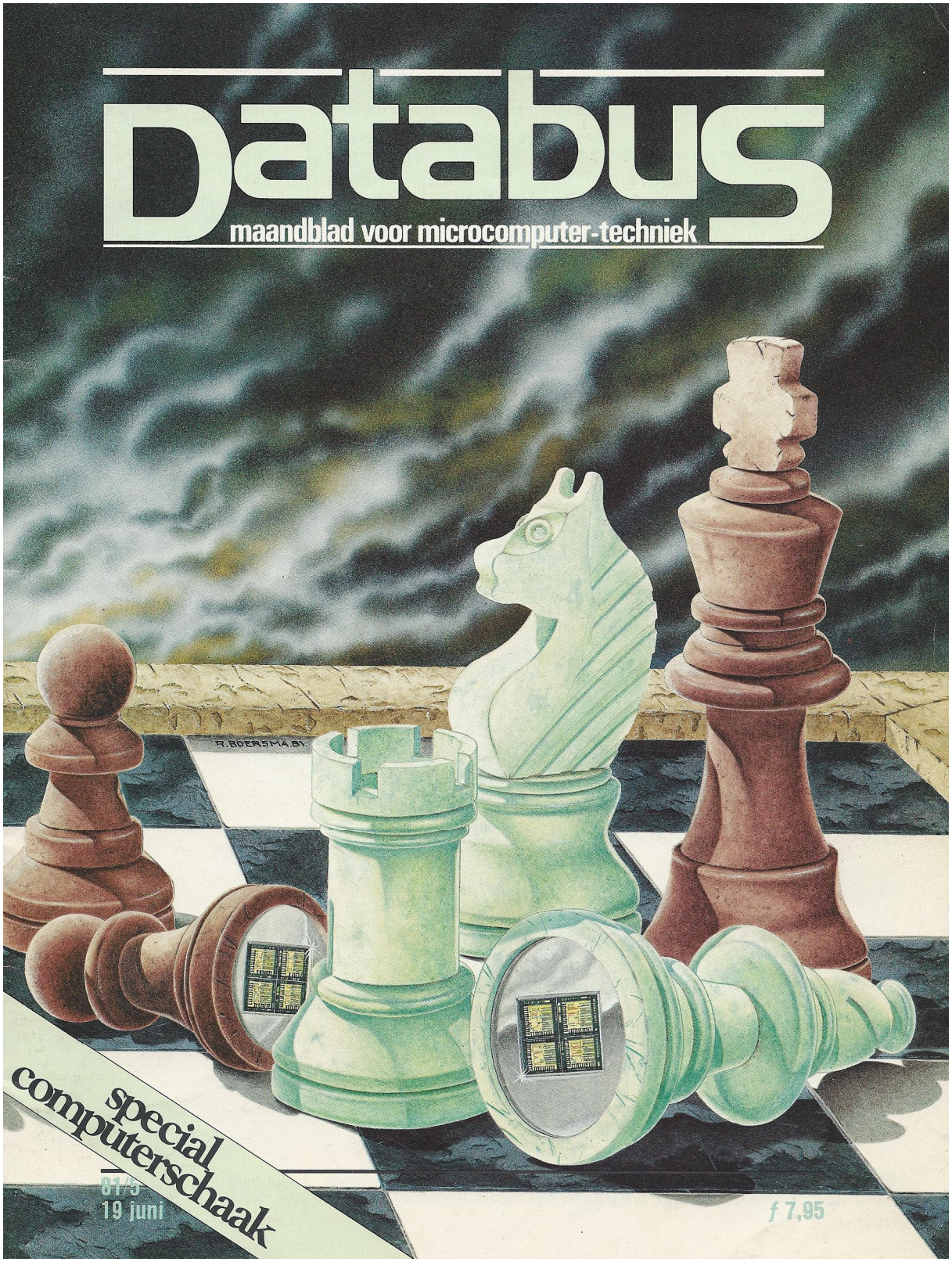

Databus (1981)

Jan Kuipers: Tiny Chess 86

Tiny Chess 86

Een schaakprogramma voor de 8088/8086

Jan Kuipers

Dit artikel beschrijft Tiny Chess 86 (in het vervolg TC86 genoemd), een schaakprogramma dat zal deelnemen aan het Nederlands kampioenschap voor schaakcomputers in september a.s. Het programma draait op een 8086- of 8088-microprocessor van Intel en is geschreven in de assembly taal hiervoor. Een speciale versie is verschenen in Elektuur (maart '81) onder de naam „Intelect".

