

De μ P Kenner



**Veertiende jaargang nr. 3
Augustus 1990**

67



In dit nummer o.a.:

**i8048 Assembler
Stoom Cursus Debug
Disk Copy voor DOS65
Minix, een ander Operating System
Assembler programmeren op de PC**

Inhoudsopgave

De μ P Kenner

Nummer 67, augustus '90
Verschijnt 5 maal per jaar
Oplage: 250 stuks
Druk: FEBO Offset, Enschede

De redactie:

Gert van Opbroek
Bram de Bruine
Antoine Megens
Nico de Vries
Joost Voorhaar

Eindredactie:

Gert van Opbroek

Vormgeving:

Joost Voorhaar

Redactieadres:

Gert van Opbroek,
Bateweg 60
2481 AN Woubrugge

De μ P Kenner nummer 68 verschijnt op
20 oktober 1990.

Copijsluitingsdatum voor nummer 68 is
vastgesteld op 6 oktober 1990.

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

Copy voor het blad dient bij voorkeur van de leden afkomstig te zijn. Deze copy kan in papier-, maar liever in machine-leesbare vorm opgestuurd worden aan het redactieadres. Copy 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.

De redactie houdt zich het recht voor copy zonder voorafgaand bericht niet of slechts gedeeltelijk te plaatsen of te wijzigen. Geplaatste artikelen blijven het eigendom van de auteur en mogen niet zonder diens voorafgaande schriftelijke toestemming door derden geplubliceerd worden, in welke vorm dan ook.

De redactie noch het bestuur kan verantwoordelijk gesteld worden voor toepassing(en) van de geplaatste copy.

Vereniging

Uitnodiging voor de clubbijeenkomst	5
Van de bestuurstafel	50
Info	51

Algemeen

Redactioneel	4
Ex Libris	11
Ook Gij, Brutus... ..	17

Fundamenten

Transputers	6
-------------------	---

Datacommunicatie

Methoden en technieken voor datacommunicatie (Deel 3)....	30
---	----

Systemen

Diskcopy voor DOS-65	12
De IBM-PC en z'n klonen (Deel 9)	18
Stoomcursus DEBUG.COM	25
8 bits microcontrollers van intel programmeren	39

Talen/software

Het programmeren van de 8088 in de IBM (Deel 1).....	7
To Share Or Not To Share, That's The Question	22
MINIX - Unix in een notedop	37
SDN file area's	38

Redactioneel

Op het moment dat u dit leest, zal de maand augustus wel zo'n beetje afgelopen zijn. Omdat in de hitte van de zomer de meeste mensen de computerhobby op een wat lager pitje zetten, kunnen we dus zeggen dat we weer aan een nieuw computerseizoen beginnen. Ook voor de redactie waren de hete dagen van de afgelopen maand enigszins problematisch. Het is moeilijk in de hitte de warme zolder op te zoeken om een artikel te gaan schrijven of om iets creatiefs met de computers te doen. Gelukkig is het, met de hulp van een aantal correspondenten, toch weer gelukt een blad samen te stellen en ik ben van mening dat de inhoud van het blad er dit keer ook weer best mag zijn.

Dat de MS-DOS machines ook binnen onze club steeds verder doordringen is te merken aan de kopij die we ter beschikking gesteld krijgen. Zoals bekend, is Nico al een tijd met een artikelenserie over MS-DOS machines bezig. Daar komt nu van Ruud Uphoff een serie bij. Deze serie van Ruud gaat over een onderwerp dat de oudere clubleden waarschijnlijk na aan het hart zal liggen, namelijk het programmeren in assembler. Evenals een DOS-65 systeem en een Junior kun je een MS-DOS machine natuurlijk ook in assembler programmeren. Dit wordt echter vrijwel niet gedaan, waarschijnlijk omdat de machines voornamelijk in bezit zijn van mensen die de bekende programma's gebruiken en verder hooguit eens in Turbo Pascal of Basic programmeren. Om een machine echter echt te leren kennen en de fijne kneepjes van het programmeren van een computer te leren, is het eigenlijk een must om eens een programma in assembler te schrijven. Met de artikelenserie van Ruud kunnen een aantal leden hier toe misschien overgehaald worden. Blijft natuurlijk de vraag wanneer de trotse bezitters van Amiga en Atari eens laten merken dat het niet alleen MS-DOS is wat de klok slaat. Ik heb onderhand de toezegging van een Macintosh bezitter dat hij zal zorgen voor wat kopij over de werking en de programmering van een dergelijk "event driven" systeem. Dus wie volgt?

Binnen het bestuur van de club hebben we het gehad over een mogelijke opvolger van DOS-65. DOS-65 is een binnen de club ontwikkeld systeem en wordt door een aantal mensen als het eigen systeem van de

KIM Gebruikersclub Nederland gezien. Dit systeem bestaat al een aantal jaren en binnen de club bestaat de behoefte aan een nieuw eigen systeem. Nu doet zich echter het feit voor dat het eigenlijk niet meer loont een systeem van de grond af op te bouwen. Het zelfbouwen van hardware is veel te kostbaar geworden en verder is het toch wel een vereiste ten minste één of meer programmeertalen ter beschikking te stellen. Het is dus vrijwel uitgesloten dat er vanuit de club nog een keer een compleet systeem op poten gezet wordt. We denken echter wel een andere mogelijkheid gevonden te hebben om de club een "eigen" systeem te geven. We praten dan niet over eigen hardware maar over een eigen operating systeem. Het bestuur overweegt namelijk om naast de huidige activiteiten het operating systeem MINIX min of meer te adopteren. Dit betekent dat de club zal trachten MINIX meer onder de aandacht van de leden te brengen en ook support voor dit operating systeem te gaan geven. Natuurlijk betekent dit niet dat de uitgezette beleidslijnen opeens veranderen, MINIX komt daar gewoon bij. De eerste stap op dit gebied zal gezet worden op de eerstvolgende bijeenkomst. Joost Voorhaar, onze layouter, zal daar een voordracht houden over MINIX en het operating systeem introduceren. Verder is ook op het bulletin board al een aparte area voor MINIX ingericht en daar is een grote hoeveelheid informatie over dit operating systeem te vinden. Als u in het bezit bent van een modem is het misschien zinvol dat u daar eens naar kijkt, want ik verwacht dat u de komende tijd nog veel over MINIX in de μ P Kenner zult lezen. Tenslotte willen we graag in contact komen met andere clubleden die kennis van of ervaring met MINIX hebben. We willen zo op relatief korte termijn een MINIX-kern in de club tot stand brengen van waaruit de verdere activiteiten op dit gebied ontwikkeld kunnen gaan worden.

Als laatste rest mij eigenlijk niets anders meer dan u ook nu weer veel leesplezier aan de nieuwe μ P Kenner te wensen en veel plezier met uw computerhobby. Verder blijft kopij in welke vorm en voor welk systeem dan ook natuurlijk van harte welkom.

Gert van Opbroek

Uitnodiging voor de clubbijeenkomst

Datum: 15 september 1990
Locatie: Wijkcentrum "De Ringvaart"
 Floris van Adrichemlaan 98
 2035 VD Haarlem
Telefoon: 023-363856

Entree: f 10,00

Thema: MINIX

Routebeschrijving

Auto

Komende uit de richting Utrecht, Amersfoort of Rotterdam:

Afslag Haarlem-Zuid; tweede stoplicht links; bij de tweede kruising met stoplichten links; Floris van Adrichemlaan.

Komende uit de richting Alkmaar:

afslag Haarlem-Zuid; verder zie boven.

Openbaar vervoer:

Vanaf het station Haarlem met buslijn 7, 71, 72 of 77; halte Floris van Adrichemlaan.

Programma:

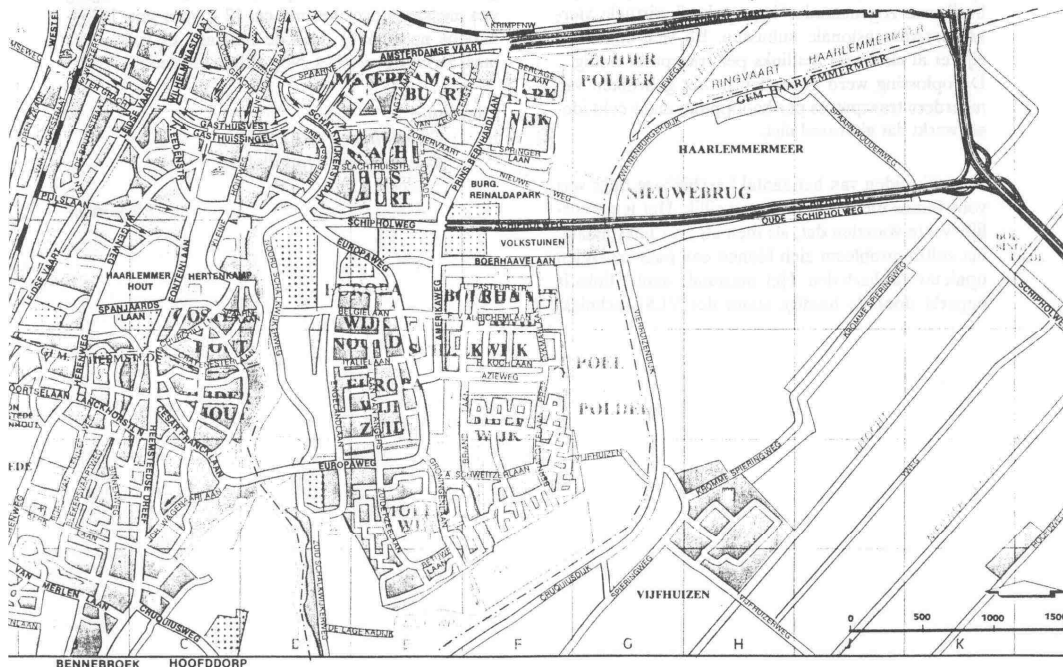
9:30 Zaal open met koffie
 10:15 Opening door de voorzitter Nico de Vries.
 10:30 Voordracht door Joost Voorhaar:
 "MINIX, een introductie"
 11:30 Forum en markt
 12:00 Lunchpauze, Koffie en broodjes op eigen kosten te verkrijgen.

Aansluitend het informele gedeelte bedoeld om kennis, ervaring Public Domain en eigen ontwikkelde software uit te wisselen met uw medeleden. Breng daarom ook uw eigen systeem mee! (En vergeet de snoeren niet...)

17:00 Sluiting

Attentie

Het is ten strengste verboden illegale kopieën van software te verspreiden. Aan personen die deze regel overtreden zal de verdere toegang tot de bijeenkomst ontzegd worden. Breng verder alleen software mee die u legaal in uw bezit heeft. Het bestuur aanvaardt geen enkele aansprakelijkheid voor de gevolgen van het in bezit hebben van illegale software.



Transputers

Zoals ik vorige aflevering aflevering al schreef, is er geen deel 4 over transputers. "Waarom staat er dan toch "Transputers" boven?", zult u zich wellicht afvragen. Wel, Murphy begon zich te bemoeien met mijn geschrijf. Wat wil namelijk het geval? Nog geen week nadat deel drie van de serie bij u op de deurmat lag, vertelde iemand mij dat Inmos bezig was met een nieuwe transputer. Deze transputer, die bij Inmos de nog de naam "H1" draagt, heeft een aantal eigenschappen die toch wel vermeldingswaard zijn.

De T800 en zijn maatjes hebben ieder de beschikking over slechts vier links. Voor de eerste proefsels is dat meer dan voldoende; een driedimensionaal netwerk is moeilijk programmeerbaar en een boomstructuur met meer dan drie vertakkingen per knoop is ook niet echt handelbaar. Althans, dat dachten de ontwerpers van Inmos. De praktijk leerde echter dat er wel degelijk programmeurs waren die prijs stelden op een driedimensionaal netwerk waarin iedere transputer met iedere andere transputer in het netwerk moest kunnen communiceren. Dat leverde de nodige problemen op, want in een kubus zitten acht hoekpunten. Dat betekent dat er per transputer minimaal drie links nodig waren. Messages kunnen dan via derden aan de geadresseerde transputer verstuurd worden, hetgeen natuurlijk de nodige software-overhead met zich mee brengt. Moeilijker wordt het bij het gebruik van zogenaamde "hypercubes", virtuele viereen meerdimensionale kubussen. Bij vier dimensies zijn er al minimaal zes links per transputer nodig... De oplossing werd gevonden in het gebruiken van meerdere transputers per knooppunt, maar echt ideaal werkt dat uiteraard niet.

Het uitbreiden van het aantal beschikbare links was voor Inmos niet zo heel erg moeilijk. Het is natuurlijk wel te voorzien dat, als men nu acht links maakt, het zelfde probleem zich binnen een paar maanden opnieuw zal herhalen. Het maximale aantal links is beperkt door de huidige stand der VLSI-techniek.

In de H1 is daarom voor een andere oplossing gekozen. Er wordt niet meer gebruik gemaakt van een beperkt aantal direct aanspreekbare links, maar van een virtuele adressering. De fysieke adressering wordt overgenomen door de speciale externe bouwsteen C104, een uitgebreide versie van de reeds in gebruik zijnde programmeerbare link-switch C004. De C104 lijkt nog het meeste op een soort telefooncentrale waarbij gewerkt wordt met speciale pakketjes informatie. Het ding heeft de naam "routing switch" meegekregen. Deze bouwsteen kan vijf maal zo veel informatie verstouwen als de snelste transputerlink; informatie wordt verwerkt met een duizelingwekkende snelheid van 100 MBit per seconde (full duplex)... Het ding kan adressen verwerken van 1 of 2 bytes, hetgeen betekent dat er maar liefst 65536 transputers in een netwerk min of meer direct aangesproken kunnen worden!

Andere verbeteringen zijn er onder meer te verwachten op het gebied van geheugenbescherming, link-multiplexing en IEEE floating point errors. De H1 bestaat volgens mijn informatie nog niet "in silicium", maar Inmos is druk bezig met het testen van enige prototypen van sub-systemen en met de simulatie van de totale chip. Deze simulaties wijzen nu duidelijk in de richting van grote snelheidswinst bij de toepassing van transputers in hypercubes. Bij het gebruik van een 64-knoops netwerk is de vertraging van messages over het net ca. 12 s, voor een 16384-knoops netwerk () doet de message er ruim twee maal zo lang over: 27 s. Nog altijd niet slecht...

Literatuur:

C't, nummer 7 (juli) 1990
Byte, nummer 4 (april) 1990

J. Voorhaar

Het programmeren van de 8088 in de IBM (Deel 1)

Inleiding.

Het hart van elke MS-DOS machine, met name de IBM PC, wordt gevormd door een 8088 of een van de opvolgers van deze processor. Het leek mij dan ook onzinnig om over het programmeren van deze processor een serie artikelen te gaan schrijven en daarbij deze typische 8088 omgeving buiten beschouwing te laten. Deze serie gaat dan ook niet over de 8088 machinetaal maar over het programmeren van een IBM of overeenkomstig MS-DOS systeem in assembler. Deze serie is niet zonder meer geschikt voor hen die volslagen beginners zijn op het gebied van programmeren in een assembleertaal. Van de lezer wordt verwacht dat hij inmiddels vertrouwd is met het verschijnsel microprocessor en het binaire en hexa-decimale getallenstelsel. Eveneens wordt van de lezer verwacht dat hij bekend is met het besturingssysteem MS-DOS op gebruikersniveau. Wie hier nog volkomen in het duister tast mag ik verwijzen naar een aantal eerder in dit blad verschenen artikelen:

- "Getallen" door Gert van Opbroek. μP kenner nummer 58 en verder.
- "Computers" door Gert van Opbroek. μP kenner nummer 59 en verder.
- "De IBM-PC en zijn klonen" door Nico de Vries. μP kenner nummer 59 en verder. Met name is het van belang dat u deel 2 (nummer 60 dus) aandachtig leest.

Benodigde documentatie.

Een programmeur moet beschikken over de nodige naslagwerken. Op zijn minst dient in de nabijheid van de computer de documentatie van MS-DOS met alle interrupts en functies, de documentatie van alle BIOS functies en de complete instructieset van de microprocessor aanwezig te zijn. Gelukkig is het niet nodig daaraan grote bedragen uit te geven. Navolgende drie handige boekjes maken dat u alles in huis hebt. U hebt ze overigens niet nodig om deze serie te kunnen begrijpen. Er zullen alternatieven genoeg zijn, maar dit is een mogelijkheid:

- "PC-Zakboekje IBM ROM BIOS" (Ray Duncan, Kluwer, ISBN 90 201 2232 0)
- "PC-Zakboekje MS DOS" functies (Ray Duncan, Kluwer, ISBN 90 201 2233 9)
- "Zakboek 8086/8088" (R.Erskine, Delfia Press, ISBN 90 6449 049 X)

Een assembler kiezen.

MASM (MacroASSEMBLER) van Microsoft is de de facto standaard voor de 8088. Helaas is MASM ongelooflijk traag. Gelukkig is er tegenwoordig een goed alternatief in de vorm van Turbo Assembler (TASM) van Borland. Het is voor 100% compatible

met MASM maar ongelooflijk veel sneller en ook nog goedkoper. Helaas is de documentatie minder duidelijk dan die van MASM. In deze serie dient onder MASM te worden verstaan: MASM of TASM.

Editor Assembler en Linker.

TASM noch MASM zijn voorzien van een editor. Het bronbestand (Source file) moet worden aangeemaakt met een willekeurige editor. U kunt bijvoorbeeld gebruik maken van navolgende editors:

- EDLIN is een deel van MS-DOS als erfenis uit CP/M. Het gebruik van EDLIN is echt monnikenwerk en doet denken aan het "teletype" tijdperk.
- WordStar in de "Non Document" mode.
- Het "Note Pad" van "SideKick". De omvang van de source is dan beperkt. *)
- De editors van Turbo Pascal en/of Turbo C *)
- De editor in PC-Fix Versie 3.00 kan 48K files aan. Versie 1.00 40K *)
- De editor in PC-Tools (vanaf versie 4.11).

Elke andere editor die een schone ASCII tekst kan maken. Een regel moet met een CR en een LF eindigen en verder is alleen TAB als stuurteken toegestaan (FF mag ook maar werkt niet)

Nog dit voorjaar hoop ik EDFIX ter beschikking van de club te kunnen stellen. Dat is een complete programmeeromgeving voor MASM/TASM met editor en een beknopte beschrijving van alle 8088 instructies en de meest gebruikte assemblerdirectives in de "on line" help file.

*) Opmerking: Al deze editors werken met de WordStar commandoset.

Het bronbestand moet op disk(ette) staan en een extensie .ASM hebben. De assembler maakt van deze brontekst een "relocatable object file". Met een linker moet daarvan een door MS-DOS executeerbare EXE file worden gemaakt. Moet het programma een COM file worden, dan moet bovendien deze EXE file nog eens met het MS-DOS utility EXE2BIN in een COM file worden omgezet. De linker maakt eveneens deel uit van MS-DOS, maar zowel bij MASM als TASM wordt een linker meegeleverd, die wat meer faciliteiten biedt. Die van TASM is o.a. in staat om rechtstreeks een COM file aan te maken.

