



‘Master mind op de KIM’

J. M. van der Peijl en D. M. de Boer

U kent waarschijnlijk wel het bekende Mastermind spel. Normaal wordt dit spel gespeeld door 2 personen, waarbij eigenlijk maar 1 persoon actief bezig is met het raden van een verborgen code. De tweede persoon heeft de passieve taak om met zwarte en witte penntjes aan te geven of de gekozen kleuren goed zijn en of zij op de goede plaats staan. Vervang de kleuren door cijfers, de zwarte en witte penntjes door letters en de KIM neemt de plaats in van de passieve speler. Hij kiest op uw commando een geheime kleurencombinatie en geeft met letters aan hoeveel cijfers juist gekozen zijn. Een speciale ‘spiek knop’ geeft de mogelijkheid om stiekem achter het schotje te kijken, de consequentie is wel dat de beurtenteller met 30 punten wordt opgehoogd.

Gebruiksaanwijzing

Als u het hele programma heeft ingetypt (adressen 0200 ... 03D) kunt u op adres 0200 het programma starten. Het display wordt dan donker en de KIM zal op uw commando steeds 1 cijfer (kleur) in zijn geheugen zetten. Na een aantal drukken op de knop is de geheime combinatie klaar en het display licht op.

Bij het kiezen van de combinatie hebben we de volgende mogelijkheden.

Enige keren drukken op toetsnummer	Elk cijfer ligt tussen:	Van de 4 cijfers zijn er dan max:
0	0..6	4 hetzelfde
1	0..6	3 hetzelfde
2	0..6	2 hetzelfde
3	0..6	4 verschillende
4	1..6	4 verschillende
5	0..6	4 verschillende
Andere toets	Geen actie	

Op deze manier kan het spel dus in verschillende moeilijkheidsgraden worden gespeeld. Met het oplichten van het display geeft de KIM aan dat de combinatie in het geheugen staat. De twee meest rechtse displays vormen beurtenteller, op de 4 linkse displays verschijnt de door u geprobeerde combinatie. Mocht u een verkeerde toets hebben ingedrukt, dan kunt u de combinatie gewoon opnieuw intypen. Bij een druk op de E (enter) wordt de door u geprobeerde combinatie vergeleken met de geheime combinatie in het geheugen. De KIM antwoordt met een C voor elke goede kleur die niet

op de goede plaats staat en met een A voor elke goede kleur die wel op de goede plaats staat. De plaats van de A's en de C's zeggen niets over de plaats van de goede kleur. Als u de volgende combinatie wilt proberen, moet u eerst met een willekeurige toets de beurtenteller ophogen en het display schoonmaken. In hopeloze gevallen geeft een druk op knop 'C' de oplossing terwijl de beurtenteller met 30 opgehoogd wordt. Een goede oplossing geeft 'AAA' met daarachter het aantal beurten. Het display gaat een aantal malen aan en uit en blijft tenslotte donker. Nu kan een nieuwe geheime code gemaakt worden voor het volgende spelletje.

Beoordelingssystemen

Nadat de redactie een aantal spelletjes op KIM gespeeld had bleek dat voor het zetten van de witte en zwarte pennetjes verschillende methoden bestaan.

Stel de situatie:

