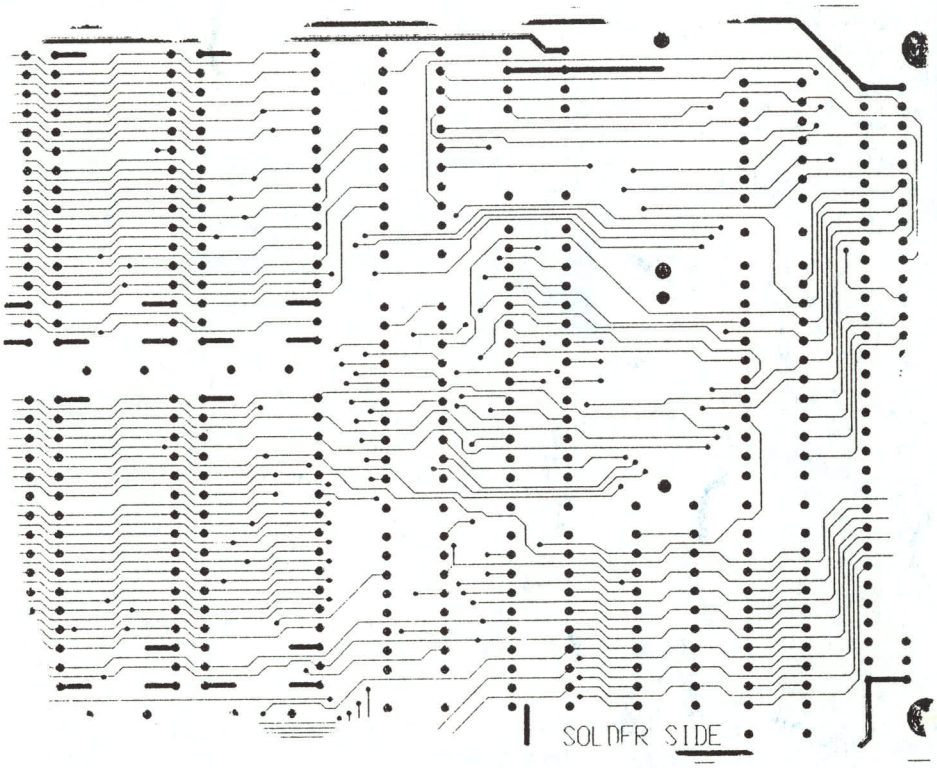
De µP Kenner

Zeventiende jaargang nr. 5
December 1993/Februari 1994
84



In dit nummer o.a.:

DOS65: RAMkaart gereed
Software via Duitse Teletekst
C: still going strong met les 4
Hevig en Nieuw: de Power PC
Simuleren leidt niet tot afkeuring

MIDI op de bijeenkomst in Geldrop

Elders in dit blad vind je de uitnodiging voor de clubbijeenkomst in Geldrop op 19 maart aanstaande. Op deze bijeenkomst gaat een oude wens van mij in vervulling, namelijk dat we binnen de KGN ook eens een keer iets over MIDI gaan doen.

MIDI is de afkorting van Musical Instrument Digital Interface. Dit is een standaard-manier om (digitale) muziek-instrumenten aan elkaar en/of aan een computer te koppelen. Je moet dan vooral denken aan instrumenten zoals Keyboards, Synthesizers en Drumcomputers. Deze apparaten communiceren met een soort opgevoerde RS-232 interface met elkaar of met een computer waardoor je verschillende mogelijkheden hebt. Zo kun je bijvoorbeeld via een keyboard een stuk muziek inspelen dat digitaal op de computer wordt opgeslagen. Vervolgens ga je met een bewerkingsprogramma (muziekverwerker naar analogie van een tekstverwerker?) de fouten en foutjes uit het stukje muziek halen waarna je het weer door je keyboard of door een synthesizer af kunt laten spelen. Ook kun je aan dit stukje een tweede, derde enzovoort partij toevoegen waardoor je een complete partituur voor een orkestje krijgt. Deze partituur kun je uitprinten of af laten spelen door een synthesizer. Op deze manier komt de muziek van bijvoorbeeld Vangelis of Jean Michel Jarre tot stand.

Het bovenstaande is zo'n beetje alles wat ik van MIDI weet; veel te weinig om op een bijeenkomst te vertellen. Bovendien heb ik wel het plan om zelf met MIDI bezig te gaan, maar ik bezit de benodigde apparatuur nog niet. We zijn daarom zeer gelukkig met het feit dat we **Robert de Jong** van de firma **Technotronics** in Wijk bij Duurstede bereid hebben gevonden een voordracht met demonstratie over MIDI te komen houden. Hij doet dit geheel belangeloos zodat ik vind dat we als club een tegenprestatie moeten leveren door massaal naar de bijeenkomst te gaan.

De firma Technotronics is gespecialiseerd op het gebied van MIDI- en Multimediasystemen zodat mijn verwachtingen hooggespannen zijn. Uiteraard kunt u één of meer gasten meebrengen tegen een toegangsprijs van fl. 10,-- per persoon. Voor clubleden is de toegang, zoals gebruikelijk, gratis.

Ik hoop velen van jullie op zaterdag 19 maart te ontmoeten en mocht je zelf iets te tonen hebben, schroom niet om het mee te brengen.

Groetjes,

Gert van Opbroek

Inhoudsopgave

De μP Kenner

Nummer 84, december 1993/feb. 1994
Verschijnt 5 maal per jaar
Oplage: 250 stuks
Druk: FEBO Offset, Enschede

De redactie:

Nico de Vries
Gert van Opbroek
Geert Stappers

Eindredactie:

Nico de Vries

Vormgeving:

Nico de Vries

Redactieadres:

p/a Nico de Vries
Van der Waalsstraat 46
2984 EP Ridderkerk

De μP Kenner nummer 85 verschijnt op 16 april 1994.

Kopijsluitingsdatum voor nummer 85 is vastgesteld op 2 april 1994.

Vereniging

Algemeen

Talen/software

Hardware/DOS65

Systemen

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

Wel eens verhuisd? Vast wel. Ondergetekende dus ook. Van Capelle aan den IJssel naar Ridderkerk. Van een gehuurd flatje naar een optrekje in eigendom. Met meer ruimte. En meer mogelijkheden. Een ieder die ook wel eens is verhuisd, weet dat je een tijd (of een tijdje) in de rommel zit. En dat het veel moeite kost om alles weer op rij te zetten. Op het moment dat deze letters worden neer-getekstverwerkt gebeurt dat in Ridderkerk. De PC staat inmiddels, na drie weken, op zijn plek. In een verder nog vrijwel lege kamer. Het beeldscherm, de muis, het modem en printer zijn ook van de partij. Het toetsenbord is terug van weggeweest: het had de verhuizing niet overleefd. De toetsen F6, 8, I, K en de komma waren in staking. Bij de huidige toetsenbordprijzen luidt het devies dan meestal: kieper maar in de vuilnisbak. Maar dit is een toetsenbord van een zeer bekend merk, dat bekend staat om zijn degelijkheid en de klik bij de aanslag. Ik wilde het niet kwijt. Eenzelfde toetsenbord nieuw kost *f* 450,-. Repareren dus, en dat viel niet mee: er bleek een breuk te zitten in een stukje flexibele print, waardoor 1 rij toetsen het niet meer deed. Een geleidende lijm bleek de oplossing: want inmiddels staat er alweer een flink aantal I's, K's en komma's in dit verhaaltje.

Met de verandering van woonplaats was het verhuizen nog niet gedaan. Wat ook verhuisd is, is het voorzitterschap: Tonny Schäffer eruit, Geert Stappers erin. Geert blijft voorlopig ook nog de KGN-68k kar trekken. Onderaan dit verhaal is te zien dat dit niet de enige verhuizing binnen het bestuur is geweest: ondergetekende is er weer ingestonken en heeft het redacteurschap overgenomen van Gert van Opbroek. Dit zat al langer in de pen, want Gert is ook nog secretaris en is ook verhuisd, met andere woorden: hij heeft het ook stervensdruk. Dus is het redacteurschap van zijn schouders gehaald. In de praktijk maakten Gert en ondergetekende het blad toch al in onderlinge tegenwerking zodat er effectief niet zoveel verandert. Behalve dan dat ondergetekende nu het redactioneel mag bedenken. Gert: bedankt voor de uitbundige medewerking tot dusverre.

Hierboven staan tussen de regels door een aantal redenen waaraan het blad zo laat is deze keer. Want het had omstreeks de afgelopen Kerstdagen bij de lezers op de mat moeten liggen. Dit is duidelijk niet gelukt. De meeste oorzaken zijn hierboven al genoemd. De belangrijkste oorzaak echter nog niet.

Het wordt eentonig, maar het is en blijft de waarheid: we hebben zoals altijd eigenlijk al veel te weinig kopij om het blad mee te vullen. Vooral Gert van Opbroek wist dat heel aardig op te vangen door bij vlagen het blad zo ongeveer in z'n eentje vol te pennen. Maar hij begint last te krijgen van hetzelfde verschijnsel waar ondergetekende ook een beetje last heeft: de leuke onderwerpen zijn een beetje op. En dat zal zich helaas wreken in dunnere nummers dan u van ons gewend bent. We blijven streven naar 48 redactionele pagina's per aflevering, vijfmaal per jaar. We blijven echter ook streven naar een beetje kwaliteit. Het blad zal dus niet gevuld worden met rommel om het toch maar vol te krijgen.

We blijven streven naar 48 redactionele pagina's per aflevering, vijfmaal per jaar.

Wat voorlopig blijft is de cursus C: er liggen na dit nummer nog twee afleveringen op de plank. Ook KGN-68k en DOS65 zullen pagina's leesvoer blijven opleveren, want op beide fronten is beweging waar te nemen. Zo zal in het volgende nummer, dat hopelijk wat vlotter bij een ieder zal arriveren de 1 Mbyte RAM-kaart voor DOS65 gepresenteerd worden, want inmiddels werken er twee prototypen die op de proefprinten gebouwd zijn naar behoren en volgens verwachting. De soldeerbouten kunnen volgende maand dus weer uit de kast.

Rest mij een ieder vanaf deze plaats een voorspoedig 1994 toe te wensen. Weet u een leuk verhaaltje? Iets aardigs op computergebied meegemaakt? Een bijzonder programma gevonden? Ontdekking gedaan op het BBS? Schrijf ze en laat het merken!

Nico de Vries

Uitnodiging clubbijeenkomst

Datum: 19 maart 1994
Lokatie: gebouw 't Kruispunt
Slachthuisstraat 22
5664 EP Geldrop
Tel: 040-857527
Thema: Muziek m.b.v. de computer en MIDI

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; tot 11 uur informele informatie-uitwisseling

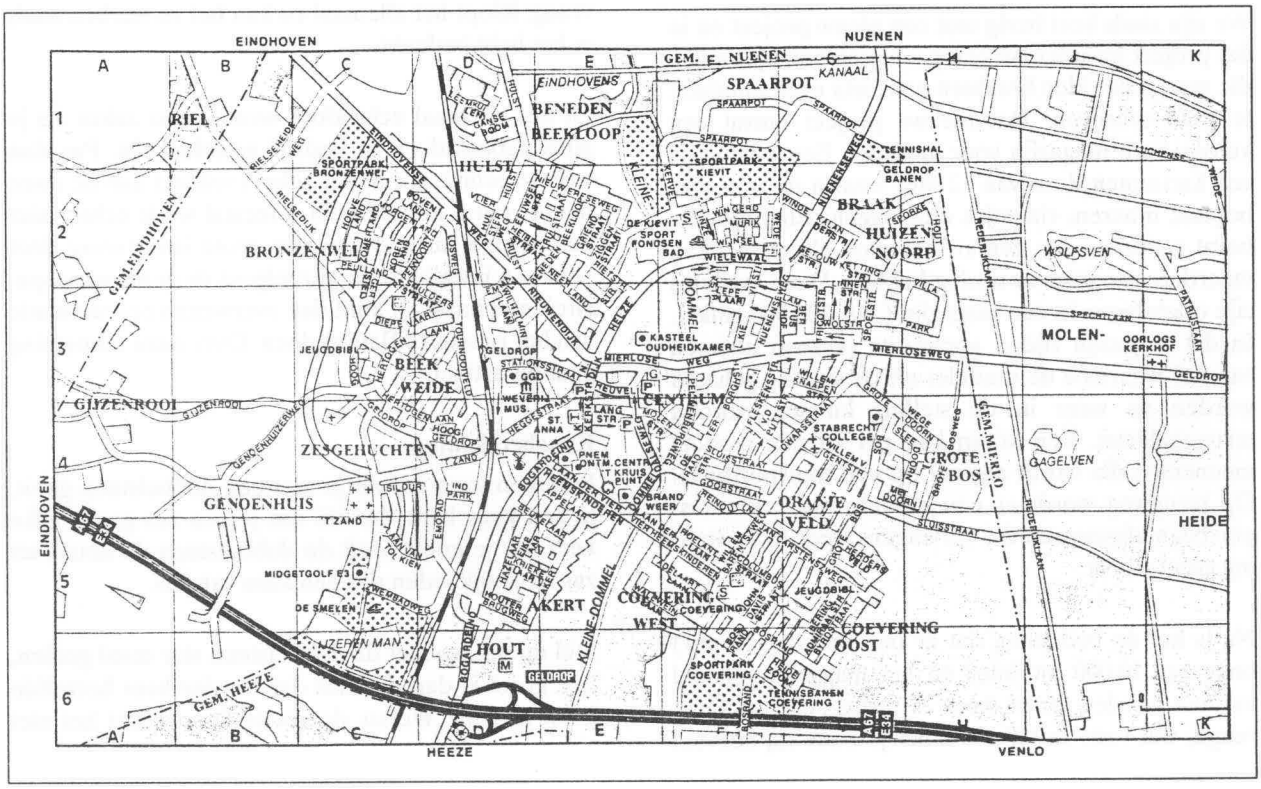
11:00 Algemene ledenvergadering met als agenda:
- Opening/vaststelling agenda
- Verslag ledenvergadering d.d. 25-09-1993 in Almelo (zie μP Kenner 83)
- Behandeling jaarverslag 1993
- Rondvraag

12:30 Lunchpauze

14:00 Forum en Markt

14:15 Voordracht en demonstraties over MIDI. Hiervoor hebben we **Robert de Jong** van de firma **Technotronics** bereid gevonden een voordracht met demonstratie te geven over MIDI.
Ben je zelf met MIDI bezig, breng dan ook wat spullen mee en demonstreer ze aan je clubgenoten.
Aansluitend het informele gedeelte met de mogelijkheid om andermans systemen en de ontwikkelingen in de werkgroepen te bewonderen en Public Domain software uit te wisselen.

17:00 Sluiting.



Simulatie: Modelbouw op de computer

Inleiding

Zoals de meeste lezers wel weten, werk ik als automatiseerder in de logistiek. Dat doe ik nu al ruim acht jaar, eerst bij een software-bureau en sinds een jaar bij een groothandel in bevestigings-materialen. Nu is de logistiek een heel breed vakgebied. Alles wat te maken heeft met het transporteren en de opslag van goederen wordt logistiek genoemd. Een heel duidelijk voorbeeld van logistiek is Van Gent en Loos. Deze firma leeft als het ware van de logistiek; ze transporteert goederen van het ene adres naar het andere.

Waar ik me vooral mee bezig houd, is het opslaan van goederen in één of meer magazijnen en alle (interne) transport die daar mee samenhangt. Goederen moeten, nadat ze bij mijn werkgever aangeleverd zijn, worden geïdentificeerd (uitzoeken wat het is) en opgeslagen worden in een magazijn. Komt er een opdracht van een klant, dan moeten de goederen weer uit het magazijn gehaald worden en verzend-gereed gemaakt worden. De automatisering waar ik me mee bezig houd begint vlak voordat de goederen bij mijn werkgever aangeleverd worden en eindigt als de goederen overgedragen zijn aan de klant of de vervoerder die de goederen naar de klant brengt.

We kwamen paar probleempjes tegen die me op het idee brachten eens iets over simulatie te gaan schrijven.

We zijn sinds kort bezig met een nieuw project en in dat project kwamen we een paar probleempjes tegen die me op het idee brachten eens iets over simulatie te gaan schrijven. Dat nieuwe project omvat een automatisch magazijn voor modules. Een module is een kartonnen doos van 12 liter waarin doosjes met bouten, moeren, ringetjes enz. liggen. Elke module bevat goederen van één artikel maar elk artikel kan meerdere modules in opslag hebben. Deze modules zijn opgeslagen in een magazijn (+/- 200.000 stuks). In dat magazijn rijden apparaten (kranen noemen we die) waarmee de modules uit de stelling gehaald worden en weer in de stelling kunnen worden teruggeplaatst. Een kraan kan per rit één module meenemen die wordt afgezet op een transportbaan. Op terugweg wordt er een module van een transportbaan afgenomen die vervolgens weer in de stelling gezet wordt.

Nu is het de bedoeling dat er per werkdag (9 uur) ongeveer 10.000 modules uit het magazijn gehaald kunnen worden en er weer in teruggeplaatst. Gevraagd, wat voor installatie moet je bouwen, hoeveel kranen heb je nodig, hoeveel mensen om de goederen uit de modules te halen enzovoort enzovoort. Vragen waarop je niet "zomaar" een antwoord kunt geven. Toch zijn dit geen onbelangrijke vragen. Een ontwerpfout kan zeer ernstige gevolgen hebben. Blijkt de werkelijke capaciteit hoger te liggen dan je nodig hebt, dan heb je onnodig veel geïnvesteerd (meerdere tonnen per kraan!), blijkt echter de capaciteit te laag, dan heb je pas echt een probleem. Je krijgt de hoeveelheid goederen die jouw klanten willen afnemen niet binnen de normale werkuren uit het magazijn. Dit betekent dus overwerk, ploegendiensten en misschien wel het voortijdig vervangen van delen van de installatie of nog erger: het verlies van belangrijke klanten omdat je niet binnen de afgesproken termijn kunt leveren.

Voor het ontwerpen van een dergelijke installatie maak je uiteraard gebruik van alle ervaring die je binnen kunt halen. Hiermee kun je dan een soort globaal ontwerp van je installatie maken. Vervolgens ga je kijken naar de capaciteit van je transportmiddelen en je maakt een schatting hoe snel de mensen in je systeem bepaalde handelingen kunnen verrichten. Op grond hiervan probeer je zo nauwkeurig mogelijk in te schatten hoeveel mensen en transportmiddelen je nodig hebt. Maar dan komt de hamvraag: Klopt het allemaal en kan het zo werken zoals je het hebt bedacht.

