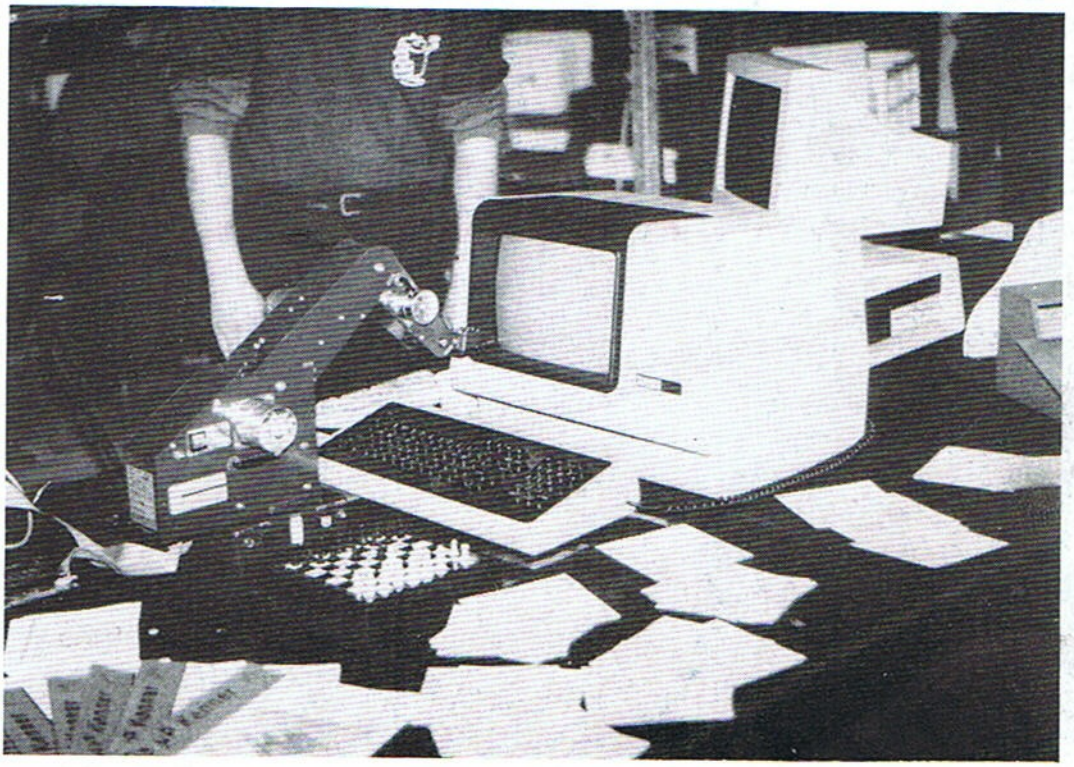
De µP Kenner

Vijftiende jaargang nr. 1
Februari 1991
70



In dit nummer o.a.:

Intel i960
Modem problemen
IBM keyboard aan DOS65
Het UNIX operating system
Shareware: PAKkers, ARCers en ZIPpers

Inhoudsopgave

De μP Kenner
Nummer 70, februari 1991
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
Nico de Vries

Redactieadres:
Gert van Opbroek
Bateweg 60
2481 AN Woubrugge

De μP Kenner nummer 71 verschijnt op 20 april 1991.

Kopijsluitingsdatum voor nummer 71 is vastgesteld op 6 april 1991.

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

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

Redactioneel

Op het moment dat ik dit schrijf, is de winter echt losgebroken. Overal ligt ijs en gisteren moest ik mijn belofte van drie jaar geleden waarmaken om bij sneeuw een slee voor de kinderen te gaan kopen. Kortom, eigenlijk is het tijd om me over te geven aan wat ijs- en sneeuwpret. Als u dit leest kan het natuurlijk allemaal al weer anders zijn, of zoals het spreekwoord luidt, het kan vriezen en het kan dooien.

Goed, behalve op het schaats- en sneeuwfront gebeuren er momenteel ook een heleboel zaken op het gebied van computers. Vooral binnen de KGN is er een heleboel beweging. In de eerste plaats is er natuurlijk het KGN-68k MINIX project. Voor dit project is een projectgroep gevormd waarvan ik de secretaris ben. Dat heeft enerzijds het voordeel dat ik de zaken uit de eerste hand weet en het hele project met publicaties in de μP Kenner kan begeleiden. Anderzijds houdt dit natuurlijk ook het risico in dat het blad een te eenzijdige vulling krijgt. Mocht het die kant uitgaan, dan wil ik graag dat u aan de (telefoon-) bel trekt; het blad hoort namelijk een zo goed mogelijke afspiegeling te zijn van wat er binnen de club leeft en het is met name de taak van de redacteur hier zo goed mogelijk voor te zorgen. Uiteraard moet het KGN-68k project wel in het blad voorkomen. Het is tenslotte een vrij belangrijke activiteit waar drie bestuursleden en een vijftal andere clubleden zeer actief mee bezig zijn. Het verslag van de projectgroep met daarin de stand van zaken staat in deze μP Kenner.

Een andere zaak die sinds november is gaan spelen zijn wat nieuwe ontwikkelingen in DOS-65. De vorige keer heb ik een oproep geplaatst om te achterhalen of er belangstelling bestaat voor nieuwe ontwikkelingen en een uitbreiding in de vorm van DOS-65 versie 3. Op deze oproep heb ik zeker tien reacties gekregen. Het blijkt dus dat DOS-65 nog steeds volop in de belangstelling staat. Er heeft zich zelfs iemand aangemeld die de rol van DOS-65 coördinator op zich wil nemen en die ten aanzien van DOS-65 versie 3 een zeer actieve rol wil gaan spelen. Hij heeft zelfs al aangegeven dat hij wil onderzoeken of er toch niet op de één of andere manier een 1 MByte RAM-kaart ontwikkeld kan worden inclusief een printontwerp. Tenslotte kun je tegenwoordig met één SIP of SIM op eenvoudige wijze iets dergelijks opzetten. Vanuit het bestuur zal dit project waarschijnlijk ook door middel van het opzetten van een projectgroep opgestart gaan worden. We willen in overleg met Jan Derksen als verantwoordelijk bestuurslid en de eventuele nieuwe DOS-65 coördinator onderzoeken hoe we dit het beste op kunnen gaan zetten. Vervolgens zal er waarschijnlijk een vergadering belegd worden waarvoor de mensen die actief met DOS-65 versie 3 aan de gang willen gaan uitgenodigd zullen worden. Vervolgens kan dan ook deze projectgroep van start gaan. Zelf denk ik dat we in de loop van maart of april duidelijkheid hebben over hoe, wat en wanneer dit allemaal gaat gebeuren. Eén ding is echter zeer duidelijk: DOS-65 is nog steeds springlevend en geniet binnen de KGN een warme belangstelling.

Goed, zoals uit het bovenstaande blijkt, zit er in de KGN zeer veel beweging. Er vinden weer een aantal activiteiten plaats, de bijeenkomsten worden weer redelijk tot goed bezocht en het ledental begint ook weer licht te stijgen. Dat zijn allemaal tekenen dat we waarschijnlijk door het dal heen zijn en dat we op de goede weg zijn. Ook wat betreft de kopij-situatie staat het er tamelijk goed voor. In de eerste plaats krijg ik steeds meer bijdragen in de vorm van programma's en artikelen, in de tweede plaats leveren ook de projecten die we onderhanden hebben voldoende stof tot schrijven. Dit houdt in dat we de laatste maanden ons vooral moeten buigen over de vraag "Wat laten we eruit" in tegenstelling tot de vraag "Wat moeten we nu weer in het blad zetten". Dit betekent natuurlijk niet dat ik liever geen kopij meer krijg. Integendeel! Hoe meer kopij we krijgen, hoe meer mogelijkheden we hebben een gevarieerd blad samen te stellen. Verder geldt natuurlijk als vanouds dat alles wat ingeleverd wordt in principe geplaatst wordt. Het kan alleen wel eens wat langer duren. Mochten we echt veel te veel kopij krijgen, dan kunnen we altijd nog met de penningmeester overleggen of we het blad dikker zullen gaan maken of dat we een zesde nummer uit gaan geven.

Tenslotte wens ik alle leden en de andere lezers natuurlijk heel veel plezier toe bij hun computerhobby. Mocht u eens persoonlijk met mij of één van de andere bestuursleden van gedachten willen wisselen, dan kunt u ons bereiken via de telefoonnummers die in het blad staan, het Bulletin Board The Ultimate of ontmoeten op één van de bijeenkomsten.

Gert van Opbroek

Uitnodiging voor de clubbijeenkomst

Datum: 16 maart 1991
Lokatie: gebouw 't Kruispunt
Slachthuisstraat 22
5664 EP Geldrop
tel: 040-857527
Thema: Netwerken
Entreeprijs:fl. 10,--

Routebeschrijving

TREIN:

Geldrop is ieder half uur bereikbaar per trein (stoptrein Eindhoven-Weert). Vanuit het station rechts afslaan, de Parallelweg, dan tweede straat links, de Laarstraat. Aan het einde daarvan rechts afslaan en direct daarna weer linksaf, de Laan der vier Heemskinderen. Op de hoek van de eerste straat links, de Slachthuisstraat, vindt u het gebouw 't Kruispunt.

AUTO:

Vanaf 's Hertogenbosch of Breda naar autoweg Eindhoven-Venlo. De eerste afslag na Eindhoven is Geldrop. Ga richting Geldrop, dan komt u vanzelf op de Laan der vier Heemskinderen, dit is nl. een verplichte afslag naar rechts. Zie verder boven.

Vanaf Eindhoven door het centrum van Geldrop richting Heeze. Na winkelstraat en daarna het ziekenhuis aan de rechterzijde de eerste straat links bij de stoplichten. Dit is de Laan der vier Heemskinderen. Zie verder boven.

Programma:

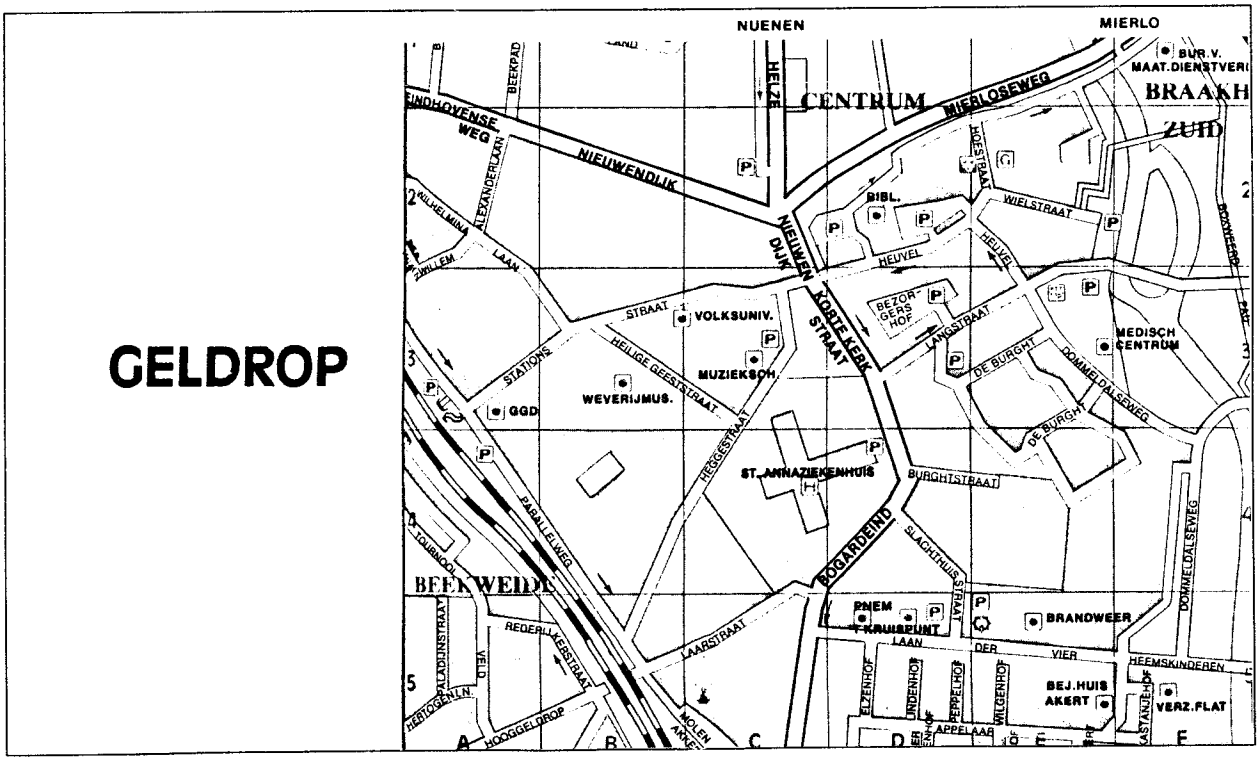
9:30	Zaal open met koffie
10:15	Opening
10:30	Voordracht. Op het moment van het ter perse gaan van dit nummer was het onderwerp, noch de spreker nog niet bekend. Zie het BBS voor het laatste nieuws.
11:30	Stand van zaken t.a.v. het project KGN-68k; gelegenheid voor het stellen van vragen aan en een discussie met vertegenwoordigers van de projectgroep.
12:00	Forum en markt
12:15	Lunchpauze

Aansluitend het informele gedeelte met de mogelijkheid om andermans systemen te bewonderen en Public Domain software uit te wisselen. U en uw systeem zijn uiteraard van harte welkom.

17:00 Sluiting

Let op:

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.



Met alleen een (goedgekeurd) modem ben je er niet...

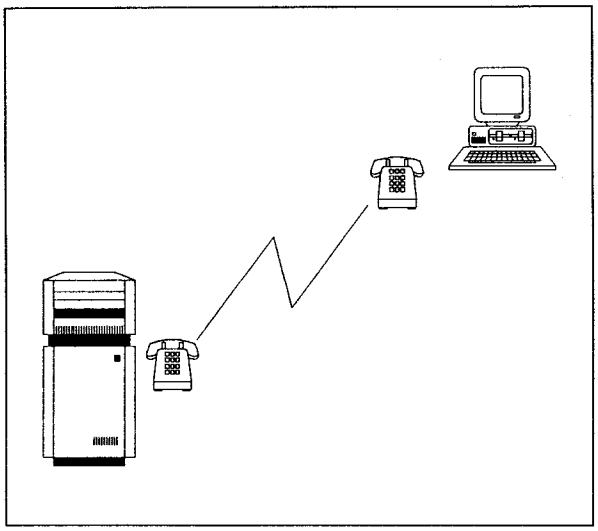
Alweer een drietal kwartalen terug verscheen er in mijn computerwereld een modem. Zo'n ding dat je tussen je telefoon-aansluiting en je PC kunt knopen, zodat je een met een andere PC kunt communiceren. Op het eerste gezicht niets bijzonders, want iedereen doet dat al jaren. Ik loop misschien zelfs wel een beetje achter.

Dat modem leek een serieus ding. Er zit bijvoorbeeld een blauw stickertje op van het Mysterie van Rijks-wat-er-staat. Ik heb begrepen dat je zonder zo'n stickertje zonder meer kans loopt op levenslange gevangenisstraf, en als je als handelaar het in je hoofd haalt om modems zonder zo'n stickertje te verkopen, kun je ter plekke zonder voorafgaand proces ge-executeerd worden, net als de files trouwens die je met zo'n modem kunt down-loaden.

Het is niet alleen een serieus modem vanwege dat stickertje. Het apparaat beheerst vrijwel alle gangbare snelheden tussen 75 baud en de al zowat illegale 120 km/h en stampt op verzoek de data nog met een factor twee in elkaar. NMP5 heet dat geloof ik. Verder neemt het ook nog vanzelf de telefoon op in AA-mode. (Fijn dat het modem van de drank af is: aan verzopen files heb ik ook niks). Tenslotte was het een modem van degelijk vaderlands en hoogstaand fabrikaat. Wat wil je nog meer...

Met bulletin boards bellen bijvoorbeeld. Om te beginnen met het bulletin board van dit clubje. ATDT053-303902 tikte ik in in PROCOMM. Wat dacht je? In gesprek natuurlijk. Onze sysop heeft zo'n interessant board, dat heel Nederland er naar toe belt. En ik maar wachten. Na verloop van tijd kon ik via het speakertje horen, dat de overkant eindelijk besloten had de lijn op te nemen. Er vlogen wat piep-tonen heen en weer. Maar meestal gebeurde er verder niets. Slechte lijn dacht ik dan. Of: ik zal wel wat verkeerd doen. Soms echter verscheen er CONNECT 2400, en moest ik plotseling allemaal moeilijke vragen beantwoorden. Waar ik woonde enzo. En wat voor password ik heb. Prompt weer vergeten dus... de volgende keer weer die vragen.

Je wilt naar verloop van tijd meer, zoals iedereen in deze welvaartsmaatschappij. Naar andere bulletin boards bellen bijvoorbeeld. Nu blijkt, dat die andere boards waarvan ik met veel moeite het nummer achterhaald had, ook voortdurend in gesprek zijn. Pas later vond ik uit, dat er een hele mooie lijst van BBS'en in omloop is. Toch kreeg ik langzamerhand een vreemd gevoel in mijn buik: het begon op te vallen, dat ik alleen met ons eigen club-BBS kon bellen, en met niemand anders. Vreemd eigenlijk, want ik had toch een serieus modem (dacht ik).



Meestal werkt dit geheel volgens verwachting...

Eens rondgevraagd. Meestal was het antwoord: is jouw modem wel goed? Waarschijnlijk niet dacht ik. Met de fabrikant gebeld. Of ik het modem on-line had. Had ik. Hij ook. Er ontstond geen verbinding. Terug op spraak zei de fabrikant: laten we het eens omdraaien: dus mijn modem in die bezopen AA-mode. Prompt verbinding. In Originate gaat het niet, en in Answer wel. Modem defect, dus opsturen maar. Fabrikant kon na met wel twintig bulletin boards gebeld te hebben niets vinden. Modem terug, zij het voor de zekerheid licht gemodificeerd. Ik aan het bellen. Alleen met de Ultimate kon contact gelegd worden. Weer met de fabrikant in de clinch. Modem weer retour. Mankeert niets aan was het resultaat. Tip van de fabrikant: waarschijnlijk is jouw telefoonlijn niet goed. Toch was dat gek: want met spraak is de lijn mooi stil: geen kraakjes, rateltjes of ruisjes, en nog goed te verstaan ook.

Hint: probeer het modem eens op een andere plaats. Op twee plaatsen geprobeerd: perfect deed het ding het. PTT storingsdienst gebeld. PTT legt uit: het telefoonnet is er voor spraak en niet voor modems. Bovendien was het zo dat modems slechtere lijnen konden gebruiken (grapje?), dus het modem was stuk en derhalve de fabrikant een nul. Nu neem ik die fabrikant zeer serieus, dus dat geloofde ik maar niet. Na enig volhouden bleek men bereid de lijn op te meten vanuit de centrale. Ophangen en wachten tot er teruggebeld wordt. Na tien minuten telefoon: de lijn was perfect in orde: het modem is duidelijk stuk.

Twee dagen later liep ik in de stad en kwam toevallig langs de Primafoonwinkel. Verkopen ze ook modems overigens. Met stickertje. Het probleem uitge-

legd. Lag aan het modem. Kan niet anders. De lijn nog een keer vanuit de centrale gemeten. Mankeert niets aan. Advies: als u nu eens datalijn huurt, dan bent u van alle problemen af. Ik kon mijn lachen bijna niet inhouden: voor een particulier is zoiets niet te betalen, en bovendien is een datalijn een vaste verbinding tussen twee punten gedurende 24 uur per etmaal. En dat zijn zaken waar ik niet echt behoefte aan heb, want ik wil ook graag met andere BBS'en werken.

Ook kreeg ik nog de schuld dat ik waarschijnlijk drie Hong-Kong speelgoed telefoons parallel had hangen aan een niet goedgekeurd modem en dat het daaraan moest liggen. Nu wil het geval dat er in mijn huishouden precies 1 telefoontoestel voorkomt, en dat is toevallig geleverd door de PTT. Ondertussen kon ik nog steeds niet met mijn technisch modemwonder uit de voeten. Maar afijn.

Ik belde als kritische consument met de PTT klantenservice met de vraag wat voor technische eisen ik aan een telefoonverbinding mocht stellen. Het antwoord was even eenvoudig als verbijsterend.

Ik besloot het probleem eens vanuit een andere hoek te benaderen. De PTT levert namelijk een dienst: vanuit mijn telefoonstopcontact kan ik verbindingen maken met de rest van de wereld. Nu is er zoiets als een bepaald kwaliteitsniveau, en zulke zaken zijn vaak geregeld in leveringsvoorwaarden. Ik belde als kritische consument met de PTT klantenservice met de vraag wat voor technische eisen ik aan een telefoonverbinding mocht stellen. Het antwoord was even eenvoudig als verbijsterend. Er worden vier simpele vraagjes gesteld. Als je die allemaal met ja kunt beantwoorden dan krijg je NIET de hoofdprijs, maar heb je een uitstekende telefoon verbinding. Hier komen ze, houdt u vast:

1. Kunt u bellen?
2. Kunt u gebeld worden?
3. Kan men u verstaan?
4. Kunt u de overkant verstaan?

Vier keer ja? Alles voor de bakker toch! Maar waarom vraagt u naar de kwaliteit van een telefoonverbinding? Deze lijn is toch prima? Probleem uitgelegd. Dit was toch wel een vreemd geval. Nooit meegemaakt. Technische dienst geraadpleegd. Kan wel wat aan gedaan worden. Moet u alleen wel de rekening betalen als de lijn in orde blijkt. Kost tussen de fl. 100,- en fl. 150,-. En, als 'ie in orde is, doen we er niets aan, want er is immers geen storing? Maar wacht even... Ik moet dus betalen als er niets gebeurt en als er wel wat is, is het gratis? En als ik dan moet betalen, heb ik nog steeds hetzelfde probleem! Inderdaad. Ook met een modem met een stickertje? Inderdaad.

Ik kreeg een idee. Vraagje aan de klantendienst: wat gaan jullie doen als ik een girotelmodem bestel en het werkt niet? Geen idee. En als ik nu eens Viditel-abonnee wil worden, wat dan? Ik kan die diensten niet bereiken! Tja... dat is inderdaad een probleem. Uiteindelijk werd ik doorverbonden met iemand die alles van lijnen afwist. Die was het met mij eens, dat je zo langerzamerhand wel van het telefoonnet mag verwachten dat je betrouwbaar met een modem kunt werken. Zeker als je met goedgekeurde spullen aan komt dragen en alles degelijk voor elkaar is. Tenslotte biedt de PTT ook zelf modemdiensten aan, nietwaar? De man kon mij vertellen dat ik relatief ver van de centrale af woon. Verder zou hij kijken wat hij kon doen. Hij gaf de suggestie eens een scoop op de lijn te zetten en de signalen te bekijken.

Heb ik gedaan. Op de lijn staat 250 mV_{tt} brom. Een mooie sinus. Spat-synchroon met de netspanning. Bel ik een BBS, dan komt de carrier met 80 mV_{tt} of minder binnen, en dat is dan een sterke. Het modem ziet door de brom het bos niet meer, dat is duidelijk. Ik ben benieuwd wat er nog gaat gebeuren. Want ik kan nog steeds alleen maar met The Ultimate bellen. Omdat de sysop het transmit-niveau wat hoger heeft gezet.

