MEGA65-TEAM

Paul Gardner-Stephen, prof.
(highlander)
Oprichter
Software- en virtuele hardwarearchitect
Woordvoerder en hoofdwetenschapper

Martin Streit
(seriously)
Video- en fotoproductie
Fiscaal en organisatie
Sociale media

Dan Sanderson
(dddaaannn)
Media en documentatie
MEGA65.ROM

Edilbert Kirk, dr.
(Bit Shifter)
MEGA65.ROM
Handleiding en gereedschappen

Gábor Lénárt
(LGB)
Emulator

Farai Aschwanden
(Tayger)
Bestands-hosting en gereedschappen
Financieel adviseur

Falk Rehwagen
(bluewaysw)
GEOS

Robert Steffens
(kibo)
Netwerktechnologie
Jacht op bugs in onze cores

Detlef Hastik
(deft)
Mede-oprichter
Projectaansturing
Marketing en verkoop

Oliver Graf
(lydon)
Releasemanagement
VHDL- en platformverbeteringen

Antti Lukats
(antti-brain)
Host-hardwareontwerp en -productie

Dieter Penner
(doubleflash)
Host-hardwareondersteuning

Mirko H.
(sy2002)
Aanvullende platforms en advies

Gürçe Işıkyıldız
(gurce)
Gereedschappen en verbeteringen

Daniel England
(Mew Pokémon)
Aanvullende code en gereedschappen

Hernán Di Pietro
(indiocolifa)
Aanvullende emulatie en gereedschappen

Roman Standzikowski
(FeralChild)
Open ROMs

Anton Schneider-Michallek
(adtbm)
Presentatie en ondersteuning

Fouten en onvolkomenheden melden

Dit boek wordt voortdurend verbeterd door de MEGA65-gemeenschap. De versie van deze editie is:

```
commit c598652ac7f860872b8be62911f59ad5ad752169
date: Sat Oct 24 17:02:30 2026 +0200
```

We willen dat dit boek zo goed mogelijk wordt. Als u fouten ontdekt, iets mist of graag meer informatie wilt, meld dit dan via het issue-tracker van de MEGA65 User's Guide:

https://github.com/mega65/mega65-user-guide/issues

U kunt daar ook controleren of iemand anders al een vergelijkbaar probleem heeft gemeld, terwijl u wacht op een update van dit boek.

Ten slotte kunt u altijd de nieuwste versies van onze boekenserie downloaden via:

- https://mega65.org/mega65-book
- https://mega65.org/user-guide
- https://mega65.org/developer-guide
- https://mega65.org/basic65-ref
- https://mega65.org/chipset-ref
- https://mega65.org/docs

MEGA65 CHIPSET NASLAGGIDS

Uitgegeven door
het MEGA Museum of Electronic Games & Art e.V., Duitsland.

IN AANBOUW

Copyright ©2019 - 2026 Paul Gardner-Stephen, het MEGA Museum of Electronic Games & Art e.V. en medewerkers.

Dit boek is beschikbaar onder de GNU Free Documentation License v1.3 of latere versie. Dit betekent dat u dit boek vrij mag wijzigen, vermenigvuldigen en herdistribueren, onder bepaalde voorwaarden. De volledige tekst van de GNU Free Documentation License v1.3 is te vinden op https://www.gnu.org/licenses/fdl-1.3.en.html. Een Nederlandse vertaling van de licentie is te vinden op https://meta.wikimedia.org/wiki/GNU_Free_Documentation_License/nl.

Deze copyrightlicentie omvat ook de toestemming om dit document geheel of gedeeltelijk te kopiëren en/of te verspreiden voor gebruik in onderwijsomgevingen. Wij willen het onderwijs van toekomstige generaties ondersteunen, dus als u vragen of bezwaren heeft, neem dan contact met ons op.

oktober 29, 2026

Inhoudsopgave

1 Systeemgeheugenindeling **1**

Inleiding . . . 3

Eigen geheugenindeling van de MEGA65 . . . 4

De eerste zestien 64KB-banken . . . 4

Kleurengeheugen . . . 5

Extra RAM . . . 5

28-bits adresruimte . . . 5

$D000 - $DFFF I/O-persoonlijkheden . . . 7

Banking van het CPU-geheugen . . . 10

C64/C65-ROM-emulatie . . . 11

Indeling van de C65-compatibiliteits-ROM . . . 12

2 45GS02-microprocessor **15**

Inleiding . . . 17

Verschillen ten opzichte van de 6502 . . . 17

Supervisor-/hypervisormodus . . . 17

Niet-bedoelde 6502-instructies . . . 18

Bugcompatibiliteit van Read-Modify-Write-instructies . . . 18

Variabele CPU-snelheid . . . 19

Langzame werking (1 MHz - 3.5 MHz) . . . 19

Instructietiming op volle snelheid (40 MHz) . . . 19

Direct Memory Access (DMA) . . . 20

Geheugen benaderen tussen 64 KB en 1 MB . . . 20

Geheugenbanking in C64-stijl . . . 20

VIC-III "ROM"-banking . . . 20

Display Address Translator van de VIC-III . . . 21

De MAP-instructie . . . 21

DMA-controller . . . 24

Platte geheugentoegang . . . 24

Toegang tot geheugen buiten het 1MB-punt . . . 24

De MAP-instructie gebruiken om >1MB te benaderen . . . 24

Platte geheugentoegang 26
Virtueel 32-bit-register 27
In het geheugen gemapte registers van de C64-CPU 30
Nieuwe in het geheugen gemapte CPU-registers 30
MEGA65 CPU-versnellingsregisters voor wiskunde 33
MEGA65-hypervisormodus 38
Reset 38
De hypervisormodus in- en uitstappen 38
Hypervisor-geheugenindeling 40
Hypervisor-virtualisatieregisters 42
Programmeren in de hypervisormodus 46

3 45GS02 & 6502-instructiesets 51
Inleiding 53
Stapelbewerkingen 53
Adresseermethoden 54
Impliciet 54
Accumulator 54
Het pseudoregister Q 54
Immediate-modus 54
Immediate Word-modus 55
Base-Page-modus 55
Base-Page Quad-modus 55
Base-Page X-Indexed-modus 56
Base-Page Quad X-Indexed-modus 56
Base-Page Y-Indexed-modus 57
Absolute modus 57
Absolute Quad-modus 57
Absolute X-Indexed-modus 58
Absolute Quad X-Indexed-modus 58
Absolute Y-Indexed-modus 58

Absolute Indirect-modus 59
Absolute Indirect X-Indexed-modus 59
Base-Page Indirect X-Indexed-modus 59
Base-Page indirect Y-geïndexeerde modus 60
Base-Page indirect Z-geïndexeerde modus 60
Base-Page Quad indirect Z-geïndexeerde modus 60
32-bits Base-Page-indirecte Z-geïndexeerde modus 61
32-bits Base-Page-(Zero-Page)-indirecte quad Z-geïndexeerde modus 61
Stapelrelatieve indirecte Y-geïndexeerde modus 62
Relatieve adresseermethode 62
Relatieve word-adresseermethode 63
6502-instructieset 64
Officiële en onbedoelde instructies 64
Opcodetabel 64
De instructieset van de 4510 106
Instructietiming 106
Opcodetabel 106
Samengestelde instructies van de 45GS02 163

4 Naslagtabellen **185**
Opslageenheden 187
Getalstelselomrekening 188

5 Steuners & donateurs **193**
Organisaties 195
Bijdragers 196
Steuners 197

Index **207**

HOOFDSTUK 1

Systeemgeheugenindeling

- Inleiding
- Eigen geheugenindeling van de MEGA65
- $D000 – $DFFF I/O-persoonlijkheden
- Banking van het CPU-geheugen
- C64/C65-ROM-emulatie

INLEIDING

De MEGA65-computer heeft een grote 28-bits adresruimte, waarmee tot 256 MB aan geheugen en in het geheugen gemapte apparaten kunnen worden aangesproken. Deze geheugenindeling kent verschillende verschijningsvormen, afhankelijk van de modus waarin de computer draait. In grove lijnen zijn er vijf hoofdmodi: (1) hypervisormodus; (2) C64-compatibiliteitsmodus; (3) C65-compatibiliteitsmodus; (4) UltiMAX-compatibiliteitsmodus; en (5) MEGA65-modus, of een van de andere modi waarbij de programmeur gebruikmaakt van de uitgebreide functies van de MEGA65.

Het is belangrijk te begrijpen dat de C65 en MEGA65, in tegenstelling tot de C128, toegang bieden tot alle uitgebreide functies vanuit C64-modus, indien de programmeur dat wenst. Dit betekent dat wij, hoewel wij vaak over "C64-modus," "C65-modus" en "MEGA65-modus" spreken, eenvoudigweg gemaksnamen gebruiken voor de MEGA65 waarvan de geheugenindeling (en soms andere functies) zodanig is ingesteld dat zij een omgeving biedt die overeenkomt met de desbetreffende modus. De kern hiervan is de flexibele geheugenindeling van de MEGA65.