Adressen in de 8088

Een 8088 kan maximaal 1Mb direct adresseren hetgeen een 20 bits adres vereist. Het is buitengewoon moeilijk om een instructieset voor een microprocessor te bedenken met een adresbereik dat de 16 bits

teboven gaat. Een volledig adres zou specificatie van 3 bytes vergen waarin 4 bits ongebruikt blijven. Bovendien zal in de praktijk lang niet elk programma een omvang van meer dan 64K hebben en indien dat toch gebeurt kan het bestaan uit een aantal kleinere modulen. De 8088 staat ons dan ook toe adressen te specificeren van 16 bits alsof we in een traditionele 64K omgeving aan het programmeren zijn. Een dergelijk gebied van 64K noemen we een SEGMENT. Bovendien kunnen we zondig het gehele segment gebruiken voor de programmacode, omdat de 8088 ons toestaat om voor variabelen een ander segment te kiezen en voor de stack weer een ander segment. Zo'n segment kan beginnen op elk absoluut 20 bits adres dat een veelvoud van 16 is. Het eerste byte in zo'n segment is dan adres 0000H. Dit maakt bovendien een programma volledig verplaatsbaar naar elk willekeurig adres, mits dat adres een veelvoud van 16 is, zonder dat aan het programma ook maar iets behoeft te worden aangepast. Zo'n adres dat deelbaar is door 16 wordt "segmentadres" of "paragraaf" genoemd. Dit wil overigens niet zeggen dat een programma niet groter zou kunnen zijn dan 64K. De 8088 beschikt namelijk over speciale versies van de spronginstructie (JMP) en de subroutineoproep (CALL) om code in een ander segment te executeren. In welke segmenten de processor code interpreteert, data opslaat en waar de stack is, ligt vast in speciaal daarvoor bestemde registers. De Segment registers. In zo'n segment register zouden we dus alsnog een 20 bits adres moeten opgeven. Aangezien dat adres altijd deelbaar is door 16 en de laagste 4 bits dus altijd nul zijn, kan ook hier worden volstaan met een 16 bits waarde, waarachter we (in hex) dus altijd een nul moeten denken. Als we een volledig adres moeten specificeren gebeurt dat in een SEG:OFFSET notatie. Heel beleefd met twee woorden dus. We zullen nu de vier segmenten bespreken die altijd in de 8088 bekend zijn omdat de begin paragraaf in de bijbehorende segmentregisters staat.

- CODE SEGMENT. Het register heet CS. In dit segment staat het programma zelf. De CODE. Het adres waarop de processor bezig is een instructie te lezen en uit te voeren staat in een register dat voor de programmeur niet direct toegankelijk is: IP van "Instruction Pointer". Het volledig adres waarop de processor aan het werk is wordt dus aangeduid met CS:IP. In register CS kan niet rechtstreeks worden geschreven maar het kan wel worden uitgelezen.
- STACK SEGMENT. Het register heet SS. Alle overige segmentregisters kunnen we wel naar believen veranderen. Het SS register bevat de paragraaf waarop de stack begint. We kunnen dus desnoods over een stack van 64K beschikken. Omdat we SS naar believen kunnen wijzigen kunnen we voor verschillende doeleinden over net zoveel verschillende stacks beschikken

als we willen. De complete stackaanwijzing gebeurt door de combinatie SS:SP. Dat tweede register is de Stack Pointer die we als een gewoon register kunnen benaderen. De stack van de 8088 staat "op zijn kop", een verschijnsel dat we ook bij andere processors zoals de 6502 aantreffen. Data op de stack zetten heeft dus een verlagende van de stack pointer tengevolge.

- DATA SEGMENT. Het register heet DS. Met de paragraaf in DS geven we aan waar het segment begint waarin data wordt opgeslagen. Alle instructies die data lezen of opslaan (Later leert u een enkele uitzondering kennen) doen dat op een adres dat is gespecificeerd als een offset in het datasegment. Er zijn speciale instructies die we "segment override" noemen om data in andere segmenten dan het datasegment te benaderen.
- EXTRA SEGMENT. Het register heet ES. Gewoon prettig om een extra segment register te hebben. We kunnen er uit lezen of er in schrijven door dat aan te geven met een "segment override". Er is echter een speciale instructie die aan neemt dat de data niet in het datasegment maar in het extra-segment moet worden opgeslagen. Daarover meer in latere afleveringen van deze serie.

De vier data registers: AX, BX, CX en DX

Om te beginnen zijn hun namen gemakkelijk te onthouden omdat ze gewoon met de eerste vier letters van het ABC beginnen. De X erachter geeft aan dat we met de 16 bit versies te doen hebben. Alle vier zijn ze te gebruiken als twee afzonderlijke 8 bit registers. Ze heten dan AL, AH, BL, BH, CL, CH, DL, DH waarbij de letters L en H voor "Low" en "High" staan om aan te geven welke helft van het 16 bits register we bedoelen. Er zijn een massa instructies waarin al deze 4 of 8 registers dezelfde mogelijkheden bieden. Als 16 bits register hebben ze echter ieder ook nog een speciale taak. Die is weer gemakkelijk te onthouden doordat ook die functies in relatie staan tot de letters ABCD.

- AX of Accumulator. Een aantal rekenkundige en logische bewerkingen zoals delen en vermenigvuldigen vereisen het gebruik van AX. Bovendien is er een speciale adresseermethode voor AX die compacter en sneller is. Bij gebruik van een assembler zoals MASM, kan de programmeur niet bewust voor die adresseermethode kiezen. MASM gebruikt hem automatisch als dat kan.
- DX of Double word register. Een aantal instructies (vermenigvuldigen en delen) kunnen betrekking hebben op een 32 bits operand. DX is dan het tweede (High) woord van de 32 bits accumulator, waarvan AX de andere helft is

- ### Speciale registers

SI of Source Index. Voor beschrijving zie onder DI hierna.

DI of Destination Index. Wederom zijn de namen te onthouden omdat de speciale functies van deze registers in hun naam is verwerkt. De I staat voor Indexregister. De S en de D staan voor respectievelijk "Source" en "Destination" het geen te maken heeft met een sub-set van instructies speciaal voor het verplaatsen van geheugen blokken van het datasegment naar het extra segment, en het zoeken naar strings in het geheugen.

BP of Base Pointer. Dit register is net als BX, SI en DI te gebruiken als basisregister of indexregister. Indien echter gebruik wordt gemaakt van BP, is het uiteindelijk address een offset in het stacksegment, dus niet in in het datasetage.

Tenslotte is er natuurlijk een vlaggenreger. In de 8088 is dat 16 bits groot maar lang niet alle bits heb-

ben een functie. De ongedefinieerde bits zijn in navolgend overzicht met een sterretje aangegeven.

bit:	15	14	13	12	11	10	09	08
vlag:	*	*	*	*	OF	DF	IF	TF

bit:	07	06	05	04	03	02	01	00
vlag:	SF	ZF	*	AF	*	PF	*	CF

OF of Overflow flag. Geeft aan dat de uitkomst van een berekening te groot is. Er zijn een of meer bits verloren gegaan.

- DF of Direction flag. Wordt in een latere aflevering toegelicht
- IF of Interrupt flag. Indien we IF wissen accepteert de processor geen andere interrupts dan NMI
- TF of Trace flag. Kan worden gezet om de processor stap voor stap te laten werken. Wordt gebruikt in debuggers.
- SF of Sign flag. Geeft aan dat het meest significante bit (teken) van een waarde is gezet.
- ZF of Zero flag. Geeft aan dat het resultaat van een bewerking nul is.
- AF of Auxiliary carry flag. Geeft aan dat een carry of borrow heeft plaats gehad tussen twee halve bytes. (van 0xH naar 1xH)
- PF of Parity flag. Geeft aan dat het resultaat van een bewerking een even pariteit heeft
- CF of Carry flag. Geeft aan dat het resultaat van een bewerking groter is dan de grootste mogelijke waarde. Het bit dat daarbij verloren is gegaan staat nu in CF.

De adresseermethoden van de 8088

Een groot aantal instructies van de 8088 werkt met twee operands volgens de algemene vorm:

instructie doelooperand,bronoperand

Bijvoorbeeld:

MOV AX,BX

Hiermee leren we dan de eerste instructie kennen. MOV (move) doet de doeloperand gelijk worden aan de brondoperand. Dit type instructie is zeer ge-

Doel	Bron	Notatie	Voorbeeld
register	constante	reg,imm	MOV AX,1234H
register	register	reg,reg	MOV AX,BX
register	geheugen	reg,mod	MOV AX,MYVAR
geheugen	register	mod,reg	MOV MYVAR,AX
segregister	register	seg,reg	MOV DS,AX
register	segregister	reg,seg	MOV AX,DS
geheugene	constante	mod,imm	MOV MYVAR,12H

*Fig. 57: schrijfwijzen
en
adresseermogelijkheden*

schikt om vanuit te gaan bij het beschrijven van de adresseermethoden die de 8088 kent zoals we die in assembleertaal moeten specificeren. We komen navolgende mogelijkheden tegen voor doel- en bronoperand, waarbij we moeten onthouden dat een register en een segmentregister twee verschillende dingen zijn.

Een adresseermethode waarbij de operand een constante is wordt meestal "immediate" genoemd, vandaar de afkorting "imm", Nogal vreemd lijkt de notatie "mod" voor een geheugenlocatie. Het is de afkorting voor "mode" en het geeft aan dat achter de simpele kreet "geheugen" iets meer schuil gaat dan alleen maar een adres. De 8088 biedt namelijk zeer uitgebreide mogelijkheden tot indexeren. Het effectief adres wordt daarbij door de processor berekend uit de som van maximaal 3 grootheden: Een basis, een index en een offset. Die offset is een constante. In de voorbeelden bij mod,reg en reg,mod hebben we een symbool gebruikt. Dat is verplicht in MASM. Normaliter zullen we daarmee geen moeite hebben, omdat niet de programmeur maar de assembler bepaald op welk adres een variabele in het datasegment terecht komt. Het zal de programmeur dus een zorg zijn op welke offset MYVAR in werkelijkheid ligt. Er zijn wel mogelijkheden om toch naar een hard adres te verwijzen, maar die mag u uit de handleiding van de assembler halen. Onthoudt liever dit: Naar een hard adres verwijzen is geen juiste techniek en het is altijd te voorkomen. MASM graaft hier een valkuil waar menig programmeur ingevallen is. MASM is namelijk een defacto standaard die echter sterk afwijkt van de standaard die Intel ooit bedacht heeft. Die oorspronkelijke standaard wordt wel gebruikt in het monitorprogramma "DEBUG" dat deel uit maakt van MS-DOS. Daar wordt aan een adres gerefereerd door dit tussen haken te plaatsen: MOV AX,[1234H]. Wordt deze notatie in MASM gebruikt, dan worden de haken eenvoudig genegeerd en dus wordt dan MOV AX,1234H (reg,imm) uitgevoerd! Laten we nu maar eens zien welke mogelijkheden de 8088 kent om naar geheugenlocaties te verwijzen:

- Direct. Het adres (een offset) wordt direct gespecificeerd in de vorm van een symbool.
- Via een register. Alleen de basisregisters BX en BP en de indexregisters DI en SI zijn daarvoor te gebruiken. Het register bevat het effectief adres. De naam van het register moet tussen haken worden geplaatst: MOV AX,[SI]. Desgewenst mag deze methode nog worden uitgebreid met een constante welke dan bij de registerinhoud wordt opgeteld. Voorbeeld: MOV AX,[MYVAR + SI]. Men mag in dit geval de haken weglaten omdat ook zonder dezen de expressie eenduidig is. Ook de vorm MYVAR[SI] is toegestaan. Wordt BP gebruikt, dan adres-

seert men naar het stacksegment, niet naar het datasegment.

- Via twee registers. Een register moet een basisregister zijn (BX of BP) en het andere moet een indexregister zijn (SI of DI). De som van de inhoud van beide registers is het effectief adres. Ter verduidelijking mogen haken worden gebruikt: MOV AX,[BX + SI]. De uitdrukking [BX + SI] is equivalent aan [BX][SI] en ook aan BX + SI. Desgewenst mag deze methode nog worden uitgebreid met een constante welke dan bij de som van de inhoud van de registers wordt opgeteld. Voorbeeld: MOV AX,[MYVAR + SI + BX] of MOV AX,MYVAR + BX + SI evz. Denk er om dat als een van de registers BP is, men het stacksegment adresseert en niet het datasegment.

Data typen

MASM eist dat datatypen en registertypen met elkaar in overeenstemming zijn. Instructies als MOV AX,BL zijn onmogelijk. (de 8088 kent ze niet) Indien een variabele in het datasegment als byte is gedeclareerd kan men daar geen woord uit lezen. Omgekeerd staat MASM niet toe dat een byte uit een locatie wordt gelezen die als woord is gedefinieerd. Toch zal het soms nodig zijn twee opeenvolgende bytes als woord te benaderen of uit een woord alleen het eerste of tweede byte te lezen. Dat kan door het gebruik van override clausules. Ik noem hier de twee meest gebruikte:

```
MOV AL,BYTE PTR MYWORDS[BX]
      (Als MYWORDS een word-array is)
MOV BH,BYTE PTR ANY_WORD
      (Als ANY_WORD een word is)
MOV AX,WORD PTR FIRST_BYTE
      (Als FIRST_BYTE de eerste van twee is)
```

Segment override

Normaliter gaat de processor er vanuit dat alle data wordt gevonden in het datasegment. Wil men echter een variabele in een ander segment benaderen, dan kan dat door de operand vooraf te laten gaan door een segment override. In feite specificeren we dan een volledig adres.

Voorbeelden:

```
MOV AX,ES:[BX + SI]
MOV AX,CS:VECTOR_1
```

Hoe we al deze adresseermethoden moeten toepassen en waarvoor we ze kunnen toepassen is het beste te leren aan de hand van werkende programmvorbelden. We hebben nu genoeg droge theorie verkocht en gaan over tot programmeren onder MS-DOS.

Ruud Uphoff

Ex Libris

Hij werkte er pas twee dagen. Toen kwam zijn collega binnen en vertelde hem dat er een balansverschil van \$0,75 op de computer rekening voorkwam. Iemand had voor 75 cent CPU tijd gebruikt en het niet betaald. Nu is een verschil van enkele duizenden dollars meestal te wijten aan een opzet, maar een verschil dat achter de komma staat duidt meestal op een afrondingsfout in de programmatuur.

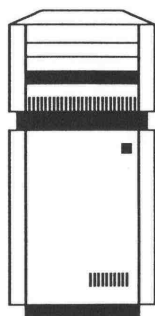
Zo rolde Clifford Stoll, een nogal verward overkomende astronoom, in een wereld van communicatie netten, inbrekers en spionage. De 75 cent bracht hem op het spoor van een hacker die de computer waar Clifford het beheer over voerde als knooppunt gebruikte voor allerlei uitstapjes naar militaire installaties. Het spoor van de hacker voerde uiteindelijk terug naar Duitsland, naar een hacker die de gevonden Amerikaanse staatsgeheimen voor redelijke geldbedragen aan de KGB verkocht.

Stoll vertelt op een spannende wijze in het boek "Het Koekoeksei" hoe de hacker misbruik kon maken van de bestaande verbindingen en hoe hij stapje voor stapje dichterbij de dader kwam. Hij beschrijft zijn problemen met instanties waar hij liever niets mee te maken wilde hebben, de inbreuk die de hacker op zijn dagelijks leven maakte en de lardeert dat rijkelijk met gebeurtenissen uit zijn ietwat verstrooide en wereldvreemde dagelijks leven. Dit alles wordt met kennis van zaken exact toegelicht.

Voor iedereen die regelmatig met grotere systemen in aanraking komt is dit boek een aanrader, voor zij die het beheer voeren over VAX-en, DEC's en andere mainframes of minicomputers is het een absolute must.

Clifford Stoll - "Het Koekoeksei"
Originele titel: "The Cuckoo's egg"
ISBN 90-269-4167-6
Prijs: ca. f 30,00.

BBS "The Ultimate" For all systems



Tel.: 053 - 303902

van 10:00 tot 22:00: V21, V22, V22b en V23
van 22:00 tot 10:00: V21, V22, V22b, 9600/HST, V42b

Diskcopy voor DOS-65

Met behulp van het onderstaande programma is het mogelijk om een complete schijf, inclusief de directory-structuur naar een andere schijf te kopiëren.