Zoals de naam suggereert heeft TC86 zeer bescheiden afmetingen: 4 Kbyte programma (in bijv. EPROM) dat 1 Kbyte RAM nodig heeft. Er is zelfs een versie van 2 Kbyte met vrijwel gelijke speelkracht, maar waarin het openingenboek ontbreekt plus de mogelijkheid om snel speciale stellingen te creëren. Ondanks zijn afmetingen speelt TC86 een redelijke partij schaak, geheel volgens de internationale spelregels.

Historie

TC86 is geen professioneel programma, in die zin dat het tot nog toe steeds een lunch- en avondproject is geweest. Ik ben met het schaakprogramma begonnen in september '78 toen de eerste „samples" van de 8086 beschikbaar waren.
De 8086 werd gekozen omdat dit de eerste micro was (en beschikbaar) met een „performance" van enkele malen die van de traditionele 8-bitters. Voor een schaakprogramma als TC86 is die „performance" verhouding zelfs groter, omdat de belangrijkste variabelen in het grote aantal interne registers worden opgeslagen.

Een tweede aspect is de zeer compacte instructiecode die mogelijk is, een compactheid die op dit moment nog door geen andere micro kon worden, geëvenaard.
De allereerste routines werden met de hand gecodeerd in machinetaal, omdat de assembler nog niet beschikbaar was. Dit was een nogal lastig karwei, daar de gebruikte JUMP en CALL-coden relatief zijn. Toch bleek de zettengenerator, het hart van het programma, zonder veel problemen te werken.

Het testen van het programma is hoofdzakelijk op een SDK86 gebeurd. De ontwikkeling van TC86 is verder (met behulp van editor en assembler) als volgt verlopen:
december '78: Werkend algoritme voor de berekening van de beste zet op basis van stukken-verlies/winst.

SOFTWARE KRONKELS

januari'79: Toepassen van het α/β minimax „afkap" algoritme, waardoor de snelheid ongeveer een factor 20 toenam, afhankelijk van de zoekdiepte.
mei'79: Completering met „specials", achtereenvolgens: pionpromotie, „en-passant" slaan en rochade.
juli '79: Uitbreiding met een nette gebruikersinterface: weergave van zetten, test van legale zetten tegenstander, zwart of wit spel en niveau-instelling.
september '79: Meer „toeters en bellen": bordweergave op het beeldscherm, mogelijkheid tot opzetten van speciale stellingen, interrupt denkproces, random zettenkeuze.
november '79: Deelname aan het internationale PCW-toernooi in Londen met een 4 Kbyte versie voor de SDK86. Resultaat: gedeelde derde plaats uit 9 deelnemers.
juni'80: Toepassing van een strategische analyse en openingenboek.

Tussentijds zijn verschillende kleine snelheidsverbeteringen aangebracht, resulterend in zo'n 50% denktijdbeperking.
Ook ander onderzoek werd verricht zoals vooruitdenken in de tijd van de tegenstander en permanente bordweergave waarop veranderingen werden uitgevoerd door middel van cursorpositioning. De resultaten hiervan zijn echter nog niet in een produkt verwerkt.

Een van de meest frusterende ontwikkelingen was wel die van de „specials", in het bijzonder de rochade. Er deden zich „bugs" voor, bijvoorbeeld na 2 minuten denktijd werd het geheugen compleet gewist. Na veel geploeter bleek dan dat er iets fout zat wanneer een pion kon promoveren terwijl deze de vijandige koning kon slaan. Dit is slechts één voorbeeld....

Zoals het er nu voorstaat

Het TC86 systeem zoals het zal verschij-

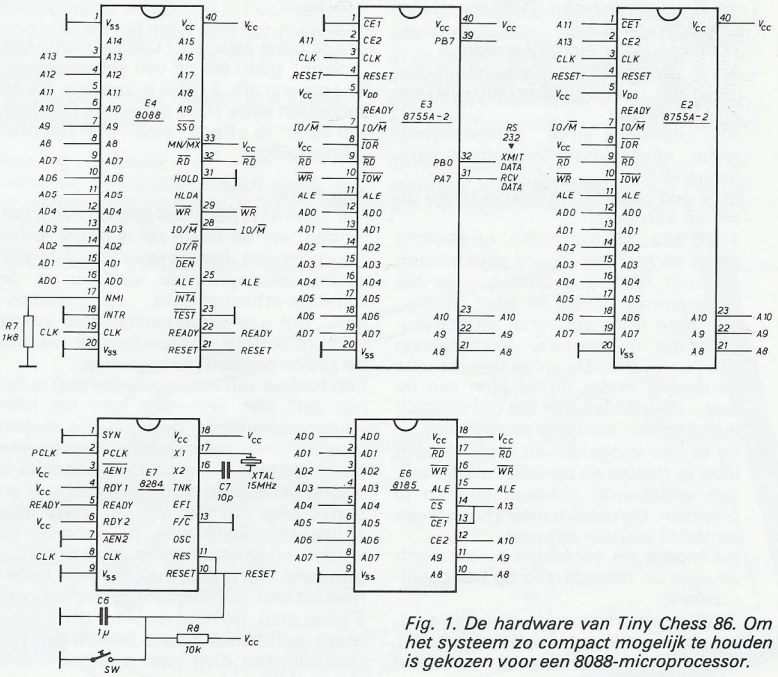

Fig. 1. De hardware van Tiny Chess 86. Om het systeem zo compact mogelijk te houden is gekozen voor een 8088-microprocessor.