Daar zit je dan. Met een duur goed-gekeurd modem. Doet het niet. Laatst leende ik van een kennis een waaibomen-modem zonder sticker uit Verweggistan. Deed het perfect op 1200 baud, en redelijk op 2400 baud. Wat willen ze nou eigenlijk bij de PTT? Want als ik mijn modem gebruik, levert dat tikken op, die op hun beurt weer betaald moeten worden. Omzet heet dat. De meeste tijd heb ik nog gebeld met de klantendienst en de storingsdienst. En die nummers zijn gratis...

Voor electronisch commentaar ben ik helaas NIET bereikbaar, op geen enkel uur van de dag, noch met V21, V22 of V23, laat staan met V22bis. Was getekend:

A. Answer-Originate

Het standaardoperating system "UNIX" (deel 3)

De shell is de interface van een UNIX systeem tussen de gebruiker en de kernel. De UNIX shell is een stuk geavanceerder gereedschap dan de command interpretor van bijvoorbeeld MS-DOS. Als ik zeg "de shell", dan bedoel ik in dit geval de standaard Bourne shell. Want ook in de UNIX wereld komen uitgebreidere shells voor. De bekendste is zonder twijfel de C-shell, de standaard shell van BSD (de Berkeley variant van UNIX). Daarnaast is de zogenaamde kshell in populariteit aan 't toenemen.

Allereerst draagt de shell zorg voor het starten van externe processen. Een proces kan heel erg simpel zijn (bijvoorbeeld het laten zien van een directory), maar kan ook een complex CAD-pakket behelzen. Door het commando te laten eindigen met een ampersant ("&"). Het programma wordt dan "geforked" en de shell wacht niet op het beëindigen van het programma, maar gaat direct door met het volgende commando. De shell antwoordt met het procesnummer van het zojuist gestarte proces. Voorbeeld:

```
cc example.c &
# start de c compiler in de achtergrond.
```

Redirection

De shell draagt ook zorg voor het redirecten van in- en output. De bekende redirection-tekens uit de DOS-wereld zijn afkomstig van UNIX. Dus: voor het redirecten van invoer wordt de "kleiner dan" ("<") gebruikt, voor redirecten van uitvoer het "groter dan" teken (">"). Achter het redirection teken volgt de naam van de file of device waar de output naar toe moet c.q. de invoer vandaan moet komen.

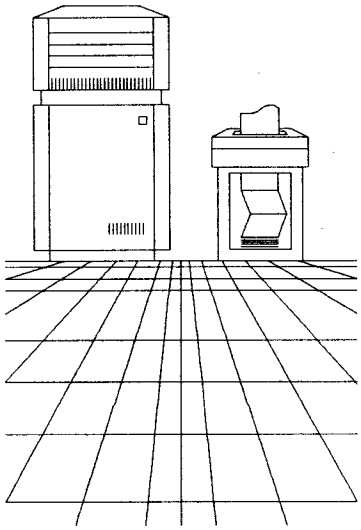
Voorbeelden:

```
ls -l > mydir.file
# Inhoud van current directory naar mydir.file
wc < tekst.file
# Tel woorden en regels in teskt.file
```

Als de output geredirect wordt naar een file die al bestaat, dan wordt de file overschreven. Is het de bedoeling dat de uitvoer van het commando aan een al bestaande file wordt toegevoegd, dan moeten er twee groter-dan tekens gebruikt worden.

Voorbeeld:

```
ls -l subdir > > mydir.file
# Voeg de inhoud van de subduirectory "subdir" toe aan mydir.file
```



Als een proces wordt gestart in de achtergrond, komt de standaard uitvoer normaliter op het scherm. De uitvoer van een dergelijk commando kan natuurlijk ge-redirect worden zodat de in de voorgrond draaidende taskniet gestoord wordt door de background task. Foutmeldingen die naar "standard error output" (in C: stderr) gestuurd worden, komen dan toch nog op het scherm. Kan in MS-DOS standard error niet geredirect worden; in UNIX kan dat wel. Redirection van het error device vindt plaats als er vóór het redirection teken een 2 gezet wordt.

Voorbeeld:

```
cc myprog.c 2> errors.lst &
# Start cc in de background en stuur eventuele errors naar "errors.lst"
```

Pipelines en filters

Soms kan het handig zijn de uitvoer van een bepaald commando te gebruiken als invoer voor een ander commando. Daarvoor kan gebruik gemaakt worden van een temporary file zoals in het volgende voorbeeld:

```
ls > /tmp/file ; wc < /tmp/file ; rm /tmp/file
# Tel files in de current subdirectory
```

Tussen de bedrijven door: in dit voorbeeld staan twee onafhankelijke commando's op dezelfde regel. Dat kan in UNIX als men er een punt-komma tussen in zet!

Veel handiger zou het zijn als de standaard output van het ene proces direct als invoer van het volgende proces gebruikt kon worden. In UNIX kan daarvoor gebruik gemaakt worden van een zogenaamde "pipe". Een pipe zorgt er dus voor dat de standard output van het ene programma gekoppeld wordt aan

de standaard input van een ander. Het symbool dat daarvoor gebruikt wordt is de |. Het bovenstaande voorbeeld kan met behulp van een pipe veel korter omschreven worden:

```
ls | wc
```

In UNIX wordt er veel gebruik gemaakt van pipes. Er zijn veel speciale programma's die speciaal van standard input lezen. Voorbeelden zijn o.a. "pg" of "more", "grep" en "wc". Een pipeline kan meerdere malen achter elkaar geleged worden. Bijvoorbeeld:

```
ls -l | grep "drwx" | more
# List alle subdirectories en laat de uitvoer per scherm zien
```

Wildcards

Naast redirecting en pipes zorgt de shell ook voor het verwerken van wildcards. Geldige wildcard characters zijn de asterix, het vraagteken en brackets. De asterix wordt geïntepreteerd als iedere willekeurige string, inclusief een lege string. Het vraagteken duidt op één willekeurig character. De brackets kunnen gebruikt worden om verzamelingen aan te geven. De verschillende elementen worden van elkaar gescheiden door komma's. Er kan ook een deelverzameling opgegeven worden tussen brackets. Het eerste character en het laatste van de reeks worden daartoe tussen brackets gezet en van elkaar gescheiden met behulp van een streepje.

Voorbeeld:

```
ls [c-e]*
# Geeft alle files aan die beginnen met een c, d of e.
```

Voor UNIX is een punt een "normaal" character. Wordt in MS-DOS de punt gebruikt als seperator tussen naam en extensie van de file, voor UNIX maakt de punt niets uit voor de filenaam. De expressie "*.*" duidt in MS-DOS juist alle files aan; in UNIX geeft het alle files aan waar een punt in de filenaam voorkomt! Er is echter één uitzondering op deze regel. Files die beginnen met punt moeten expliciet gespecificeerd worden. De expressie "echo *" geeft dus alle files aan waarvan de naam *niet* met een punt begint. "echo .*" geeft alleen de files die beginnen met een punt!

Helaas wordt van deze wildcard characters afgeweken als het gaat om expressies voor bijvoorbeeld zoeken en vervangen in de meeste UNIX-editors. Maar daarover wellicht meer in een volgende aflevering in deze serie.

Quoting

Characters met een speciale betekenis zoals < > * | en &, worden aangeduid met de term "metacharacters". Wordt een character voorafgegaan door een backslash ("\"), dan verliest het zijn speciale betekenis. Wil men bijvoorbeeld een vraag stellen in een shellscript, dan moet het vraagteken voorafgegaan worden door een backslash.

Bijvoorbeeld:

```
echo Uw naam \?
# Stelt de vraag "Uw naam?"
```

De backslash is op zich ook weer een metacharacter. Wil men de backslash gebruiken, dan kan dit character ook weer van zijn speciale betekenis ontdaan worden. Dus: "echo \\" zal slechts één backslash laten zien.

Het gebruik van de backslash is natuurlijk niet handig voor meer dan een zeer beperkt aantal characters. Willen we bijvoorbeeld de string "**** Forced logout from system" op het scherm zetten, dan zouden alle vier asterix' voorafgegaan moeten worden door een backslash. Gelukkig biedt de shell ook de mogelijkheid stukken tekst te quoten, waardoor de tekst tussen de quotes niet geïntrepreteerd wordt.

Bijvoorbeeld:

```
echo '**** Forced logout from system'
# Zet de tekst "**** Forced logout ..." op het scherm
```

De dubbele quote heeft ook een speciale betekenis in UNIX. Metacharacters in strings, omgeven door dubbele quotes, worden gedeeltelijk van hun speciale betekenis ontdaan. De metacharacters voor parameter substitutie, command substitutie en backslash houden hun metacharacter functie.

Prompting

De uitspraak "UNIX is not a user friendly operating system; it merely requires a friendly user" krijgt pas echt betekenis als we kijken naar de shell interface. De shell meldt zich met behulp van een prompt en staat vervolgens vrolijk op invoer te wachten. Onder UNIX zijn er drie verschillende prompts gangbaar. De eerste standaardprompt bestaat uit een dollarteken. Normale users zullen meestal met deze prompt geconfronteerd worden. De twee mogelijkheid is het matje ("#"). De super-user (root) krijgt deze prompt voor z'n neus in plaats van het dollarteken. Deze twee prompts worden ook wel aangeduid met de term "primary prompt".

In UNIX kan een commando ook over meerdere regels verdeeld worden. De shell geeft aan dat een commando nog niet afgesloten is met behulp van de "secondary prompt". Deze prompt staat standaard ingesteld als een groter-dan teken.

Beide prompts kunnen anders ingesteld worden. Ze zijn opgeslagen in de environment variabelen met respectievelijk de namen "PS1" en "PS2". Ze kunnen anders ingesteld worden met behulp van de volgende opdrachten:

PS1 = 'Command: '

PS2 = 'More: '

De huidige settings kunnen bekeken worden met de set-opdracht: "set" geeft een overzicht van de environment variabelen.

Shell scripts

Een script of shell procedure is vergelijkbaar met een batchfile onder MS-DOS. Een script bestaat uit een opeenvolging van commando's in één file. Om een shell uit te voeren zijn drie verschillende methodes beschikbaar. Men kan gewoon de shell opnieuw opstarten, geredirect vanaf een command file. De shell accepteert echter ook een scriptfile als parameter; de opdracht "sh go" voert het shellscript met de naam "go" uit. De derde mogelijkheid is het zetten van het executable vlaggetje van het shellscript. Voor een shellscript met de naam "go" kan dat met behulp van de opdracht "chmod +x go". Is het executable bitje van een script gezet, dan hoeft er niet expliciet een shell opgestart te worden. Is het x-bitje van de file "go" gezet, dan kan het shellscript met z'n eigen naam opgestart worden: "go".

Parameters en variabelen

aan een shellscript kunnen parameters meegegeven worden. In het shellscript zijn de parameters opvraagbaar als $1, $2, $3 et cetera tot en met $9. De naam van het shellscript is opvraagbaar als $0. Als er meer parameters beschikbaar zijn, kan het rijtje parameters doorgeschoven worden met behulp van het shift-commando. Stel dat $1 de waarde "Hallo" heeft en $2 de waarde "world!". Na het commando "shift" heeft $1 de waarde "world!" gekregen.

Naast deze parameters zijn er nog heel wat speciale variabelen beschikbaar in het shellscript. Ten eerste natuurlijk alle variabelen die in de environment gezet kunnen worden. De belangrijkste zijn HOME, PATH, PS1, PS2 en IFS. Als er een mail systeem draait kan ook de variabele MAIL nog een speciale betekenis hebben. De waarde van een variabele wordt aangeduid door er een $ voor te zetten. De waarde van HOME is dus op te vragen als $HOME. Het setten van een waarde in een variabele kan op de volgende wijze:

VAR = waarde

Een aantal speciale variabelen zijn niet opgenomen in de environment. Hieronder volgt een rijtje met de belangrijkste variabelen die speciaal in een shellscript aardig gebruikt kunnen worden:

$?	De return code van het laatst uitgevoerde commando.
$#	Het aantal parameters, opgegeven aan de shell.
$$	Het proces nummer van de huidige shell (in decimaal). Deze variabele wordt meestal gebruikt voor het genereren van unieke filenamen.
$!	Het procesnummer van het laatste proces dat in de background opgestart is.
$-	De huidige shell vlaggetjes. Hun betekenis komt in een volgende aflevering aan bod.

De shell heeft nog een hele hoop mogelijkheden meer te bieden. In de volgende aflevering zal ik het gaan hebben over o.a. flow-control, here-documents, debugging, parameter substitution en -transmission, command substitutie en nog veel meer zaken die allemaal direct of indirect met de shell te maken hebben. Alles bij elkaar genomen is het wellicht handiger nu even stap op de plaats te maken en een overzicht te geven van een aantal standaard utilities van UNIX. Ik geeft slechts een klein lijstje van de beschikbare voorraad, met een zeer korte omschrijving erbij. De meeste commando's hebben nog reeksen vlaggetjes te beschikking die ik ook niet allemaal ga noemen. Een goed UNIX boek kan zeker verder helpen...

Deze lijst is zeker niet compleet. Een aantal commando's ontbreken; een aantal staan er ook tussen waarvan menigeen zich af zal vragen wáárom.

cat	Voegt de inhoud van de files samen en stuurt ze naar standard output.
chmod	Verandert de file protectie van de gespecificeerde file(s).
cp	Copieert file(s).
date	Toont datum en tijd.
df	Geeft weer hoeveel ruimte er (nog) beschikbaar is op disk.
du	Geeft weer hoe het met het diskgebruik gesteld is. Dit commando geeft andere informatie dan df.
echo	Print de argumenten.
grep	Zoekt naar een patroon en drukt de regels af waarin het ge-

	specificeerde patroon gevonden is. Dit is een commando dat zeer veel in scriptfiles voorkomt!
head	Drukt de eerste xx regels af van een file.
kill	Stuur een signal naar een proces.
ln	Creëert een extra file entry naar een bestaande file of directory.
lpr	Stuurt een file naar de printer.
ls	Geeft de inhoud van een directory.
mail	Stuur en/of lees post van/aan andere users.
man	Geeft informatie uit de online manual.
mkdir	Maakt een nieuwe (sub-) directory.
more	Drukt een document af per scherm op standaard uitvoer.
mv	Verhuist en/of geeft een nieuwe naam aan de gespecificeerde file(s).
passwd	Verander het login password.
pr	Deel een document op in pagina's om naar de printer te sturen.
ps	Geeft een overzicht van de lopende processen.
pwd	"Print working directory". Geeft de naam van de huidige directory.
rm	Verwijder de opgegeven file(s).
rmdir	Verwijdert de opgegeven (sub-)directorie(s).
sh	De shell.
sleep	Wacht het opgegeven aantal seconden. Typisch voor het gebruik in bepaalde shell scripts.
sort	Sorteer een ASCII-file.
split	Verdeel één grote file in diverse kleinere.
stty	Set en/of bekijk terminal parameters.
tail	Drukt de laatste xx regels af van een file.
tee	Stuurt standaard invoer naar standard uitvoer en de gespecificeerde file(s).
test	Voert tests uit voor o.a. verschillende filetypen. Meestal gebruikt in samenhang met if-then statements in shellscripts.
time	Geeft weer hoeveel tijd een proces nodig heeft gehad.
wc	Telt characters, woorden en regels in een file.
who	Geeft een lijstje van alle ingelogde users.
write	Stuur een message naar een andere on-line user.

Literatuur:

1: S.R. Bourne, An Introduction To The UNIX Shell, Bell Laboratories

2: Dr. Ir. K.L. Boon & Ir. A.R.Th. Pelsmaeker, UNIX compact Gids, Addison-Wesley; OMIKRON (ISBN 90-6789-020-0)

3: G.J.M. Austen, UNIX: Het standaard operating system, Academic Service (ISBN 90-6233-217-X)

Joost Voorhaar

Erratum

Bij het stukje over UNIX is in de vorige μP Kenner een verkeerde illustratie geplaatst. De geplaatste illustratie toont de "stamboom" van UNIX, terwijl de nevenstaande tekening een standaard directory structure voorstelt.

Joost Voorhaar

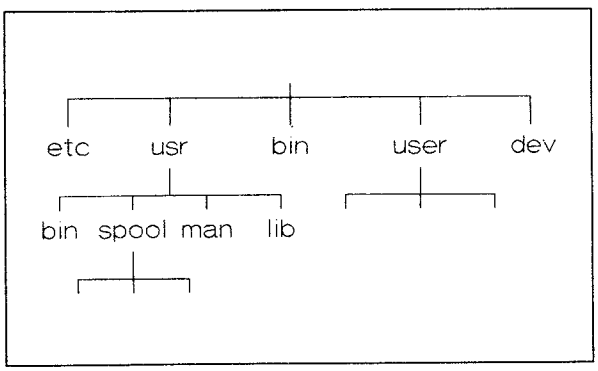
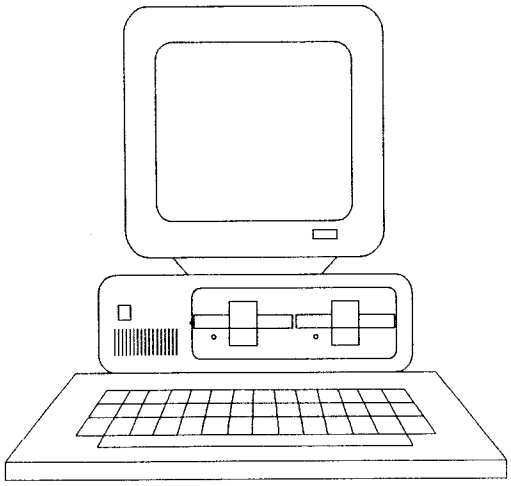


Fig. 1: directory-structuur van de meeste UNIX systemen

De IBM-PC en z'n klonen (Deel 12)

In het laatste deel bent u onledig gehouden over het BIOS van de IBM PC/AT. We zagen dat dat BIOS in de eerste plaats verregaand opwaarts compatibel was met de BIOS uit de PC(/XT). Verder bleek het BIOS zo ongeveer uitgebreid met alles wat een normaal gezond denkend mens er nog bij kon verzinnen. IBM had zijn huiswerk voor de AT goed gedaan. Je moet de bugs en andere ontwerpfouten met een gigantische zaklantaarn gaan zoeken, en dan vind je er nog nauwelijks. De machine werd daarom een succes. Op het kantoor zelfs een groot succes. Dit ondanks de in het begin wat magere clocksnelheid van 6 MHz. IBM maakte dat een beetje goed met de AT/339: die liep op 8 MHz. Doch de techniek schrijdt voort. Er verschenen 10, 12.5 en zelfs 16 MHz versies van de 80286 processor. De geheugens werden sneller. IBM leek te slapen. Zij was bezig met het ontwikkelen van de nooit echt aangeslagen reeks van PS/2 machines. Intussen zaten de nijvere werkertjes in het verre Oosten niet stil. Aanzienlijk minder stil zelfs dan ze bij de PC en de PC/XT hadden gedaan...



AT-klonen

Na het verschijnen van de oer-AT duurde het maar heel even, of de klonenbouwers sloegen toe. Eerst weer met ongeveer-kopieën. Rechtstreekse kopieën waren er natuurlijk ook, maar sst..., niet tegen IBM zeggen hoor! Het duurde eveneens niet zolang of er kwamen 8 MHz 1 wait state moederborden op de markt. Met een wat uitgekiende timing kun je op zo'n bord nog steeds 150 ns DRAMs gebruiken.

Intussen zaten de nijvere werkertjes in het verre Oosten niet stil.

Wat wel direct bij alle klonen gold, is dat er 640 kbyte of soms ook 1 Mbyte RAM op het moederbord kon. Al die kloon-moederborden waren net zo groot als het voorbeeld van IBM. Dat bleef zo tot ongeveer 1986. Toen waren de klonenbouwers toe aan moederborden die 10 MHz, zero wait state liepen. En dat is tweemaal zo snel als de oorspronkelijke IBM AT.

Rond die tijd verscheen zoals in deel 10 al even aangeroerd een gedeeltelijk geïntegreerde AT, in de vorm van 5 chips van de fabrikant Chip en Technologies. De klonenbouwers reageerden met een nieuw soort moederborden: de baby-AT. Een baby-AT moederbord is net zo breed als een PC(/XT) moederbord, en is ongeveer 3 centimeter langer. Dankzij deze maten kan zo'n moederbord zowel in een PC(/XT), als in een originele PC/AT kast worden ingebouwd. De chipset was in eerste instantie beperkt tot 8 MHz, 1 wait state, zodat nog een tijdje full-size moederborden werden gebouwd die op 10 MHz liepen. C&T zat niet stil, en met het verschijnen van de chipset geschikt voor 12.5 MHz, 0 wait state was het lot van het grote AT-kloon-moederbord definitief bezegeld. De huidige generatie 80286 moederborden loopt maximaal op 20 MHz, zero wait state, en heeft soms ook nog een instructiecache met supersnel statisch RAM. Daarmee worden snelheden tot 6 maal die van de IBM oer-AT gehaald. En die liep al ongeveer 4 maal zo hard als de oorspronkelijke IBM PC...

Kloon BIOSsen

Op al die kloon-borden moest natuurlijk ook een BIOS: anders heb je geen computer. De fabrikanten zorgden er wel voor, dat uiteraard het IBM-origineel vlekkeloos op hun moederborden kon draaien, maar daarmee kun je nog geen moederbord verkopen. Bij de PC zagen we eerst allerlei pruts-BIOSsen, die allesbehalve compatibel waren met het IBM-origineel. Bij de PC/XT klonen werd dat een stuk beter: compatibiliteitsproblemen traden al bijna niet meer op, en IBM ondernam niet veel om de echt compatibele BIOSsen van de markt te weren: PC's waren zo gewild dat IBM alleen de markt toch niet kon voeden.

Bij de AT-klonen trad een geheel andere ontwikkeling op: er ontstonden na korte tijd compatibele BIOSsen van een aantal fabrikanten die na enige tijd als zeer serieus werden beschouwd. Net als bij Ame-

rikaanse auto-fabrikanten kun je spreken van de grote drie: in uw AT-kloon zit zeer waarschijnlijk een BIOS van Award, Phoenix of AMI. Dat laatste staat voor American Megatrends, Inc. Een wat kleinere BIOS-fabrikant is Quadtel.

Al deze fabrikanten hebben hun hoofdkwartier in de USA, en leveren voor ieder moederbord een BIOS op maat, desgewenst met de naam van de fabrikant van het moederbord erin. In ieder BIOS staat keurig een copyright-statement, en vaak ook het volledige adres van de fabrikant. IBM reageerde niet met processen. Award, Phoenix en AMI zijn vandaag de dag de standaard BIOS-leveranciers voor de klonen bouwers. Niet alleen van voor moederborden, ook hebben ze BIOSsen voor EGA- en VGA-kaarten in het programma.

In eerste instantie hadden de kloon-BIOSsen dezelfde functionaliteit als het voorbeeld van IBM. Omdat de BASIC-ROMs om duidelijke redenen niet aanwezig waren, waren deze BIOSsen half zo groot: twee stuks 27128 EPROMs waren voldoende. Zoals we in het vorige deel al gezien hebben, leverde IBM het SETUP-programma om de CMOS-RAM in te stellen los op floppy mee. Dat was niet altijd handig. AMI komt dan ook de eer toe de eerste te zijn geweest die een BIOS heeft geleverd met ingebouwd SETUP-programma. Iedere BIOS-fabrikant heeft hierbij zo zijn eigen stijl om de SETUP gestalte te geven. Die laatste versies hebben zelfs de mogelijkheid om de parameters van de harde schijf op te geven als deze niet in de standaardlijst voorkomt.