Frank Bens

```
; File :          DCOPY.MAC
;
;          Written by Frank Bens for DOS65
;
;          Creating date: 14-03-90
;
;          opt      nogen

waitfl      equ    $d9          ; wait flag
track       equ    $da          ; track number
trkcopy     equ    $db          ; copy of track number
trkmax      equ    $dc          ; max number of tracks
sector      equ    $dd          ; sector number
seccopy     equ    $de          ; copy of sector number
secmax      equ    $df          ; max number of sectors
rwpoin      equ    $e8          ; read/write pointer
srdrv       equ    $f0          ; source drive number
tardrv      equ    $f2          ; target drive number

datablk     equ    $0200        ; start address data block
sisblk      equ    $9f00        ; System Info Sector (256 bytes)

input       equ    $c020        ; get a character
prchar      equ    $c023        ; print a character
inecho      equ    $c026        ; get and print a character
crlf        equ    $c02f        ; print a cr & lf
hnout       equ    $c035        ; print a hex-nibble
hexout      equ    $c038        ; print a hex-byte
prttext     equ    $c03b        ; print a string until $00
sopt        equ    $c068        ; scan options
spar        equ    $c06b        ; scan parameters
ermes       equ    $d0b7        ; print error message
readsec     equ    $d0c0        ; read a sector
wrchsec     equ    $d0c9        ; write & verify a sector
posit       equ    $f024        ; position the cursor

;**** Macro definitions ****

poscur      macro    xpos,ypos
            ldx      #xpos
            ldy      #ypos
            jsr      posit
            endm
```


print	macro	\$text	
	jsr	prtext	
	fcc	\$text,0	
	endm		
	org	\$a000	
	fcc	\$c8,\$c5,\$cc,\$d0,'xxx'	
	fcc	'Function : Copying an Entire Disc.\r'	
	fcc	'Syntax : DCOPY [-W] sourcedrive,targetdrive\r'	
	fcc	'Options : -W : Wait to allow change of discettes.\r'	
	fcc	'Abbreviation : None.\r'	
	fcc	'Example : DCOPY 1,0 copy disc from drive 1: to drive 0:\r'	
	fcc	'Note : The source and target discs must'	
	fcc	'be the same size discs.',0	
er	jmp	error	; error exit
dcopy	print	'\\f\\t\\t\\t*** DISKCOPY ***'	
	jsr	sopt	; check option
	fcc	'W',0	; option is W
	stx	waitfl	; save option
	bcs	er	; branch on option error
	ldx	#-1	; preset parameters
	stx	srcdrv	
	stx	tardrv	
	ldx	##10100000	
	jsr	spar	; get parameters
	fcc	srcdrv,tardrv,0	; 2 parms
	bcs	er	; branch on error
	lda	srcdrv	; get source drive number
	ora	tardrv	; 'or' with target drive number
	bpl	waityn	
	poscur	1,3	; position the cursor
	print	'Which drive contain the source disc? : '	
	jsr	inecho	; wait for input
	and	#3	; mask only 2 LSB's
	sta	srcdrv	; and save it
	print	'\\rWhich drive contain the target disc? : '	
	jsr	inecho	
	and	#3	
	sta	tardrv	
waityn	poscur	1,6	; position the cursor
	ldx	#-1	
	lda	tardrv	; get target drive number
	cmp	srcdrv	; check target with source drive
	beq	setwait	; branch if equal
	lda	waitfl	; get wait flag
	beq	cstart	; no wait statement ?
	print	'Hit a key when ready...'	
	jsr	input	; else wait...
	inx		; dont wait any more
setwait	stx	waitfl	

cstart	print	'\r\rCopy started...\r'	
	poscur	9,14	
	print	'track: 00 sector: 01'	
	ldy	#0	; track 0 / sector 1
	ldx	#0	
	stx	track	
;**** read track/sector ****			
readsrc	stx	trkcopy	
	iny		
	sty	sector	; save sector
	sty	seccopy	
	poscur	1,12	
	ldx	waitfl	; wait ??
	beq	prread	; No
	print	'Insert \Eisource\En disc and hit a key when ready...'	
	jsr	input	
prread	print	'\r\rReading'	
	jsr	filpoin	; set pointer of sector in memory
	ldy	sector	; restore sector
datain	jsr	ptrsec	; print track & sector number
	lda	srdrv	; get drive number
	jsr	readsec	; read data sector
	bcc	getbuf	; branch if no read error
	jmp	error	; error exit
getbuf	lda	track	
	bne	1.f	
	lda	sector	
	cmp	#1	
	bne	1.f	; only in track 0 / sector 1
	lda	datablk + \$21	; get max number of tracks
	sta	trkmax	; and save it
	lda	datablk + \$22	; get max number of sectors
	sta	secmax	; and save it
1	inc	rwpoim + 1	; next address in data block
	lda	rwpoim + 1	; get current address
	cmp	#\$a0	; end of data block
	beq	coptar	
	ldy	sector	; get sector number
	cpy	secmax	; all sectors scanned
	bne	2.f	; branch if not
	ldy	#0	; set first sector
	inc	track	; next track
2	iny		
	lda	track	; get track number
	cmp	trkmax	; all tracks/sectors copied ?
	bne	datain	; branch if not completed

;***** write and check sector *****

```

coptar      lda      trkcopy
            ldy      seccopy
            sta      track
            sty      sector
            poscur   1,12
            ldx      waitfl          ; wait ??
            beq      rdsisbl         ; No
            print    'Insert \Eitarget\En disc and hit a key when ready...'
            jsr      input

rdsisbl     lda      track
            bne      prwrite         ; branch if track = / 0
            lda      #sisblk&255     ; set read pointer to system block
            ldy      #sisblk > 8
            sta      rwpoin
            sty      rwpoin + 1
            lda      tardrv          ; get target drive number
            ldx      #0              ; set track number
            ldy      #1              ; set sector number
            jsr      readsec         ; read system sector
            bcc      gettrsc         ; branch if no read error
            jmp      error           ; error exit

notequ      poscur   1,10
            print    'The target disc must be the same size as the source disc.'
            jmp      exit

gettrsc     lda      sisblk + $21     ; get number of tracks
            cmp      trkmax          ; check with max of track numbers
            bne      notequ         ; branch if not equal
            lda      sisblk + $22     ; get number of sectors
            cmp      secmax          ; check with max of sector numbers
            bne      notequ         ; branch if not equal

prwrite     print    '\r\rWriting'
            jsr      filpoin         ; set pointer of memory in sector
            ldy      sector          ; restore sector

dataout     jsr      ptrsec          ; print track & sector number
            lda      tardrv          ; get drive number
            jsr      wrchsec         ; write & check sector
            bcs      error           ; branch on error
            inc      rwpoin + 1      ; next address in data block
            lda      rwpoin + 1      ; get current address
            cmp      #$a0            ; end of data block
            beq      2.f             ;
            cpy      secmax          ; all sectors scanned
            bne      1.f             ; branch if not
            ldy      #0              ; set first sector
            inx                  ; next track
            stx      track
            cpx      trkmax          ; all tracks/sectors copied ?
            beq      progend         ; branch if so

```

```

1      iny
      bne    dataout      ; always

2      cpy    secmax      ; all sectors scanned
      bne    3.f          ;
      ldy    #0           ; set first sector
      inx
      stx    track       ; next track

3      jmp    readsrc     ; continue

progend      poscur    1,10
              print    'Copy completed.'
exit        poscur    1,16
              rts        ; program exit

error       poscur    1,15
              sec
              jmp    ermes ; print an error

;*****
;
; Subroutines
;
;*****
prtrsec     sty    sector ; save sector number
              tya
              poscur  28,14 ; posit cursor for sector number
              jsr    hexout ; and print it
              poscur  16,14 ; posit cursor for track number
              ldy    sector ; get sector number
              ldx    track ; get track number
              txa
              jmp    hexout ; and print it

filpoir     lda    #datablk&255 ; set read/write pointer
              sta    rwpoin
              lda    #datablk>8
              sta    rwpoin+1
              rts

              end    dcopy

```

Ook Gij, Brutus...

Zin en onzin over de sociale aspecten van het toenemend computergebruik in de huishoudelijke kring.

Welwel, wat een subtitel zeg. Als zich daar geen literair hoogstaand promotieverslag achter kan ver-
schuilen, dan weet ik het ook niet meer. Weest
echter niet bevreesd; dit artikelje zal allesbehalve
een zwaar te verteren semie-sociale verhandeling
worden. Lichtvoetig ga ik proberen u het komende
kwartier de nachtmerrie van de modernste polygame
samenlevingsvorm te schilderen. Polygaam ja: de
driehoeksverhouding tussen u, uw levenspartner
(m/v) en uw bitverslinder.

Bent u wellicht nog een verstokte vrijgezel? Denkt u
dus dat de inhoud van dit artikel volkomen onbe-
langrijk is voor u? "Voorkomen is beter dan gene-
zen", zegt dan de gemiddelde medicus. En adviseert
u vervolgens dringend het artikel te lezen voor het
geval dát...

In mijn omgeving heb ik de meest extreme oplossing
van het menselijke time-slicing probleem een aantal
malen van dichtbij kunnen bestuderen. Eén vrouw-
lijk slachtoffer dat zich met recht een "computer-
weduwe" kon noemen vertelde mij eens dat ze zich
die ochtend bij het wakker worden verbaasd afvroeg
of hij nog steeds of alwéér achter zijn computer zat.
Hun verhouding heeft niet lang meer geduurd; vol-

gens hem had de PC de oudste rechten en dus kon
zij opstappen...

Van veel mensen hoor ik regelmatig soortgelijke ver-
halen. Meestal met een wat minder rigoreuze afloop;
maar de wrijvingen die in huishoudelijke kring gein-
troduceerd worden bij de intrede van een computer
zijn vaak wel degelijk aanwezig.

Een fraaie oplossing kwam ik tegen in de echomail
deze week. Een bepaald programma werd niet als
"Public Domain" of "Shareware" gereleased maar
als "dinnerware". De auteur van het pakket ver-
wacht niet anders van zijn/haar gebruikers dan dat
ze hun partner tenminste één maal mee uit eten ne-
men. Zo wilde ik voorstellen dat de μ P Kenner on-
der de zelfde noemer uitgebracht werd. Iedere keer
dat de μ P Kenner bij u op de deurmat ligt zet u uw
computer uit. U trekt een avondje uit voor een eer-
ste impressie van de inhoud en voor ieder artikel dat
u bevalt neemt u uw partner mee uit eten.

"Ha! Ik heb geen partner", riep u toen. Mooi, dan
pas ik de regels iets aan... Indien u geen partner
heeft neemt u de auteur van het desbetreffende arti-
kel mee uit eten! Smakelijk eten...

Drs. C. Aesar.

(Advertentie)

KOBRA PC SYSTEMS

Aanbieding: diskettes (NoName)

- 5.25" DD6,95
- 5.25" HD14,95
- 3.5" DD13,95
- 3.5" HD27,95

Alle diskettes in doosjes van 10 stuks.

Motherboards:

- Motherboard AT 6/12 MHz. 399
- Motherboard DX-386 20 Mhz. 1799
- Motherboard DX-386 33 MHz. 2450
- Motherboard 80486 25 MHz. 5200

Printerlinten

- Epson LX800 enz.9,95
- Star LC109,95

100% Garantie (ook op noname diskettes!)

Prijzen incl. BTW

Bestellingen:

Vooruitbetaling op NMB 66.53.97.186
of onder rembours (+ f 6,00)

KOBRA PC SYSTEMS

Postbus 40195

7504 RD Enschede

Tel.: 053 - 776101

Technische ondersteuning tussen 19:00 en 22:00 uur.

De IBM-PC en z'n klonen (Deel 9)

In deze artikelenreeks is het tot dusver vrij tam toegestaan: meer dan verhaaltjes lezen kwam er eigenlijk niet aan te pas. Het onderwerp van al dat geschrijf, de PC zelf, stond al die tijd dan ook een beetje werkloos terzijde. Daar gaan deze keer eens wat verandering in brengen. In de vorige twee delen is uitgelegd wat het BIOS allemaal doet en wat het voor de gebruiker kan betekenen. In de delen daarvoor is uitgelegd hoe de hardware grotendeels in elkaar steekt. We hebben dus nu voldoende handvatten om de machine eens aan het werk te zetten. Het zal geen hoogdravende toestand worden, maar meer een smaakmakertje om de barriere van het 'in de PC rondneuzen' en assembler-knoeien op zo'n ding een beetje aan te moedigen. Voordat we gaan beginnen nog even een klein uitstapje over BIOS wetenswaardigheden, en wel wat er precies gebeurt als er een harde schijf in het systeem zit.

Harde feiten over harde schijven

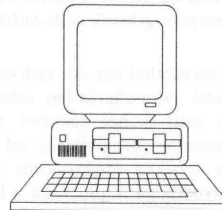
De vorige keer is uitgelegd, dat INT 13h er voor de floppy disks was. Sommigen zullen zich toen hebben afgevraagd, hoe het dan met de harde schijf zit. Zo:

Voor de harde schijf is in de PC/XT een apart BIOS aanwezig, dat op de harde schijf controller zit. Dit BIOS bevindt zich meestal op segment C800, en is bijna altijd 8k groot. Het wordt geactiveerd door de ROMscan van de POST.

Bij de aanroep van de ROMscan doet het harde schijf BIOS meestal het volgende. Eerst gaat het kijken of er een harde schijf aan de controller hangt. Als dat niet het geval is verschijnt er meestal een 1701-error of een tekstmelding. Is de harde schijf er wel, dan gebeurt wat meer: INT 13h wordt omgelegd naar het harde schijf BIOS. De oude INT 13h van het ROM BIOS wordt ook omgelegd, en wel naar INT 40h. Tenslotte wordt ook INT 19h omgelegd om ook van harde schijf te kunnen booten.

Nadat dit gedaan is, blijft INT 13h de BIOS call voor schijfgebruik, zowel voor floppies als voor de harde schijf. Het verschil wordt herkend in het drive nummer, dat in DL wordt overgedragen: bij een harde schijf is het MSbit gezet, dit drive C: wordt aangeduid met 80h (drive A: blijft dus 00h). Het winchester BIOS kijkt dus of het MSbit van het drive nummer is gezet. Is dat zo, dan wordt de eigen code aangeroepen, anders roept het winchester BIOS INT 40h aan om de floppies te activeren.

In de parameters verandert bij harde schijven eigenlijk niets. Het kopnummer (DH) kan nu waarden aannemen tot 15. Bij floppies wordt het tracknum-



mer in CH overgedragen. Dat is bij harde schijven, die immers meer dan 256 cilinders kunnen hebben niet voldoende. Daarom worden de bovenste twee cilinderbits (bits 8 en 9) overgebracht naar de bovenste twee bits van het kopnummer (bits 6 en 7 van DH), zodat op deze manier toch maximaal 1024 cilinders kunnen worden aangesproken.

Tot zo ver het harde schijf-uitstapje. Nu gaan we maar eens in het BIOS poken.

BIOS calls: Eerst wat algemene zaken

Zoals al aangekondigd zal gebruik gemaakt worden van het bij iedere machine aanwezige programma DEBUG. DEBUG is een relatief eenvoudige machinetaalmonitor met ongeveer dezelfde mogelijkheden als MON65 uit DOS65. Daarboven kan DEBUG ook nog op een primitieve manier assembleren, zodat we niet alle opcodes uit ons hoofd hoeven te leren. Dat zou ook niet gaan, want de 8088 heeft er meer dan 1000 als je alle adresseermodes ook gaat meetellen. Helaas is het zo, dat lang niet iedereen een goede gebruiksaanwijzing voor DEBUG bezit. Daarom is in een apart verhaal een beknopte gebruiksaanwijzing van DEBUG opgenomen, en ik hoop dat de redactie die in dit nummer heeft afgedrukt.

Een tweede zaak die men al lezend en typend in de gaten moet houden, dat er hier met het BIOS wordt gewerkt, en niet met DOS. Als er een programma geschreven wordt, dat onder DOS moet draaien, moet dat programma ook met behulp van DOS calls met de machine communiceren. Het is dan niet verstandig om met BIOS calls te werken, die onder andere het nadeel hebben dat redirection niet mogelijk is. Een andere reden om in DOS calls te programmeren is dat Microsoft er tot dusver voor heeft gezorgd, dat de DOS calls opwaarts compatibel zijn gebleven. Dat kan van sommige BIOS calls niet altijd worden gezegd, ofschoon in deze serie alleen gebruikt gemaakt wordt de standaard IBM calls.

Een laatste punt voordat we in DEBUG duiken is dit: het maakt niet uit op wat voor machine er pre-

cies gewerkt wordt: PC, PC/XT, AT, 80386, of andere. De 80XX processor-reeks is ook steeds opwaarts opcode-compatibel, en de hier gebruikte voorbeelden gebruiken alle 8088 instructies. Hetzelfde geldt voor de machinesamenstelling: de hier gebruikte BIOS calls zijn in alle machinetypen beschikbaar, ook als de configuratie eens wat anders is, zoals een EGA- in plaats van een CGA-kaart. De BIOS calls van de PC(/XT) worden namelijk ook in alle andere machines en configuraties ondersteund.

DEBUG

In alle gegeven voorbeelden wordt ervan uitgegaan, dat DEBUG is opgestart zonder verdere argumenten op de commando-regel:

```
DEBUG <Enter>
```

DEBUG heeft een minnetje als prompt. Sommige zeer antieke versies (o.a. MS-DOS 1.25) kunnen een groter-dan-teken (>) als prompt laten zien.

Als DEBUG op deze manier wordt geactiveerd neemt DEBUG aan, dat de gebruiker een programma wil gaan maken van het .COM-type (het andere type is .EXE). Dit houdt in, dat alle segment registers dezelfde waarde krijgen toegewezen, en wel een zodanige waarde dat CS:0 het eerste vrije RAMbyte aanwijst op een paragraaf-grens. (Een paragraaf is een groepje van 16 bytes, waarvan het adres van het eerste byte op nul eindigt). Vanaf CS:0 is er dus een blok van 64 kbyte beschikbaar om code in te schrijven.

De andere registers worden op nul gezet, op twee na. De stack pointer SP krijgt de waarde FFEeh, met andere woorden, de stack zit bovenin datzelfde 64 kbyte blok. De programmateller IP wordt op 0100h gezet, conform de layout die .COM programma altijd heeft.

Een programmaatje wordt als volgt gemaakt:

```
Type eerst: A 100 <enter>
```

Dit start de assembler. Deze zal beginnen op CS:100, het punt waarop de programmateller IP per default staat. DEBUG print vervolgens A 100 op de volgende regels, met de cursor daarachter, en verwacht van de gebruiker dat hij op die plek een mnemonic, eventueel gevolgd door 1 of meer operanden intypt. Binnen DEBUG moet de mnemonic een geldige 8088 mnemonic zijn. De operanden kunnen bestaan uit de gebruikelijke standaard registernamen (AL, AH, AX, BL, BH, BX, enz.), constanten (altijd in hex, geen expressies mogelijk) of de standaard schrijfwijze voor een bepaalde adresseermode. Op

deze laatste regel is 1 uitzondering, en dat is directe geheugenadressering.

Binnen een echte assembler is het duidelijk wat het symbool JOOP voorstelt als de programmeur opschrijft: MOV AL,JOOP. Is aan JOOP een waarde toegekend door een EQU directief, dan wordt de adresseermode automatisch immediate, ofwel JOOP is een constante waarde. Is JOOP een label, dan wordt de adresseermode memory direct en is JOOP een geheugenadres. Omdat DEBUG alleen constanten kent is voor memory direct een afwijkende schrijfwijze ingevoerd: het adres tussen vierkante haken. Als de inhoud van de geheugenlocatie DS:40 in AL geladen moet worden, dan schrijft men in DEBUG: MOV AL,[40]. Zou men MOV AL,40 intypen, dan assembleert DEBUG dit als immediate mode.

Wat nog niet is vermeld, is dat in de DEBUG-assembler achter relatieve sprongen het absolute adres moet worden ingetypt: de assembler rekent het relatieve adres dan uit. Maar nu genoeg gekletst. Actieveer DEBUG eens voor uw eerste programma in assembler op de PC.

Assembleren maar!

Dat eerste programma wordt heel kolossaal: 1 instructie! We gaan aan de PC vragen hoeveel RAM hij heeft. In het vorige deel is verteld, dat het BIOS dat kan meedelen via de call INT 12h. Typ eens in (de output van DEBUG is onderstreept):

```
A 100 <Enter>
```

```
XXXX:0100 INT 12 <Enter>
```

```
XXXX:0102 <Enter>
```

Als het goed is, is de prompt weer terug. Nu gaan we het programma uitvoeren. Om nadat het programma is doorlopen weer in DEBUG terug te keren, moet er achter het programma een brekpunt komen. Dat kunnen we daar neer zetten door met G-commando, waarmee het programma wordt uitgevoerd, het brekpunt adres op te geven. Om het programma uit te voeren typen we daarom in:

```
G 102 <Enter>
```

Als alles goed is, staat er nu een registerdump op het scherm, en bevat AX een getal. Bij een 640k machine is dat 280, bij 512k 200, enzovoort. Dat was het eerste programmaatje alweer: het BIOS is er voor u. Het kan natuurlijk veel uitgebreider.