0113 geheime code

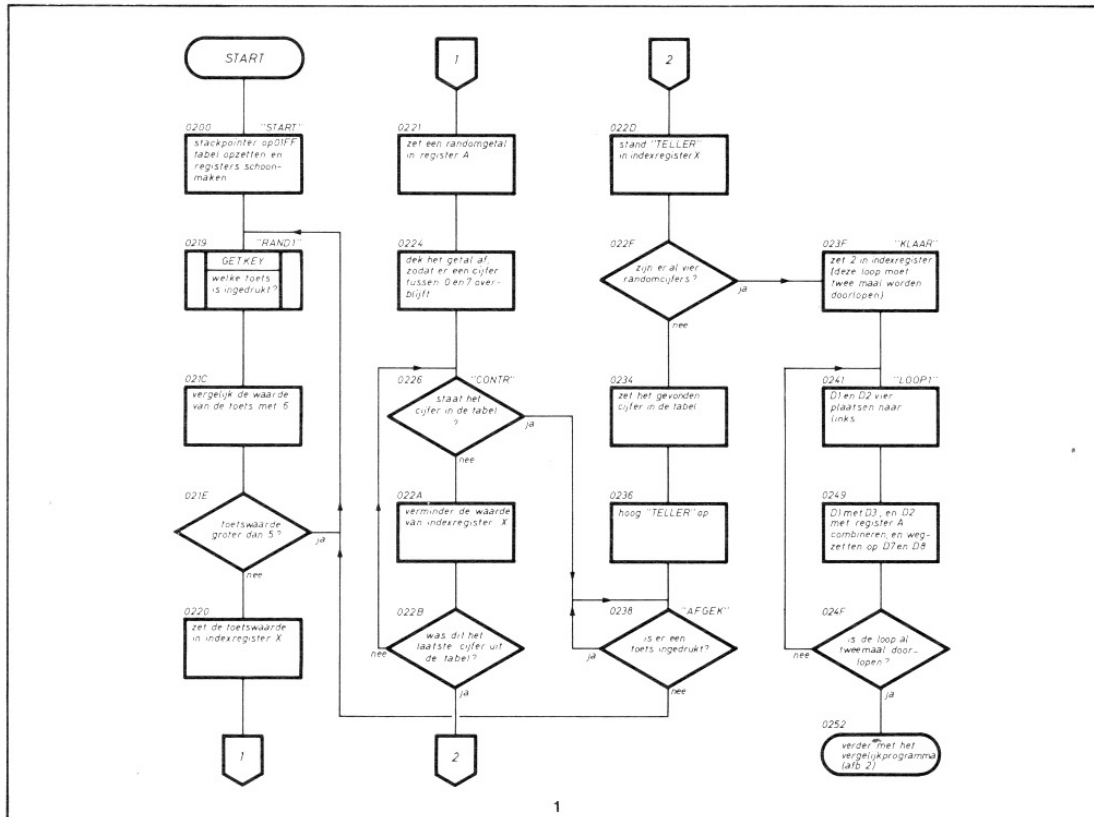
3331 geprobeerde code

KIM zal op deze situatie reageren met $2 \times$ de letter C (2 witte pennetjes). Er zit in de geprobeerde code immers de 3 (op de verkeerde plaats) en ook de 1 is een goed cijfer. Er bestaat echter ook een systeem dat zegt dat de 3 driemaal goed is geraden, maar te boven de 1 éénmaal goed is. Totaal eenmaal goed en dus 4 witte pennetjes komen.

Bent u dit systeem gewend? De KIM past zich aan uw wensen aan, indien u de **code EA op adres 036** zet. (Normaal staat op dit adres code E4.) Een derde systeem zegt: Het cijfer 1 komt $2 \times$ in de geheime code voor, de 3 komt $1 \times$ in de geheime code voor, bij elkaar dus 3 witte pennetjes. **Code EA op adres 032E** en de KIM denkt net als u. (Normaal staat hier E6.)

De werking

Voor degene die geïnteresseerd is in de werking en opbouw van het programma volgt hier een beschrijving van het Mastermind programma. Het programma valt uiteen in 2 belangrijke gedeeltes, nl. de randomgenerator (afb. 1) die zorgt dat de KIM een geheime code kan kiezen zonder voorkeur voor bepaalde combinaties, en het vergelijk programma (afb. 2). Dit laatste programmadeel verzorgt de vergelijking van de geprobeerde code met de geheime code. We zullen deze twee delen apart behandelen.



1 De flowchart van de randomgenerator

De randomgenerator (principe)

Het principe van de randomgenerator is heel eenvoudig. Op het moment dat een toets wordt ingedrukt kijken we op welke stand de timer van de KIM staat. (Als u adres 1706 invoert dan ziet u de timer lopen.) Doordat het moment van drukken onafhankelijk is van de stand van deze teller, zal de verkregen waarde volkomen willekeurig zijn. Bij de randomgenerator voor dit spelletje worden aan de verkregen getallen wel wat eisen gesteld. Het verkregen cijfer moet immers, afhankelijk van de ingedrukte toets, tussen de 0 en 6 of 1 en 5 liggen. Omdat al deze getallen niet meer dan 3 bits nodig hebben, kunnen we in ieder geval de eerste 5 bits van het verkregen random getal '0' maken. Hierdoor blijft een cijfer tussen de 0 en 7 (00000000 ... 00000111) over. In het geheugen wordt nu een tabel van verboden cijfers geschreven en het verkregen cijfer wordt met de in de tabel voorkomende cijfers vergeleken. Wanneer een gelijkheid wordt gevonden, wordt het verkregen cijfer afgekeurd, en zal er een keer extra op een toets moeten worden gedrukt. De tabel ziet er als volgt uit:

Adres	Inhouds
00D5	06
00D4	00

00D3	FF
00D2	FF
00D1	FF
00D0	07

Indien we bij het kiezen van een randomcijfer op de '0' drukken zal het verkregen cijfer alleen worden vergeleken met de inhoud van adres 00D0, zodat alleen de 7 uitgesloten is. Het verkregen cijfer zal dus tussen 0 en 6 liggen. Als we bij het kiezen van een randomcijfer op de '3' drukken, zal het verkregen cijfer in een loop vergeleken worden met de inhoud van de adressen 00D3 ... 00D0.

Is het eerste cijfer gekozen, stel dat dit een 3 is, dan wordt dit cijfer vergeleken met FF, FF, FF en 07. Omdat geen gelijkheid wordt gevonden, wordt het getal goedgekeurd en op adres 00D1 onthouden. Het getal wordt in de tabel van verboden cijfers geschreven, en de 3 kan geen tweede keer meer voorkomen. Stel dat bij de tweede druk op de '3' een 1 gevonden wordt, deze 1 wordt nu vergeleken met FF, 03 en 07. Ook de 1 wordt goedgekeurd en op adres 00D2 onthouden. Als we bij de derde druk op de '3' weer een 1 krijgen, wordt hij afgekeurd. De 1 wordt nu immers vergeleken met FF, 01, 03 en 07, zodat een gelijkheid wordt geconstateerd. Indien een goed derde cijfer is gevonden wordt dit op adres 00D3 gezet. Het 4de cijfer wordt nog wel vergeleken met de tabel, maar omdat er geen 5de cijfer meer komt is het niet nodig dit cijfer in de tabel te zetten.

Nadat we inplaats van op de '3' op de '5' drukken, dan worden de verkregen getallen ook nog vergeleken met '0' en '6', zodat ook deze cijfers uitgesloten zouden zijn. Het zal duidelijk zijn dat we voor het krijgen van een geheime code minstens 4 × op de 0, 1, 2, 3, 4 of 5 moeten drukken. Voor elk getal dat afgekeurd wordt moeten we nog een keer extra drukken. De schakeling laat niet zien wanneer een getal afgekeurd is. Wel laat zij zien wanneer de geheime code compleet is. De microprocessor verlaat dan het randomprogramma en begint aan het vergelijkprogramma. Het display licht op, met de beurtenteller op 1, ten teken dat u uw eerste code kunt invoeren.

