DX,Offset OUT_BUF
                MOV     AH,040H             ;write to handle
                INT     21H
                RET                         ;and exit
WRITE_BLOCK     ENDP

                ;**************************************
                ;*
                ;* Write to the standard output device.
                ;* This output can be redirected.
                ;*
                ;**************************************
WRITE_STD       PROC    NEAR
                MOV     BX,OUTPUT_HANDLE    ;get handle
                MOV     AH,40H              ;write to handle
                INT     21H                 ;
                RET                         ;and exit
WRITE_STD       ENDP

                ;**************************************
                ;*
                ;* Write to the standard error device.
                ;* This output can not be redirected.
                ;*
                ;**************************************
WRITE_ERROR     PROC    NEAR
                MOV     BX,ERROR_HANDLE     ;get handle
                MOV     AH,40H              ;write to handle
                INT     21H                 ;
                RET                         ;and exit
WRITE_ERROR     ENDP

CSEG            ENDS

                ;**************************************
```

```
                        ;*
                        ;* Work and buffer area.
                        ;*
                        ;***************************************
DATA                    SEGMENT  PARA PUBLIC 'DATA'

                        ;***************************************
                        ;*
                        ;* Buffers for file names.
                        ;*
                        ;***************************************
INL_NAME                DB       64 DUP (0)          ;buffer for input name, even bytes
INL_HANDLE              DW       0                   ;buffer for handle number
INL_PTR                 DW       0                   ;pointer into input buffer, even
INL_SIZEH               DW       0                   ;file length, hi word
INL_SIZEL               DW       0                   ;file length, lo word

INH_NAME                DB       64 DUP (0)          ;buffer for input name, odd bytes
INH_HANDLE              DW       0                   ;buffer for handle number
INH_PTR                 DW       0                   ;pointer into input buffer, odd
INH_SIZEH               DW       0                   ;file length, hi word
INH_SIZEL               DW       0                   ;file length, lo word

IN_SIZEH                DW       0                   ;longest input file length, hi word
IN_SIZEL                DW       0                   ;longest input file length, lo word

OUT_NAME                DB       64 DUP (0)          ;buffer for output name
OUT_HANDLE              DW       0                   ;buffer for handle number
OUT_PTR                 DW       0                   ;pointer into output buffer

                        ;***************************************
                        ;*
                        ;* Buffers for data processing.
                        ;*
                        ;***************************************
INL_BUFFER              DB       B_SIZ/2 DUP (?)     ;input buffer for DOS read, even
INH_BUFFER              DB       B_SIZ/2 DUP (?)     ;input buffer for DOS read, odd
OUT_BUF                 DB       B_SIZ DUP (?)       ;output buffer


                        ;***************************************
                        ;*
                        ;* Text messages.
                        ;*
                        ;***************************************
SIGN_ON                 DB       CR,LF,'SHUFFLE V1.0'
                        DB       CR,LF,CR,LF,'by NdV, (C)1990 KIM Gebruikersclub Nederland'
                        DB       CR,LF
                        DB       CR,LF,'Shuffles two input files into one output file.'
                        DB       CR,LF,'One input file (extension .LO) will supply'
                        DB       CR,LF,'all even bytes in the output file, the other'
                        DB       CR,LF,'input file (extension .HI) will supply all'
                        DB       CR,LF,'even bytes.'
                        DB       CR,LF,'The input files inherit their path and name'
                        DB       CR,LF,'from the output file, but will have the'
                        DB       CR,LF,'extensions .LO and .HI by default.',CR,LF
                        DB       CR,LF,'Usage: SHUFFLE [d:][path]outfilename[.ext]',CR,LF
```

```
MSG2                DB      CR,LF,'The output file name is missing.',CR,LF
SIGN_ON_LENGTH      EQU     $-SIGN_ON
MSG2_LENGTH         EQU     $-MSG2

FILE_NT_FND         DB      CR,LF,'Cannot find input file(s).',CR,LF
FILE_NT_FND_L       EQU     $-FILE_NT_FND

DOS_ERR             DB      CR,LF,'Incorrect DOS version, must be DOS 2.0'
                    DB      CR,LF,'or higher.',CR,LF,'$'