Of het allemaal echt klopt weet je pas zeker als je alles gebouwd en in bedrijf gesteld hebt. Pas dan weet je echt zeker of het geheel voldoet aan de eisen die je vooraf gesteld hebt. Meestal wil je echter niet het risico nemen dat je een grote investering doet voor een installatie die vervolgens de geplande capaciteit niet haalt. Je kunt dan overwegen een simulatie van het proces te (laten) doen. Over deze toepassing gaat dit artikel.

Kansberekening

Iedereen weet dat als je met een dobbelsteen gooit, je een kans hebt van 1/6 dat je een zes gooit. Elke keer dat je gooit heeft de dobbelsteen de keuze uit zes mogelijkheden dus een kans van 1/6.

Stel nu dat we met de dobbelsteen vier maal gooien, hoe groot is dan de kans dat we vier keer hetzelfde getal gooien? Welnu, de eerste keer maakt het niet

uit. Deze kans is dus 6/6 of 1. De tweede keer hebben we een kans van 1/6 om hetzelfde getal te gooien als de eerste keer. Hetzelfde geldt voor de derde en vierde keer gooien zodat de kans op vier keer hetzelfde getal wordt:

P = 6/6 * 1/6 * 1/6 * 1/6.

Willen we de kans weten dat we in zes beurten (of met zes dobbelsteen in één keer) zes verschillende getallen gegooid hebben, dan geldt voor de eerste steen 6/6 (tenslotte kan deze steen nog vrij kiezen). De tweede steen heeft vijf mogelijkheden (dus 5/6) etc. De kans op zes verschillende getallen in zes beurten wordt dus:

P = 6/6 * 5/6 * 4/6 * 3/6 * 2/6 * 1/6.

Een dergelijk sommetje kun je nog wel uitschrijven. Stel nu dat je gevraagd wordt te bepalen hoe vaak je moet gooien om er voor meer dan 95% zeker van te zijn dat je alle zes getallen een keer gehad hebt? Na twee dagen over een dergelijk probleem gerekend te hebben en bladzijden vol met (onjuiste) formules te hebben opgeschreven, heb ik het anders aangepakt.

Simulatie

Als je er met simpel redeneerwerk of met de statistiek-boeken die je onder handbereik hebt niet meer uitkomt, dan kun je natuurlijk altijd nog met een dobbelsteen gaan gooien. Stel je gooit 10 keer met een dobbelsteen. Hoeveel verschillende getallen heb je dan gehad en hoeveel keer heb je hetzelfde getal gegooid. Dat kun je opschrijven. Doe je dat een tweede keer, dan zal er waarschijnlijk een ander antwoord uitkomen. Doe je het echter veel keer (enkele duizenden keren), dan zal het aantal keren dat je in tien worpen zes verschillende getallen gegooid hebt gedeeld door het aantal keren dat je tien keer met de dobbelsteen hebt gegooid vrijwel de kans zijn dat je in tien worpen zes verschillende getallen gooit. Deze regel wordt vaak de wet van de grote getallen genoemd.

Nu kun je natuurlijk echt duizenden keren met een dobbelsteen gaan gooien, maar beter is het daarvoor een slaafje te nemen, een computer-programma dus. Dit noemen we "simulatie". We bouwen in een computer-programma een model van de werkelijkheid en doen daarmee experimenten waarvan we de uitkomsten statistisch verwerken. In figuur 1 staat een programma in Turbo C voor het beschreven probleem van de dobbelsteen.

```
/* Programma dobbelsteen-simulatie
   Probleemstelling:

   - Zoek door middel van simulatie een antwoord op de volgende vraag:
                Ik gooi N maal met een dobbelsteen. Hoe groot is de kans dat ik X
                (0 < 1 < = X) verschillende getallen gooi.

                Copyright (c) 1993: G. van Opbroek

                Versie 1.0    d.d. 20 december 1993
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <conio.h>

#define CIJFERS          6                  /* Zes verschillende cijfers          */
#define MAX_RUNS         100000             /* Maximaal aantal experimenten       */
#define MAX_WORPEN       250                /* Maximaal aantal worpen             */

long int aantal_versch[CIJFERS + 1];        /* Aantal verschillende cijfers       */
long int aantal_worpen[MAX_WORPEN + 1];     /* Aantal keren dezelfde              */

long int N;                                 /* Aantal worpen                      */
```

Fig. 1: simulatieprogramma voor het gooien met dobbelstenen

```
long int number_of_runs;                    /* Aantal runs                  */
long int actuele_run;                       /* Actuele run                  */

void global_initialize()
{ int i;
 randomize();                               /* initialiseer random numbers  */
 clrscr();                                  /* maak het scherm schoon       */
 for (i = 0 ; i < CIJFERS ; i++)
          aantal_versch[i] = 0;
 for (i = 0 ; i < MAX_WORPEN ; i++)
          aantal_worpen[i] = 0;
}

void get_input()
{ char s[100];

 clrscr();
 printf("Simulatie-programma voor het gooien met een dobbelsteen\n");
 printf("\n");
 printf("Auteur: G. van Opbroek BLP\n");
 printf("Versie: 1.0 datum 20 december 1993 \n\n");
 do
 { N = -1;
          printf("\nGeef het aantal worpen : ");
          if (scanf ("%D",&N) == 0)
           scanf("%s",s);
          if (N <= 0 || N > MAX_WORPEN)
           printf("\n0 < N <= %d, N = %d\n",MAX_WORPEN,N);
 } while (N <= 0 || N > MAX_WORPEN);
 do
 { number_of_runs = -1;
          printf("\nGeef het aantal runs  : ");
          if (scanf ("%D",&number_of_runs) == 0)
           scanf("%s",s);
          if (number_of_runs <= 0 || number_of_runs > MAX_RUNS)
           printf("\n0 < runs <= %d, runs = %d\n",MAX_RUNS,number_of_runs);
 } while (number_of_runs <= 0 || number_of_runs > MAX_RUNS);
}

void print_results()
{ int i, som;

 gotoxy(1,1);
 cprintf("Resultaten na %6d runs : \r\n\n",actuele_run);
 cprintf("Aantal worpen : %3d\r\n",N);
 cprintf("\r\n\nAantal verschillende cijfers:\r\n");
 for (i = 0; i <= CIJFERS; i++)
 { cprintf("%3d : %4.1f %  ",i,100.0*aantal_versch[i]/actuele_run);
          if (i % 5 == 4)
           cprintf("\r\n");
 }
 cprintf("\r\n");
 cprintf("\r\n\nAantal worpen per cijfer:\r\n");
 for (i = 0,som = 0; i <= N && som <= 990; i++)
 { cprintf("%3d : %4.1f %  ",
          i,100.0*aantal_worpen[i]/(actuele_run*CIJFERS));
          if (i % 5 == 4)
```

```
                cprintf("\r\n");
                som += (int) ((1000.0*aantal_worpen[i]/(actuele_run*CIJFERS) + 0.5));
    }
}

void main()
{ int  verdeling[CIJFERS], i,
                worp,
                aantal;

  get_input();
  global_initialize();
  for (actuele_run = 1; actuele_run <= number_of_runs; actuele_run++)
  {
                /* initialiseer de run                          */

                for (i = 0; i < CIJFERS ; i++)
                 verdeling[i] = 0;

                /* verdeel de N worpen random over de cijfers       */

                for (worp = 0 ; worp < N ; worp++)
                { i = random(CIJFERS);                  /* Geeft 0 <= i < 6                 */
                 verdeling[i]++;
                }

                /* Einde van de run, neem de resultaten over        */

                aantal = 0;

                for (i = 0 ; i < CIJFERS ; i++)
                { if (verdeling[i] )                    /* Indien > 0 dan tellen            */
                                aantal++;
                  aantal_worpen[verdeling[i]]++;
                }
                aantal_versch[aantal]++;

                /* En druk de voorlopige resultaten af   */

                print_results();
   }
}
```

Laten we het programma eens wat nauwkeuriger bekijken. Naast de normale #defines <stdio.h> en <stdlib.h> zijn ook <time.h> en <conio.h> opgenomen. De eerste is nodig voor de random number generator die in het programma gebruikt wordt en de tweede bevat de zogenaamde console I/O routines uit Turbo C. Wil je het programma door een andere C compiler laten vertalen, dan zul je dus enkele dingen aan moeten passen.

De constanten die gedefinieerd worden spreken voor zich. Kom ik bij de twee arrays. In de eerste (aantal_versch) wordt bijgehouden hoe vaak er 0, 1, 2 etc. verschillende getallen in een experiment voorgekomen zijn. In de tweede array (aantal_worpen) wordt bijgehouden hoe vaak het voorkomt dat in een worp een cijfer 0 keer, 1 keer etc. gegooid wordt. De rest van de globale variabelen spreken voor zich.

In de functie global_initialize() wordt de random number generator gestart en worden de globale arrays op nul geïnitialiseerd. Vervolgens wordt de invoer ingelezen met behulp van de functie get_input(). De functie print_results() wordt ge-

```
Resultaten na  10000 runs :
Aantal worpen :  10

Aantal verschillende cijfers:
0 : 0.0 %    1 : 0.0 %    2 : 0.0 %    3 : 1.9 %    4 : 19.7 %
5 : 50.6 %   6 : 27.9 %

Aantal worpen per cijfer:
0 : 15.9 %   1 : 32.5 %   2 : 29.2 %   3 : 15.5 %   4 : 5.3 %
5 : 1.3 %    6 : 0.0%
```

Fig. 2: resultaat van DOBBEL.C

bruikt om de resultaten af te drukken. Deze routine wordt na elke run aangeroepen. De resultaten worden uitgedrukt in procenten.

In het hoofdprogramma vindt de eigenlijke simulatie plaats. We hebben een array "verdeling" die loopt van 0 tot 6. Aangezien ook de random number generator in de functie random(x) een resultaat 0 < = resultaat < x geeft, hebben we dus een dobbelsteen met de cijfers 0 t/m 5. Voor het resultaat is dit uiteraard niet van belang. De rest van de simulatie bestaat uit twee lussen. De buitenste lus telt het aantal runs of experimenten af, de binnenste lus gooit een aantal keren met de dobbelsteen. In verdeling wordt steeds bijgehouden hoe vaak een bepaald cijfer gegooid is. Na afloop van de run worden de resultaten overgenomen in de globale variabelen en wordt het voorlopige resultaat afgedrukt.

In figuur 2 zijn de resultaten van een simulatie afgedrukt. We zien dat bij 10 worpen per beurt (of als we met 10 dobbelstenen gooien) we in ongeveer 28 % van de gevallen zes verschillende cijfers gooien. Verder zien we dat in ongeveer 30% van de gevallen een cijfer twee maal voorkomt.

Nog wat spelen met het programma leert dat bij 27 worpen in een beurt de oorspronkelijke doelstelling dat in minstens 95 % van de beurten alle zes cijfers gegooid worden gehaald wordt. Volgens de wet van de grote getallen komt dit overeen met 95 % waarschijnlijkheid dat alle zes de cijfers voorkomen.

Nog één kanttekening bij het programma. De betrouwbaarheid van het resultaat is er sterk van afhankelijk dat de random number generator als een "eerlijke" dobbelsteen werkt. Dat wil zeggen dat de kans op elk cijfer evengroot is en dat er geen herhalende reeksen voorkomen. Om te bepalen of aan deze voorwaarden is voldaan, is uitgebreid onderzoek nodig en dat is voor Turbo C door mij niet gedaan. Als we met een eerlijke dobbelsteen gooien, dan is de kans op elk cijfer evengroot. In dat geval zeggen we dat we te maken hebben met een uniforme verdeling.

Een wachtrij probleem

Laten we nu eens een wat moeilijker probleem kiezen. Stel, we hebben een winkel en op zaterdag komt daar gemiddeld één klant per 3 minuten binnen. Nu komen die klanten uiteraard niet precies om de 3 minuten, nee, de kans dat er in een minuut een klant binnenkomt is 1/3 en de kans dat er in een bepaalde minuut geen klant binnenkomt is 2/3. Op deze manier vertoont de tijd tussen twee klanten een bepaalde spreiding. Deze spreiding noemt men een binominaal-verdeling.

We hebben in de winkel 3 bedienden. Elke bediende kan uiteraard maar één klant tegelijk bedienen. Nu is het uiteraard niet zo dat elke klant evenlang werk heeft, nee ook deze tijd vertoont een verdeling. Laten we zeggen dat de helft van de klanten binnen 5 minuten (= 300 seconden) nadat hij aan de beurt is de winkel weer verlaat. Na 10 minuten heeft driekwart de winkel weer verlaten etc. Een dergelijke verdeling noemt men een negatief-exponentiële verdeling. Vraag: hoe lang moeten klanten gemiddeld wachten voordat ze aan de beurt zijn.

Om dit probleem op te lossen heb ik het programma uit figuur 3 (zie achter dit artikel) geschreven. Dit programma simuleert het bovengenoemde wachtrijprobleem. Een voorbeeld van de resultaten van dit programma staat afgedrukt in figuur 4.

We zien in figuur 4 dat er om dinsdag (na 1 dag) om 10:11 186 klanten de winkel binnengekomen zijn. Verder zien we dat er 182 klanten de winkel weer

```
Resultaten op tijdstip : 1 10:11    De winkel is : Open

Totaal aantal klanten
  Binnengekomen          :     186
  Vertrokken             :     182
  In de winkel           :       4
  In behandeling         :       3

Gemiddelde wachttijd     : 7.81 minuten  Maximale wachttijd :  29 minuten
```

Fig. 4: resultaat van WACHTRIJ.C

hebben verlaten. Aangezien we drie bedienden hebben, staat er dus één klant te wachten. Gemiddeld moest men tot nu toe 7.81 minuten wachten en de langste wachttijd is 29 minuten. Nu zijn 186 klanten nog geen aantal waarmee je goed statistiek kunt bedrijven. Laten we het programma langer lopen, dan zullen deze resultaten nog wel iets veranderen.

Ik wil ook nog een paar dingen zeggen over het programma zelf. De functies global_initialize(), get_input() en print_results() hebben dezelfde functie als in het programma dobbel. De functie winkel_open() geeft voor een bepaald tijdstip aan of de winkel open of dicht is.

De kern van de simulatie zit in de functies aankomst() en bepaal_actieftijd(). De eerste functie bepaalt met behulp van de random number generator of er in de afgelopen minuut een klant binnengekomen is. Hiervoor wordt er een willekeurig getal tussen 0 en 100 bepaald die vervolgens wordt vergeleken met de parameter "kans". Is het willekeurige getal groter dan of gelijk aan "kans", dan is er zogenaamd een klant binnengekomen.

De functie bepaal_actieftijd() bepaalt met behulp van de random number generator en de globale variabele "lambda" hoeveel tijd deze klant nodig heeft om geholpen te worden. Hierbij is het minimum op 1 minuut gesteld. Bovendien is er een maximum ingesteld door bij het willekeurige getal tussen 0 en 10000 de waarde 1 op te tellen. Zou dit namelijk niet worden gedaan, dan krijgen voor het random getal 0 een overflow op de logaritme.

In het hoofdprogramma (functie main()) wordt de sturing verricht. Voor maximaal vijftig klanten die wachten wordt bijgehouden wanneer de klant binnengekomen is en hoe lang de behandelingsduur van deze klant is. Verder wordt voor elke bediende bijgehouden of hij iemand helpt, hoe lang hij al met deze persoon bezig is en hoe lang hij met de persoon bezig zal zijn. Komt er een bediende vrij en staan er klanten te wachten of is er een bediende vrij en komt er een klant binnen, dan wordt de klant die aan de beurt is door de vrije bediende geholpen en schuiven alle klanten in de wachtrij één plaats op. Bovendien wordt dan vastgelegd hoe lang de klant die aan de beurt is heeft staan wachten.

Om de simulatie te laten lopen, wordt de tijd in stapjes van 1 minuut verhoogd.

Uitbreidingen

Nu zijn de gepresenteerde programma's nog zeer eenvoudig. Bovendien wordt in het programma "wachtrij" slechts één van de interessante gegevens bekeken (de wachttijd). Om een besluit over de inzet van personeel te kunnen nemen, zijn er meer gegevens interessant. Zo zou je bijvoorbeeld uit kunnen zoeken hoeveel procent van de tijd je een vrije bediende hebt of te wel wat de bezettingsgraad van je personeel is. Verder kun je uitzoeken hoeveel tijd na sluitingstijd je nog bezig bent de aanwezige klanten af te werken. Tenslotte zullen de functies aankomst() en bepaal_actieftijd() niet het werkelijke gedrag van klanten laten zien; hier kun je dus ook betere functies in gaan zetten of praktijk-gegevens invoeren. Kortom, mogelijkheden te over.

Wil je zelf wat met de programma's stoeien, dan hoef je ze niet in te tikken. Op hert moment van verschijnen van de μP Kenner zijn de programma;s ook beschikbaar op The Ultimate.

Professionele toepassingen

Ik heb al aangegeven dat simulatie vrij vaak ingezet wordt bij het analyseren van goederentransport. Hiervoor worden meestal geen C programma's gemaakt. Voor dergelijke toepassingen zijn kant en klare pakketten beschikbaar. Een deel van deze pakketten bestaat uit een zogenaamde simulatie-taal met behulp waarvan een programma op eenvoudige

wijze ontwikkeld kan worden. Een ander deel van de pakketten biedt een soort kant en klare blokkendoos met behulp waarvan je een computermodel van het proces kunt bouwen.

Verder bieden alle pakketten uitgebreide mogelijkheden voor het statistisch verwerken van de resultaten. Een deel van de pakketten biedt bovendien de mogelijkheid een animatie (tot 3 dimensionale afbeeldingen met schaduwwerking toe) van het gesimuleerde proces te laten zien. In dat geval zie je op de beeldscherm de goederen bewegen, mensen een machine bedienen etc. Op die manier kun je optisch het proces in de gaten houden en als er zich knelpunten voordoen, dan zie je dat meestal meteen op je beeldscherm. Bovendien zijn die animaties zeer geschikt om managers er van te overtuigen dat de nieuwe produktielijn echt nodig is en kun je de beslissers en de toekomstige gebruikers laten zien hoe het er allemaal uit komt te zien.

Het grote voordeel van simulatie is uiteraard dat je, zonder één cent aan werkelijke bouwkosten uit te geven, snel een aantal alternatieven kunt onderzoeken. Bovendien kun je, mits je model goed is, van te voren bepalen welke capaciteit de installatie zal hebben. Ook de gevolgen van storingen kun je vrij gemakkelijk analyseren, zo zou je in onze winkel in kunnen programmeren dat één van de bedienden om 14:30 naar de tandarts gaat en om 15:30 terug is.