De flowchart van het randomprogramma

De flowchart van het randomprogramma zien we in afb. 1. Op adres 200 beginnen we met het goed zetten van verschillende tellers. Ook de tabel op de plaatsen 00D0 t/m 00D5 wordt door dit deel gezet. De volgende stap is het bepalen welke toets is ingedrukt (adres 0219). Is er geen toets ingedrukt dan krijgt register A de waarde \$15 (\$ betekent weer: hexadecimaal). We zien dat de microprocessor deze zelfde loop doorloopt tot een van de getallen 0 ... 5 worden ingedrukt. De waarde van de toets wordt in indexregister X gezet (adres 0220). Vervolgens wordt de stand van de timer in register A gezet, en met 00000111 afgedekt, zodat een getal tussen 0 en 7 overblijft. Op adres 0226 begint de controleloop. Afhankelijk van de inhoud van indexregister X (dus waarvan de ingedrukte toets) wordt het verkregen cijfer vergeleken met een groot of klein deel van de tabel. Als een gelijkheid wordt gevonden, springt de microprocessor naar 'AFGEK' adres 0238.

Hier wordt gewacht tot de toets is losgelaten. Zodra dit het geval is wordt er naar het begin van het programma gesprongen. Indien in de controleloop (0226) geen gelijkheid wordt gevonden, kijkt de microprocessor naar de teller (adres 00D6) om te zien of we al 4 randomcijfers hebben. Is dit niet het geval, dan wordt het gevonden cijfer op de juiste plaats in de tabel gezet, de teller wordt opgehoogd en zodra de toets wordt losgelaten beginnen we weer bij 'rand 1' om het volgende randomcijfer te vinden. Zodra we 4 cijfers hebben gaat de microprocessor op adres 0241 de cijfers combineren. Op dit moment staan de eerste drie cijfers op de adressen 00D1 ... 00D3, en het 4de cijfer staat in register A. Om te laten zien hoe het combineren in zijn werk gaat zullen we als voorbeeld aannemen dat op de plaatsen 00D1 ... 00D3 de getallen 01 ... 03 staan en in register A 04. Op adres 023F (programma, lijst 1) wordt in de register X de waarde 02 gezet. Op de adressen 0241 ... 0247 wordt de inhoud van adres 00D + X = 00D2 4 plaatsen naar links geschoven. Het getal 02 is nu 20 geworden. Op adres 0249 wordt de inhoud van diezelfde geheugenplaats opgeteld bij de inhoud van register A. We krijgen dus $4 + 20 = 24$. Het resultaat 24 wordt nu op adres $00D6 + X = 00D8$ gezet. In register A komt nu de inhoud van adres 00D3 te staan (03). Indexregister X wordt met 1 verminderd (dit wordt 01) en de loop wordt voor de tweede maal doorlopen. Nu wordt de inhoud van adres 00D1 geshift, en bij de inhoud van register A opgeteld. Het resultaat wordt 13. Deze waarde komt op adres 00D7 te staan. Indexregister X wordt weer verminderd, met als resultaat 00. Hierdoor wordt niet gesprongen naar 'loop 1', en de microprocessor begint aan het vergelijkprogramma.

Vergelijkprogramma (principe)

Het vergelijkprogramma valt ook weer uiteen in verschillende delen. Allereerst is er het invoerdeel. Met dit stukje programma is het mogelijk uw code in te voeren. Alleen getallen tussen 0 en 6 kunnen worden ingevoerd. Steeds als een nieuw cijfer wordt ingevoerd, schuiven de reeds aanwezige cijfers op het display een plaats naar links (net als bij het invoeren van een adres). Om dit te bereiken moet het gehele getal dat bestaat uit 2×8 bits (4 hexadecimale cijfers, 1 hexadecimaal cijfer bestaat uit 4 bits, 1 cijfer naar links komt dus overeen met 4 bits naar links) geschoven worden. Dit gebeurt in de subroutine 'SHIFT'. Het is alleen mogelijk het invoerdeel te verlaten door een druk op de 'C' of de 'E'. Met een druk op 'C' wordt de geheime code op POINTO en POINTH gezet, waardoor ze later in het programma zichtbaar worden op het display. Tevens wordt in register A het getal 29 gezet, en naar de ophoogsubroutine gesprongen. Deze subroutine zet de decimal mode flag waardoor er decimaal wordt gerekend. Ook de carry wordt gezet, zodat het uiteindelijke resultaat aan de beurt teller met 30 wordt opgehoogd. Wanneer we van de subroutine terugkeren, beginnen we weer met invoer, zodat we weer nieuwe codes kunnen invoeren. Als we op de 'E' drukken, verlaten we het invoerdeel, en beginnen we het vergelijkprogramma dat de A's en C's op het display zet. Dit programma is gebaseerd op de subroutine 'VERG'. Deze subroutine doet niets anders dan het verticaal vergelijken van de getallen. Bij een geconstateerde gelijkheid wordt een A of C

in het display geschoven met behulp van de eerder besproken subroutine 'SHIFT'. We zullen dit aan de hand van een voorbeeld duidelijk maken. Stel, we hebben:

4341 geheime code
4313 geprobeerde code.