MSG4                DB      CR,LF,'Nothing to do: input file(s) is/are empty!',CR,LF
MSG4_LENGTH         EQU     $-MSG4

MSG5                DB      CR,LF,'SHUFFLE succesfully completed.',CR,LF
MSG5_LENGTH         EQU     $-MSG5

DIFF                DB      CR,LF,'WARNING: The two input files hold different'
                    DB      CR,LF,'numbers of bytes. The output file will be'
                    DB      CR,LF,'twice the length of the longest input file,'
                    DB      CR,LF,'and the missing bytes will be zeroed.',CR,LF
DIFF_LENGTH         EQU     $-DIFF

WR_ERR              DB      CR,LF,'Error writing output file.',CR,LF
WR_ERR_LENGTH       EQU     $-WR_ERR

DATA                ENDS

                    ;****************************************
                    ;*
                    ;* The stack.
                    ;*
                    ;****************************************
STACK               SEGMENT PARA STACK 'STACK'

                    DB      64 DUP (?)

STACK               ENDS

                    END     SHUFFLE
```

Van de bestuurstafel

De bestuurstafel stond deze keer in Overloon. Een plaatsje dat ergens achter Zeeland ligt. Na een hartelijk ontvangst en even bijpraten zijn we toen maar met de vergadering begonnen. Volgens de penningmeester had vrijwel iedereen z'n contributie betaald. Zo positief kon de redactie van ons lijfblad niet zijn. Het tekort aan copy doet ons binnenkort waarschijnlijk naar noodmaatregelen grijpen. Een van die noodmaatregelen zou kunnen zijn dat we ter vulling maar eens het getalletje PI afdrukken. Volgens de laatst bekende gegevens wordt het dan toch nog een behoorlijk dikke P Kenner. Het 68k project zat met het probleem dat het in gebruik zijnde takenpakket en router te klein was om een goede lay-out te maken voor de printboer. De projectleider en uw voorzitter hebben inmiddels een hoopvol gesprek gevoerd met een leverancier van een pakket dat dat wel aan kan. In de toekomst zou het zelfs voor andere projecten gebruikt kunnen worden. We houden u op de hoogte. Ons BBS draait nu met drie lijnen en is daardoor nu beter bereikbaar dan ooit daarvoor. We vonden het als bestuur toch wel jammer dat er geen leden hebben gereageerd op de functie voor PR-medewerker. Als dit u op andere gedachten brengt neem dan even contact op met een van de bestuursleden. We zullen dat zeker op prijs stellen. De vereniging beschikt thans over een creditcard. Jullie horen binnenkort hoe de regels voor het gebruik door leden gaan worden. Uiteraard zal e.e.a. via de penningmeester lopen. Als je denkt nu al dringend van dat ding gebruik te moeten maken, neem dan eens contact met hem op. Vanuit de regio Enschede werd gevraagd of er geen regionale bijeenkomsten georganiseerd konden worden. Het bestuur staat daar niet afwijzend tegenover. Zij die dat willen organiseren worden verzocht om contact met het bestuur op te nemen, zodat dat onder de vlag van de KGN kan.

Het tekort aan copy doet ons waarschijnlijk naar noodmaatregelen grijpen. Een van die noodmaatregelen zou kunnen zijn dat we ter vulling maar eens het getalletje PI afdrukken.

Op de bijeenkomst in Geldrop stond het verslag van de kascontrolecommissie ter behandeling. Helaas konden de heren Steunenberg en Herlaar daarbij niet aanwezig zijn. Enkele opmerkingen uit verslag zijn:

- Bewondering voor de professionele wijze waarop de boekhouding gevoerd wordt;
- Een compliment aan de vrouw van de penningmeester, omdat zij hem met raad en daad bijstaat;
- De financiële inhoud klopt tot op de cent nauwkeurig.

Met andere woorden: zij hebben de boekhouding over 1991 dan ook goedgekeurd.

De aanwezige leden hebben zich daar unaniem bij aangesloten. Namens het bestuur wil ik de beide heren bedanken voor de voortreffelijke manier waarop zij hun taak hebben volbracht.