In deze bijlage beschrijven wij eerst de eigen geheugenindeling van de MEGA65, dat wil zeggen waar al het geheugen, de I/O-apparaten en andere functies hun plaats hebben in de 28-bits adresruimte. Vervolgens leggen wij uit hoe C64- en C65-compatibele geheugenindelingen vanuit deze 28-bits adresruimte worden benaderd.

EIGEN GEHEUGENINDELING VAN DE MEGA65

De eerste zestien 64KB-banken

De MEGA65 gebruikt voor het eerste MB geheugen een soortgelijke geheugenindeling als de C65, te weten 16 geheugenbanken van 64 KB elk. Dit komt doordat de 4510-CPU van de C65 slechts 1 MB adresruimte kan benaderen. Deze banken kunnen vanuit BASIC 65 worden benaderd met de commando's **BANK**, **DMA**, **PEEK** en **POKE**. De volgende tabel vat de inhoud van de eerste 16 banken samen:

HEX	DEC	Adres	Inhoud
0	0	$0xxxx	Eerste 64KB RAM. Dit is het RAM-geheugen dat in de C64-modus zichtbaar is.
1	1	$1xxxx	Tweede 64KB RAM. Dit is het tweede 64KB RAM-geheugen dat op een C65 aanwezig is.
2	2	$2xxxx	Eerste helft van de C65-ROM (C64-modus en gedeelde onderdelen) *of* RAM
3	3	$3xxxx	Tweede helft van de C65-ROM (C65-modus-onderdelen) *of* RAM
4	4	$4xxxx	Extra RAM (384KB of grotere chip-RAM-modellen)
5	5	$5xxxx	Extra RAM (384KB of grotere chip-RAM-modellen)
6	6	$6xxxx	Extra RAM (*512KB of grotere chip-RAM-modellen)
7	7	$7xxxx	Extra RAM (*512KB of grotere chip-RAM-modellen)
8	8	$8xxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
9	9	$9xxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
A	10	$Axxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
B	11	$Bxxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
C	12	$Cxxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
D	13	$Dxxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
E	14	$Exxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)
F	15	$Fxxxx	Extra RAM (*1MB of grotere chip-RAM-modellen)

* Let op: de MEGA65 biedt momenteel alleen een model met 384KB chip-RAM. Toekomstige modellen kunnen grotere hoeveelheden chip-RAM bevatten (zoals 512KB en 1MB).

De belangrijkste kenmerken van deze adresruimte zijn de 128KB RAM in de eerste twee banken, die ook op de C65 aanwezig is. Indien u programma's wilt schrijven die ook op een C65 kunnen draaien, dient u alleen deze twee banken RAM te gebruiken.

Op alle modellen is het mogelijk om alle of een deel van de 128KB "ROM"-ruimte als RAM te gebruiken. Daartoe moet u eerst de Hypervisor verzoeken de schrijfbeveiliging van dit gebied op te heffen, voordat u de inhoud kunt wijzigen. Indien u een

programma schrijft dat vanuit C64-modus start, of anderszins overschakelt naar het C64-deel van de ROM in plaats van het C65-deel, dan kan de tweede helft van die ruimte, te weten BANK 3, veilig voor uw programma's worden gebruikt. Dit geeft een totaal van 192KB RAM, dat op alle modellen van de MEGA65 beschikbaar is.

Op modellen met 384KB of meer chip-RAM zijn BANK 4 en 5 eveneens beschikbaar. Modellen die 1MB of meer chip-RAM bieden, hebben daarnaast BANK 6 tot en met 15 beschikbaar, wat een totaal van 896KB (of 960KB, indien alleen het C64-deel van de ROM nodig is) RAM oplevert voor uw programma's. Let op: de ingebouwde freeze-cartridge van de MEGA65 bevriest momenteel alleen de eerste 384KB RAM.

Kleurengeheugen

De VIC-IV-videokaart van de MEGA65 ondersteunt veel grotere schermen dan de VIC-II of VIC-III. Zij heeft daarom toegang tot een apart kleurengeheugen (colour-RAM), vergelijkbaar met dat van de C64. Voor compatibiliteit met de C65 zijn de eerste twee kilobytes hiervan toegankelijk op $1F800 - $1FFFF. Het volledige 32KB kleurengeheugen bevindt zich op $FF80000 - $FF87FFF. Dit is het eenvoudigst toegankelijk via geavanceerde DMA-bewerkingen of de 32-bits indirecte adresseermethode op de basispagina (zero page) van de processor.

Op het moment van schrijven kunnen de **BANK**- en **DMA**-commando's niet worden gebruikt om het resterende kleurengeheugen te bereiken, omdat het kleurengeheugen zich niet in het eerste megabyte van de adresruimte bevindt. Dit kan in een toekomstige revisie van de MEGA65 worden gecorrigeerd, waardoor toegang tot het volledige kleurengeheugen via BANK 15 of een equivalente DMA-taak mogelijk wordt.

Extra RAM

Naast de 384KB chip-RAM die standaard op alle MEGA65-modellen aanwezig is, hebben de meeste modellen (devkit, release-borden en xemu, maar momenteel NIET op Nexys-borden) ook 8MB extra RAM vanaf $8000000, aangeduid als 'ATTIC RAM'. Dit is niet zichtbaar voor de andere chips (vic/sid/enz.) en kan niet worden gebruikt voor audio-DMA, maar er kan code op worden uitgevoerd (weliswaar langzamer) of het kan worden gebruikt om inhoud op te slaan en via DMA in en uit het chip-RAM te verplaatsen.

Er zijn ook plannen om een PMOD hyperRAM-module te ondersteunen (geïnstalleerd via het klepje onder de MEGA65) om nog eens 8MB RAM vanaf $8800000 te bieden, aangeduid als 'CELLAR RAM'.

28-bits adresruimte

Naast de C65-stijl 1MB adresruimte breidt de MEGA65 deze uit tot 256MB met 28-bits adressen. De volgende tabel toont de indeling op hoofdlijnen van deze adresruimte.

HEX	DEC	Grootte	Inhoud
0000000	0	1	Richtingsregister van de CPU-I/O-poortgegevens
0000001	1	1	CPU-I/O-poortgegevens
0000002 - 005FFFF	2 - 384KB	384KB	Snel chip-RAM (40MHz)
0060000 - 0FFFFFF	384KB - 16MB	15.6MB	Gereserveerd voor toekomstige chip-RAM-uitbreiding
1000000 - 3FFFFFF	16MB - 64MB	48MB	Gereserveerd
4000000 - 7FFFFFF	64MB - 128MB	64MB	Cartridgepoort en andere apparaten op de langzame bus (1 - 10 MHz)
8000000 - 87FFFFF	128MB - 135MB	8MB	8MB ATTIC-RAM (momenteel op alle modellen behalve Nexys)
8800000 - 8FFFFFF	135MB - 144MB	8MB	8MB CELLAR-RAM (geplande PMOD-module, geïnstalleerd via het klepje)
9000000 - EFFFFFF	144MB - 240MB	96MB	Gereserveerd voor toekomstig uitbreidings-RAM
F000000 - FF7DFFF	240MB - 255.49MB	15.49MB	Gereserveerd voor toekomstige I/O-uitbreiding
FF7E000 - FF7EFFF	255.49MB - 255.49MB	4KB	VIC-IV-teken-ROM (alleen schrijven)
FF80000 - FF87FFF	255.5MB - 255.53MB	32KB	VIC-IV-kleurengeheugen (32KB kleurengeheugen - beschikbaar op alle modellen)
FF88000 - FF8FFFF	255.53MB - 255.57MB	32KB	Gereserveerd
FF90000 - FFCAFFF	255.53MB - 255.80MB	216KB	Gereserveerd
FFCB000 - FFCBFFF	255.80MB - 255.80MB	4KB	Geëmuleerd C1541-RAM
FFCC000 - FFCFFFF	255.80MB - 255.81MB	16KB	Geëmuleerde C1541-ROM
FFD0000 - FFD0FFF	255.81MB - 255.81MB	4KB	C64-$Dxxx-I/O-persoonlijkheid
FFD1000 - FFD1FFF	255.81MB - 255.82MB	4KB	C65-$Dxxx-I/O-persoonlijkheid
FFD2000 - FFD2FFF	255.82MB - 255.82MB	4KB	MEGA65-$Dxxx-Ethernet-I/O-persoonlijkheid
FFD3000 - FFD3FFF	255.82MB - 255.82MB	4KB	MEGA65-$Dxxx-normale-I/O-persoonlijkheid

Vervolgd ...