Intel heeft in haar ogen de fout gemaakt second source licenties af te geven voor de 80286 CPU.

Met de andere helft van de BIOS adresruimte werd in eerste instantie niets gedaan. Later verscheen onder andere van AMI met een BIOS dat in de andere helft een uitgebreide machinetest had, compleet met low-level format van de harde schijf. Weer wat later kwam Chip & Technologies met hun revolutionaire NEAT chipset, die helemaal programmeerbaar was. Tegenwoordig wordt de andere adreshelft dan ook meestal gevuld met een meestal zeer uitgebreide setup voor de NEAT parameters. Diegenen die al eens gekeken hebben naar zo'n setup-programma weten, dat je zo ongeveer omkomt in de mogelijkheden met zo'n set.

Verdere ontwikkelingen

De PC/AT architectuur heeft een nog grotere vlucht genomen: het bleef niet bij grotere snelheden en meer RAM op het moederbord. Intel had namelijk de volgende processor in de 80X86 reeks klaar: de 32-bit 80386. Het kon ook niet uitblijven: er kwamen 80386 AT-compatibele moederborden op de markt. Eerst alleen weer in full size, vaak met 64 kbyte cache geheugen en ruimte voor 4 Mbyte of nog meer RAM. In het begin waren dergelijke borden niet te betalen: alleen de processor kostte al meer dan duizend gulden. Uiteindelijk draaien deze borden met duizelingwekkende snelheden: tegenwoordig kun je ze kopen met 40 MHz clock en 128k 8 ns cache geheugen. Die machines lopen met gemak honderd maal zo hard als een XT-tje. Lust u nog peultjes? En wat nog het meest vervelende is: de 80386 heeft een nog mooiere protected mode als de 80286 al had. En die mode gebruiken we nog steeds niet.

Trek je de lijn door, dan verbaast het niemand dat je nu ook al 80486 borden kun kopen, nog steeds met diezelfde AT-architectuur. Nu lijkt die 80486 wel een gloednieuwe CPU, maar als je goed hebt opgelet is het niets anders dan een 80386, een 80387 coprocessor en wat cache geheugen op een enkele chip.

Een zijpaadje wordt gevormd door de 80386SX CPU. Dat is een 16-bit 80386. Je zou zeggen dat dat hetzelfde oplevert als een 80286 maar dat is vanwege de andere protected mode niet het geval. Intel heeft in haar ogen de fout gemaakt second source licenties af te geven voor de 80286 CPU. De PC-markt werd echter zo groot dat een aantal second sources sterk onder de Intel-prijs konden leveren. Het resultaat was, dat Intel wel de ontwikkeling de CPU gedaan had, maar naar haar zin niet echt de volle vruchten van het werk kon plukken. Daarom werd met de 80386 een andere weg gevolgd: er zijn geen second source licenties verstrekt. En de 80386 heeft een uitgebreidere protected mode. Er verschenen pakketten die van die mode gebruik maken, en Intel zag zijn kans schoon: de 80386SX maakt het mogelijk die pakketten te draaien op een relatief goedkope 16-bit machine, en alleen Intel kan de CPU leveren, en dus de prijs bepalen. En dus zijn er ook 80386SX moederborden, meestal iets duurder dan een 80286 bord met dezelfde specificaties.

Het laatste nieuws op dit gebied is, dat AMD de 80386 heeft nagemaakt, op dezelfde manier als NEC heeft gedaan met de 8088 en de V20. Intel begon (natuurlijk) een proces tegen AMD wegens ontwerp-diefstal. AMD heeft gewonnen, dus is er binnenkort een (goedkoper?) alternatief voor de 80386 op de markt. De Taiwanezen kozen een andere

oplossing: vanwege de voortdurende, door Intel bewust gecreëerde prijsopdrijvende schaarste aan 80386 CPU's, maakten ze gewoon de 80386 na in een eigen IC-fabriek. Taiwan erkent het Amerikaanse copyright toch niet. Er schijnen wel een paar luisjes in deze kopie-CPU te zitten, dat wel...

Geheugensoorten

Even over iets heel anders nu. In de vorige afleveringen hebben we gezien, dat een PC/AT 640 kbyte RAM beneden de door de 8088 real address mode grens van 1Mbyte kan hebben. De 80286 in die machine kan in protected mode echter in totaal 16 Mbyte adresseren. IBM heeft daar ook al rekening mee gehouden: het BIOS zit tweemaal in de adresmap, 1 keer bovenin de eerste Mbyte, en een keer bovenin de laatste Mbyte. Alles daar tussenin mag van IBM RAM zijn. In een AT kun je dus bijna de hele 16 Mbyte vullen met RAM als je dat wilt.

RAM dat beneden de 1 Mbyte grens woont, dus het RAM dat DOS kan gebruiken, wordt zoals reeds gemeld base memory genoemd. De rest van het RAM in een PC/AT, dat derhalve alleen in protected mode aanspreekbaar is en boven de 1 Mbyte woont heet extended memory. Maar er is nog een soort geheugen verzonnen.

Er schijnen wel een paar luisjes in deze kopie-CPU te zitten...

En wel door Lotus, Intel en Microsoft. Ofwel LIM, afgekort. Het werd het Expanded Memory System genoemd, of alweer afgekort EMS. De reden voor het maken van EMS ligt wel voor de hand. Er zijn op deze wereld altijd mensen te vinden voor wie het grootste geheugen nog veel te klein is. Meestal blijken die mensen dol op spreadsheets te zijn, vooral hele grote. Er was derhalve al spoedig een groep Lotus-123 gebruikers die erin slaagde de complete 640 kbyte RAM van een PC(/XT) vol te krijgen. Lotus begon de adresmap van de PC/XT eens te bestuderen en vond het reeds bekende gat tussen het videogeheugen en de BIOS-ROMs. Nu mogen daar volgens IBM alleen uitbreidings-ROMs wonen, maar iedereen weet zo langzamerhand wel dat dat gebied zelden of nooit helemaal vol zit met ROM. Met een beetje schipperen kun je in de buurt van adres D0000h nog wel een stuk van 64k vrije ruimte creeren. Nu zet 64k RAM niet echt zoden aan de dijk als je aan 640k al niet genoeg hebt. Dus werd die 64k een venster en werd er bankswitched RAM achter gezet. Die 64k werd overigens opgedeeld in 4 pagina's van ieder 16 kbyte.

Nu moet je ook nog aanwijzen wat er in dat venster voor RAMbank verschijnt. Dat kan bij de 80X86 CPU-serie prachtig via een I/O poort. EMS voorziet dan ook in een poort met als grootte de byte. Je kunt dus 256 blokken van 4 maal 16 kbyte aanwijzen, zodat het maximum EMS geheugen 16 Mbyte bedraagt. Dat is de hardwarekant. Er is ook nog een softwarekant: de jongens van LIM verzonnen er meteen maar een complete specificatie bij. In die specificatie staat hoe je met EMS-geheugen moet omspringen en hoe je het moet benaderen. Dat gaat heel gemakkelijk. INT 67h werd namelijk voor die taak ingehuurd. Via die interrupt kun je een complete memory manager aanroepen, die kan bijhouden welk geheugen er nog vrij is, en welke stukken door wie gebruikt worden. Want bij de toewijzing moet je ook een procesnummer (16-bits) opgeven. De eerste specificatie droeg het versienummer 3.2. Thans kennen LIM-EMS 4.0, dat ook wel LIM-EMM genoemd wordt.

Intel, Microsoft en Lotus hadden hun huiswerk goed gedaan: LIM-EMS dekte duidelijk een behoefte in de markt en is uitgegroeid tot een defacto standaard. Veel softwarepakketten kunnen gebruik maken van LIM-EMS. Soms zit de driver in een ROM, maar meestal staat hij in RAM. De zeer uitgebreide NEAT chipset van Chip & Technologies ondersteunt ook hardware EMS. Meestal wordt bij een NEAT-moederbord ook een driver meegeleverd, zodat er echt met EMS gewerkt kan worden.

Omdat EMS geheel in de onderste 1 Mbyte van de adresmap werkt, is het zoals oorspronkelijk bedoeld, ook voor PC(/XT)'s te gebruiken.

Wat later leek er nog een geheugensoort bij te komen, maar dat viel mee: XMS. Dat is niets anders als een manager voor extended geheugen op een 80286/386 machine, ongeveer op de manier van EMS. Ook in XMS wordt van ieder blok geheugen bijgehouden wie de gebruiker is en wat de toegangsrechten zijn. Het gaat nog een stapje verder: er bestaan zelfs programma's die EMS kunnen emuleren met behulp van extended memory. Kortom, als er RAM over is in de machine kun je er vrijwel altijd iets mee doen.

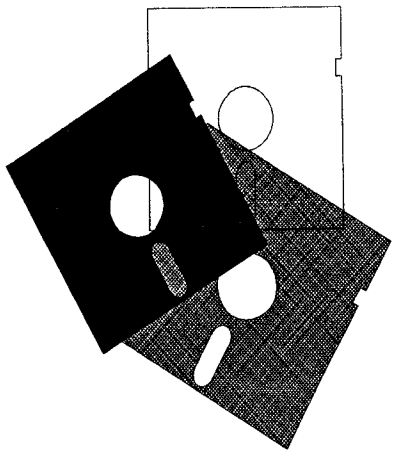
De volgende keer...

besteden we wat aandacht aan de VGA-kaart. En misschien nog wel wat. Ik begin een beetje door mijn stof heen te raken. Heeft u iets gemist? Heb ik iets niet genoemd? Laat het eens weten.

Nico de Vries

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

Wat voor computer men ook bezit, iedereen heeft wel eens gemerkt dat harddisks de vreemde eigenschap bezitten altijd te klein te zijn. Telefoonkosten zijn altijd te hoog en floppen zijn er nooit genoeg in voorraad. Of de moderne software pakketten zijn te groot, dat kan ook natuurlijk. Op de keper beschouwd is ieder pakket te groot; er zit altijd een hoop redundante informatie in ieder stuk software. Gelukkig zijn er methoden om die informatie samen te pakken en te comprimeren: de beroemde (en beruchte) packers, arcers en zippers en al wat dies meer zij. In het woud van compressing software is geen afzonderlijke boom meer te zien. De hoogste tijd om eens flink te inventariseren en reorganiseren...



Voor de populairste computers zijn een aantal "standaard" compressie- en archive utilities ontstaan die min of meer systeemonafhankelijk zijn. Een selectie van de meest gebruikte archivers leverde de volgende verzameling op (versie nummers zijn geldig voor PC/MS-DOS):

Software	Versie	Copyright
LHARC	1.13c	Haruyasu Yoshizaki
PAK	2.51	NoGate Consulting
PKPAK	3.61	PKWARE Inc.
PKZIP	1.10	PKWARE Inc.
ZOO	2.01	Rahul Dhesi

LHArc is een Japanse packer van Haruyasu Yoshizaki ("Yushi"). Het ding maakt files aan met de extensie ".lzh". LHArc is een van de meest verspreide archiverings utilities, een gevolg van het feit dat LHArc de naam had kleine archives te maken en de source zat er vanaf het prille begin al bij. LHArc is zeer goed verspreid onder de Amiga- en minix gebruikers.

Alle files op het SDN/Works! netwerk worden gepackt met PAK. De reden daarvoor was, dat PAK als één der eerste archiverings utilities een zogenaamde "security envelope" kon maken. Tijdens het uitpakken kan PAK controleren of de archive ongewijzigd is sinds de originele distributeur het uitgaf.

PKPAK is een van de twee archive utilities van Phill Katz. In voorgaande versies ging PKPAK als "PKARC" door het leven. PKPAK was volledig compatible met ARC van System Enhancements Associates (SEA), en de naamgeving van PKARC werd door SEA misleidend voor de consument gevonden. Na een uitgebreid proces stelde de rechter de commerciële SEA in het gelijk en sindsdien gaat PKARC als PKPAK door het leven.

De andere packer van PKWARE Inc. (lees: Phile Katz) is PKZIP. Voor een groot aantal systemen is wel een unzipper beschikbaar, maar zijn er geen zippers. Zo ook voor minix. Samen met PKPAK is PKZIP de enige utility die in een aparte packer en een unpacker gescheiden is.

Uit de UNIX wereld is ZOO afkomstig. Op sommige systemen zie je nog wel eens wat verdwaalde files met de extensie ".zoo", maar de belangstelling voor dit produkt is duidelijk tanende. Van ZOO zijn complete sources beschikbaar, zoals te doen gebruikelijk in de UNIX wereld.

De vergelijkende test

Om wat zinnigs te kunnen zeggen over de betreffende utilities moeten ze uiteraard op een verantwoorde wijze getest worden. Tekst files hebben een veel homogenere verdeling van de afzonderlijke characters dan bijvoorbeeld een executable. Als een utility iedere file afzonderlijk in elkaar drukt en daarna pas in de archive propt is het resultaat uiteraard ook weer heel anders dan als alle files bij elkaar gezet worden en daarna pas gecomprimeerd. Daarom is het moeilijk om een echt zinnige test te bedenken.

Om een beeld te geven van de sterke en zwakke kanten van de diverse programma's hebben alle utilities drie testen gelopen waarbij gelet werd op de snelheid zowel als op de grootte van de resulterende file. De eerste test bestond uit het archiveren van de technische documentatie van het fidonet. In totaal is dat 334 kByte aan informatie, verpakt in tien afzonderlijke tekstfiles, variërend in grootte van ca. 2 kByte tot ruim 92 kByte per stuk.

De tweede opgave is er één uit de directe praktijk: de distributie van het communicatie pakket Telix

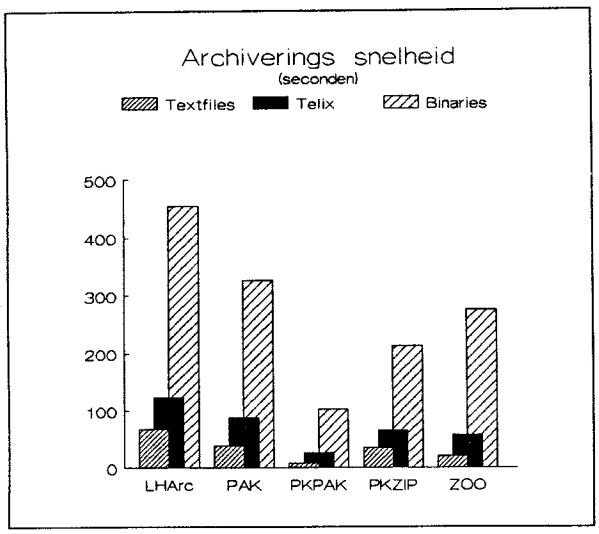


Fig. 1: archiveringstijd van de diverse archivers

(versie 3.12), in totaal 780 kByte waarvan 316 kByte aan executables.

De derde test is een soort bulk-test waarin de utility directory van het test-systeem "even" gearchiveerd moest worden. Deze directory bevat maar liefst 115, meest wat kleinere executables met hier en daar een grote (binair gecodeerde) helpfile. Deze opgave is goed voor ruim 2720 kByte (ca. 2.65 MByte) aan testmateriaal dat in één archive gecomprimeerd dient te worden. Er werd alleen gekeken naar de pack-resultaten; unpacken van de directory was niet mogelijk door een nijpend gebrek aan diskruimte; er zou dan een tweede copie van de originele directory gefabriceerd worden en die 4 MByte extra is helaas niet over...

Resultaten

De resultaten zijn samengevat in een drietal grafiekjes die elk de snelheid en mate van compressie weergeven. De meeste compressie utilities melden zeer trots hoeveel ze van de originele files af hebben gehaald; de grafieken geven weer hoeveel van de files overgebleven is. Op deze manier geven kortere staven zowel bij snelheid als grootte een betere prestatie weer.

Als snelheid een overwegende rol speelt is het wellicht interessant eens een blik te werpen op PKPAK (zie figuur 1). Vooral als men te maken heeft met een situatie waarin men veel betaalt voor een aansluiting op een remote systeem en een pakket over wil halen naar het eigen systeem waarvoor remote gepackt moet worden. De snelheid van PKPAK kom het verschil in extra transfertijd die een gevolg is van de grotere omvang van het pakket wel eens helemaal teniet doen. Zeker als er een snelle verbinding (een modemverbinding op 9600 bps...) gebruikt wordt en het remote systeem niet al te snel is. Een tip voor de XRS-ers: bekijk eens de totale tijd die je nodig hebt om een mailbag in te pakken en te versturen als je PKPAK gebruikt en vergelijk die met de resultaten van andere archivers.

Op het testsysteem staat heel wat technische documentatie die regelmatig geraadpleegd moet worden. Deze documentatie is in compressed vorm opgeslagen om ruimte te besparen. Met behulp van een alias definitie kan de documentatie uitgepakt en doorgekeken worden: zie figuur 2.

De uitpak snelheid (figuur 3) is dan een belangrijke factor om het nagelbijten in te dammen. Wat onmiddelijk opvalt is de relatief lage uitpaksnelheid van PKPAK en ZOO in vergelijking met de paktijd. PKZIP lijkt hier een goede keuze.

Snelheid bij het inpakken gaat duidelijk ten koste

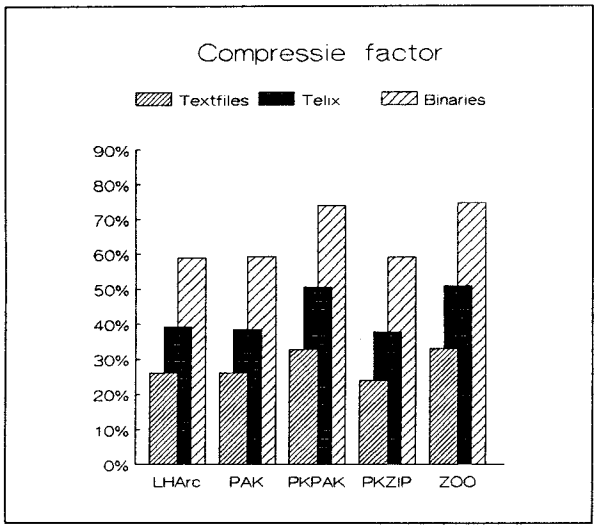


Fig. 3: effectiviteitsvergelijking

van de resulterende compressiefactor (zie figuur 4, volgende bladzijde). De minder snelle utilities ontlopen elkaar weinig op dit gebied. De verschillen in compressiefactoren zijn marginaal: een half procentje hier een kwart procentje daar; soms in het voordeel van de ene utility, soms in het voordeel van de andere.

```
alias show pkunzip -c \document\%1 %2 %3 %4 %5 | *list
```

Fig. 2: alias commando om documentatiefiles te bekijken

Documentatie

Bijna alle utilities zijn uitstekend gedocumenteerd. De meeste utilities zijn voorzien van een uitgebreide omschrijving van de mogelijke command-switches en een aantal programma's is ook voorzien van extra informatie omtrend de gebruikte methodes. LHArc is voorzien van een min of meer uitgebreide omschrijving van alle mogelijke command line opties. Verder wordt de geschiedenis van het pakketje uit de doeken gedaan. Van LHArc zijn de volledige sources beschikbaar. NoGate consulting geeft ongeveer de zelfde informatie in de documentatie van PAK. Van PAK zijn de sources (voor zover mij bekend) niet verkrijgbaar. Phill Katz geeft ongeveer dezelfde informatie in de documentatie van PKPAK en PKZIP. Daarnaast zijn de beide utilities voorzien van meer technische documentatie waarin de opbouw van de archives omschreven wordt. In de documentatie van PKZIP wordt ook omschreven hoe de (de-)compressie van archives precies in elkaar steekt. Daardoor zijn er al verscheidene versies voor andere dan MS-DOS systemen beschikbaar. De originele sources van Phill blijven echter veilig en wel achter slot en grendel. ZOO tenslotte is voorzien van een aantal losse tekstfiles die in een algemene UNIX-stijl het pakket omschrijven. De documentatie van ZOO is pover, zeker in vergelijking met de tientallen kilobytes aan documentatie van de andere pakketten.

Betrouwbaarheid

Er is nog één facet geheel onbesproken gelaten: de betrouwbaarheid van de verschillende packers. De stabiliteit van de pakketten is moeilijk te bepalen; tijdens de test leverde geen der utilities beschadigde archives op. De kinderziekten van PKZIP lijken opgelost; er waren in voorgaande versies problemen met archives waar veel files in opgeslagen worden.

Sommige utilities bieden mogelijkheden gedeeltelijk beschadigde te redden. In de praktijk blijken deze

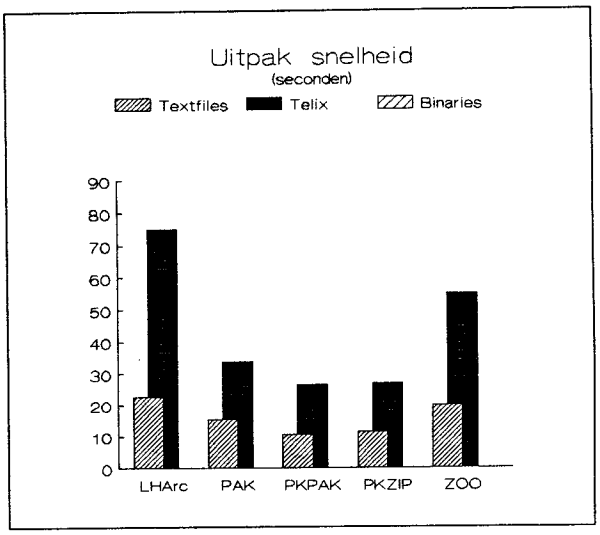


Fig. 4: vergelijking van de uitpak snelheden

"recover" faciliteiten echter niet of nauwelijks bruikbaar te zijn; een beschadigde archive is niet meer te herstellen en de inhoud mag worden beschouwd als "verloren gegaan".

Conclusie

De conclusie van dit verhaal bevestigt slechts wat door velen zonder gegronde reden werd aangenomen: PKZIP is "the way to go" als het om MS/PC-DOS gaat, in vrijwel alle andere gevallen verdient LHArc de voorkeur.

De ontwikkelingen gaan snel en het is best mogelijk dat de situatie wellicht al weer veranderd is. Nieuwe versies van bestaande programmatuur of geheel nieuwe programma's crunchen, imploden en shrinken tóch weer sneller en beter dan het programma van de concurrent. Toch maakt PKZIP naar mijn idee de beste kansen op een gezonde toekomst...

Joost Voorhaar

Product	LHArc	PAK	PKPAK	PKZIP	ZOO
	1.13c	2.51	3.61	1.10	2.01
Registratiekosten	0.00	$20.00	$20 of $45	$25 of $47	0.00
Password	Nee	Ja	Ja	Ja	Nee
Security envelope	Nee	Ja	Ja	Ja	Nee
Config file	Nee	Ja	Ja	Ja	Nee
Pathnames	Ja	Ja	Ja	Ja	Ja
Attributes	Ja	Nee	Nee	Ja	Ja
Archive comment	Nee	Ja	Ja	Ja	Nee
File comments	Nee	Ja	Ja	Ja	Ja
Extract to stdout	Ja	Ja	Ja	Ja	Ja
Errorcodes	Ja	Ja	Ja	Ja	Nee
Self extractors	Ja	Ja	Ja	Ja	Nee
Recover faciliteiten	Nee	Nee	Nee	Ja	Ja
Sources	Ja	Nee	Nee	Nee	Ja
Filename	LH113C.COM	PAK251.EXE	PK361.EXE	PKZ110.EXE	ZOO201.EXE
Grootte	36963	103338	119808	149504	82486

Fig. 5: vergelijking tussen de diverse archivers

Het programmeren van de 8088 in de IBM (Deel 4)

Niet zoveel tekst deze keer. Ik vrees dat de listing van het programma bij deze aflevering al een voldoende grote aanslag pleegt op de beschikbare pagina's in deze μP-Kenner.

