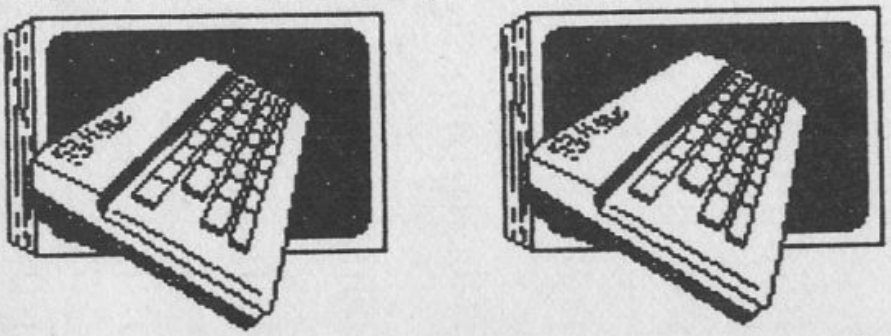
DE SPECTRUM DE SPECTRUM DE SPECTRUM

DE SPECTRUM DE SPECTRUM DE SPECTRUM

DE SPECTRUM DE SPECTRUM



SPECIAAL KRAAK NUMMER 4

Lessen in het kraken van spelen deel 1:

Schijnbaar kan niemand meer tegenwoordig een eenvoudig spel kraken. Met uitzondering van Multicrack, Ram en RL dan. De andere mensen gebruiken liever een copieerdoos met een knopje er op. Zodra je op dit knopje drukt wordt het geheugen naar tape gecopieerd en daarmee wordt het spel gecopieerd.

Met deze stukjes hoop ik dat er enkele mensen zo ver komen dat ze in staat zijn om zonder deze doosjes te kraken. Voor het kraken heb je een redelijke dosis machinetaal kennis nodig. Heb je die niet, dan hoef je het alvast niet te proberen. Oefening baart kunst is het gezegde. Met een beetje oefening kun je een kleine hoeveelheid machinetaal kennis vrij vlug vergroten.

Ik zal aan de hand van verscheidene voorbeelden gaan uitleggen hoe je spelletjes moet kraken, ook hier geldt weer oefening baart kunst.

Wat heb je nog meer nodig om te kraken? Het antwoord hierop is vrij simpel, een Spectrum, een cassetterecorder, een assembler en een disassembler. Een printer zou ook erg welkom kunnen zijn, echter is een printer niet altijd verstandig.

Welke assembler en welke disassembler moet je gebruiken? Op deze vraag geef ik het antwoord DEVPAC, het liefst een versie waar MONS3 nog klein is, 6K of zo. Dit omdat MONS3 klein is, handig in het gebruik (als je er aan gewend bent), omdat je breakpoints kunt zetten en omdat je kunt singlesteppen. Maar het belangrijkste is dat MONS3 relocatable is!

Eerst even wat informatie over allerlei loaders. Er zijn verschillende mogelijkheden om een spel in te laden, via machinetaal of via BASIC. Spelen beginnen meestal met een BASIC loader. Degene die deze BASIC loaders doorziet is al een heel eind op de weg. Het beste op zijn plaats is het volgende voorbeeld.

We beginnen met een eenvoudig spel, MATCHPOINT. Deze is niet te copieren met Mister Copy maar wel met allerlei andere copieerprogramma's (zelf gebruik ik vaak SPY 007). De wekwijze is als volgt. MERGE het programma naar binnen. In de listing staat:

10 PAPER 5:INK 5:BORDER 5:CLS:FOR N=23296 TO 23313:READ A:POKE N,A:NEXT N
20LOAD "" CODE 30000:RANDOMIZE USR 30000
200 DATA ..... (Zie een origineel voor de rest).

Met een tapeheader reader kunnen we zien dat de basic start op regel 0. (Gebruik bijvoorbeeld HREADRAM of READ mode (4) van Miter Copy). Wel, het eerste belangrijke dat we zien gebeuren is dat er iets in de printerbuffer wordt gepoked. WE KUNNEN ONZE PRINTER DUS NIET MEER GEBRUIKEN WANT DEZE GEBRUIKT OOK DE PRINTERBUFFER. Er wordt niet ergens anders buiten de printerbuffer gepoked, dus treden er geen extra complicaties op. Op regel 20 staat een LOAD "" CODE 30000. Deze voeren wij even met de hand uit. Met een tapeheaderreader vinden we uit dat dit blok ongeveer 4300 bytes groot is, naar het gehoor is het een tekening. Als je dan op een schatting van 7000 bytes gaat zitten heb je ruimte genoeg over.