Stel we willen een mededeling afdrukken, en vervolgens op een toets wachten, een functie die nog weleens opduikt. Dat kan allemaal met BIOS calls. Om de string af te drukken kunnen we gebruik maken van INT 10h, AH=0Eh, Write TTY. Om te kijken of er een toets is ingedrukt, is er de functie INT 16h, AH=01h. Om de toets te lezen en de buffer te legen, kan INT 16h, AH=00h gebruikt worden. Met deze gegevens kunnen we het volgende programmaatje maken, dat ook nog een subroutine bevat:

A 100 <Enter>

```
XXXX:0100 MOV SI,123 <Enter>
XXXX:0103 CALL 115 <Enter>
XXXX:0106 MOV AH,01 <Enter>
XXXX:0108 INT 16 <Enter>
XXXX:010A JZ 106 <Enter>
XXXX:010C MOV AH,00 <Enter>
XXXX:010E INT 16 <Enter>
XXXX:0110 MOV AX,4C00 <Enter>
XXXX:0113 INT 21 <Enter>
XXXX:0115 MOV AL,[SI] <Enter>
XXXX:0117 OR AL,AL <Enter>
XXXX:0119 JZ 122 <Enter>
XXXX:011B INC SI <Enter>
XXXX:011C MOV AH,0E <Enter>
XXXX:011E INT 10 <Enter>
XXXX:0120 JMP 115 <Enter>
XXXX:0122 RET <Enter>
XXXX:0123 <Enter>
E 123 D,A,'Press any key to continue....',d,a,0 <Enter>
ter >
```

Eerst even een beetje uitleg. Op adres 115 begint de subroutine, die een string afdruckt. De string moet eindigen op een nul-byte. Het adres van de string wordt meegegeven in register SI, dat ook voor de indirecte adressering wordt gebruikt. De BIOS-functie write TTY wordt in de subroutine gebruikt om de karakters op het scherm te printen.

Op adres 100 en 103 wordt dus de string afgedrukt. Daarna wordt de BIOS functie INT 16h AH=1 gebruikt om te kijken of er een toets beschikbaar is. Is dat niet zo, dan blijft de Z-vlag gezet, en wordt de functie opnieuw aangeroepen. Is er wel een toets, dan wordt deze met INT 16h, AH=0 uit de toetsenbordbuffer gelezen. In dit programma doen we verder niets met de toetswaarde.

Als laatste in het hoofdprogramma gebruiken we toch een DOS-call: INT 21h, met AH=4Ch. Het is voldoende te weten dat er met deze functie wordt teruggekeerd naar DOS.

Om het programma te kunnen proberen, moet het eerst op schijf worden gezet. Eerst geven we het een naam met:

N ANY.COM <Enter>

Daarna vertellen we aan DEBUG hoe lang het programma is. Door middel van een D(ump) kunt u er achter komen op welk adres het laatste nul-byte van de string staat. Dit zal waarschijnlijk 142 of 143 zijn. Het programma is dan 42h of 43h bytes lang. Die waarde stoppen we in het CX-register met:

R CX <Enter>

CX 0000: 43 <Enter>

Vervolgens schrijven we ANY.COM naar disk met:

W <Enter>

en verlaten we DEBUG met Q <Enter>

We zijn weer bij de vertrouwde DOS prompt. Met ANY <Enter> kunnen we het programma starten. Probeer maar eens of het werkt.

Van het subroutine-tje dat de string beginnend op het adres in SI en eindigend op een nul-byte afdruckt is in deze vorm in vrijwel ieder BIOS te vinden. Hij wordt gebruikt om de diverse meldingen op het scherm te printen in de POST.

Een sector van floppy lezen

Ook dat kunnen we het BIOS laten doen. Het stelt wederom niet zoveel voor: registers laden en het BIOS aanroepen. Om de bootsector (track 0, head 0, sector 1) van drive A: te lezen moet het volgende gebeuren:

A 100 <Enter>

```
XXXX:0100 MOV AX,CS <Enter>
XXXX:0102 MOV ES,AX <Enter>
XXXX:0104 MOV BX,200 <Enter>
XXXX:0107 MOV DL,0 <Enter>
XXXX:0109 MOV DH,0 <Enter>
XXXX:010B MOV CH,0 <Enter>
XXXX:010D MOV CL,1 <Enter>
XXXX:010F MOV AL,1 <Enter>
XXXX:0111 MOV AH,2 <Enter>
XXXX:0113 INT 13 <Enter>
XXXX:0115 <Enter>
```

De instructies op 100 tot 104 zorgen ervoor dat ES:BX naar adres 200 wijzen, in hetzelfde segment als het codesegment. Op die plek kan de gelezen

sector dus worden teruggevonden. Vervolgens wordt drive A: (DL), track 0 (CH), head 0 (DH), sector 1 (CL) gekozen, alsmede het aantal te lezen sectoren (AL). AH=2 is lees sector. Het programmaatje wordt uitgevoerd met:

G 115 <Enter>

Als alles goed gaat, gaat drive A: even lopen, en verschijnt er een registerdump. Belangrijk is de waarde in AX. Als AH nul is, is de operatie geslaagd, en staat er in AL hoeveel sectoren er gelezen werden (1 dus). Als er iets fout is is de carry gezet en bevat AH de errorcode, die bij INT 13h de volgende waarden kan aannemen:

- 01h:Fout commando
- 02h:Bad address mark
- 03h:Write protect violation
- 04h:Record not found
- 08h:DMA fout opgetreden
- 09h:DMA over 64k grens
- 10h:CRC fout
- 20h:Floppy controller defect
- 40h:Seek fout
- 80h: Time out

De laatste waarde treedt bijvoorbeeld op, als er geen schijfje in de drive zit.

Van harde schijf lezen gaat ook: verander op adres 107 MOV DL,0 in MOV DL,80. Dan wordt drive C: gelezen in plaats van drive A:.

Andere BIOS functies

Op deze manier kunnen de meeste BIOS functies wel worden uitgeprobeerd. Dit geldt bijvoorbeeld voor de meeste subfuncties van het video BIOS INT 10h. Analooog kunnen, als tenminste de poorten zijn aangesloten de RS-232 functies (INT 14h) en de parallel printer (INT 17h) functies getest worden. An-

dere functies zijn moeilijker: INT 19h bijvoorbeeld zou het DOS moeten booten. Dat doet het ook wel, maar bij sommige machines wordt eerst het RAM gewist, en daarbij verdwijnt het probeerprogramma ook in het niet, met als gevolg een hangende machine of andere spectaculaire resultaten. Wat blijft gelden: proberen kan altijd. Een reset-knop en ook geduld kan handig blijken.

De interrupts die een tabel aanwijzen hebben minder nut voor de gemiddelde gebruiker. Die worden dan ook meestal alleen intern in het BIOS gebruikt, en in sommige gevallen ook de het DOS. In laatste categorie valt bijvoorbeeld de tabel met floppy parameters aangewezen door INT 1Eh.

BIOS listing

Voor de werkelijk geïnteresseerden in het complete BIOS verhaal is het misschien de moeite waard om eens een BIOS listing te bestuderen. Daarin staan alle mogelijke calls opgesomd, compleet met register layouts voor zowel in- als output registers.

Nu zijn dergelijke listings niet erg dik gezaaid, maar ze bestaan wel. De Technical Reference Manual van de IBM PC bevat bijvoorbeeld een complete listing die redelijk van commentaar is voorzien. Een andere mogelijkheid is de listing van het KGN-BIOS, die sinds kort op floppy disk te verkrijgen is (alleen voor leden!).

De volgende keer...

gaan we eens naar wat wildere machines kijken: de PC/AT, en hoe IBM zich in allerlei bochten moest wringen om die machine opwaarts compatibel te houden met de PC(XT). Tot dan.

Nico de Vries

Ik heb interesse in de KGN en wil

☐ Lid worden van de KGN

☐ Meer informatie over de KGN

Naam: _____

Adres: _____

Postcode en woonplaats: _____

Datum: _____ Handtekening: _____

Dit strookje kunt u ingevuld opsturen aan het secretariaat van: KIM Gebruikersclub Nederland
Davidvosstraat 29,
1063 HV Amsterdam

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

De KIM Gebruikersclub Nederland heeft het stoeien met assemblers hoog in het vaandel staan. Reden genoeg om ook in de shareware hoek eens op zoek te gaan naar assemblers en debuggers.

In de commerciële hoek wordt de wereld van assemblers en debuggers beheerst door MicroSoft en Borland. MASM is het product van MicroSoft; Borland heeft zijn assembler "TASM" gedoopt. Naast deze twee giganten zijn er nog een aantal assemblers die meegeleverd worden met bepaalde compilers. Zo hoort er bij de DeSmet C-compiler een kleine assembler die geheel ter ondersteuning van de C-compiler bedoeld is.

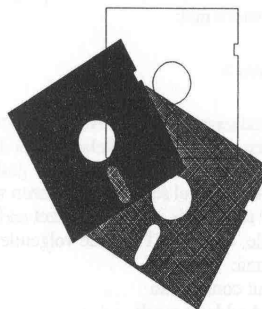
In de shareware wereld is één van de grootste en tevens snelste assemblers voor de PC van het ogenblik te vinden. Het ding luistert naar de nogal fantasievolle naam "A86". Ik had versie 3.21 op de testbank. A86 heeft een nogal uitgebreid verleden. Begonnen werd met een versie voor Xenix, een UNIX kloon voor de PC. De eerste versie voor MS-DOS draagt versienummer 2.10 en aanschouwde het eerste levenslicht in Juni 1986. Sindsdien zijn er heel wat andere versies uitgekomen. De hier geteste release kwam uit in Augustus 1989 en is de meest recente tot op de dag van vandaag.

Over de auteur

A86 is geschreven door Eric Isaacson. Hij werkte al met Intel processoren sinds het bestaan van de 8080, zoals hij zelf schrijft. Tijdens zijn werktijd bij Intel was hij het, samen met een mij onbekende persoon, die de eerste ASM86 assembler schreef. In de documentatie van A86 noemt hij zichzelf een "Independent software consultant", wat dat ook behelzen moge. Daarnaast mag hij zichzelf auteur noemen van een boekwerkje over de 80386 processor en de 80387 number crunching co-processor. Een commercieel ingesteld mannetje blijktbaar, die Eric. Want zijn producten zijn allesbehalve goedkoop en zijn manier van reclame maken komt op mij nogal bot over. Van het bedoelde boekje over de 80386/387 meldt hij in de documentatie van de assembler dat het U.S. \$25,00 moet kosten... compleet met ISBN nummer en zijn adres om het werkje te bestellen als je het niet bij de plaatselijke boekwinkel kan vinden.

De software

Terug naar de software. Daar gaat het per slot van rekening toch om... De assembler en bijbehorende debugger zijn echte shareware. Niet goedkoop trouwens: de assembler kost 52 keiharde dollars. De debugger moet apart betaald worden en kost eveneens



52 dollar... Wil je ze beide hebben, dan kost dat 82 dollartjes, nog altijd goed voor zo'n 160 Hollandse florijnen. Tel daar de overmakingskosten bij en je zit op een gulden of 200. Ter vergelijking: TASM, de commerciële assembler van Borland, gaat over de toonbank voor f 300,00 en dan krijg je Turbo Debug, de full-screen debugger die ook zo leuk samenwerkt met de hogere programmeertalen van Borland, er nog gratis en voor niets bij ook. Na registratie krijg je de nieuwste versie op 5.25" flop opgestuurd. Op deze diskette staat ook nog A86LIB, een library binder, die niet in de vrij verspreidbare package zit. De manual kun je ook nog bij Eric Isaacson bestellen. In goedkope copier-ringband kost het ding U.S. \$15,00...

A86 is beschikbaar op het bulletinboard van de vereniging in drie verschillende pakketten. De debugger is iets kleiner en paste met gemak in twee handelbare pakketten. Het totaalpakket behelst de volgende files:

- A86v321a.Zip - A86 assembler, utils en documentatie
- A86v321b.Zip - A86 documentatie deel 2
- A86v321c.Zip - A86 documentatie deel 3
- D86v321a.Zip - D86 debugger, utils en documentatie
- D86v321b.Zip - D86 documentatie deel 2

De drie A86-files zijn in totaal maar liefst 171678 bytes groot. De debugger doet het wat rustiger aan maar beslaat nog altijd een respectabele 86110 bytes... Op 2400 baud (230 Chars/sec) is dat ca. 21 minuten downloaden. In de avonduren op lokaal tarief dus nog prima te doen, maar interlokaal met een langzamer modem kan dat aardig oplopen. De beroemde flop richting de penningmeester met het vriendelijke verzoek de genoemde software even op te sturen is wellicht een stuk handiger. Doe daar dan wel even een voldoende gefrankeerde enveloppe bij, voorzien van eigen naam en adres...

Gebruik

A86 definieert, zoals iedere zichzelf respecterende assembler schijnt te moeten doen, een ruime commandline. De specificatie van de sourcefile mag de wildcard tekens "*" en "?" bevatten. De gespecificeerde files worden door A86 in alfabetische volgorde geassembleerd. De assembler kan zelf object files, binary images en .COM-style executables genereren. Het formaat wordt bepaald door de extensie van de output file specificatie en kan eventueel nog beïnvloed worden door verschillende command line switches. Onhandig is hierbij wel dat AT86 standaard ".COM" veronderstelt. Logischer zou het zijn om standaard ".BIN" formaten te ondersteunen.

Van idee naar source

Het belangrijkste van de assembler is natuurlijk de generatie van de code uit de source zelf. De assembler ondersteunt de enigszins omslachtige standaard schrijfwijze van procedure declaraties zoals te doen gebruikelijk in MASM. Daarnaast is het in A86 mogelijk "uit de losse pols" te programmeren. De procedure definities met de omslachtige "PROC BYTE PUBLIC Blabla - ENDP" structuur is niet noodzakelijk. Mijn ervaring is echter wel dat juist die gestructureerde opzet de zaak een stuk duidelijker maakt, met name bij het schrijven van stukken software waarvan de source niet meer in een kByte of wat past...

Naast de standaard 8086/8088 instructieset worden ook de instructiesets van de NEC-processoren V20, V30 en Intels 80286, 8087 en 80287 ondersteund. Ondersteuning voor de 80386 is er niet in deze versie. Jammer, daar de auteur waarschijnlijk zelf genoeg weet van de 80386 om ook de extra 386-instructies in te bouwen.

Werken met A86 lijkt in een aantal gevallen meer op programmeren in een processor gerichte hogere programmeertaal dan op werkelijk assembler programmeren. Zo heeft A86 een aantal "fake" instructies die geëxpandeerd worden naar een aantal instructies, de enhancements behelzen o.a. een directe move van waarden naar een segment register; iets dat de processor helemaal niet kan! De instructie "Mov DS,123H" resulteert dan ook in 4 instructies: "Push AX", "Mov AX,123H", "Mov DS,AX" en "Pop AX". Bij het debuggen kan dit voor verrassende effecten zorgen omdat je op sommige plekken je eigen source niet meer terugkent. Daarnaast zijn er ook een aantal instructies niet meer aanwezig... "Lea SI,label" wordt keihard omgezet in "Mov SI,Offset label", hetgeen voor nogal wat verwarring kan zorgen. Door het ontbreken van een ASSUME instructie die aangeeft wat de waarden van bepaalde segmentregisters bij aanvang van het programma hebben kan dit enge gevolgen hebben als er met een

executable gewerkt wordt in de .EXE vorm. Het onderscheid tussen .EXE en .COM style programma's in MS/PC-DOS zit 'm onder andere hierin dat in .COM programma's alle segmenten elkaar overlappen. In een .EXE file is dat meestal niet het geval waardoor een .EXE groter dan de maximale 64 kByte van een .COM formaat kan zijn.

Tenslotte nog een blik op de debugger. Die is de 52 dollar naar mijn idee absoluut niet waard. Het ding is alleen geschikt om origineel in assembly geschreven programma's mee te debuggen, terwijl de algemene trend juist is dat assembly veel in combinatie met hogere programmeertalen gebruikt wordt. Schermafhandeling is ook niet het sterkste punt van de debugger. Het ding schakelt niet zoals te doen gebruikelijk tussen twee schermen heen en weer, waarbij een virtueel scherm door de debugger als "debugscreen" gebruikt wordt. Heeft men twee schermen, dan is dit probleem opgelost. Maar ja, wie heeft er nu twee schermen aan zijn PC hangen?

Documentatie

Vaak laat de documentatie van shareware nog wel eens wat te wensen over. A86 slaat wat dit betreft alles, maar dan wel in positieve zin. De docs van A86 zijn verdeeld in 18 hoofdstukken en beslaan zo'n 360 kByte op mijn harddisk. De debugger is ook goed gedocumenteerd: 11 hoofdstukken en 129 kByte Hard-Disk-ruimte... Om de paar honderd Engelstalige bladzijden door te worstelen is heel wat moed nodig. Gelukkig is een groot deel puur referentie materiaal dat alleen als naslagwerk waarde heeft.

Conclusie

Met A86 haal je een leuk pakket binnen met een originele invalshoek. Het idee om veel voorkomende instructiecombinaties samen te voegen tot een nieuwe instructie is, nadat je er aan gewend bent geraakt, bijzonder aardig. De assembler plaatst standaard zijn error messages in de source, iets waar ik absoluut niet van gecharmeerd ben. Hij gooit ze er tijdens het assembleerproces wel weer uit, maar van m'n source moet 't ding afblijven. De assembler lijkt heel geschikt voor het ietwat verouderde .COM-formaat, voor het "echte programmeerwerk" is A86 niet geschikt.

Een nadeel van de originele ideeën zit 'm in de compatibiliteitshoek. Wie regelmatig sources van anderen bijschaaft weet wat ik bedoel. Het is al moeilijk genoeg om sources te doorzien in het standaard MASM formaat, maar in A86 wordt het echt een probleem...

A86 is een ver uitontwikkelde assembler met compiler-achtige trekjes. Snel is 't ding wel... tot zo'n 1000

regels per seconde zegt de auteur zelf. Voor die snelheid boet je wel sterk in aan compatibiliteit. De prijs is in vergelijking met bijvoorbeeld TASM te hoog. Een shareware product moet beduidend goedkoper zijn om verantwoord te kunnen worden.

van gebruikersvriendelijkheid. Commerciële debuggers zoals Turbo Debug en AT86 hebben een relatief lagere prijs en doen hun werk stukken beter.

Joost Voorhaar

D86 is gewoon te duur. De opzet van de debugger doet ouderwets aan en is niet bepaald een voorbeeld

Besproken produkt	: A86 & D86 assembler/debugger 3.21
Categorie	: Programmeertalen
Registratiekosten A86	: U.S. \$52,00
Registratiekosten D86	: U.S. \$52,00
Kosten A86 & D86	: U.S. \$82,00
Auteur/Leverancier	: Eric Isaacson, Indiana, U.S.A.
Verkrijgbaarheid	: The Ultimate, assembler area.
	A86 is verpakt in 3 files (A86v321a.Zip, A86v321b.Zip en A86v321c.Zip).
	D86 is verpakt in 2 files (D86v321a.Zip en D86v321b.Zip)
	Totaal: A86: ruim 167 kByte. D86: ruim 84 kByte

Minimale systeemeisen:

Standaard PC
 Minimaal 54 kByte RAM vrij. A86 gebruikt tot 256 kByte RAM.
 PC/MS-DOS 2.00 of hoger
 1 360 kByte diskdrive. Een HD is wel een stuk handiger...
 Beeldscherm: CGA, EGA, VGA, MDA, Hercules etc. etc.