...Vervolgd

HEX	DEC	Grootte	Inhoud
FFD4000 - FFD5FFF	255.82MB - 255.83MB	8KB	Gereserveerd
FFD6000 - FFD67FF	255.83MB - 255.83MB	2KB	Tijdelijk geheugen voor de hypervisor
FFD6000 - FFD6BFF	255.83MB - 255.83MB	3KB	Tijdelijk geheugen voor de hypervisor
FFD6C00 - FFD6DFF	255.83MB - 255.83MB	512	Sectorbuffer van de F011-diskettecontroller
FFD6E00 - FFD6FFF	255.83MB - 255.83MB	512	Sectorbuffer van de SD-kaartcontroller
FFD7000 - FFD70FF	255.83MB - 255.83MB	256	I2C-apparaten van de MEGAphone r1
FFD7100 - FFD71FF	255.83MB - 255.83MB	256	I2C-apparaten van de MEGA65 r2
FFD7200 - FFD72FF	255.83MB - 255.83MB	256	MEGA65-HDMI-I2C-registers (alleen voor R2 en oudere modellen met de ADV7511-HDMI-driverchip)
FFD7300 - FFD7FFF	255.83MB - 255.84MB	3.25KB	Gereserveerd voor toekomstige I2C-apparaten
FFD8000 - FFDBFFF	255.83MB - 255.86MB	16KB	Hypervisor-ROM (alleen zichtbaar in de hypervisormodus)
FFDC000 - FFDDFFF	255.86MB - 255.87MB	8KB	Gereserveerd voor ROM-uitbreiding van de hypervisormodus
FFDE000 - FFDE7FF	255.87MB - 255.87MB	2KB	Gereserveerd voor uitbreiding van de Ethernet-buffer
FFDE800 - FFDEFFF	255.87MB - 255.87MB	2KB	Ethernet-frameleesbuffer (alleen lezen) en Ethernet-frameschrijfbuffer (alleen schrijven)
FFDF000 - FFDFFFF	255.87MB - 255.87MB	4KB	Virtuele FPGA-registers (alleen op geselecteerde modellen)
FFE0000 - FFFFFFF	255.87MB - 256MB	128KB	Gereserveerd

$D000 – $DFFF I/O-PERSOONLIJKHEDEN

De MEGA65 ondersteunt vier verschillende I/O-persoonlijkheden. Deze worden geselecteerd door de juiste waarden naar het $D02F KEY-register te schrijven, dat in alle

vier de I/O-persoonlijkheden zichtbaar is. In *het grote Mega65-boek*, C64-, C65- en MEGA65-modi (hoofdstuk 12) vindt u meer informatie over het gebruik van het KEY-register.

De volgende tabel toont welke I/O-apparaten in elk van deze I/O-modi zichtbaar zijn, evenals de KEY-registerwaarden die worden gebruikt om de I/O-persoonlijkheid te selecteren.

HEX	C64	C65	MEGA65 ETHERNET	MEGA65
KEY	$00	$A5, $96	$45, $54	$47, $53
$D000 - $D02F	VIC-II	VIC-II	VIC-II	VIC-II
$D030 - $D07F	VIC-II[1]	VIC-III	VIC-III	VIC-III
$D080 - $D08F	VIC-II	F011	F011	F011
$D090 - $D09F	VIC-II	-	SD-kaart	SD-kaart
$D0A0 - $D0FF	VIC-II	RAM EXPAND CONTROL	-	-
$D100 - $D1FF	VIC-II	ROOD PALET	ROOD PALET	ROOD PALET
$D200 - $D2FF	VIC-II	GROEN PALET	GROEN PALET	GROEN PALET
$D300 - $D3FF	VIC-II	BLAUW PALET	BLAUW PALET	BLAUW PALET
$D400 - $D41F	SID rechts #1	SID rechts #1	SID rechts #1	SID rechts #1
$D420 - $D43F	SID rechts #2	SID rechts #2	SID rechts #2	SID rechts #2
$D440 - $D45F	SID links #1	SID links #1	SID links #1	SID links #1
$D460 - $D47F	SID links #2	SID links #2	SID links #2	SID links #2
$D480 - $D49F	SID rechts #1	SID rechts #1	SID rechts #1	SID rechts #1
$D4A0 - $D4BF	SID rechts #2	SID rechts #2	SID rechts #2	SID rechts #2
$D4C0 - $D4DF	SID links #1	SID links #1	SID links #1	SID links #1
$D4E0 - $D4FF	SID links #2	SID links #2	SID links #2	SID links #2
$D500 - $D5FF	SID-kopieën	-	Gereser-veerd	Gereser-veerd
$D600 - $D63F	-	UART	UART	UART
$D640 - $D67F	-	UART-kopieën	HyperTrap-registers	HyperTrap-registers
$D680 - $D6FF	-	-	MEGA65-apparaten	MEGA65-apparaten
$D700 - $D7FF	-	-	MEGA65-apparaten	MEGA65-apparaten
$D800 - $DBFF	KLEURENGE-HEUGEN	KLEURENGE-HEUGEN	ETHERNET-buffer	KLEURENGE-HEUGEN
$DC00 - $DDFF	CIAs	CIAs / KLEURENGE-HEUGEN	ETHERNET-buffer	CIAs / KLEURENGE-HEUGEN
$DE00 - $DFFF	CART I/O	CART I/O	ETHERNET-buffer	CART I/O / SD-sectorbuffer

Vervolgd ...

...Vervolgd

HEX	C64	C65	MEGA65 ETHERNET	MEGA65

[1] In de C64-I/O-persoonlijkheid gedraagt $D030 zich zoals op de C128, waarmee kan worden geschakeld tussen 1MHz en 2MHz CPU-snelheid.
[2] De extra MEGA65-SID's zijn zichtbaar in alle I/O-persoonlijkheden.
[3] Sommige modellen kunnen de herhaalde kopieën van de eerste vier SID's vervangen door vier extra SID's, voor een totaal van 8 SID's.

BANKING VAN HET CPU-GEHEUGEN

De 45GS02-processor kan, net als de 6502, slechts 64 KB geheugen tegelijk "zien". Toegang tot aanvullend geheugen verloopt via een reeks bankswitchingmechanismen. Voor backward-compatibiliteit met de C64 en C65 worden de geheugenbankmechanismen van beide computers ondersteund op de MEGA65:

1. Banking via de MAP-instructie in C65-stijl
2. $D030-banking in C65-stijl
3. Cartridge-banking in C64-stijl
4. $00 / $01-banking in C64-stijl

Het MAP-register overschrijdt alle andere bankmechanismen. Dit mechanisme selecteert welke van de acht 8 KB-gebieden van de 16-bits adresruimte $0000 - $FFFF via een offset op andere adressen worden gemapt. Wanneer een gebied gemapt is, zijn de overige bankmechanismen niet van toepassing. Dit geldt zelfs wanneer de offset 0 is, waardoor de 16-bits adressen toegang krijgen tot het RAM in bank 0 (zoals adres 0.D000).

C65-stijl $D030-banking en C64-stijl $00 / $01-banking selecteren beide gebieden die naar bank 2 worden gemapt, die standaard de C64-ROM-code bevat. Deze twee mechanismen overlappen in de gebieden die zij naar ROM kunnen mappen. Wanneer een van de twee mechanismen een gebied naar ROM mappt (en het niet elders door het MAP-register is gemapt), wordt het naar ROM gemapt.

De volgende figuur toont de verschillende vormen van banking die van toepassing kunnen zijn op de verschillende gebieden van de 64 KB die de CPU kan zien.

MAP	MAP LO (4 x 8KB-gebieden)	MAP HI (4 x 8KB-gebieden)				
I/O/CART		CART ROMLO	CART ROMHI		I/O	CART ROMHI
D030		CHARROM	BASIC	INTER-FACE		KERNAL
C64			BASIC		CHAR ROM	KERNAL
RAM	RAM	RAM	RAM	RAM	RAM	RAM
	$0000 - $7FFF	$8000 - $9FFF	$A000 - $BFFF	$C000 - $CFFF	$D000 - $DFFF	$E000 - $FFFF

Er zijn enkele bijkomende complicaties. Wanneer de cartridge bijvoorbeeld de UltiMAX™-gamemodus selecteert, is slechts de eerste 4 KB RAM zichtbaar en wordt de rest van de adresruimte niet gemapt, zodat deze door de cartridge kan worden aangeboden.

C64/C65-ROM-EMULATIE

De C64 en C65 gebruiken ROM-geheugen om het KERNAL- en BASIC-systeem op te slaan. De MEGA65 werkt anders: zij gebruikt 128 KB van de 384 KB snelle chip-RAM op $20000 - $3FFFF (banken 2 en 3) om deze systeemprogramma's op te slaan. Hierdoor is het mogelijk de "ROM" die de MEGA65 gebruikt te wijzigen of bij te werken zonder de computer te openen. Het is zelfs mogelijk het Freeze-menu van de MEGA65 te gebruiken om de gebruikte "ROM" te wijzigen terwijl een programma draait.

De geheugenbankmethoden van de C64 en C65 gebruiken dit 128 KB-gebied wanneer zij ROM-banken zichtbaar maken. Wanneer de RAM-banken gemapt zijn, zijn zij altijd alleen-lezen. Wanneer echter de MAP-instructie of DMA wordt gebruikt om dat adresgebied te bereiken, is het mogelijk ernaar te schrijven. Voor verbeterde backward-compatibiliteit is het volledige 128 KB-geheugengebied standaard op alleen-lezen ingesteld.