Mons3 moet nu worden ingeladen, bijvoorbeeld op 30000+7000=37000. Even een USR op 37000 geven en we zitten in MONS3. Het volgende dat zou gebeuren was RANDOMIZE USR 30000 dus kijken we op 30000. We voeren in H 30000 en krijgen te zien dat 30000 decimaal gelijk is aan 7530 hexadecimaal. We zetten de memorypointer op 7530 door in te voeren M 7530. Dan doen we Symbol Shift 4 en krijgen een disassembly die er als volgt uit ziet.

```
CALL  #7536      <--Voer iets uit.
JP    #5B00      <--Hier wordt naar de printerbuffer gesprongen.
LD    HL,#4000   <--Dit wordt er uitgevoerd. #4000=16384.
LD    DE,#7560      We mogen concluderen dat het scherm wordt
LD    BC,#5B00      opgebouwd. Onze eerste gok is dat daarna
... etc.            keurig een RET volgt zodat we bij de JP uit
                    komen.
```

We gaan dus op #5B00 kijken. Sym Shift 4 invoeren om weer de memorypointer te kunnen verzetten. M 5B00 om naar de printerbuffer te gaan en Sym Shift 4 om de listing te zien.
Op #5B00 staat dan:

```
LD    SP,#5BCC    <-Zet de stackpointer in de printerbuffer
                    Dit is vrij ongevaarlijk, wij hebben Mons3
                    niet in de weg staan.
LD    IX,#5C6B    <-Zet het IX register op de plek waar het
                    volgende blok wordt ingeladen.
LD    DE,#A394    <-Geef de lengte van dat blok.
```

```
XOR   A           <-Geef aan dat het een headerblok is.
                    Hier hebben we de oorzaak gevonden dat
                    Mister copy dit blok niet over kan nemen.
                    Mister copy ziet een blok headerless alleen
                    als er een flagbyte van #FF aan staat en
                    en niet als er een 0 aan staat. (XOR A
                    levert 0 op).
SCF               <-Carryflag zetten om in te laden.
                    Resetten van de carryflag voor VERIFY.
CALL  #0556       <-Load bytes. (zie THE COMPLETE SPECTRUM ROM
                    DISASSEMBLY door IAN LOGAN of neem het als
                    een fait a compli van mij aan).
JP    #6300       <-Dit is zeer waarschijnlijk een sprong
                    naar de start van het spel.
```

De reken machine er even bij:#5C6B=23659
#A394=41876 +

65535

Het gehele geheugen wordt dus vanaf #5C6B gebruikt. Het beste kunnen we dit grote blok dus in twee headerless blokken verdelen. We resetten de Spectrum even en maken het onze eigen loader.

```
10 INK 0:PAPER 0:BORDER 0:CLS:LOAD ""CODE:RANDOMIZE USR 30000
```

Deze saven als MATCHPOINT met LINE 10.
Dan moeten we de tekening nog even overnemen. Dit kan goed met Mister copy gebeuren. We weten dat na het maken van de tekening naar de printerbuffer wordt gesprongen dus hoeven we niets extra's toe te voegen.
Dan geven we een CLEAR 29999 en laden GENS3 in op 30000. Starten met een RANDOMIZE USR 30000. Even bufferspace 0 invoeren, iets anders heb je vaak nooit nodig. Dan de volgende dingen invoeren,

```
I                 <- Naar inputmode.
*D+                     <- Alles decimaal.
      ORG   23296
      ENT   $
      LD    SP,#5BCC
      LD    IX,#5C6B
      LD    DE,#A394
      XOR   A
      SCF
      CALL  #556        <- Inladen van het blok, zie ook boven.
```

```
      DI                  <- BELANGRIJK! Als uit #556 terug wordt
                          gekomen staan de interrupts weer aan en
                          loopt de spectrumklok, dus enkele systeem-
                          variabelen veranderen. Het is een goede
                          gewoonte om er voor te zorgen dat het te
                          copieren blok niet van inhoud verandert.
                          Je weet immers nooit of er op wordt gecon-
                          troleerd.
      LD    IX,#5C6B    <- Start eerste te saven headerless blok.
      LD    DE,#5394    <- Lengte hiervan.
      CALL  SAVE        <- Save dit blok.
      LD    IX,#5C6B+#5394   <- Start 2e blok.
      LD    DE,#5000         <- Lengte hiervan.
      CALL  SAVE             <- Save dit blok.

SAVE  XOR   A           <- Reageer op alle toetsen.
      IN    A,(#FE)     <- Lees toetsenbord.
      BIT   0,A         <- Reageer op buitenste toetsen.
      JR    NZ,SAVE     <- Niet ingedrukt?
      LD    A,#FF       <- Wel ingedrukt, zet flagbyte op data-
                           blok. (Zie Rom-disassembly)
      CALL  #4C2        <- Save blok
      DI                <- Zet interrupts uit. Zie boven.
      RET
```

In plaats van BIT 0,A en JR NZ,SAVE kun je ook RRCA en JR C,SAVE gebruiken, dat is 1 byte korter.
Even assembleren: A <ENTER>
<ENTER>
<ENTER>

Onthoud het eindadres.
Terug naar BASIC met B.
Dan SAVE "MLS"CODE 23296,150 en het stukje even op een vrije tape saven. GENS 3 weer in door RANDOMIZE USR 30059.
I

```
      ORG   23296
      ENT   $
      LD    SP,#5BCC
      LD    IX,#5C6B
      LD    DE,#5394
      LD    A,#FF
      SCF
      CALL  #556
```

```
LD      IX,#5C6B+#5394
LD      DE,#5000
LD      A,#FF
SCF
CALL    #556
JP      #6300
```

Moet in worden gevoerd. Assembleren met A <enter><enter><enter>. B om terug naar basic te gaan. SAVE "LOADER"CODE 23296,150 invoeren en het stukje code na de tekening saven.
Nu moet het stukje dat we op de vrije tape hadden gesaved worden ingeladen. Draai het Matchpoint origineel bij het juiste blok, dit is na de tekening. Voer een USR 23296 uit en laadt het matchpoint blok in. Als het is ingeladen kun je de beide te saven blokken saven door op een buitenste toets te drukken. Zo save je beide blokken.
Het werk is dan bijna gedaan, je hoeft alleen nog je gekraakte versie even uit te proberen.