Afsluiting

Er is in de μP Kenner al een aantal keren iets verteld over toepassingen van computersystemen. In dit geval wist ik al geruime tijd dat simulatie als toepassing bestaat; ik had er zelf echter nog nooit direct mee te maken gehad. Nu dit wel gebeurde ging er bij mij een lichtje branden dat met dit onderwerp vast wel iets voor de μP Kenner te doen valt. Vandaar dat ik dit artikel en de bijbehorende programma's heb geschreven.

Ik hoop dat er meer mensen zijn die eens de moeite willen nemen iets te schrijven over een bepaalde toepassing. Ik denk dat er nog voorbeelden genoeg zijn waar computers al dan niet met speciale hard- en/of software ingezet worden. Denk bijvoorbeeld aan het besturen van een modelspoorbaan, besturen van draai- en/of freesbanken, temperatuurregeling in de woonkamer, beheer van een collectie platen en CD's etc. Kortom, wie neemt de handschoen op en vertelt ook eens iets over een interessante toepassing.

Gert van Opbroek

```
/* Programma wachtrij-simulatie
            Probleemstelling:

            - Bepaal door middel van simulatie hoe lang een klant gemiddeld in
            een winkel moet wachten.

            Copyright (c) 1993: G. van Opbroek

            Versie 1.0    d.d. 20 december 1993
 */

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

#define TRUE            1
#define FALSE           0

#define max_t           6 * 24 * 60   /* Een week                              */
#define max_klanten     50            /* Maximaal aantal klanten in de winkel  */
#define max_pers        10            /* Maximaal aantal bedienden             */
#define inactief        -1            /* Element wordt niet gebruikt           */
```

Fig. 3: simulatieprogramma voor een wachtrijprobleem

```
#define wachtend        0       /* Klant/bediende staat te wachten          */
#define actief          1       /* Klant/bediende is bezig          */

typedef struct
{       int status;
        long int tijdstip_wacht,        /* Tijdstip waarop status 0 werd    */
                 tijdstip_actief;       /* Tijdstip waarop status 1 werd    */
                 actief_tijd;           /* Bepaalde behandelingsduur voor klant */
} t_persoon;

long int n,                     /* Aantal klanten binnengekomen             */
        m,                      /* Aantal klanten vertrokken                */
        l,                      /* Aantal klanten geholpen                  */
        t,                      /* Verstreken tijd in minuten               */
        wachttijd_totaal,       /* Totale wachttijd in minuten              */
        max_wachttijd;          /* Maximale wachttijd voorgekomen           */

        int p1,                 /* Kans in %% dat er een klant komt         */
            k,                  /* Aantal klanten in de winkel              */
            pers;               /* Aantal personeelsleden                   */

        int winkel_status;      /* Winkel open (TRUE) of dicht (FALSE)      */

float   lambda;                 /* Stuur-grootheid voor de wachttijd        */

t_persoon klant[max_klanten],   /* De klanten in de winkel                  */
          bediende[max_pers];   /* De bedienden                             */

int winkel_open(int uur, int minuut)

{ /* Deze functie geeft aan of de winkel open is op tijdstip
        uur : minuut.

        Openingstijden: 8:30 - 12:00 uur en 12:30 - 18:00
        */
        int resultaat;

        if (uur > 8 && uur < 12 || uur > 13 && uur < 18)
        resultaat = TRUE;
        else if (uur == 8 && minuut > 30 || uur == 12 && minuut > 30)
        resultaat = TRUE;
        else
        resultaat = FALSE;
        return resultaat;
}

int aankomst(int kans)
{ /* Deze functie geeft aan of er in de afgelopen minuut een klant
        binnenkwam als de kans op een klant "kans" procent per minuut is.
        */

        int x;
        x = random(100);                /* Genereert een getal 0 <= x < 100 */
        return (x + 1 > kans);
}

int bepaal_actieftijd()
```

```
{ /* Deze functie bepaalt hoe lang een bediende met deze klant bezig
             zal zijn. De tijd volgt een zogenaamde negatief exponentiele
             verdeling.

             */
             /* P(t) = lambda * exp(-lambda * t)
             */

             float r,t;
             int          x,resultaat;

             x = random(10000);
             r = (x + 1.0)/10000;
             t = - log(r) / lambda;
             resultaat = (int) (t + 1);          /* Minimaal 1 minuut..... */
             return resultaat;
}

void global_initialize()
{ int i;
             randomize();                        /* initialiseer random numbers      */
             clrscr();                           /* maak het scherm schoon           */
             n = m = l = 0;
             wachttijd_totaal = max_wachttijd = 0;
             for (i = 0 ; i < max_klanten ; i+ +)
             klant[i].status = inactief;
             for (i = 0 ; i < max_pers ; i+ +)
             bediende[i].status = inactief;
}

void get_input()
{ int dt;
             char s[100];

             clrscr();
             printf("Simulatie-programma voor klanten in een winkel\n");
             printf("\n");
             printf("Auteur: G. van Opbroek BLP\n");
             printf("Versie: 1.0 datum 27 december 1993 \n\n");
             do
             { dt = -1;
             printf("\nGeef de gemiddelde tijd tussen twee klanten (min)      : ");
             if (scanf ("%D",&dt) = = 0)
             scanf("%s",s);
             if (dt < = 0 || dt > 99)
             printf("\n0 < dt < = %d, dt = %d\n",99,dt);
             } while (dt < = 0 || dt > 99);
             p1 = (int) (100.0 - (100.0/dt));

             do
             { pers = -1;
             printf("\nGeef het aantal winkelbedienden                        : ");
             if (scanf ("%D",&pers) = = 0)
             scanf("%s",s);
             if (pers < = 0 || pers > max_pers)
             printf("\n0 < pers < = %d, pers = %d\n",max_pers,pers);
             } while (pers < = 0 || dt > max_pers);
```

```
            do
            { dt = -1;
            printf("\nGeef de halfwaardetijd voor de behandelingsduur (sec): ");
            if (scanf ("%D",&dt) = = 0)
            scanf("%s",s);
            if (dt < = 0 || dt > 600)
            printf("\n0 < dt < = %d, dt = %d\n",600,dt);
            } while (dt < = 0 || dt > 600);
            lambda =    -60.0 * log(0.5) / dt;

            clrscr();
}

void genereer_tijd(int t, char *tijd)
{ int dag,uur,minuut;

            dag =       t / (24 * 60);
            uur =       (t / 60) % 24;
            minuut =    (t % 60);

            sprintf(tijd,"%1d %02d:%02d",dag,uur,minuut);
}

void print_results(int t)
{ char      tijd[10];
            float gem_wachttijd;

            if (l > 0)
            gem_wachttijd = (float) wachttijd_totaal / l;
            else
            gem_wachttijd = 0.0;

            gotoxy(1,1);
            genereer_tijd(t,tijd);
            cprintf("Resultaten op tijdstip : %s",tijd);
            cprintf("      De winkel is : ");
            if (winkel_status)
            cprintf("Open                \r\n\n");
            else
            cprintf("Dicht \r\n\n");

            cprintf("Totaal aantal klanten \r\n");
            cprintf("      Binnengekomen          : %5d\r\n",n);
            cprintf("      Vertrokken   : %5d\r\n",m);
            cprintf("      In de winkel  : %5d\r\n",n-m);
            cprintf("      In behandeling : %5d\r\n",l-m);
            cprintf("\r\n");
            cprintf("Gemiddelde wachttijd : %6.2f minuten",gem_wachttijd);
            cprintf("      Maximale wachttijd : %4d minuten",max_wachttijd);
}

main ()
{ int       uur,minuut;
            int         i,j;

            global_initialize();
```

```
        get_input();

        /* Zet alle aanwezige bedienden op "wachten"..... */

        for (i = 0; i < pers; i++)
        { bediende[i].status                    = wachtend;
        bediende[i].tijdstip_wacht = 0;
        }

        for (t = 1; t <= max_t; t++)
        { uur                    = (t / 60) % 24;
        minuut       = t         % 60;
        winkel_status = winkel_open(uur,minuut);

        if (winkel_status && (n - m) < max_klanten && aankomst(p1))
        { klant[n-l].status          = wachtend;
        klant[n-l].tijdstip_wacht = t;
        klant[n-l].actief_tijd       = bepaal_actieftijd();
        klant[n-l].tijdstip_actief= 0;
        n++;
        }
        for (i = 0; i < pers; i++)
        { if (bediende[i].status                  == actief &&
                    bediende[i].actief_tijd +
                    bediende[i].tijdstip_actief == t)
        { /* Deze klant is geholpen                                    */
        bediende[i].status                  = wachtend;
        bediende[i].tijdstip_wacht = t;
        m++;
        }
        if (bediende[i].status                  == wachtend &&
                    klant[0].status                       == wachtend)
        { /* De volgende klant kan geholpen worden      */
        bediende[i].status                  = actief;
        bediende[i].actief_tijd     = klant[0].actief_tijd;
        bediende[i].tijdstip_actief = t;
        wachttijd_totaal += (t - klant[0].tijdstip_wacht);
        max_wachttijd               = max(max_wachttijd,t - klant[0].tijdstip_wacht);

        for (j = 1; j < max_klanten ; j++)
        { klant[j-1].status                 = klant[j].status;
                    klant[j-1].actief_tijd      = klant[j].actief_tijd;
                    klant[j-1].tijdstip_actief  = klant[j].tijdstip_actief;
                    klant[j-1].tijdstip_wacht   = klant[j].tijdstip_wacht;
        }
        klant[max_klanten - 1].status       = inactief;
        l++;
        }
        }
        print_results(t);
        }
}
```

Voortgang DOS65

Het is zover. Van de bestelde vijf protoprinten zijn er inmiddels twee bestukt. De bestukkers hadden de euvele moed om de print in hun DOS65-wonder te steken. En wat bleek: de kaart werkte. De batterij-backup is nog niet getest, omdat de DS1210 battery controller moeilijk te leveren blijkt te zijn. Voor de rest blijkt alles naar verwachting te werken.

Daarom dit artikeltje.

Veranderingen

Er zijn ten opzichte van de reeds gepubliceerde schema's een paar veranderingen doorgevoerd. De simpelste verandering is dat de nummers van de verschillende IC's op de kop zijn gezet. Dat is het gevolg van het feit dat er een een keer een auto-annotate is gedaan in het schema-tekenpakket. De IC's worden dat in volgorde van plaatsing genummerd.

Ook al gemeld is de toevoeging van de DS1210 battery controller en een lithium batterijtje. De bedoeling hiervan zal iedereen duidelijk zijn: de RAMmetjes worden daar aanzienlijk minder vergeetachtig van.

De derde veranderinging is de pinout van de PAL op de print. We hebben gepoogd het printtekenpakket het leven zo min mogelijk zuur te maken door een paar pootjes te verplaatsen.

Aanwijzingen voor de bouw

Hieronder staat een boodschappenlijstje afgedrukt. Alles staat erop, ook de de IC-voetjes. U kent het verhaal wel: gebruik altijd voetjes in uw hobby-project want dat is gemakkelijker met foutzoeken en veranderingen aanbrengen. Gebruik ook fatsoenlijke voetjes,dus die wonders met gedraaide contacten.

Het opbouwen van de print is niet moeilijk. De print is dubbelzijdig uitgevoerd en het verschijnsel draadbrug wordt u dan ook bespaard. Breng eerst de IC-voetjes aan, gevolgd door de elco en de 100 nF condensatortjes. Deze laatste hebben een steek van 7.5 mm. De stuklijst gaat ervan uit, dat het voetje voor de 8kx8 SRAM wordt gemaakt door twee 14-pens voetjes te gebruiken. Mocht ook het smalle 24-pens voetje een probleem zijn, dan kan hiervoor een 8-pens en een 16-pens voetje worden benut. Let er verder even op dat de geheugen IC's pin 1 aan de andere kant hebben ten opzichte van de andere IC's. Monteer daarna de busconnector. Vergeet de twee boutjes niet. Laat de IC's en de batterij nog even weg.

Meet nu eerst tussen de pennen 32a/c en 1a/c van de busconnector of er misschien een sluiting in de voeding zit. Zo ja, deze eerst opsporen. Meet u niets of een zich opladende elco, dat is alles OK.

Wel of geen batterij

Nu komt de keuze of er wel of geen batterij op de geheugenkaart moet komen. Als u pseudo-statische RAMs gaat toepassen heeft de batterij geen zin: hij zou in korte tijd leeg zijn, en de RAMs krijgen toch geen refresh (activiteit op de adreslijnen) als de computer uit staat. Een batterijtje en een DS1210 hebben dus alleen zijn bij toepassing van echte CMOS statische RAMs.

Als het batterijtje niet wordt gebruikt vervalt dus ook de DS1210 (IC10) wen mag de 74HCT138 (IC8) ook wel een 74LS138 worden. Verbindt dan de pinnen 1 en 8 (VccI en VccO), en 5 en 6 (CEI en CEO) met twee draadjes door.

Verder opbouwen

Steek nu alle IC's, behalve IC1 (de databusbuffer) in hun voetjes. Sluit de batterij provisorisch aan en meet de stroomopname. Deze mag niet meer bedragen dan 50 uA: de battrij voetd de DS1210 (minder dan 100 nA), de 74HCT138 (4 uA) en 8 SRAMs (totaal maximaal 40 uA). Is het meer, verwijder dan een voor een de genoemde IC's tot de stroomvreter is gevonden.

Plaats de print, zonder IC1 in uw systeem. Laat de bestaande geheugenkaart op zijn plaats. Na inschakelen moet het sysateem normaal opstarten. Indien niet, eerst de oorzaak (sluiting van buslijnen) proberen te vinden.

Start het systeem wel normaal op, dan komt nu het grote moment: beide geheugenkaartenverwijderen, IC1 plaatsen de nieuwe RAMkaart plaatsen. Als het systeem nu nog steeds start, is de kaart zeer waarschijnlijk OK.

Virtual disk

Tijdens de evalutatie van de oude virtual diskkaart met DRAMs en de nieuwe die hier besproken wordt is er nog een verschilletje boven water gekomen: in de oude kaart wordt de mappinginformatie door 74LS189's geinverteerd opgeslagen, maar in de nieuwe normaal. Dit levert geen problemen op, als de kaart helemaal volbestukt is met 8 SRAMs.

Is de kaart echter gedeeltelijk leeg, dan is er een kunstje. Vervang IC2 dan door een 74LS640, dit is

een inverterende 74LS245. De kaart is dan echt compatibel met de oude VDISK-kaart.

Moeilijke onderdelen

Een aantal onderdelen is wat moeilijker verkrijgbaar. Zo blijkt bijna niemand de DS1210 te leveren, en een geprogrammeerde PAL of GAL is ook niet even eenvoudig. Verder blijken ook 74ALS541's niet zo een, twee, drie te verkrijgen te zijn.

Voor de 74ALS541 kan eventueel ook een gewone 74(A)LS245 gebruikt worden, mits pin 1 niet in de voet wordt gestoken, maar over de bovenzijde van het IC wordt verbonden met pin 20 (Vcc).

Voor de overige onderdelen wordt op dit moment onderhandeld om deze via de club te gaan leveren. De volgende prijzen lijken hierbij realiseerbaar:

DS1210	*f*	10,00
GAL22V10 (geprogr.)	*f*	15,00
8kx8 SRAM, 25 ns	*f*	8,00
128kx8 CMOS SRAM	*f*	30,00 (dagprijs!)

In het volgende nummer zullen de definitieve prijzen, en de prijs voor de print (dubbelzijdig, doorgemetalliseerd, met soldeermasker en opdruk) bekend gemaakt worden, en hoe u dit allemaal kunt bestellen.

Tot de volgende keer.

Nico de Vries

Item	Aantal	Referentie	Omschrijving
1	1	C1	100uF, 10V
2	17	C2 t/m C18	100nF
3	2	IC1,IC2	74LS245
4	1	IC3	GAL20V8 of PAL22V10
5	1	IC4	SRAM, 8kx8-25ns (0.3 inch behuizing)
6	1	IC5	74ALS573
7	2	IC6,IC7	74ALS541
8	1	IC8	74LS138
9	1	IC9	74LS30
10	1	IC10	Dallas DS1210
11	8	IC11 t/m IC18	1 Mbit (128kx8) SRAM of Pseudo SRAM
12	1	J1	Connector DIN 41612, a/c, male haaks
13	1	BT1	Batterij, lithium, 3.6V
14	1	-	Print
15	1	-	IC voetje, 8 pins
16	3	-	IC voetje, 14 pins
17	1	-	IC voetje, 16 pins
18	5	-	IC voetje, 20 pins
19	1	-	IC voetje, 24 pins, 0.3 inch
20	8	-	IC voetje, 32 pins

Mogelijkheden voor SRAM, 8kx8:
UMC: UM6164AK-25
Cypress: CY7C185-25
Toshiba: TC5588P-25
Fujitsu: MB81C78A-25, of equivalenten.

Mogelijkheden voor 1Mbit SRAM of Pseudo SRAM:
Toshiba: TC551001P (CMOS SRAM) of TC518129P (Pseudo Static)
Hitachi: HM628128 (CMOS SRAM) of HM658128 (Pseudo Static)
NEC: μPD431000 (CMOS SRAM), of equivalenten.
Snelheid 200ns (bij 1 MHz) of 150 ns (bij 2 MHz) of sneller.
LET OP: bij toepassing van de DS1210 en de batterij alleen CMOS SRAM gebruiken!

