Miner 2049'er

Ärgert es Sie auch, daß Sie beim Miner 2049'er nie die letzten Bilder zu sehen bekommen ? Der Ärger ist unnötig, denn es gibt einen ganz einfachen Weg, um bis in die letzten Spielstufen zu gelangen. Sie brauchen nur die Leertaste oder wahlweise auch den Feuerknopf am Joystick konstant gedrückt halten. Nach einiger Zeit wird die aktuelle Spielstufe übersprungen, und man gelangt von selbst ins folgende Bild.

(Armin Robl)

HilfsPOKE für »Gangster«

Besitzer des Spieles »Gangster« können aufatmen, denn ab sofort ist es möglich, auch in die letzten der 30 Bilder zu gelangen. Mit »POKE 5989,58« hat man statt der mageren fünf Spielfiguren nun sage und schreibe 128 (!) Gangster zur Verfügung. Sollte auch damit noch jemand Probleme haben, so ist ihm dringend zu raten, das Spiel zu wechseln.

Die merkwürdigen Zeichen, die nach diesem POKE in der Anzeige der verfügbaren Gangster erscheinen, haben auf die Spielbarkeit keinen Einfluß und können ignoriert werden.

Da die Gefahr besteht, daß man die Freude am Spiel verliert, sobald man alle Schwierigkeitsgrade überwunden hat, sollte man ab und zu auch regulär spielen. Es ist dann meist verblüffend, wie weit man auch ohne eine starke Gangster-Truppe kommt.

(Frank Herrmann)

»Frantic Freddy « als Trainingsversion

Auf sehr einfache Art und Weise kommt man zu einer Trainingsversion von »Frantic Freddy«: Einfach das Spiel laden, dann »POKE 34535,24« eingeben und schließlich mit »RUN« starten. Man kann sich jetzt ohne Gefahr durch das ganze Spiel bewegen.

(Volker Bellendorf)

Die Lösung von »Aztec-Tomb Part 1«

Ich löse mit Vorliebe Adventures und möchte mit dem Lösungsweg für »Aztec-Tomb Part 1« allen verzweifelten Abenteurern auf die Sprünge helfen, die bei diesem Spiel an irgendeiner Stelle nicht mehr weiterkommen. Hier ist die kürzeste Lösung:

go ladder, take chest, go down, go south, look hall, take jar, go west, look bed, go trapdoor, take cloak, wear cloak, look cellar, take key, open chest, drop chest, drop key, take sword, take rope, go up, open draw, look draw, take key, go east, open door, drop key, go door, climb building, take wood, go down, go south, drop wood, go east, look pool, catch fish, go west, go bridge, go south, take mouse, go north, go west, remove cloak, throw cloak, go gate, take cloak, look fish, fill jar, empty jar, fill jar, empty jar, climb beanstork, drop mouse, go path, go valley, go south, throw rope, climb rope, give cloak, take box, open box, take map, look map, drop box, go down, go north, go east, go harbour, go boat, look boat, go cabin, take torch, go hatch, cross north, go island, light torch, go hole, take jacket, wear jacket, go up, unlit torch, go boat, cross north, cross east, cross south, jump over, swim, go beach, go forest, go north, climb statue, take diamond, go down, go east, look wall, insert diamond, light torch, go passage.

Damit haben Sie den ersten Teil von Aztek-Tomb erfolgreich hinter sich gebracht. Ich hoffe, daß Ihnen diese Lösung in der einen oder anderen verfahrenen Situation weiterhelfen kann.

(Wolfgang Habich)

Hex-ereien: undefinierte Opcodes beim 6502

Im folgenden Artikel sollen einmal diejenigen Hex-Zahlen unter die Lupe genommen werden, die ein 6502-Disassembler normalerweise nur mit einem höhnischen Fragezeichen quittiert.

Sicherlich ist Ihnen, sofern Sie sich mit Maschinensprache befassen, schon aufgefallen, daß der bekannte Befehlssatz des 6502-Mikroprozessors nicht alle 256 Bitkombinationen eines Bytes ausnutzt. Es existieren neben den bekannten, in jedem Assembler-Handbuch dokumentierten 6502-Befehlen noch eine ganze Reihe undefinierter Opcodes. Der Begriff »undefiniert« bedeutet dabei nur soviel wie »nicht dokumentiert«, denn diese Opcodes haben in vielen Fällen durchaus eine sinnvolle Wirkung.