Kraken les2: het "kraken" van een BASIC loader.

De meeste spelen bestaan uit meerdere blokken waarvan het eerste blok een BASIC loader is. Om spelen te kraken moet deze BASIC loader meestal eerst worden gekraakt. Ik behandel achtereenvolgens verschillende manieren van aanpakken voor het kraken van een BASIC loader.

Voor het kraken van een BASIC loader hebben we een assembler nodig. Ik gebruik hier meestal GENS3 voor door zijn kleine formaat. Ook is het handig dat je een tapeheader reader hebt die de volledige informatie geeft over een header.
Hier volgt de sourcecode voor een eenvoudige tapeheader reader.

```
      ORG   60000
      ENT   $
      LD    IX,23296      ! HEADER OP 23296 LADEN
      LD    DE,17         ! HEADERS ZIJN 17 BYTES LANG
      XOR   A             ! FLAGBYTE 0
      SCF                 ! VOOR HET LADEN
      CALL  #556          ! LAAD BLOK, ZIE ROM DISASSEMBLY
      RET
```

Hier hoort een stukje basic bij:

```
10 PAPER 0:INK 7:CLS:CLEAR 59999:LOAD "HCODE"CODE 60000
20 CLS: PRINT #0;"LOAD HEADER NOW": RANDOMIZE USR 60000
30 PRINT "FLAGBYTE=";PEEK 23296
40 PRINT "NAAM=";:FOR N=23297 TO 23306: LET A=PEEK N: PRINT
   A;" ";(CHR$ A AND A>=32);: NEXT N
50 PRINT '"LENGTE=";PEEK 23307+256*PEEK 23308
60 PRINT "STARTREGEL=";PEEK 23309+256*PEEK 23310
70 PRINT "VARIABELEN VERPLAATSING=";PEEK 23311+256*PEEK 23312
80 PRINT '"PRESS ANY KEY TO CONTINUE":PAUSE 0:BEEP .1,-30:GOTO
   20
```

Deze uitkomsten moeten even op een stukje papier worden genoteerd zodat ze later makkelijk kunnen worden terug gevonden.

Sommige spelen kunnen niet worden gemerged, deze spelen geven een OUT OF MEMORY error als ze worden gemerged of ze blijven gewoon hangen. Het beeld schuift dan naar rechts. Bijvoorbeeld het mergen van de loader van Doomdarks revenge zorgt voor een OUT OF MEMORY error. Het MERGEN van ALIEN8 of HYPERSPORTS of welke andere speedlock dan ook zorgt voor een vasthangend beeld.

Wat we nu gaan doen is een nonauto start versie van zo'n BASIC loader maken. Dat gaat als volgt. We laden de header in een tapeheaderreader en met behulp van de nonauto save mode van deze tapeheaderreader saven wij deze header. Gebruik hiervoor niet SPY 007 de A mode, deze werkt verkeerd! Even de sourcecode voor een nonautosave routine.

```
      ORG   60000
      ENT   $
      LD    IX,23296
      LD    DE,17
      XOR   A
      SCF
      CALL  #556
      LD    BC,#FFFF     !GEEF FOUT AAN
      RET   NC           !TERUG ALS HET LADEN MISLUKTE
      LD    IX,23296
      LD    (IX+13),#FF  !MAAK NONAUTOSTART HEADER
      LD    (IX+14),#FF
      LD    DE,17
WAIT  XOR   A
      IN    A,(#FE)      !TEST OP 2,W,S,Z,9,O,L,SYM SHIFT
      BIT   1,A
      JR    NZ,WAIT
      XOR   A
      CALL  #4C2
      LD    BC,#FFFE     !BREAK KEY PRESSED ERROR
      RET   NC
      LD    BC,0         !ALL OK, NO ERROR
      RET
```

Het BASIC gedeelte luidt als volgt:

```
10 PAPER 0:INK 7:CLS:CLEAR 59999:LOAD "LS"CODE 60000
20 CLS:PRINT #0;AT 0,0;"LOAD HEADER PLEASE    ":LET A=USR 60000
30 IF A=65535 THEN PRINT #0;AT 0,0;"NO HEADER LOADED     "
40 IF A=65534 THEN PRINT #0;AT 0,0;"BREAK KEY PRESSED    "
50 IF A=0 THEN PRINT #0;AT 0,0;"HEADER SAVED      "
60 GOTO 20
```

Dan hoef je alleen de spectrum nog te resetten en het volgende te doen. Op het origineel moet het blok data achter de BASIC header worden gezocht. Rewind de gesavede header en doe als volgt. LOAD "" en laad de gesavede header in. Stop de tape en doe het origineel er in. Start de tape. Zie daar, het BASIC gedeelte is zo ingeladen. Nu nog even gauw saven voor de zekerheid. Gebruik hiervoor SAVE "Spel".