Een programma kan evenwel lees-schrijftoegang tot dit 128 KB-geheugengebied aanvragen, zodat het programma het volledige 384 KB chip-RAM van de MEGA65 kan benutten. Dit wordt gerealiseerd door de *ROM Write-enable* systeemtrap van de hypervisor te activeren. Het volgende codefragment demonstreert hoe dit werkt.

```
LDA #$02
STA $D641
CLV
```

Dit fragment werkt door subfunctie $02 (ROM write-enable) van hypervisortrap $01 aan te roepen. Subfunctie $00 van dezelfde trap schakelt de schrijfbeveiliging weer in. De instructie na de STA moet een instructie met één cyclus zijn, doorgaans CLV: de CPU voert die bij het terugkeren uit de Hypervisor al dan niet uit. NOP wordt niet

aanbevolen, omdat deze op de 45GS02 als prefix voor de volgende instructie kan werken. De MEGA65 I/O-persoonlijkheid moet eerst worden geselecteerd, zodat de hypervisor-trap-registers ($D640 - $D67F) zichtbaar worden.

De huidige schrijfbeveiligingstoestand kan worden getest door te proberen naar dit geheugengebied te schrijven. Daarnaast kunt u de huidige toestand inspecteren en wijzigen in het Freeze-menu van de MEGA65.

Opmerking: wanneer u het programma vanuit C65-modus start, moet u er eerst voor zorgen dat het I/O-gebied zichtbaar is op $D000-$DFFF. De eenvoudigste manier om dit te doen is de MAP-instructie te gebruiken met alle waarden nul in de registers. Het volgende fragment demonstreert dit en zorgt er bovendien voor dat de MEGA65 I/O-context actief is, zodat de hypervisortrap kan worden geactiveerd:

```
; Wis de C65-geheugenindeling
LDA #$00
TAX
TAY
TAZ
MAP
; Bank I/O in via het C64-mechanisme
LDA #$35
STA $01
; Activeer de MEGA65 / VIC-IV-I/O-persoonlijkheid
LDA #$47
STA $D02F
LDA #$53
STA $D02F
; Einde van de MAP-reeks, zodat interrupts weer kunnen optreden
EOM
; Hypervisor-aanroep om de schrijfbeveiliging van het ROM-gebied op te heffen
LDA #$02
STA $D641
CLV
```

Indeling van de C65-compatibiliteits-ROM

De indeling van het 128KB C65-compatibiliteits-ROM-gebied komt overeen met die van de C65:

HEX	Inhoud
$3E000 -- $3FFFF	C65 KERNAL
$3D000 -- $3DFFF	CHARSET B
$3C000 -- $3CFFF	GERESERVEERD
$38000 -- $3BFFF	C65-BASIC-grafische routines
$32000 -- $37FFF	C65-BASIC
$30000 -- $31FFF	MONITOR (wordt op $6000 -- $7FFF gemapt)
$2E000 -- $2FFFF	C64 KERNAL
$2D000 -- $2DFFF	CHARSET C
$2C800 -- $2CFFF	INTERFACE
$2C000 -- $2C7FF	GERESERVEERD
$2A000 -- $2BFFF	C64-BASIC
$29000 -- $29FFF	CHARSET A
$24000 -- $28FFF	GERESERVEERD
$20000 -- $23FFF	DOS (wordt op $8000 -- $BFFF gemapt)

Het INTERFACE-programma is een reeks routines die door de C65 worden gebruikt om te wisselen tussen C64-modus, C65-modus en het ingebouwde DOS van de C65. Het DOS bevindt zich in het onderste achtste van het ROM.

HOOFDSTUK 2

45GS02-microprocessor

- Inleiding
- Verschillen ten opzichte van de 6502
- In het geheugen gemapte registers van de C64-CPU
- Nieuwe in het geheugen gemapte CPU-registers
- MEGA65 CPU-versnellingsregisters voor wiskunde
- MEGA65-hypervisormodus

INLEIDING

De 45GS02 is een verbeterde versie van het processorgedeelte van de CSG4510 en van de F018 "DMAgic" DMA-controller die werd gebruikt in de prototypes van de Commodore 65-computer. De 4510 is op zijn beurt een verbeterde versie van de 65CE02. Voor de achterwaarts compatibele werking van de 45GS02 verwijzen wij naar de uitgebreide documentatie die beschikbaar is voor de 6502- en 65CE02-processors.

Dit hoofdstuk zal zich richten op de verschillen tussen de 45GS02 en de eerdere 6502-klasse processors, en op de beschrijving van de vele ingebouwde in het geheugen gemapte I/O-registers van de 45GS02.

VERSCHILLEN TEN OPZICHTE VAN DE 6502

De 45GS02 heeft een aantal belangrijke verschillen met eerdere 6502-klasse processors:

Supervisor-/hypervisormodus

In tegenstelling tot de eerdere 6502-varianten beschikt de 45GS02 over een geprivilegieerde modus. Deze modus is bedoeld voor gebruik door een besturingssysteem of een type-1-hypervisor. De dubbelzinnigheid tussen besturingssysteem en Hypervisor op de MEGA65 vloeit voort uit het feit dat het besturingssysteem van de MEGA65 in feite niet meer is dan een loader en taakwisselaar voor C64- en C65-omgevingen, oftewel: het functioneert in feite als een hypervisor, maar biedt slechts beperkte virtualisatie van de hardware.

De belangrijkste verschillen tussen de normale modus en de supervisor-modus op de MEGA65 zijn dat in supervisor-modus:

- Een speciaal 16KB-geheugengebied wordt toegewezen aan $8000 - $BFFF, dat zowel het programma als de gegevens van het hypervisor-/supervisorprogramma bevat. Dit is doorgaans het Hyppo-programma. Dit geheugen kan niet worden toegewezen wanneer de processor zich in de normale modus bevindt (de chipselectlijn ernaartoe is geblokkeerd), waardoor het wordt beschermd tegen onbedoelde of kwaadwillende toegang.
- De 64 SYSCALL-trapregisters in de MEGA65 I/O-modus op $D640 - $D67F worden vervangen door de virtualisatieregisters. Deze registers bieden volledige controle over het systeem, en het is hun toegang die het privilege van de supervisor-modus werkelijk definieert.

- De processor werkt altijd op volle snelheid (40 MHz) en in de 4510-processorpersoonlijkheid.

De hypervisormodus wordt later in deze bijlage uitgebreider beschreven.

Niet-bedoelde 6502-instructies

De 65C02-, 65CE02- en CSG4510-processors hebben de originele 6502-processor uitgebreid door eerder niet-toegewezen opcodes van de 6502 te gebruiken om extra instructies te bieden. Alle software die de officiële documentatie van de 6502-processor volgde, zal daarom op deze nieuwere processors werken, mogelijk met een andere instructietiming. De veelgebruikte praktijk op de C64 en andere homecomputers van het gebruik van ongedefinieerde of niet-bedoelde opcodes (vaak aangeduid als "illegale opcodes", hoewel er geen wet is die het gebruik ervan verbiedt), betekent echter dat veel bestaande programma's niet op deze nieuwere processors zullen werken.

Om dit probleem te verhelpen, kan de 45GS02 wisselen tussen processorpersoonlijkheden van de 4510 en 6502. Het effect is dat in 6502-modus geen van de nieuwe opcodes van de 65C02, 65CE02, 4510 of 45GS02 beschikbaar zijn, en worden vervangen door het originele, vaak vreemde gedrag van de ongedefinieerde opcodes van de 6502.

> **WAARSCHUWING:** Deze functie is onvolledig en ongetest. De meeste niet-bedoelde 6502-instructies werken niet correct wanneer de 6502-persoonlijkheid is ingeschakeld.

Bugcompatibiliteit van Read-Modify-Write-instructies

De 65CE02-processor heeft een groep instructies geoptimaliseerd die Read-Modify-Write (RMW)-instructies worden genoemd. Bij dergelijke instructies, zoals INC, die de inhoud van een geheugenlocatie verhogen, zou de 6502 de oorspronkelijke waarde lezen en deze ongewijzigd terugschrijven, voordat deze met de nieuwe verhoogde waarde werd teruggeschreven. Voor de meeste doelen veroorzaakte dit geen problemen. Het bleek echter een snelle manier te zijn om VIC-II-onderbrekingen te bevestigen, omdat het terugschrijven van de oorspronkelijke waarde (wat de instructie niet hoeft te doen) de onderbreking bevestigt. Deze methode is sneller en gebruikt minder bytes dan elk alternatief, en werd daarom veel gebruikt in C64-software.

Het probleem ontstond bij de C65 met de van de 65CE02 afgeleide CSG4510, die deze extra schrijfbeweging tijdens de RMW-instructies niet uitvoerde. Hierdoor waren de RMW-instructies één cyclus sneller, waardoor software iets sneller draaide. Dit betekende helaas ook dat veel bestaande C64-software eenvoudigweg niet op een C65 zal werken, tenzij de onderbrekingsbevestigingscode in elk programma wordt aangepast om dit probleem te omzeilen. Dit is de belangrijkste reden waarom veel C64-spellen en andere softwaretitels niet op een C65 werken.