We zetten nu 0A op adres 00E2 en springen naar de subroutine 'VERG'. Deze subroutine constateert 2 gelijkheden, nl. in de 1e cijfers en in de 2e cijfers (de 4 en de 3). Hierdoor zal ook 2 x de inhoud van adres 00E2 in het display geschoven worden (er staan dus nu twee A's). Om te voorkomen dat de gelijkheden nog een keer herkend worden moeten de getallen worden veranderd, zodanig dat zij gegarandeerd anders zijn dan alle voorkomende cijfers. Bij de cijfers uit de geheime code tellen we 8 op, zodat een cijfer tussen de 8 en E (= 14) ontstaat. De cijfers uit de geprobeerde code worden vervangen door 'F' (= 15). Als we terug komen uit de subroutine zien de codes er als volgt uit:

CB41 geheime code
FF13 geprobeerde code.

In de volgende stap gaan we kijken of er 'schuine' gelijkheden zijn (zoals de '1'). Allereerst zetten we nu 0C op adres 00E2, en we laten de geprobeerde code 1 plaats roteren. De codes zien er nu zo uit:

CB41 geheime code
F13F geprobeerde code.

We springen weer naar dezelfde subroutine, en er wordt geen gelijkheid geconstateerd. Het proces wordt nog een keer herhaald. Na de rotatie krijgen we:

CB41 geheime code
13FF geprobeerde code.

Ook hier wordt geen gelijkheid geconstateerd. Hadden we bij de eerste vergelijking echter niets afgedekt, dan zou er hebben gestaan:

4341 geheime code
1343 geprobeerde code.

Met het gevolg dat 2 gelijkheden geconstateerd worden en dat er ten onrechte 2 C's op het display zouden verschijnen. Hieruit blijkt dus de noodzaak van het afdekken. De laatste keer roteren geeft:

CB41 geheime code
3FF1 geprobeerde code.

Nu wordt het 4e cijfer als gelijk herkend (de '1'), en de inhoud van adres 00E2 (nu 'C') wordt in het display geschoven. Nog een keer roteren heeft geen zin meer, omdat we dan weer in de uitgangspositie komen. Het eindresultaat op het display is dus '0AAC'.

[illegible]

De flowchart van het vergelijkprogramma (afb. 2)

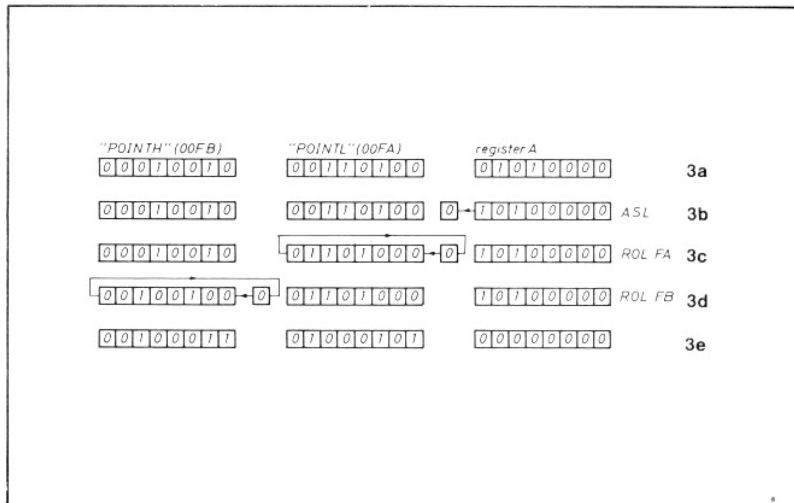
Het eerste deel van de flowchart spreekt voor zichzelf. Op adres 025B springt de microprocessor naar subroutine 'DISPL', en hij blijft daar totdat een toets is ingedrukt. Alle indrukken tussen 0 en 6 hebben tot gevolg dat de microprocessor via adres 025E en 0260 bij de subroutine 'SHIFT' komt, waar het naar links schuiven van de cijfers op het display verzorgd wordt, terwijl op het meest rechtse display het nieuwe cijfer wordt

gezet. Als de subroutine weer verlaten wordt is altijd het Z-bit uit het status register 1'. Hiervan maken we gebruik op adres 263, waar we via een BEQ (spring als Z = 1) onvoorwaardelijk terug kunnen springen naar loop 2 (025B). Bij alle ingedrukte toetsen groter dan 6 en ongelijk aan E of C gaat de microprocessor via 0265 en 0269 weer terug naar 'loop 2', en wordt er verder geen actie ondernomen. Wordt een 'E' ingedrukt, dan verlaat de microprocessor het invoerdeel en begint op adres 027A aan het vergelijkprogramma. Om te beginnen met de geraden code, die nu in het displaygeheugen staat, in andere geheugenplaatsen worden gezet. Het displaygeheugen (POINTLO en POINTHI) moet straks immers worden gebruikt om de A's en de C's in te bergen. Op adres 0282 wordt het displaygeheugen met nullen gevuld, zodat later op de plaatsen waar geen A of C staat een '0' komt te staan. Nu begint het vergelijken. Eerst zetten we een A op 'shift' (adres 00E2) omdat we beginnen met het vergelijken van getallen die op de goede plaats staan. Op adres 028C springt de microprocessor naar de subroutine 'VERG', en wanneer hij hiervan terugkeert staat er voor alle cijfers die op de goede plaats stonden een A in het displaygeheugen. Bovendien zijn de gelijkheden vervangen door andere cijfers, zoals besproken bij het principe van het vergelijkprogramma.