Doorgaans kun je geen regels toevoegen of verwijderen omdat dan de variabelen worden verplaatst. Bij sommige loaders (Speedlock bijvoorbeeld) staan de machinecode die de volgende blokken in laadt op de plek van de variabelen.

Wat we nu kunnen zien is een BASIC loader. We nemen bijvoorbeeld DOOMDARKS REVENGE:
Als we nu een list geven zien we alleen maar flauwekul staan, dit moet dus nog even nader worden bestudeerd.
Van de DOOMDARKS REVENGE weten we dat hij 483 bytes lang is (tapeheaderreader) en dus hoeven we alleen nog maar de volgende regel in te typen.

```
FOR N=23755 TO 23755+483:LET A=PEEK N:PRINT N;TAB 15;A;TAB 20;
(CHR$ A AND A>=32):NEXT N
```

Met het spectrumboekje erbij zien wij dan de volgende listing komen:

```
REM    HEY! LOOK BILL ANOTHER PIRATE (flash 1 bright 1)
POWER-LOAD 48 (flash bright etc)  TAG 1984 BY INCENIVE
SOFTWARE (ink 7 paper 7 13=enter)
(ink 7 etc) POKE 23693,56:POKE 23624,63:CLEAR 57999:POKE
23659,0:FOR N=30 TO 36:BEEP .075,N:NEXT N (INK 7):RANDOMIZE USR
24061:RANDOMIZE USR 0 (13=ENTER)
```

Daarna volgt flauwekul, dit klopt we zijn immers bij de machinetaal aangeland.

Eerst even nog iets tussendoor wat van groot belang is: we zien hier staan bijvoorbeeld CLEAR 57999. Dit is vertaald als

```
253   CLEAR
53    5
55    7
57    9
57    9
57    9
14    <- Dit betekent dat er een nummer volgt:
0
0
215   Deze twee bytes bevatten het getal namelijk
189   189*256+215=48599 Het programma doet een CLEAR op 48599
0     in plaats van op 57999!
58    :
.. ETC
```

Zo ook, het programma doet een RANDOMIZE USR 24129 in plaats van een RANDOMIZE USR 24061. Soms (Chip Fact=Technician Ted) kan het voorkomen dat alle 5 bytes achter de numberbyte 14 zijn ingevuld. Dit is ook geen probleem, we hoeven dan alleen de waarden over te nemen en het volgende basicprogramma uit te proberen.

```
10 LET A=0:PRINT A:FOR N=23764 TO 23768:INPUT (N);":";P: POKE N,
P: NEXT N: GOTO 10
```

Op deze manier krijgen we de juiste inhoud van de variabelen afgedrukt en niet de asciiwaarden.

Nu zien we dat we de USR hebben gevonden en we kunnen beginnen met het echte kraakwerk, dwz MONS3 inladen en de machinetaal nalopen. MONS3 moet natuurlijk wel op een goede plek worden ingeladen, bijvoorbeeld op 40000.

Kraken les 3:STARSTRIKE 2

Evenals zoals voor vele spelen geld voor STARSTRIKE 2 ook dat als je eenmaal de BASIC loader in kunt komen dat je dan het belangrijkste deel van het werk al hebt gedaan.

Als we de BASIC loader in trachten te mergen zien we dat het programma blijft hangen. Dus moeten we eerst een nonauto startversie van deze loader maken. Als we deze loader dan op de bekende (zie vorige lessen) manier bestuderen, komen we op de volgende listing uit.

Startregel is line 10.

```
10 <- <- <- ... etc ,zo vele malen chr$8.
   CLEAR 32767
20 LET RED=0 (209 IN HET ECHT)
30 LET HERRING=0 (3)
40 LET FISH=23659 (23613)
50 LET COLOUR=23665 (23614)
60  POKE  COLOUR,RED:POKE  FISH,HERRING   <PAPER   7>:PRINT   AT
PI/PI,NOT PI;"Programmed"
70 LOAD ""CODE 23613
80 LOAD ""CODE 32768 (16384)
90 RANDOMIZE USR 64770
100 LET A=4+PEEK23635+256*PEEK23636
110 IF PEEK A<>32 THEN STOP
120 POKE A,8:LET A=A+1:GOTO 110
65535 REM crashline
```

Regel 10,20,30,40 worden normaal uitgevoerd.

In regel 50 wordt de systeemvariabele ERRSP gepoked. Als er nu een foutmelding zou komen (bijv. D:BREAK CONT REPEATS) dan zou op de plek worden gekeken waar 23613/14 heen wijzen. Daar staat dan het adres waar heen wordt gesprongen. Dit voeren we dus niet uit, omdat er direkt na een LOAD ""CODE 23613 volgt. Deze laden we even op een veilige plek in. Dan zien we dat er 3 bytes worden ingeladen, 169,26,99. 169+256*26=6825 en op 6825 staat 183,17,... 183+256*17=4535=#11B7 is het adres voor NEW.