Invoer en uitvoer buiten DOS om

Door de I/O van een programma altijd via de daarvoor bestemde DOS functies te laten lopen, bereiken we dat ons programma op elke machine waarop MS-DOS als operating system draait, zal functioneren. Helaas kleven er aan deze methode ook een aantal nadelen. DOS is niet alleen traag, maar er ontbreken ook nogal wat mogelijkheden, zoals het wissen van het beeldscherm, het positioneren van de cursor en het kiezen van de gewenste kleuren op een systeem dat in kleur kan werken. Daarvoor kan men dan de speciale driver ANSI.SYS installeren, maar die maakt alles alleen nog een beetje trager. Het kan allemaal veel sneller en veel mooier als we gebruik maken van het BIOS of rechtstreeks in het videogeheugen schrijven. We moeten er dan wel rekening mee houden dat dergelijke software machinetype afhankelijk is. De meeste machines waarop MS-DOS draait zijn tegenwoordig echter compatible met de IBM systemen, zodat dit niet zo'n groot bezwaar behoeft te zijn. Er zitten echter ook nog andere addertjes onder het gras:

- Lang niet alle bestaande systemen beschikken over dezelfde faciliteiten in het BIOS. Wie dus "optimaal" gebruik gaat maken van de mogelijkheden die het BIOS in zijn prachtige systeem heeft, produceert gegarandeerd iets dat op andere systemen niet meer werkt. In het algemeen zijn alleen functies waarover ook de oer-PC beschikt bruikbaar.
- De meeste netwerken verwachten I/O via DOS. In het algemeen is software die van deze technieken gebruik maakt dan ook beslist ongeschikt voor gebruik binnen een netwerkt. Als een dergelijk programma "remote" wordt ge-executeerd is men aan de andere kant gegarandeerd elke controle over de gang van zaken kwijt.
- Omleiding van de uitvoer is niet meer mogelijk.

Nog eens HEXDUMP maar nu via het BIOS

Laten we nu het programma "HEXDUMP" uit de vorige aflevering eens voorzien van uitvoer via het BIOS. We behoeven daartoe alleen maar de routine CHAROUT te herschrijven. In figuur 1 is te zien wat er voor het stukje programma vanaf het label CHAROUT in de plaats komt. Functie 0EH van BIOS interrupt 10H drukt het teken dat in register AL staat af op het beeldscherm. Daarbij worden speciale tekens zoals CR en LF alsmede Bell en Back Space keurig netjes geïnterpreteerd. Ook zorgt

```
CHAROUT:  PUSH   BX          ;Laat BX ongemoeid!
          MOV    AH,0EH      ;Functie 0EH van INT 10H
          XOR    BH,BH       ;voor schermpagina 0
          CALL   INTR10      ;drukt het teken in AL af
          POP    BX
          RET
WRWORD    ENDP

          ;Aanroep van BIOS interrupt 10H
          ;Hierbij gaan SI, DI en ES verloren!

INTR10    PROC   NEAR
          PUSH   SI          ;De nodige registers bewaren
          PUSH   DI
          PUSH   ES
          INT    10H         ;Interrupt aanroepen
          POP    ES          ;en de registers weer terughalen
          POP    DI
          POP    SI
          RET
INTR10    ENDP
```

Fig. 1: BIOS-aanroep veranderingen in HEXDUMP.ASM

de routine ervoor dat het scherm een regel omhoog rolt als de laatste regel van het scherm is bereikt. In register BH moeten we aangeven op welke schermpagina zal worden afgedrukt. Normaal is dat pagina 0. Niet alle systemen beschikken over meerdere pagina's. Vergeet ook niet het BX register te bewaren. Daarin staat immers de handle van de open file!

Na deze wijziging lijkt het programma nog precies hetzelfde te werken. Alleen is het merkbaar sneller geworden. Doch als u probeert om de uitvoer met bijvoorbeeld >PRN naar de printer te sturen, zult u merken dat dat niet meer mogelijk is. Ook worden geen TAB's ondersteund. Het gebruik van het BIOS is echter een vlotte methode in programma's die toch uitsluitend voor lokaal gebruik zijn.

Tekst zoeken in een tekstfile

Het programma van figuur 2 doet ongeveer zoiets als de DOS commando's FIND en TYPE. Waar het om gaat is dat we hier kunnen zien hoe we regel voor regel gebufferd een tekstfile kunnen lezen en afdrukken. De bedoeling is hier dat elke regel wordt onderzocht op aanwezigheid van een bepaalde string en dat telkens als deze wordt gevonden het programma even op een toetsdruk wacht. Er zijn dus twee parameters nodig: De naam van de file en de op te zoeken string. Om het niet te ingewikkeld te maken gaan we er van uit dat in de tekst file elke regel met een CRLF combinatie eindigd. Het programma demonstreert de kracht van twee stringinstructies die we nog niet eerder zagen. De eerste is SCAS (SCAn String). Deze instructie tast de string op ES:DI met een lengte van CX bytes af naar het teken in AL. We gebruiken hem om de spatie tussen de parameters te vinden en om de extra spaties over te slaan. Denk er om dat DI indien we vonden wat we zochten altijd een plaats voorbij het gezochte teken staat!

De andere rappe jongen gebruiken we om de ingelezen regel af te zoeken naar de string van de tweede parameter. CMPS vergelijkt twee strings ter lengte van CX tekens in een keer! De een wordt aangewezen met ES:DI en de andere met DS:SI. Als de string werd gevonden wachten we even op een toetsdruk.

Merk op dat omleiding van de uitvoer niet erg zinvol is als die naar een disk file wordt gestuurd. We zitten dan immers met die toetsdruk! Volgende keer zullen we dit programma dan, zonder de hele listing hier af te drukken, omwerken tot een "DOS filter". Filters zijn over het algemeen erg gemakkelijk in assembler te schrijven.

Ruud Uphoff

Fig. 2: het programma FINDS in assembler

```
                NAME    FINDS_ASM
                ;
DISPHAND        EQU     2               ;Bekenden uit de vorige aflevering
CR              EQU     13
LF              EQU     10

OFS             EQU     OFFSET

JMPS            MACRO TARGET
                JMP     SHORT TARGET
                ENDM

BUFSIZE         EQU     512             ;Omvang van het buffer.

                ;---------- DATASEGMENT ----------

DATA            SEGMENT

MASKRL          DB      0               ;Het lengte byte
MASKR           DB      255 DUP (?)     ;van de string die we willebn zoeken
HANDLE          DW      0               ;Opslag voor de file handle
PSPSEG          DW      (?)
BUFFER          DB      BUFSIZE DUP (?) ;Het file buffer
```

```
BUFPTR    DW    0                ;De leesindex in dat buffer
BUFEND    DW    0                ;en de positie van het laatste byte
LINEL     DB    0                ;Deze string is de regel die we..
LINE      DB    255 DUP (?)      ;.. aan het verwerken zijn.

ERRMSG    DB    ENDMSG-ERRMSG-1
          DB    'Bad or missing parameters',CR,LF
ENDMSG    EQU   $

DATA      ENDS

          ;---------- CODESEGMENT ----------

CODE      SEGMENT

          ASSUME               CS:CODE, DS:DATA, SS:PGMSTACK

GETFILE   PROC  NEAR
          PUSH  DS             ;Laat ES:DI naar het buffer wijzen
          POP   ES
          MOV   DI,OFS BUFFER
          PUSH  DS
          MOV   SI,80H         ;En DS:SI naar de parameters in het PSP
          MOV   AX,PSPSEG
          MOV   DS,AX
          CLD
          LODSB                ;De lengte van die string
          MOV   CL,AL
          XOR   CH,CH          ;in CX halen
UNLEAD:   STC                  ;Foutvlag gereedhouden
          JCXZ  NONAME         ;Geen parameters? Jammer dan!
          LODSB                ;Weg met voorafgaande spaties
          DEC   CX
          CMP   AL,20H
          JZ    UNLEAD
          PUSH  CX             ;De lengte in CX bewaren
          STOSB                ;Eerste teken opslaan
          REP   MOVSB          ;en de rest er achteraan
          MOV   DI,OFS BUFFER  ;ES:DI weer naar het buffer zetten
          POP   CX             ;en de lengte terug van de stack
          INC   CX             ;inclusief dat eerste teken!
          MOV   AL,20H         ;We gaan naar een spatie zoeken
          REPNE SCASB          ;En dat gaat rap met een 8088!
          STC                  ;Heel pessimistisch foutvlag zetten
          JNE   NONAME         ;Zie je wel: Geen spatie!
          DEC   DI             ;Wel gevonden? dan DI terug en erop.
          XOR   AL,AL          ;en er een ASCIIZ nul overheen zetten
          STOSB
          MOV   AL,20H         ;Weer een spatie in AL
          REPE  SCASB          ;Nu om spaties over te slaan
          DEC   DI             ;Denk hier dus altijd om!
          JZ    NONAME         ;Helaas: alleen maar spaties
          CLC                  ;Niet fout dus
NONAME:   POP   DS             ;DS terughalen
          JC    NOSECPAR       ;Waren we heer met CF = 1 dan er uit!
          MOV   MASKRL,CL      ;Lengte van de 2e parameter opslaan
          MOV   SI,DI          ;DS:DI naar 2e parameter in buffer
```

```
                MOV     DI,OFS MASKR    ;en die naar zijn plekje brengen
                REP     MOVSB
NOSECPAR:       RET
GETFILE         ENDP

                ;De file openen

FILEOPEN        PROC    NEAR
                MOV     DX,OFS BUFFER   ;DS:DX op de naam richten
                MOV     AX,3D00H        ;en file openen voor lezen
                INT     21H
                JNC     OPOK            ;Indien CF=0 dan is het gelukt
                JMP     ERREXIT         ;anders afbreken met foutmelding
OPOK:           MOV     HANDLE,AX       ;De handle opslaan
                MOV     BUFPTR,0        ;Buffer ptr op nul zetten
                MOV     BUFEND,0        ;en het buffer is leeg
                RET
FILEOPEN        ENDP

                ;Teken uit de file lezen

READBYT         PROC    NEAR
                MOV     BX,BUFPTR       ;Buffer ptr in BX halen
                CMP     BX,BUFEND       ;Einde buffer?
                JB      PICKBYT

                MOV     BX,HANDLE       ;Dan buffer eerst vullen
                MOV     CX,BUFSIZE      ;met maximaal BUFSIZE bytes
                MOV     DX,OFS BUFFER
                MOV     AH,3FH
                INT     21H
                JNC     BLOCKOK
                JMP     ERREXIT         ;Jammer van die leesfout..

BLOCKOK:        XOR     BX,BX           ;Na lezen blok pointer op 0 zetten
                MOV     BUFPTR,BX
                MOV     BUFEND,AX       ;en aantal bytes in buffer opslaan
                OR      AX,AX           ;Niets gelezen?
                JZ      ENDFILE         ;Oeps! Dat is einde file.

PICKBYT:        MOV     AL,BUFFER[BX]   ;Teken uit buffer in AL halen
                INC     BUFPTR          ;en pointer er voorbij zetten
                CMP     AL,1AH          ;Geen CTRL-Z
                JNE     MORE            ;dan er uit met ZF=0

ENDFILE:        MOV     BX,HANDLE       ;anders de file sluiten
                MOV     AH,3EH
                INT     21H
                MOV     AL,1AH          ;CTRL-Z laten zien
                CMP     AL,AL           ;en ZF=1 teruggeven

MORE:           RET
READBYT         ENDP

                ;Regel in buffer lezen

READLN          PROC    NEAR
```

```
                MOV     DI,OFS LINE
                PUSH    DS
                POP     ES
                CLD
                MOV     LINEL,0         ;Begin met lengte 0
MORESTR:        CALL    READBYT         ;Byte uit de file lezen
                JZ      LINEDN          ;Einde file? Dus ook einde regel
                STOSB
                INC     LINEL
                CMP     AL,0AH          ;LF?
                JNE     MORESTR         ;Nee, dus nog even door gaan

LINEDN:         PUSH    AX              ;Laatste teken bewaren
                MOV     DX,OFS LINE     ;DS:DX op regel laten wijzen
                MOV     CL,LINEL        ;en lengte in CX
                XOR     CH,CH
                CALL    WRITESTR        ;Regel schrijven
                CALL    SEARCH          ;Staat de gezochte string hier?
                POP     AX              ;Terug met laatste teken in AL
                RET
READLN          ENDP

                ;Regel beoordelen

SEARCH          PROC    NEAR
                PUSH    DS              ;ES = DS zetten voor string compare
                POP     ES
                CLD
                MOV     SI,OFS LINE     ;DS:SI wijst naar de regel in buffer
                MOV     CL,LINEL        ;Lengte van de regel
                SUB     CL,MASKRL       ;Min lengte van het masker is genoeg
                JS      ENDSRCH         ;In deze hoeven we dus niet te zoeken
                XOR     CH,CH

SLOOP:          PUSH    SI              ;Source index bewaren
                PUSH    CX              ;en loopteller bewaren
                MOV     DI,OFS MASKR    ;ES:DI naar het masker laten wijzen
                MOV     CL,MASKRL       ;en de lengte in CX
                REP     CMPSB           ;Strings vergelijken
                POP     CX              ;CX en SI weer ophalen
                POP     SI
                JZ      FOUND           ;String niet gevonden?
                INC     SI              ;dan van een teken verder proberen
                LOOP    SLOOP
                JMPS    ENDSRCH         ;Niets gevonden dus

FOUND:          MOV     AH,08H          ;Gevonden! Even op een toets wachten
                INT     21H
ENDSRCH:        RET
SEARCH          ENDP

        ;Afbreken met foumelding

ERREXIT         PROC    NEAR
                PUSH    AX              ;Foutcode bewaren
                MOV     DX,OFS ERRMSG+1 ;DS:DX op de foutmelding richten
```

```
                MOV     CL,ERRMSG       ;en lengte in CX halen
                XOR     CH,CH
                CALL    WRITESTR
                POP     AX              ;Foutcode terug in AL
                MOV     AH,4CH          ;en achterlaten als "Error level"
                INT     21H             ;Terug naar de de DOS promt dus.
ERREXIT         ENDP

                ;String afdrukken op het scherm

WRITESTR        PROC    NEAR
                MOV     BX,DISPHAND     ;Display handle in BX
                MOV     AH,40H          ;en foutmelding schrijven
                INT     21H
                RET
WRITESTR        ENDP

                ;Hoofdprogramma

FIND            PROC    NEAR
                MOV     AX,SEG DATA     ;De verplichte figuren.
                MOV     DS,AX           ; zie vorige aflevering
                MOV     PSPSEG,ES
                MOV     BX,SEG ZZZZZZZZ
                MOV     AX,ES
                SUB     BX,AX
                MOV     AH,4AH
                INT     21H
                CALL    GETFILE         ;Parameters ophalen
                JC      ERREXIT         ;en stoppen indien fout
                CALL    FILEOPEN        ;Anders file openen
MORELINES:      CALL    READLN          ;Regels lezen en afdrukken
                CMP     AL,1AH
                JNE     MORELINES       ;tot einde file
                MOV     AX,4C00H
                INT     21H
FIND            ENDP

CODE            ENDS

                ;---------- STACKSEGMENT ----------

PGMSTACK        SEGMENT                 STACK
                DW      100H DUP (?)
PGMSTACK        ENDS

                ;---------- VRIJ GEHEUGEN ----------

ZZZZZZZZ        SEGMENT
ZZZZZZZZ        ENDS
```

Voortgang KGN-68k MINIX project

Inleiding

Eén van de voorwaarden voor de goedkeuring van het KGN-68k project is reportage aan de leden. Daarom zal er in de μP Kenner vanaf nu dus iedere keer een artikel "Voortgang KGN-68k MINIX project" staan.

Eerst nog even de doelstellingen van de KGN:

- 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 toepassing.

Het KGN-68k project is een duidelijke invulling van onze doelstellingen. Door het opstarten van het project wordt er binnen de vereniging kennis vergaard over de manier waarop je tegenwoordig een computer in elkaar zet. Om deze kennis weer te verspreiden onder de leden, worden de resultaten van het project weer ter beschikking gesteld van de leden. Dat betekent dat het ontwikkelde systeem voor leden beschikbaar komt en dat het project zal leiden tot een aantal artikelen in de μP Kenner om de in het kader van dit project vergaarde kennis verder te verspreiden.

Een telefoontje naar Geert (04788-1279) is toch een kleine moeite!

De projectgroep staat trouwens altijd open voor inbreng van de leden, ideeën, suggesties. Uiteraard ook voor vragen, ook als ze gesteld worden in de trant van: "Het klinkt misschien dom, maar houden jullie ook rekening met ....?". Verder kunt u natuurlijk altijd uw mening over het project geven. De projectgroep staat ten alle tijde open voor inbreng van de leden en stelt elke bijdrage, hoe klein ook zeer op prijs. Een telefoontje naar Geert (04788-1279) is toch een kleine moeite! Hetzelfde geldt natuurlijk ook voor de rest van de vereniging; u kunt de bestuursleden en of de redactie altijd benaderen met uw vragen op- en aanmerkingen. Zo'n club als de KGN is toch een geweldige denktank en de telefoonnummers van het bestuur staan tenslotte niet voor niets in het blad afgedrukt. Uiteraard is de projectgroep ook aanspreekbaar op de bijeenkomsten. Op elke bijeenkomst zal er minstens één lid van de projectgroep als aanspreekpersoon aanwezig zijn. Bij hem kunt u dus ook met uw vragen, ideeën en opmerkingen terecht. Verder zal er op de bijeenkomsten voldoende tijd ingeruimd worden voor een discussies over het project, ook als dit onderdeel niet expliciet op het programma voor de bijeenkomst staat.

Het verspreiden van de volledige kennis kan bij de afronding van het project, volgens de planning in november 1991, beginnen. Tussentijdse inzage is natuurlijk ook mogelijk. Hierbij moeten we echter wel een beperking opleggen. Omdat het er op lijkt dat het project ook voor mensen buiten de vereniging zeer interessant is, willen we liever niet dat er teveel detail-informatie naar buiten gaat. Om deze reden is inzage in de documentatie, schema's programma's etc. zoals die tijdens de loop van het project geproduceerd worden voorbehouden aan leden van de KGN. Kortom, bent u lid dan kunt u kennis nemen van alles wat de projectgroep produceert en kunt u meepraten, bent u geen lid en wilt u een KGN-68k, dan kunt u maar het beste lid worden want het KGN-68k MINIX systeem kan alleen door leden nagebouwd worden.

Het stimuleren en ondersteunen vanuit bestuur en projectgroep zal worden waargemaakt door het ontwikkelen van een printplaat, eventuele gezamelijke component inkoop(korting), het beschikbaar stellen van geprogrammeerde logica, hulp bij bouwen en testen en een vanuit het bestuur gestimuleerde verdere ontwikkeling van hard- en software voor het systeem.

Mensen die meer willen weten over hoe de projectgroep ingericht is, kunnen het plan van aanpak voor dit project inzien. In dit document wordt de samenstelling van de projectgroep beschreven, worden de verantwoordelijkheden van bestuur en projectgroep geregeld en is vastgelegd hoe de rapportage naar bestuur en leden zal plaatsvinden. Verder staat er een indeling van het project in fasen in beschreven met voor elke fase een aantal eindresultaten. Uiteraard is er een globale planning voor het project in het plan van aanpak opgenomen.

MINIX

Waarom eigenlijk mini UNIX? Vanwege de voordelen van UNIX: wereldwijd, hardware onafhankelijk en de zee aan software. Verder MINIX omdat de source beschikbaar is en omdat het draait op betaalbare machines te weten: MS-DOS, Amiga, Apple MacIntosh & Atari. Dit Operating System wordt trouwens in het onderwijs gebruikt bij het practicum van lessen over Operating Systemen. Toch ideaal om mee te hobbyen! En het werkt ook. De huidige MI-

NIX versie, 1.5, kost $169 en een prijsverlaging is niet uitgesloten. 1 punt 5 is het platform c.q. de standaard die op alle verschillende computers gelijk is en waarbij bij het schrijven van applicaties van wordt uitgegaan.

Het feit dat MINIX op veel systemen beschikbaar is, maakt dit Operating System uitermate interessant voor de KGN. Voor iedereen die zich minimaal één van de bovengenoemde computers (bijvoorbeeld een MS-DOS machine van +/- fl. 1000,--) kan veroorloven, is er wel een MINIX oplossing. Wil je goedkoop, dan kun je bijvoorbeeld uitgaan van een MS-DOS computer of een Atari ST. Wil je snel, dan koop een (duurdere) 80286 of 80386 machine. Wil je zelfbouw en snel, dan kun je overwegen mee te doen met het KGN-68k project waarvan de hardware specifiek t.b.v. MINIX ontwikkeld wordt. Het grote voordeel is echter dat bijna alle software voor MINIX op alle machines draait omdat deze software in C-source verspreid wordt.

Zelfbouw

Waarom eigenlijk zelfbouw? We willen kennis vergaren en verspreiden van de technische kant van computers. We bieden onze leden daarom dan ook een systeem aan, inclusief schema's, listings van de bootrom, de programmeerbare logica en de sources van de ontwikkelde software om MINIX draaiende te krijgen. Verder hebben de leden inbreng bij het ontwerp en toegang tot de ontwerpers. Kortom, zelfbouw is de beste mogelijkheid een machine van haver tot gort te leren kennen.

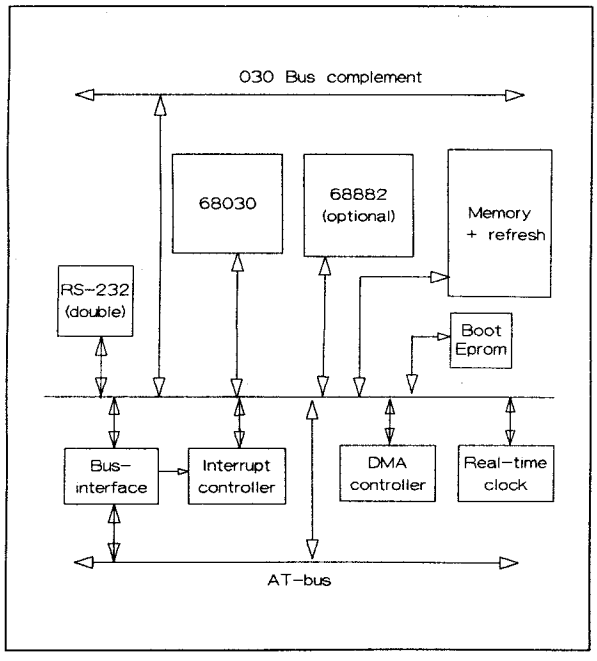


Fig. 1: blokschema van het KGN-68k systeem

Specificaties

Het in figuur 1 getekende blokschema geeft de huidige ideeën over het systeem weer. In principe bestaat het systeem uit 9 onderdelen:

1: De processor. Zoals uit het blokschema blijkt is gekozen voor een 68030.
2: Een 68881 of 68882 coprocessor. Deze rekenkundige bouwsteen is optioneel.
3: Het geheugen. Dit is op te vatten als een zelfstandige eenheid die zelf voor de refresh zorgt. Er wordt rekening gehouden met minimaal 4 megabyte dynamische ram in de vorm van 1 MByte sim- of sip-modules.
4: De DMA-controller.
5: De interrupt controller die als voornaamste taak heeft de interrupts die via de AT-bus binnenkomen om te zetten naar voor een Motorola te verwerken interrupts.
6: De bus-interface. Deze module moet er voor zorgen dat de interne stuursignalen (controlbus) geconverteerd worden naar signalen die AT-kaarten kunnen begrijpen en andersom.
7: De seriële aansluitingen volgens de RS-232C standaard.
8: De real-time clock. Deze verzorgt ook de interrupts voor de taakwisselingen.
9: De boot-eprom. Deze bevat het boot-programma en eventueel nog een monitor-programma.