Als er geen 'AAA' in het displaygeheugen staat, springt de microprocessor naar 'C' (adres 02B0), waar we beginnen met 03 op adres 00E1 te zetten. Deze geheugenplaats dient als teller, en houdt bij hoeveel keer de geraden code is geroteerd. Dit roteren begint op adres 02B8. Op adres 02C4 worden de codes dan weer vergeleken. Vervolgens wordt de teller met 1 verminderd. Dit hele proces wordt 3 x herhaald, en bij elke geconstateerde gelijkheid wordt er een 'C' in het display geschoven. Het principe van dit roteren is gelijk aan het principe van de subroutine 'SHIFT', wel nog besproken zal worden wanneer de vergelijkloop doorlopen, springen we weer naar subroutine 'DISPL'.

Hier worden de A's en de C's eigenlijk pas zichtbaar gemaakt. De microprocessor blijft in deze subroutine totdat een toets wordt ingedrukt. Op dat moment wordt het vergelijkgedeelte opnieuw gestart (adres 0252). Het display wordt op '0000' gezet, de beurt teller opgehoogd en de geheime code wordt opnieuw op de adressen 00E5 en 00E6 gezet, omdat de code in de vorige vergelijkloop is verminkt. De cijfers worden immers veranderd om te voorkomen dat dubbele gelijkheden worden gevonden, zie principe vergelijkprogramma. Op adres 025B ('loop 2') beginnen we dan weer met het invoerdeel, en de nieuwe code kan worden ingevoerd. Als de code goed is geraden, zullen op adres 028F vier A's op het display worden herkend. Hierdoor zal de microprocessor op adres 0299 beginnen aan een loop waarin het display brandt en op adres 02A0 een loop waarin het display uit is. Dit geheel wordt 6 x herhaald, zodat het display 6 x aan en uit gaat. Het principe van loops zal duidelijk zijn, geheugenplaats 00D4 wordt gebruikt als teller. Deze teller begint op 00 en telt naar beneden via FF weer naar 00. Omdat deze teller steeds met één wordt opgehoogd, hoeft er aan het begin van een loop niet steeds '00' geschreven te worden. Teller 00D5 houdt het aantal

knipperingen bij. Wanneer het display voor de laatste keer uitgaat, springen we weer naar het begin van het programma, en kan er weer een nieuwe geheime code gemaakt worden.



3a – De toestanden van de registers nadat register A 4 × naar links geschoven is De inhoud van het displaygeheugen is '1234'.

3b – De instructie ASL ASL, bit 7 schuift in het Carry-bit.

3c – De instructie ROL FA welke in het Carry-bit stond schuift in bit 0 van 'POINTL', terwijl bit 7 van 'POINTL' weer in het Carry-bit schuift.

3d – De instructie ROL FB Bit 7 van 'POINTL' komt nu in bit 0 van 'POINTH'.

3e – De toestand na 4 × roteren De inhoud van het displaygeheugen is nu '2345'

De subroutines

Het programma maakt gebruik van 4 subroutines, nl. 'INCR', deze hoogt de beurt teller op, 'DISPL', deze activeert het display en bepaalt welke toets is ingedrukt, 'SHIFT' zorgt dat een cijfer welke in register A staat, in het display wordt geschoven, en tot slot 'VERG', welke de codes vergelijkt. We zullen de subroutines nu afzonderlijk behandelen.

INCR (adres 02D1)

Omdat de instructie 'INC' niet decimaal werkt, moeten we voor het ophogen van de beurt teller gebruik maken van de optel instructie 'ADC'. Omdat de beurt teller decimaal moet worden opgehoogd, moet eerst de 'decimal mode flag' gezet worden. Ook de carry wordt gezet. Het eindresultaat op de beurt teller wordt dus: inhoud register A + oude inhoud beurt teller + 1. Wanneer we de teller met 1 willen ophogen, moet in register A dus de waarde 00 staan. Deze subroutine zorgt tevens voor het kopiëren van de geheime code naar de plaatsen 00E5 en 00E6. Het zal de kritische lezer opgevallen zijn dat deze subroutine slechts éénmaal wordt aangeroepen. In feite is het dus helemaal niet nodig dit stukje programma als subroutine uit te voeren. Bij een uitbreiding van het

programma, door een inventieve geest, kan echter handig gebruik gemaakt worden van deze subroutine door b.v. het toekennen van strafpunten.

DISPL (adres 02E2)

Deze subroutine zorgt er steeds voor dat de cijfers, welke in het displaygeheugen staan (POINTLO, POINTHI), op het display zichtbaar worden. Tevens wordt bepaald of er een toets is ingedrukt, en zo ja welke. Voor het zichtbaar maken van de gegevens maakt deze subroutine gebruik van een subroutine in het monitorprogramma (SCANDS). Als de microprocessor terug komt van 'SCANDS' heeft de register A de waarde '00' zolang er geen toets is ingedrukt. De eerste loop loopt van adres 02E2 ... 02E6. In deze loop wordt gewacht totdat er geen toets meer ingedrukt is. Dit is nodig omdat de vorige toets meestal nog ingedrukt is. Was er geen toets ingedrukt, of de vorige toets wordt losgelaten, dan komt de microprocessor in de 2e loop (02E7 ... 02EB). Hier wordt gewacht totdat er een toets wordt ingedrukt. Zodra dit het geval is wordt op de adressen 02EC ... 02F0 nog een keer gecontroleerd of er echt een toets ingedrukt was. Het is nl. ook mogelijk dat de loop verlaten wordt door een stoortje. Op adres 02F1 wordt bepaald welke toets is ingedrukt. Tot slot wordt de toets met \$07 vergeleken, zodat de N-bit in het statusregister wordt gezet voor de toetsen tussen 0 en 6. Later kunnen hier de voorwaardelijke sprongen mee gemaakt worden.