Bij een break doet het programma dus een NEW. Laden we dan het volgende stuk in op een veilige plek (32768) dan zien we dat na een tijdje een TAPE LOADING ERROR volgt. Dit programma gaat dus lopen door een error te krijgen. Als we de bytes natellen zien we dat er 7230 bytes worden ingeladen.

Het natellen kan mbv het volgende programmaatje worden gedaan.

```
      ORG   60000
      ENT   $
      LD    IX,0
      LD    DE,#FFFF
      LD    A,#FF
      SCF
      CALL  #556
      LD    HL,#FFFF
      SBC   HL,DE
      LD    B,H
      LD    C,L
      RET
```

Een PRINT USR van 60000 en het inladen van het datablok van de loader geeft het getal 7230 te zien. Even optellen 16384+7230=23614. Dus 23613 en 23614 krijgen hier ook weer een nieuwe waarde. Als we dat controleren zien we dat op de plaats voor 23613/14 (print peek(23613+16384),peek(23614+16384)) staat #FE,#5B. Dus moet je opdezelfde plaats kijken (#5BFE+16384 als je het blok nog steeds op 32768 ingeladen hebt staan). Daar staat #00,#5B,#FF,... dus er wordt gesprongen naar #5B00. Daar staat het volgende:

```
DI
LD    HL,#5B0F
LD    DE,#FF00
LD    BC,#0100
LDIR
JP    #FF00
LD    SP,#0000
LD    IX,#5C40
LD    DE,#AC20
LD    HL,#50BF
LD    (#5C3D),HL
LD    HL,#FFC0
PUSH  HL
... etc, ongeveer gelijk aan #556 ROM loading routine
onderweg staat nog een stukje:
PUSH  HL
LD    HL,(#5C3D)
LD    A,H
DEC   H
... tot en met
LD    (HL),#5A
LD    (#5C3D),HL
```

dit stukje is om de thermometer aan de rechterkant van het scherm te verhogen.
Het programma eindigt met het stukje

```
LD    A,H
CP    #01
RET
```

#FFC0 wordt van stack gehaald en daar wordt heen gesprongen.
Op #FFC0 zou dan staan

```
LD    HL,#61A8
LD    A,(#5C3A)
JR    NC,#FFC8
ADD   HL,HL
DEC   A
LD    (#FFCF),A
PUSH  HL
RST   #8
```

<-- #FFC8 komt dan hier terecht.
Opvallend is de PUSH HL waarvan nog niet te zien is waar hij voor moet worden gebruikt. (Opvallend was ook dat er op 23613 3 bytes werden ingeladen, terwijl er maar twee nodig waren).
Wij vullen het bovenstaande stukje met

```
AF           XOR   A  <-----+
DB FE        IN    A,(#FE)  ¦
CB 47        BIT   0,A      ¦
20 F9        JR    NZ,------+
DD 21 00 40  LD    IX,#4000
11 00 BF     LD    DE,#BF00
3E FF        LD    A,#FF
CD C2 04     CALL  #04C2
```

Nu nog even MONS3 inladen, als je dat al niet had gedaan. De tape moet even terug worden gedraaid naar het datablok van het blok wat een tapeerror gaf (R G S). MONS3 in en het volgende invoeren op bijvoorbeeld #FE00. (Je moet MONS3 natuurlijk wel op 40000 plaatsen, die plek is nog vrij).

```
21 00 80     LD    HL,32768
11 00 40     LD    DE,16384
01 3E 1C     LD    BC,7230
ED B0        LDIR
DD 21 00 00  LD    IX,#0000
11 FF FF     LD    DE,#FFFF
3E FF        LD    A,#FF
37           SCF
```

```
CD 56 05    CALL  #0556
C3 00 5B    JP    #5B00
```

Dan hoef je alleen nog de code op #FE00 uit te voeren en de tape te starten. Let op dat je het juiste blok er voor hebt, anders kun je weer helemaal opnieuw beginnen!
We zien dan het scherm geplaast worden, daarna gaat de thermometer lopen. Als hij klaar is met laden staat de thermometer ook vol. Dan zetten we de cassetterecorder op saven en drukken een buitenste toets in waarna het spel wordt gesaved. Na het saven kunnen we de Spectrum resetten, en GENS3 inladen. Met behulp hiervan laden we het grote blok dat we net hebben gesaved in en saven het in 3 kleinere gedeeltes.

```
      ORG   #FF00
      ENT   $
      LD    SP,0
      LD    IX,#4000
      LD    DE,#BF00
      LD    A,#FF
      SCF
      CALL  #556
LOOP  LD    IX,#4000
      LD    DE,#1B00
      CALL  SAVE
      LD    IX,#5B00
      LD    DE,#B000-#5B00
      CALL  SAVE
      LD    IX,#B000
      LD    DE,#4F00
      CALL  SAVE
      JR    LOOP
SAVE  DI
SAVE2 XOR   A
      IN    A,(#FE)
      BIT   0,A
      JR    NZ,SAVE2
      LD    A,#FF
      CALL  #4C2
      RET
```

Even de code uitvoeren (#FF00=65280) en de drie blokken zo saven dat we e makkelijk een header voor aan kunnen zetten.