nen op het Nederlandse kampioenschap bestaat uit een printplaatje van 10 x 10 cm met de volgende componenten: 8088 (processor), 8284A (klokgenerator) 8185 (1 Kbyte RAM) en 2x 8755A 2 (elk 2 Kbyte EPROM + I/O)
Er is hier voor de 8088 gekozen om het systeem zo compact mogelijk te houden. De 8088 draait op 5 MHz zonder waitstates. Er is ongeveer 10% snelheidsverlies t.o.v. een gelijkwaardig 8086-systeem.
De schakeling heeft slechts een 5 V-voeding nodig en een terminal (toetsenbord + beeldscherm) met een standaard RS232-aansluiting. (Er bestaat ook een versie voor een kleine druktoetsenbord met 7-segment display, met dezelfde schaakkwaliteiten, doch met minder mogelijkheden).
Het programma is 3 Kbyte groot terwijl de rest van de EPROM-ruimte (1 Kbyte) wordt ingenomen door het openingenboek.
Het programma biedt de volgende mogelijkheden:

- Op ieder moment – zelfs tijdens het denkproces – is de zoekdiepte instelbaar, vanaf enkele seconden tot meer dan een etmaal bedenktijd.
- Normaal gesproken wordt gecontroleerd of de ingetoetste zetten zijn toegelaten, echter er is een mogelijkheid illegale zetten uit te voeren om veranderingen aan te brengen op het bord. Men kan bijv. een stuk elimineren door er een leeg vakje naar toe te schuiven.
- Telkens als u aan zet bent kan TC86 uw kleur overnemen. TC86 speelt dus zowel zwart als wit.
- TC86 kan tegen zichzelf spelen.
- Als er verschillende goede zetten mogelijk zijn, maakt TC86 er een willekeurige keuze uit.
- Alle ,,operator input'' is beschermd, zodat alleen zinvolle commando's mogelijk zijn.
- Er is een ,,edit'' mogelijkheid voor de invoer van zetten .
- TC86 kent pionpromotie, en-passant slaan en rochade, zowel voor zichzelf als voor zijn tegenstander, zodat het programma moeite zal doen om bijv. een pion te promoveren op het moment dat hij promotie ,,ruikt'' binnen zijn zoekdiepte. De enige theoretische beperking is dat hij de pion van de tegenstander (en ook die van zichzelf) automatisch tot dame promoveert.
- Er is een mogelijkheid om het bord leeg te maken en op willekeurige vakken willekeurig gekozen stukken te plaatsen. Op deze manier kan snel een eindspel worden opgezet.
- Na iedere zet verschijnt automatisch de nieuwe bordsituatie op het beeldscherm.
- TC86 geeft in sommige gevallen commentaar op uw zet of op z'n eigen zet.
- TC86 vertelt u wanneer hij u schaak of schaakmat zet en geeft uit zichzelf op als hij z'n eigen situatie hopeloos acht.
- TC86 kent het belangrijkste van de zettenherhalingsregels (die vrij ingewikkeld zijn). Hij zal zelf, noch zijn tegenstander een zet laten uitvoeren die voor de 3e maal achter elkaar tot dezelfde spelsituatie leidt.
- TC86 kent pat en zal dit vermijden als hij denkt dat er winst mogelijk is.

Speelsterkte
De speelsterte hangt uiteraard af van de zoekdiepte en dus van de bedenktijd (zeer menselijk). De *,,ELO RATING''* is echter nooit gemeten bij TC86.
,,Goed'' en ,,slecht'' zijn subjectieve waarden. Het beste is misschien enkele partijen te bestuderen die in het PCW-kampioenschap zijn gespeeld.