Da viele Commodore-Systeme mit dem 6502 arbeiten, habe ich mir einmal die Mühe gemacht, die für diese CPU offiziell nicht implementierten Hex-Zahlen, die sich immer wieder vereinzelt zwischen die bekannten 6502-Opcodes einschieben, auf ihre Wirkung hin zu untersuchen. Dabei hat sich manch unscheinbare Hex-Zahl als recht brauchbar entpuppt. Nachfolgende Erkenntnisse habe ich auf meinem C 64 gesammelt, der ja mit einem 6510 bestückt ist, die Ergebnisse sind aber auch auf den 6502 übertragbar.

Die untersuchten Bit-Kombinationen lassen sich in vier Befehlskategorien unterteilen:

Gruppe 1: Diese Befehle führen den Prozessor zu einem Systemabsturz, Sie können dann den Computer nur noch mit einem RESET aus seinem Dornröschenschlaf befreien (oder das Gerät ausschalten). Alle »Absturzbefehle« sind dadurch kenntlich, daß ihr niederwertiges Nibble (1 Nibble = 4 Bit) »2« ist (Beispiel: $02,$12). Aber: Nicht alle Hexzahlen mit niederwertigem Nibble »2« führen zum Absturz (Beispiel: $A2 entspricht in Assemblersprache »LDX«).

Gruppe 2: Diese Befehlsgruppe bewirkt rein gar nichts, es gibt sie in 3-, 2- und 1-Byte-Ausführungen, zu den letztgenannten gehört ja auch unser gutes altes »NOP«.

Die Befehle der Gruppen 1 und 2 sind in Tabelle 1 zusammengestellt.

Gruppe 3: Nun wird's interessant. Der Prozessor führt zwei »offizielle« Befehle unmittelbar hintereinander aus. Diese verwenden die gleiche Adressierungsart. Ein Beispiel: Die Hex-Zahlenkombination E7 DD führt die beiden Assemblerbefehle »INC DD;SBC DD« (Zeropageadressierung) aus, sie bewerkstelligt dieses in 2 Byte, wozu der »offizielle« Befehlssatz 4 Byte benötigte. Es sind alle vom »STA«-Befehl her bekannten Adressierungsarten auf die Befehlsgruppe 3 anwendbar. Dar-

aus resultiert folgendes Phänomen: Die Hexkombination E3 DD führt die Assemblerbefehle »INC (DD,X)« aus, obwohl es die indirekt-indizierte Adressierung für den »INC«-Befehl eigentlich gar nicht gibt.

Tabelle 2 zeigt alle möglichen Kombinationen. Beispiel: Sie möchten wissen, was geschieht, wenn der 6502 auf die Hex-Zahl C7 trifft. Dazu suchen Sie diese Zahl in der Tabelle 2. Sie steht in der letzten Spalte der Zeile 5. Am linken Zeilenrand finden Sie nun die Befehlskombination, die obersten beiden Zeilen der letzten Spalte geben die Adressierungsart sowie die Befehlslänge in Bytes an. C7 MM würde also dieselbe Wirkung haben, wie die Assemblerbefehle »DEC MM:CMP MM« zusammen.

Die Ausführungszeit der Befehle der dritten Befehlsgruppe ist dieselbe, als wenn nur ein offizieller Opcode der gleichen Adressierungsart ausgeführt würde. E7 DD ist in der Ausführung also genauso schnell wie der Assemblerbefehl »INC DD«.

Als besonders brauchbar würde ich die Befehlskombination »DEC ...: CMP ...« (für Schleifenzwecke) und Rotations/Schiebebefehle in den hierfür sonst nicht vorhandenen Adressierungsarten »(..,X)« und »(..),Y« (für hochauflösende Grafik) bezeichnen.

Gruppe 4: Diese Befehle sind nur sehr schlecht durch bekannte Opcodes zu ersetzen; sie sind in Tabelle 3 beschrieben.