Omdat dit probleem zo vaak voorkomt, bevat de 45GS02 van de MEGA65 bugcompatibiliteit met deze veelgebruikte functie van de originele 6502. Dit doet de pro-

cessor door te controleren of het doel van een RMW-instructie $D019 is, oftewel het onderbrekingsstatusregister van de VIC-II. Indien dit het geval is, voert de 45GS02 de dummy-schrijfbeweging uit, waardoor veel C64-softwaretitels die niet op een C65-prototype werken, ongewijzigd op de MEGA65 kunnen worden uitgevoerd. Door de dummy-schrijfbeweging alleen uit te voeren wanneer het adres $D019 is, behoudt de MEGA65 C64-compatibiliteit, zonder de snelheidsverbetering op te offeren voor alle andere toepassingen van deze instructies.

Variabele CPU-snelheid

De 45GS02 kan draaien op 1MHz, 2MHz, 3.5MHz en 40MHz, om software te ondersteunen die is ontworpen voor de C64, C128 in C64-modus, C65 en MEGA65.

Langzame werking (1 MHz - 3.5 MHz)

In deze modi vertraagt de 45GS02-processor, zodat hetzelfde aantal instructies per videoframe wordt uitgevoerd als op een PAL- of NTSC-C64, C128 in C64-modus of een C65-prototype. Dit stelt bestaande software in staat om op de MEGA65 op de juiste snelheid en met minimale weergaveproblemen te draaien. De VIC-IV-videoregelaar levert cyclusindicatiepulsen aan de 45GS02 die worden gebruikt om de tijd bij te houden.

In deze modi hebben opcodes hetzelfde aantal cycli als een 6502. Geheugentoegangen binnen een instructie zijn echter niet gegarandeerd in dezelfde cyclus te vinden als op een 1MHz-6502. Normaal gesproken is het effect dat instructies sneller voltooid worden en de processor wacht totdat het juiste aantal cycli is verstreken. Dit betekent dat de timing tot 7 microseconden onjuist kan zijn. Dit is doorgaans geen probleem, en zelfs veel C64-snelladers werken correct. Bijvoorbeeld kan het GEOS™ Graphical Operating System voor de C64 worden opgestart en gebruikt vanaf een 1541 die op de seriële poort van de MEGA65 is aangesloten.

Enkele geavanceerde VIC-II-grafiektrucs, zoals Variable Screen Position (VSP), zullen hoogstwaarschijnlijk niet correct werken vanwege de onzekerheid in de timing van de geheugenschrijfcycli van instructies. In de meeste gevallen kunnen dergelijke problemen echter eenvoudig worden opgelost door de geavanceerde mogelijkheden van de VIC-IV-videoregelaar van de MEGA65 te gebruiken. VSP is bijvoorbeeld overbodig op de MEGA65, omdat u het schermRAM-adres op elke locatie in het geheugen kunt instellen.

Instructietiming op volle snelheid (40 MHz)

Wanneer de processor van de MEGA65 op volle snelheid werkt (momenteel 40 MHz), komt de instructietiming niet langer exact overeen met die van de 6502: instructies die in minder cycli kunnen worden uitgevoerd, doen dat ook. Zo vereisen sprongen op de 45GS02 doorgaans minder cycli. Er zijn ook instructies die op de 45GS02 meer cycli vereisen, met name de instructies LDA, LDX, LDY en LDZ. Deze instructies vereisen doorgaans één extra cyclus. Omdat de processor echter op 40 MHz werkt, worden

deze instructies nog steeds veel sneller uitgevoerd dan op een C65 of C64 met een accelerator.

Direct Memory Access (DMA)

Direct Memory Access (DMA) is een methode om geheugenregio's snel te vullen, te kopiëren of te verwisselen. De MEGA65 implementeert een verbeterde versie van de F018 "DMAgic" DMA-controller van de C65-prototypen. Anders dan bij de C65-prototypen maakt de DMA-controller deel uit van de CPU op de MEGA65.

Gedetailleerde informatie over het gebruik van de DMA-controller en deze geavanceerde mogelijkheden is te vinden in *het grote Mega65-boek*, F018-compatibele controller voor Direct Memory Access (DMA) (Bijlage P)

Geheugen benaderen tussen 64 KB en 1 MB

De C65 bood vier manieren om geheugen voorbij het 64 KB-punt te benaderen: drie methoden die beperkt, gespecialiseerd of beide zijn, en twee algemeen toepasbare methoden. Eerst bespreken we de beperkte methoden, waarna we de algemeen toepasbare methoden documenteren.

Geheugenbanking in C64-stijl

De eerste methode is het gebruik van de C64-stijl $00/$01 ROM/RAM-banking. Deze methode is echter zeer beperkt, omdat zij alleen het in- en uitbanken van de twee 8 KB-regio's toestaat die overeenkomen met de C64 BASIC- en KERNAL-ROM's. Deze bevinden zich op $2A000 en $2E000 in de 20-bits C65-adresruimte, d.w.z. $002A000 en $002E000 in de 28-bits adresruimte van de MEGA65. Zij kan ook toegang geven tot de C64 character-ROM-gegevens op $D000, die zich op $2D000 in de C65-geheugenindeling bevinden, en dus op $002D000 in de adresruimte van de MEGA65. Behalve dat deze methode beperkt is in welke regio's zij kan benaderen, biedt zij ook alleen leestoegang tot deze geheugenregio's, d.w.z. dat zij niet kan worden gebruikt om deze geheugenregio's te wijzigen.

VIC-III "ROM"-banking

De C65 bevat nog een voorziening om C64-ROM-regio's zichtbaar te maken in de C65-modus, naast het regelen van de zichtbaarheid van het kleurengeheugen in de I/O-registerregio. Deze worden in- en uitgebankt met behulp van diverse bits van het $D030-register van de VIC-III:

$D030 Bit	Signaal-naam	20-bitsadres	16-bitsadres	Lees-/Schrijf-toegang
0	CRAM2K	$1F800 - $1FFFF, $FF80000 - $FF807FF	$D800 - $DFFF	J
3	ROM8	$28000 - $29FFF	$8000 - $9FFF	N
4	ROMA	$2A000 - $2BFFF	$A000 - $BFFF	N
5	ROMC	$2C000 - $2CFFF	$C000 - $CFFF	N
7	ROME	$2E000 - $2FFFF	$E000 - $FFFF	N

De eerste 2 KB van het kleurengeheugen zijn altijd zichtbaar op $1F800 - $1FFFF. De eerste 1 KB hiervan is ook zichtbaar op $D800 - $DBFF. Wanneer het CRAM2K-signaal 1 is, wordt de tweede kilobyte beschikbaar gesteld op $DC00 - $DFFF. Anders dan bij de C64 is het kleurengeheugen op de MEGA65 altijd zichtbaar als 8-bits bytes. Ook is het kleurengeheugen op de MEGA65 32 KB groot en bevindt het zich op $FF80000 - $FF87FFF. De zichtbaarheid van het kleurengeheugen op $1F800 - $1FFFF wordt bereikt door schrijfbewerkingen naar beide regio's te spiegelen wanneer het kleurengeheugen via dit mechanisme wordt benaderd.

Zowel de VIC-III-banking als de C64-stijl-banking kunnen van toepassing zijn op $A000 - $BFFF en $E000 - $FFFF. Zij hebben hetzelfde effect van het mappen van die regio's naar ROM-adressen in bank 2. Indien een van beide mechanismen actief is voor een regio, dan wordt de regio naar ROM gemapt.

VIC-ROM-banking heeft alleen toegang tot de C64-ROM. De C65/MEGA65-ROM wordt benaderd met behulp van het MAP-register.

Display Address Translator van de VIC-III

De derde gespecialiseerde manier om toegang te krijgen tot geheugen boven het 64 KB-punt is het gebruik van de Display Address Translator van de VIC-III. Het gebruik van dit mechanisme is gedocumenteerd in *het grote Mega65-boek*, VIC-IV Video Interface Controller (Bijlage Q).