Zoals uit de figuur blijkt, communiceren de bovengenoemde modules via een interne bus met elkaar. Deze interne bus voert alle signalen die voor een optimaal functioneren van het systeem noodzakelijk zijn. Een aantal van deze signalen (voornamelijk voedings, adres- en datalijnen) worden rechtstreeks aangesloten op de overeenkomstige lijnen van de AT-bus. Dit betekent dat 16 data-lijnen en 24 adreslijnen over de AT-bus naar buiten gevoerd worden. De overige signalen (dus 16 datalijnen en 8 adreslijnen plus een belangrijk deel van de stuursignalen) worden via de 030 complement bus naar buiten gevoerd. Op deze wijze kunnen aan toekomstige 32 bits uitbreidingskaarten toch alle benodigde signalen aangeboden worden. Om de AT-kaarten te kunnen gebruiken, wordt in het bus-interface de conversie tussen de stuursignalen in de AT-bus en de Motorola-bus uitgevoerd.

De belangrijkste beslissing die ondertussen genomen is, is dat we de MC68030 als CPU genomen hebben. De '030 is een 68020 met een ingebouwde Memory Management Unit. Een Memory Management Unit heeft als voordeel dat, in een Multi Tasking systeem, dat MINIX ook is, het omschakelen van taken een fluitje van een cent is. Wat de specifie-

ke voordelen van een MMU zijn wordt t.z.t. in een artikel in de μP Kenner verder uitgelegd.

Onze processorkaart krijgt 4Mb DRAM opgebouwd als 32 * 1Mbit. Samen met de processor bepaalt de hoeveelheid RAM de performance en ook de prijs. Vandaar dat getracht wordt om een realisatie met 256 kBit IC's als optie open te houden. Gedacht wordt aan minimaal één set van 32 * 256 kBit met, als de printruimte het toelaat, een mogelijkheid voor een tweede set van nog eens 32 IC's. Voor hen met hogere eisen en hoger zakgeld biedt dit bovendien de mogelijkheid uit te breiden naar acht Mb (totaal 64 * 1 MBit). De tweede bank is echter afhankelijk van de nog vrije printplaat grootte. Een alternatief voor een goedkoper geheugen is de mogelijkheid niet de volle breedte van het geheugen te gebruiken. De 68020, '30 en '40 kunnen namelijk omgaan met geheugenbreedtes van 8, 16 en 32 bit. In dat geval ga je uit van 16 maal 1 MBit of desnoods 8 maal 1 MBit. Dit gaat, in tegenstelling tot de eerste oplossing, ook bij geringer geheugengebruik meteen ten koste van de snelheid van het systeem. Het instellen van de geheugenbreedte kan door middel van een draadbrug o.i.d. zodat een latere uitbreiding van geheugen en een verhoging van de snelheid op eenvoudige gedaan kan worden.

Dat houdt het nog leuk om ROM software aan te passen van een eigenlijk 32bits brede (4 EPROMS) machine.

Het van buiten de processor vertellen hoe breed het aangesproken geheugen is (Dynamic Bus Sizing) maakt het ook mogelijk om een 8bits brede bootrom te gebruiken. Dat houdt het nog leuk om ROM software aan te passen van een eigenlijk 32bits brede (4 EPROMS) machine. Het zal mogelijk zijn om meer dan 64 Kb in ROM te zetten want er wordt een 32 pins socket voor gereserveerd.

De twee seriële poorten, die het board krijgt, komen uit een Dual Asynchronous Receiver/Transmitter, DUART, de MC68681 van Motorola. Het niveau van de signalen aan de buitenkant zal overeenkomen met RS232.

Motorola maakt voor haar 68000 familie drie Direct Memory Access Controllers (gegevens van 25 januari 1991). Hiervan heeft elke controller zijn specifieke voor- en nadelen. Welke controller het gaat worden, is momenteel nog niet precies bekend.

Eén van de werkgroepleden heeft een zijns inziens prima toepasbare interrupt controller gevonden: de 68155. Dit device zou de 'interrupt-bottleneck' tussen Motorola en Intel omgeving adequaat op kunnen lossen. Nadere studie wordt verricht, tevens wordt er geïnformeerd naar prijs en verkrijgbaarheid.

Wat de (math) coprocessor betreft is er niets veranderd: voor een 68881 of '882, is geen extra hardware nodig. Het is niet noodzakelijk om het systeem te laten draaien. Voor de ontwerpers is het enige waar rekening mee gehouden moet worden +/- 4 vierkante centimeter printplaat. Gezien de kracht van een co-processor is het verantwoord deze ruimte vrij te houden.

De definitieve Real Time Clock is niet bekend, wel zou deze chip RAM moeten hebben om een aantal configuratie parameters in op te slaan.

De processor gaat zover mogelijk naar buiten via de AT bus. De overgebleven adres- en datalijnen en de specifiek Motorola lijnen gaan via een andere connector naar buiten. Toekomstige 32-bit uitbreidingskaarten kunnen op deze manier beschikken over alle signalen van de interne bus zodat niets een uitbreiding met geheugenkaarten, videokaarten en I/O-kaarten in de weg staat. Er kan gebruik gemaakt worden van AT-kaarten en van binnen de club ontwikkelde kaarten voor KGN-68k, kortom vrijwel niets is op dat gebied onmogelijk.

Afsluiting

Tot zover de rapportage voor deze maand. U ziet, dat er al een aantal zaken gedaan zijn. De volgende stap is het in grote lijnen opzetten van het schema waarbij ook de problemen van het bus-interface opgelost moeten worden. Verder gaat de softwaregroep ook zeer binnenkort van start. Voor de port van MINIX gaan we uit van de Atari versie (die t.z.t. dus ook door de nabouwers gekocht zal moeten worden). Hiervoor heeft de projectgroep een harde schijf voor een Atari nodig en eventueel nog een Atari machine. Als u dus dergelijke hardware hebt staan en u gebruikt het niet, dan vragen we u dit tijdelijk ter beschikking te stellen.
Tot zover de rapportage van deze maand. Meer over het project op de bijeenkomst in Geldrop en in de volgende uitgave van de μP Kenner.

Namens de projectgroep:

Geert Stappers(projectleider)
Gert van Opbroek (secretaris)

Een IBM keyboard aan DOS65

Het is nu mogelijk om, op een simpele manier, een IBM PC/XT keyboard aan de DOS65 te koppelen. Deze sessie is zowel hard- als software ondersteund. Zowel de hard- als de software zijn volledig zelfsupporting, d.w.z. er hoeft niets in de DOS65 omgesoldeerd, danwel opnieuw geprogrammeerd te worden. Dit geeft gelijk het zeer grote voordeel aan deze oplossing t.o.v. eerder gepubliceerde artikelen.

De software

Hiervoor wordt het programma "IBM.MAC" en "ASCII.MAC" van Wim Schimmel te voorschijn gehaald. Hier en daar wat wijzigen en het is helemaal geschikt voor dit project. (zie de 6502 kenners no.53, blz 7 e.v.). Tevens is er een mogelijkheid voor "printscreen" ingebouwd. Iedere keer als de toets SHIFT "*/PrtSc" ingedrukt wordt, wordt eerst controleert of het eigenlijke printscreen programma in het geheugen staat ($A000-einde). Is dit niet het geval dan wordt het programma van disk gehaald en gerunned, staat het programma wel in het geheugen dan wordt het direct gerunned.

IBM_keyb.mac is het toets afvraagprogramma. Het staat in figuur 2.
IBM_table.mac bevat de ASCII toets-waarden en staat in figuur 3. Figuur 4 tenslotte bevat scrdmp.mac, een simpele printer-driver.

De hardware

De hardware bestaat uit een 3-tal IC's, 3 weerstanden, 2 diodes en 2 condensatoren. De schakeling wordt enerzijds d.m.v. een 5 polige DIN-plug met de toetsenbordconnector verbonden en anderzijds door middel van bijvoorbeeld een flatcable aan de toetsenbordingang van de computer verbonden. De computer levert de benodigde voeding.

De werking

Als er een toets ingedrukt wordt genereert het toetsenbord een reeks klokpulsen en een 8 bits toetscode, welke in serie aangeboden wordt, beginnend met D0.

De klokpulsen gaan via N1 naar een pulsvertragend netwerk (R1-R2/C1) om dan als klokpuls aan de 2 SIPO registers aangeboden te worden. Parallel aan deze klokpulsen wordt de datalijn rechtstreeks aan deze SIPO aangesloten. Het pulsvertragend netwerk is aanwezig om straks het KS signaal op het juiste moment te laten komen, zodat er niet tijdens een flankverandering geclocked kan worden. Na 8 vertraagde klokpulsen wordt poort N4 vrijgegeven, zodat de 9e klokpuls de KeyStrobe genereert. Deze strobe blijft de "on" periode aanwezig. Het toetsenbord levert 10 klokpulsen, waardoor aan het eind van de omzetting de schakeling zichzelf reset m.b.v. N3/D1. De combinatie R3/C2-D2 zorgt dat tijdens het inschakelen van de systeem de 2 SIPO's gereset worden.

Opmerking:
Nu zullen enkelen onder ons zich afvragen: "Leuk, maar hoe boot ik nu het systeem op, daar er nog geen keyboard-driver in mijn computer zit !?!?" Heel eenvoudig; door het drukken op toets F8, want deze heeft de toetscode $42 = B(oot) en hoppekee de systeem drive gaat lopen en het computer plezier kan aanvangen, nu met een "echt" toetsenbord.
Uiteraard wel even de keyboard-driver vanuit de LOGIN.COM installeren.

Veel succes,

Frank Bens

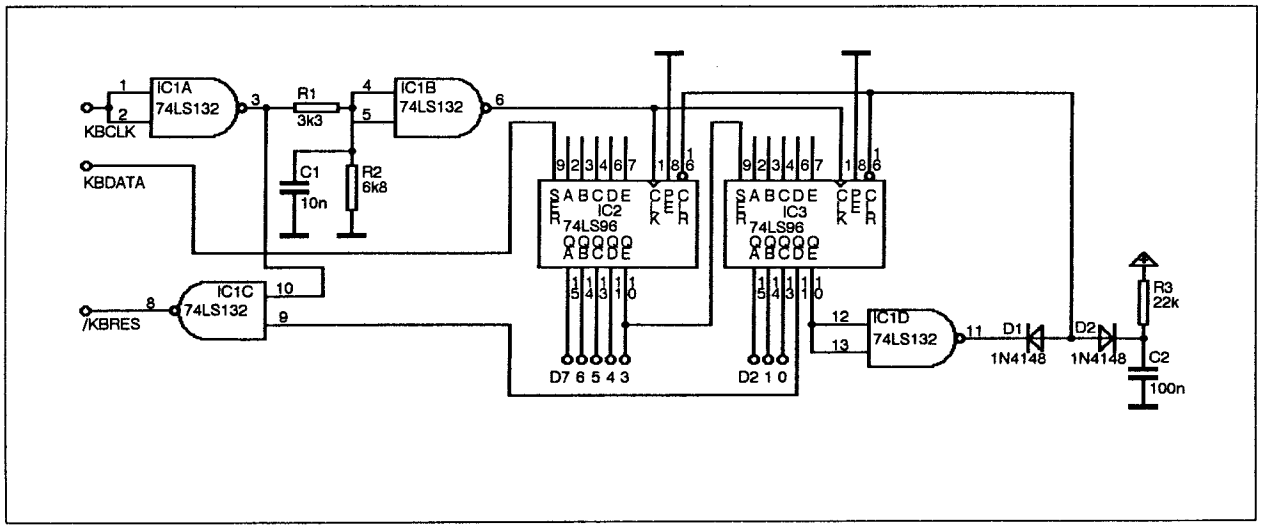


Fig. 1: toetsenbordinterface schakeling

```
; File : ibm_keyb.mac
;
;               IBM toetsenbord vertaler voor DOS-65 computer
;               Een idee van: Wim Schimmel (6502 kenner no.53)
;               Datum : 23-OKT-86
;               Update/aangepast : Frank Bens dd. 24-DEC-90
;
              opt     nogen

;*
;*
;*  Variables & subroutines
;*
;*
workmem       equ     $a000
initorg       equ     $a000
introrg       equ     $e4af

jsr           equ     $20             ; OPcode
bit           equ     $24

command       equ     $c006           ; DOScommand

viaa          equ     $e100           ; VIA 1 (o.a keyboard)
vapad         equ     viaa+1          ; Poort A Data

conint        equ     $cb2e
keypnt        equ     $e7b7           ; Pointer in keyboardbuffer
tstnd         equ     $e7e0           ; Test speciale toets adres

              org     initorg
ibm           php
              sei                     ; Disable interrupts
              lda     #0
              sta     keypnt          ; Reset keyboardbuffer pointer
              sta     tstnd           ; Initieele waarde is 0 (alles uit)
              lda     #jsr            ; JSR INTER
              sta     conint
              lda     #inter&255
              sta     conint+1
              lda     #inter>>8
              sta     conint+2
              lda     #bit            ; Skip RTS instruktie
              sta     conint+3
              plp                     ; Enable interrupts
              rts                     ; Einde initialisatie routine

              org     introrg

inter         php                     ; Bewaar status register
              and     #%01111111      ; Verwijder het MSB
              tay     ; Toetswaarde in Y
              ldx     tascii-1,y      ; Haal ASCII waarde uit de tabel
              txa
```

Fig. 2: sourcetekst van IBM_keyb.mac

```
            plp                         ; Herstel status register
            bpl     maak                ; Spring als toets is ingedrukt
            cmp     #numlck
            bcc     brkend              ; Spring indien geen control toets
            cmp     #caplck+1
            bcc     lockkey             ; Spring indien CAPS- of NUM-lock toets
            and     tstnd               ; Reset het bijbehorende bit
spclend     sta     tstnd
brkend      pla                         ; Geen geldige ASCII waarde,
            pla                         ; Verwijder terugkeeradres
            rts
            ;
lockkey     lda     #~ctrl
            bit     tstnd               ; Test Ctrl bit
            bne     brkend              ; Spring als Ctrl ingedrukt is
            txa
            eor     #$ff                ; CAPS- of NUM-lock toets
            eor     tstnd               ; Inverteer het 'toggle' bit
            bcc     spclend             ; Altijd
            ;
maak        cmp     #numlck
            bcc     no_ctrl             ; Spring indien geen control toets
            cmp     #caplck+1
            bcc     brkend              ; Spring indien CAPS- of NUM-lock toets
            eor     #$ff                ; Ctrl, Alt of Shift toets
            ora     tstnd               ; Zet het bijbehorende bit aan
            bcs     spclend             ; Altijd
            ;
no_ctrl     cmp     #'a'
            bcc     no_lttr
            cmp     #'z'+1
            bcc     no_mrk1
no_lttr     cpy     #$47                ; Numeriek pad toets?
            bcc     no_mrk              ; Spring indien niet numeriek
            lda     tstnd
            and     #~numlck|~shift
            beq     mkend               ; Spring als NUM-lock en Shift uit zijn
            cmp     #~numlck|~shift
            beq     mkend               ; Spring als NUM-lock en Shift aan zijn
            ldx     tnumlck-$47,y       ; Haal numeriek pad toets
            bpl     mkend               ; Altijd
            ;
no_mrk1     lda     #~caplck|~shift
            fcc     $2c                 ; Dummy BIT instruktie
            ;
no_mrk      lda     #~shift
            bit     tstnd               ; Test Shift bit
            beq     mkend               ; Spring als Shift niet ingedrukt
            ldx     tshift-1,y          ; Haal shift toets
            cpx     #prtscr             ; Print scherm ?
            beq     prscr
            ;
mkend       lda     #~ctrl
            bit     tstnd               ; Test Ctrl bit
            beq     einde               ; Spring als Ctrl niet ingedrukt
            txa
```

```
                cmp     #ins
                bcs     curctrl         ; Als ASCII waarde > = cursor-control
                and     #%00011111      ; Control code ($00 - $1F)
                fcc     $2c             ; Dummy BIT instruktie
                ;
curctrl         adc     #15             ; Cursor control code
                tax
                ;
einde           txa                     ; Karakter in A-register
                rts                     ; Geldige ASCII waarde
                ;
prscr           ldx     #6              ; Controleer of file
check           dex     ; reeds in geheugen
                bmi     exec            ; aanwezig is
                lda     scrdmp,x
                cmp     workmem+3,x
                beq     check
loadfil         lda     #getdmp>>8      ; Zoniet, lees file
                ldy     #getdmp&255
                jsr     command
                lda     #'\r'
                bvs     exit            ; Spring indien file niet bestaat
exec            jsr     workmem         ; Executeer programma
                pla                     ; Verwijder terugkeeradres
                pla
exit            rts
```

```
; File  : ibm_tabl.mac
;
;               ASCII converter voor IBM (/XT) toetsenbord.
;               Naar een idee van: Wim Schimmel (6502 kenner no.53)
;               Datum : 22-OKT-86
;
;*
;*
;*              Variables
;*
;*
break           equ     $03             ; shift scroll lock
scrlck          equ     $13             ; scroll lock (DC3)
esc             equ     $1b             ; escape
spcbar          equ     $20             ; spacebar
del             equ     $7f             ; delete

                ; funktie toetsen

f1              equ     $80
f2              equ     $81
f3              equ     $82
f4              equ     $83
```

Fig. 3: toetsentabel voor IBM toetsenbord

```
f5          equ     $84
f6          equ     $85
f7          equ     $86
f8          equ     $87
f9          equ     $88
f10         equ     $89

tab         equ     $8a             ; tab rechts

            ; Shift funktie toesten

sf1         equ     $8b
sf2         equ     $8c
sf3         equ     $8d
sf4         equ     $8e
sf5         equ     $8f
sf6         equ     $90
sf7         equ     $91
sf8         equ     $92
sf9         equ     $93
sf10        equ     $94

stab        equ     $95             ; tab links

prtscr      equ     $96             ; print screen

            ; numeriek pad toetsen

ins         equ     $a0
end         equ     ins+1
down        equ     ins+2
pgdn        equ     ins+3           ; page down
left        equ     ins+4
centre      equ     ins+5
right       equ     ins+6
home        equ     ins+7
up          equ     ins+8
pgup        equ     ins+9           ; page up
delkp       equ     ins+10          ; delete (keypad)

            ; Speciale control karakters

numlck      equ     %11101111
caplck      equ     %11110111
alt         equ     %11111011
shift       equ     %11111101
ctrl        equ     %11111110

            ; Tabellen

tascii      fcc     esc,'1234567890-=',del
            fcc     tab,'qwertyuiop[]\r'
            fcc     ctrl,'asdfghjkl;\"'
            fcc     shift,'\\zxcvbnm,./',shift,'*'
            fcc     alt,spcbar,caplck
            fcc     f1,f2,f3,f4,f5,f6,f7,f8,f9,f10
```

```
                fcc     numlck,scrlck
                fcc     home,up,pgup,'-'
                fcc     left,centre,right,'+'
                fcc     end,down,pgdn
                fcc     ins,delkp

tshift          fcc     esc,'!@#$%^&*()_+',del
                fcc     stab,'QWERTYUIOP{}\r'
                fcc     0,'ASDFGHJKL:"~'
                fcc     0,'|ZXCVBNM<>?',0,prtscr
                fcc     0,spcbar,0
                fcc     sf1,sf2,sf3,sf4,sf5,sf6,sf7,sf8,sf9,sf10
                fcc     0,break
tnumlck         fcc     '789-'
                fcc     '456+'
                fcc     '123'
                fcc     '0.'

                end     ibm
```

```
;
; File : SCRDMP.MAC
;
; Door          : Frank Bens
;
;               Screendump routine
;
; Datum : 24-DEC-90
;
                opt     nogen

;*
;*
;*              Variables & subroutines
;*
;*
workmem         equ     $a000

cold            equ     $c000
statpri equ     cold+$a15               ; Status printer
putpri          equ     cold+$a18       ; Print always (ignore status)
userbrk         equ     cold+$a1e       ; Check user break

start           equ     $e7ad           ; Top current videoscreen (2 bytes)

vidram          equ     $e800           ; Start of videoram

jmp2byt         equ     $2c             ; OPcode BIT xxxx

hight           equ     24              ; Rows per screen
width           equ     80              ; Character per row
```

Fig. 4: eenvoudige Print_Screen driver

```
                org     workmem

stdump          jmp     scrdmp

file_id         fcc     'scrdmp'        ; File identifier (always 6 bytes long)
version         fcc     ' V1.0'         ; Version number
linecnt         res     1               ; Linecounter
;*
;*
;*              Screendumproutine
;*
;*
scrdmp          cli                     ; Printing possible
                lda     start           ; Print screen
                ldx     start+1
                sta     getscr+1
                stx     getscr+2
                ldx     #hight          ; Set line-counter
                stx     linecnt
prloop          ldx     #width          ; Set colomn-counter
getscr          lda     getscr          ; Get screen data
                and     #%01111111      ; Strip hi bit/deselect inverse video
                cmp     #' '
                bcs     2.f             ; Branch when > = $20
                lda     #'\'            ; Print special character
2               jsr     centro          ; To centronic device
                bcs     prerr           ; Branch on error
                inc     getscr+1        ; Incerease currentaddress videoram
                bne     chkcol
                inc     getscr+2
                lda     getscr+2        ; Check videoramborders
                and     #vidram+$0700>>8                ; Max. $EFFF
                ora     #8              ; Min. $E800
                sta     getscr+2
chkcol          dex
                bne     getscr          ; End of line ?
                lda     #'\r'           ; Start on new line
                jsr     centro          ; To centronic device
                lda     #'\n'
                jsr     centro
                dec     linecnt         ; Linecounter -1
                bne     prloop          ; Entire screen printed ?
prerr           rts                     ; End of print screen routine

centro          tay                     ; Save accu
wait            jsr     userbrk         ; Test breakkey
                bcs     90.f            ; Breakkey pressed, yes then exit
                jsr     statpri         ; Get printer status
                bcs     wait            ; ACK received, no then wait
                tya                     ; Restore accu
                jsr     putpri          ; Print always (ignore status)
90              rts                     ; Exit

                end
```

Methoden en technieken voor datacommunicatie (Deel 6)

Inleiding

Zoals in de vorige aflevering al is aangegeven, wordt in deze aflevering wat dieper ingegaan op enkele protocollen voor file-transfer zoals die bij hobby-computers gebruikt worden. Het betreft in deze aflevering protocollen voor de uitwisseling van files, dat wil zeggen dat een brok informatie (een programma, een tekstbestand of iets dergelijks) van de ene computer naar de andere computer overgestuurd moet worden. Het gaat hierbij meestal om grote brokken informatie (meerdere kilobytes of zelfs megabytes) die overgestuurd worden om vervolgens bij de ontvanger op schijf opgeslagen te worden. Dit dus in tegenstelling tot bijvoorbeeld het uitwisselen van besturingsopdrachten waarbij een opdracht naar een robot of iets dergelijks over het algemeen kort is (minder dan een kilobyte) en door de ontvanger meestal meteen uitgevoerd wordt.