Onderdelenlijst

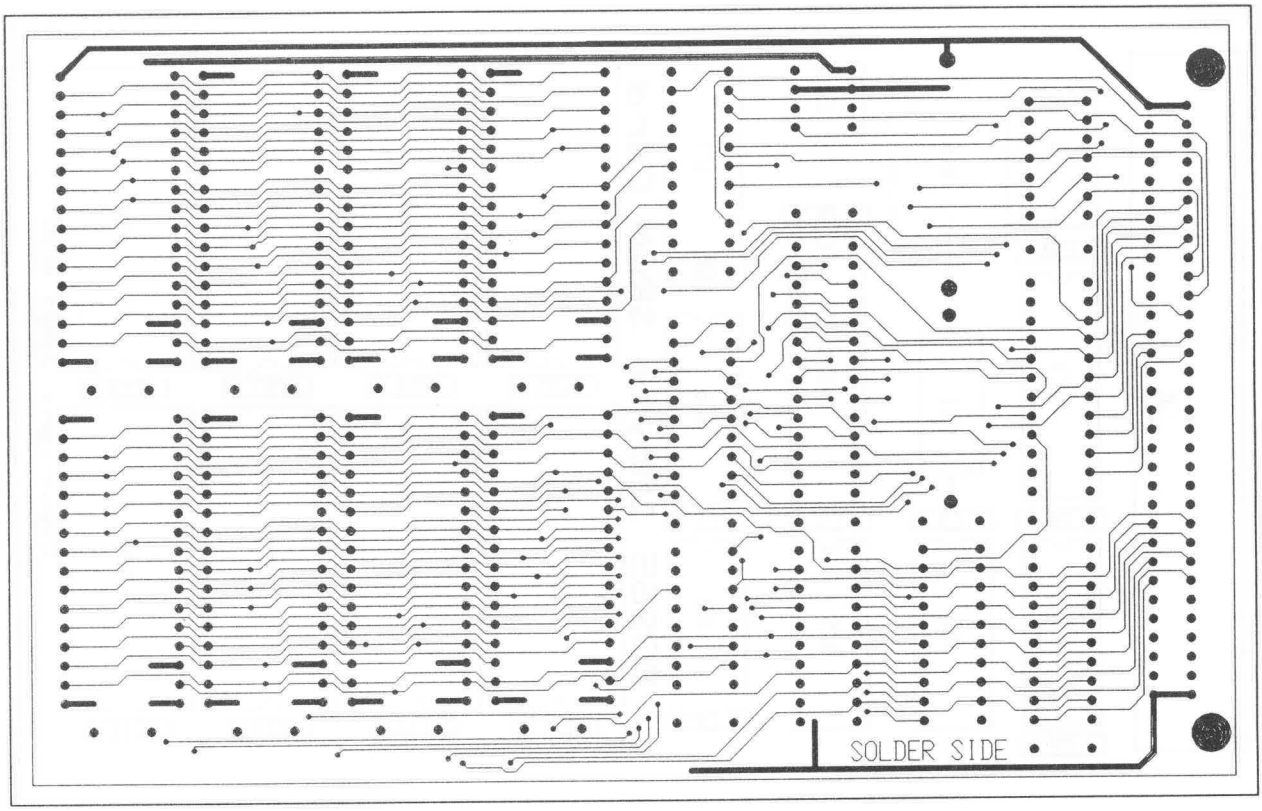


Fig. 1: print, sporen aan soldeerzijde

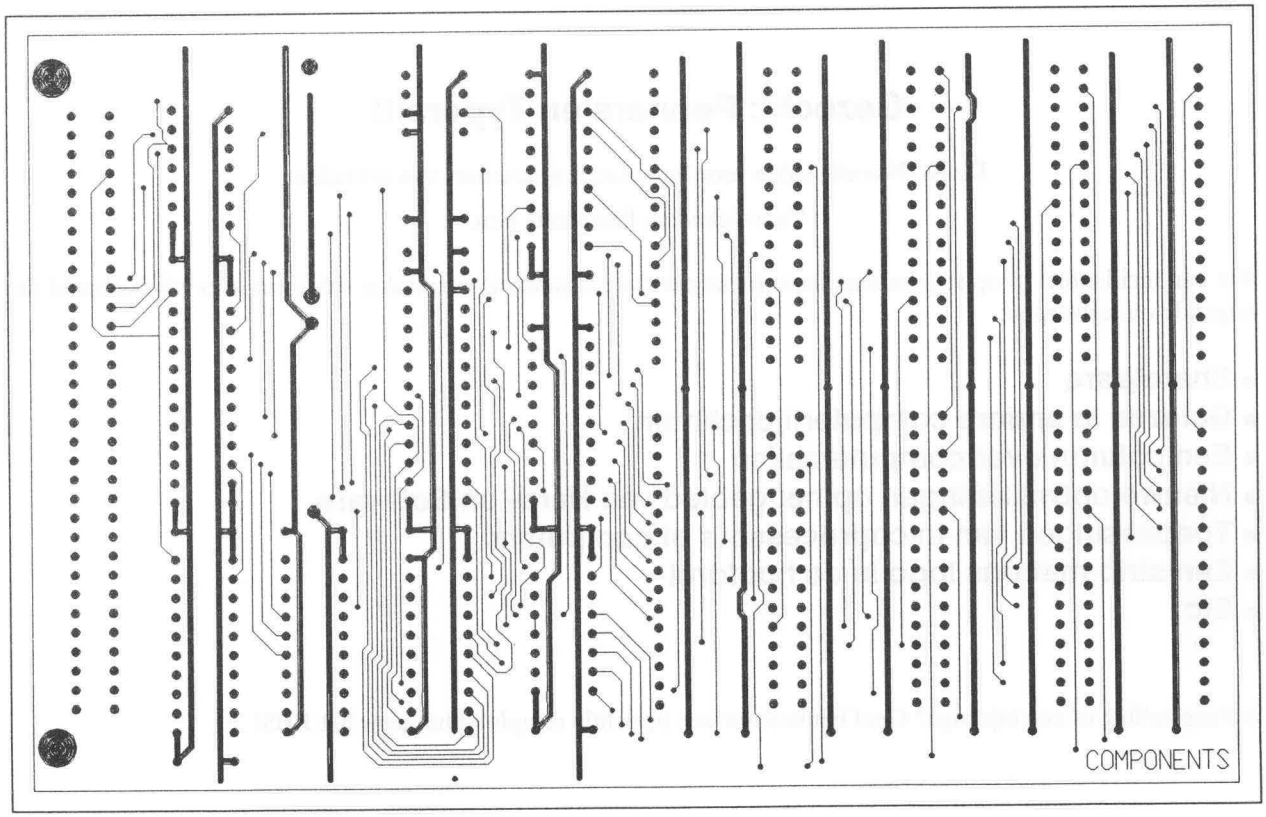


Fig 2: print, sporen aan componentenzijde

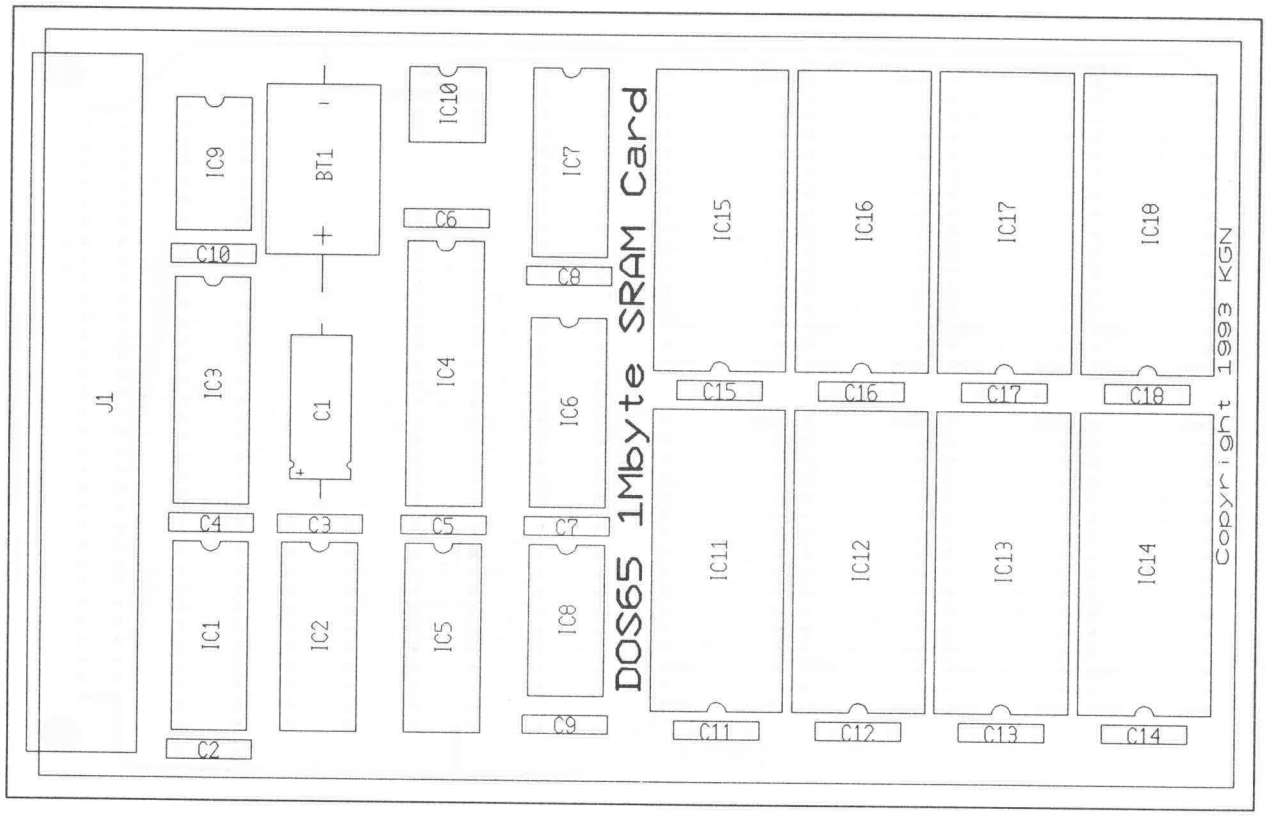


Fig 3: print, componentenopstelling

Gezocht: Penners en Typers!!!

De KGN zoekt leden voor de redactie en auteurs van artikelen.

Vaardigheden: Een vlotte pen.

Wie kan incidenteel of op regelmatige basis (eens per twee maanden) een stukje schrijven over bijvoorbeeld de volgende onderwerpen:

- **ShareWare**
- **Gelezen in andere computertijdschiften**
- **Een column over computerzaken**
- **Nieuwe ontwikkelingen op het gebied van Hard- en Software**
- **Toepassingen van micoprocessors of computers**
- **Een strip met ons logo in de hoofdrol**
- **Etc.**

Bellangstelling of een bijdrage? Geef het bestuur een berichtje of upload het naar het BBS!

```
GAL20V8                                                    GAL DESIGN SPECIFICATION
VDISK3 5th VERSION                                         N. DE VRIES  07-02-1994
8kx8 SRAM CONTROLLER PAL FOR DOS65 VDISK CARD WITH SRAMS, 1M VERSION
NDV FIRMWARE
RW  PHI2 A0    A1      A2    A3  A4  A12 A13 A14  A15  GND
NC1 NC2  G245  WE2064 G573  Q3  Q2  Q1  Q0  G138 LS30 VCC

/G138  = PHI2*/A15                          ;access
       + PHI2*   /A14                       ;to
       + PHI2*      /A13                    ;RAM
       + PHI2* A15* A14*/A13                ;array

/G245  = /RW* A15* A14* A13* A12*/LS30*/A4                ;write to $FFE0-$FFEF
       + /RW* A15* A14* A13* A12*/LS30* A4*/A3*/A2*/A1*/A0        ;write to $FFF0
       +          /A13                                  ;access
       +       /A14                                     ;to
       +    /A15                                        ;RAM
       +     A15* A14*/A13                              ;array

/WE2064 = PHI2*/RW* A15* A14* A13* A12*/LS30*/A4          ;write to $FFE0-$FFEF

G573   = PHI2*/RW* A15* A14* A13* A12*/LS30* A4*/A3*/A2*/A1*/A0     ;write to $FFF0

/Q0    = /A0* A15* A14* A13* A12*/LS30*/A4    ;write to $FFEX: A0 is 8kx8 SRAM addressline
       +             /A12                    ;else let A12 through

/Q1    = /A1* A15* A14* A13* A12*/LS30*/A4    ;write to $FFEX: A1 is 8kx8 SRAM addressline
       +          /A13                       ;else let A13 through

/Q2    = /A2* A15* A14* A13* A12*/LS30*/A4    ;write to $FFEX: A2 is 8kx8 SRAM addressline
       +       /A14                          ;else let A14 through

/Q3    = /A3* A15* A14* A13* A12*/LS30*/A4    ;write to $FFEX: A3 is 8kx8 SRAM addressline
       +   /A15                              ;else let A15 through


DESCRIPTION

G245 is the enable for the databusbuffer

G573 is the enable for the task number register (must be active high)

G138 is the enable for the RAM array address decoder

WE2064 is the write enable input for the 8kx8 SRAM and the enable for the
       write data buffer to the 8kx8 SRAM

Q0..Q3 are the lower addresslines for the 8kx8 SRAM
```

Vergelijkingen voor de GAL/PAL

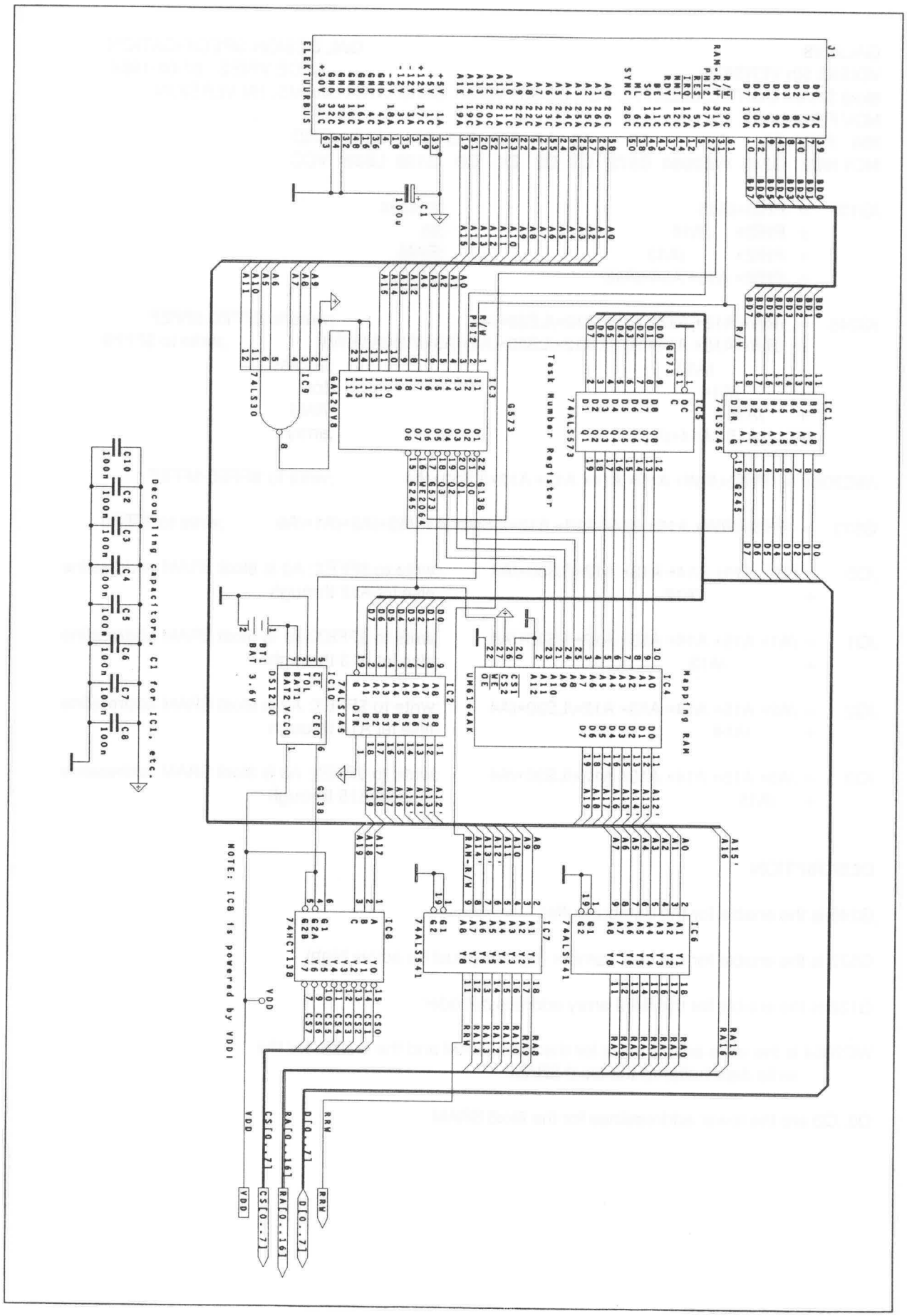


Fig. 5: Schema van het stuurgedeelte

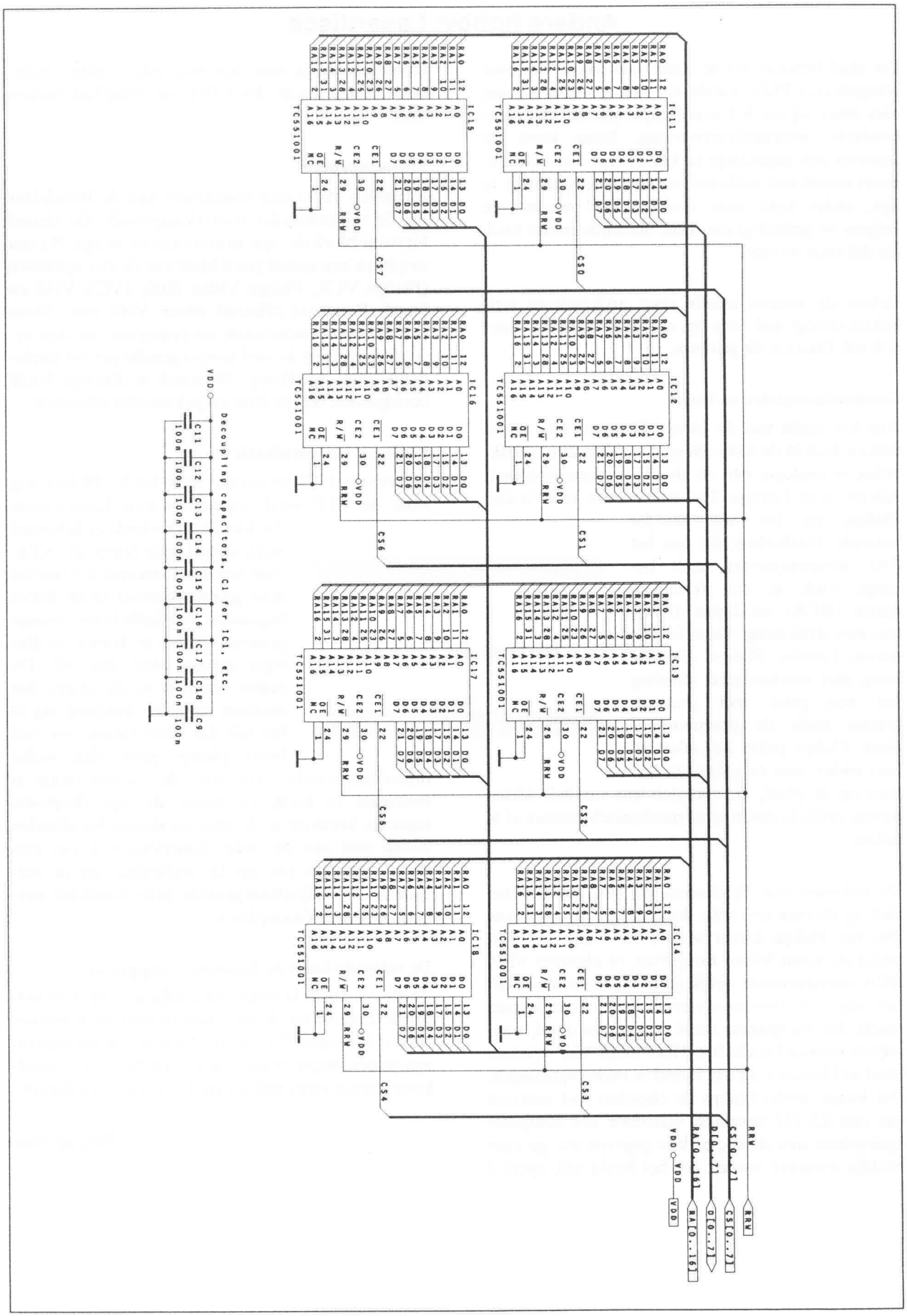


Fig. 6: Schema van het geheugengedeelte

Andere hobby: Laserdiscs

Dit blad behoort vol te staan met zaken die over computers, CPU's, hardware, zachte waren en wat dies meer zij op het wonderlijke gebied van elektronische informatieverwerking. Soms komt de layouter een paginaatje te kort. Want het blad behoort steeds een veelvoud van vier bladzijden dik te zijn, ander kijkt onze drukker heel vreemd en krijgen we geheel gratis witte bladzijde in het blad. En dat staat zo raar.