De MAP-instructie

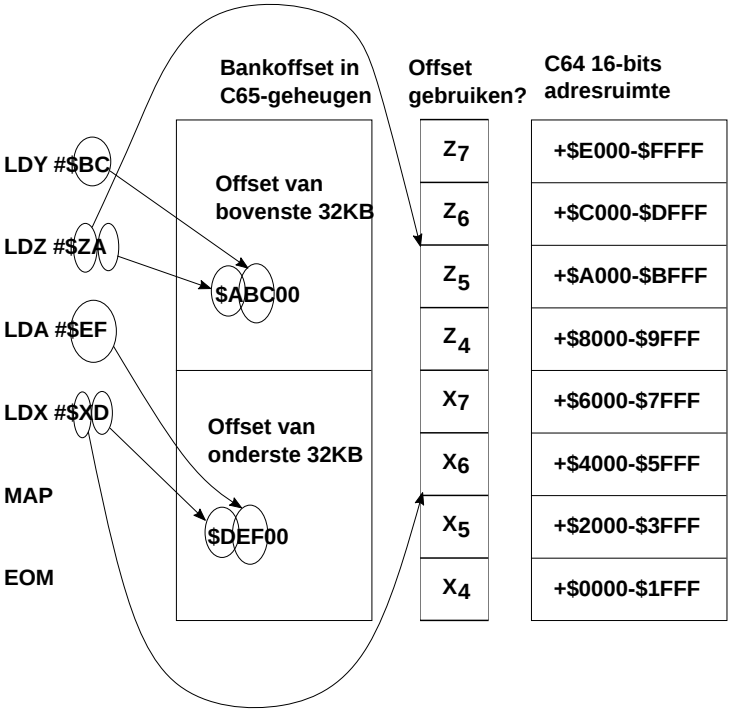
De eerste algemeen toepasbare manier van toegang tot het geheugen is de MAP-instructie van de 4510-processor. De 45GS02-processor van de MEGA65 ondersteunt dit mechanisme eveneens. Deze instructie verdeelt het 64 KB-adres van de 6502 in acht blokken van 8 KB elk. Voor elk van deze blokken kan het blok normaal worden benaderd, d.w.z. dat een 8 KB-regio van de eerste 64 KB RAM van het systeem wordt benaderd. Alternatief kan elk blok in plaats daarvan worden hermapt (vandaar

de naam van de MAP-instructie) naar een andere plaats in de adresruimte, door een offset aan het adres toe te voegen. Gemapte adressen in de eerste 32 KB gebruiken een offset, de lagere offset, en de tweede 32 KB gebruikt een andere, de hogere offset. Het hermappen van geheugen met behulp van de MAP-instructie gaat vóór alle andere bankingmechanismen.

De offsets moeten een veelvoud van 256 bytes zijn en dus uit 12 bits bestaan, om een willekeurige offset in de 1 MB-adresruimte van de C65 mogelijk te maken. Aangezien elk 8 KB-blok in een 32 KB-helft van het geheugen al dan niet kan worden gemapt, is hiervoor één bit per 8 KB-blok vereist. De processor vereist dus 16 bits aan informatie voor elke geheugenhelft, voor in totaal 32 bits aan informatie. Dit wordt bereikt door de A- en X-registers in te stellen voor de onderste geheugenhelft en de Y- en Z-registers voor de bovenste geheugenhelft, alvorens de MAP-instructie uit te voeren.

De MAP-instructie kopieert de inhoud van deze registers naar de interne registers van de processor die de mappinginformatie bevatten.

Het volgende diagram illustreert hoe de MAP-instructie de waarden van de vier A-, X-, Y- en Z-registers gebruikt om de bovenste en onderste adresoffsets te berekenen en de bank-enablebits in te stellen voor elk van de acht 8KB-geheugenregio's van de 6502-adresruimte:

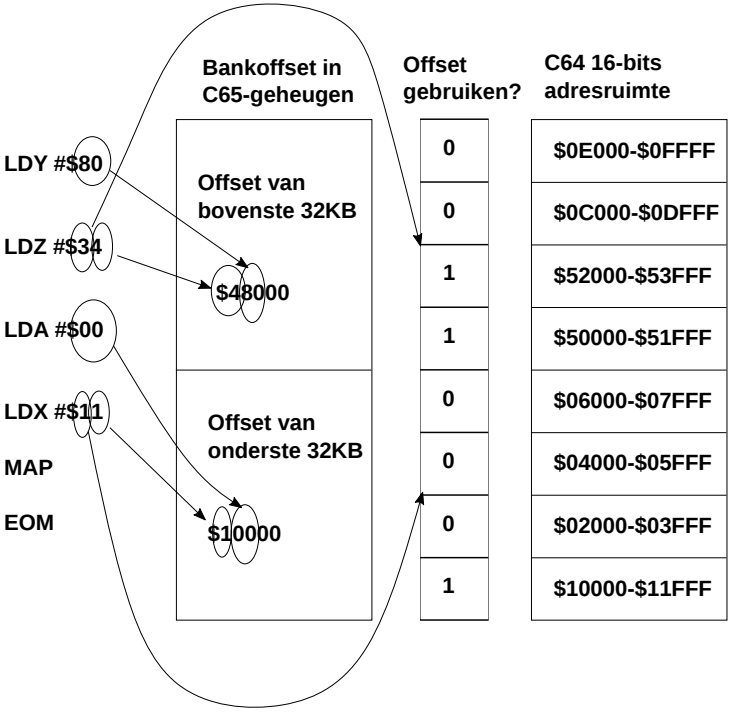


De inhoud van het A-register en de onderste nibble van het X-register vormen samen een 12-bit waarde die met 256 wordt vermenigvuldigd om de offset te produceren die wordt gebruikt voor elk van de 8KB-banken in het onderste 32KB-gedeelte van de 16-bit adresruimte van de 6502. De bovenste nibble van het X-register wordt als vlaggen gebruikt om aan te geven welke van de vier 8KB-blokken in dat 32KB-gedeelte van de 6502-adresruimte de offset bij hun adres moeten optellen om het werkelijke adres te berekenen.

De Y- en Z-registers worden op vergelijkbare manier gebruikt om de offset te produceren voor het bovenste 32KB-gedeelte van de 6502-adresruimte, en de vlaggen om aan te geven of de offset wordt gebruikt voor elk van de vier 8KB-blokken in dat gedeelte van de adresruimte.

De onderste 8 bits van de offset kunnen niet worden ingesteld. De offset is dus een veelvoud van 256 bytes, in tegenstelling tot sommige uitgebreide 6502-processoren. In de praktijk is deze beperking echter zelden van invloed.

Om te begrijpen hoe dit in de praktijk werkt, laat het volgende voorbeeld zien hoe dit werkt met een concreet voorbeeld, met de adresbereiken die zichtbaar zijn in elk van de 8KB-segmenten van de 64KB-adresruimte van de 6502:



Merk op dat de offsets voor elk van de twee 32KB-adresbereiken worden opgeteld bij het 6502-adres. Vandaar dat de offset van $48000 voor het bovenste 32KB-gedeelte een adres van $50000 genereert bij het 6502-adres $8000.

Zie ook "De MAP-instructie gebruiken om >1MB te benaderen" voor verdere uitleg.

DMA-controller

De F018/F018A DMA-controller van de C65 maakt het mogelijk om gegevens in het geheugen op willekeurige locaties in de 1MB-adresruimte snel te vullen, te kopiëren en te wisselen. Gedetailleerde informatie over de F018 DMA-controller en de verbeteringen die de MEGA65 daarop heeft aangebracht, vindt u in *het grote Mega65-boek*, F018-compatibele controller voor Direct Memory Access (DMA) (Bijlage P)

Platte geheugentoegang

Toegang tot geheugen buiten het 1MB-punt

De MEGA65 kan tot 256MB geheugen ondersteunen. Dit is meer dan de 1MB-adresruimte van de CSG4510 waarop deze is gebaseerd. Er zijn verschillende manieren om dit te bereiken.

De MAP-instructie gebruiken om >1MB te benaderen

De volledige adresruimte is beschikbaar voor de MAP-instructie voor traditionele C65-stijl geheugenmapping, hoewel enige voorzichtigheid geboden is, omdat de MAP-instructie tot drie keer moet worden aangeroepen. De reden hiervoor is dat de MAP-instructie eerst moet worden aangeroepen om te selecteren welk megabyte geheugen zal worden gebruikt voor de onderste en bovenste mapregio's, voordat deze opnieuw op de normale manier wordt aangeroepen om de geheugenmapping in te stellen. Omdat tussen deze twee aanroepen de geheugenmapping-offset een mix van oude en nieuwe adressen zal zijn, moet alle mapping eerst worden uitgeschakeld via de MAP-instructie. Dit betekent dat de code voor het hermappen van geheugen zich in het onderste 64KB RAM of in een van de ROM-bankbare regio's moet bevinden, zodat deze tijdens het mappingproces zichtbaar blijft.

Als deze situatie niet goed wordt afgehandeld, zal de processor instructies uitvoeren vanuit een onverwachte geheugenlocatie halverwege het proces, doordat de routine die hij uitvoert om de mapping uit te voeren plotseling niet meer in het geheugen gemapt is.

Vanwege de relatieve complexiteit van dit proces en de andere beperkingen van de MAP-instructie als middel om geheugen te benaderen, raden we u aan om voor gegevens buiten de huidige geheugenindeling ofwel DMA of de flat-memory adresvoorzieningen van de 45GS02 te gebruiken die hieronder worden beschreven. Toegang tot de volledige adresruimte via de MAP-instructie is alleen beschikbaar voor de volledigheid.

Als een ander voorbeeld van hoe de MAP-instructie kan worden gebruikt om een geheugengebied uit de uitgebreide adresruimte te mappen, laat het volgende programma de Ethernet-framebuffer vanaf de natuurlijke locatie op $FFDE8000 op $6800 verschijnen. Om het voorbeeld zo eenvoudig mogelijk te houden, gaan we ervan uit dat de code in het onderste 64KB RAM draait en niet in het gebied tussen $6000 - $8000.