8	..	**	..	**	..	**	..	**
7	**	BK	**	WP	**	..	**	..
6	..	**	BN	**	..	**	..	WR
5	**	..	**	..	**	..	**	..
4	..	**	..	**	..	**	..	**
3	WK	..	**	..	**	..	**	..
2	..	**	..	**	..	**	..	**
1	**	..	**	..	**	..	**	..
	A	B	C	D	E	F	G	H

Afb. 2. Uit deze spelsituatie koos TC 86 de zet H6 x C6. Door een toren te offeren, kan de pion op D7 promoveren tot dame.
Deze afbeelding toont tevens de wijze waarop TC 86 het schaakbord op het beeldscherm weergeeft.

Om te laten zien dat TC86 promotie kan ,,ruiken'' gaan we uit van een spelsituatie volgens afb. 2. (Wit is aan zet). Na 36 seconden koos TC86 de zet H6xC6. Door zijn toren te offeren stelt TC86 zijn promotie veilig!

Algoritme
Het algoritme dat TC86 gebruikt voor het vinden van de beste zet is nogal recht-toe-recht-aan. Details voor een dergelijk onderzoekprogramma vindt u in de Databus-artikelenreeks ,,computerspelen''. Het α/β-afkapmechanisme wordt verbeterd door vooronderzoek waarna de zetten worden gerangschikt.
Een nadeel van het vooronderzoek is dat het zelf niet ten volle van het α/β-afkapmechanisme gebruik kan maken teneinde een betrouwbare waarde aan de zetten toe te kennen. Het onderzoek is vooral gebaseerd op *brute force:* snel en eenvoudig. Op die manier wordt elke 100μs (afhankelijk van de situatie) de verlies- en winstrekening van een zet opgemaakt. Het is daarmee mogelijk in eén minuut een volledig analyse te doen op 5 plies diep. Bovendien vindt een strategisch onderzoek plaats (simultaan) dat minder plies diep gaat omdat dit veel tijdrovender is. Het strategische onderzoek bepaalt centrumbezetting, mobiliteit, dubbelpionnen en de verdediging van de koning.
Om het geheel zo eenvoudig, en vooral zo snel mogelijk te houden, herkent TC86 de stukken aan hun waarde. Die waarde is vast en onafhankelijk van de spelsituatie. Uiteraard zijn de waarden met een bepaalde factor (hier 3) vermenigvuldigd om een reëel verschil te krijgen tussen paard en loper. Een voordeel van deze benadering is dat het aantal stukken van een soort onbeperkt is, zodat inderdaad meerdere koninginnen kunnen meespelen. Bij sommige programma's is men beperkt tot slechts één koningin per kleur.
Het hart van het programma, de zettengenerator, is een re-entrant routine, een subroutine die zichzelf aanroept tot de vastgestelde zoekdiepte. De parameters van elke zet worden in de stack ,,gesaved'', zodat totaal 16 bytes stackruimte per zet is vereist. Met de toegekende stackruimte van 192 bytes kan aldus 12 plies diep worden gezocht.
De zet-parameters zijn; vakje van oorsprong, vakje van bestemming, geslagen stuk, rochade, en-passant slaan, promotiestatus en de richtingpointer. De richtingpointer komt voort uit het feit dat de zettengenerator voor een gedeelte ,,table driven'' is.
Behalve de stack is er natuurlijk geheugenruimte aanwezig waarin zich het speelbord bevindt. Normaal gesproken vraagt dit 1 byte per vakje. De bordruimte voor TC86 is echter groter dan 64 bytes om sneller zetten buiten het bord te kunnen detecteren. Het hoeft geen betoog dat dergelijke routines zeer tijdkritisch zijn voor het denkproces.
Naast stack en schaakbord bevinden zich nog enkele variabelen in RAM, zoals: iteratiediepte, de beste-zet-tot-dan-toe, kleur zwart of wit en random nummer. De toegangssnelheid tot deze variabelen is van weinig belang. De variabelen waarbij dat wel van belang is bevinden zich permanent in de CPU-registers. De rest van het 1 Kbyte geheugen is vrij voor een lijst van zetten met de daarbij behorende verlies/winstwaarde. Deze lijst wordt ook weer gebruikt om de zetten van de tegenstander te controleren op geldigheid.

Programma-details
Hetgeen we hier nader onder de loupe nemen is maar een klein gedeelte van TC86 (< 1Kbyte), namelijk het gedeelte dat uit een willekeurige situatie de beste zet berekent.
De ,,input''-gegevens voor deze routine zijn: de situatie op het bord, de kleur, de iteratiediepte in plies en gegevens omtrent rochade en en-passant mogelijkheden. Het resultaat van de routine is de beste zet (die nog niet is uitgevoerd in zijn geheugen).