Leden die in Geldrop niet aanwezig waren hebben de voordracht van Adri Hankel gemist. Het woord gemist zou eigenlijk in hoofdletters moeten worden geschreven. De uiteenzetting die Adri gaf, over single chips, blonk uit door duidelijkheid en was zeer interessant. Zijn uitleg was een voorbereiding op de werking van SIMON, een seriële interface monitor. Binnenkort kunnen we hierover meer verwachten in dit lijfblad. Van de aanwezigen heb ik gehoord dat ze daar met spanning naar uit zullen zien. Jullie voorzitter doet dat zeker. Adri heeft bij mij in ieder geval de soldeerhonger weten op te wekken.

Zo het is weer laat geworden. Dat betekend dat de woorden steeds moeilijker aan mijn brein ontspruiten en dat het gapen steeds meer toeneemt. Blijft over dat ik hoop jullie op 23 mei te mogen begroeten in Almere-Stad.

Jullie hongerende voorzitter

Informatie

De μP Kenner (De microprocessor Kenner) is een uitgave van de KIM gebruikersclub Nederland. Deze vereniging is volledig onafhankelijk, is statutair opgericht op 22 juni 1978 en ingeschreven bij de Kamer van Koophandel en Fabrieken voor Hollands Noorderkwartier te Alkmaar, onder nummer 634305. Het gironummer van de vereniging is 3757649.

De doelstellingen van de vereniging zijn sinds 1 januari 1989 als volgt geformuleerd:

- Het vergaren en verspreiden van kennis over componenten van microcomputers, de microcomputers zelf en de bijbehorende systeemsoftware.
- Het stimuleren en ondersteunen van het gebruik van micro-computers in de meer technische toepassingen.

Om deze doelstellingen zo goed mogelijk in te vullen, wordt onder andere 5 maal per jaar de μP Kenner uitgegeven. Verder worden er door het bestuur per jaar 5 landelijke bijeenkomsten georganiseerd, beheert het bestuur een Bulletin Board en wordt er een softwarebibliotheek en een technisch forum voor dediverse systemen in stand gehouden.

Landelijke bijeenkomsten:

Deze worden gehouden op bij voorkeur de derde zaterdag van de maanden januari, maart, mei, september en november. De exacte plaats en datum worden steeds in de μP Kenner bekend gemaakt in de rubriek Uitnodiging.

Bulletin Board:

Voor het uitwisselen van mededelingen, het stellen en beantwoorden van vragen en de verspreiding van software wordt er door de vereniging een Bulletin Board beschikbaar gesteld.
Het telefoonnummer is: 053-328506 of 053-303902 .

Software Bibliotheek en Technisch Forum:

Voor het beheer van de Software Bibliotheek en technischeondersteuning streeft het bestuur ernaar zgn. systeemcoordinators te benoemen. Van tijd tot tijd zal in de μP Kenner een overzicht gepubliceerd worden. Dit overzicht staat ook op het Bulletin Board.

Correspondentie adres

Alle correspondentie betreffende verenigingszaken kan gestuurd worden aan:

KIM Gebruikersclub Nederland
Postbus 1336
7500 BH Enschede

Het Bestuur

Het bestuur van de vereniging wordt gevormd door een dagelijks bestuur bestaande uit een voorzitter, een secretaris en een penningmeester en een viertal gewone leden.

Tonny Schäffer (voorzitter)
Paul Krügerstraat 27
7532 PW Enschede
Telefoon 053-613678

Jacques H.G.M. Banser (penningmeester)
Haaksbergerstraat 199
7513 EM Enschede
Telefoon 053-324137

Gert van Opbroek (secretaris)
Bateweg 60
2481 AN Woubrugge
Telefoon 01729-8636

Joost Voorhaar (redactie μP Kenner)
Jekerstraat 67
7523 VP Enschede
Telefoon 053-333483

Jan D.J. Derksen (DOS65 coordinator)
Verfailleweg 434
1783 BP Den Helder
Telefoon 02230-28168

Geert Stappers (KGN/68k coordinator)
Engelseweg 7
5825 BT Overloon
Telefoon 04788-1279

Ereleden:

Naast het bestuur zijn er een aantal ereleden, die zich in het verleden bijzonder verdienstelijk voor de club hebben gemaakt:

Erevoorzitter:
Siep de Vries

Ereleden:
Mevr. H. de Vries-van der Winden
Anton Müller
Rinus Vleesch Dubois