Algemene beoordeling

Documentatie : zeer uitvoerig, Engelstalig
 Online help : D86 heeft een beperkte online help mogelijkheid.
 Gebruikersinterface : Keyboard
 Muisondersteuning : Alleen jonge kaas. Niet ondersteund dus.

Positief

Snel, erg snel
 Weinig geheugen nodig
 Multi-file assembly eenvoudig
 Uitgebreide macro processor
 Duidelijke error messages
 Uitgebreide expressie parser die ook gebroken getallen aan kan

Negatief

Niet compatible met MASM en TASM
 Error messages worden standaard in de source opgenomen
 Minder geschikt voor .EXE-style programma's
 Inconsequente regels t.a.v. operands
 A86 assembleert niet alleen maar probeert ook code te optimaliseren
 Niet geschikt als men met TASM of MASM heeft geprogrammeerd of wil gaan programmeren
 Te duur

Eindbeoordeling

Stabiliteit : 7
 Bruikbaarheid : 6
 Totaalresultaat : 6

Stoomcursus DEBUG.COM

In onze club weet vrijwel iedereen wat een assembler is. Ook de kreet machinetaalmonitor is geen vies woord maar synoniem voor een handig programma waarmee je rechtstreeks in het geheugen, de I/O en de CPU kunt spitten. De gemiddelde DOS65 gebruiker springt moeiteloos met MON65 om. Maar op de PC, onder MS/PC-DOS wordt dat plotseling anders. Want er wordt zelden of nooit een handleiding voor DEBUG.COM meegeleverd met de PC, ofschoon DEBUG altijd op de DOS-schijf staat. Het lijkt daarom tijd een beknopte handleiding voor DEBUG te publiceren. Hier is 'ie dan:

Aanroepen

DEBUG kan op twee manieren aangeroepen worden: met en zonder argument. Zonder argument is haast triviaal:

```
DEBUG <Enter>
```

Met argument kan ook:

```
DEBUG programmaam[.EXE] [programma argumenten] <Enter>
```

Het argument is dan een uitvoerbare file van het type .COM of .EXE. De extensie .COM hoeft niet te worden opgegeven; de extensie .EXE echter wel. De programmaam mag gevolgd worden door alle argumenten die het programma normaliter ook zou accepteren. DEBUG laadt het programma in het geheugen zoals DOS dat zou doen. Na het laden wijst in DEBUG CS:IP naar het aanroeppunt van het programma.

Opmerking: DEBUG is niet in staat een file van het type .EXE terug te schrijven naar disk.

Als DEBUG is opgestart, verschijnt de prompt: een minnetje. Achter het minnetje kan een DEBUG-commando worden ingetypt dat minimaal 1 letter is.

DEBUG commando's

DEBUG kent 17 verschillende commando's die allemaal gegeven worden met 1 letter. Het commando wordt vrijwel altijd aangevuld met 1 of meerdere argumenten. De commando's zijn:

A(ssemble)	[adres]
C(ompare)	bereik adres
D(ump)	[bereik]
E(nter)	adres [lijst]
F(ill)	bereik [lijst]
G(o)	[= adres [adres...]]
H(ex)	adres adres

I(nput)	getal
L(oad)	[adres [drive sector1 sector2]]
M(ove)	bereik adres
N(ame)	filespec
O(utput)	getal byte
Q(uit)	
R(egister)	[naam]
S(earch)	bereik lijst
T(race)	[= adres] [getal]
U(nassemble)	[bereik]
W(rite)	[adres [drive sector1 sector2]]

Delen tussen teksthaken ([]) zijn optioneel, delen zonder teksthaken zijn verplicht op te geven parameters.

Verder zijn:

Adres:

1. een offset. In dit geval wordt het default segment gebruikt.
2. een segmentregisternaam of hex getal, gevolgd door een dubbele punt en een offset.

Bereik:

1. twee offsets, gescheiden door een komma of een spatie, of:
2. een offset, gevolgd door een spatie/komma en Lgetal, waarbij getal het aantal herhalingen van het commando is, of:
3. een offset, waarbij per default L80 wordt aangenomen.

Bij twee offsets liggen de offsets beide in hetzelfde segment. Het segment mag bij de (eerste) offset gespecificeerd worden als een hex getal of als een segmentregisternaam, gevolgd door een dubbele punt. Bij het weglaten van het segment wordt het default segment gebruikt. De eerste offset moet altijd lager zijn dan de tweede

Byte:

1. 1- of 2-cijferig hex getal.

Drive:

1. 1-cijferig hex getal. 0 = A:, 1 = B:, 2 = C: enzovoort.

Filespec:

1. Een geldige MS/PC-DOS filespecificatie, die kan bestaan uit minimaal een filenaam, naar inzicht uitgebreid met een extensie, drievenaam en/of pad.

Getal:

1. 1- tot 4 cijferig hex getal dat een poortadres aangeeft of hoeveel maal een commando herhaald moet worden.

Lijst:

1. Een lijst van bytes, gescheiden door spaties of komma's.
2. Een ASCII string, omgeven door enkele of dubbele quotes.

Een lijst is altijd het laatste argument op een commandoregel.

Sector:

1. 1- tot 3-cijferig hex getal dat een absolute disk sector aangeeft.
2. 1- tot 3-cijferig getal dat aangeeft hoeveel sectoren er geschreven/gelezen moeten worden.

Het default segment is DS voor alle commando's die op het geheugen betrekking hebben, behalve G, L, T, U en W. Deze commando's hebben in principe betrekking op code en hebben daarom CS als default segment.

Assemble commando

Met dit commando wordt een eenvoudige assembler geactiveerd. De assembler accepteert dezelfde syntax als de disassembler genereert. Daar waar een relatief adres zou moeten staan, mag een absoluut adres worden opgegeven. De assembler rekent dat het relatieve adres uit.

Na het intikken van het commando, toont de assembler het opgegeven adres. Men kan nu een mnemonic, gevolgd door de vereiste operand(en) intypen. Wordt op Enter gedrukt, dan wordt de instructie geassembleerd en de gegenereerde object in het geheugen gezet. Vervolgens wordt het volgende opcode adres uitgerekend, en aan de gebruiker getoond, die dan weer een instructie kan intypen, enzovoort. Dit proces wordt voorgezet totdat de gebruiker geen instructie invoert, maar alleen een Enter. Dan verschijnt de prompt weer.

Voorbeeld:

A 100 <Enter>

XXXX:0100 JMP 105 <Enter>

Opmerking: XXXX is de huidige CS.

XXXX:0102 MOV AX,1234 <Enter>

XXXX:0105 CLD <Enter>

XXXX:0106 <Enter>

Compare commando

Dit commando vergelijkt een blok geheugen met een ander blok geheugen. Deze blokken mogen in verschillende segmenten liggen. Het default segment is

DS. Verschillen worden gemeld, waarbij bron en bestemmingsoffsets en -segmenten, alsmede de data van beide getoond worden.

Voorbeelden:

C 100,1FF,3000 <Enter>

C 100,L100,3000 <Enter> (Doet hetzelfde!)

C CS:1000,1234,034D:786 <Enter>

Dump commando

Dit commando produceert een hexdump van een blok geheugen op het scherm, compleet met ASCII presentatie. Het default segment is DS. De regels worden zo geformatteerd, dat ze altijd op een 16-byte grens beginnen. Voorbeelden:

D <Enter> (Dumpt 80h bytes vanaf DS:100, nogmaals D <Enter> dump de volgende 80h)

D CS:E000,FFFF <Enter>

Enter commando

Dit command laat toe, het geheugen byte voor byte met de hand te vullen met data. Als prompt verschijnt de huidige geheugeninhoud. Op iedere 8-byte grens wordt op een nieuwe regel begonnen. Als de huidige waarde niet veranderd moet worden, kan naar het volgende adres gesprongen worden met <Spatie>. Men kan naar het vorige adres springen met <->. Het display begint dan op een nieuwe regel. Met <Enter> wordt het commando weer verlaten.

In plaats van losse bytes kan ook een string worden opgegeven. In dit geval wordt het commando na het in het geheugen laden van de string onmiddellijk weer verlaten. Als de string quotes moet bevatten de string met dubbele quotes omringen, en andersom.

Het default segment is DS.

Voorbeelden:

E 100 <Enter>

XXXX:0100 EB. (display van debug)

E CS:E000,'Dit is een string.' <Enter>

E 1234,'Deze string bevat 'zogenaamde' quotes.'

E 5000,'En deze "dubbele quotes".'

Fill commando

Vult een blok geheugen met een byte, een reeks bytes of een string. Het default segment is DS.

Voorbeelden:

F 100,FFF,FF <Enter>
 F CS:1000, 23FF, 4E, 69, 63, 6F, 20, 72, 20 <Enter>
 F 5678:0,0FFF,'Alweer een string.' <Enter>

Go commando

Met dit commando kan een stuk code in het geheugen worden uitgevoerd. Alleen G <Enter> start de code die staat op adres CS:IP. G adres <Enter> doet hetzelfde, maar op adres aangekomen wordt de uitvoering van de code beëindigd en verschijnt de prompt weer. G=adres voert de code uit die staat op absoluut adres adres. G=adres adres doet hetzelfde, met wederom een brekpunt op het tweede adres.

Er kunnen tot tien brekpunten worden opgegeven. Het default segment is CS.

Voorbeelden:

G <Enter>
 G = C800:5 <Enter>

Hex commando

In hex rekenen is meestal niet iemands sterkste punt: daarvoor is het Hex commando er dan ook. Het leest twee hexadecimale argumenten, en laat op de volgende regel de som van die twee en het verschil zien.

Voorbeeld:

H 4363 87AE <Enter>

Input commando

Met dit commando kan een input poort gelezen worden. Het heeft het te lezen adres als argument. De poort is altijd een 8-bit poort. (AT's en andere 80286 machines kennen ook 16-bit I/O-poorten).

Voorbeeld:

I 61 <Enter>

Load commando

Met dit commando kan een file of een blok willekeurige (aaneensluitende) sectoren van schijf in het geheugen geladen worden. Als er een file wordt geladen, moet eerst met het N-commando een naam zijn opgegeven, en moet de opgave van het sectornummer en het aantal sectoren worden weggelaten. Bij het laden van een nader opgegeven aantal secto-

ren wordt de filenaam niet gebruikt. Na het laden geeft de inhoud van de registers BX:CX het aantal geladen bytes aan.

Wordt geen laadadres opgegeven, dan wordt de eerste sector op CS:100 geladen.

WAARSCHUWING: direct sectoren laden wordt via het BIOS gedaan en passeert het DOS filesysteem volledig. Weet wat u doet! De sectoren zijn eenvoudig doorlopend genummerd: Track 0 Head 0 Sector 1 is sector 1. Bij 9 sectoren per track is Track 0 Head 1 Sector 1 dus sector 10, en Track 1 Head 0 Sector 1 sector 19. Gebruik van PCTOOLS of Norton Utilities is duidelijker en veiliger.

Voorbeelden:

L <Enter> ;laadt de file N op CS:100

L DS:8000 <Enter> ;laadt de file N op DS:8000
 L 648A:6080 2 F 6D <Enter>

Het laatste voorbeeld laadt 6D sectoren van drive C op adres 648A:6080, te beginnen met sector F.

Move commando

Met dit commando wordt een blok geheugen verplaatst, met uiteraard die verstande, dat het source blok niet gewist wordt. Voor voorbeelden, zie het C-commando (wel even M voor C denken natuurlijk).

Name commando

Met dit commando wordt een filenaam opgegeven voor de L- en W-commando's, als die tenminste een file moeten laden. De filenaam moet voldoen aan de normale MS-DOS conventies, en mag zowel een drive letter met colon als een extensie hebben. Directory informatie is niet toegestaan. Eenmaal opgegeven, blijft de filenaam dezelfde totdat er een nieuwe wordt gespecificeerd.

Wordt na de eerste naam nog meer informatie getypt, dat wordt ook dit opgeslagen, en behandeld alsof dit als argument aan het programma NAAM1 zou zijn meegegeven.

Voorbeeld:

N FILENAAM.EXT <Enter>

Output commando

Dit het omgekeerde van het I-commando: schrijf een waarde in een output poort.

Ook nu alleen voor 8-bit poorten. Voorbeeld:

O 461 80 <Enter>

Quit commando

Het meest spectaculaire commando: ga terug naar DOS. Voorbeeld:

Q <Enter>

Register commando

Hiermee kunnen de CPU registerinhouden getoond en/of veranderd worden. R zonder argumenten toont alle registers en ook de vlaggen. Wil men een register of de valgen een nieuwe inhoud geven, dan moet het desbetreffende register worden opgegeven. DEBUG toont dan eerst de huidige waarde, en vraagt vervolgens om een nieuwe. Geldige register-namen zijn: AX, BX, CX, DX, BP, SI, DI, SP, CS, DS, ES, SS, IP, PC en F. PC en IP zijn beide de programmateller, F stelt de vlaggen voor.

Wordt F opgegeven, dan worden de vlaggen als lettercombinaties gedumpt, en om te veranderen moeten ook lettercombinaties worden ingevoerd:

Vlag	Gezet	Clear
Overflow	OV	NV
Zero	ZR	NZ
Direction	DN	UP
Aux carry	AC	NA
Interrupt	EI	DI
Parity	PE	PO
Sign	NG	PL
Carry	CY	NC

Voorbeelden:

R <Enter>
R CS <Enter>
RF <Enter>

Search commando

Zoekt in het geheugen naar reeksen bytes of een string.

Voorbeelden:

S CS:100,17FF,90,EB,00 <Enter>
S 1234:9876,AAAA,'Dit is de zoekstring' <Enter>

Trace commando

Met dit command kan er single step door een programma worden gewandeld, waarbij na iedere instructie de registers worden gedumpt zoals het R-commando zou doen. Daarna kan men weer een

DEBUG commando intypen (bijvoorbeeld T <Enter>). Het programma wordt gestart op CS:IP.

Wordt ook = adres opgegeven dat wordt op CS:IP gestart, waarna op adres aangekomen er wordt overgegaan op single step.

Geeft men alleen een getal op, dan wordt er gedurende getal instructies getraced, waarna weer een DEBUG commando kan worden ingegeven.

Voorbeelden:

T <Enter> ;doe 1 instructie
T = CS:200 <Enter> ;ga op IP = 200 over op single step
T 100 <Enter> ;start op CS:IP en trace 100h instructies

Unassemble commando

Disassembleert 8088 machinecode, ook in een 80286 machine! De mogelijkheden zijn dezelfde als met het D-commando, met dien verstande dat er default 16 instructie gedisassembleerd worden, en niet 80h. De output is helaas niet helemaal compatible met MASM. Het default segment is CS, de default offset IP.

Voorbeelden:

U <Enter>
U F000:E000,FFFF <Enter>

Write commando

Het omgekeerde van het L-commando: schrijft een file of een setje sectoren op schijf. Voor voorbeelden en waarschuwing, zie L-commando.

Foutmeldingen

DEBUG geeft in een beperkt aantal gevallen een foutmelding. De eerste soort is een fout in de opgegeven parameters van een commando. DEBUG zal trachten aan te geven waar het misging, door een pijltje naar boven af te drukken op de volgende regel, gevolgd door het woord 'Error'.

De tweede soort foutmelding is er een met twee letters:

BF = Bad Flag.

Er is een lettercombinatie voor een vlag gebruikt die niet geldig is.

BP = teveel BreakPoints.

Er zijn meer dan 10 breekpunten gedefinieerd.

BR = Bad Register

Er is een lettercombinatie voor een register-naam gebruikt die niet geldig is.

DF = Double Flag

In de vlaggenlijst komt een vlag tweemaal met tegengestelde waarde voor.

Samenvatting

DEBUG is een simpele rechttoe, rechtaan ontluizer, ongeveer van de klasse van MON65. Ik vind dat er twee belangrijke gebreken aan zitten: In een 80286 machine wordt je nog steeds vergast op 8088 disas-

semblies en de disassembler geeft een output die niet overkomt met MASM, de standaard assembler. Beide gebreken worden grotendeels teniet gedaan door SYMDEB, de symbolische debugger die bij MASM hoort.

Wat nog niet verteld is, is dat de vertrouwde commando-regel editor met F1, F2, F3, F4 en F5 ook binnen DEBUG werkt. En dat was alles dus.

Nico de Vries

Listing van KGN XT-BIOS toch beschikbaar op floppy.

Eigenlijk vanaf het begin van de verkoop van het KGN-BIOS was er de vraag: is er een listing van het BIOS op floppy? Het antwoord tot dusver was: nee, alleen op papier. Dat was niet zonder reden: met de listing op floppy is het relatief weinig werk om deze terug te werken naar assembler-source, en van daaruit dus een andere BIOS te creëren. En voor je het weet, heb je een wildgroei van allerlei BIOSsen, en dat is onmogelijk goed te ondersteunen.

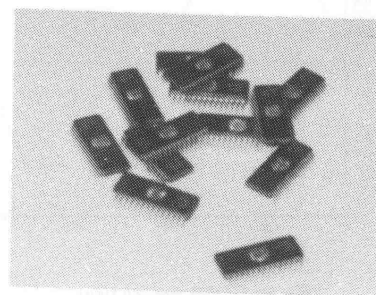
Toch bleef de vraag binnenkomen: listing op floppy alstublieft. Die werd trouwens wel steeds gesteld door mensen die wisten wat ze wilden. Daarom toch een verandering in de politiek. Vanaf vandaag kunt u de listing gewoon bij de penningmeester bestellen door f 20,00 over te maken op de girorekening van de club onder vermelding van: KGN-BIOS listing.

De volgende spelregels gelden hierbij: de versterkte listing is niet van de laatste versie (thans 3.10) doch van drie versies daarvoor (dus nu 3.07). De listing staat op een 360k floppy (720k kan op verzoek), en is in schone ASCII vorm, direct gereed om op een 132 koloms printer te worden afgedrukt. De listing bevat TABs, en gaat ervan uit, dat de TAB-stops om de 8 karakters staan. Als de listing wordt afgedrukt, krijgt u een verhaal van ongeveer 125 pagina's lang, inclusief de cross-reference. We hopen op deze manier weer in een behoefte te voorzien.

Een ieder blijft natuurlijk zelf verantwoordelijk voor eventueel zelf bedachte veranderingen. Met klem vragen we verder: gewijzigde BIOSsen alstublieft NIET, herhaal NIET verspreiden. Ontdekt u een bug (die kan inmiddels al verholpen zijn, de release versie is immers drie revisies verder) of groeit er een idee voor een zinvolle uitbreiding, dan horen we dat natuurlijk graag. Uitbreidingen zijn trouwens moeilijk, want het BIOS zit werkelijk tot de nok toe vol, iets wat de listingbestellers spoedig zullen ontdekken.

De listing op papier blijft gewoon beschikbaar en kost f 25,00

Nico de Vries



Methoden en technieken voor datacommunicatie (deel 3).

Inleiding

In het vorige deel hebben we een klein uitstapje gemaakt naar het telefoonnet. In dit deel van de serie over datacommunicatie gaan we het hebben over modems die over het algemeen weer aangesloten worden op dat telefoonnet. Het artikel volgt voor het grootste deel mijn voordracht op de clubbijeenkomst in Almere over dit onderwerp.