In afb. 3 ziet u het programma dat een vakje vindt met een stuk van zijn kleur. Aan de hand van de waarde van het stuk herkent hij het en springt naar verschillende plaatsen met een richtingspointer in het SI-register. Voordat TC86 alle 64 vakjes aftast gaat hij eerst na of er geen mogelijkheid is voor rochade links of rechts.
(Voor degenen die niet op de hoogte zijn van de architectuur van de 8086, van adresseermogelijkheden en instructieset, kunnen de eerder verschenen artikelen in Databus over de 8086 van nut zijn; Databus 79/9, pagina 22 en Databus 81/6, pagina 7).
De routine PATH in afb. 4 behandelt alle stukken die langs een rechte lijn kunnen bewegen: loper, toren en dame. PATH genereert alle mogelijke zetten vanuit de huidige positie van dat stuk (in het BX-register). De richting van elke beweging wordt uit een tabel gelezen. De beweging langs de lijn wordt onderbroken op het moment dat het bestemmingsvakje niet leeg is of het stuk van het bord dreigt te lopen.
De GETDES routine genereert alle mogelijke zetten van de koning of het paard. Ook hier komt de staprichting uit een tabel.

Uit afb. 3 blijkt dat de koning en de dame gebruik maken van dezelfde richtingstabel. Het heeft hier geen zin de dame te behandelen als combinatie loper/toren. Het enige dat men wint is 9 bytes tabelruimte, terwijl aan de andere kant het programma uitgebreider en slechter leesbaar wordt, en waarschijnlijk ook langzamer. De routines PATH en GETDES voeren de zetten uit; zij bewegen de stukken slechts in gedachte.

Voor schaakprogramma's is de pion het „enfant terrible" van het bord: vooruitlopen, schuin slaan, en-passant slaan en promotie. De behandeling van de pion wordt hier niet weergegeven...

Indien de huidige zoekdiepte nog niet gelijk is aan de maximale diepte (MDEPTH) dan „groeit" er een tak aan de boom in CONT (afb. 5). Is de zoekdiepte gelijk aan het maximale aantal plies, dan volgt MOVPOS waarin de score voor dat eindpunt wordt berekend.
CONT is een typisch voorbeeld van een „re-entrant" routine die, aangeroepen door een CALL-instructie, gegevens in de stack plaatst en daarna zichzelf weer aanroept, zij het indirect via JMP MX. De subroutine STATST verandert eventueel de CASTLE STATUS indien de koning of één van de torens worden verplaatst . Als de vijandelijke koning op slag staat heeft verder doorzoeken geen zin meer. Het programma breekt de tak op dat moment af en geeft een hoge score. Dit gebeurt in regel 206 in het programma.

In afb. 6 ziet u de listing van het programmadeel dat de rochade behandelt, voor zowel wit als zwart, lang en kort. Deze informatie komt binnen via pointer DI. Een „look up" tabel geeft de nummers van de vakjes die leeg zouden moeten zijn (tussen koning en toren). Zijn die vakjes niet leeg, dan vindt een RETURN plaats. Voordat het programma TSTCAS binnenkomt is het reeds zeker dat de koning en bewuste toren niet van hun plaats zijn geweest.
Als een vakje leeg is wordt nagegaan of dit vakje niet wordt aangevallen door een vijandig stuk. Deze controle gebeurt in regel 257...385. In feite doet deze routine het omgekeerde van de zettengenerator.

```
05CD E88403          64   FNDBST: CALL    CLEAN              ;ERASE MOVELIST
05D0 C706980011C003  65           MOV     LSTPTR,OFFSET LST
05D6 8A3E9801        66           MOV     BH,DS:BYTE PTR OFFSET COMCOL+1
05DA E8DC02          67           CALL    FETCH              ;IS CASTLING POSSIBLE?
05DD A804            68           TEST    AL,4
05DF 7517            69           JNZ     M7                 ;NO, MAY BE AT RIGHT SID
05E1 A808            70           TEST    AL,8
05E3 7506            71           JNZ     M5
05E5 BFA105          72           MOV     DI,OFFSET PTRL     ;NO PIECES BETWEEN
05E8 E84202          73           CALL    TSTCAS             ;KING AND ROCK?
05EB EBCB02          74   M5:     CALL    FETCH
05EE A801            75           TEST    AL,1
05F0 7506            76           JNZ     M7
05F2 BFAA05          77           MOV     DI,OFFSET PTRR     ;SAME FOR RIGHT SIDE
05F5 E83502          78           CALL    TSTCAS
05F8 B300            79   M7:     MOV     BL,0               ;START LOOKING AT SQUARE
05FA 803F00          80   M6:     CMP     BYTE PTR [BX],COL  ;WHAT IS ON THAT SQ.?
05FD 7F10            81           JG      M2                 ;A PIECE OF MY COLOR!
05FF 43              82   M3:     INC     BX                 ;NO, NEXT SQ.
0600 F6C388          83           TEST    BL,88H             ;OFF BOARD?
0603 74F5            84           JZ      M6                 ;NO, GO ON
0605 7805            85           JS      M4                 ;LAST SQ. DONE
0607 80C308          86           ADD     BL,8               ;NEXT ROW
060A EBEF            87           JMP     M6
060C E91A01          88   M4:     JMP     NOMOV
060F B400            89   M2:     MOV     DH,0               ;CLR EXT. OF DL REG
0611 8A07            90           MOV     AL,[BX]            ;GET PIECE INTO AL
0613 3C08            91           CMP     AL,KNV             ;IS IT KNIGHT?
0615 7861            92           JS      PAWN               ;SMALLER THUS PAWN
0617 BE8F05          93           MOV     SI,OFFSET KNPTR    ;LOAD TABLE PTR
061A 7413            94           JZ      GETDES             ;YES KNIGHT!
061C BE9805          95           MOV     SI,OFFSET BIPTR    ;PTR FOR BISHOP
061F 3C12            96           CMP     AL,ROV             ;IS IT ROCK?
0621 782E            97           JS      PATH               ;SMALLER THUS BISHOP
0623 BE8A05          98           MOV     SI,OFFSET ROPTR    ;LOAD TABLE PTR
0626 7429            99           JZ      PATH               ;YES IT IS ROCK!

0628 BE8105         100           MOV     SI,OFFSET KIPTR    ;LOAD PTR
062B 3C40           101           CMP     AL,KIV             ;IS IT KING?
062D 7822           102           JS      PATH               ;NO IT IS QUEEN
```