De laatste instructie die werd uitgevoerd was RST #8 dus moeten we het blok inladen dat 23613/23614 bevat. Als we dat dan doen en we bekijken dan de inhoud van 23613/23614 dan zien we dat deze #1F45 bevatten. We laden dus het scherm in op een veilige plek en kijken op #451F. Deze bevat #5A,#EF. Er wordt dus naar #EF5A gesprongen. Dus laden we dat stuk in en kijken op #EF5A. Daar staat het volgende

```
LD    SP,#FFFE
LD    HL,#5900     Hier wordt het beginscherm opgebouwd.
XOR   A            Attributes van het middelste deel worden zwart
LD    (HL),A
INC   L
JR    NZ,#EE6D
LD    HL,#CDC8
LD    (#4800),HL
LD     A,(#5C3F)   Hier onze 3e byte van het kleine
                   codeblokje, zie voorheen.
LD    HL,#F300     Plaats pixels op het bovenste deel
LD    DE,#4000
LD    BC,#0600
LDIR
LD    H,#F3        Plaats pixels op het onderste deel
LD    B,#08
LD    D,#50
LDIR
INC   B            Attributes bovenste deel
LDIR
DEC   H            Attributes onderste deel
INC   B
INC   D
LDIR
LD    HL,#4800
LD    (#4802),HL   Nu staat op #4800 RET Z
                                  CALL #4800 dus hangup als
CP    #63                                  Z=0
LD    A,#3B
LD    I,A
CA    #4800
LD    A,(IY+0)
```

```
CP    #19
CALL  #4800
LD    SP,#5CC0
JP    #AFC8       Startadres.
```

Even op #AFC8 kijken, daar zien we dat de stackpointer wordt geladen, interrupt mode 2 wordt gezet etc. Het spel begint dus hier. Wat ons op moet vallen is

```
LD    A,#39
LD    I,A
IM    2
```

Op #39FF/#3A00 moet dan het adres staan waar de processor kijkt om IM2 uit te voeren. Daar staat #FFFF. Op #FFFF stond nog steeds #C350 gepushed, zie voorheen. Dus dat moet daar altijd blijven staan.
Nog even het blok inladen waar het BASIC stuk zou moeten staan. Daar vinden we dan

```
CLEAR 24511:LOAD""CODE:LOAD""CODE16384:RANDOMIZE USR 65280
```

We weten hieruit dat het spel loopt van 24512 tot en met 65279. Over blijft maken van de gekraakte versie door het maken van een eigen BASIC loader, het plaatsen van CRACKED BY DR.WHO ALIAS RAM in de tekening en het samenvoegen van de twee overige blokken tot 1 blok van 24512 tot 65280.
In de loader moet dan POKE 65535,80:POKE 65534,195 staan voor IM2.

Kraken les 4:
Het kraken van Doomdarks revenge.
Net zoals bij alle andere spelen moet hier ook eerst de BASICloader worden onderzocht. Dus eerst even een nonautostart versie van maken en even op de bekende manier onderzoeken (les 2).
Daar komt dan het volgende uit:
Naam revenge
Line 0
lang 483 bytes

0 REM ... HEY! LOOK BILL ANOTHER PIRATE .... POWER LOAD 48 TAG 1984 BY INCENTIVE SOFTWARE...
10 ... POKE 23693,56:POKE 23624,63:CLEAR 57999:POKE 23659,0:FOR N=30 TO 36:BEEP .075,N:NEXT N:RANDOMIZE USR 24129:RANDOMIZE USR 0
24129=#5E41 (Tip zet Mons3 op 50000 neer)
Daar staat:

```
DI
LD    HL,#0000
ADD   HL,SP
LD    (#5DC8),HL
SP    SP,#5E84
LD    H,#5E
PUSH  HL
LD    HL,#5E57
JP    (HL) (DUS NAAR #5E57)
```

Op #5E57 staat:

```
LD    A,#01
LD    (#5E82),A
POP   HL
PUSH  HL
POP   DE
JP    NC,#0000
RET  (Dus naar #5E65)
```

Op #5E65 staat:

```
POP   BC
LD    A,(HL)<+
NEG          |
INC   HL     |
DJNZ  -------+
```

```
POP   HL
LD    (#5E67),HL
POP   BC
LD    A,#C9
LD    (#5E69),A
PUSH  DE
POP   HL   <-------Dit is dus een mooie plek voor een breakpoint
RET
```