SHIFT (adres 02F7)

De werking van deze subroutine zullen we aan de hand van een voorbeeld uitleggen. Stel dat op het display staat: 1234, en in register A de waarde 05. Allereerst wordt op de adressen 02F7 ... 02FA register A 4 maal naar links geschoven. Het resultaat wordt: 50. In afb. 3 zien we nu wat er achtereenvolgens gebeurt. In afb. 3a laat de situatie zien zoals hij is nadat register A 4 x geschoven is. Op adres 02FD wordt register A nog een keer geschoven (afb. 3b). Het bit dat uit register A schuift komt in de 'carry' bit van het statusregister te staan. Bij de eerste ROL-instructie (02FE en afb. 3c) wordt dit bit in geheugenplaats 'POINTLO' geschoven, terwijl het bit dat hier naar buiten wordt geschoven, weer in de carry komt te staan. Bij de ROL-instructie op adres 0300 (afb. 3d) wordt dan dit bit weer in 'POINTHI' geschoven. Op deze manier is het geheel 1 plaats naar links geschoven. Dit proces wordt in een loop via een maal herhaald, zodat aan het eind van deze subroutine alle bits 4 plaatsen zijn opgeschoven (afb. 3e).

VERG (adres 0306,

In 'VERG' worden 2 codes met elkaar vergeleken. De subroutine is te verdelen in 3 belangrijke gedeelten, nl. het frame, op de adressen 0306 ... 0314, het vergelijkdeel op de adressen 0315 ... 0328 en het afdekdeel op de adressen 0329 ... 033D. In het vergelijkdeel op de adressen 0315 ... 0328 en het afdekdeel op de adressen 0329 ... 033D. In het vergelijkdeel wordt steeds één van de twee geheugenplaatsen van de geheime code vergeleken met de overeenkomstige geheugenplaats van de geraden code. Omdat er op één geheugenplaats 2 cijfers staan, moet er steeds 1 cijfer worden

afgedekt. We moeten immers maar 1 cijfer tegelijk vergelijken. In het framedel wordt eerst op adres 00E0 de waarde \$F0 gezet. Het vergelijkdeel gebruikt deze waarde later om via de 'AND'-instructie het rechtse cijfer 0 te maken, zodat het linker cijfer vergeleken kan worden. Op adres 030A springt de microprocessor voor de eerste keer naar het vergelijkdeel op adres 0315. Omdat dit deel 2 x doorlopen moet worden zetten we eerst in register X de waarde 02. Vervolgens wordt een deel van de geheime code in register A gezet (de inhoud van adres 00E6), afgedekt met F0 (de inhoud van adres 00E0). Het gevolg is dat van de totaal 4 cijfers er nu één in register A staat. Dit cijfer wordt zolang op adres 00E0 gezet. We zijn inmiddels beland op adres 031D van het programma. Hier wordt een overeenkomstig deel van de geraden code in register A gezet en afgedekt (00E8). Op adres 0321 worden de cijfers vergeleken. Hierna wordt het vergelijkdeel nog een keer doorlopen. Nu wordt echter het linker cijfer van de geheugenplaats 00E5 en 00E7 vergeleken.

Als dit gebeurd is, springen we terug naar het framedeel (030D). Nu zetten we geen \$F0 maar \$0F op adres 00E0, en springen weer naar het vergelijkdeel. Het hele proces wordt herhaald, maar nu voor het rechter cijfer. Wanneer een gelijkheid wordt gevonden springen we naar het afdekdeel op adres 0329. Hier wordt eerst het betreffende cijfer van de geheime code afgedekt. Op adres 032F en 0331 wordt in register A de waarde \$80 of \$08 gezet. Met de 'ORA'-instructie wordt deze 8 dan opgeteld bij het desbetreffende cijfer van de geraden code. Aan het eind van dit deel wordt de inhoud van adres 00E2 ('A' of 'C') met behulp van subroutine 'SHIFT' op het display gezet.

UITBREIDINGEN

Met dit programma kunt u een aardig spelletje spelen. Maar probeer ook eens een kleine dingen aan het programma te veranderen. U kunt b.v. eens proberen i.p.v. A en C de E en F te gebruiken als codepinnetjes. U kunt de beurten teller zo maken dat hij niet verder telt dan 99, zodat er niet gesmokkeld kan worden met de C-toets (3 x drukken op deze toets en de beurten teller wijst 10 punten minder aan).

Voor de gevorderde knutselaar: Probeer eens voor elkaar te krijgen dat na elke 30 sec. bedenktijd de beurten teller automatisch opgehoogd wordt, dat brengt wat spanning in het spel. U kunt ook nog iets maken dat automatisch de score van verschillende spelers bijhoudt. Het voordeel van de programmeer-hobby is dat een instructie niets kost, en dat u geen transistortjes op kunt blazen met verkeerde instructies.