Afb. 3.Deze routine zoekt en vindt alle stukken van zijn kleur. Aan de hand van de waarde die elk stuk heeft, herkent het programma om welk stuk het gaat.

```
                    103
                    104           ;ROUTINE TO GENERATE ALL LEGAL MOVES
                    105           ;FOR KNIGHT OR KING
                    106
062F 8BFB           107   GETDES: MOV     DI,BX              ;DST=ORG
0631 2E8A14         108           MOV     DL,CS:[SI]         ;GET DIRECTION
0634 03FA           109           ADD     DI,DX              ;CALC. SQ. MOVING TO
0636 F7C78800       110           TEST    DI,88H             ;OFF BOARD?
063A 750C           111           JNZ     G4                 ;YES
063C 81E77706       112           AND     DI,677H            ;NO, CLEAN UP DI
0640 803D00         113           CMP     BYTE PTR [DI],COL        ;WHAT IS ON THAT
0643 7F03           114           JNLE    G4                 ;OCCUPIED BY MYSELF
0645 E89700         115           CALL    MOVPOS             ;EMPTY OR OPPONENT
0648 46             116   G4:     INC     SI                 ;NEXT DIRECTION
0649 2EF60499       117           TEST    BYTE PTR CS:[SI],99H     ;END OF TABLE?
064D 75E0           118           JNZ     GETDES             ;NOT YET
064F EBAE           119           JMP     M3                 ;YES, STOP IT
                    120
                    121           ;ROUTINE TO GENERATE ALL LEGAL MOVES FOR PIECES
                    122           ;MOVING ALONG A LINE:BISHOP,ROCK AND QUEEN
                    123
0651 8BFB           124   PATH:   MOV     DI,BX              ;DST=SRC
0653 2E8A14         125           MOV     DL,CS:[SI]         ;GET DIRECTION
0656 03FA           126   S2:     ADD     DI,DX              ;NEXT SQ. IN THAT DIRECTI
0658 F7C78800       127           TEST    DI,88H             ;OFF BOARD?
065C 7511           128           JNZ     S3                 ;YES
065E 81E77706       129           AND     DI,677H            ;NO, CLEAN UP DI
0662 803D00         130           CMP     BYTE PTR [DI],COL        ;WHAT IS THERE?
0665 7F03           131           JG      S4                 ;OCCUPIED BY MYSELF
0667 E87500         132           CALL    MOVPOS             ;EMPTY OR OPPONENT
066A 803D00         133   S4:     CMP     BYTE PTR [DI],0    ;IF IT WAS EMPTY,
066D 74E7           134           JZ      S2                 ;CONTINUE THAT DIRECTION
066F 46             135   S3:     INC     SI                 ;GET NEXT DIRECTION
0670 2EF60499       136           TEST    BYTE PTR CS:[SI],99H ;ALL DIRECTIONS DONE
0674 75DB           137           JNZ     PATH               ;NO, CONTINUE
0676 EB87           138   S1:     JMP     M3                 ;YES, STOP IT
                    139
```

