KUN JE NOG POKEN POOK DAN MEE

POKEN ONDER DBASE

Inleiding

Hoe mooi zou het niet zijn als je onder DBASE II gebruik kon maken van de instructies POKE, PEEK en CALL. Dat was de hartekreet van een verwoed gebruiker en auteur van het DBASE II programma. Ik las dat in een gerenomeerd MSX-blad. Kennelijk zijn gebruiksaanwijzingen ook niet alles. Of zou het kunnen dat voor het gebruik van het DBASE II programma onder MSXDOS geen gebruiksaanwijzing beschikbaar is? In ieder geval opent het gebruik van instructies onder DBASE II die rechtstreeks het geheugen manipuleren onvermoede mogelijkheden.
Wat te denken van het poken van de hoofdlettertoets (het kan echt !) of van het poken van een machinetaalprogramma in het geheugen dat met de CALL-instructie wordt gestart en de status van de printer vaststelt (ook dat werkt perfect !). Over de talloze andere mogelijkheden, zoals het poken van de datum uit de computer in een DBASE-programma, nog maar te zwijgen.
Laat ik de vele gebruikers/programmeurs van dit prima programma de creatieve vrijheden die een goede programmeur eigen zijn.
In het navolgende ga ik alleen in op het gebruik van de instructies, de vrijheden die hierbij gepermitteerd zijn en de beperkingen die door de syntax van de instructies worden opgelegd. Voor de gerechtaarde DBASE II gebruiker lijkt mij dit ook voldoende.
Een waarschuwing is hier overigens wel op z'n plaats. Het is absoluut noodzakelijk dat men kennis heeft over de geheugenstructuur en het inwendige van de computer. Men dient tenminste te weten welke geheugenadressen door het systeem en de geladen programma's worden gebruikt.
Als naslagwerken beveel ik van harte aan het "MSX 2 zakboekje" van uitgeverij STARK-TEXEL en de "Software-vraagbaak dBASE II" van SYBEX. In het laatste boekje staat ook de syntax van de te beschrijven instructies. Verder merk ik nog op dat voor het testen van de instructies gebruik is gemaakt van een NMS 8245 computer. Niet dat voor het testen van de instructies het type computer belangrijk is maar wel de geheugenstructuur en de adressen van de clock-chip.

De instructie PEEK

De PEEK-instructie is uit het rijtje instructies die het geheugen manipuleren de makkelijkste instructie.
Op het gevaar af dat ik een open deur intrap meld ik dat het met deze instructie mogelijk is een waarde die in een geheugenadres is opgeslagen in te lezen en te gebruiken in een DBASE II-programma.
In de software-vraagbaak wordt de syntax van de instructie als volgt omschreven:

PEEK (<adres>)

In de beschrijving van de instructie wordt verder opgemerkt dat het PEEK-adres decimaal moet worden opgegeven. Dat wil zeggen dat het te lezen geheugenadres decimaal met deze instructie moet worden meegegeven. Het resultaat zal ook decimaal te beschikking komen. Laat ik het een en ander met een kort DBASE II-programma verduidelijken:

```
STORE 0 TO CAPS
STORE PEEK (64683) TO CAPS
DO WHILE CAPS > 0
    @ 3,20 SAY "CAPS-LOCK UITZETTEN !!!!!!!"
    STORE PEEK (64683) TO CAPS
    @ 3,0
ENDDO
```

Wat doet dit programma.
Het test de status van de CAPS-LOCK toets. Eerst wordt in de geheugenvariabele CAPS de waarde "0" gezet. Vervolgens wordt door de instructie STORE PEEK (64683) TO CAPS de momentele waarde van de geheugenvariabele CAPS in geheugenlocatie 64683 geschreven. Daarna wordt in een DO WHILE-lus gekeken of de waarde die is ingelezen groter is dan nul. Is dat het geval, en dat is het geval indien de CAPS-LOCK lamp brand, dan blijft de lus van kracht en zal de mededeling CAPS-LOCK UITZETTEN in een aan/uit mode op het scherm verschijnen. Bedien je nu de CAPS-LOCK toets dan is de voorwaarde in de

DO-WHILE lus niet meer van kracht en gaat het programma verder.
In dit simpele programma komt dan de beroemde DBASE II-prompt weer te voorschijn.
Voorwaar geen wereldschokkende gebeurtenis maar wel handig in het gebruik.
Verder wordt nog opgemerkt dat als de geheugenwaarde in de geheugenvariabele is gePEEKd verdere bewerkingen mogelijk zijn. Hierbij kan men denken aan rekenkundige bewerkingen enz.