Echter de stroom ideeën voor artikelen en artikeltjes droogt wel eens op, en die ene pagina moet ook vol. Daarom dit gezwam.

Grammofoonplaten en beeld

Aan het einde van de jaren '60 waren een aantal firma's druk in de weer om een beeldplaat te ontwikkelen, in analogie met de aloude grammofoonplaat. Allereerst in Europa. Dat waren onze vaderlandse Philips, en het wat minder bekende Telefunken, dat ons het PAL kleurensysteem heeft bezorgd. Ook in de Verenigde staten (RCA) en Japan (JVC) was men druk bezig. Deze firma's waren, behalve Philips allemaal bezig met mechanische aftasting van een plaat met groeven, precies zoals de grammofoonplaat. Philips pakte het allemaal heel anders aan en gebruikte een laser op de plaat, die voorzien was van hele kleine putjes, optisch, dus zonder mechanisch contact af te tasten.

Daarom dit gezwam.

De systemen van Telefunken en JVC haalden het niet: zij stierven een stille dood in het laboratorium. Dat van Philips kwam wel op de markt in 1972, onder de naam Video Long Play, of afgekort VLP. RCA introduceerde gelijktijdig het mechanisch aftastende Selectavision-systeem op de Amerikaanse markt. RCA's systeem stierf een vroege dood, want vrijwel niemand kocht het. Het Philips VLP-systeem bleef wel bestaan, zij het vooral in onderwijskringen. Dat kwam omdat Philips de afspelers had voorzien van een RS-232 interface, waardoor een computer opdrachten aan de speler kon gegeven als: ga naar beeldje nummer zoveel, zet het beeld stil, speel 2 seconden af, ga naar het volgende beeldje, enzovoort. En u dacht dat CD-I iets compleet nieuws was?

Videorecorders

Inmiddels rukte een concurrent van de beeldplaat op: de videorecorder voor thuisgebruik. De sleutel hiervoor bleek de tape in een cassette te zijn. Na een strijd van een aantal jaren bleef van de vier systemen (Philips VCR, Philips Video 2000, JVC's VHS en Sony's Betamax) effectief alleen VHS over. Door een enorme concurrentie en prijserosie werden recorders en tape in snel tempo goedkoper en uiteindelijk zelfs goedkoop. Niemand in Europa kocht beeldplaten: veel te duur en je kunt niet opnemen.

Poging 2: herintroductie

Omstreeks 1978 probeerde men het bij Philips nog eens: de VLP werd omgedoopt naar LaserVision. Technisch veranderde er helemaal niets. Zowel in de States als in Europe werd het systeem ten tweede male geïntroduceerd. In de States begonnen filmliefhebbers mondjesmaat spelers te kopen, in Europa echter bleef het stil. De reden waarom in de States het medium enigszins aansloeg lag in het feit dat LaserVision een veel beter plaatje geeft dan welke thuisvideorecorder dan ook: de beelddefinitie is tweemaal zo hoog, en zaken als tape dropouts, taperuis, kreukels in de tape en slijtage bij afspelen waren niet aan de orde. LaserVision is dus een ideaal medium om op te archiveren en te verzamelen. Mondjesmaat groeide in de States het aanbod aan LaserVisionplaten.

De volgende keer: de Japanners stappen in

Zo de bladzijde is weer een eindje gevuld. Een volgende keer vertel ik weer verder over mijn nieuwe hobby: Laserdisc. Een waarschuwing is misschien op zijn plaats: begin er nooit aan want het is niet goedkoop, en het vreet tijd om als dat moois te bekijken!

Nico de Vries

Vierde les C

Ik heb enkele vragen gekregen over de cursus en ik wil beginnen met daarop antwoord te geven.

- Iemand vraagt mij om de oplossingen van de oefeningen. Ik heb dat bewust weggelaten omdat:
    - ik uw oplossingen niet kan beoordelen. Normaal geef ik deze cursus als een klassikale cursus en kan ik de oplossingen van cursisten beoordelen en bespreken.
    - alle oefeningen zijn overgenomen uit de literatuur zoals vermeld aan het eind van les 1. U vindt hierin nog tal van andere voorbeelden.
    - een schriftelijke cursus zoals deze, waarbij grote intervallen zijn tussen de verschillende afleveringen, velen aan het denken en puzzelen zet. Al gauw zul je proberen een 'eigen' vraagstuk op te lossen door daarvoor een programma(tje) te schrijven in 'C'. Daarbij moet je al snel naar de boeken grijpen - in feite studeer je zelf sneller dan de cursus aangeeft.

- Een andere vraag betreft het 'SWITCH' statement. Ik heb daarbij aangegeven (blz. 37 van nr. 83) dat de 'DEFAULT' verplicht is. Helemaal waar is dit niet. Je mag de 'DEFAULT' weglaten mits je 100% zeker weet dat altijd een van de 'CASEs' voldoet en gevolgd wordt door een 'BREAK'. Gemakshalve leek het mij daarom beter de 'DEFAULT' verplicht te stellen....

- Dan een vraag over 'ELSE IF'. Helaas blijkt dat enkele misverstanden te hebben opgeroepen (ligt aan mij; ik had duidelijker moeten zijn). Daarom probeer ik het hier nogmaals te verklaren. Het is mogelijk - en zal het ook veel voorkomen - dat een 'IF' - 'ELSE' construktie op zich zelf, dus binnen de structure van deze 'IF - ELSE', weer opnieuw een 'IF - ELSE' constructie bevat. In dat geval wordt vaak de 'IF' van de tweede constructie (of structure) direkt achter de 'ELSE' van de eerste constructie geplaatst. Dat wordt in de literatuur de 'ELSE - IF' genoemd. Ik heb dit in de cursus opgenomen omdat dit begrip vaak gehanteerd wordt. In feite betekent het niets anders dan dat je begrijpt dat hiermee een nieuwe 'IF - ELSE' structure bedoeld wordt binnen een andere 'IF - ELSE'. Dit verschijnsel wordt ook wel 'NESTING' genoemd. De 'IF' geplaatst direkt achter de vorige 'ELSE' is eigenlijk een voorbeeld van **NIET - GESTRUCTUREERD** programmeren.

- De laatste vraag sluit aan op de vorige; zijn er richtlijnen te geven voor een goede layout van een programma? Ik wil met dit onderwerp beginnen.

'Structured' programmeren

Structured programmeren is op de eerste plaats een dicipline waarbij je een programma zodanig schrijft, dat een duidelijk en overzichtelijk geheel onstaat. Behalve een dicipline kan het ook een kwestie zijn van smaak. Als je trouwens (maanden) later een niet goed gestruktureerd programma wilt wijzigen begrijp je je eigen coding niet meer.... Dus is het schrijven van een ge-structureerd programma niet alleen mooi, het dient tegelijk ook het gemak.

Een voorbeeld:

```
main()
{
int tel = 0; int totaal = 0; while ( tel < 10 )
{
totaal + = tel; printf("tel = %d, totaal = %d\n", tel + +, totaal);    }
}
```

Dit voorbeeld ben je al tegengekomen bij de behandeling van WHILE. Nu echter heb ik veel statements gewoon achter elkaar geschreven. Als je dit vergelijkt met het oorspronkelijke voorbeeld (blz. 33 nr. 83) zal onmiddellijk opvallen dat de manier hierboven niet direkt duidelijk overkomt. Er is bij het schrijven ook een risico; u vergeet gemakkelijk een ; waarmee elk statement wordt afgesloten.

Ik had het nog bonter kunnen maken door ook de accolades achter elkaar en voor of achter de andere statements te plaatsen. Het hele voorbeeld had ook in een enkele regel geschreven kunnen zijn.

Enkele richtlijnen voor gestructureerd programmeren:

- Gebruik voor elk afzonderlijk statement een aparte regel. Plaats het 'MAIN()' statement op de eerste regel met daaronder de accolade voor de structure van de main. Mijn gewoonte is om dan daaronder, in dezelfde positie meteen de accolade te plaatsen waarmee de main structure wordt afgesloten. Daarmee voorkom ik dat ik het vergeet.

- Bouw nu het programma op tussen beide accolades. Bij vergelijkingen en loops laat je de statements binnen de vergelijking of loop iets inspringen. Ook hierbij heb ik de gewoonte de accolades indien van toepassing alvast te plaatsen en daarna de statements tussen te voegen.

- Bij nested loops opnieuw iets inspringen.

- Vergeet NIET kommentaar te schrijven in een programma. Het is een goede gewoonte elke 'gebeurtenis' in een programma in enkele regels toe te lichten. Meestal wordt dit als kop boven het betreffende programmadeel geplaatst; soms kan het ook handig zijn het commentaar achter de statements te zetten. Je voorkeur is een kwestie van smaak.
  Zie blz. 32 nr. 81 voor een voorbeeld.

  Het plaatsen van b.v. een loop binnen een loop wordt NESTING genoemd. Als je gestructureerd een programma schrijft waarin nesting wordt toegepast, valt een geneste structure doordat het inspringt en aan het einde weer terugspringt, direkt op.

  Het lijkt wat kinderachtig maar ik geef je de verzekering dat je jezelf later wel om de oren kunt slaan als je geen kommentaar regels hebt geschreven (en dan denk je aan deze cursus!).

Zo, het vragenuurtje hebben wij nu gehad.

Ik behandel in deze aflevering wat theorie als inleiding voor 'pointers'. Weliswaar valt de behandeling van pointers buiten het bestek van de cursus maar wat basisinformatie kan geen kwaad.

De laatste twee delen zullen wij behandelen het lezen - en schrijven van en naar files en interne subroutines. Daarna ben je in staat komplete doch eenvoudige programma's te schrijven.

Arrays en strings

Om je meer begrip bij te brengen over de behandeling van strings, zal ik proberen uit te leggen dat een string met datatype 'CHAR' in feite een array is.

Een array - ook wel een matrix genoemd - is voor te stellen als een vierkant, verdeeld in hokjes. Een dergelijk vierkant kan voorzien worden van aanduidingen langs de X en langs de Y as waarmee de mogelijkheid ontstaat, elk hokje aan te duiden.

Een dergelijke array wordt een array met 2 dimensies genoemd.

Als we dit projecteren naar 'C' zal je begrijpen dat een array een gebied(je) is in het geheugen en dat elk vakje een of meer bytes zijn in dat gebied, al dan niet gevuld met data. Daarnaast moet er een mogelijkheid zijn om aan de compiler duidelijk te maken dat wij een array wensen.

Een array met 3 dimensies is vergelijkbaar met een kubus met een X, Y en Z as. Arrays met nog meer dimensies zijn ook mogelijk maar kunnen wij ons moeilijker voorstellen....

Voorbeeld (array met 1 dimensie)

Wij bekijken eerst enkele voorbeelden van array's met 1 dimensie.

```
main()
{
int temp[7];
```

Reserveert 7 aaneengesloten gebieden in storage, met vaste afmetingen (wegens 'int'). Deze vaste afmetingen zijn overigens afhankelijk van het type processor. Zo reserveert een 80286 2 bytes voor elk gebiedje. Het hoogste bit wordt gebruikt als sign bit zodat er maximaal 32 kByte beschikbaar is.

Nu een kompleet voorbeeld:

```
/* Bereken de gemiddelde temperatuur over een week */
main()
{
 int temp[7];              /* array declaratie */    int dag, som;

 for (dag = 0; dag < 7; dag + +) /* plaats temp's in array */      {
   printf("Voer temperatuur in voor dag %d: ", dag);      scanf("%d", &temp[dag]);
   }

 som = 0;
 for (dag = 0; dag < 7; dag + +)
  som + = temp[dag];
 printf("Gemiddelde temperatuur is %d. ", som/7);
}
```

In scanf() MOET de addressoperator worden gebruikt. De addressoperator is het & teken voor de naam van de variabele:

```
scanf("%d", &temp[dag]);
```

&temp is de naam van het adres in storage. Daarin wordt het eerste adres gevonden waarin de waarde staat. De 'dag' als element van &temp (&temp[dag]) bepaalt de zoveelste byte, gerekend vanaf het begin adres.

Tijdens het compileren worden namen van variabelen vervangen door werkelijke adressen in het geheugen en verder zodanig uitgewerkt dat een sprongopdracht wordt uitgevoerd naar het begin van dat addres.

Dus:

&temp = 1491 (verwijst naar beginpositie van array)
&lbrk.dag&rbrk. = 1 (en wordt telkens automatisch verhoogd)
&temp&lbrk.dag&rbrk. = 22 (de inhoud van de plaats in het array)

Doordat wij & gebruiken wordt het beginadres - en dus NIET de waarde van dit array aan de functie scanf() overgedragen. Zo wil de functie scanf() het nu eenmaal hebben...
Zou je de addressoperator weglaten, dan wordt het beginadres van de array getoond.

Als meer wordt ingelezen dan de array groot is, wordt dit NIET gekontroleerd. Je raakt dan bytes kwijt!

Arrays van andere data typen

Voorbeeld:

```
main()
{
float temp[7];
char naam[81];
```

'float' werkt hetzelfde als 'int' maar 'char' is afwijkend...

maar eerst gaan wij nog even verder met het assignen van 'int' of 'float' variabelen. Daarbij zullen wij ook arrays met 2 dimensies bekijken. Eerst enkele voorbeelden met 1 dimensie.

```
/* Bereken de gemiddelde temperatuur over een maand */
#define LIM 40

main()
{
 float temp[LIM];              /* array declaratie */
 float som = 0;
 int num, dag = 0;

 do                        /* plaats temp's in array */
   {
   printf("Voer temperatuur in voor dag %d: ", dag);
    scanf("%f", &temp[dag]);
   }
 while (temp[dag++] > 0);        /* stop invoer indien 0 */


 num = dag - 1;              /* aantal ingevoerde temp's */
 for (dag = 0; dag < num; dag ++)  /* bereken totaal       */
   som += temp[dag];
 printf("Gemiddelde temperatuur is %.1f", som/num);
}
```

- Het #define statement bepaalt het aantal elementen in variabele 'LIM'
- Alle statements, geplaatst boven (of voor) de main() en meestal herkenbaar aan het # teken, zijn opdrachten voor de compiler en/of linker. In dit voorbeeld 'weet' de compiler dat met het woord 'LIM' overal waar dit in het programma wordt gebruikt, de waarde: '40' moet worden ingevuld.
- Het aantal elementen waaruit de array moet bestaan is in 'float' declaratie vervangen door de naam van de variabele: 'LIM'

Nog een voorbeeld. In dit voorbeeld wordt de inhoud van variabele 'LIM' gebruikt in een kontrole op het aantal ingetikte elementen:

```
/* Bereken de gemiddelde temperatuur over een maand */
/* maar nu met een kontrole op de grootte van de array */

#define LIM 10

main()
{
 float temp[LIM];              /* array declaratie */
  float som = 0;
 int num, dag = 0;

 do                          /* plaats temp's in array */
   {
   if ( dag > = LIM )
    {
    printf("Buffer vol.\n");
    dag+ +;
    break;
    }

   printf("Voer temperatuur in voor dag %d: ", dag);
    scanf("%f", &temp[dag]);
   }
 while (temp[dag+ +] > 0);        /* stop invoer indien 0  */


 num = dag - 1;               /* aantal ingevoerde temp's */
  for (dag = 0; dag < num; dag + +)  /* bereken totaal       */
   som + = temp[dag];
 printf("Gemiddelde temperatuur is %.1f", som/num);
}
```

Met deze kontrole loopt u niet het gevaar dat er gegevens zoek raken ten gevolge van het vollopen van de array (zie waarschuwing hiervoor).

In het volgende voorbeeld wordt een array gedeclareerd en daarna assigned:

```
/* Reken een bedrag om in munten              */

#define LIM 5
int tabel[LIM] = { 250, 100, 25, 10, 5 };

main()
{
 int dex, bedrag, aantal;
 printf("Voer bedrag in uitgedrukt in centen (b.v: 636): ");
  scanf("%d", &bedrag);
 for (dex = 0; dex < LIM; dex + +)
   {
   aantal = bedrag / tabel[dex];
   printf("Waarde van munt=%2d, ", tabel[dex] );
   printf("aantal munten=%2d\n", aantal );
   bedrag = bedrag % tabel[dex];
   }
}
```

De array (tabel) wordt gevuld met vijf waarden. Deze waarden moeten worden opgesloten tussen { en }. De waarden worden onderling gescheiden met een komma.

Het toekennen van waarden aan een array kan ook plaatsvinden binnen de body van het programma. Dat kan zodanig gebeuren dat deze array (met zijn inhoud) uitsluitend bekend is aan het programma:

```
/* Reken een bedrag om in munten          */

#define LIM 5

main()
{
 static int tabel[] = { 250, 100, 25, 10, 5 };
 int dex, bedrag, aantal;
 printf("Voer bedrag in uitgedrukt in centen (b.v: 636): ");   scanf("%d", &bedrag);
 for (dex = 0; dex < LIM; dex + +)
   {
   aantal = bedrag / tabel[dex];
   printf("Waarde van munt = %2d, ", tabel[dex] );
   printf("aantal munten = %2d\n", aantal );
   bedrag = bedrag % tabel[dex];
   }
}
```

- Aantal elementen van de array volgt automatisch uit het aantal waarden tussen { en } bij het static statement
- 'static' geeft aan dat de array met inhoud uitsluitend bekend is aan dit programma. Dit worden lokale variabelen genoemd.
- #define geeft het aantal elementen aan waaruit de array is opgebouwd. In dit voorbeeld wordt de variabele 'LIM' niet gebruikt in de declaratie en assingment van 'tabel' maar wel gebruikt in de 'for' loop. Het feit dat achter de declaratie van variabele 'tabel' tussen accolades 5 waarden zijn gegeven, zorgt automatisch voor een array van 5 gebiedjes.