Afb. 4. De routine PATH genereert alle mogelijke zetten van een loper, toren en dame.

```
                    186
                    187             ;LEGAL MOVE IS POSSIBLE
                    188             ;BX=ORG.SQ.,DI=DEST.SQ.
                    189
06DF 392E9201       190     MOVPOS: CMP     MDEPTH,BP       ;COMP. ACTUAL PLY DEPTH
06E3 7F12           191             JG      CONT            ;LOWER: EXPAND TREE
06E5 8AC5           192             MOV     AL,CH           ;PROFIT & LOSS
06E7 2A05           193             SUB     AL,[DI]         ;UPDATE IT
06E9 3A462D         194             CMP     AL,[BP+DEP]     ;COMPARE WITH LOWER NODE
06EC 7E08           195             JLE     MP2             ;WORSE: FORGET IT!
06EE 884600         196             MOV     [BP],AL         ;SUBSTITUTE SCORE
06F1 02462C         197             ADD     AL,[BP+DEP-1]   ;ALPHA BETHA MINMAX?
06F4 7D32           198             JNL     SPEED           ;YES, CUT BRANCH
06F6 C3             199     MP2:    RET                     ;NO, TRY NEXT MOVE
                    200
                    201             ;ROUTINE TO GROW TREE
                    202             ;LEGAL MOVE IS EXECUTED AFTER HISTORICAL
                    203             ;DATA ARE SAVED ON STACK
                    204
06F7 803DC0         205     CONT:   CMP     BYTE PTR [DI],-KIV      ;KING TAKEN?
06FA 7428           206             JZ      CHECK           ;YES:NO NEED TO CONTINUE
06FC E88802         207             CALL    STATST          ;DID KING OR ROCK MOVE?
06FF 56             208     C7:     PUSH    SI              ;PUSH VITALE DATA
0700 52             209             PUSH    DX
0701 FF35           210             PUSH    WORD PTR [DI]
0703 57             211             PUSH    DI
0704 53             212             PUSH    BX
0705 51             213             PUSH    CX
0706 8A0EA101       214             MOV     CL,WARN         ;SAVE
070A C606A10100     215             MOV     WARN,0          ;EP STATUS
070F 45             216             INC     BP              ;ACTUAL PLY DEPTH
0710 E8F300  218    217         MOV CALL,7EMOVE         ;LOAD;EXECUTE MOVE ON BOARD
0715 8B07           219             MOV     AX,[BX]         ;CASTLE STATUS
0717 50             220             PUSH    AX              ;AND SAVE IT
0718 8AC4           221             MOV     AL,AH           ;UPDATE
071A 8907           222             MOV     [BX],AX         ;CASTLE STATUS
071C 8BDE           223             MOV     BX,SI           ;CHANGE COLOR
071E E93DFD 225     224             JMP     MX              ;AND MOVE AGA
```