Omdat de MAP-instructie normaal gesproken alleen de C65-stijl 20-bit adressen kent, moet de MEGA65-uitbreiding op de instructie worden gebruikt om de bovenste 8 bits van de 28-bit MEGA65-adressen in te stellen, dat wil zeggen welk megabyte adresruimte moet worden gebruikt voor de adresvertaling. Dit wordt gedaan door het X-register in te stellen op $0F wanneer het megabyte-nummer voor de onderste 32KB van de C64-stijl 64KB-adresruimte wordt gedefinieerd. Dit veroorzaakt geen incompatibiliteit met enig redelijk gebruik van de MAP-instructie op een C65, omdat deze waarde aangeeft dat geen van de vier 8KB-geheugenblokken opnieuw zal worden gemapt, maar tegelijkertijd specificeert dat de bovenste 4 bits van de adresoffset voor het hergemapte blok de niet-nul-waarde $F vormt. Het megabyte-nummer wordt vervolgens gespecificeerd door het A-register in te stellen.

Dezelfde aanpak geldt voor de bovenste 32KB, maar met de Z- en Y-registers in plaats van de X- en A-registers. In dit geval hoeven we de bovenste 32KB geheugen niet te hermappen; we laten de Z- en Y-registers op nul staan. We moeten echter X en A instellen om het megabyte-nummer voor de onderste 32KB op $FF te zetten. A moet dus de waarde $FF hebben. Om de onderste 20 bits van de adresoffset in te stellen, gebruiken we de MAP-instructie een tweede keer, dit keer op de normale C65-manier. Omdat we $6800 naar $FFDE800 willen hermappen en al met de $FFxxxxx-offset via het megabyte-nummer hebben afgerekend, hoeven we alleen de offset toe te passen zodat $6800 naar $DE800 wijst. $DE800 minus $6800 = $D8000. Omdat de MAP-instructie werkt met een hermappinggranulariteit van 256 bytes = $100, kunnen we de laatste twee cijfers van $D8000 laten vallen om de MAP-offset van $D80 te verkrijgen. De onderste 8 bits, $80, moeten in het A-register worden geladen. De bovenste 4 bits, $D, moeten in de onderste nibble van het X-register worden geladen. Omdat we de mapping alleen op het vierde van de vier 8KB-blokken willen toepassen die de onderste 32KB-helft van de C64-geheugenindeling vormen, moeten we het 4e bit van de bovenste nibble instellen. Dat wil zeggen, de bovenste nibble moet worden ingesteld op %1000, oftewel $8. Het X-register moet dus worden geladen met $8D. Hiermee verkrijgen we het volledige voorbeeldprogramma:

```
; Map de Ethernet-registers op $6000 - $7FFF

; De Ethernet-controller bevindt zich feitelijk op $FFDE000 - $FFDEFFF, dus selectee
LDA #$ff
LDX #$0f
LDY #$00
LDZ #$00
map

; Activeer nu het mappen van $DE000-$DFFFF op $6000
; De MAP's zijn op offset gebaseerd, dus moet $6000 van het doeladres worden afgetro
; $DE000 - $6000 = $D8000
LDA #$80
LDX #$8d
LDY #$00
LDZ #$00
map
EOM

; De Ethernet-buffer is nu zichtbaar op $6800 - $6FFF
```

De EOM (End Of Mapping)-instructie (hetzelfde als NOP op een 6502, oftewel opcode $EA) werd alleen na de laatste MAP-instructie opgenomen om er zeker van te zijn dat er geen interrupts konden optreden terwijl de geheugenindeling gemengde waarden bevatte met het megabyte-nummer ingesteld, maar de onderste bits van het mappingadres nog niet waren bijgewerkt.

Er is geen voorbeeld in BASIC voor de MAP-instructie mogelijk, omdat de MAP een machinetaalinstructie is van de 4510/45GS02-processoren.

Platte geheugentoegang

De 45GS02 maakt het eenvoudig om een byte vanuit elke geheugenlocatie te lezen of te schrijven door de Zero-Page Indirect-adresseermethode een 32-bit pointer te laten gebruiken in plaats van de gebruikelijke 16-bit pointer. Dit wordt bereikt door de Z-geïndexeerde Zero-Page Indirect-adresseermethode voor de toegang te gebruiken, en de instructie direct te laten voorafgaan door een NOP-instructie (opcode $EA). Bijvoorbeeld:

```
NOP
LDA ($45),Z
```

Als u de ACME-assembler of een andere assembler gebruikt die de 45GS02-uitbreidingen ondersteunt, kunt u in plaats daarvan vierkante haakjes gebruiken om aan te geven dat u een flat-memory-bewerking uitvoert. Dergelijke assemblers voegen de $EA-prefix automatisch voor u in. Bijvoorbeeld:

```
LDA [$45],Z
```

Ongeacht welk hulpmiddel u gebruikt, in dit voorbeeld zouden de vier bytes Zero-Page-geheugen op $45 - $48 lezen om een 32-bit geheugenadres te vormen, en de waarde van het Z-register hieraan optellen om het werkelijke adres te vormen waaruit wordt gelezen. De bytevolgorde in het adres is hetzelfde als bij de 6502, oftewel het rechtse (minst significante) byte van het adres wordt als eerste gelezen vanaf het eerste adres ($45 in dit geval), enzovoort, totdat het meest linkse (meest significante) byte wordt gelezen vanaf $48. Om bijvoorbeeld vanaf geheugenlocatie $12345678 te lezen, moet de inhoud van het geheugen beginnend op $45 bestaan uit 78 56 34 12.

Deze methode is veel efficiënter en ook eenvoudiger dan het gebruik van de MAP-instructie of de DMA-controller voor enkelvoudige geheugentoegangen, en is wat wij doorgaans aanbevelen. De DMA-controller kan worden gebruikt voor het verplaatsen/vullen van grotere geheugenregio's. Wij bevelen de MAP-instructie alleen aan voor het banken van code, of in zeldzame situaties waarbij uitgebreide toegang tot een klein geheugengebied vereist is en de extra cycli voor het lezen van de 32-bit-adressen problematisch zijn.

Virtueel 32-bit-register

De 45GS02 maakt het mogelijk om zijn vier algemene registers, A, X, Y en Z (A is LSB, Z is MSB), als één enkel virtueel 32-bit-register te gebruiken, ook wel het *Q-pseudoregister* genoemd. Dit kan veelvoorkomende bewerkingen aanzienlijk vereenvoudigen en versnellen, en helpt veelvoorkomende programmeerfouten te voorkomen. Het optellen van twee 16-bit- of 32-bit-waarden kan bijvoorbeeld eenvoudig worden uitgevoerd met iets als:

```
; Wis de carry v  r het optellen, zoals gebruikelijk
CLC
; Laat twee NEG-instructies voorafgaan om de virtuele 32-bit-registermodus te sele
NEG
NEG
LDA $1234  ; Laadt de inhoud van $1234-$1237 in respectievelijk A, X, Y en Z
; En nogmaals, voor de optelling
NEG
NEG
ADC $1238  ; Telt de inhoud van $1238-$123B op
; Het resultaat van de optelling staat nu in A, X, Y en Z.
; En kan geheel of gedeeltelijk worden weggeschreven

; Om het geheel weg te schrijven is wederom de NEG + NEG-prefix nodig
NEG
NEG
STA $123C ; Schrijft het geheel weg naar $123C-$123F

; Of om de laagste bytes weg te schrijven, kan eenvoudigweg de inhoud van A en X r
STA $1240
STX $1241
```

Deze aanpak werkt met de LDA-, STA-, ADC-, SBC-, CMP-, EOR-, AND-, BIT-, ORA-, ASL-, ASR-, LSR-, ROL-, ROR-, INC- en DEC-instructies. Indien u ACME of een andere 45GS02-ondersteunende assembler gebruikt, kunt u in plaats daarvan de nieuwe **LDQ**, **STQ**, **ADCQ**, **SBCQ**, **CPQ**, **EORQ**, **ANDQ**, **BITQ**, **ORQ**, **ASLQ**, **ASRQ**, **LSRQ**, **ROLQ**, **RORQ**, **INQ** en **DEQ** mnemonics gebruiken. Het vorige voorbeeld wordt dan:

```
; Wis de carry v  r het optellen, zoals gebruikelijk
CLC
LDQ  $1234  ; Laadt de inhoud van $1234-$1237 in respectievelijk A, X, Y en Z
ADCQ $1238  ; Telt de inhoud van $1238-$123B op
; Het resultaat van de optelling staat nu in A, X, Y en Z.
; En kan geheel of gedeeltelijk worden weggeschreven

STQ $123C ; Schrijft het geheel weg naar $123C-$123F

; Of om de laagste bytes weg te schrijven, kan eenvoudigweg de inhoud van A en X r
STA $1240
STX $1241
```

De virtuele 32-bit-adresseermethode werkt met elke adresseermethode. Geïndexeerde adresseermethoden, waarbij X, Y of Z bij het adres wordt opgeteld, moeten echter met voorzichtigheid worden gebruikt, omdat deze registers in feite een deel van een 32-bit-waarde kunnen bevatten.