Diverse protocollen

Voor het uitwisselen van files zijn een groot aantal protocollen beschikbaar. Ons Bulletin Board The Ultimate ondersteunt bijvoorbeeld de volgende protocollen:

Super8K
K9X
BiModem
Lynx
Puma
HyperDrive
JModem
Kermit
Translink
MegaLink
SEALink
True YModem
WXModem
XModem
YModem
ZModem

Kortom: een grote verscheidenheid aan protocollen. Uiteraard hoeft een gebruiker niet alle protocollen te kennen. Als hij ervoor zorgt dat degene die de file verstuurt en degene die de file ontvangt hetzelfde protocol gebruiken, dan kan er weinig mis gaan. Uiteraard zijn er wel verschillen tussen de protocollen. Zo hebben de oudere protocollen zoals bijvoorbeeld XModem slechts weinig mogelijkheden. Verder zijn deze protocollen vaak langzamer dan de meer moderne protocollen. Een modern protocol heeft meer mogelijkheden en is vaak ook sneller. Zo heeft BiModem bijvoorbeeld de mogelijkheid gelijktijdig met het oversturen van een file nog andere dingen (bijvoorbeeld post bekijken) te doen.

Kermit is een oud protocol maar heeft een aantal moderne uitbreidingen. De oerversie van Kermit is heel traag omdat het met zeer korte packets (ongeveer 90 byte) werkt. Een moderne versie van Kermit kan echter ingesteld worden op extented packets tot 1 of 2 kilobyte. Daarmee wordt Kermit ongeveer net zo snel als ZModem, een modern protocol met een packet-lengte van 1 kilobyte. Helaas wordt (werd?) deze uitbreiding niet door The Ultimate ondersteund. Een zeer groot voordeel van Kermit is dat het protocol en het bijbehorende communicatiepakket voor zo'n beetje alle computers (micro, mini, mainframe en super) beschikbaar is en dat het protocol in staat is zich aan te passen aan de definitie (aantal bits, pariteit) van de seriële poort bij zender en ontvanger. Rond het Kermit-protocol is voor de meeste machines een Kermit-communicatieprogramma ontwikkeld. Met behulp van dit programma kan men op comfortabele wijze twee computers met elkaar laten communiceren en kunnen files uitgewisseld worden. Voor mensen die meer over Kermit willen weten, verwijs ik naar de in de referenties genoemde literatuur [1,2,3,4].

Het XModem-protocol

Hoewel XModem een oud protocol is, wordt hij toch in dit artikel behandeld. In de eerste plaats omdat het XModem protocol vaak de basis van andere protocollen (YModem, Telink, Megalink ...) is en in de tweede plaats omdat XModem een protocol is dat eenvoudig van opzet is en eventueel op eenvoudige wijze zelf geprogrammeerd kan worden (C-source bij de auteur beschikbaar).

Het protocol is als eerste geprogrammeerd in 1979 door Ward Christensen in een programma met de naam Modem2. Om deze reden wordt XModem ook wel het Modem of Modem2 protocol genoemd.

Het protocol maakt gebruik van brokjes informatie met een vaste lengte van 128 bytes. Verder wordt de informatie in bytes, dus met een lengte van 8 bits, overgestuurd. XModem is niet in staat informatie die uit bytes van 8 bits bestaat (bijvoorbeeld binaire files) over een communicatielijn te sturen die werkt met 7 bits en een pariteit. Verder is het zo dat er altijd 128 tekens per packet overgestuurd worden, dit betekent dat aan het einde van de file eventueel extra filler-bytes toegevoegd worden. Deze extra bytes worden ook bij de ontvanger op schijf gezet en behoren, wat betreft de ontvanger, gewoon tot de file. Het oversturen van een file met behulp van XModem maakt een file dus iets langer. Als filler wordt het end-of-file teken van de afzender gebruikt. Voor MS-DOS is dit CTRL-Z (HEX 1A).

De opbouw van een XModem packet is weergegeven in figuur 1.

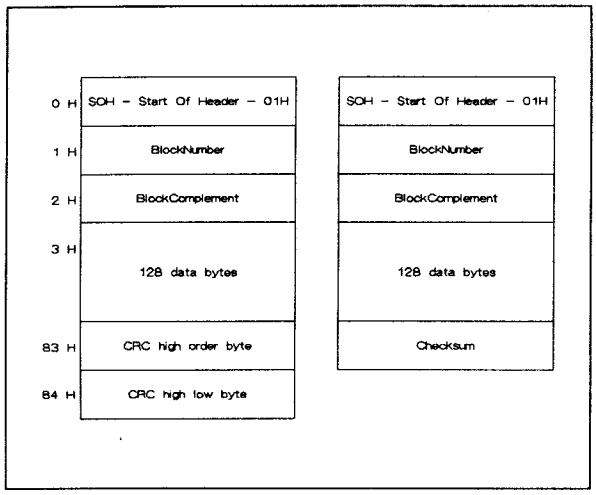


Fig. 1: inhoud van een XModem packet

Een XModem packet begint met een speciaal teken dat SOH (Start of Header) genoemd wordt. Hiervoor wordt het ASCII-teken HEX 01 gebruikt. Vervolgens volgt een byte dat het volgnummer van het packet aangeeft. Dit volgnummer heeft de waarde 1 (HEX 01) voor het eerste packet, de waarde 2 (HEX 02) voor het tweede enz. tot de waarde 255 (HEX FF) voor het 255-ste packet. Het 256-ste packet krijgt volgnummer 0 (HEX 00) waarna weer doorgenummerd wordt naar 2, 3 enz. Het derde byte in de header bevat het (1's) complement van het volgnummer; dit geeft een mogelijkheid het volgnummer op zijn juistheid te controleren. Vervolgens komen er 128 bytes data die verder niet door het protocol geïnterpreteerd worden. Het data-packet wordt afgesloten met een checksum van 1 byte of een CRC van 2 byte. Die uit de complete inhoud van header en data-packet berekend worden.

Opvallend is dat er packets zijn met een totale lengte van 132 byte en packets met een lengte van 133 byte. Het verschil zit in het gebruik van een checksum van 1 byte of een CRC met een lengte van 2 byte (CRC-16). De 1 byte checksum wordt berekend door de exclusive or van alle bytes in het packet te nemen, de CRC wordt (in principe) berekend door het bijbehorende polynoom:

$$2^{16} + 2^{12} + 2^5 + 2^0$$

op het packet los te laten . Hierbij stelt ** de tot-de-macht operatie voor. Het high-order byte wordt als eerste gestuurd, gevolgd door het low-order byte. Hoe een dergelijke CRC in de praktijk berekend wordt, staat zeer fraai beschreven in het artikel van Bram de Bruine [5]. Het voordeel van een CRC boven een eenvoudige checksum is dat de mogelijkheden voor foutdetectie en eventueel -correctie bij een CRC beter zijn. Bij een eenvoudige checksum is het onmogelijk een fout in een even aantal bits in dezelfde kolom te detecteren. Een CRC biedt deze mogelijkheid wel.

Of er gebruik gemaakt wordt van CRC of van checksum wordt bij de start van de file-transfer vastgesteld.

Hoe gaat het nu verder allemaal in zijn werk? Welnu, uitgangspunt is dat de ontvanger bepaalt wat er allemaal gebeurt. Bij de start van de file-transfer vraagt de ontvanger aan de zender een packet te sturen. Hij doet dit door het teken 'C' (HEX 43) of het teken NAK (HEX 15). Stuurt hij een 'C, dan betekent dit dat de voorkeur van de ontvanger uitgaat naar communicatie met een 16 bit CRC; stuurt hij een NAK, dan betekent dit dat hij graag wil dat er met een checksum (1 byte) gecommuniceerd wordt. Kan de zender niet werken met een CRC, dan kan hij rustig afwachten tot de ontvanger inziet dat hij geen packets met CRC zal krijgen en een NAK voor een packet met checksum stuurt. Nadat de ontvanger de 'C' of NAK ontvangen heeft, stuurt hij het eerste packet. Komt dit packet goed over, dan stuurt de ontvanger een ACK (HEX 05), komt dit packet niet aan of verkeerd over, dan stuurt hij een NAK (HEX 15). Na een ACK stuurt de zender het volgende packet, na een NAK stuurt hij nogmaals het packet dat niet goed overkwam. Heeft de zender niets meer te sturen, dan stuurt hij een enkel teken, namelijk de EOT (HEX 04).

Om de fout-detectie eenvoudig te maken, wordt aangeraden geen informatie over te sturen als er van de andere kant nog informatie komt. De ontvanger mag dus geen NAK sturen midden in het oversturen van een data-packet van zender naar ontvanger. Verder mag de zender alleen iets sturen als daar door de ontvanger om gevraagd wordt. De ontvanger is dus ten alle tijde de master over de communicatielijn. Tenslotte wordt het oversturen van informatie in beide richtingen met time-outs bewaakt. Komt een packet niet binnen een vaste tijd na het vragen erom bij de ontvanger binnen, dan zal hij hier opnieuw om vragen. Hij doet dat door het opnieuw sturen van een NAK. Was de ACK (of NAK) op het vorige packet zoekgeraakt, dan wordt het vorige packet op deze manier nogmaals gestuurd. Voor de ontvanger is dat geen enkel probleem, hij ziet dat hij het packet nog een keer krijgt en zal deze met ACK beantwoorden en vervolgens toch weggooien.

Nadelen van XModem

Aan XModem zijn nogal wat nadelen verbonden. In de eerste plaats is er in het protocol geen enkele voorziening voor het oversturen van file-namen. In de praktijk betekent dit dat eerst aan de zender verteld moet worden welke file opgestuurd moet gaan worden waarna vervolgens aan de ontvanger verteld wordt wat de naam van de file die binnenkomt moet worden. Verder is er uiteraard ook geen mogelijkheid de zogenaamde file-attributen (lengte, bescherming, datum en tijd van aanmaak etc.) over te sturen. De ontvanger weet alleen dat er een file aankomt en hoe die moet gaan heten. Hoe groot de file is, dus hoe lang het gaat duren en hoeveel schijfruimte hij nodig is wordt hem niet verteld.

Het tweede nadeel houdt nauw verband met het eerste. Het is niet mogelijk zogenaamde batch-transfers uit te voeren. Dit wil zeggen dat voor elke file een aparte XModem-sessie gestart moet worden. Je kunt dus niet drie files, al dan niet met zogenaamde wildcards ("sterretjes") in de filenaam op laten sturen. XModem heeft namelijk niet de mogelijkheden om uit de stroom packets af te leiden of er een nieuwe file aankomt en hoe die nu weer moet gaan heten.

Het derde nadeel is de vaste lengte van een datablock van 128 byte. In de eerste plaats is deze lengte, zeker met de moderne modems, aan de korte kant. Voor elke 128 bytes die effectief overgestuurd worden, worden 4 of 5 extra bytes in het packet gestuurd en moet de zender ook nog wachten op een byte van de ontvanger. Aangezien de ontvanger elk packet dat binnengekomen is eerst nog even moet analyseren (CRC of checksum berekenen) en soms op schijf moet schrijven gaat hiermee ook een beetje tijd verloren. Kortom, vrij veel overhead. Tenslotte is het toevoegen van bytes aan een file omdat de grootte van de file geen veelvoud van 128 is natuurlijk een vreemde zaak.

Om deze nadelen op te lossen, zijn er een groot aantal variaties op het XModem protocol bedacht. Er zijn versies die met een lengte van 1024 byte werken en er zijn versie die aan de file-transfer die met volgnummer 01 begint een packet met volgnummer 00 toevoegen. In dit eerste packet wordt dat informatie over de file geschreven. Een voorbeeld hiervan is het zogenaamde TeLink-protocol dat onder andere door FIDO voor communicatie tussen de BBSsen gebruikt is. De inhoud van dit beschrijvingsblock is in figuur 2 getekend.

In figuur 2 zien we dat dit packet altijd met checksum verstuurd wordt. In het packet staat een byte dat aangeeft of er verder met checksum of met CRC gewerkt wordt. In de praktijk zal dit echter altijd een

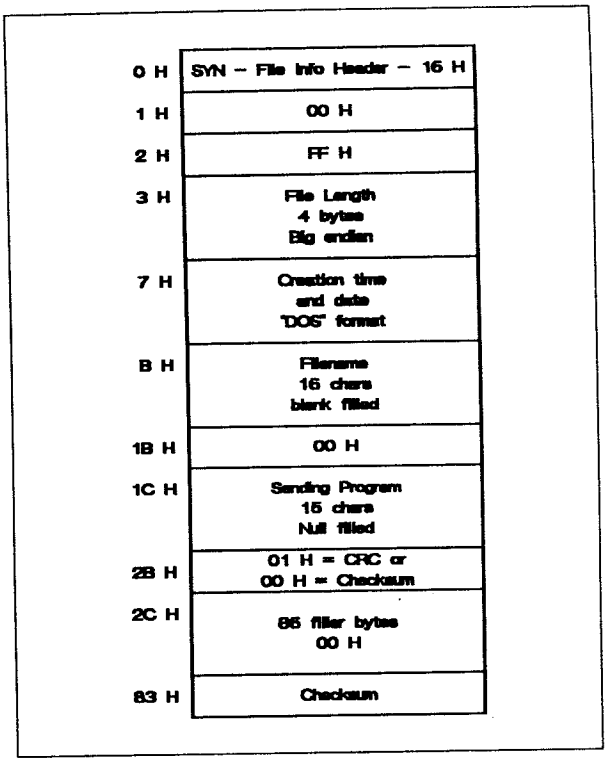


Fig. 2: beschrijvings-packet van het TeLink protocol

CRC zijn omdat er geen implementaties die met checksum werken zijn. De overige packets worden normaal met het XModem protocol (met CRC) verstuurd.

ZModem

ZModem is het communicatie-protocol dat meestal door de auteur gebruikt wordt. Dit protocol is zeer betrouwbaar gebleken en haalt een hoge gemiddelde transmissiesneldheid. De beperking blijkt voor 1200 en 2400 bps verbindingen bijna zonder uitzondering de snelheid van het gebruikte modem te zijn. Bij een modem volgens V22 (1200 bps) wordt in de praktijk zo'n 114 tekens per seconde gehaald. Dit betekent dat de overhead in het ZModem-protocol erg klein is.

ZModem kent zogenaamde data-packets en zogenaamde header-packets. Een data-packet bevat uiteraard gegevens uit de file die overgestuurd moet worden. Header-packets zijn eigenlijk stuuropdrachten voor de andere partij. Vooral van deze header-packets zijn er binnen ZModem een groot aantal aanwezig. Met behulp van deze header-packets kunnen zender en ontvanger samen er voor zorgen dat de file(s) goed overgestuurd worden.

Om een goede foutdetectie-mogelijkheid te hebben, maakt ZModem gebruik van CRC-berekeningen. Voor de header-packets is dit altijd een CRC-16 met

een lengte van twee byte; voor de data-packets is dit, afhankelijk van de afspraken tussen zender en ontvanger een 4 bytes CRC-32 of een CRC-16.

Bij de start van het protocol wisselen zender en ontvanger zogenaamde initialisatie-packets uit. Hiermee maken ze afspraken over de wijze waarop de overdracht plaatsvindt. Het betreft hierbij afspraken over de lengte van de packets, de wijze van CRC-berekening voor de data-packets (CRC-16 of 4 CRC-32) of er sprake is van full duplex communicatie etc. en na hoeveel packets de zender op een bevestiging (Acknoledge) van de ontvanger moet wachten. Ook de lengte van een packet is instelbaar. In de praktijk wordt deze lengte afgeleid van de transmissiesnelheid en de kwaliteit van de lijn. Bij een snelle communicatielijn wordt de maximale lengte van 1024 byte ingesteld, bij een langzame lijn (300 bps) wordt gebruik gemaakt van kortere packets. Dit is gedaan om bij storingen toch binnen enkele seconden te kunnen bepalen dat er iets met een packet verkeerd gegaan is; zou men namelijk bij 300 bps met packets van 1024 byte werken, dan duurt het oversturen van een compleet packet ruim 40 seconden hetgeen betekent dat het altijd 40 seconden duurt voordat de ontvanger weet dat er iets mis is. ZModem kan, afhankelijk van de kwaliteit van de communicatielijn, ook de lengte van een packet aanpassen aan de omstandigheden. Bij een slechte lijn wordt de lengte in stappen gereduceerd tot 32 bytes, bij een goede lijn en een voldoende hoge snelheid van de lijn kan de packetlengte oplopen tot 1024 byte.

ZModem kan afhankelijk van de kwaliteit van de communicatielijn de lengte van een packet aanpassen aan de omstandigheden.

Na het initialiseren van het protocol, kunnen er zogenaamde batches van files overgestuurd worden. Een batch bestaat uit één of meer files. Hierbij wordt elke file voorafgegaan door een packet waarin de filenaam, de datum waarop de file aangemaakt is en de lengte van de file staat. Met behulp van de lengte van de file kan de ontvanger uitrekenen hoe lang het oversturen van de file gaat duren en kan hij bepalen of er voldoende ruimte voor de file op de schijf is. Aan het einde van een file wordt een apart packet gestuurd dat aangeeft dat de file afgelopen is. Verder zijn er nog packets gedefinieerd om aan te geven dat het oversturen van een bepaalde file afgebroken wordt of zelfs dat de hele batch afgebroken wordt. Ook kan aangegeven worden dat het oversturen van een file opnieuw moet beginnen (bijvoorbeeld omdat er teveel fouten geconstateerd zijn).

Het kenmerkende van ZModem is dat de packets niet zondermeer genummerd worden zoals bij XModem het geval is. De data-packets worden genummerd met de positie van het eerste byte in het packet ten opzichte van het begin van de file. Zo krijgt het eerste packet adres nul, het tweede packet adres 1024 (bij een packet-lengte van 1024 bytes) enzovoort. De ontvanger heeft de mogelijkheid aan de zender door te seinen welk packet hij wil ontvangen. bij het starten van het oversturen van een file is dat packet 0 verderop in de file is dit uiteraard een hoger adres.

Komt een packet niet goed binnen, dan zal de ontvanger aangeven dat het packet opnieuw gestuurd moet worden. Hij doet dit door het sturen van een header-packet met daarin het adres van het packet dat niet goed aangekomen is. Komt het packet wel goed binnen, dan hangt het van de situatie af er gebeurt. Als de zender wil dat de ontvanger aangeeft dat een packet goed aangekomen is, dan wordt dit door een vaste combinatie van tekens aan het eind van het datapacket aangegeven. Na ontvangst van deze tekens zal de ontvanger een acknoledge sturen. Stuurt de zender een andere combinatie van tekens, dan houdt dit in dat het volgende datapacket meteen achter het huidige gestuurd wordt. ZModem werkt dan in de zogenaamde stream mode waarin de packets in een min of meer continue stroom gestuurd worden. Of van deze mogelijkheid gebruik gemaakt mag worden, wordt door de ontvanger bij het initialiseren aan de zender verteld, vervolgens beslist de zender wanneer het een bevestiging van de ontvanger wil ontvangen. Bij het detecteren van fouten, zal de ontvanger echter altijd spontaan om herhaling van één of meer packets vragen.

Op deze manier wordt de hele file overgestuurd. ZModem heeft binnen het protocol voorzieningen om kortere packets te sturen dan de afgesproken maximale lengte. Bij ZModem wordt dus niet, zoals bij XModem, gebruik gemaakt van filler-bytes in het laatste packet; het laatste packet is gewoon korter.

Nadat een file overgestuurd is, kan de zender een volgende file sturen of aan de ontvanger vertellen dat er niets meer te sturen valt. In dat laatste geval zullen de zender en ontvanger stoppen met het ZModem-protocol.

Tenslotte nog een mogelijkheid van het ZModem-protocol die onder bepaalde omstandigheden best zinvol kan zijn. Omdat de ontvanger aan de zender door kan geven vanaf welke positie de informatie gestuurd moet worden, kan men in principe op een willekeurige plaats in de file met de transmissie starten. Dit is vooral practisch als je voor het oversturen van een file al een half uur telefoontijd verbruikt hebt en na dat halve uur breekt de verbinding spontaan af. Op dat moment is het zinvol dat je niet alles weer opnieuw hoeft te doen doch dat je vanaf het punt waar gestopt is verder kunt gaan. Zelf heb ik met deze mogelijkheid geen ervaring doch het protocol staat deze manier van werken toe. Om van deze mogelijkheid gebruik te maken, moet voor het protocol ZModem-Resume gekozen worden zodat zender en ontvanger er op voorbereid zijn dat er midden in de file gestart wordt.

Afsluiting

In deze aflevering hebben we het gehad over enkele protocollen voor de uitwisseling van files tussen computer-systemen. Het XModem-protocol is uitgebreid behandeld en van het ZModem-protocol zijn de belangrijkste eigenschappen beschreven.
In de volgende aflevering wil ik wat dieper ingaan op wat er zoal op een Bulletin Board gebeurt en wat je er mee kunt doen.

Referenties en literatuur

1: Frank da Cruz: Kermit Protocol Manual; June 1986
2: Frank da Cruz e.a.: Kermit User Guide, Unix Kermit; January 1988
3: Gert Klein: Kermit een file transfer protocol; de 6502 Kenner 54 (Februari 1988)
4: Gert van Opbroek: Kermit, het communicatieprogramma; de 6502 Kenner 57 (Augustus 1988)
5: Bram de Bruine: Datacommunicatie met micro's; de μP Kenner 60 (Februari 1989)
6: Chuck Forsberg: XModem/YModem Protocol Reference; 5-30-85
7: FidoNet (tm) Protocol Standard; August 18, 1986
8: Paul Meiners: Megalink A File Transfer Protocol; August 9, 1987
9: Peter Boswel: Xmoden, CRC Modem, Wxmodem File Transfer Protocols; June 20,1986
10: J.R. Louvau: ZModem Source Listing for Turbo Pascal; (1988)

Gert van Opbroek

Een frustratie en een ervaring rijker

Nu is het al een poos geleden, dat ik gevraagd ben door de club om SysOp te gaan spelen op het BBS. Maar soms kom je dan best vervelende dingen tegen. Zo heb ik het ongeveer een half jaar geleden voor elkaar gekregen de 130 Mb harddisk van het BBS kiet om te ruilen voor een 160 Mb schijf. Op zich natuurlijk perfect ware het niet dat ik helemaal verzot ben op Compaq DOS 3.31. Ik vond het namelijk maar niets om allemaal van die "kleine" partities van 32 Mb op de schijf te hebben. Het overzicht was ik steeds weer kwijt en moest steeds schuiven met een bepaalde file area omdat de betreffende partitie volliep. Compaq DOS 3.31 is namelijk verder compleet hetzelfde als PC-DOS 3.30 en dat is dus een versie waar je echt goed mee kunt werken.

Maar goed, mijn probleem dus...

In eerste instantie denk je natuurlijk dat zo'n schijf nooit vol zal raken, maar ja er komt een tijd, komt bytes zullen we maar zeggen... Om even duidelijkheid te geven voor straks verder in het verhaal: Bij elke DOS-versie wordt het programmaatje FDISK geleverd waarmee je dus je HD in kunt gaan delen. Nu had ik die er dus niet bijgekregen (hoezo?) en gebruikte ik hier dus SStor voor (uitspreken als Speedstore), die kan dat alles en ook nog iets meer, dus geen problemen tot dusver. Nu gebeurde me een maand geleden iets zeer vreemds. Ik was op de tweede partitie een file aan het ARCen (= uit elkaar halen) omdat ik deze file wilde bekijken. Op een gegeven moment meldde het systeem mij dat hij een bepaalde file op schijf C: niet kon vinden. Vreemd dacht ik nog. Die heb ik net nog gebruikt. Dus wat doe je dan. Je gaat naar schijf C: en kijkt wat er aan de hand is. Tot mijn schrik en stomme verbazing was schijf C: compleet vernield...