Afb. 5. De routine CONT.

```
0820 B400           343     TSTCAS: MOV     DH,0            ;CLEAR DL EXTENSION
0822 56             344             PUSH    SI              ;SAVE POINTER
0823 F6C704         345             TEST    BH,4            ;LEFT CASTLE?
0826 7403           346             JZ      TC6             ;ELSE ADD OFFSET
0828 83C712         347             ADD     DI,OFFSET PTRLBL-OFFSET PTRL
082B 2E8A1D         348     TC6:    MOV     BL,CS:[DI]      ;GET SQUARE
082E F6C330         349             TEST    BL,30H          ;LAST ONE?
0831 7547           350             JNZ     TC7
0833 003F00         351             CMP     BYTE PTR [BX],0 ;EMPTY?
0836 7407           352             JE      TC0             ;YES, OK
0838 803F40         353             CMP     BYTE PTR [BX],KIV;ELSE IS IT KING?
083B 7402           354             JE      TC0             ;IF NOT
083D 5E             355             POP     SI              ;STOP TEST AND
083E C3             356             RET                     ;RETURN
083F 57             357     TC0:    PUSH    DI              ;ARE THESE SQUARES ATTACKE
0840 BE8005         358             MOV     SI,OFFSET PTRCAS;GET NEW POINTER
0843 2E8B04         359     TC8:    MOV     AX,CS:[SI]      ;LOAD 2 BYTES AT THE TIME
0846 3C60           360             CMP     AL,60H          ;END OF TEST?
0848 742C           361             JZ      TC5             ;YES
084A 46             362             INC     SI              ;NO GET DIRECTION POINTER
084B 46             363             INC     SI
084C 8BF8           364     TC1:    MOV     DI,BX
084E 2E8A14         365             MOV     DL,CS:[SI]      ;LOOK UP DIRECTION
0851 03FA           366     TC2:    ADD     DI,DX           ;AND CALCULATE DESTINATIO
0853 F7C78800       367             TEST    DI,88H          ;OF BOARD?
0857 7514           368             JNZ     TC3
0859 81E777 06      369             AND     DI,677H         ;NO CLEAN UP DI
085D 3B25           370             CMP     [DI],AH         ;IS IT THE SAME PIECE?
085F 7503           371             JNE     TC4
0861 5F             372             POP     DI              ;YES,NO CASTLING POSSIBLE
0862 5E             373             POP     SI
0863 C3             374             RET
0864 3C00           375     TC4:    CMP     AL,0            ;NO,MOVES IT ALONG LINE?
0866 7505           376             JNZ     TC3
0868 803D00         377             CMP     BYTE PTR [DI],0 ;YES,SEE IF CAN GO FURTHE
086B 74E4           378             JZ      TC2
086D 46             379     TC3:    INC     SI              ;NEW DIRECTION
086E 2EF60499       380             TEST    BYTE PTR CS:[SI],99H
0872 7508           381             JNZ     TC1             ;CONTINUE TESTING
0874 EBCD           382             JMP     TC8
0876 5F             383     TC5:    POP     DI              ;ON NEXT EMPTY SQUARE
0877 47             384             INC     DI
0878 EB81           385             JMP     TC6
087A E82F00         386     TC7:    CALL    FETCH           ;CASTLING ALLOWED
087D 8AE0           387             MOV     AH,AL           ;MODIFY CASTLE STATUS
087F 80C404         388             ADD     AH,4
0882 02C3           389             ADD     AL,BL
0884 E82C00         390             CALL    STORE           ;AND SAVE IT
0887 8BF7           391             MOV     SI,DI
0889 E80F00         392             CALL    UPDAT           ;LOOK-UP ROCK MOVE
088C 56             393             PUSH    SI
088D EB69FF         394             CALL    MOVE            ;MOVE ROCK
0890 5E             395             POP     SI
0891 E80700         396             CALL    UPDAT           ;SEE HOW KING MOVES
0894 5E             397             POP     SI              ;RETREIVE ORIGINAL
0895 80C501         398             ADD     CH,1            ;ADJUST LOSS
0898 E957FE         399             JMP     C7              ;AND CONTINUE TREESEARCH
                    400
```

Afb. 6. Deze routine behandelt de rochade.

Als voorbeeld: vanuit het lege vakje gaan we diagonaal in alle richtingen om te zien of er een vijandige loper is te vinden. Vinden we een ander stuk, of de rand van het bord, dan wordt het lege vakje niet door een loper aangevallen. En zo wordt het bewuste vakje verder onderzocht op aanvallen door andere stukken.
Het is mogelijk TSTCAS sneller te maken door éérst te onderzoeken of alle vakjes leeg zijn en dan pas na te gaan of al deze vakjes niet worden aangevallen (zoals een menselijke schaker ook zou doen: het brein is soms erg efficiënt al is men zich er niet altijd van bewust). Reden dat deze snelheidsverbetering nog niet is uitgevoerd, is de geringe invloed van TSTCAS op de totale snelheid.
Nadat aan alle rochadevoorwaarden is voldaan springt het programma naar label C7 in CONT. De routine STATST is overbodig omdat verdere rochades nu onmogelijk zijn gemaakt.

Omschakeling

De ontwikkeling van TC86 is voor 90% gebeurd op de SDK-86, een „design kit" gebaseerd op de 8086, voorzien van 4 Kbyte RAM en een monitorprogramma. De software is geëdit en geassembleerd op een Intel-ontwikkelsysteem en vervolgens via een seriële interface in de SDK-86 geladen. Debugging gebeurde met behulp van de breakpoint – en single-step mogelijkheden van het monitorprogramma. Slechts een gedeelte van het testen van het eindprodukt dat (nog) geen monitor bezat, is uitgevoerd met behulp van een ICE (In Circuit Emulator). Op dat moment was ik al zeker van het algoritme; slechts enkele typische karakteristieken, zoals I/O-communicatie, moesten worden getest.

Tot slot: in de toekomst zullen mijn inspanningen vooral zijn gericht op een verbetering van het eindspel. Daarnaast verwacht ik geen spectaculaire vooruitgang. De ontwikkeling van meer gecompliceerde, meer „menselijk denkende" programma's is zeer tijdrovend en gewoonlijk erg teleurstellend. Men zal een verbetering teweeg brengen in de ene situatie, terwijl in andere omstandigheden weer wakker wordt gespeeld. Vergeet niet dat het onderzoeken van finesses altijd zeer veel programmatijd vereist.

Databus (1981) - Jan Kuipers: Tiny Chess 86