De instructie POKE

Nu wordt het menens. Dit is de instructie die kennelijk de meeste problemen veroorzaakt. Overigens verbaas ik mij daar niet meer over nu ik weet dat de beschrijving van de syntax van de instructie in de software-vraagbaak van SYBEX niet juist is.
In de software-vraagbaak wordt de syntax van de instructie als volgt beschreven:

POKE <adres> <byte-lijst>

Verder wordt opgemerkt dat het <adres> en byte-waarden decimaal moeten worden opgegeven en dat de byte-waarden vanaf het <adres> in het werkgeheugen van de computer worden gePOKEd.
Toen ik deze instructie wat nader bestudeerde viel mij meteen al op dat men in de toelichting/opmerking spreekt over adressen. Meervoud dus. Dat verbaasde mij des te meer omdat ook over een byte-lijst wordt gesproken. Zou het dan zo zijn dat voor elk te poken byte een aparte opdracht gegeven moet worden? Nee dus. Het adres moet alleen, in afwijking overigens van de andere DBASE II-instructies, geplaatst zijn in een geheugenvariabele.
Hier kan derhalve niet rechtstreeks een decimaal adres worden opgegeven!
Het voorbeeld in de software-vraagbaak POKE 42072 27,153 enz werkt daarom niet.
Plaatst men het eerste geheugenadres in b.v. de geheugenvariabele TEST door STORE 42072 TO TEST dan werkt het wel! In dit geval moet het als volgt worden uitgevoerd:

POKE TEST,27,153, enz.

Let goed op tussen TEST en 27 staat een komma. Ook deze komma is van groot belang en wordt niet in de beschrijving vermeld.

In het nu volgende beschrijf ik in MNEMONIC een kort machinetaalprogramma en zal daarna laten zien hoe dit wordt gePOKEd en gestart.
Het machinetaalprogramma maakt gebruik van MSXDOS-aanroepen en zet de MSX-characterset op het scherm. Na een return keert het programma weer terug in DBASE II. Een weinig verheffend programma maar daar gaat het nu even niet om. Als demonstratie voor het gebruik van de instructie is dat meer dan voldoende.

MNEMONIC	HEXA	DECIMAAL	OPMERKINGEN
equ MSXDOS,	$05		; BDOS-call
equ CONOUT,	$02		; CONsole OUTput schrijven naar scherm
equ CONSTAT,	$Ob		; CONsole STATus
equ DOEL,	$d000	53248	; start adres programma in geheugen
push af	$f5	245	
push bc	$c5	197	
push de	$d5	213	
LOOP:			
id e,	$20		; 1e 30 characters
LOOP1:			
id c, CONOUT	0e 02	14 02	
push de	d1	209	
call MSXDOS	cd 05 00	205b05 00	; character to screen
pop de	d1	209	
inc e	1c	28	; volgend character
ld a,e	7b	123	
cp 7f	fe 7f	254 127	; einde character code
jr nz, LOOP1	20 f3	32 243	
ld c, CONSTAT	0e 0b	14 11	
call MSXDOS	cd 05 00	205 05 00	Toets gedrukt?
or a	b7	183	
jr z, LOOP	28 e9	40 233	

pop de	d1	209	
pop bc	c1	193	
pop af	f1	175	
ret	c9	201	terugkeer naar DBASE II

Het in het geheugen van de computer poken van het programma gaat nu als volgt. Start het DBASE II programma en creëer een nieuw commandofile met de naam TEST.CMD en schrijf het volgende programma onder DBASE:

```
**********************************TEST. CMD*****************************************
*
*                        MACHINETAAL TEST ONDER DBASE
*
*************************************************************************************
*  HET DEMONSTRATIE PROGRAMMA ZET EEN SET ASCII-CHARACTERS VAN DE MSX OP HET
*  BEELDSCHERM. HET PROGRAMMA IS TE BEEINDIGEN MET EEN RETURN.
*

STORE 53248 TO ADRES
POKE ADRES, 245,197,213,30,32,14,02,2131.205,05,00,209,28,123, POKE ADRES+l41254,l27,32,243,l4,11,
205,05,00,183,40,233,225, POKE ADRES+27,193,241,201

SET CALL TO ADRES
CALL
RETURN
*************************************************************************************
```