Het static statement mag ook op de volgende manieren worden geschreven:

```
static int tabel[LIM] = { 250, 100, 25, 10, 5 };
of:
static int tabel[5] = { 250, 100, 25, 10, 5 };
```

En dan nu een voorbeeld van een array met twee dimensies:

```
/* Maak een werkstaat per medewerker          */

#define RIJEN 10
#define KOLOMMEN 2

main()
{
 float medewerker [RIJEN] [KOLOMMEN];
 int index=0, uitdex;

 printf("Voer empl.nr van 3 cyfers in,\n");
 printf("en dan het aantal gewerkte uren\n");
 printf("Typ 0 0 om programma te verlaten.\n");
 do
  {
  printf("Nummer en aantal werkuren: ");
  scanf(" %f %f", &medewerker[index] [0], &medewerker[index] [1] );
   }
 while ( medewerker[index++] [0] != 0 );

 for (uitdex=0; uitdex < index-1; uitdex++)
  {
  printf("Medewerker %3.0f ", medewerker[uitdex] [0] );
   printf("heeft gewerkt %7.2f.\n", medewerker[uitdex] [1] );
   }
}
```

Het aantal 'kolommen' is ingesteld op '2'; het aantal 'rijen' op '10'. Dat kan worden voorgesteld als:

kolom 0	kolom 1	
101	8.3	
102	8	
104	7.5	10 rijen
119	12	
...	...	

De variabele 'index' bevat de eerste keer het begin adres van het array; elke volgende keer wordt dit verhoogd om daarmee de plaats van de volgende rij te kunnen bereiken. Daarom is het mogelijk de variabele 'index' te gebruiken in een loop om daarmee het aantal cycles van de loop in te stellen.

Door middel van de scanf() functie worden er telkens twee waarden ingelezen en geplaatst in de beide kolommen 0 en 1. De rij wordt bepaald door de teller 'index'. Deze teller wordt in de while loop verhoogd. Omdat kolom 0 en kolom 1 tegelijk kunnen worden ingelezen (in de scanf()), zijn de nummers vast benoemd. Het zal je opvallen dat de volgorde begint bij 0.

Een string is een array!

In tegenstelling tot een eenvoudige declaratie van 'float' of 'int' wordt bij declaratie van een variabele met het datatype 'char' altijd een array gebruikt. Dat is de reden voor het weglaten van de address operator bij de scanf() functie. Er wordt direkt naar het begin van de array gesprongen.

Bij 'int' en 'float' hebben wij immers twee mogelijkheden. Of, het is een normale declaratie voor het reserveren van 1 cyfer of getal of, wij wensen een array. Bij declaratie van een variabele 'char' is altijd sprake van een array. Helaas is jammer dat de bedenkers van deze taal dit niet uniform hebben gemaakt. Nu moet je dit zelf goed in de gaten houden. .sp

Bekijken wij nu een voorbeeld met CHAR:

```
/* Voorbeeld van het lezen van een string */
main()
{
 char naam[15];

 printf("Type uw naam in: ");
 scanf("%s", naam);
 printf("Groetjes, %s.", naam);
}
```

Als het programma wordt uitgevoerd en een string wordt ingetikt dat bestaat uit meerdere woorden, zal blijken dat alleen het eerste woord is opgeslagen. Hoe dat komt zal ik grafisch verklaren:

	15 bytes gereserveerd onder "naam"	vult array totdat een blank gevonden	
naam:	0	H	
	1	A	
	2	N	
	3	S	
	enz.	\0	rest is leeg

Het \0 teken wordt automatisch in de array geplaatst zodra in de input een spatie wordt gevonden. Dit teken wordt het 'NULL' karakter genoemd. Alle bytes na het NULL teken zijn - en blijven leeg.

Om nu te demonstreren op welk adres de array begint kun je gebruik maken van het nu komende voorbeeld:

```
/* Voorbeeld van het lezen van een string */
main()
{
 char naam[15];

 printf("Type uw naam in: ");
 scanf("%s", naam);
 printf("Groetjes, %s. Adres: %u", naam , &naam);
}
```

Dit voorbeeld wijkt af van het vorige voorbeeld; de printf() functie is uitgebreid met een &naam. Daardoor wordt het beginadres van de array zichtbaar.

Er bestaan speciale functies voor string bewerkingen zonder het vervelende weglaten van woorden zoals bij 'scanf()'. In het nu volgende voorbeeld is gebruik gemaakt van 'puts()' en 'gets()', welke in het vorige deel zijn behandeld.

```
/* Voorbeeld van het lezen van een string */
/* maar nu met gebruik van GETS en PUTS   */

main()
{
 char naam[15];

 puts("Type uw naam in: ");
 gets(naam);
 puts("Groetjes, ");
 puts(naam);
}
```

Als nu een naam wordt ingetikt bestaande uit meerdere woorden / losse letters zal het geheel weer getoond worden door het programma. Daarvoor zorgt 'gets()'.
De functie 'puts()' heeft als onhebbelijkheid dat het geen literal string met een variabele tegelijk kan verwerken. Daarom zijn twee puts() functies gebruikt; een maal voor het tonen van 'Groetjes' en een maal voor het tonen van de inhoud van array 'naam'.
Je kunt beide puts() functies vervangen door een printf() functie, dan ben je het probleem kwijt.

Tot slot een voorbeeld dat een array, ontstaan door de declaratie 'char' byte voor byte op het scherm toont. Het laat elk karakter zien, het adres in het geheugen en de plaats van het karakter in de ASCII tabel.

```
/* Programma bekijkt string in het geheugen     */

main()
{
 char naam[81];
 int dex;

 puts("Type uw naam: ");
 gets(naam);
 for (dex = 0; dex < strlen(naam) + 4; dex++ )
  printf("Adres=%5u  char='%c' = %3d \n",
      &naam[dex], naam[dex], naam[dex] );
}
```

De variabele 'naam' is gedeclareerd met een lengte van 81 posities. Daarmee is het mogelijk een regel in te tikken van 80 (!) lang. De laatste positie wordt automatisch gevuld met een linefeed - carriage return character. Dat is nodig om dit record gescheiden te houden van het volgende record. Hoe dat in zijn werk gaat bespreken wij de volgende keer.

Tot slot

Gebruik voor (karakter) strings bij voorkeur:

- puts()
- gets()

Voor het tonen van een kombinatie: literal string - variabele mag je ook i.p.v. 'puts()' de functie: 'printf()' gebruiken.

Hans van Boheemen

VIDEODAT, Een ander communicatie medium

Inleiding

Nadat in de zeventiger jaren de computers kwamen opzetten, kwam ook het probleem van uitwisseling van programma's. In 1978 is Hobbyscoop begonnen met proeven om programma's via de radio uit te zenden. Heel veel mensen dachten toen dat ze bezig waren met het testen van cirkelzagen. Het uitzenden van programma's werd een vast onderdeel toen een samenwerkingsverband met Teleac werd aangegaan in 1979. Omdat er toen vier computers populair waren in Nederland, te weten : Apple, Exidy Sorcerer, PET/CBM en TRS 80, kwamen elk van de merken maar eenmaal per maand aan de beurt, elk met zijn eigen geluiden. De KIM was in Hilversum niet bekend, bij de echte hobbyisten natuurlijk wel. Voor de programmamakers was die diversiteit een probleem, naar radio-normen bereikte Hobbyscoop een te klein publiek.

Hobbyscoop

De bekende zendamateur Klaas Robers kwam na klachten van Hobbyscoop met een schitterend technisch idee, dat later door samenwerking met vele computerfanaten en gebruikersgroepen zijn definitieve gestalte kreeg.

Hij kwam ook met de naam, ontleed aan Basic en code, kortweg BASICODE. Later werkte hij samen met Jochem Herrmann Basicode verder uit om het universeler te maken. Basicode deed zijn intrede in diverse landen. Tal van radio en TV stations namen het idee over, waaronder ook de WDR. Deze gebruikte Basicode om tijdens hun TV programma Computerclub programmatuur uit te zenden. Men heeft dit tot 1986 gedaan, maar de mensen gingen zich ergeren aan de hardbit rock van de programma's tijdens de TV uitzendingen. Men is toen bezig gegaan om programma's uit te zenden in het niet zichtbare deel van het beeldscherm. Bij sommige toestellen was soms een klein zwart blokje te zien links boven in beeld. Een nieuwe standaard was geboren: "Videodat 300".

Videodat 300

Sinds 1986 worden met meerdere wetenschappelijke en technische programma's van de WDR "Videodat" - informatie uitgezonden. Deze worden gemeenschappelijk met het TV beeld, in het niet zichtbare deel uitgezonden. Het kleine zwarte balkje in de linker bovenhoek van het scherm is het kenmerk van deze nieuwe overdrachtsvorm. Gedurende de data overdracht worden bepaalde regels in dit vlak wit uitgezonden. Dit gebeurt met een asynchroon, serieel communicatie protocol.

Figuur van het door de WDR gekozen formaat:

A = Synchronisatie pulsen
B = Color burst

			Regel in 1. halfbeeld	Regel in 2.halfbeeld	
A	B	zwart	23	336	
		zwart	24	337	
		wit	startbit	25	338
		zw/wz	1 data bit	26	339
		zw/wz	2. " "	27	340
		zw/wz	3. " "	28	341
		zw/wz	4. " "	29	342
		zw/wz	5. " "	30	343
		zw/wz	6. " "	31	344
		zw/wz	7. " "	32	345
		zw/wz	8. " "	33	346
		zwart	stopbit	34	347
		zwart	stopbit	35	348
		zwart		36	349

+ 17.5 micro sec +

Aan dit figuur kunt U zien dat de zwarte vlakjes beginnen in de 23^{e} lijn van het eerste halfbeeld en de 336^{e} lijn van het tweede halfbeeld. De begrenzing is de 36^{e} lijn van het eerste halfbeeld en de 349^{e} lijn van het tweede halfbeeld.

Het venster begint direct na het zwart niveau. Het startbit bevindt zich telkens in lijn 25 of 338, het is steeds wit.

Dit is vergelijkbaar met LOW op TTL niveau, de data bits gebruiken de lijnen 26 tot 33 en 339 tot 346. Aan de hand van de data informatie zijn deze lijnen zwart of wit. Hierna komen de stopbits, deze zijn altijd zwart.

Dan nu de snelheid van dit protocol:

In elk halfbeeld kan 1 teken verzonden worden. De televisienorm is dat elke seconde 50 halfbeelden worden verzonden. Dit geeft een data overdracht van 55 Byte oftewel 50*(1+8+2), geeft dit maximaal 550 Bit per seconde. Om compatibel te blijven met communicatiesoftware worden door de WDR maximaal 300 Bits doorgegeven.

Dit systeem heeft men ongeveer tot 1991 gebruikt. Er is toen een andere uitvoering gekomen van dit protocol. Wat ook betekende dat er een geheel

nieuwe decoder kwam. De naam is ook gewijzigd in WDR Videodat.

WDR Videodat

Voor overdracht worden steeds de 24e en 25e lijn van een raster gebruikt. In een lijn worden 2 tekens uitgezonden. Per seconde worden 25 even en 25 oneven rasters uitgezonden.

Dit is een overdracht capaciteit van:

2 tekens * 2 lijnen * 50 rasters/sec.

Dat is dus 200 tekens per seconde.

Gedurende de 45 minuten van een Computerclub uitzending kunnen

circa 45 * 60 * 200 = 540.000 tekens (540 kB) meegestuurd worden. 540.000 tekens zijn ongeveer 150 A4-tjes aan tekst. Naast teksten worden in Videodat ook grafieken en computerprogramma's uitgezonden. Deze Videodat informatie kan alleen ontvangen worden, als men beschikt over de Videodat decoder, welke men tussen de video uitgang van een TV of videorecorder en de printerpoort (LPT1) van een PC plaatst. Een stuk software zorgt er voor dat deze informatie in de computer terecht komt.

Al met al ging het toch niet al te snel, tot dat in 1991 de firma Wiegand startte met een commerciële uitvoering van Videodat.

De schakeling isoleert uit een raster de 24^{e} en 25^{e} lijn. Dat is 2 * 0.00064 sec. = 128 microsec. Uit de inhoud van deze lijnen wordt elke 0.5 microsec. duale samples (LOW is wit en HIGH is zwart) in een 256 staps schuifregister opgeslagen. Na opslag van alle 256 samples wordt via de printerpoort een interrupt verstuurt. Dan zorgt de software ervoor dat via een klok en een datalijn alle informatie van het schuifregister naar de PC wordt verstuurt. Hierna onderzoekt de software de 256 samples op start bits en isoleert de volgende 4 data bits. De geïsoleerde data wordt op het beeldscherm getoond en op diskette of hard disk weggeschreven. Na het einde van de uitzending wordt de file gesloten en kan dan bewerkt worden. Aangezien het een programma is wat in Duitsland uitgezonden wordt, is alle informatie en programma's in het Duits.

Al met al ging het toch niet al te snel, tot dat in 1991 de firma Wiegand startte met een commerciële uitvoering van Videodat. De naam werd ook weer veranderd, nu in Channel Videodat.

Channel Videodat

Hoe werkt nu deze uitvoering. De beeldinformatie wordt in twee delen van elk 312 lijnen aan de TV aangeleverd. De lijnen worden dus om en om ververst, om technische redenen zijn er meer lijnen dan er op het beeld worden vertoond. Deze worden gebruikt voor synchronisatie, teletekst of in ons geval Videodat data.

Het Videodat-systeem gebruikt alleen de lijnen 11-13 in het eerste en 324-326 in het tweede beeldgedeelte voor het doorgeven van een scala aan computerprogramma's en informatie. Elke transmissie begint na synchronisatiepulsen en colorburst met een startbit in ultra zwart (1V), vervolgens acht databits en tenslotte een stopbit (logisch nul) gevolgd door 2 seconden ultra zwart als afsluiting van de regel. In de databits wordt de 1 als een signaal van 0.7 V zwart verzonden, de logische nul als 0.2 V onder wit niveau. Hier tussen blijven dan voor de te verzenden data 50 seconden over. Bij een bitlengte van 500nSec. (frequentie 1 mHz) passen in elke regel dus 50 Sec/ 500 nSec = 100bit oftewel 10 hele bytes. Binnen drie beeldlijnen zijn dat dus 300 bits. Bij een beeld frequentie van 50 halfbeelden per seconde komen we op een totale bagage van 15000 bits per seconde. Dat is dus aardig wat meer dan de WDR een jaar of zes geleden aangaf.

Overigens worden de programma's in pakketten verzonden. Aangezien er meerdere pakketten gelijktijdig worden aangereikt en door de decoder naar keuze ook gelijktijdig kunnen worden ontvangen en via de bijbehorende software in files kunnen worden weggeschreven gaat de daadwerkelijke transmissie van een datapakket langzamer dan deze standaard toelaat.

Het decodersysteem is gebaseerd op individuele adressering, dat wil zeggen dat elke in gebruik zijnde decoder via een aparte code aan- of uitgeschakeld kan worden. Dit maakt het mogelijk om niet alleen free- of shareware te verzenden die voor iedereen ontvangbaar zijn, maar ook tegen betaling complete commerciële programma's te verhandelen. Dit laatste gebeurt op een zeer eenvoudige wijze. Men stuurt een bij de decoder verpakte questionaire in, bedenkt daarbij een eigen cijfercode en stuurt deze samen met het serienummer van de decoder naar

die firma toe. Eventueel met een bedrag in DM als beginstand van een conto. Hier bij krijgt men een diskette terug waar het bedrag is opgeboekt. Wil men nu een commercieel programma aanschaffen dan wordt het bedrag van de diskette afgeboekt. Persoonlijk geloof ik dat dit nu net de zwakste schakel in het systeem is. Videodat informatie wordt uitgezonden door PRO7, West 3 en TV Weiss-Blau. Het aanbod van PRO7 kan men ongeveer vergelijken met wat ook wordt aangeboden op de meeste BBS-en. Alleen komt het de laatste tijd steeds meer voor dat men voor programma's moet betalen. Dus wat men verdient aan het ontbreken van telefoonkosten geeft men weer uit aan betaling voor programma's. Sedert een half jaar neemt de WDR Computerclub ook proeven met dit systeem, en met ingang van 1-1-1994 gaan ze ook met deze norm werken, zodat ze compatibel zijn met de rest.

Herman Hek

Literatuur

1. Hobby Heft 4-1986, WAR Keulen.
2. Hobby Heft 2-1991, WAR Keulen
3. Satelliet Amateur magazine 3-1993

Voortgang 14 (aanvulling)

Er wordt nog al eens gezegd geen nieuws is goed nieuws. Dus eigenlijk ben ik al klaar. Maar toch wil ik jullie vertellen dat we op het moment "voorliggen" op het schema.

Hardware

We hadden gepland dat we met pasen (begin April) het prototype van de DRAM kaarten klaar hadden om te testen. Nu blijkt dat begin Maart al haalbaar is. De kaarten die reeds bestaan zitten gesamenlijk in een 19 inch rek waar ook een 3 1/2" disk drive, een SCSI harddisk en voeding in zitten. Dit geheel is nu beter handteerbaar en makkelijker mee te nemen naar Beurzen, KGN-bijeenkomsten & KGN68k werkgroepbesprekingen.

Software

Van de Assembler is er een nieuwe versie vrijgegeven. De vooruitgang is de mogelijkheid om GNU-objects te generen en de ondersteuning van macro's. Wat er nu in EPROM zit zijn we tijdelijk tevreden mee. De volgende stap voor de (boot)rom is dat de diverse files samen gevoegd worden de GNU-linker.

Printed Circuit Board ontwerp

Door wat met de taken te schuiven hebben we weer iemand die zich kan concenteren op het ontwerpen van de prints.

Geert Stappers

Linux vs HP-UX

Voor een kleine toepassing op het werk heb ik een PC gekaapt waarop ik Linux heb geïnstalleerd (SLS versie die ook op de ULTIMATE staat). In combinatie met een 'C'-applicatie verzamelt de Linux PC gegevens uit onze telefooncentrale.

Omdat wij ook HP UNIX machines hebben staan en de PC ook op het netwerk van deze computers is aangesloten heb ik eens geprobeerd Linux en HP met elkaar te laten praten. Mijn eerste probleem was dat de PC geen compatible netwerkkaart had, dus die heb ik maar snel even verwisseld met een NE2000 kaart uit een andere PC. Na het instellen van de juiste hostadressen en het in de lucht brengen van de juiste deamons (portmap, inetd) kon ik proberen of het werkte.

In de tabel staan verschillende dingen die ik geprobeerd heb. De meeste werkten goed.