Oorzaak??? Op dat moment denk je aan de ergste dingen, maar een mogelijke oorzaak vinden? Ho maar, niet dus. Dus dan alles maar weer gaan installeren (met een backup van 2-3 maand oud, stom he...) Nou, na een uurtje of zes draaide alles weer. Dus maar snel naar bed. Het was tenslotte al behoorlijk laat geworden.

Nou als je zo iets gebeurt en je systeem staat dag in dag uit aan, en aangesloten op de telefoonlijn dan is het natuurlijk vervelend als het systeem plat gaat. Je

hebt tenslotte een aantal vaste inloggers, die ervan uitgaan dat ze op het systeem terecht kunnen. Dus elke ochtend direkt naar het BBS rennen om te kijken of alles nog wel draaide. Je belt zelfs overdag vanuit je werk om te kijken of het BBS de telefoon nog wel opneemt. Zo ja, dan ben je toch weer gerustgesteld.

Nou dat ging dus een goede maand goed. Tot op een goede donderdagnacht... Kom ik vrijdagochtend weer kijken bij het systeem en is hetzelfde als een aantal weken terug gebeurt. Weer de complete partitie C: verdwenen. Gelukkig had ik nog een ATV dag tegoed, dus weer de hele dag sleutelen en doen, zodat het BBS tegen de avond weer online kon (volledig). Dus alles weer OK en je gaat een rustig weekend tegemoet (gelukkig gebeurde er niets). 's zondags "even" de HD gebackupd (2 uurtjes werk) want je weet tenslotte nooit he...

En maandags ook niets aan de hand (gelukkig!) dus kan ik eens lekker vroeg naar bed, want dat schijnt soms ook te moeten (Wie heeft dat nu uitgevonden!) Kom ik dinsdags beneden, en ja hoor... ligt het zaakje weer plat vanaf de vorige avond 00.10 uur (maar goed dat ik toen al in bed lag...) Dus maar even naar het werk bellen dat we niet komen omdat het BBS plat ligt. Maar dit keer neem ik me voor om even een paar tests uit te gaan voeren. Gelukkig draaide er op mijn eigen systeem een copie van het BBS dus die maar opgestart, zodat men tenminste in kan loggen om berichten te lezen. En dan maar weer het BBS systeem aan de praat brengen. Toevallig belde er die dag iemand die me een hint gaf wat het wel eens zou kunnen zijn, wat mijn problemen veroorzaakte. Veel BIOSen ondersteunen namelijk maar schijven die maximaal 1024 cilinders hebben. Nu had Nico de Vries een BIOS speciaal voor me aangemaakt, waarmee de schijf van het BBS ondersteund zou worden. Deze heeft namelijk 1224 cilinders en 15 koppen. Toen ik met Fdisk (ik had in de tussentijd de volledige Compaq 3.31 gekregen) de schijf weer opnieuw wilde gaan indelen meldde, tot mijn stomme verbazing, Fdisk dat ik maar 1024 cilinders had. Dus wat doe je dan, je neemt een andere versie DOS. Dus PC-DOS 3.30 uit de kast en eens kijken wat die ervan maakt. En ja hoor ook 1024 cilinders. Dan ga je al denken.... hier klopt iets niet. Nou ja, dat denk je dan. MS-DOS 4.01 kan zelfs schijven van 1 Gbyte (dat is dus 1024 Mbyte) dus die eens even geprobeerd. Maar ook die meldde dat ik "maar" 1024 cilinders op de schijf had. En dat terwijl ik toch zeker wist dat het er 1224 waren. Wat dan met die laatste 200 cilinders. Dat is toch wel 30 Mb waar je niet bij kunt.

Nou ja, Speedstor dan maar weer. En ja hoor, 1224 cilinders, vertelde me Speedstor, ik wist wel dat ik me niet vergiste. Dus weer aan het werk 1e partitie 30 Mb, de tweede 130 Mb, alles weer terug gezet van de Backup en maar eens even een testje. Wat gebeurt er als we die 1e partitie vol gaan schrijven met allerlei bagger. Resultaat: alles voldoet naar wens. Maar nu eens even die 2e partitie vol gaan maken. Daar hebben we nog 40 Mb op vrij, dus we zitten nog dik onder die 1024 cilinders die Fdisk me meldde. Dus even een paar directories van de eerste partitie over zetten nar de tweede partitie en dan steeds even tussendoor kijken of de eerste partitie nog zinnige informatie vertoonde. Op een gegeven ogenblik, ja hoor, de eerste partitie vernield. Conclusie: Op het moment dat het systeem die 1025ste cilinder wil gaan schrijven, zegt het BIOS, nee nee dat is cilinder 0, hoppa eerste partitie wordt overschreven.

Maar ook die meldde dat ik "maar" 1024 cilinders op de schijf had. En dat terwijl ik toch zeker wist dat het er 1224 waren.

En dat is me dus al drie keer gebeurd. Alleen kom je daar niet op totdat iemand opeens een helder ogenblik heeft en je op die gedachte brengt. Gelukkig draait alles nu weer perfect. De oplossing? Heel simpel eigenlijk. Met Speedstor heb ik nu 3 partities aangemaakt. De eerste is 15 Mb groot. Daar staat dan het complete BBS op (plaatjes en de eigenlijke programmatuur). De tweede partitie heb ik bij cilinder 1000 laten stoppen. Daar staan alle files op die op het BBS te downloaden zijn. De derde partitie loopt dus van 1001 t/m 1224 en heb ik een SStor partitie gemaakt. Wat er nu gebeurde dat er tijdens het opstarten van een driver geladen wordt die het onderhoud van deze partitie voor zijn rekening neemt. Op het moment staan alle berichten op deze partitie en het gaat al een maand goed. (Al herhaaldelijk dingen geprobeerd om de schijf vol te laten lopen, maar er gebeurt, gelukkig, niets meer). Dus dat was behoorlijk frusterend die hele geschiedenis. Maar ja, het heeft toch ook weer een ervaring opgeleverd!!!

Jacques Banser

MINIX, NLMUG en de KIM Gebruikersclub Nederland

Al lezend in de laatste paar nummers van de uP-Kenner komen we toch steeds vaker de namen "MINIX" en "NLMUG" c.q. "MINIX Gebruikersgroep" tegen. Ook is al een paar keer de naam "Fred van Kempen" tegen, en dat zou dan een of andere MINIX-goeroe zijn. Het lijkt daarom een goed idee om eens toe te lichten wat de aanwezigheid van deze termen in dit blad voor een bedoeling heeft...

MINIX?

Om maar te beginnen met MINIX: dat is een (betrekkelijk nieuw) besturingssysteem voor hedendaagse microcomputers (dus ook uw PC!). Het lijkt erg veel op het aloude UNIX besturingssysteem, dat vooral erg prettig draait op de wat grotere systemen. Een bepaalde machine kan dan door meer dan een persoon tegelijk gebruikt worden, en elke gebruiker op zich kan weer meer dan een programma tegelijk laten draaien. Dit noemen we, zoals U ook al in de uitleg van Joost Voorhaar heeft kunnen vernemen, multi-user respectievelijk multi-tasking.

Nu zullen er weinig mensen zijn die thuis (nog?) een grote machine als een VAX of een forse PDP-11 hebben staan, waarop zij dit, volgens sommigen, zooooo geweldige UNIX kunnen draaien...

Daarom is voor velen het MINIX-systeem een eenvoudige manier om te kunnen proeven van de UNIX-omgeving. Het draait op een (betrekkelijk) eenvoudige machine als PC/XT/AT of Atari-ST, en kan dan heel aardige dingen doen. Dit is een van de redenen waarom de KIM Gebruikersclub heeft besloten om ook eens wat te gaan experimenteren met MINIX.

We hebben al kunnen lezen, wat de plannen met betrekking tot MINIX zijn: wat cursussen, wellicht een zelf-ontwikkeld MINIX systeem, en er zijn contacten gelegd met de MINIX Gebruikersgroep Nederland. Dit laatste brengt me op het volgende punt....

NLMUG?

De MINIX Gebruikersgroep Nederland (afgekort NLMUG) is een vereniging voor van MINIX-gebruikers in Nederland, die zich ten doel heeft gesteld het gebruik en begrip van het MINIX-systeem te bevorderen. Oorspronkelijk ontstaan uit een groep mensen rond een bulletin board voor MINIX (het toenmalige "MINIXUG-ONLINE"), is deze vereniging nu een internationaal erkende organisatie op het gebied van MINIX-support en -ontwikkeling.

SAMEN STERK?

Mede als gevolg van deze kennis heeft de KGN contact opgenomen met ons, om eens te horen wat MINIX kon. Dit contact is inmiddels uitgegroeid tot een samenwerkingsverband tussen de beide verenigingen, met als (helaas, weer een fanaatje erbij) gevolg dat U onze bijdragen kunt vinden in dit blad. Er is inmiddels ook overleg geweest over het al dan niet samen organiseren van bijeenkomsten, cursussen en dergelijke, omdat het een grote verspilling van tijd en moeite zou zijn om dat soort zaken uitsluitend binnen de eigen sfeer te houden...

DIT BLAD

In dit blad zal nu ook de NLMUG haar visie geven op de hedendaagse techniek, en dan vooral op het *NIX (dit staat overigens voor "alle soorten en maten van UNIX en UNIX-achtige systemen") front. Er zullen, naast de wat algemenere zaken, ook details van het MINIX-systeem worden besproken, bijvoorbeeld "hoe schrijft men een device driver voor MINIX". Daar er ook vanuit de KGN en haar leden veel belangstelling is voor ons netwerk (MUGNET genaamd) zal ook aandacht worden besteed aan het "hoe en waarom van een netwerk", met natuurlijk speciale aandacht voor MUGNET en haar koppeling met het "The Ultimate" bulletin board van de KGN.
Met een vriendelijke groet,

Fred 'The Rebel' van Kempen
MINIX User Group Holland (NLMUG)

Enkele aanvullingen van de redactie

Het komt op een aantal lezers misschien wat vreemd over opeens informatie van en over de NLMUG in de μP Kenner te vinden. Dit is, zoals Fred terecht opmerkt, een gevolg van een wat nauwere samenwerking tussen de KGN en NLMUG. Deze samenwerking is op de ledenvergadering in Krommenie in januari 1991 aan de daar aanwezige leden voorgelegd en goedgekeurd.

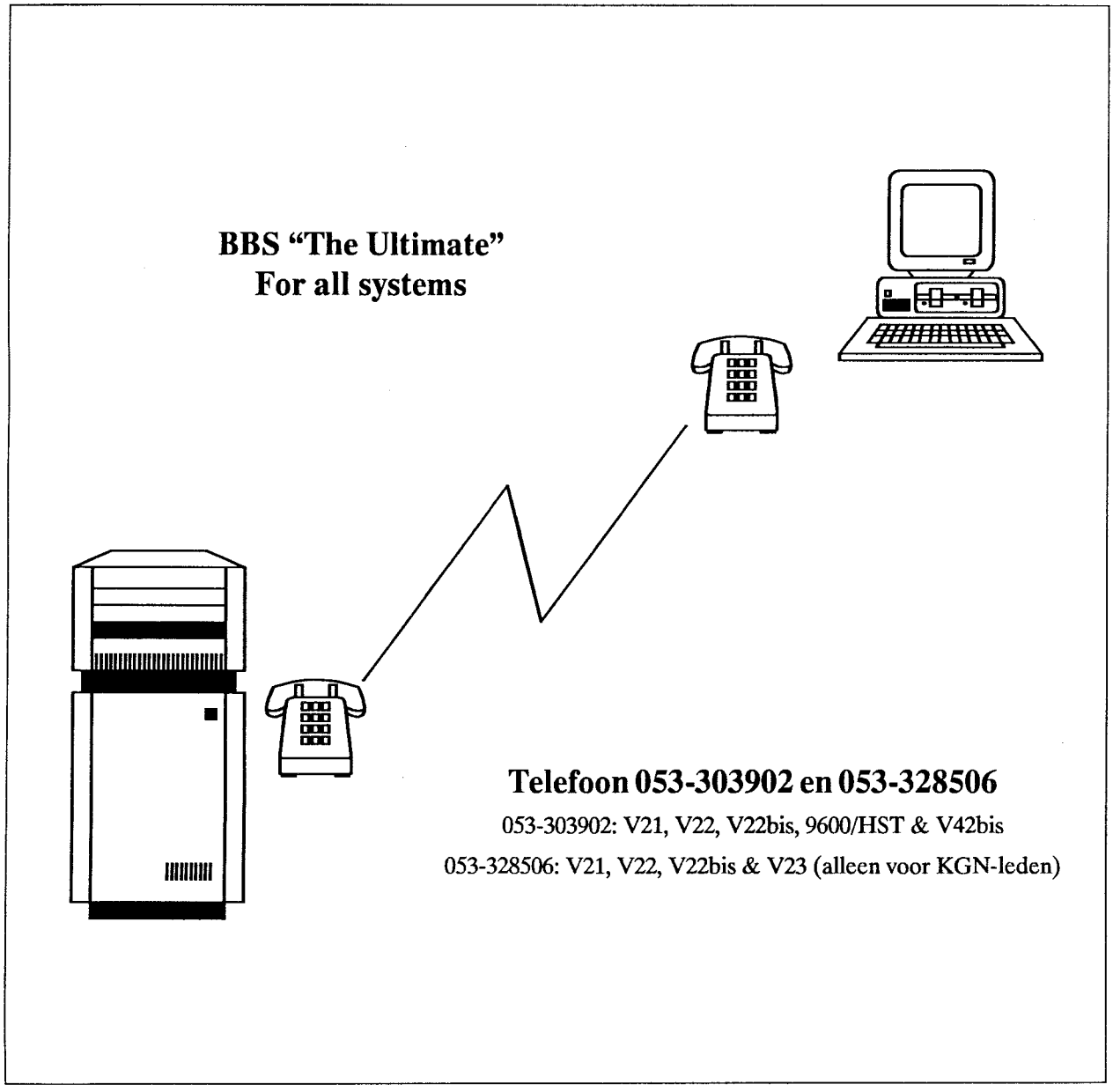
Deze samenwerking bestaat uit de volgende punten:

- De NLMUG wordt in de gelegenheid gesteld maximaal 4 pagina's van de μP Kenner te vullen. De redactionele verantwoordelijkheid blijft echter bij de redactiesecretaris van de KGN.
- Op de normale clubbijeenkomsten (niet de ledenvergaderingen) hebben leden van de NLMUG tegen de normale condities voor leden (fl. 10,--) toegang. Dit geldt uiteraard niet voor ledenvergaderingen die uitsluitend toegankelijk zijn voor leden van de KGN.

- Er wordt een koppeling gerealiseerd tussen ons Bulletin Board The Ultimate en het BBS van de NLMUG waardoor MINIX-informatie via The Ultimate toegankelijk is.
- De leden van de KGN hebben toegang tot de normale bijeenkomsten van NLMUG tegen de condities die ook voor leden van NLMUG gelden.
- Via Fred van Kempen kunnen leden van de NLMUG een abonnement, zonder lidmaatschap op de μP Kenner nemen. De opbrengsten uit deze abonnementen zullen o.a. gebruikt worden voor een uitbreiding van de μP Kenner met 4 pagina's zodat de bijdragen van de NLMUG niet ten koste gaan van de bijdragen van eigen leden. Uiteraard heeft dit tot gevolg dat de financiële clubinformatie (jaarverslag, begroting) niet meer in het blad afgedrukt wordt doch als los blad wordt bijgevoegd.
- Onderzocht wordt of het mogelijk en wenselijk is enkele andere gezamelijke activiteiten (extra gezamelijke bijeenkomsten) te organiseren.

De samenwerking tussen NLMUG en KGN is voorlopig voor een periode van maximaal twee jaar waarna bekeken wordt of en zo ja hoe, de samenwerking voortgezet wordt.

Gert van Opbroek



"Vernieuwd", nu met originele handbediening!

Vorige week vertelde een kennis vol trots dat ze een nieuwe strijkbout gekocht had. Eentje met een computer erin volgens de verkoper. Inderdaad, als je het ding openschroeft zit er een klein stukje electronica in. Een automatische thermostaat die niet werkt met een stukje bi-metaal maar met een warmtegevoelig stukje microelectronica. De hoge heren ontwerpers vonden het nodig er ook nog even een microcontroller in te zetten waarmee het mogelijk is de temperatuur van het ding tot op een tiende graad nauwkeurig in te stellen. Een LCD schermpje heeft de aloude draaiknop met symbooltjes of teksten voor de verschillende soorten wasgoed vervangen en met behulp van een tweetal tiptoetsen stel je de gewenste temperatuur in. Er zit een boekje bij waar bijvoorbeeld precies instaat hoe heet katoenen blouses gestreken moeten worden. Ze zijn alleen vergeten er een cursus "Taiwanees voor beginners" bij te voegen...

De nieuwe foodprocessor van de buren is ook al voorzien van een "computer". Het ding heeft een aantal standen waarmee je op kunt geven of er geraspt, gesneden of gepureerd moet gaan worden. De trotse eigenaar geeft bij ieder bezoek een overweldigende demonstratie die duidelijk aantoont dat het apparaat eigenlijk alleen maar heel erg goed is in het pureren van tomaten.

Is het woord "computer" een extra aanbeveling op huishoudelijke apparaten geworden? KEMA test tegenwoordig huishoudelijke apparaten met behulp van de logic analyzer. Je kunt geen apparaat bedenken of er staat wel "nu computergestuurd" of "automatisch" in de advertentie. Koffiezetapparaten, waterkokers, elektrische tandenborstels en scheerapparaten... "automatisch", "vrij programmeerbaar"! Zelf de stofzuiger ontkomt niet aan de automatisering; gelukkig had ik van tevoren al bedacht dat je hoogpolig tapijt beter niet in de 3000 Watt stand kunt bewerken...

Het effect van deze drift tot modernisering laat zich raden: mijn scheerapparaat deed het vanochtend opeens niet meer. "Syntax error", stond er in het LCD-schermpje. De koffiemachine is inmiddels half uitgefikt ("Overflow error in 35") en de thee is niet meer te drinken ("NMI - Parity error at $1A13, <I>gnore or <Reboot?") Nee, ik heb de oude strijkbout van mijn oma maar weer eens opgesnord. Wel even op het gasfornuis warmstoken voor je kunt gaan strijken, maar nu is het tenminste mijn eigen schuld dat er een gat in mijn smokingblouse zit...

Dr. Scepsis, voormalig antique-handelaar

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
Postbus 99650
1000 NA Amsterdam

i80960: intel's jongste heeft een beetje RISC en veel MIPS

De i80960 is van gisteren, voor vandaag, voor morgen

Ontwikkelingen gaan zo snel niet meer - een nieuwe microprocessor moet duidelijke voordelen over gevestigde technologie hebben om een kans te maken. De i80960 lijkt de intel chip voor de jaren negentig, ontwikkeling ervan begon begin jaren tachtig. Intel nam toen twee strategische beslissingen. Intel besloot, ondanks het succes van de 80x86 familie, een geheel nieuwe, zij het incompatibele, microprocessorarchitectuur te ontwikkelen. Je kunt tenslotte niet op oude successen blijven teren, je wilt niet eeuwig door het verleden achtervolgd worden. Intel besloot twee verschillende 32-bit microprocessors tegelijkertijd te ontwikkelen, de 80386 èn de nieuwe architectuur, de i80960.

De architectuur

De i960 is geen gewone 32-bitter, een aantal eigenschappen maken hem bijzonder. De i960 is ontworpen voor parallellisme. De architectuur is RISC-achtig omdat CISC, vanwege de variabele duur en lengte van instructies minder multi-processing potentieel biedt. De i960 is ontworpen als eerste van een product-lijn die lang software-compatibel zal blijven. De i960 kent meerdere architectuur-niveau's, de FPU is geïntegreerd deel van de architectuur en de CPU kent twee instructiebronnen: het "eigen" programma en boodschappen van andere CPU's. De i960 is geen papieren product, maar was bij aankondiging al twee jaar operationeel. De chip kan op dit moment niet alleen in samples, maar ook in productie-kwaliteit batches geleverd worden.

Je kunt tenslotte niet op oude successen blijven teren, je wilt niet eeuwig door het verleden achtervolgd worden.

De i80960 architectuur heet met recht een architectuur. De i80960 kent drie niveaus, de i80960 Core Architecture, de i80960 Numerics Architecture en de i80960 Protected Architecture. Iedere architectuur is een superset van de voorgaande. Ze bestaan bekend als de i80960 CA, NA en PA. De eerste generatie i80960-familie chips heet de K-serie. Deze serie implementeert alle drie de architecturen:

- De i80960 KA is de eerste implementatie van de Core Architecture. Die lijkt een vrij normale RISC machine. Dit architectuur-niveau definieert een register-georienteerde instructie-set, een register-model en mechanismen voor diverse zaken als interrupts en faults.
- De i80960 KB is de eerste implementatie van de Numerics Architecture. Deze is een extensie van de Core Architecture. De i80960 KB biedt de floating-point support die de i80960 KA geheel mist. De floating-point faciliteiten integreren naadloos met de rest van de architectuur.
- De intel M80960MC implementeert er nog twee flinke scheppen boven op. De Protected Architecture ondersteunt virtueel geheugen (met memory protection en paging) en task- en process-management. De PA biedt secundaire extensies, zoals instructies voor string-processing. Minder opvallend, niet minder belangrijk, is de multiple-processor support.
- Er is een vierde architectuur niveau, de eXtended Architecture (XA). Die is ontwikkeld voor gebruik door intel zelf. Meer is (nog) niet bekend.

Voor- en nadelen

De voordelen van deze multi-level architectuur laten zich raden. Intel's klanten hoeven zo niet te betalen voor meer dan ze nodig hebben. Omdat ieder hoger niveau een superset van het niveau er onder is, is het eenvoudig compilers voor alle architectuurniveaus te maken. De splitsing in meerdere niveaus garandeert een breed toepassingsterrein. Dat klinkt ongelofelijker dan het is - omwille van de flexibiliteit zijn niet alle details van de verschillende niveaus hetzelfde en kunnen sommigen zelfs wijzigen. De architectuur definieert dergelijke aspecten duidelijk. Intel had voor verschillende architecturen in plaats van verschillende niveaus kunnen kiezen. Die zouden dan onderling incompatibel zijn geweest en afzonderlijke ontwikkeltools vereisen. Programmeurs zouden verschillende instructiesets moeten leren. Intel's huidige keus biedt de mogelijkheid i80960 KA applicaties op een i80960 MC te runnen: om een wasmachine-programma op een workstation te testen bijvoorbeeld.

Intel's keuze heeft natuurlijk ook nadelen. Verschillende architectuurniveaus naadloos integreren door niveau B op niveau A te bouwen gaat niet zomaar. De architectuur moest als één geheel, een enkel modulair ontwerp, ontworpen worden. Voor ieder niveau bestaat een afzonderlijke specificatie, maar de lagere niveaus dragen onmiskenbaar sporen van de hogere niveaus. Er zijn "magische getallen", bits die 1 of nul *moeten* zijn, zonder dat ze enige functie ver-

vullen. De verschillende niveaus kennen verschillende implementatie-specifieke faciliteiten. Veel dingen zijn van laagste tot hoogste niveau hetzelfde, een groepje zaken kan van niveau tot niveau verschillen. Zo voorkwam intel onverstandige beperkingen en kan ze nu al waarschuwen waar in de toekomst eventueel veranderingen die in de implementatie kunnen gaan plaatsvinden.