Als u het machinetaalprogramma goed hebt bestudeerd zal het u op gevallen zijn dat het begint met het veilig stellen van de registers van de Z80. Dit is absoluut noodzakelijk om geen vast gelopen programma te krijgen. Kennelijk kan het DBASE II-programma daar niet goed tegen. Verder valt het u misschien op dat ik het hl-register niet heb gesaved. Dit is niet noodzakelijk maar ook niet handig omdat het in samenhang met de CALL-instructie een waarde kan meegeven aan het DBASE II-programma! Dit zal in het vervolg nog worden gedemonstreerd. Verder spreekt het machinetaalprogramma voor zich zelf.
Laten we een gedetailleerde bespreking maar voor wat het is en concentreren we ons op het DBASE II-deel.
Het start met het vastleggen van het startadres in het geheugen door STORE 53428 TO ADRES. Hexadecimaal is dit d000. Daarna poken we de data vanaf het startadres in het geheugen. Kan de data niet in z'n geheel op een regel dan starten we een nieuwe instructieregel door bij het startadres het aantal te poken bytes op te tellen. Simpel niet waar! Je moet er maar opkomen.
Let op, het startadres is het nuladres en telt wel mee in de offset!! Daarna is het DBASE II-programma minder interessant.
Met de instructie SET CALL TO ADRES wordt de CALL-aanroep voorbereid. Geef je nu een CALL-instructie en een return dan zal het machinetaalprogramma, als alles is goed gegaan, starten door op het scherm de character-set weer te geven.
Hiermee hebben we de instructie POKE ook klein gekregen. Het is als of je een tranquillizer hebt gebruikt. Kijken of we nu ook nog een waarde vanuit het DBASE II-programma mee kunnen geven aan het machinetaalprogramma. Dat geeft helemaal onvermoede mogelijkheden. Wat te denken van het besturen van de printer. Italic, proportioneel, dubbele breedte enz. behoren dan tot de mogelijkheden.

De instructie CALL
De instructie CALL zonder toevoegingen hebben we al leren kennen. Hiermee verder gaan heeft weinig zin. Het meegeven van een variabele daar gaat het nu om.
We maken wederom gebruik van de software-vraagbaak voor uitleg van de syntax.
De CALL-instructie staat er als volgt beschreven:

CALL [<variabele>]

Hierbij wordt opgemerkt:

- Met <variabele> kan een parameter aan de subroutine worden meegegeven.
- Bij een alfanumerieke variabele bevat register "hl adres" van het eerste byte.

Analyseren we wat er staat dan stoten we op onduidelijkheden. Wat wordt er bedoeld met "hl adres"? Wordt hiermee bedoeld dat het hl-register het adres aanwijst waar DBASE II de meegegeven variabele opslaat? Als dat waar is dan ontstaat er een conflict met command.com, immers men kan volgens de syntax van de instructie niet meer meegeven dan een acht bit breed woord. Geef je een alfanumerieke of een numerieke variabele mee dan weigert DBASE II medewerking onder vermelding van een syntax fout of nog erger het laat geen enkele reactie zien.
Plaats je een numerieke variabele in een geheugenvariabele b.v. de geheugenvariabele "P" en geef je de geheugenvariabele mee met de CALL-instructie als CALL P dan werkt DBASE plotseling wel mee! Hoewel, plotseling worden alle variabelen in DBASE II numeriek (decimaal) verwerkt. Het zou verwondering wekken als nu in eens een alfanumerieke variabele moet worden gebruikt. Dit is dan ook fout weer gegeven in de softwarevraagbaak. Evenzo als het fout is dat de numerieke waarde in de directe mode, dat wil zeggen direct achter de CALL-instructie, kan worden meegegeven. De variabele moet numeriek in een geheugenvariabele worden meegegeven. Maar pas op, het meegeven gaat op wel een heel bijzondere wijze! **Niet het getal als waarde is belangrijk maar de rangorde, de positie, speelt een grote rol.** Kennelijk om onmogelijke grote getallen te voorkomen heeft men daarbij gebruik gemaakt van quotes, haken, negatieve getallen enz. Een volledig inzicht in de werking heb ik nog niet. Echter het nu volgende is voorlopig voldoende voor verdere experimenten. Een aantal voorbeelden:

GEHEUGEN VARIABELE	NUMERIEKEWAARDE	HEXADECIMALE WAARDE
P	1	40
P	10	41
P	-1	c0
P	-10	c1
P	"1"	00
P	"10"	01
BEEP	"1000000"	07
LINE FEED	"1000000000"	0a

Als de geheugenvariabele "P" een waarde <-1> heeft dan komt dit overeen met de hexadecimale waarde "CO". Een Waarde van <-10) komt overeen met de waarde "Cl", een waarde van <-100> komt overeen met de waarde "C2" enz.
Laat men het min teken. weg dan wordt de waarde <1> gelijk aan "40" en de waarde <10> aan "41" enz. Voorwaar een vreemde manier om een parameter aan een machinetaal-programma mee te geven.

Om de CALL-instructie nader te analyseren is een eenvoudig programma in machinetaal geschreven dat wordt gestart onder DBASE II. Hiermee wordt het experimenteren met deze instructie gekanaliseerd en zijn de resultaten daarvan goed te bestuderen.
Het programma neemt via een CALL-instructie een getal over in het hl-register, zet de waarde in grafische vorm op het scherm en wacht vervolgens op het drukken van de returntoets om daarna terug te keren in DBASE II. Ook dit programma maakt weer gebruik van MSXDOS-aanroepen.

Na de noodzakelijke voorbereidingen, zoals het pushen van de registers, heeft plaatsgevonden wordt de inhoud van het hl-register in het e-register gelezen. Daarna wordt in het c-register de waarde van CONOUT geplaatst waarna MSXDOS wordt aangeroepen. Voor de systeem aanroep wordt register "de" nog gesaved. Dit is noodzakelijk omdat bleek dat het systeemprogramma het de-register gebruikt. De "LOOP" is in machinetaalprogramma hiervoor reeds beschreven.

```
        MNEMONIC        HEXA        DECIMAAL            OPMERKINGEN
equ MSXDOS,             $05                         ; Aanroep MSXDOS
equ CONOUT,             $02                         ; CONsole OUTput
equ CONSTAT,            $Ob                         ; CONsole STATus

org                     $d000       53248
```

```
push af          $f5       245
push bc          $c5       197
push de          $d5       213
nop              00
id e, (hl)       5e        094          ;
id c, CONOUT     0e 02     14 02
push de          d1        209
call MSXDOS      cd 05 00  205 005 00   ; character to screen
pop de           d1        209
LOOP:
 ld c, CONSTAT   0e 0b     14 11
call MSXDOS      cd 05 00  205 05 00    Toets gedrukt?
or a             b7        183
jr z, LOOP       28 e9     40 233
nop              00
pop de           d1        209
pop bc           c1        193
pop af           f1        175
ret              c9        201          terugkeer naar DBASE II
```

Dit programma poken we weer onder DBASE in het geheugen.

****

```
STORE 53248 TO ADRES

POKE ADRES,245,197,213,00,94,14,02,213,205,05,00, 209,14,11,205,05,00
POKE ADRES+17,183,40,248,00,209,193,241,201

SET CALL TO ADRES

RETURN
```

****

Als dit programma in het geheugen van de computer staat dan kan het worden gestart met CALL <variabele> waarbij variabele een geheugen variabele moet zijn die een numerieke waarde bevat.
Met de grafische representatie van de alfanumerieke waarde en het handboek van de MSX kan nu worden nagegaan welke hexadecimale waarde behoort bij de numerieke waarde die is meegegeven met de CALL instructie. Op deze wijze is het mij gelukt een deel van de syntax van de CALL instructie te doorgronden. Het zal duidelijk zijn dat voor de hexadecimale waarden kleiner dan 20 geen grafische representatie te zien valt op het scherm. Dit zijn de waarden die binnen de ASCII-characterset worden gebruikt voor besturingsdoeleinden. Zo zal de bovenomschreven geheugenvariabele BEEP een beeptoon ten gehore brengen.

Hiermee is dan mijn bijdrage aan het vergroten van de DBASE mogelijkheden voorlopig uitgeput. Hopelijk heb ik de vele gebruikers van het DBASE-programma voedsel verschaft voor vele uurtjes programmeer plezier en kan ik nog eens genieten van hun experimenten.

H.Reudink PA3BYB