Het laatste kan ik nog niet goed voor elkaar krijgen, het lijkt eerst te werken maar na een poosje (10 min) geeft de linux nfs deamon foutmeldingen, waarna deze ermee stopt.

Dingen die ik nog uit wil proberen zijn:

- Linux machine als werkstation te gebruiken met X windows, en applicaties te draaien op de HP (X windows applicaties). Dit is gedeeltelijk gelukt want ik heb nog niet alle software op de linux machine goed geinstalleerd.
- Mail versturen tussen beide machines.
- Linux filesystem op hp mounten (zonder dat er fouten optreden).

Jan Veninga

remote login van linux naar hp:	rlogin hp -l jan
remote login van hp naar linux:	rlogin linux -l jan
telnet sessie starten:	telnet open linux of telnet open hp
hp filesystem mounten onder linux:	mount -t nfs hp:/ /mnt
linux filesystem mounten onder hp:	mount -t nfs linux:/ /mnt

KGN promotie team

Jullie als lid van de KGN mogen deel uitmaken van het KGN promotie team.

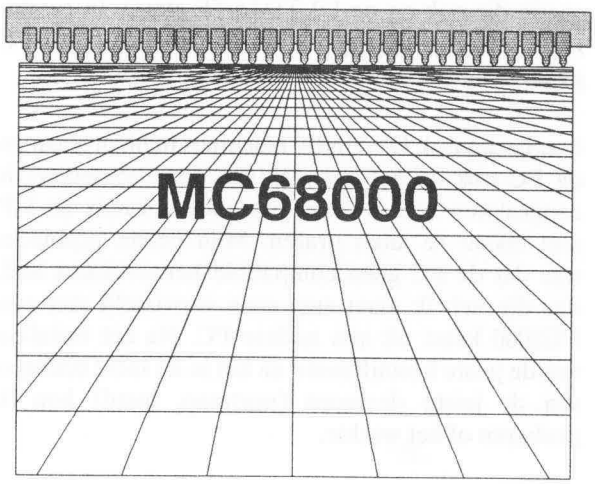
Het is gelukkig geen full-time job, want we hebben het waarschijnlijk toch wel druk genoeg. We zoeken een groep mensen die bereid zijn eventueel een stand te bemannen tijdens een beurs of ander evenement. Als je nu ja zegt betekent dat niet dat je straks nog aan die verplichting vastzit. Pas als de KGN naar een namelijk in het aansluiten van de te demonstreren computer(toepassing). Tijdens de beurs zelf promotiemateriaal uitdelen, eventueel wat vertellen. In overleg met andere KGN promotie team leden wie er bij de stand blijft en wie er even over de beurs wandelt. Aan het einde van de dag de zaken weer netjes afbreken.

Hoe wordt je KGN promotie team lid?

Een briefje, bericht of telefoontje naar Geert Stappers!

Voortgang KGN68k, deel 14

Bijgeloof wordt in de computerwereld niet gebruikt, maar we zijn wel aflevering 13 voorbij en het gaat weer goed. Dombo68k, de diskcontrollerkaart werkt! Eens proberen of ik het succes verhaal kan vertellen, voor alle pikante details is helaas tijd, want ik ben weer laat.



Hardware

Na enkele mislukte pogingen om de Dombo printplaat met de hand door te metalliseren, zijn we overgegaan naar wire-wrap. Met de processorkaart hadden we goede ervaringen met 'draadjes-draaien' opgedaan. Omdat het wel gelukt was om de Combo68k PCB zelf door te metalliseren, hebben we het toch eerst geprobeerd. Achteraf jammer van de tijd en moeite. Willem, die destijds ook Combo in elkaar heeft gezet, heeft de meeste draadjes gedraaid. 's Maandags na het sinterklaasweekend kreeg ik een telefoontje van hem. "Geert, hij is af, op de GND en VCC na. Ik zal voorlopig geen tijd hebben om verder te gaan, kun jij het verder afmaken?" "Laat maar komen, wire wrappen is een goede afwisseling voor mij." stemt ik toe. Vervolgens heb ik een afspraak voor een testsessie bij Ad Brouwer thuis gemaakt. 12 december konden we de registers van de floppy disk controller lezen en schrijven. 19 december was Ad in Overloon te gast, die dag hebben we de diskdrives aangesloten en laten stappen, die sector van diskette afgehaald, maar nog niet opgeslagen in RAM. De SCSI controller zijn registers gelezen & geschreven. Een SCSI device tevergeefs aangesproken. En geconstateerd dat er met de chipselects van de FIFO nog wat moest gebeuren. Woensdag 29 december was Ad opnieuw in Overloon. De modificatie op de processorkaart had ik ondertussen uitgevoerd, maar had Ad nodig om hem weer aan de praat te krijgen. Die woensdag was een zeer vruchtbare dag: Lezen van floppy, via de FIFO, het geheugen in. Schrijven op floppy. Select van een SCSI device. Met behulp van Pseudo DMA bytes van een SCSI device gelezen. Serial I/O onder interrupt. Allemaal die dag aan de praat gekregen.

Allemaal die dag aan de praat gekregen.

Software

Zaterdag 18 december was de werkgroep bij elkaar in Udenhout. Daar hebben we besloten om in software hetzelfde te doen als wat we met de hardware gedaan hebben. Werken met kleine overzichtelijke stukken. Toen we destijds de printplaat in kleinere stukken hebben gekapt, ging het met sprongen vooruit. Dat is wat ik nu ook met de software zie gaan gebeuren. Ad heeft voor zijn 68008 machine een Operating System geschreven, een versie voor de KGN68k is er ondertussen al. Daar bovenop draait een command line interpreter, die nogal wat interne commando's kent en als je een andere opdracht geeft wordt er op schijf gekeken. Die schijf is een MS-DOS compatible diskette waar behalve executables ook batch files op kunnen staan. En dan ook nog uitgevoerd kunnen worden. We kunnen nu dus gebruik maken MS-DOS floppies als download medium. Door de mogelijkheden die nu zijn gaan we RAM te kort komen.

Printed Circuit Board ontwerp

Het lay-outen van de prints gaat nu op een langzamer tempo door. Hugo heeft zich verkiesbaar gesteld voor de gemeenteraad, tot mei 1994 zijn we hem zeker kwijt als werkgroeplid. Door zijn hoge plaats op de lijst waarschijnlijk ook. Wat er nu als eerste moet gebeuren is het bepalen van een componenten opstelling voor de DRAM kaart. Zodat we daar gemakkelijker een prototype van kunnen gaan bouwen.

Zo en nu zorgen dat deze bijdrage nog op tijd bij de layouter is.

Geert Stappers.

IBM, Apple en Motorola verenigd in de Power PC

Inleiding

Op dit moment heb je zo grofweg twee computersystemen die de markt beheersen. In de eerste plaats zijn dat natuurlijk de nazaten van de IBM PC met als hart een Intel 0, 2, 3 of 486, een Pentium of een kloon van één van deze chips. Verder heb je de Macintosh, een produkt van de firma Apple, gebaseerd op de Motorola 68000 familie.

Sinds kort is er een nieuwe familie computers op de markt waarvan mijn verwachtingen in ieder geval heel hoog gespannen zijn. Het betreft een produkt van een samenwerking van drie groten in de wereld van chips en computers namelijk de zogenaamde Power PC, het produkt van een samenwerkingsverband van de firma's Apple, IBM en Motorola. Er is zo hier en daar al wat literatuur over deze PC en de chip die er in zit verschenen, hoog tijd dus om er ook in de μP Kenner eens aandacht aan te gaan besteden.

Het hart van de Power PC is ontwikkeld vanuit een computersysteem van IBM, het RISC System/6000 of RS/6000. Deze machine is al enige tijd (sinds 1990) op de markt en wordt vooral gebruikt in situaties waarin een PC te klein is en een AS/400 (mid-range computer) of een mainframe te groot. Het hart van deze machine bestaat uit een processor met de naam POWER van Performance Optimized With Enhanced RISC. Deze processor is in staat meerdere instructies per klokcyclus uit te voeren. Hierbij gaat men nog verder dan bij de meeste RISC processoren die per klokcyclus één instructie uitvoeren.

Deze POWER processor uit de RS/6000 bestaat niet uit één plakje silicium met daarop de complete CPU, maar uit meerdere losse chips die door middel van een door IBM gepatenteerde Multi Chip Carrier met elkaar zijn verbonden. Op deze manier kan deze processor op eenvoudige wijze uitgebreid worden met extra cache of een bredere databus.

Deze POWER processor uit de RS/6000 bestaat niet uit één plakje silicium.

Een jaar na de introductie, in oktober 1991, hebben IBM, Apple en Motorola een samenwerkingsverband AIM opgericht met als opdracht een fabrikantonafhankelijke en een vrij beschikbare micro-processor te ontwikkelen op basis van de POWER processor van IBM. Deze microprocessor kreeg de naam Power PC naar "POWER Processor Chip". Verder is op basis van de micro-processor een complete specificatie gemaakt voor een nieuwe familie van personal-computers, maar daarover straks meer. Om deze chip te kunnen ontwikkelen, hebben de drie firma's in mei 1992 in Austin, Texas het ontwikkelingscentrum "Sommerset" geopend waar zo'n 300 ontwikkelaars gingen werken. Hiervan waren 10 man van Apple afkomstig en de rest kwam voor de helft van IBM en de helft van Motorola. De mensen van IBM brachten de kennis van de RS/6000 processor in, de Motorola-mensen de kennis voor het ontwerpen van RISC micro-processors (uit de 88100 familie) en de

Processor	Pentium	468DX2	R4400SC	SPARC+	Alpha21064	MPC 601
SPECint92	64,5	34	88	65,2	84,4	60
SPECfp92	59,9	16,1	97	83	127,7	80
Ext. Cache	256 kB	256 kB	1 MB	1 MB	512 kB	-
On Chip Cache	16 kB	8 kB	32 kB	36 kB	16 kB	32 kB
Clock freq.	66 MHz	66 MHz	150 MHz	50 MHz	150 MHz	66 MHz
Techniek	BiCMOS	CMOS	CMOS	BiCMOS	CMOS	CMOS
	0,8 μm	0,8 μm	0,6 μm	0,7μm	0,75 μm	0,65 μm
Metaallagen	3	3	3	3	3	4
Vermogen	15 W	9 W	-	-	-	9 W
Afmeting	264 mm^2	82 mm^2	184 mm^2	256 mm^2	234 mm^2	121 mm^2
Transistoren	3,1 M	1,2 M	2,2 M	3,1 M	1,7 M	1,8 M
Prijs	$995	$542	$1120	$1199	$1096	$450

Tabel 1: vergelijking tussen enkele moderne processoren

Apple mensen waren specialist op het gebied van operating systems.

In oktober 1992 was het zo ver. De eerste Power PC werd aangekondigd. Het betrof de MPC 601 met 2,7 miljoen transistoren in een structuur van 0,65 μm. De oppervlakte van de chip bedraagt 121 vierkante mm. Nog weer een half jaar later waren de eerste MPC 601 chips beschikbaar. Intussen is ook de volgende chip beschikbaar. Dit is de MPC 603 die speciaal ontwikkeld is voor notebook computers. Verder zijn er nog twee chips aangekondigd: de MPC 604 die een verbeterde MPC 601 is en de MPC 620 die speciaal voor multi-processor omgevingen (meerdere CPU's in één computer) bedoeld is. Deze chips komen in het derde resp. het vierde kwartaal van dit jaar beschikbaar.

Vergelijking tussen enkele nieuwe processoren

Om de plaats van de Power PC te kunnen bepalen zijn in tabel 1 enkele eigenschappen van moderne processoren afgedrukt. De Pentium en 486DX2 zijn de nieuwste loot aan de Intel 80x86 lijn en zijn voorganger. De R4400SC is een processorchip van de firma MIPS die te vinden is in enkele Unix Workstations. De SPARC+ is een RISC-processor die gebruikt wordt in de SPARC workstations van SUN en de Alpha 21064 is de nieuwe Risc-processor van de firma Digital.

In de tabel is de prestatie van een processor uitgedrukt in de zogenaamde SPECMarks, een combinatie van testprogramma's die enerzijds het gedrag van de processor meten bij gebruik van gehele getallen (SPECint92) en anderzijds speciaal op het gebied van floating point getallen testen (SPECfp92). Deze tests zijn ontwikkeld in 1992.

We zien in de tabel dat de Power PC geen slecht figuur slaat in vergelijking met de Pentium. Ten opzicht van de Alpha moet hij nog wel wat toegeven, maar deze chip wordt momenteel ook aangeduid als de snelste microprocessor ter wereld. We moeten echter wel bedenken dat de MPC 601 nog maar het begin is van een nieuwe en waarschijnlijk zeer interessante lijn van micro-processors zodat de vergelijking er over een jaar best heel anders uit kan zien.

Van de processor-chips in de tabel zijn de eerste twee zogenaamde CISC, van Complex Instruction Set Computer, processoren; processoren met veel en vaak zeer complexe instructies. De overige zijn RISC processoren van Reduced Instruction Set Computer. Dit type processor kent minder instructies die over het algemeen veel eenvoudiger van opzet zijn. Neem als voorbeeld het geval dat alle rekenkundige operaties in de Power PC op registers uitgevoerd worden. Dit houdt bijvoorbeeld in dat we om twee getallen bij elkaar op te kunnen tellen we ervoor moeten zorgen dat deze getallen eerst beide in de interne registers staan. Ook het resultaat komt in een register terecht. Voor het laden en terugschrijven van registers heeft de processor aparte instructies die meerdere registers in één keer op kunnen halen of weg kunnen schrijven, waarbij optimaal gebruik wordt gemaakt van de mogelijkheid van het geheugen om snel opeenvolgende geheugenplaatsen te benaderen (burst mode). Kenmerk van een RISC processor is dat de processor veel tot zeer veel registers heeft. De Power PC heeft 32 integer registers, 32 floating point registers en nog enkele tientallen registers voor speciale doeleinden; veel meer dus dan de gemiddelde CISC processor.

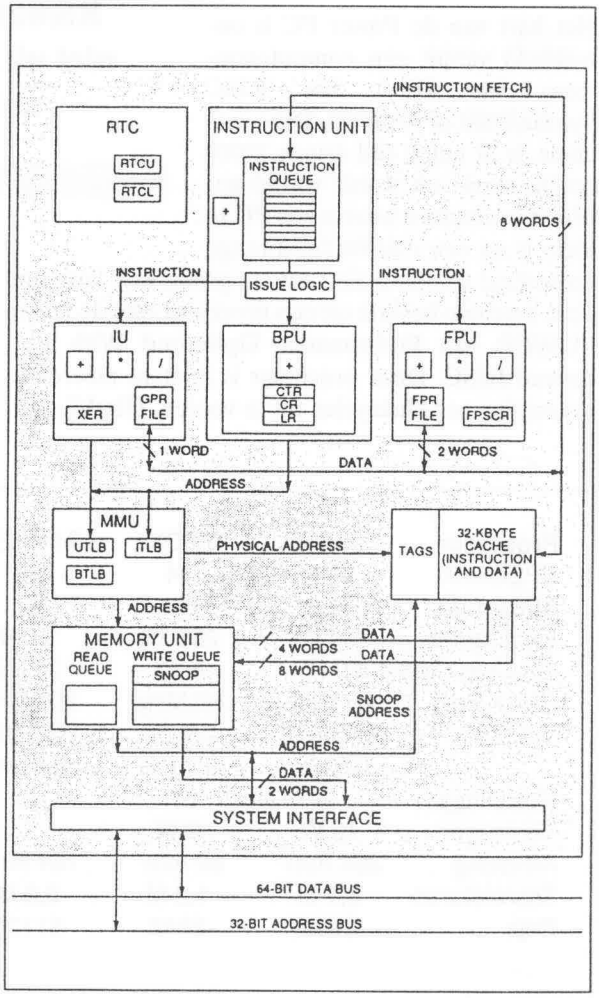


Fig. 1: interne opbouw van de MPC 601 Power PC

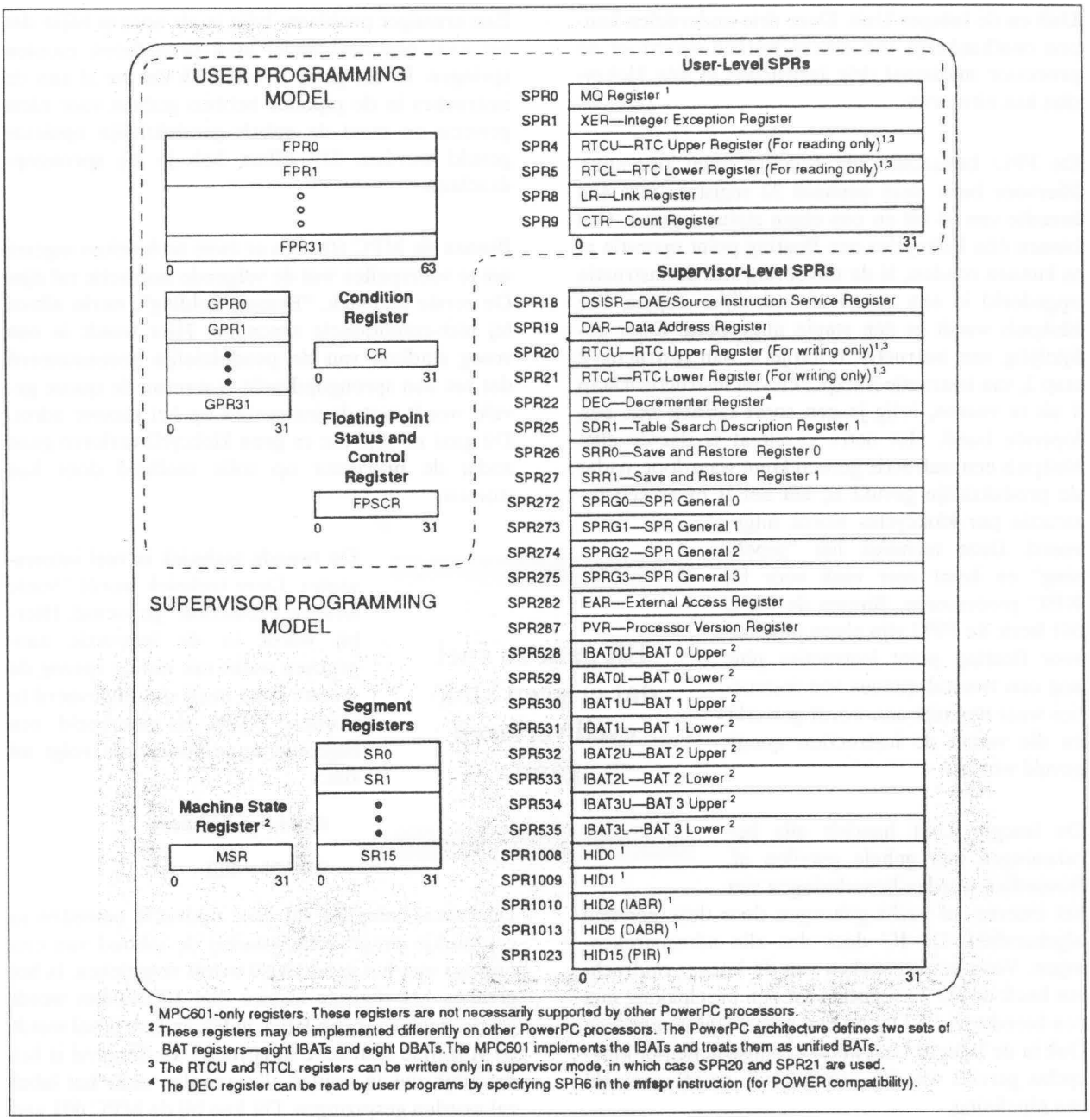


Fig. 1: programmeermodel van de MPC 601 Power PC

Architectuur

In figuur 1 is de inwendige opbouw van de Power PC MPC 601 schematisch weergegeven. Verder is in figuur 2 het programmeermodel van deze chip afgedrukt. We zien dat de chip een 64 bits brede databus bezit en een 32 bits brede adresbus. Binnen in de chip zien we rechts boven de Real Time Clock. Deze bestaat uit twee registers met een breedte van 32 bits. Van het laagste register worden slechts 24 bits gebruikt. De RTC wordt aangestuurd door een klokpuls op een aparte pen van de processor. Motorola adviseert hier een kristal van 7,8125 MHz op aan te sluiten zodat de RTC telt in eenheden van 128 ns. De twee registers samen geven dan een periode van 136 jaar aan wat voor de meeste toepassingen wel voldoende is.