Getallen

De intel i80960 is een 32-bits microprocessor. Er zijn 16, 20, 25 en 33 MHz versies te koop. De 66 MHz versie moet spoedig op de markt zijn en intel zou graag dit jaar nog een 100 MHz (!) versie produceren. De i80960 is met meer dan 200 instructies niet bepaald een RISC, het ontwerp is wel erg "RISCy". Veel instructies worden in één clockcyclus uitgevoerd - 100 MHz betekent dus een peak-performance van niet veel minder dan 100 MIPS! De beestjes kosten een paar honderd dollar per stuk - de KA en KB tenminste. De KA en KB voldoen aan commerciële eisen. De MC voldoet aan militaire eisen (MIL-STD-883) en kost meer dan $1000.

De 66 MHz versie moet spoedig op de markt zijn en intel zou graag dit jaar nog een 100 MHz (!) versie produceren.

Een probleem met RISC-achtige architecturen is dat de MIPS-rating zo weinig zegt. De 20 MHz versies van de i80960 zijn goed voor ongeveer 7.5 "standard MIPS" (VAX-11/780, een klassieke CISC). Dezelfde chips zijn goed voor 14,9 miljoen 32-bits Dhrystones, 4,2 miljoen 32-bits Whetstones of 4,0 miljoen 64-bits Whetstones per seconde. Het prestatie-verlies bij hogere precisie is opvallend gering.

Architectuur overzicht

De i80960 is een 'little-endian' architectuur. Bit 0 is het minst significante bit (LSB), bit 31 is het meest significante bit (MSB). Het minst significante byte van een variabele heeft het laagste adres. De belangrijkste reden voor deze keuze is data-type consistentie met de intel 80x86 architectuur. Ter verduidelijking: Motorola's 680x0 familie is een 'big-endian' architectuur. De VAX-11 serie is little-endian, de IBM S/370 lijn is big-endian.

De i80960 architectuur verwacht operanden op hun natuurlijke boundaries, eist dus alignment. Dat geldt voor zowel de Core als de Numerics Architecture. De Protected Architecture vindt alles best. Voor sommige applicaties is die flexibiliteit gewenst, maar portabiliteit eist dat je, wanneer het maar kan, toch voor alignment kiest.

Bits 0-2 van het Arithmetic Control (AC) register bevatten de conditiecodes (ccode). De conditiecodes worden in drie bits gerepresenteerd om ze niet te hoeven coderen en decoderen. Bit 2 betekent less than, bit 1 betekent equal of true en bit 0 betekent greater than of false.

Alle instructies zijn één woord lang en moeten aligned zijn. Er zijn vijf instructieformaten. Alle instructies hebben een 8-bit opcode. De meeste instructies hebben het zgn. REG formaat. Het REG formaat heeft 4 opcodebits extra, voor een totaal van 12. De i80960 architectuur gaat bewust kwistig met de opcodespace om. Weinig formaten in veel bits stoppen reduceert namelijk de benodigde decoding-time. De instructieset ondersteunt het gebruik van literals 0-31 expliciet en de load-address (LDA) instructie kan worden misbruikt voor het laden van literals 0-4095.

Voor COBR en CTRL, de twee branch formaten, specificeert de architectuur een "branch prediction bit". Het bp-bit kan worden gebruikt voor "static branch prediction". Dit vertelt de processor of een branch te verwachten is - in geval van een loop weet een compiler zoiets van te voren, en daar moet je gebruik van maken. In principe kan de processor dit (in combinatie met pipelining) gebruiken om de gemiddelde branch time te verbeteren. De eerste-generatie chips gebruiken het bit niet en de eerste assembler biedt geen support voor het setten ervan.

De i80960 architectuur staat de processor toe instructies in afwijkende volgorde of tegelijk uit te voeren. Instructie i en i+1 kunnen gelijktijdig of in omgekeerde volgorde uitgevoerd worden wanneer:

1) de resultaten van i en i+1 onafhankelijk zijn
2) instructie i+1 de input-waarden van i niet verandert
3) de inputwaarden van i+1 niet afhangen van de output van i
4) tracing ge-disabled is

De architectuur definieert bepaalde "faults" als "imprecise": de processor zal het adres van de schuldige instructie juist rapporteren, maar de architectuur garandeert *niet* dat opvolgende instructies niet zijn uitgevoerd.

Vier registers, r0-r2 en g15, dienen speciale doelen en zijn daarom niet beschikbaar voor algemeen gebruik. De stack groeit omhoog.

Instructies

De i80960 heeft een RISC-achtige instructie-set. Alle instructies zijn 32 bits lang en vereisen word-boundary alignment. De instructieset is register-georienteerd. Bijna alle instructies zijn register-to-register instructies. Alleen in load-, store- en branch-instructies komen geheugen operanden voor. De meeste instructies zijn 3-register instructies, maar de register-to-register move, load- en store-instructies kunnen vier registers tegelijk aan. De chip ondersteunt *alle* logische operaties, niet slechts de gebruikelijke subset. De meeste instructies decoderen in één clockcyclus èn worden in slechts één clockcyclus uitgevoerd. Dankzij pipelining is de "peak rate" van een 25 MHz processor bijna 25 MIPS. De instructies zijn zó ontworpen dat meerdere instructies tegelijk uitgevoerd kunnen worden. De nog maar net geïntroduceerde 33 MHz i960 heeft een peak performance van 66 MIPS...

De i960 heeft een speciale shift-right instructie, voor het delen door machten van twee, die óók voor negatieve getallen juiste resultaten levert. De chips bieden ook erg veel bit-georiënteerde instructies. Alle niveaus, behalve de Core Architecture, bieden een uitgebreide set floating-point instructies en de Protected Architecture biedt onder andere string-instructies.

De i960 heeft een microcode ROM, maar de meeste instructies worden direct door de hardware uitgevoerd. De microcode is nodig voor complexe functies en instructies, zoals process-management support, inter-communication messages, afhandeling van floating-point special cases (zo heeft de FPU een aantal veel benodigde constanten in ROM), fault generation, initialisatie en de CPU self-test.
De microcode ROM is 3072 42-bits woorden groot. Het instructie-formaat is een superset van dat van de Core Architecture. De microcode heeft beschikking over eigen, extra registers, waar gewone code niet bij kan.

Pipeline

De i960 heeft een vijf-traps pipeline die er ruwweg zo uit ziet:
- fetch instruction i+3
- decode instruction i+2
- queue instruction i+1 for execution
- execute instruction i
- write results instruction i-1

In het beste geval haalt deze pipeline één instructie per clockcyclus. De nieuwste i960 haalt *twee* instructies per clockcyclus. Conflicten tussen de laatste twee stappen van de pipeline worden door een speciaal daarvoor ontworpen circuit opgelost. De conflicten ontstaan vanzelf. Zo kan bijvoorbeeld per cyclus maar één register gelezen en één register geschreven worden. Wil een proces er twee lezen, dan kost dat twee cycli.
De i960 instructie-cache is 512 bytes. De cache is een direct-mapped cache met een 16-byte line size.

Registers

Ieder i80960 programma kan op ieder moment over 32 32-bits registers beschikken. 16 van deze registers zijn globale registers, g0-g15. De andere 16 zijn de lokale registers, r0-r15. De Numerics Architecture voegt daar 4 80-bits FP registers aan toe. De lokale registers zijn lokaal voor procedures; hun waarde wordt beinvloed door call- en return-instructies. Globale registers trekken zich daar niets van aan. De i80960 heeft veel meer dan 32 registers - hoeveel is implementatie-afhankelijk. De 16 globale registers zijn gewone registers. De 16 lokale registers zijn een vormen een klein venster op de uitgebreide verzameling overige registers. Bij een call-instructie wordt dit window opgeschoven - en voilà, 16 nieuwe lokale registers die naar hartelust gebruikt kunnen worden. Een return-instructie schuift het venster weer terug - en voilà, r0 - r15 bevatten hun oude waarden weer. Dit alles gaat zeer snel - totdat de registers op zijn. In dat geval wordt een andere set registers vrijgemaakt voor hergebruik door de registerwaarden te redden in een stack frame. Aangezien de meeste programma's oscilleren tussen 2 en 4 procedure niveaus zal dit zelden nodig zijn. Redden en terughalen van registers is voor de i80960 geen regel, maar uitzondering.

Redden en terughalen van registers is voor de i80960 geen regel, maar uitzondering.

Je kunt de lokale registers als een logisch deel van het stackframe zien. De meeste procedures kunnen al hun lokale variabelen in die registers alloceren. Dat is leuk, maar het maakt een echt stack frame nog niet overbodig - je moet je "linkage information" nog altijd ergens kwijt. Het adres van het voorgaande "stack frame", de oude stack-pointer en de oude instructie-pointer worden bewaard in respectievelijk registers r0 en r2 van de nieuwe registerset en r1 van de voorgaande registerset. Tenzij alle registers in gebruik zijn, hoeft de i80960 voor call- en return-instructies géén geheugenoperaties uit te voeren. Het relatief grote aantal registers staat toe parameters via registers door te geven. Intel's compilers gebruiken waar mogelijk register-parameter passing en gebruiken de stack alleen wanneer het aantal parameters dat noodzakelijk maakt.

De verdwenen addresseermodes

De i80960 architectuur kan een branch en een arithmetische instructie tegelijk uitvoeren. Echt indrukwekkend klinkt dat niet. De PDP-11 kende reeds de SOB, Substract One and Branch. En die machine is de bron van de populariteit die auto-increment mode bij C-programmeurs geniet.

De i80960 kènt auto-increment mode niet eens - de Protected Mode string instructies zijn geen sier, maar bittere noodzaak. Auto-increment mode implementeren op een processor die virtueel geheugen ondersteunt is nogal gecompliceerd en zijeffecten zijn vervelend als je meerdere instructies tegelijk wilt uitvoeren. Wat de i80960 kan is best bijzonder: de i80960 kan een willekeurige branch instructie gelijk met een willekeurige arithmetische instructie uitvoeren.
Ook de "memory deferred" mode (ook wel "memory indirect" genoemd) ontbreekt. Als eerste omdat het zelden worden gebruikt. Ten tweede omdat het instructie decoding aanzienlijk moeilijker gemaakt zou hebben. Ten derde is het langzaam omdat de load/store instructie *twee* memory-accessen moet uitvoeren. Zèlf twee load instructies uitvoeren is sneller - met name omdat je er een derde instructie tussen kan proppen.

De PDP-11 kende reeds de SOB. En die machine is de bron van de populariteit die auto-increment mode geniet.

Floating points

De floating point unit is geen coprocessor, maar een integraal onderdeel van de architectuur. Dat is niet alleen goedkoper, het levert ook een kleine snelheidswinst op. De FPU architectuur voldoet aan de ANSI/IEEE 754 standaard. De FPU gaat verder dan die standaard minimaal vereist. De i80960 bevat vrijwel alle in de standaard gesuggereerde extensies en biedt ook transcendentale functies. De FPU kent drie data typen; 32-bits, 64-bits en 80-bits. De FPU heeft vier 80-bits registers en kan gebruik maken van alle 32 32-bits registers - dat zou een ordinaire coprocessor niet kunnen.

De Numerics Architecture definieert vier additionele data typen: real, long real, extended real, en decimal digit. Het gebruik van het woord "real" in de betekenis van "float" is verwerpelijk, maar we zullen ermee moeten leven. De ontwerpers excuseren zich door te wijzen naar FORTRAN en ALGOL, waarin hetzelfde gebeurt (net als in Pascal).

De NA support wel veel extensies, maar ondersteunt de IEEE floating-point standaard zelf niet volledig. De i80960 hardware implementatie is erg compleet, maar niet echt volledig. Belangrijke afwezigen zijn operaties voor het converteren van en naar decimale formaten. Die moeten in software gemaakt worden.

De extended real support is volledig. De processor kent de (IEEE) bijzondere waarden ∞, minus ∞, NaN (Not a Number), +0 en -0. De wortel uit -0 is -0. De wortel uit -1 is NaN. De 960 support "denormals" niet (dat eist namelijk complexere hardware), maar genereert een fault, die naar believen afgehandeld kan worden - simulatie van denormal-support behoort dus tot de mogelijkheden.

De Numerics Architecture definieert natuurlijk meer AC flags dan de Core Architecture. In de eerste implementaties van de NA is het datapad naar de FP ALU 32 bits wijd. Gebruik van de remainder instructie is niet aan te bevelen: de processor berekent resten door herhaald aftrekken. De rest zelf is een getal tussen 0 en A, zoals je meestal wil, niet tussen -A/2 en A/2, zoals de IEEE standaard specificeert. De processor biedt status info voor eventuele correctie (correctie van de standaard lijkt meer op z'n plaats). De architectuur eist dat alle FP functies monotoon zijn (dus bijvoorbeeld ln(x) < = ln(x+e). voor e > 0 en klein en alle x). Dit eiste zowel "gewoon" testen als mathematische analyse van de hardware en microcode - en die testen zijn uitgevoerd.

Virtueel geheugen

De i80960 heeft een 2^{32} byte virtuele adresruimte. Deze is via two-level page tables verdeeld in 4096-byte pages. De processor beschikt over een Translation-Lookaside-Buffer (TLB). De processor ondersteunt twee privilege niveaus, user mode en supervisor mode. Iedere page-table entry heeft "accessed" en "altered" bits, voor gebruik door page replacement algoritmen. De architectuur definieert ook een "cacheable" flag, die aangeeft of een page al dan niet door een externe cache gecached mag worden.

Interrupts

De architectuur kent verschillende interrupts verschillende prioriteiten toe en onthoudt "hangende" interrupts. De CPU kan aan bijvoorbeeld een 8259 interrupt controller gekoppeld worden, maar heeft

ook een bescheiden interrupt controller ingebouwd. De i80960 faults en interrupts worden op vergelijkbare wijze afgehandeld. Faults mogen "imprecies" zijn, zodat instructies tegelijkertijd of zelfs in afwijkende volgorde uitgevoerd kunnen worden.
De architectuur definieert 256 interrupts, verdeeld over 32 prioriteitniveaus van ieder 8 interrupts. De laagste 8 (prioriteit 0) worden niet gebruikt. De interrupttabel kent twee vectoren, één 32-bits "pending priorities"-vector en één 256-bits "pending interrupts"-vector. De architectuur staat toe dat meerdere processoren één interrupttabel delen en gezamelijk de interrupts afhandelen.
De processor heeft vier interruptpinnen. De betekenis daarvan staat niet vast. Het interrupt control register bepaalt welke vector aan welke pin hangt.

Process management

Een proces, ook wel task genoemd, wordt voorgesteld door een process control block. Het PCB definieert een virtuele adresruimte, de prioriteit en bevat een register save area.
De i80960 ondersteunt twee methoden van process switching, manual en automatic. Manual is door een operating system, automatic is door de i80960 hardware. De CPU biedt daarnaast message-based en semaphore-based interprocess communication instructies.

De Inter-Agent Communication (IAC) faciliteiten zijn onderdeel van de Core Architecture. Ze vervullen een functie vergelijkbaar met instructies. Een proces op één processor kan een boodschap naar een andere processor sturen. Voor de IAC's is een gedeelte van de fysieke adresruimte gereserveerd.
De Protected Architecture definieert een Process Control Block. De adresruimte van een proces bestaat uit drie gebieden (regions). Alleen nul tot en met twee zijn privé. Alle processen hebben toegang tot gebied 3. De CPU heeft een Translation-Look-aside Buffer, die normaal gesproken bij een process-switch gewist wordt - maar niet voor regio 3. Adresvertaling kan uitgeschakeld worden, en dan is er 2^32 bytes fysiek geheugen. Een regio kan in nul, één of twee page table niveaus naar een page gemapped worden. De PA kent send- en receive-instructies.

Tracing , debugging en Multi-processing

De i80960 ondersteunt tracing en debugging via trace-control registers en twee instruction breakpoint registers (handig voor het debuggen van EPROMs). Daarnaast bevat de i80960 nog wat "proprietary" debug support voor gebruik in in-circuit emulatoren (ICEs).
De i960 biedt multi-processing support. Veel daarvan is vrij multi-processing architectuur-onafhankelijk, maar de ondersteuning is enigszins gericht op "tightly coupled, shared-memory" multi-processing systems, waarin processen van processor naar processor kunnen migreren. De processor ondersteunt twintig verschillende inter-processor boodschappen. In de i80960 algoritmen (o.a. de dispatching logic) is rekening gehouden met eventueel gebruik in multi-processing systemen.

De i960 heeft een 32-bits gemultiplexte externe bus met burst mode mogelijkheid. De bus timing specificaties zijn strak, en dat is gunstig voor systeemontwerpers. Het is ongunstig voor intel, die nu meer chips moet afkeuren. De i960 accepteert een vrij slordige klok. Dat is met name prettig voor ontwerpers van multi-processor systemen. De meeste 960-systemen zullen zero-wait-state systemen zijn.

Conclusie

De i960 is heeft krachtige architectuur, waar heel aardige machines omheen gebouwd kunnen worden. Buitengewoon aardige machines zelfs. Intel positioneert de chips als embedded processors, maar in een personal hypercube zou het zeker niet misstaan.

Tamura Jones

Referenties:

1. "The i80960 Microprocessor Architecture", Glenford J. Myers & David L. Budde, John Wiley & Sons Ltd., 1988, ISBN 0-471-61857-8, 255 pagina's
2. "intel Guns for 66 MIPS", Ernest L. Meyer, Embedded Systems Programming, februari 1989, ISSN 1040-3272, 'chip of the month'
3. "More RISCy Business", Bob Hale, Embedded Systems Programming, mei 1989, ISSN 1040-3272, 'Parity Bit'
4. "Math Coprocessors", Hands On, Under the Hood, L. Brett Glass, BYTE magazine, vol. 15, no. 1, januari 1990, p.337-348

i80960 Data typen

integer	. 8, 16, 32, 64 . .
ordinal	. 8, 16, 32, 64 . .
triple-word	. . . . . 96 .
quad-word	 128
bit	1 . . . . .
bit-field	1,2....31,32 . . .

byte: 8-bits, short: 16-bits, word: 32-bits, long: 64-bits
De fundamentele i80960 data typen zijn 32-bit signed en unsigned integers, in two's-complement representatie. De Numerics Architecture definieert vier additionele data typen:
real 32, long real 64, extended real 80 en
decimal digit 8

Van de bestuurstafel

Het jaar is alweer een tijdje op weg en de eerste bijeenkomst is alweer verleden tijd. Op die bijeenkomst is zoals gebruikelijk de tweede algemene ledenvergadering gehouden (de andere is de vergadering in november). Op die tweede ledenvergadering hebben de leden de gelegenheid om het financieel jaarverslag aan de tand te voelen en de penningmeester stekelige vragen te stellen over het beheer van de centjes. De vergadering verliep vrijwel analoog aan de vergaderingen van andere jaren: er werd weer kritisch gekeken en geluisterd, maar uiteindelijk blijkt alles goed voor elkaar te zijn. Wat ook ieder jaar hetzelfde is, zijn de kleine blundertjes in het jaarverslag dat op de vergadering wordt uitgereikt. De penningmeester stopt die dingen er kennelijk in om de oplettendheid onder de leden op peil te houden...

In dit nummer treft u het (uiteraard gecorrigeerde) jaarverslag en het verslag van de ledenvergadering aan. Op een los ingevoegd blaadje. Dat heeft te maken dat het blad ook naar leden van de NLMUG gaat, en die hoeven niets te weten van het interne reilen en zeilen binnen de KGN. Dat geldt natuurlijk andersom ook.

Het feit dat het blad ook aan de NLMUG wordt aangeboden was overigens ook een agendapunt van de ledenvergadering in Krommenie. Het bestuur wilde graag van de leden horen in welke vorm de samenwerking met de NLMUG gegoten moest worden. Een van de besluiten was, dat de NLMUG voor een nummer van de μP Kenner hetzelfde moet betalen als een KGN-lid, en zeer beslist niet minder. Verder is goedgekeurd, dat KGN-leden welkom zijn op NLMUG-bijeenkomsten en andersom, behalve op de officiële gedeeltes zoals bestuursvergaderingen. Verder zal er gekeken worden hoe de samenwerking verder gestalte kan krijgen. De op de vergadering aanwezige leden legden er verder de nadruk op, dat de KGN de KGN moet blijven, of anders gezegd, dat wij als club ons eigen gezicht moeten blijven houden.

Trouwens over het blad gesproken: het decembernummer was niet zoals het wezen moest. De voorzitter heeft hierover de layouter aangesproken, en deze moest dan ook met het schaamrood op de kaken bekennen dat het een en ander veroorzaakt was door een wellicht wat grote tijdsdruk gecombineerd met een wat slappe controle op het eindresultaat. Wat 1 enkele vergeten click op een muisbutton niet teweeg kan brengen... De layouter heeft de voorzitter nederig verbetering in het vooruitzicht gesteld, vergezeld met de excuses voor de puinhoop. Afijn, we zullen zien of het wat wordt met die layouter.

Buiten valt de sneeuw in dikke pakken uit de hemel. Er verschijnen sneeuwmannen en andere witte gedrochten. Auto's schuiven over en schuifelen langs de nauwelijks herkenbare wegen. Alle wateren in Nederland zijn bevroren en er wordt veel en vaak geschaatst. Dat houdt volgens mij in, dat er binnen gewerkt wordt aan erwtensoep, stamppot en andere stevige winterkost, maar ook dat de winchesters weer uitbundig snorren, processors zich zwaar in het zweet moeten werken en dat er weer vele toetsenborden worden versleten, om nog maar te zwijgen van kilometers soldeertin en onnoemelijke hoeveelheden elektronische onderdelen die in de diverse projecten verdwijnen. Kortom, er wordt weer flink geliefhebberd. Dat is te merken: het KGN-68k project loopt nog steeds (zie het verslag in dit nummer), DOS65 lijkt echt een opvolger in de vorm van versie 3.0 te krijgen en er is zelfs een nieuwe DOS65 coördinator gevonden. Er gebeurt derhalve heel wat binnen de club.

Voor je het weet is de bladzijde al weer bijna vol. Zo ook dit maal. Ik hoop dat dit nummer weer het gebruikelijke leesplezier heeft opgeleverd. Veel pret bij al het gesoldeer en geprogrammeer.

De lijder van 't zootje,

Nico de Vries

Informatie

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

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

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

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

Landelijke bijeenkomsten:

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

Bulletin Board:

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

Software Bibliotheek en Technisch Forum:

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

Correspondentie adres

Alle correspondentie betreffende verenigingszaken kan gestuurd worden aan:

KIM Gebruikersclub Nederland
Postbus 99650
1000 NA Amsterdam

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

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

Jan D.J. Derksen(secretaris)
C.P. Soeteliefstraat 41
1785 CC Den Helder
Telefoon 02230-35002

Gert van Opbroek (redactie μP Kenner)
Bateweg 60
2481 AN Woubrugge
Telefoon 01729-8636

Mick Agterberg
Davidvosstraat 29
1063 HV Amsterdam
Telefoon 020-131538

Geert Stappers
Engelseweg 7
5825 BT Overloon
Telefoon 04788-1279

Ton Smits
De Meren 39
4731 WB Oudenbosch

Ereleden:

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

Erevoorzitter:
Siep de Vries

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