Dus met Mons 3 het eerste breakpoint zetten op #5E7D. Dan PC op #5E41 zetten, de memorypointer op #5E41 zetten en Sym K doen. Altijd als we Sym K in Mons3 doen moet de meorypointer gelijk zijn aan de inhoud van PC! zien dan dat #5E66 op de stack staat en dat er dus weer in dezelfde lus wordt gesprongen die er nu als volgt uit ziet:

```
#5E66:
LD    A,(HL)
RRD
NOP
INC   HL
DJNZ  #5E66
RET   <---Hier kan dus het 2e breakpoint mooi staan. Dus 2e
```

breakpoint op #5E6D en Sym K geven. Deze RET laat naar #5E01 springen, daar staat het volgende:

```
LD    HL,#5FB4
LD    DE,#5FB5
LD    BC,#5DC0
LDIR                Verschuif een stuk van het geheugen 1 plaats.
POP   HL
LD    D,H
LD    E,L
INC   E
POP   BC
LDIR                Verschuif de rest vh geheugen 1 plaats.
LD    B,#1E          Maak hier dus NOP NOP van, anders verschuift
POP   HL              hij Mons3.
LD    A,(HL)<-----+  HL is de eerste keer #5E1D.
XO    #A3         !  Dus een aantal keren singlesteppen, er
LD    (HL),A      !  is nml geen ruimte voor een breakpoint.
INC   HL          !  Bijvoorbeeld tm B=#19.
DJNZ    ----------+
..... <-----------#5E1D
```

Dan kan op #5E1D een breakpoint worden geplaatst en er kan verder worden gegaan met Sym K. Na het doorlopen daarvan zien we op #5E1D het volgende staan:

```
#5E1D
POP   HL
LD    (#5DF1),HL
POP   HL
LD    (#5DF4),HL
SCF
LD    A,#07
CALL  #5DEF
JP    NC,#0001
LD    HL,#9C40
LD    B,#ff
CALL  #5E66
LD    B,#FF
CALL  #5E66
DI
RET   <---#5E3C
```

Met op #5DEF het volgende:

```
LD    IX,#9C40
LD    DE,#0190
INC   D
EX    AF,AF'
DEC   D
LD    A,#0F
OUT   (#FE),A
CALL  #0562
RET
```

Duidelijk is dus dat we een breakpoint moeten zetten op de RET van #5E3C. Dat doen we dan waarna we vervolgen met Sym K en waarna we de band starten om het volgende stuk in te laden. Dat volgende stuk komt op #9C40 te staan. kijken we op #9C40 waar staat:

```
LD    HL,#9C52
LD    BC,#0190
LD    D,#5E
LD    A,(HL) <---#9C48
XOR   D
LD    (HL),A
INC   HL
DEC   BC
LD    A,B
OR    C
JP    NZ,#9C48
..... <-------#9C5D
```

Duidelijk is dat dit een op zichzelf staand stuk is dus saven we dit stuk even. Als het dan verkeerd gaat hoeven we alleen dit stukje opnieuw in te laden en hoeven we niet opnieuw door het BASIC heen te gaan. Dus voeren we in op #5B00:

```
DD 21 40 9C LD IX,#9C40
11 90 01    LD DE,#0190
3E FF       LD A,#FF
CD 56 05    CALL #0556
                    <--- Zet dan hier een breakpoint.
```

Even dit stuk executeren en we saven het stukje op 40000.
We kunnen natuurlijk ook naar het BASIC terugkeren en gewoon SAVE "STUKJE"CODE 40000,400 invoeren waarmee we dit stukje ook saven.
Goed, even terug naar #9C40. Op #9C5D wordt de code aangepast, dus zullen we eerst enkele keren moeten singlesteppen. We singlesteppen totdat HL=#9C56, dan weten we dat tot #9C56 alles al is aangepast. Dan zetten we een breakpoint op #9C52 en gaan vanaf de plek waar we waren verder met Sym K.
Op #9C52 staat dan:

```
LD    HL,#9C63
LD    DE,#B05D
LD    BC,#0190
LDIR
LD    SP,#5CCA
JP    #B05D
....
```

Dit voeren we uit en dan kijken we op #B05D:

```
LD    A,#FF
LD    DE,#1C94
LD    IX,#4000
CALL  #B076       Laadt het 1e stuk in.
LD    DE,#A335   <-----#B069
LD    IX,#5CCB
CALL  #B0C5       Laadt het 2e stuk in.
JP    #0000  <----#B073
....
```

We laden het 1e stuk dan even in en zetten op #B069 het volgende: C3 00 FF JP #FF00.
Op #FF00 plaatsen we dan het volgende:

```
F3              DI
AF              XOR   A
DB FE           IN    A,(#FE)
CB 47           BIT   0,A
20 F9           JR    NZ,...
DD 21 00 40     LD    IX,#4000
11 94 1C        LD    DE,#1C94
3E FF           LD    A,#FF
CD C2 04        CALL  #04C2
                <-----Plaats hier een breakpoint.
```

Even het eerste stuk inladen door op #B050 een Sym K te geven. Als het eerste stuk is ingeladen wacht de computer op een toetsdruk van een van de buitenste toetsen om te gaan saven. We saven dan even het stukje en herstellen het stuk op #B069. Op #5B00 moeten wij dan onze eigen save routine plaatsen waarmee we het 2e deel kunnen saven. Let op, deze loader overschrijft zichzelf, dus de JP #0000 wordt ook overschreven.