Verder naar rechts zien we de zogenaamde "Instruction Unit". Dit is het deel van de processor dat het (laten) uitvoeren van de instructies voor zijn rekening neemt. Onderdeel van deze eenheid is de zogenaamde "Instruction Queue" met een lengte van 8 instructies. De gehele queue kan in één keer (in één klokcyclus) uit de cache worden geladen.

Voor het uitvoeren van de instructies heeft de processor drie aparte verwerkings-eenheden. Dat zijn de Floating Point Unit (FPU), de Branch Processing

Unit en de Integer Unit. Deze drie onderdelen kunnen onafhankelijk van elkaar werken waardoor de processor maximaal drie instructies in één klokcyclus kan uitvoeren.

De FPU behandelt alle floating point operaties. Hiervoor bezit deze eenheid 32 registers met een breedte van 64 bit en een eigen status register. Om binnen één klokcyclus een floating point operatie af te kunnen ronden, is de uitvoering van de instructie opgedeeld in een aantal afzonderlijke stapjes. Per klokpuls wordt er één stapje uitgevoerd. Door gelijktijdig van instructie 1 stapje 4, van instructie 2 stap 3, van instructie 3 stap 2 en van instructie 4 stap 1 uit te voeren, krijg je een soort fabriek met een lopende band. Het netto resultaat is dat er elke klokpuls een instructie gereed komt waardoor, nadat de produktielijn gevuld is, het net is of er één instructie per klokcyclus wordt uitgevoerd. Deze techniek het "pipelining" en komt zeer vaak voor bij RISC processoren. Binnen de MPC 601 bezit de FPU zijn eigen pipeline voor floating point instructies plus nog een tweetal queues van instructies waar nog niet aan wordt gewerkt en die vanuit de instruction queue gevuld worden.

De Integer Unit handelt alle berekeningen met gehele getallen af. Bovendien worden benaderingen van het externe- of cache-geheugen door deze eenheid afgehandeld. De IU doet dus alle adresberekeningen. Voor het verwerken van de integer-opdrachten heeft de IU 32 registers tot zijn beschikking met een breedte van 32 bit en een aantal status registers. Ook in de Integer Unit komt één instructie per klokcyclus gereed waarbij ook gebruik wordt gemaakt van pipelining.

De laatste eenheid is ook de meest interessantste. Dit is de zogenaamde Branch Processing Unit. In de vorige alinea's is het begrip "pipelining" al naar voren gekomen. Deze techniek werkt zeer goed als het resultaat van een instructie niet afhangt van het resultaat van de vorige instructie en als alle instructies strikt in volgorde worden uitgevoerd. Zouden we bijvoorbeeld in een register in twee instructies drie getallen bij elkaar op willen tellen, dan kunnen we pas daadwerkelijk met de tweede optelling beginnen als het resultaat van de eerste optelling bekend is. Dit effect kan een vertraging in de produktielijn geven omdat de uitvoering van de tweede optelling even moet wachten op de eerste optelling.

Een ernstiger probleem krijg je als opeens blijkt dat we naar een heel ander stuk programma moeten springen. In dat geval is het werk dat we al aan de instructies in de pipeline hebben gedaan voor niets geweest en moet de gehele produktielijn opnieuw gevuld worden. Dit effect heb je bij sprongopdrachten.

Binnen de MPC 601 zijn er twee technieken ingezet om te voorspellen wat de volgende instructie zal zijn. De eerste techniek, "Branch Folding", werkt alleen bij niet-conditionele sprongen. Hier wordt in een vroeg stadium van de produktielijn geconstateerd dat het een sprongopdracht is waarna de queue gevuld wordt met de instructies op het nieuwe adres. Dit gaat zo snel dat er geen klokcycli verloren gaan zodat de processor op volle snelheid door kan stomen.

Dit gaat zo snel dat er geen klokcycli verloren gaan.

De tweede techniek is veel interessanter. Deze techniek wordt "Static Branch Prediction" genoemd. Hierbij wordt in de instructie aangegeven welke tak van de sprong de meeste kans heeft om uitgevoerd te worden. Neem als voorbeeld een lusje die er in Pascal als volgt uit ziet:

```
FOR i := 1 TO 10000
DO A[i] = B[i]
```

De Pascal compiler zal deze opdracht omzetten in een stukje programma waarbij de inhoud van een register met het getal 10000 wordt vergeleken. Is het getal in het register kleiner dan 10000, dan wordt gesprongen naar een label, in het andere geval wordt de volgende instructie uitgevoerd. In dit geval is het duidelijk dat in vrijwel alle gevallen naar het label zal worden gesprongen. Dit kan bij de MPC 601 aan de processor worden verteld waardoor de BPU er vanuit zal gaan dat de sprong altijd uitgevoerd zal worden zodat hij zijn instructie-queues in 99,99 % van de gevallen met de juiste instructies zal vullen.

Naast de verwerkings eenheden bezit de processor nog een MMU en een gecombineerde instructie- en data-cache met een afmeting van 32 kilobyte.

Laten we nu verder gaan met het programmeermodel. De MPC 601 kent, evenals de 68000-familie van Motorola twee toestanden van de processor. Het betreft een User Mode en een Supervisor Mode. Hierbij is Supervisor Mode bedoeld voor het Operating System en de User Mode voor de gebruikersprogramma's. Wat de processor ook heeft, en dat is echt uniek, is een interne vlag waarmee kan worden aangegeven of de byte-volgorde in het geheugen van

meest- naar minst significant loopt (zoals bij de Motorola 68000) of andersom (zoals bij Intel). De reden hiervoor is eigenlijk heel duidelijk als je de plannen van IBM en Apple leest. Het is de bedoeling dat IBM diverse eigen operating systemen naar de Power PC gaat porteren waaronder naar alle waarschijnlijkheid ook OS/2. Bovendien zal waarschijnlijk ook Windows NT geporteerd gaan worden. Apple gaat zijn nieuwe generatie Macintosh-machines uitbrengen met de Power PC zodat in dat geval op de processor programmatuur moet gaan draaien die voorheen op een 68000 draaide. Sterker nog, er komt een emulator waardoor de Power PC een 68040 op 25 MHz simuleert. Op deze manier slaat de Power PC dus een brug tussen twee heel verschillende werelden.

Gaan we terug naar het programmeermodel, dan valt op dat we een conditie-register hebben met een breedte van 32 bits. Dit register is opgedeeld in 8 registers met een breedte van vier bit. Eén van deze delen is speciaal voor integer-berekeningen, één is er voor de floating point unit en de overige zes zijn voor vergelijkings-opdrachten (compare). Hierbij zijn de zes delen gelijkwaardig zodat in de compare-instructie moet worden aangegeven welke van de zes conditie-code registers gebruikt moet worden; hetzelfde geldt voor de sprongopdrachten. Dit heeft als voordeel dat je meerdere compare's na elkaar kunt doen zonder dat het resultaat van de voorgaande compare door de nieuwe compare wordt overschreven.

Op deze manier slaat de Power PC dus een brug tussen twee heel verschillende werelden.

Omdat de integer-registers een breedte hebben van 32 bits, is er een apart register, het MQ register die als uitbreiding van deze registers bedoeld is. Vermenigvuldigen we bijvoorbeeld twee 32 bit getallen met elkaar, dan ontstaat een 64 bit getal. Hiervan staan er dan 32 in het MQ register. Nu is het wel de bedoeling dat in de toekomst, bij nieuwe versies van de Power PC familie, dit register komt te vervallen omdat alle integer registers ook een breedte van 64 bit zullen krijgen.

Een ander aardig register is het Link Register. Hierin kan bij het uitvoeren van een branch-instructie het adres van de instructie volgend op de branch geschreven worden zodat men eventueel gemakkelijk terug kan springen. Iets dergelijks geldt ook voor het Count Register. In dit register kan men o.a. laten tellen hoe vaak een branch is uitgevoerd zodat je gemakkelijk lussen zoals in het Pascal-voorbeeld kunt programmeren.

Het laatste register dat in User Mode bereikbaar is wordt Integer Exception Register genoemd. Hierin worden de resultaten van Integer berekeningen weggeschreven (Carry, Overflow etc.).

Uiteraard is er over de architectuur van de Power PC nog veel meer te vertellen. Zo zijn er naast de behandelde registers nog een groot aantal registers die alleen in Supervisor Mode te benaderen zijn. Bovendien hebben we het helemaal nog niet gehad over de instructie-set en de adresseermogelijkheden. Voor dit moment lijkt het mij echter wel voldoende. Als mijn verwachtingen uitkomen, dan zullen we op deze zaken in de toekomst zeker nog terugkomen.

Het Power PC Reference Platform

Naast de Power PC chip hebben de samenwerkende firma's ook een document vervaardigd waar in staat hoe een computer-systeem rond een Power PC opgebouwd moet zijn. Hierin staan details over geheugen-indeling, randapparatuur, manier van opstarten etc. Het doel van deze specificatie is ervoor te zorgen dat software die voor het ene systeem met een Power PC ontwikkeld is ook zal kunnen draaien op een systeem van een andere fabrikant. Hiermee hoopt men de compatibiliteits-problemen die in de begintijd van de IBM-PC optraden te voorkomen. Verder staat er in hoe de interactie tussen gebruikersprogramma's en de hardware moet verlopen. Dit gaat door middel van een tussenlaag waarvan de interfaces naar de gebruikersprogramma's nauwkeurig gespecificeerd zijn. Dit is een kenmerk van zogenaamde "Open systemen" waar ik in de volgende μP Kenner nog een artikel over wil schrijven.

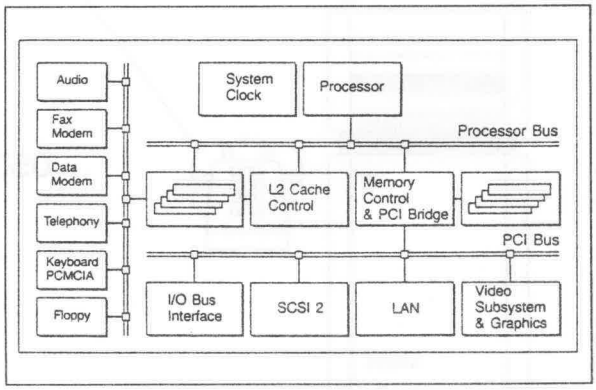


Fig. 3: voorbeeld van een Power PC configuratie

Als voorbeeld is in figuur 3 een overzicht gegeven van een mogelijke systeemconfiguratie.

Meer details over dit onderwerp zijn te vinden in het artikel in de MC van maart 1994.

Afsluiting

De tijd heeft sinds de aankondiging van de MPC 601 niet stil gestaan. In de PCM van januari 1994 wordt een systeem rond deze processor getest. Het betreft de IBM RS6000/250T. Dit systeem is voor ongeveer fl. 35.000,-- exclusief btw te koop. De verwachting is echter dat de systemen rond de Power PC nog wel een stuk goedkoper zullen worden. Momenteel is als operating system de eigen IBM versie van Unix, AIX beschikbaar. Op termijn zullen onder ander Windows NT, OS/2, System 7 (Macintosh) en Macintosh Application Services ook beschikbaar komen. De laatstgenoemde (MAS) emuleert op een Power PC een Motorola 68040 waardoor ook oude Macintosh applicaties op de Power PC blijven draaien. In dat geval zal een systeem ongeveer even snel zijn als een Quadra 700 met een 25 MHz 68040.

Verder las ik onlangs in de Computable dat IBM ook de AS/400 en het systeem 36 wil gaan emuleren op de Power PC om zo de gebruikers die de sterk verouderde en afgeschreven 36 machines nog gebruiken een alternatief te bieden.

Naar mijn bescheiden mening staan we aan de vooravond van een nieuw computer-tijdperk en ik denk dat de Power PC een belangrijk processorfamilie zal worden. Zelf heb ik in elk geval besloten mijn 10 MHz XT met NEC V20 voorlopig niet in te ruilen voor een 486 racemonster maar even rustig de ontwikkelingen af te wachten.

Gert van Opbroek

Referenties

Dit artikel is gebaseerd op de volgende artikelen:

1: Henrik Fisch: Power-PC: Der Mikroprozessor; mc november 1993
2: Henrik Fisch: Status Quo - Power PC; mc februari 1994
3: Henrik Fisch: Hardware des Power-PC; mc maart 1994
4: Dennis Kuit: Voorschot op de toekomst; PCM januari 1994
5: Wisse Hettinga: Toekomst in een Pizzadoos; PCM januari 1994
6: Ferdinand Sennema: Alles in één emmer; PCM januari 1994

Van de voorzitter

Dag KGN-leden,

Eerst maar eens de beste wensen voor 1994. Behalve dat jullie aan een nieuw jaar zijn begonnen, hebben jullie ook een nieuwe voorzitter.

Geert Stappers is nu ook aanspreekbaar als voorzitter van de KGN. Ik ben al tijden aanspreekbaar als bestuurslid, net als alle andere bestuursleden. De adressen staan gewoon achterin De μP Kenner. Meestal is de voorzitter wat ouder dan de gemiddelde leeftijd van de clubleden. Dat ik onder dat gemiddelde zit zie ik als een uitdaging, maar weet ook dat ik het een verantwoordelijk taak is. Beloftes wil ik niet maken, maar gewoon dingen waar maken. Een van de dingen die waar ik samen met anderen met bezig ben is het KGN68k project. Iets wat er aan het komen is, niet zo snel als eerst gewild, maar toch.

Geert Stappers is nu ook aanspreekbaar als voorzitter van de KGN.

Bestuur

Als voorzitter ook informatie van het bestuur. Het ledental is het afgelopen gelijk gebleven, zelfs iets toegenomen. Het aantal bestuursleden is ook weer bijna op gewenste (7) sterkte. Zes man is een hele verbetering ten opzichte van vier. Het beste nieuws is eigenlijk wel dat we weer DOS65 mannen, te weten Henk Speksnijder & Nico de Vries, in het bestuur hebben. Dit is interessant voor de 'installed base' en de toekomst met DOS65 Versie 3.0.

1994

Vertel in de nieuwe jaren wel in De μP Kenner wat voor jouw de (technische) computer toepassingen zijn. Jullie zijn lid van een vereniging, wat jullie verenigt is de interesse voor de technische kant van de computer, vertel er over. De beste stuurlieden blijven toch aan wal staan, maar die <MI>lui<D> interesseren me niet, wel de mensen die zelf varen.

Aanvulling

Het late verschijnen van de μP kenner heeft ook (kleine) voordelen. Men kan diverse zaken nog aanvullen. Over geen beloftes maken, maar gewoon doen. Public Relation (PR) is bij ons techneuten eigenlijk een ondergeschoven kindje. Wij als bestuur hebben er dit jaar al wat aan gedaan. De KGN was vertegenwoordigd op COMPUTERWARE beurs van 19 februari in Utrecht. Stappenmotoren, zelfbouw & Linux hadden wel degelijk belangstelling bij het op koopjes jagende publiek. Hoe we het de volgende keer beter kunnen doen weten we ook al. Hiervoor hebben we het KGN promotie team in het leven geroepen.

Geert Stappers

KGN promotie team

Jullie als lid van de KGN mogen deel uitmaken van het KGN promotie team.

Het is gelukkig geen full-time job, want we hebben het waarschijnlijk toch wel druk genoeg. We zoeken een groep mensen die bereid zijn eventueel een stand te bemannen tijdens een beurs of ander evenement. Als je nu ja zegt betekent dat niet dat je straks nog aan die verplichting vastzit. Pas als de KGN naar een namelijk in het aansluiten van de te demonstreren computer(toepassing). Tijdens de beurs zelf promotiemateriaal uitdelen, eventueel wat vertellen. In overleg met andere KGN promotie team leden wie er bij de stand blijft en wie er even over de beurs wandelt. Aan het einde van de dag de zaken weer netjes afbreken.

Hoe wordt je KGN promotie team lid?

Een briefje, bericht of telefoontje naar Geert Stappers!

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 de diverse 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 (BBS) beschikbaar gesteld.
De telefoonnummers zijn: 053-328506, 053-303902 of 053-327457.

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 BBS.

Correspondentie adres

Alle correspondentie betreffende verenigingszaken kan gestuurd worden aan:

KIM Gebruikersclub Nederland
Postbus 1336
7500 BH Enschede

Het Bestuur

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

Geert Stappers (Voorzitter, KGN/68k coordinator)
Engelseweg 7
5825 BT Overloon
Telefoon 04781-41279

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

Gert van Opbroek (secretaris)
Del Del 16
5071 TT Udenhout
Telefoon 04241-3795

Jan Veninga
Klimopstraat 51
7601 SJ Almelo
Telefoon 05490-27910

Henk Speksnijder
Albert Cuijpstraat 43
2902 GA Capelle aan den IJssel
Telefoon 010-4586879

Nico de Vries (redactie μP Kenner)
Van der Waalsstraat 46
2984 EP Ridderkerk
Telefoon 01804-29207

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