Lijst 1

0200	D8	START	CLD-impl		D nul maken.	022B	10 F9	BPL-rel	CONTR	} Laatste vergelijking?
0201	EA		NOP-impl			022D	A6 D6	LDX-Z page	TELLER	
0202	A2 FF		LDX-imm	SFF	} Stackpointer op 01FF.	022F	EO 03	CPX-imm	S03	} Indien er 4 getallen zijn naar 'KLAAR'.
0204	9A		TXS-impl			0231	FO OC	BEQ-rel	KLAAR	
0205	86 D1		STX-Z page	TABEL+1	} Tabel opzetten.	0233	EA	NOP-impl	TABEL+1	} Zet het gevonden cijfer in de tabel. Hoog de teller op. Wacht tot de toets is losgelaten. Naar 'RAND1'. Begin combineerloop.
0207	86 D2		STX-Z page	TABEL+2		0234	95 D1	STA-Zp, X	TELLER	
0209	86 D3		STX-Z page	TABEL+3		0236	E6 D6	INC-Z page	AK	
020B	E8		INX-impl			0238	20 FE 1E	JSR-abs	AFGEK	
020C	86 D4		STX-Z page	TABEL+4		023B	DO FB	BNE-rel	RAND1	
020E	A0 07		LDY-imm	S07	} 'TELLER' en de beurtenteller schoonmaken. Welke toets?	023D	FO DA	BEQ-rel	S02	} Cijfers 4 plaatsen naar links.
0210	84 D0		STY-Z page	TABEL		023F	A2 02	LDX-imm	TABEL	
0212	88		DEY-impl		} Indien groter dan 5 naar 'RAND1'.	0241	16 DO	ASL-Zp, X	TABEL	} Cijfers combineren. Combinatie wegzetten.
0213	84 D5		STY-Z page	TABEL+5		0243	16 DO	ASL-Zp, X	TABEL	
0215	86 D6		STX-Z page	TELLER	} Toets naar X. Randomcijfer in A. Cijfer afdekken.	0245	16 DO	ASL-Zp, X	TABEL	} Eind combineerloop.
0217	86 F9		STX-Z page	INH		0247	16 DO	ASL-Zp, X	TABEL	
0219	20 6A 1F	RAND 1	JSR-abs	GETKEY	} Vergelijk het gevonden cijfer met de tabel.	0249	75 DO	ADC-Zp, X	TABEL	
021C	C9 06		CMP-imm	S06		024B	95 D6	STA-Zp, X	TELLER	
021E	10 F9		BPL-rel	RAND1		024D	A5 D3	LDA-Z page	TABEL+3	
0220	AA		TAX-impl			024F	CA	DEX-impl	LOOP1	
0221	AD 06 17		LDA-abs	TIMER		0250	DO EF	BNE-rel		
0224	29 07	CONTR	AND-imm	S07						
0226	D5 D0		CMP-Zp, X							
0228	F0 OE		BEQ-rel	AFGEK						
022A	CA		DEX-impl							

Lijst 2

0252	A9 00	KIES	LDA-imm	\$00	Display
0254	85 FA		STA-Z page	POINTL	schoonmaken.
0256	85 FB		STA-Z page	POINTH	
0258	20 D1 02	LOOP3	JSR-abs	INCR	Beurten teller ophogen.
025B	20 E2 02	LOOP2	JSR-abs	DISPL	Welke toets?
025E	10 05		BPL-rel	VERDER	Indien groter dan 6 naar 'VERDER'.
0260	20 F7 02		JSR-abs	SHIFT	Schuif het display door en spring terug.
0263	F0 F6		BEQ-rel	LOOP2	
0265	C9 0E	VERDER	CMP-imm	\$0E	Indien 'E' is ingedrukt naar 'SET' springen.
0267	F0 11		BEQ-rel	SET	Indien 'C' niet is ingedrukt, naar 'LOOP2'.
0269	C9 0C		CMP-imm	\$0C	
026B	D0 EE		BNE-rel	LOOP2	
026D	A5 D7		LDA-Z page	GEHLO	Zet de geheime code op het display.
026F	85 FA		STA-Z page	POINTL	
0271	A5 D8		LDA-Z page	GEHHI	
0273	85 FB		STA-Z page	POINTH	
0275	A9 29		LDA-imm	\$29	Zet 29 in reg A, en ga via 'LOOP3' naar de subroutine 'INCR'.
0277	EA		NOP-impl		
0278	10 DE		BPL-rel	LOOP3	
027A	A5 FA	SET	LDA-Z page	POINTL	
027C	85 E7		STA-Z page	GEPRLO	Het geprobeerde getal wegzetten.
027E	A5 F0		LDA-Z page	POINTH	
0280	85 E8		STA-Z page	GEPRHI	
0282	A9 00		LDA-imm	\$00	
0284	85 FA		STA-Z page	POINTL	Maak het display schoon.
0286	85 FB		STA-Z page	POINTH	
0288	A9 0A		LDA-imm	\$0A	'A' in 'SCHUIF', zodat er A's op het display komen, en spring naar 'VERG'.
028A	85 E2		STA-Z page	SCHUIF	
028C	20 06 03		JSR-abs	VERG	
028F	A9 AA		LDA-imm	\$AA	
0291	C5 FA		CMP-Z page	POINTL	Staan er 4 A's op het display?
0293	D0 1B		BNE-rel	C	Zo nee, spring naar 'C'.
0295	C5 FB		CMP-Z page	POINTH	
0297	D0 17		BNE-rel	C	
0299	20 1F 1F	AAN	JSR-abs	SCANDS	
029C	C6 D4		DEC-Z page	TABEL+4	Display aan.
029E	D0 F9		BNE-rel	AAN	
02A0	A0 00	UIT-LOOP4	LDY-imm	\$00	
02A2	88		DEY-impl		
02A3	D0 FD		BNE-rel	LOOP4	Display uit.
02A5	C6 D4		DEC-Z page	TABEL+4	
02A7	D0 F7		BNE-rel	UIT	
02A9	C6 D5		DEC-Z page	TABEL+5	Laatste keer geknipperd?
02AB	D0 EC		BNE-rel	AAN	Zo nee, naar 'AAN'.
02AD	4C 00 02		JMP-abs	START	Terug naar 'START'.
02B0	A9 03	C	LDA-imm	\$03	'TEL2' houdt bij hoeveel maal geroteerd wordt.
02B2	85 E1		STA-Z page	TEL2	
02B4	A9 0C		LDA-imm	\$0C	'C' in 'SCHUIF', zodat er C's op het display komen.
02B6	85 E2		STA-Z page	SCHUIF	Vier maal schuiven.
02B8	A0 04	LOOP6	LDY-imm	\$04	
02BA	A5 E8	LOOP5	LDA-Z page	GEPRHI	
02BC	0A		ASL-accu		
02BD	26 E7		ROL-Z page	GEPRLO	Alle cijfers in de geprobeerde code 1 plaats naar links.
02BF	26 E8		ROL-Z page	GEPRHI	
02C1	88		DEY-impl		
02C2	D0 F6		BNE-rel	LOOP5	
02C4	20 06 03		JSR-abs	VERG	Schuif zonnig een C op het display.
02C7	C6 E1		DEC-Z page	TEL2	Laatste maal geroteerd?
02C9	D0 ED		BNE-rel	LOOP6	
02CB	20 E2 02		JSR-abs	DISPL	Wacht op toetsin-druk en spring terug.
02CE	4C 52 02		JMP-abs	KIES	