Zoals we in de vorige afleveringen gezien hebben, is er voor de amateur maar één mogelijkheid om een computer over grotere afstanden met andere computers te laten communiceren: met behulp van een modem via het telefoonnet. Hiervoor is een vorm van MODulatie en DEModulatie nodig om de bits door de voor spraak ontworpen telefoonlijn te kunnen persen. In deze aflevering wordt beschreven hoe dit proces in zijn werk gaat en wat er dus in het zwarte of zilverkleurige doosje gebeurt. Verder worden er ook enkele veel gebruikte standaards uitgelegd zodat u, als u een modem aan wilt schaffen, kunt bepalen of u een V21 of een V32 modem nodig heeft.

Een klein beetje wis- en natuurkunde

Als we over de werking van modems gaan praten, dan komen er een aantal begrippen uit de wis- en natuurkunde naar voren die mogelijk niet (meer) tot de parate kennis van iedereen behoren.

In figuur 1 is een sinusfunctie getekend. Hierbij is de waarde van de sinus als functie van de hoek (in graden) uitgezet. In de natuurkunde is de sinusfunctie een functie die de verhouding tussen de lengte van twee zijden in een rechthoekige driehoek aangeeft.

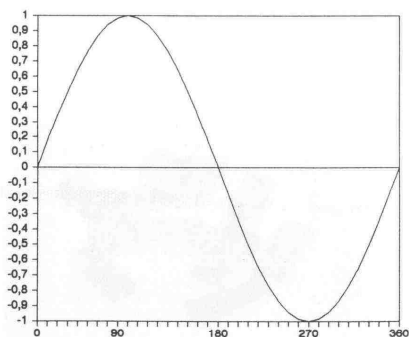


Fig. 58: Grafiek van een sinus

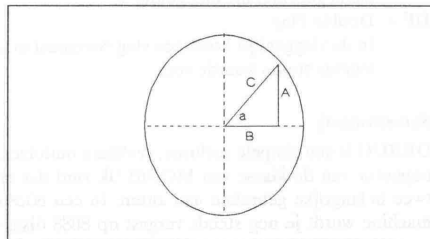


Fig. 59: Bepalen van de sinus van een hoek

In figuur 2 is een dergelijke driehoek getekend. De sinus van hoek a [dus $\sin(a)$] is in dit geval A/C . Bij een hoek van 30 graden is de sinus 0.5, dit wil dus zeggen dat de schuine zijde C twee keer zo lang is als de rechthoekszijde A . Als we hoek a kleiner maken, dan wordt automatisch zijde A ook kleiner. Als hoek a 0 graden is, dan verdwijnt ook zijde A en is de sinus automatisch ook 0. Bij 90 graden heeft de sinus de waarde 1.

Een broertje van de sinus is de cosinus. Deze functie geeft de verhouding tussen de zijden B en C . Deze functie begint bij 1, neemt af tot 0 bij 90 graden en -1 bij 180 graden en neemt daarna weer toe tot 1. Het is dus net of de cosinusfunctie een beetje (90 graden) voorloopt op de sinusfunctie.

In de electronica en bij modems komt de sinusfunctie voor in de vorm van een sinusvormige wisselspanning. Bij een dergelijke spanning is de spanning, als functie van de tijd, een sinusvormige figuur. De spanning neemt dus vanaf 0 volt toe tot zijn maximum, neemt vervolgens af tot 0 volt en weer tot de negatieve waarde van zijn maximum en vervolgens weer naar 0 volt. Bij het lichtnet van 220 volt is de maximum spanning die bereikt wordt 310 volt en wordt de sinus 50 keer per seconde doorlopen.

Bij wisselspanningen wordt meestal niet de maximale waarde van de spanning aangegeven maar de zogenaamde effectieve waarde. Deze effectieve waarde is een soort gemiddelde waarde voor de spanning. Bij een sinusvormige spanning is de maximale waarde wortel $2 = 1.4142$ keer zo groot. Als niet direct duidelijk is dat de effectieve waarde gebruikt wordt, dan wordt dit vaak aangegeven met de afkorting RMS van Root Mean Square; de wijze waarop het gemiddelde bepaald wordt. Als u een (sinusvormige) wisselspanning met een voltmeter meet, dan wordt ook deze RMS-waarde door de voltmeter aangegeven. In de voltmeter wordt meestal wel de maximale waarde bepaald maar wordt deze door de bovengenoemde factor gedeeld.

Als de spanning niet sinusvormig verloopt, dan is de factor tussen RMS-waarde en maximale spanning ook anders. In dat geval zullen de meeste (goedkope) voltmeters een verkeerde spanning aangegeven.

Aan de hand van figuur 1 worden enige begrippen beschreven.

Amplitude

Dit is de maximale spanning die bereikt wordt. Dit is dus de waarde in volts op de top van de sinus. Aan de andere kant, in het dal heeft de spanning de negatieve waarde van de amplitude.

Periode

Dit is de tijd (in seconden) die nodig is om de spanning een volledige sinus te laten doorlopen. De periode wordt aangegeven in seconden.

Frequentie

Dit is het aantal sinussen dat in een seconde gaat. De frequentie wordt aangegeven in Hertz (Hz).

Fase

Zoals beschreven is, wordt de sinus gebruikt bij berekeningen met hocken. Een hock wordt aangegeven in graden. De fase is het deel van de sinus die al afgelegd is. Een hele sinus is dan 360 graden, alleen de eerste berg is 180 graden etc.

Voor een sinusvormige wisselspanning kan de volgende formule opgesteld worden:

$$V(t) = A \sin(t * f * 360)$$

met:

$V(t)$ is de spanning als functie van de tijd t

A is de amplitude

f is de frequentie

$t * f * 360$ is de fase en omdat een sinus bij 360 graden weer opnieuw begint wordt de fase meestal modulo 360 genomen.

Zoals als is aangegeven, is de cosinus een broertje van de sinus. De cosinus en de sinus hebben een faseverschil van 90 graden ten opzichte van elkaar dus:

$$V(t) = A \cos(t * f * 360 + 90)$$

Seriële communicatie

Als we communiceren door middel van een modem, dan gaat het altijd om een seriële communicatie. Verder gaat het bij de personal en hobby computers vrijwel zonder uitzondering om een asynchrone communicatie. Hierbij nog even een kleine samenvatting.

Bij seriële communicatie worden de bits achter elkaar overgestuurd over bijvoorbeeld twee lijnen. Hierbij wordt van een byte het minst significante bit (bit 0) als eerste over de lijn gestuurd. Tegenwoordig wordt meestal gewerkt met 8 bits, zonder pari-

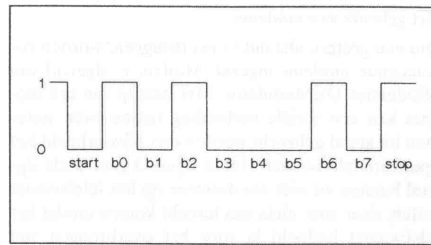


Fig. 60: Het bitpatroon van "E" over de seriële

teit. Dat wil zeggen dat de informatie in groepjes van 8 bits (een byte dus) overgestuurd wordt. Verder wordt er geen pariteitsbit meegestuurd om te controleren of de informatie goed overkomt. Bij een asynchrone communicatie wordt in de rusttoestand van de lijn continue een bit één (mark) overgestuurd. Als er nu een byte overgestuurd moet worden, dan wordt er gedurende één bittijd een nul (space) op de lijn gezet. Vervolgens worden de 8 databits overgestuurd en tenslotte minimaal één stopbit die ook de waarde één heeft.

Hoe lang een bit duurt, de bittijd, is ingesteld op de zender en de ontvanger en wordt aangegeven door de bitrate die meestal ten onrechte de baudrate genoemd wordt. De bitrate geeft aan hoeveel bits er per seconde overgestuurd kunnen worden. In figuur 3 is getekend hoe de letter 'E' over de lijn gestuurd wordt.

De meeste seriële computeraansluitingen zijn volgens de zogenaamde RS 232 C standaard. Hierbij wordt een 0 als +12 Volt en een 1 als -12 Volt over de lijn gestuurd. De volledige RS 232 C standaard is in figuur 3 getekend. Behalve een aansluiting voor zenden (TXD) en ontvangen (RXD) zijn er een aantal stuursignalen. Met behulp van deze signalen kan de computer of een aangesloten modem controleren of alles in orde is.

Als we een communicatielijntussen twee systemen hebben, dan kunnen we in principe te maken hebben met de volgende situaties:

Simplex

Er is slechts communicatie in één richting mogelijk

Half duplex

Er is communicatie in twee richtingen mogelijk doch nooit tegelijk

Full duplex

Er is, volledig onafhankelijk van elkaar, communicatie in twee richtingen mogelijk

Het gebruik van modems

Om een grotere afstand te overbruggen, worden zogenaamde modems ingezet. Modem is afgeleid van MODulator DEModulator. Met behulp van een modem kan een seriële verbinding tussen twee systemen tot stand gebracht worden met bijvoorbeeld het openbare telefoonnet. Het in figuur 3 getekende signaal kunnen we niet zondermeer op het telefoonnet zetten, daar zou niets van terecht komen omdat het telefoonnet bedoeld is voor het overbrengen van spraak en niet voor het overbrengen van dergelijke spanningen. Wat dus door een modem gedaan wordt, is het omzetten van het spanningsverloop in een signaal dat via het telefoonnet overgestuurd kan worden.

Via het telefoonnet kunnen geluiden van verschillende toonhoogten overgebracht worden. Deze geluiden worden opgevangen door een microfoon en omgezet in een wisselspanning. De frequentie van deze wisselspanning is een maat voor de toonhoogte. Bij de ontvanger wordt de wisselspanning aan een soort luidspreker aangeboden die de wisselspanning weer omzet in een geluidstrilling. De bandbreedte van het telefoonnet geeft aan welke frequenties overgebracht kunnen worden. Bij het openbare telefoonnet loopt deze bandbreedte van zo'n 300 Hz tot ongeveer 3400 Hz.

Vanwege PTT regels en de wens zo weinig mogelijk aan het telefoonnet te "sleutelen" werden modems een paar jaar geleden niet altijd direct op het telefoonnet aangesloten maar via een acoustische koppeling. De telefoonhoorn werd op het modem gelegd zodat via een microfoon en een luidspreker het modem en de telefoon geluiden uit konden wisselen. De tegenwoordige modems worden vrijwel zonder uitzondering rechtstreeks aan het telefoonnet gekoppeld. Voor de werking maakt dat in principe niet uit alleen is het nogal omslachtig een elektrisch signaal eerst om te zetten in een geluidssignaal en dat op te vangen door een microfoon die het geluidssignaal weer omzet naar een elektrisch signaal.

Tegenwoordig wordt een modem vrijwel zonder uitzondering direct op het telefoonnet aangesloten. Dit mag echter alleen als het modem door de PTT goedgekeurd is. Bijna alle modems die momenteel in de winkel verkocht worden hebben die PTT goedkeuring, herkenbaar aan een stickertje met een goed-

keuringsnummer. Niet goedgekeurde modems mogen niet op het telefoonnet aangesloten worden, netzomin als niet goedgekeurde telefoontoestellen. In de praktijk blijken deze apparaten meestal net zo goed te werken als de goedgekeurde exemplaren.

Modulatietechnieken

Zoals al is aangegeven, kan het telefoonnet niet zo maar digitale signalen overbrengen. Op de één of andere manier moet een digitaal signaal omgezet worden in een geluidssignaal. De meest gemakkelijke manier hierbij is bijvoorbeeld de afspraak dat geen signaal betekent dat er een 1 overgestuurd wordt en wel een signaal een 0. Dit is dan een vorm van amplitude modulatie; in de amplitude van het signaal wordt aangegeven of er een 0 of een 1 bedoeld wordt. Voor het geluidssignaal kan dan een wisselspanning met een geschikte frequentie gebruikt worden. Bij mijn weten wordt deze vorm van modulatie niet (meer) gebruikt; waarschijnlijk is ze ook nooit voor computercommunicatie gebruikt. Deze aan-uit modulatie wordt ook wel On Off Keying genoemd.

Een tweede mogelijkheid, die wel veel gebruikt wordt, is een zogenaamde frequentie-modulatie. Afgesproken is dat voor de 0 een bepaalde frequentie gebruikt wordt en voor de 1 een andere frequentie. Als er dus na een bit 0 een bit 1 overgestuurd wordt, vindt er een frequentiesprong plaats, vandaar de naam Frequency Shift Keying. Het aantal frequentiesprongen dat per seconde gemaakt kan worden is de zogenaamde baudrate en aangezien bij elke bitwijziging de frequentie wijzigt, is dit toch gelijk aan de bitrate.

De maximale bitrate die met behulp van FSK overgestuurd kan worden wordt bepaald door de bandbreedte van de telefoon. Het is namelijk niet zo dat alleen de gebruikte frequenties door de telefoon overgestuurd moeten worden, het moet ook nog mogelijk blijven dat de frequentiesprongen goed gedetecteerd kunnen worden en dat de signalen van zender en ontvanger van elkaar onderscheiden kunnen worden. Hoe vaker de frequentie in een seconde wijzigt, dus hoe hoger de baudrate is, hoe groter de bandbreedte moet zijn. De benodigde bandbreedte voor een kanaal is ongeveer twee maal de baudrate voor dit kanaal.

Voor een full duplex verbinding wordt meestal 300 b.p.s. gebruikt (V21 norm). Hierbij hebben de twee

Vanwege PTT regels en de wens zo weinig mogelijk aan het telefoonnet te "sleutelen" werden modems een paar jaar geleden niet altijd direct op het telefoonnet aangesloten maar via een acoustische koppeling.

partijen elk een frequentie voor het zenden van een 0 en een frequentie voor het zenden van een 1 toegevoegd gekregen. Er wordt gebruik gemaakt van de frequenties 980 Hz (1) en 1180 Hz (0) en van 1650 Hz (1) en 1850 Hz (0). Voor een half duplex verbinding kunnen hogere baudrates gebruikt worden. Nu hoeven de twee partijen de telefoonlijn niet te delen en kan men tot maximaal zo'n 2000 bps gaan.

Voor onder andere Viditel bleek dat er veel meer gegevens in de ene richting gaan dan in de andere. In de richting van de database is een snelheid van zo'n 75 bps meer dan genoeg. De meeste mensen kunnen toch niet sneller tikken dan 100 tekens per minuut en dat zijn nog geen 20 bits per seconde. De andere kant op gaan schermen vol met informatie en daarbij geldt hoe sneller hoe beter. Voor deze toepassing is de V23 norm ontwikkeld waarbij in de ene richting een baudrate van 1200 (dus 120 tekens per seconde) gehanteerd wordt en in de andere richting een snelheid van 75 baud (dus 7,5 tekens per seconde). Voor de snelle lijn worden de frequenties 1300 Hz en 2100 Hz gebruikt en voor de langzame (back channel) de frequenties 390 Hz en 450 Hz.

Met behulp van frequentiemodulatie zijn we bij de bovengenoemde snelheden uitgepraat. Hogere snelheden vragen een grotere bandbreedte en die hebben we nog niet. Aangezien de behoefte bestond toch een hogere snelheid te bereiken, is gezocht naar een andere modulatie-techniek. In plaats van de

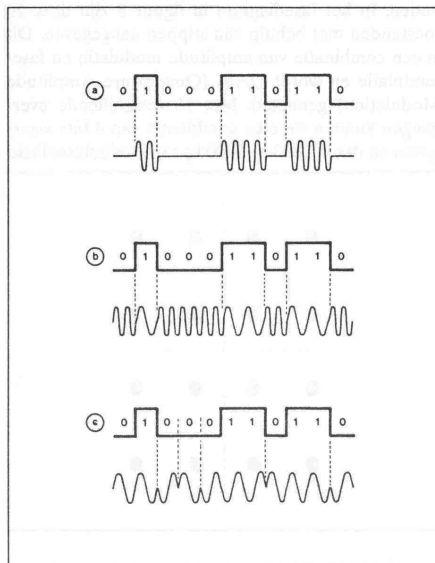


Fig. 61: Drie vormen van modulatie

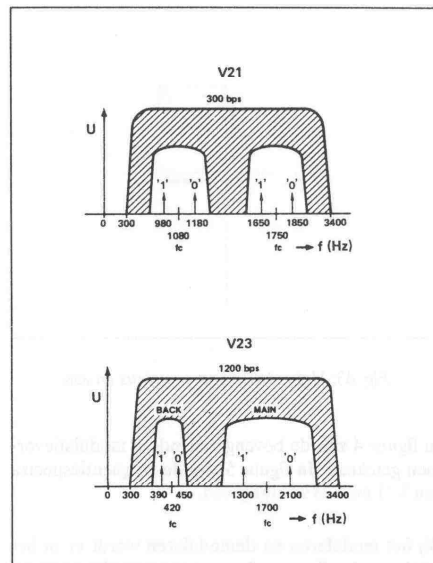


Fig. 62: Frequentiespectra bij V21 en V23

frequentie te variëren, wordt tegenwoordig vaak de fase veranderd. Het aantal keren dat de fase gewijzigd kan worden is nu de baudrate en die is voor full duplex verbindingen 600 baud. Voor half duplex verbindingen is dit op te voeren naar maximaal 2400 baud, eventueel met back channel. Bij full duplex verbindingen wordt gebruik gemaakt van de frequenties 1200 Hz en 2400 Hz.

Om een nog hogere bitrate te krijgen worden de bits twee aan twee samengevoegd. Bij een bit 0 gevolgd door een bit 1, is de fasesprong 0 graden en is er dus geen fasesprong. Bij een bit 0, gevolgd door een bit 0, is de fasesprong 90 graden, bij een 1 gevolgd door een 0 180 graden en bij een 1 gevolgd door een 1 is de fasesprong 270 graden. Dus 600 keer per seconde worden er twee bits doorgegeven waardoor we dus 1200 bits per seconde (full duplex) door kunnen geven. Dit is de V22 norm. Deze vorm van modulatie heeft DPSK (Differential Phase Shift Keying). We hebben nu dus als het ware een modulatie periode (1/baudrate) verdeeld in 4 verschillende fasesprongen en hiermee kunnen we vier verschillende combinaties van twee bit coderen. De snelheid in bits per seconde is dus twee maal de baudrate en bedraagt dus 1200 bps. In de praktijk wordt meestal gesproken over een 1200 baud full duplex modem en verbinding; hoewel de baudrate strikt genomen slechts 600 is. De norm die bij deze modulatievorm hoort is de V22 norm.

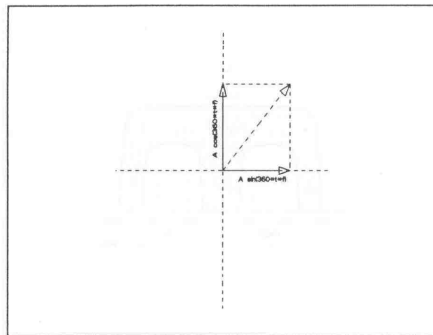


Fig. 63: Het optellen van een sinus en een cosinus

In figuur 4 zijn de bovengenoemde 3 modatievormen getekend. In figuur 5 zijn de frequentiespectra van V21 en V23 weergegeven.