Op #5B00 zetten we:

```
3E FF           LD    A,#FF
11 94 1C        LD    DE,#1C94
DD 21 00 00     LD    IX,#0000
CD 76 B0        CALL  #B076
11 35 A3        LD    DE,#A335
DD 21 CB 5C     LD    IX,#5CCB
CD C5 B0        CALL  #B0C5
F3              DI
AF              3E FF
DB FE           IN    A,(#FE)
CB 47           BIT   0,A
20 F9           JR    NZ,....
DD 21 CB 5C     LD    IX,#5CCB
11 35 A3        LD    DE,#A335
3E FF           LD    A,#FF
CD C2 04        CALL  #04C2
```

We draaien de tape terug naar het begin van de fastloader en starten de code op #5B00 met Sym K.(Vergeet PC en de memerypointer niet te zetten). Start dan de tape en laad de code in. De computer wacht daar op een toetsdruk van een buitenste toets alvorens te gaan saven. Dus hoeven we alleen de tape te verwisselen en te gaan saven. Als we hebben gesaved resetten we de Spectrum even.
Dan maken we in GENS3 het volgende programma:

```
      ORG #5B00
      ENT$
      LD    SP,#5C00
      LD    IX,#5CCB
      LD    DE,#A335
      LD    A,#FF
      SCF
      CALL  #556
      LD    IX,#5CCB
      LD    DE,#5000
      CALL  SAVE
      LD    IX,#5CCB+#5000
      LD    DE,#A335-#5000
      CALL  SAVE
```

```
SAVE  DI
      XOR   A
      IN    A,(#FE)
      BIT   0,A
      JR    NZ,SAVE
      LD    A,#FF
      CALL  #4C2
      RET
```

Met het bovenstaande programmaatje laden we het net gesavede grote blok in en saven we het als 2 kleinere blokjes. Dit laden en saven voeren we even uit.
Laden we dan met behulp van:

```
DD 21 CB AC LD    IX,#ACCB
11 35 53    LD    DE,#5335
3E FF       LD    A,#FF
37          SCF
CD 56 05    CALL  #556
```

even het 2e kleine blok in dan kunnen we met behulp van MONS3 op #B05D kijken. Daar staat dan:

```
#B05D
LD    A,#FF
LD    DE,#1C94
LD    IX,#4000
CALL  #B076
LD    DE,#A335
LD    IX,#5CCB
CALL  #B0C5
JP    #B117       We kijken dus op #B117.
Daar staat:
#B117
LD    A,#00       Hier worden allerlei controleroutines
OUT   (#FE),A     uigevoerd. Deze kunnen dus worden over-
CALL  #B14A       geslagen. NOP van maken dus.
LD    HL,#B15A
CP    (HL)
LD    (#B122),0
LD    HL,#B133
LD    DE,#5B90
LD    BC,#0064
```

```
LDIR
JP    #5B90
LD    HL,#5CCB
LD    BC,#012C
LDIR
EI
JP    #5E00     <----Dit is duidelijk het startadres.
LD    A,#00     <----#B14A
LD    HL,#4000
LD    DE,#5C94
XOR   (HL)      <----#B152
INC   HL
SBC   HL,DE
ADD   HL,DE
JR    C,#B152
RET
```

We passen dus even het stuk hierboven aan en saven dit stuk weer op de volgende manier. SAVE "STUK2" CODE 44235,20480 We starten de tape pas als de header is geweest.

Bij deze is het spel dus gekraakt, een basicloader maken:

```
10 CLEAR 29999:LOAD ""CODE:RANDOMIZE USR 45335
```

De machinetaalloader:

```
DD 21 00 40 LD    IX,#4000
11 94 1C    LD    DE,#1C94
3E FF       LD    A,#FF
37          SCF
CD 56 05    CALL  #556
DD 21 CB 5C LD    IX,#5CCB
11 00 50    LD    DE,#5000
3E FF       LD    A,#FF
37          SCF
CD 56 05    CALL  #556
DD 21 CB AC LD    IX,#ACCB
11 35 53    LD    DE,#5335
3E FF       LD    A,#FF
37          SCF
CD 56 05    CALL  #556
C3 00 5E    JP    #5E00
```

Deze loader saven plaatsen we op B117, we saven het blok opnieuw maar dan met SAVE "DR WHO"CODE 44235,21301 en klaar is kees. We konden natuurlijk ook het scherm wel aanpassen maar aannemende dat iedereen zo'n kleinigheidje wel kan noem ik dat hier maar niet. Als we nog even in de code hadden rond gekeken was ons opgevallen dat op #AFC8 de fastsave routine nog staat.Deze kunnen we natuurlijk ook voor ons zelf gebruiken of we kunnen een fastsave versie van Doomdarks Revenge maken.
Gelieve dan de #556 te vervangen door #B076 en de drie blokken te saven door ipv #4C2 #B14A te gebruiken. Ook kun je hem saven als origineel zijnde en gewoon #AFC8 laten uitvoeren, let dan wel op dat je #AFEB even van RST 0 afhaalt en er C9=RET van maakt. Copieer dan even de fastloader en gebruik deze als loader bij het inladen van het grote blok. De fastloader loopt van #B076 tm #B116.

RAM alias DR. WHO

AFZ: DE SPECTRUM
VUURSE DREEF 75
3739 KS HOLLANDSCHE RADING

PORT BETAALD

HILVERSUM

DRUKWERK

AAN :