Lijst 3

02D1	F8	INCR	SED-impl		
02D2	38		SEC-impl		
02D3	65 F9		ADC-Z page	INH	Tel decimaal de inhoud van reg. A op bij 'INH'.
02D5	85 F9		STA-Z page	INH	
02D7	D8		CLD-impl		
02D8	EA		NOP-impl		
02D9	A5 D7		LDA-Z page	GEHLO	
02DB	85 E5		STA-Z page	COPLO	Copieer de geheime code.
02DD	A5 D8		LDA-Z page	GEHHI	
02DF	85 E6		STA-Z page	COPHI	
02E1	60		RTS-impl		Terug.
02E2	20 1F 1F	DISPL	JSR-abs	SCANDS	Wacht tot de toets wordt losgelaten.
02E5	D0 FB		BNE-rel	DISPL	Wacht tot de toets wordt ingedrukt.
02E7	20 1F 1F	LOOP8	JSR-abs	SCANDS	Is de toets wel ingedrukt?
02EA	F0 FB		BEQ-rel	LOOP8	Welke toets is ingedrukt?
02EC	20 1F 1F		JSR-abs	SCANDS	Kleiner dan 7?
02EF	F0 F6		BEQ-rel	LOOP8	Terug.
02F1	20 6A 1F		JSR-abs	GETKEY	
02F4	C9 07		CMP-imm	\$07	
02F6	60		RTS-impl		
02F7	0A	SHIFT	ASL-accu		Cijfer 4 plaatsen naar links schuiven.
02F8	0A		ASL-accu		
02F9	0A		ASL-accu		
02FA	0A		ASL-accu		
02FB	A0 04		LDY-imm	\$04	Vier maal schuiven.
02FD	0A	LOOP7	ASL-accu		Afb. 3b
02FE	26 FA		ROL-Z page	POINTL	Afb. 3c
0300	26 FB		ROL-Z page	POINTH	Afb. 3d
0302	88		DEY-impl		
0303	D0 F8		BNE-rel	LOOP7	Indien nog geen 4 maal, naar 'LOOP7'.
0305	60		RTS-impl		Terug.
0306	A9 F0	VERG	LDA-imm	\$F0	Vergelijk het linker cijfer.
0308	85 E0		STA-Z page	MODE	
030A	20 15 03		JSR-abs	VERG2	
030D	A9 0F		LDA-imm	\$0F	Vergelijk het rechter cijfer.
030F	85 E0		STA-Z page	MODE	
0311	20 15 03		JSR-abs	VERG2	
0314	60		RTS-impl		Terug.
0315	A2 02	VERG2	LDX-imm	\$02	Wordt 2 maal doorlopen.
0317	B5 E4	LOOP9	LDA-Zp, X	COPIE	Haal twee cijfers v.d. geheime code, en zet er één op E3 of E4.
0319	25 E0		AND-Z page	MODE	
031B	95 E2		STA-Zp, X	SCHUIF	Haal het overeenkomstige cijfer van de geprobeerde code en vergelijk.
031D	B5 E6		LDA-Zp, X	COPHI	Indien gelijk naar 'AFDEK'.
031F	25 E0		AND-Z page	MODE	
0321	D5 E2		CMP-Zp, X	SCHUIF	
0323	F0 04		BEQ-rel	AFDEK	
0325	CA	VERG3	DEX-impl		Indien nog geen twee maal doorlopen naar 'LOOP9'.
0326	D0 EF		BNE-rel	LOOP9	Terug.
0328	60		RTS-impl		
0329	B5 E6	AFDEK	LDA-Zp, X	COPHI	Dek een cijfer uit de geheime code af met 'F'.
032B	05 E0		ORA-Z page	MODE	
032D	95 E6		STA-Zp, X	COPHI	
032F	A9 88		LDA-imm	S88	Afh. van 'MODE' komt er 08 of 80 in reg. A.
0321	25 E0		AND-Z page	MODE	
0333	15 E4		ORA-Zp, X	COPIE	Dek een cijfer uit de geprobeerde code af met 'B'.
0335	95 E4		STA-Zp, X	COPIE	Schuif een 'A' of een 'C' in het display.
0337	A5 E2		LDA-Z page	SCHUIF	Terug naar 'VERG3'.
0339	20 F7 02		JSR-abs	SHIFT	
033C	F0 E7		BEQ-rel	VERG3	