Bij het moduleren en demoduleren wordt er in het modem gebruik gemaakt van twee signalen met dezelfde frequentie die echter 90 graden met elkaar in fase verschillen. Dit is dus een sinusvormig signaal en een cosinusvormig signaal. Als deze signalen met gelijke amplitude bij elkaar opgeteld worden, dan krijgen we een signaal met dezelfde frequentie dat echter in fase 45 graden verschoven is met het sinusvormige signaal.

In figuur 6 is een zogenaamd fasediagram getekend. In een dergelijk plaatje wordt een sinusvormig signaal aangegeven door middel van een pijl. Een ander signaal, met dezelfde frequentie wordt aangegeven met een andere pijl waarbij de hoek tussen de beide pijlen het faseverschil aangeeft. De lengte van een pijl geeft de amplitude van het signaal aan. Twee signalen kunnen op de aangegeven wijze in het fasediagram bij elkaar opgeteld worden en de pijl die dan ontstaat geeft de amplitude van het somsignaal en het faseverschil met de oorspronkelijke signalen weer. De frequentie van het somsignaal is gelijk aan de frequentie van de beide oorspronkelijke signalen.

In figuur 6 is het sinussignaal aangegeven d.m.v. de liggende pijl naar rechts. Het cosinussignaal loopt 90 graden voor op het sinussignaal en wordt aangegeven door de staande pijl naar boven. Tellen we deze signalen bij elkaar op, dan ontstaat de gestippelde pijl, die een grotere lengte heeft en met beide oorspronkelijke pijlen een hoek van 45 graden maakt. Als we het sinussignaal inverteren, dan wordt de pijl naar rechts een pijl naar links. Wat dat voor het somsignaal betekent, kunt u zelf afleiden, de pijl gaat nu naar linksboven in plaats van rechtsboven en

het faseverschil met de oorspronkelijke sinus is nu 135 graden i.p.v. 45 graden.

De in het fasediagram getekende pijlen zijn zogenaamde "vectoren". Een vector is iets met een lengte en een richting waarbij in ons geval de lengte de amplitude en de richting de fase van een signaal aangeeft. Er wordt ook wel eens gezegd dat een wisselspanning afgebeeld kan worden door middel van een draaiende vector. De vectorpijl draait dan in het fasediagram waarbij hij evenveel rondjes per seconde maakt als zijn frequentie is. Aangezien wij echter geïnteresseerd zijn in faseverschillen, is het niet zinvol dit tijdsaspect af te beelden en kunnen we één van de signalen rustig vast langs de horizontale as leggen en als nulpunt gebruiken. Meestal wordt dan het sinusvormige hiervoor gekozen.

Om de bovengenoemde faseverschuivingen van 0, 90, 180 en 270 graden te kunnen genereren en detecteren is het voldoende twee signalen (een sinus en een cosinus) al dan niet geïnverteerd bij elkaar op te tellen. In de elektronica zijn inverteren en optellen eenvoudige bewerkingen waarvoor standaardcomponenten gebruikt kunnen worden. We beginnen dan bijvoorbeeld met een niet geïnverteerde sinus en een niet geïnverteerde cosinus. Willen we nu een faseverdraaiing van 90 graden, dan moeten we de sinus inverteren. Door de sinus en de cosinus te inverteren, krijgen we een fasesprong van 180 graden en het inverteren van alleen de cosinus geeft een fasesprong van 270 graden.

Behalve het inverteren van het signaal zouden we bijvoorbeeld ook de amplitude door 3 kunnen delen. We hebben dan in totaal 16 verschillende mogelijkheden. In het fasediagram in figuur 8 zijn deze 16 toestanden met behulp van stippen aangegeven. Dit is een combinatie van amplitude modulatie en fase-modulatie en wordt QAM (Quadrature Amplitude Modulation) genoemd. Met 16 verschillende overgangen kunnen we elke combinatie van 4 bits weergeven en dus $4 \times 600 = 2400$ bps transmissiesnelheid

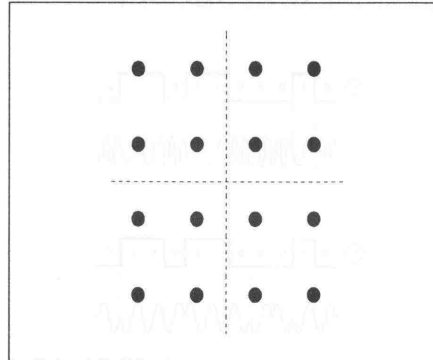


Fig. 64: Fasediagram van de V22bis norm

halen. Dit is de V22bis norm en is tegenwoordig bij

veel amateurs de standaard. Een modem die aan deze standaard voldoet, wordt meestal, ten onrechte, een 2400 baud full duplex modem genoemd.

Als we een viervoudige modulatie toepassen bij een halfduplex verbinding, dan krijgen we dus maximaal $4 * 2400 = 9600$ bps. Deze vorm heeft als norm V29 en wordt gebruikt bij Telefax-apparatuur. De gebruikte frequentie is dan 1700 Hz.

Toekomst

Momenteel is V22bis voor amateurs zo'n beetje "state of the art". Het blijkt echter dat de ontwikkelingen zich in snel tempo opvolgen en dat ook de zogenaamde high speed of ultra high speed modems tot de amateurs door gaan dringen.

Om hogere snelheden bij met name full duplex verbindingen te halen zijn er een aantal technieken ontwikkeld. In de eerste plaats is er een techniek die in de literatuur het "Ping Pong" mechanisme genoemd wordt. Deze techniek is afgeleid van een half duplex V29 verbinding. In het modem zit een hoeveelheid buffergeheugen waarin de informatie die overgestuurd moet worden tijdelijk opgeslagen wordt. De modems werken strikt volgens half duplex waarbij, afhankelijk van de hoeveelheid informatie in het buffergeheugen zender en ontvanger bliksemsnel omgeschakeld kunnen worden. Door middel van extra stuursignalen kunnen de modems elkaar de status van hun buffer en van de lijn laten weten.

Een variant hierop is een systeem waarbij behalve het 9600 bps hoofdkanaal een 300 bps bijkanaal (back channel, vergelijk V23) gedefinieerd is. Afhankelijk van de situatie wordt het hoofdkanaal gegeven aan de richting die de meeste data overbrengt en het bijkanaal aan de andere partij. Ook in dit geval kunnen hoofd- en bijkanaal bliksemsnel omgewisseld worden.

Het nadeel van de bovengenoemde oplossingen is dat er geen echte standaards voor bestaan. De V29 standaard beschrijft een zuivere half duplex verbinding en de bovengenoemde uitbreidingen kennen geen standaard in de door de CITT uitgebrachte V-standaards. Er bestaat wel een gestandaardiseerde 9600 bps full duplex verbinding: De V32 standaard. Modems uit deze categorie komen nu langzaam maar zeker op de markt voor zo'n fl 2500,-. Binnen

de V32 standaard werkt gewerkt met slechts één dragerfrequentie van 1800 Hz. De beide partijen werken dus met dezelfde dragerfrequentie! Hoe kunnen ze dan toch gelijktijdig informatie oversturen en elkaars signaal onderscheiden? Hiervoor wordt gebruik gemaakt van moderne digitale signaal processoren met hoge verwerkingssnelheid. In het modem wordt berekend hoe de telefoonlijn reageert op het aangeboden signaal. De echo van het uitgestuurd signaal wordt als het ware berekend. Door deze echo van het werkelijke signaal op de lijn af te trekken, krijgen we het signaal dat door de tegenpartij op de lijn gezet is. Om dit mogelijk te maken, wisselen de modems bij het leggen van de verbinding eerst een aantal testsignalen uit om te onderzoeken wat de precieze eigenschappen van de gebruikte telefoonlijn zijn.

Behalve het opvoeren van het aantal bits dat over de lijn gestuurd wordt, is men momenteel ook bezig met een real-time datacompressie en -expansie in het modem.

In de V32 standaard kunnen 5 bits per baudrate-periode overgestuurd worden. Hier van worden 4 bits gebruikt voor het oversturen van de informatie. Het vijfde bit wordt gebruikt als een soort pariteitscontrole voor de andere vier bits. Om vijf bits te kunnen coderen, hebben we in totaal 32 verschillende toestanden in het fasediagram nodig.

Behalve het opvoeren van het aantal bits dat over de lijn gestuurd wordt, is men momenteel ook bezig met een real-time datacompressie en -expansie in het modem. Hiermee kan men tot 3 maal de bitrate aan informatie oversturen. De effectieve bitrate kan dus al in de orde van 38400 bps liggen. Modems die gebruik maken van dergelijke technieken worden ingedeeld in de zogenaamde MNP (Microcom Networking Protocol) klassen. Binnen deze klassen, zijn behalve definities voor de wijze waarop de compressie tot stand gebracht wordt ook definities voor foutcorrecties aanwezig. Modems uit de MNP klasse 4 met een effectieve overdracht van 120% van de werkelijke snelheid (2400 bps) en MNP klasse 5 met een effectieve snelheid van 200% zijn momenteel goed te verkrijgen waarbij de prijs echter nog tamelijk hoog is.

Bij gebruik van datacompressietechnieken is er echter een "leuk" verschijnsel aan de hand. Het is momenteel bij amateurs gebruikelijk een file (bestand) eerst met behulp van een programma zoals PKZIP, PKARC etc. te comprimeren alvorens ze overgestuurd wordt. De in deze programma's gebruikte algoritmen zijn echter zo effectief dat de files bij

gebruik van een modem met hardware data-compressie groter wordt in plaats van kleiner..... In dat geval neemt de effectieve bitrate dus af in plaats van toe.

De begrenzing van datacommunicatie over het telefoonnet wordt gevormd door het medium koperdraad dat prima geschikt is voor het overbrengen van spraak maar slecht geschikt is voor het overbrengen van digitale informatie. Als nieuwste ontwikkeling kan de vervanging van het koperdraad door glasvezel genoemd worden. Als werkelijk het hele telefoonnet uitgevoerd is met glasvezel, dan wordt spraak ook gedigitaliseerd en wordt alles op een digitale in plaats van een analoge wijze overgestuurd. In dat geval nemen de communicatiesnelheden nog factoren toe.

Afsluiting

Als u nu een modem wilt kopen, wat moet u dan kopen. Uiteraard kan (en wil) ik hierop voor uw specifieke geval geen antwoord geven. Het kiezen van een modem is een afweging tussen verschillende factoren. Twee hiervan zijn de prijs van het modem en de telefoonkosten. Als u veel files uitwisselt met een Bulletin Board die niet bij u in het telefoondistrict zit, dan is een hoge snelheid voordelig omdat u dan veel minder telefoontijd verbruikt. Een 9600 bps modem kost momenteel zo'n fl. 2500,- terwijl een 2400 bps modem al voor +/- fl. 500,- te koop is. Een half uur telefoontijd in de goedkope uren kost ongeveer fl. 3,-; in de dure uren het dubbele. Als u dus veel informatie uitwisselt, dan is de aanschaf van een 9600 bps modem misschien rendabel. Wisselt u maar weinig files uit, dan denk ik dat de aanschaf van een 9600 bps modem niet loont. Met een 2400 bps of zelfs 1200 bps modem valt goed te werken waarbij 2400 bps mijn voorkeur heeft omdat bij het converseren met het bulletin board de schermen sneller opgebouwd worden.

Hebt u al een modem, dan is de situatie weer anders. De prijzen van de high speed modems dalen toch vrij snel en de verwachting is dat er binnen af-

zienbare tijd high speed modems voor ongeveer fl. 1000,- gulden te koop zullen zijn. Daarmee komen we dus op het bekende probleem in onze hobby; als je een jaartje wacht krijg je de dubbele capaciteit voor de halve prijs. Je krijgt nu een V22bis modem voor een lagere prijs als een V22 modem enkele jaren geleden.

V21 en V23 worden nog wel gebruikt doch ik zou geen modem meer kopen die alleen deze twee standards ondersteunt. V21 is erg traag (300 bps) en V23 is alleen geschikt om informatie ergens vandaan te halen. Files naar een bulletin board sturen met V23 is onmogelijk omdat u dan slechts 7.5 tekens per seconde over kan brengen. V23 is alleen geschikt voor viditel etc. en voor gebruik bij bedrijven waarbij je met de PC bestellingen kunt doen.

Extra geld uitgeven voor een modem die behalve andere snelheden ook V23 ondersteunt is ook zo iets.

Ik modem nu ongeveer 3 jaar en heb slechts één maal contact gehad met een systeem dat alleen V23 aan kan. Er zijn wel meer van dat soort (BBS-) systemen maar over het algemeen ondersteunen ze ook bijvoorbeeld V22. Kortom ook hier hangt het er weer vanaf wat u wilt en hoeveel geld u uit wilt geven. De verwachting is dat u binnen afzienbare tijd overal met V22 en V22bis terecht kunt en bij een heleboel bulletin boards met V32.

Als werkelijk het hele telefoonnet uitgevoerd is met glasvezel, dan wordt spraak ook gedigitaliseerd en wordt alles op een digitale in plaats van een analoge wijze overgestuurd.

Zo, dit was het voor deze keer. Waar ik het de volgende keer over ga hebben weet ik nog niet; dat is dus nog een verrassing. Zeker is, dat ik het in deze serie nog ga hebben over datacompressie van files en over de protocollen waarmee files overgestuurd worden. Mogelijk ga ik het verderop in de serie ook nog over de geavanceerdere datacommunicatie hebben zoals X25 en TCP/IP. Hebt u ideeën voor onderwerpen, laat ze dan even aan me weten. Waarschijnlijk kan ook met uw wensen rekening gehouden worden.

Gert van Opbroek

MINIX - Unix in een notedop

Unix is een multitasking, multiuser operating system. Unix is een multitasking, multiuser operating system. Het is een van de oudste operating systems die nog steeds gangbaar zijn en de Unix wereld groeit nog steeds. Het begon allemaal in 1969 in de Bell Laboratories, een dochteronderneming van het Amerikaanse firma's AT&T en Western Electric. In dat jaar voltooide Ken Thompson een operating system onder de naam Unix. Het OS draaide op een PDP-7, een mini-computer van DEC en was geheel in assembler geschreven. Het operating system maakte het mogelijk dat meerdere programmeurs tegelijkertijd interactief van het systeem gebruik konden maken via een shell.

Daar assembler programmatuur van enige omvang niet echt makkelijk te onderhouden is, werd al snel besloten het OS te herschrijven in een hogere taal. Thompson ontwikkelde speciaal daarvoor de taal B. Dennis Richie nam de draad over en ontwikkelde B verder uit tot de taal C. Sinds 1973 is C de meest gebruikte taal in Unix programmeeromgevingen, een direct gevolg van het feit dat Unix zelf in C geschreven was. De source van Unix werd toen nog bij de programmatuur geleverd waardoor het OS makkelijk op andere systemen overgezet werd. Veel universiteiten kochten de source tegen kostprijs en gebruikten die ter ondersteuning van hun informatica colleges over systeembouw en soortgelijke vakken. Unix vond zijn weg in de technische wereld vanuit de universiteiten en tegenwoordig heeft Unix zich ontwikkeld tot een krachtig OS dat op vele computers verkrijgbaar is. De nieuwste versies ondersteunen windows en gespecialiseerde GUI's, een acroniem van "Graphics User Interface".

In 1979 kwam Unix 7 uit. Vanaf die versie kwamen andere fabrikanten eigenlijk pas goed in de markt. De belangrijkste afwijkende versies zijn Xenix, BSD en Ultrix. Xenix is een product van Microsoft en loopt op een PC met een 80286, 386 of 486 processor. BSD is een speciale versie van Unix, ontwikkeld op de University of California in Berkeley. Berkeley Unix is zo'n beetje de grootste concurrent voor het "originale" AT&T Unix. Daarnaast is er speciaal voor de VAX door Digital Equipment een versie ontwikkeld onder de naam "Ultrix".

Met het verschijnen van Unix versie 7 in 1979 werd de source echter een stuk duurder en mocht niet meer gebruikt worden voor onderwijsdoeleinden. Daardoor ontstonden er op verschillende universiteiten alternatieve operating systems met grappig klinkende namen als "XINU" (een acroniem van "Xinu Is Not Unix"). Op de VU in Amsterdam vond minix zijn oorsprong. Het minix project werd in beginsel opgezet door Andrew S. Tanenbaum, en werd begeleid door een redelijk dik boek "Operating Systems: design and implementation" van diezelfde Tanenbaum. In die boek worden de beginselen der OS-bouw aan de hand van een uitgebreide tour door minix uit de doeken gedaan. Helaas is het boek blijven steken in minix 1.1, en sindsdien heeft minix zich ontpopt tot een redelijk uitgebreid "echt" operating system.

In principe draait minix ook op een PC met maar twee floppen, maar de beperkte diskruimte zal al spoedig beperkingen opleggen aan de gebruiker

Het minix operating system draait inmiddels op IBM-compatible PC's en de Atari ST. Aan het einde van deze zomer komen er ook versies voor de Commodore Amiga en Apple Macintosh op de markt. Het operating system kost ca. f 300,00 en is verkrijgbaar bij de betere vakhandel. Hoewel de meest recente versie het nummer 1.5.10 draagt, levert Prentice Hall versie 1.3 als laatste versie. De updates naar 1.5.10 zijn verkrijgbaar via de commerciële netwerken (Internet, Bitnet, Surfnet etc.).

Versie 1.5.10 draait bij mij thuis op een PC/XT met een 32 MByte harddisk. Van die harddisk is vijf megabyte gereserveerd voor minix, de rest wordt in beslag genomen door PC-DOS. In principe draait minix ook op een PC met maar twee floppen, maar de beperkte diskruimte zal al spoedig beperkingen opleggen aan de gebruiker. Doordat het gehele systeem inclusief source (in C) geleverd wordt, kunnen aanpassingen door iedereen makkelijk doorgevoerd worden. Wijzigingen in de kernel vereisen natuurlijk wel dat de gehele kernel opnieuw gecompileerd wordt, hetgeen zonder harddisk niet mogelijk is. Het geheel ondersteunt standaard drie terminals: de console en twee terminals welke via de seriële poorten aangesloten kunnen worden. Een modem kan ook als terminal fungeren.

J.Voorhaar

SDN file area's

Op het BBS bevinden zich sinds enige tijd enkele nieuwe file gebieden. Dit zijn de zogenaamde SDN file area's. SDN staat voor Shareware Distribution Network. Dit houdt in dat de files die in deze area's te vinden zijn onder de noemer Shareware vallen, dit houdt dus in dat de gebruiker de software eerst kan proberen voordat hij/zij besluit tot registratie en/of aankoop overgaat van het pakket. De files worden elke maandag, woensdag en vrijdag automatisch op een centraal knooppunt opgehaald. De files zijn verdeeld zijn verdeeld over verschillende onderwerpen. Zo bestaan er de volgende onderwerpen waaronder de files zijn gerangschikt: Business/Financieel, DataBase/Spreadsheet, Non Shareware specials, Grafische applicaties, Programming, Utilities, Tekstverwerking en aanverwante files, Shareware misc en Communicatie software.