Zum Abschluß noch eine Warnung. Überlegen Sie sich den Gebrauch dieser neuen Befehle gut, denn sie machen jedes noch so gut strukturierte Programm zunichte. Insbesondere ist zu beachten, daß die hier beschriebenen undefinierten Opcodes nicht bei jeder Prozessorversion die gleiche Wirkung haben müssen. Schließlich handelt es sich dabei ja nur um unbeabsichtigte Nebenprodukte bei der Implementierung des 6502-Standard-Befehlssatzes. (Jürgen Urban/ev)

Wirkung	Hex-Zahlen
Absturz	02,12,22,32,42,52,62,72,92,B2,D2,F2
keine (1 Byte)	1A,3A,5A,7A,DA,EA(NOP),FA
keine (2 Byte)	04,14,34,44,54,64,74, 80,82,89,C2,D4,E2,F4
keine (3 Byte)	0C,1C,3C,5C,7C,DC,FC

Tabelle 1. Die Befehle der Gruppen 1 und 2

	(IND,X)	(IND),Y	ABSOLUT	ABSOLUT,X	ABSOLUT,Y	ZEROPAGE,X	ZEROPAGE
	2	2	3	3	3	2	2
ASL:ORA	03	13	0F	1F	1B	17	07
ROL:AND	23	33	2F	3F	3B	37	27
LSR:EOR	43	53	4F	5F	5B	57	47
ROR:ADC	63	73	6F	7F	7B	77	67
DEC:CMP	C3	D3	CF	DF	DB	D7	C7
INC:SBC	E3	F3	EF	FF	FB	F7	E7

Tabelle 2. Die Befehle der Gruppe 3 fassen jeweils zwei Standard-Befehle in einem Opcode zusammen.

Befehl	Beschreibung
0B MM	führt »AND MM« aus und transportiert Negativflag ins Carry
2B MM	wie 0B MM
4B MM	führt »AND #MM : LSR A« aus
6B MM	wenn Dezimalflag =0: führt »AND #MM : ROR A« aus, danach kommt Bit 0 des Akkus ins Carry und das Overflowflag ist eine EXCLUSIV- ODER Verbindung des Bits 5 mit Bit 6 des Akkus. Wenn Dezimalflag =1: Wirkung noch nicht ermittelt.
83 MM	die UND-Verknüpfung zwischen Akku und X-Register wird in (MM,X) gespeichert
87 MM	die UND-Verknüpfung zwischen Akku und X-Register wird in der Zeropage in MM gespeichert
8B MM	führt »TXA : AND #MM« aus
8F MM NN	die UND-Verknüpfung zwischen Akku und X-Register wird in NNMM gespeichert
93 MM	die UND-Verknüpfung zwischen Akku, X-Register und der Summe aus 1 und dem Inhalt der Speicherzelle MM+1 wird in (MM),y gespeichert
97 MM	die UND-Verknüpfung zwischen Akku und X-Register wird in MM,X gespeichert
9B MM NN	die UND-Verknüpfung zwischen Akku und X-Register wird im Stackpointer abgelegt, danach wird die UND-Verknüpfung zwischen Stackpointer und #NN+1 in NNMM,Y gespeichert
9C MM NN	die UND-Verknüpfung zwischen Y-Register und #NN+1 wird in NNMM,X gespeichert
9E MM NN	wie 9C MM NN, nur mit vertauschten Bedeutungen für X- und Y-Register
9F MM NN	die UND-Verknüpfung zwischen Akku, X-Register und #NN+1 wird in NNMM,Y gespeichert
A3 MM	führt »LDA (MM,X):TAX« aus
A7 MM	führt »LDA MM:TAX« aus
AB MM	Wirkung noch nicht ermittelt
AF MM NN	führt »LDA NNMM:TAX« aus
B3 MM	führt »LDA (MM),Y:TAX« aus
B7 MM	führt »LDA MM,Y:TAX« aus
BB MM NN	die UND-Verknüpfung zwischen Stackpointer und der Speicherzelle NNMM,Y wird im X-Register abgelegt, danach wird »TXS:TXA« ausgeführt
BF MM NN	führt »LDA NNMM,Y:TAX« aus
CB MM	die UND-Verknüpfung zwischen X-Register und Akku wird im X-Register abgelegt, danach wird vom X-Register #MM ohne Berücksichtigung des Carry-Flag subtrahiert
EB MM	entspricht dem Assemblerbefehl »SBC #MM«

Tabelle 3. Diese Gruppe enthält Befehle, die teilweise recht komplexe Operationen durchführen. Bei einigen speziellen Kombinationen konnte die genaue Wirkung noch nicht ermittelt werden. Für Hinweise ist die Redaktion jederzeit dankbar.