De files richten zich allen op de PC (helaas voor de vele DOS65, Amiga, en Atari gebruikers). Het grote nadeel van de files is misschien dat ze allen zo groot van omvang zijn. 200-400 Kb is geen uitzondering. Het telefonisch weghalen van het BBS is daardoor niet altijd even leuk. Vooral als je vanuit buiten het basistarief van Enschede belt. Het valt natuurlijk mee als je een zo snel mogelijke modem gebruikt. Dan vallen de kosten wel weer mee.

Op het moment is het bijvoorbeeld mogelijk om met een snelheid van 14.400 Bps contact met het BBS op te nemen. Onderstaand een overzichtje wat het aan tijd scheelt bij het ophalen van een 200Kb grote file op verschillende snelheden:

1200 Bps : 28 minuten
2400 Bps : 14 minuten
14.400 Bps : 2.3 minuten

Als je vanuit je lokale basistarief belt, maakt het allemaal niet zo heel veel uit. Dan gaat de tikkenteller tenslotte maar eens per 5 of 10 minuten. Maar buiten het basistarief is dat elke 47 of 94 seconden. Dat gaat dan dus weer een stuk harder...

Wat de kosten dan zijn bij eenzelfde file zie je hieronder:

	Basistarief		Interlokaal	
	Dure tijd	Goedkope tijd	Dure tijd	Goedkope tijd
1200 Bps	0.90	0.45	5.40	2.70
2400 Bps	0.45	0.30	2.70	1.35
14.400 Bps	0.15	0.15	0.60	0.30

Hieruit blijkt dus dat een snellere modem zich op den duur vanzelf terugbetaald. Natuurlijk is een snelle modem in aanschaf een stuk duurder, maar korte verbindingstijden zijn natuurlijk nooit weg. (Hoeveel winst had de PTT het afgelopen jaar ?).

Doordat door mij besloten was om het BBS op het SDN-net aan te sluiten, is ook door mij besloten om zo'n snel modem aan te schaffen. De files voor de SDN area's moeten we tenslotte zelf ophalen.

Ik hoop dat als jullie weer eens op het BBS langskomen, jullie de area's kunnen waarderen. Zoals gezegd alleen voor de PC-gebruikers onder ons. Maar ik sluit niet uit dat daar in de naaste toekomst verandering in komt.

Jacques Banser

SOFT-WORDS! (sm)

8 bits microcontrollers van intel programmeren via DOS65 en AS

Je realiseert je het niet zo snel in deze tijd waarin een 7400 al een computer heet in de ogen van de reclamemensen maar er worden werkelijk 'overall' computers ingebouwd. Neem nou die hele oude Philips videorecorders (2020 ed.) of veel moderne en zelfs iets minder moderne toetsenborden, diastuurapparaten, huishoudelijke apparatuur enz. enz. Vaak gaat het dan niet om een 80486 met 16 Mbytes maar dat is voor dergelijke besturingsfuncties ook niet nodig. In deze situaties is een compacte microprocessor op zijn plaats, liefst met wat ram en ROM en parallel I/O aan boord, met een directe aansluiting voor een kristal als clock. Aan al deze voorwaarden voldoen de microcontrollers van intel. Er zijn overigens niet alleen 8 bit versies maar dit zijn nog wel de meest voorkomende, zeker in de dump. Intel begon met de 8041, 1k ROM en 64 bytes ram aan boord en 18 I/O lijnen. Hierna kwam de 8048 en dit is een werkelijk veel voorkomend IC. Er bestaat een versie van met eeprom (8748), meestal met een venster zodat de eeprom gewist kan worden maar soms ook in een plastic uitvoering zonder venster, eenmalig te programmeren door de gebruiker. Dan is er ook een versie zonder ROM, te gebruiken met een externe eeprom (de 8035). Iets later kwam een iets uitgebreidere versie op de markt (8049/8749/8039) met weer iets meer ROM en ram aan boord maar dezelfde instructieset. De 8048 werkt standaard met een Xtal van 6MHz, de 8049 kan zelfs met een Xtal van 11MHz aan de slag. Wat kan zo'n ding? Heel veel. Eigenlijk is er niet zoveel verschil met een echte microprocessor. Alleen is een microcontroller meer gericht op het bekijken van signalen, testen en dan bijvoorbeeld conditioneel springen in plaats van de brute rekenkracht die je in de grotere broers vindt. Als je dus iets wilt ontwerpen waarbij het aankomt op testen en controleren, dan loont het de moeite om even in de dumpbak te kijken. Ik haalde pas nog een 8748 uit een gesloopte 5Mb winchester drive, je weet wel, zo'n hoge. Een Duits elektronica bedrijf

adverteteerde enige tijd terug met pakketjes van 5 x 8049 voor drie mark. Daar zit dan interne (geprogrammeerde) ROM in maar daar heeft intel het volgende op gevonden. Er bevindt zich op die controllers met ingebouwde ROM een input EA (external access). Als je deze ingang hoog maakt dan schakel je de interne ROM uit en zoekt de controller z'n instructies buitenshuis dwz. uit een eeprom. De controllers hebben meerdere I/O lijnen (27 voor de 8048/49) waarvan een gedeelte als databus en adresbus te gebruiken is. Zo kun je externe chips aansluiten. Er is ook een speciale I/O chip voor op de markt gebracht, de intel 8043. Zo kun je dan het aantal I/O lijnen eenvoudig uitbreiden, maar dat kan natuurlijk ook op andere manieren. Om het programmeren van de 8048 familie wat eenvoudiger te maken heb ik een macro file gemaakt voor de as-

sembler van DOS65, waarin je alle instructies kunt vinden die in de instructieset van de 8048 voorkomen. Als je hiermee een programma schrijft en deze macro file in je programma oproept mbv. 'lib i8048', dan wordt het geheel keurig netjes in 8048 machine code vertaald. De 8049 heeft exact dezelfde instructieset dus ook hiervoor kun je dezelfde file gebruiken. Wie met de modernere 8051 of 8052 aan de gang wil kan gerust uitgaan van deze macro listing maar zal er het een en

ander aan moeten toevoegen. Per slot van rekening zijn deze controllers weer een stukje uitgebreider (RAM, ROM, I/O, timers, en zelfs seriële in- en output). Ik hoop dat deze macrolisting van nut kan zijn. Het demonstreert evenals de door mij ooit eens gepubliceerde 8080 macro's en de door iemand anders in de club geschreven 65C816 macro's dat de DOS65 macroassembler een zeer veelzijdige assembler is en zeker voor 'thuis' projecten een waardevol stuk gereedschap. Dit artikel werd trouwens geschreven op een DOS65 computer met daarin een.....8039 met 6MHz Xtal en een 2716.

Ernst Elderenbosch

Alleen is een microcontroller meer gericht op het bekijken van signalen, testen en dan bijvoorbeeld conditioneel springen in plaats van de brute rekenkracht die je in de grotere broers vindt

```

;filei8048.mac
                                opt      nolis
;
;                                macro file for 8048/35/49/39/8748/49 code assembly
;
;                                all devices with internal rom can be forced to use
;                                external rom by applying +5v to ea (pin 7). this way
;                                you can use an old microcontroller from a keyboard
;                                or a videorecorder or whatever for a new project!
;
;                                intel 8 bit micro controller family
;
;                                rom      eprom      extern      romrami/otimerclck
;
;                                8041      8741      1k          641816MHz
;                                8048      8748      8035       1k642716MHz
;                                8049      8749      8039       2k12827111MHz
;                                8051      8751      8031       4k12832212MHz
;                                8052      8752      8032       8k25632312MHz
;
;
;                                intel 8048/8748/8035 code macro's
;
;                                registers 0-7 have values 0-7
;
;                                accumulator instructions
;
;                                iadd      macro    byt          add a, #data
;                                fcc      $03,byt
;                                endm
;
;                                iadr      macro    reg          add a, reg
;                                fcc      $68|reg
;                                endm
;
;                                iadri     macro    reg          add a, [reg 0/1]
;                                fcc      $60|reg
;                                endm
;
;                                iaddc     macro    byt          addc a, #data
;                                fcc      $13,byt
;                                endm
;
;                                iadcr     macro    reg          addc a, reg
;                                fcc      $78|reg
;                                endm
;
;                                iadcri    macro    reg          addc a, [reg 0/1]
;                                fcc      $70|reg
;                                endm
;

```

ianl	macro	byt	anl a, #data
	fcc	\$53,byt	
	endm		
;			
ianr	macro	reg	anl a, reg
	fcc	\$58 reg	
	endm		
;			
ianri	macro	reg	anl a, [reg 0/1]
	fcc	\$50 reg	
	endm		
;			
icpl	macro	cpl a	
	fcc	\$37	
	endm		
;			
iclr	macro	clr a	
	fcc	\$27	
	endm		
;			
idaa	macro	da a	
	fcc	\$57	
	endm		
;			
idec	macro	dec a	
	fcc	\$07	
	endm		
;			
iinc	macro	inc a	
	fcc	\$17	
	endm		
;			
iorl	macro	byt	orl a, #data
	fcc	\$43,byt	
	endm		
;			
iorr	macro	reg	orl a, reg
	fcc	\$48 reg	
	endm		
;			
iorri	macro	reg	orl a, [reg 0/1]
	fcc	\$40 reg	
	endm		
;			
irl	macro	rl a	
	fcc	\$e7	
	endm		
;			
irlc	macro	rlc a	
	fcc	\$f7	
	endm		
;			
irr	macro	rr a	
	fcc	\$77	
	endm		
;			
irrc	macro	rrc a	

	fcc	\$67			
	endm				
;					
iswap	macro	swap a			
	fcc	\$47			
	endm				
;					
ixrl	macro	byt	xrl a, #data		
	fcc	\$d3,byt			
	endm				
;					
ixrr	macro	reg	xrl a, reg		
	fcc	\$d8 reg			
	endm				
;					
ixrri	macro	reg	xrl a, [reg 0/1]		
	fcc	\$d0 reg			
	endm				
;					
;					
		branch instructions			
;					
idjnz	macro	reg,addr	dnjz reg,addr		
	fcc	\$e8 reg,addr			
	endm				
;					
ijbb	macro	bit,addr	jbb addr		
	fcc	\$12 bit < 5,addr			
	endm				
;					
ijc	macro	addr	jc addr		
	fcc	\$f6,addr			
	endm				
;					
ijf0	macro	addr	jf0 addr		
	fcc	\$b6,addr			
	endm				
;					
ijf1	macro	addr	jf1 addr		
	fcc	\$76,addr			
	endm				
;					
ijmp	macro	addrh,addrl	jmp addr (2k)		
	fcc	\$04 addrh < 5,addrl			
	endm				
;					
ijmpp	macro	jmp [a] (within current page)			
	fcc	\$b3			
	endm				
;					
ijn	macro	addr	jnc addr		
	fcc	\$e6,addr			
	endm				
;					
ijni	macro	addr	jni addr		
	fcc	\$86,addr			
	endm				

```

;
;ijnt0      macro   addr          jnt0 addr
;            fcc     $26,addr
;            endm
;
;ijnt1      macro   addr          jnt1 addr
;            fcc     $46,addr
;            endm
;
;ijnz       macro   addr          jnz addr
;            fcc     $96,addr
;            endm
;
;ijtf       macro   addr          jtf addr
;            fcc     $16,addr
;            endm
;
;ijt0       macro   addr          jt0 addr
;            fcc     $36,addr
;            endm
;
;ijt1       macro   addr          jt1 addr
;            fcc     $56,addr
;            endm
;
;ijz        macro   addr          jz addr
;            fcc     $c6,addr
;            endm
;
;
;            control instructions
;
;ieni       macro   eni
;            fcc     $05
;            endm
;
;idisi      macro   disi
;            fcc     $15
;            endm
;
;ient0c     macro   ent0clk
;            fcc     $75
;            endm
;
;ismb0      macro   selmb0 (select first 2k prog mem)
;            fcc     $e5
;            endm
;
;ismb1      macro   selmb1 (select sec. 2k prog mem)
;            fcc     $f5
;            endm
;
;isrb0      macro   selrb0 (select reg at data addr 0-7)
;            fcc     $c5
;            endm
;
;isrb1      macro   selrb1 (select reg at data addr 24-31)
;            fcc     $d5
;            endm

```

44

	endm		
; imxria	macro fcc endm	reg \$90 reg	mov ext[reg 0/1], a
; ixar	macro fcc endm	reg \$28 reg	xch a, reg
; ixari	macro fcc endm	reg \$20 reg	xch a, [reg 0/1]
; ixdari	macro fcc endm	reg \$30 reg	xch a, [reg 0/1] (lower nibble only)
; ; ;	flag instructions		
; icpc	macro fcc endm	cpl c \$a7	
; icpf0	macro fcc endm	cpl f0 \$95	
; icpf1	macro fcc endm	cpl f1 \$b5	
; iclc	macro fcc endm	clr c \$97	
; iclf0	macro fcc endm	clr f0 \$85	
; iclf1	macro fcc endm	clr f1 \$a5	
; ; ;	input / ouput instructions		
; ianb	macro fcc endm	data \$98,data	anl bus, #data
; ianp	macro fcc endm	port,data \$98 port,data	anl port 1-2, #data
; iand	macro fcc endm	port \$9c port	andl port (bit 4-7), a (bit 0-3)

```

;
iinp      macro    port          in a, port 1-2
fcc       $08|port
endm

;
iinsb     macro    ins a, bus
fcc       $08
endm

;
indap     macro    port          movd a (bit 0-3), port (bit 4-7)
fcc       $0c|port
endm

;
imdpa     macro    port          movd port (bit 4-7), a (bit 0-3)
fcc       $3c|port
endm

;
iorb      macro    data          orl bus, #data
fcc       $88,data
endm

;
iordp     macro    port          orld port (bit 4-7), a (bit 0-3)
fcc       $8c|port
endm

;
iorp      macro    port,data     orl port 1-2, #data
fcc       $88|port,data
endm

;
ioutb     macro    out bus, a
fcc       $02
endm

;
ioutp     macro    port          out port 1-2, a
fcc       $38|port
endm

;
;
;      register instructions
;
idecr     macro    reg          dec reg
fcc       $c8|reg
endm

;
iincr     macro    reg          inc reg
fcc       $18|reg
endm

;
iincrl    macro    reg          inc [reg 0-1]
fcc       $10|reg
endm

;
;
;      subroutine instructions
;
icall     macro    addrh,addrl    call addr (2k)
fcc       $14|addrh<5,addrl
endm

```

```

;
; ired          macro    retr (no psw)
;                fcc      $83
;                endm

;
; iredr         macro    retr (restore psw)
;                fcc      $93
;                endm

;
;              timer / counter instructions
;
; ienti         macro    en tcnti
;                fcc      $25
;                endm

;
; idisti        macro    dis tcnti
;                fcc      $35
;                endm

;
; imat          macro    mov a, t
;                fcc      $42
;                endm

;
; imta          macro    mov t, a
;                fcc      $62
;                endm

;
; istopc        macro    stop tcnt
;                fcc      $65
;                endm

;
; istrtc        macro    start cnt
;                fcc      $45
;                endm

;
; istrtt        macro    start tim
;                fcc      $55
;                endm

;
; inop          macro    nop
;                fcc      $00
;                endm

;
;
;              end of macro listing intel 8048/8049 microcontrollers
;
;              opt lis

```

Van de bestuurstafel

Staat er voor het eerst "bestuurstafel" boven dit verhaal, heb je meteen een probleem: het is al zowat een hele (vooral hete) zomer geleden dat het bestuur de koppen bij elkaar stak. Even graven in het geheugen en de stapel verslagen dus.

De belangrijkste ontwikkeling op de laatste bestuursvergadering was toch wel de kennismaking met MINIX. MINIX is een UNIX-achtig operating system voor bijvoorbeeld PC's. Weer eens iets anders dan dat eeuwige MS-DOS. Het blijkt dat de MINIX-mensen behoefte hebben aan wat meer technische backup op bijvoorbeeld hardwaregebied. En dat sluit mooi aan bij wat de club kan. Ook bestaat misschien de mogelijkheid dat we actief kunnen meewerken aan bijvoorbeeld een volgende release. Het een en ander zou kunnen uitmonden in iets wat wel lijkt op wat er vroeger met DOS65 gebeurt, en dat lijkt een positieve ontwikkeling.

Wat ook weer terug is, is de aanwezigheid van de club op de HCC-dagen. Het bestuur wil toch weer proberen op die happening de belangstelling te wekken van potentiële nieuwe leden. Er zijn al een aantal aardige ideeën over de tafel gegaan over wat we daar allemaal zouden kunnen gaan doen. We broeden verder.

Op de volgende bijeenkomst in Haarlem zal Joost Voorhaar ons allemaal laten kennismaken met MINIX. Ik hoop dat u net zo nieuwsgierig bent als ondergetekende. Over Joost gesproken: onze lay-out-man moet binnenkort onder de wapenen. Er zal gepoogd worden het lay-outen over te dragen aan iemand anders. Aan mij dus. Als het volgende nummer er dus minder mooi uit ziet, weet u waar u moet klagen. Bij:

Uw veurzitter,

Nico de Vries.

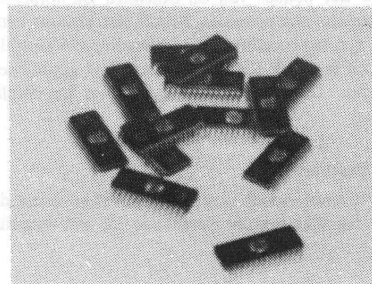
(Advertentie)

**Basis I/O met haken en ogen?
Alweer een 1782 Disk error?
Waar hangen mijn poorten uit?**

Dan biedt de KGN de oplossing... een nieuw BIOS! Het speciale KGN-BIOS bijvoorbeeld. Alleen verkrijgbaar voor leden...

Bestellen:

Maak f 25,00 over aan de KIM gebruikersclub Nederland in Enschede, giro 3757649 o.v.v. "KIM XT-BIOS".
U krijgt de EPROM dan zo snel mogelijk thuis gestuurd.



Informatie.

De μ P 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 μ P 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 μ P 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: 053-303902.

Software Bibliotheek en Technisch Forum:

Voor het beheer van de Software Bibliotheek en technische ondersteuning streeft het bestuur ernaar zgn. systeemcoördinatoren te benoemen. Van tijd tot tijd zal in de μ P Kenner een overzicht gepubliceerd worden. Dit overzicht staat ook op het Bulletin Board.

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.

Nico de Vries (voorzitter)
Mari Andriessenrade 49
2907 MA Capelle a/d IJssel
Telefoon 010-4517154

Mick Agterberg (secretaris)
Davidvosstraat 29
1063 HV Amsterdam
Telefoon 020-131538

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

Gert van Opbroek (Redactie μ P Kenner)
Bateweg 60
2481 AN Woubrugge
Telefoon 01729-8636

Jan D.J. Derksen
Ed Verkadestraat 9-1
7558 TH Hengelo
Telefoon 074-770970

Geert Stappers
Engelseweg 7
5825 BT Overloon
Telefoon 04788-1279

Ton Smits
De Meren 39
4731 WB Oudbosch

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 Mueller