De uitzondering is de Zero-Page Indirect Z-geïndexeerde adresseermethode: in dit geval wordt het Z-register NIET bij het doeladres opgeteld (met uitzondering van de LDQ-opcode), in tegenstelling tot wat normaal het geval is. Hierdoor kan het virtuele 32-bit-register worden gebruikt met platte-geheugentoegang met de gecombineerde prefix **NEG NEG NOP** vóór de instructie, zodat een 32-bit-waarde ergens in het geheugen met één instructie kan worden benaderd.

Merk op dat het virtuele 32-bit-register niet kan worden gebruikt in de immediate-modus, bijvoorbeeld om een constante in de vier algemene registers te laden of om een constante waarde op te tellen of af te trekken. Dit dient om problemen met instructies van variabele lengte te voorkomen.

Voor LDQ en STQ bespaart dit hoogstens één byte ten opzichte van LDA #$nn ... LDZ #$nn, en zou het niet sneller zijn. Voor veel veelvoorkomende waarden, zoals #$00000000, bestaan er zelfs verkortingen, zoals:

```
LDA #$00
TAX
TAY
TAZ
```

Indien u een 32-bit-immediate-waarde moet optellen of aftrekken, kan dit vereisen dat u de argumenten herschikt of andere kleine aanpassingen uitvoert. Om bijvoorbeeld de som van de inhoud van het geheugen en een immediate-waarde te berekenen, kunt u de A-, X-, Y- en Z-registers laden met de immediate-waarde en vervolgens **ADCQ** gebruiken met het geheugenadres:

```
; Haal de immediate-waarde #$12345678 in Q
LDA #$78
LDX #$56
LDY #$34
LDZ #$12
; Telt de inhoud van de geheugenlocaties $1234-$1237 op
NEG
NEG
ADC $1234
; Slaat het resultaat op in $1234-$1237
NEG
NEG
STA $1234
```

Wanneer u opnieuw de ACME- of een andere 45GS02-ondersteunende assembler gebruikt, kan dit compacter en duidelijker worden geschreven als volgt. Merk op dat in beide gevallen dezelfde bytevolgorde van machinetaal wordt geproduceerd en het programma hetzelfde aantal cycli zal kosten om uit te voeren.

```
; Haal de immediate-waarde #$12345678 in Q
LDA #$78
LDX #$56
LDY #$34
LDZ #$12
; Telt de inhoud van de geheugenlocaties $1234-$1237 op
ADCQ $1234
; Slaat het resultaat op in $1234-$1237
STQ $1234
```

IN HET GEHEUGEN GEMAPTE REGISTERS VAN DE C64-CPU

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
00	0	PORTDDR							
01	1	PORT					C64MAP		

- **C64MAP** Geheugenmapper van de processorport
- **PORT** 6510/45GS10 CPU-portgegevens
- **PORTDDR** 6510/45GS10 CPU-port-DDR. Schrijf 65 om de 40MHz-modus af te dwingen, of 64 om de normale snelheidscontroles weer in te schakelen.

NIEUWE IN HET GEHEUGEN GEMAPTE CPU-REGISTERS

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D640	54848	HTRAP00							
D641	54849	HTRAP01							
D642	54850	HTRAP02							
D643	54851	HTRAP03							
D644	54852	HTRAP04							
D645	54853	HTRAP05							
D646	54854	HTRAP06							
D647	54855	HTRAP07							
D648	54856	HTRAP08							
D649	54857	HTRAP09							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D64A	54858	HTRAP0A							
D64B	54859	HTRAP0B							
D64C	54860	HTRAP0C							
D64D	54861	HTRAP0D							
D64E	54862	HTRAP0E							
D64F	54863	HTRAP0F							
D650	54864	HTRAP10							
D651	54865	HTRAP11							
D652	54866	HTRAP12							
D653	54867	HTRAP13							
D654	54868	HTRAP14							
D655	54869	HTRAP15							
D656	54870	HTRAP16							
D657	54871	HTRAP17							
D658	54872	HTRAP18							
D659	54873	HTRAP19							
D65A	54874	HTRAP1A							
D65B	54875	HTRAP1B							
D65C	54876	HTRAP1C							
D65D	54877	HTRAP1D							
D65E	54878	HTRAP1E							
D65F	54879	HTRAP1F							
D660	54880	HTRAP20							
D661	54881	HTRAP21							
D662	54882	HTRAP22							
D663	54883	HTRAP23							
D664	54884	HTRAP24							
D665	54885	HTRAP25							
D666	54886	HTRAP26							
D667	54887	HTRAP27							
D668	54888	HTRAP28							
D669	54889	HTRAP29							
D66A	54890	HTRAP2A							
D66B	54891	HTRAP2B							
D66C	54892	HTRAP2C							
D66D	54893	HTRAP2D							
D66E	54894	HTRAP2E							
D66F	54895	HTRAP2F							
D670	54896	HTRAP30							
D671	54897	HTRAP31							
D672	54898	HTRAP32							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D673	54899	HTRAP33							
D674	54900	HTRAP34							
D675	54901	HTRAP35							
D676	54902	HTRAP36							
D677	54903	HTRAP37							
D678	54904	HTRAP38							
D679	54905	HTRAP39							
D67A	54906	HTRAP3A							
D67B	54907	HTRAP3B							
D67C	54908	HTRAP3C							
D67D	54909	HTRAP3D							
D67E	54910	HTRAP3E							
D67F	54911	HTRAP3F							
D710	55056	-		BADEXTRA		BRCOST	-	SLIEN	BADLEN
D7EF	55279	HWRNGRAND							
D7FA	55290	FRAMECOUNT							
D7FB	55291	-						CARTEN	-
D7FD	55293	NOEX-ROM	NOGAME	-					POWEREN
D7FE	55294	HWRNG-NOTRDY	-	SLOWS-DRAM	SELS-DRAM	SLOWCACHE-ADV	SLOWCA-CHE	OCEANA	PRE-FETCH

- **BADEXTRA** Kosten van badlines minus 40. D.w.z. 00=40 cycli, 11 = 43 cycli.
- **BADLEN** Schakelt badline-emulatie in
- **BRCOST** 1=extra cyclus(sen) rekenen voor genomen vertakkingen
- **CARTEN** 1=cartridges inschakelen
- **FRAMECOUNT** Telt het aantal verstreken videoframes
- **HTRAPXX** Schrijven activeert een hypervisor-trap $XX
- **HWRNGNOTRDY** Echte RNG-willekeurig getal van de hardware nog niet gereed (alleen-lezen)
- **HWRNGRAND** Echt willekeurig getal van de hardware-RNG (lezen activeert de generatie)
- **NOEXROM** Overschrijving voor /EXROM: moet 0 zijn om het /EXROM-signaal in te schakelen
- **NOGAME** Overschrijving voor /GAME: moet 0 zijn om het /GAME-signaal in te schakelen
- **OCEANA** Schakelt emulatie van cartridge van het type Ocean A in

- **POWEREN** Zet op nul om de computer uit te schakelen op ondersteunde systemen. ALLEEN-SCHRIJVEN.
- **PREFETCH** Schakelt de prefetchlogica voor uitgebreid RAM in
- **SELSDRAM** Selecteert SDRAM in plaats van HyperRAM voor Attic RAM waar beschikbaar
- **SLIEN** Schakelt interrupts in stijl 6502 (langzaam, 7 cycli) in
- **SLOWCACHE** Schakelt langzamere regelcaching voor Attic RAM in
- **SLOWCACHEADV** Schakelt de geavanceerde langzamere regelcaching voor Attic RAM in
- **SLOWSDRAM** Selecteert de langzame (81MHz) SDRAM-klok

MEGA65 CPU-VERSNELLINGSREGISTERS VOOR WISKUNDE

Elke MEGA65 bevat een gecombineerde 32-bit-hardwarevermenigvuldiger en -deler. Dit apparaat neemt twee 32-bit-invoeren, **MULTINA** en **MULTINB**, en berekent tegelijkertijd:

- het 64-bit-product **MULTOUT** van **MULTINA** en **MULTINB**
- het 32-bit-gehele deel **DIVOUT**(4-7) van **MULTINA** gedeeld door **MULTINB**
- het 32-bit-breukdeel **DIVOUT**(0-3) van **MULTINA** gedeeld door **MULTINB**

De uitvoer wordt altijd op basis van de invoer bijgewerkt, zodat er geen speciale actie nodig is bij het wijzigen van de invoer. De vermenigvuldiger heeft 1 cyclus nodig om te berekenen, zodat het bijgewerkte resultaat onmiddellijk beschikbaar is (een **MULBUSY**-bit is gedefinieerd, maar deze zal momenteel niet worden gezet). De hardwaredeler kan echter tot 20 cycli kosten, afhankelijk van de specifieke invoer. De programmeur moet de **DIVBUSY**-bit controleren indien de deler nog bezig is met berekenen:

```
loop:   BIT $D70F   ; brengt de DIVBUSY-bit over naar de N-vlag
        BMI loop    ; zolang deze is gezet, moet worden gewacht
```

De MEGA65 zal ook een programmeerbare rekenunit bevatten, die helpt bij het versnellen van de berekening van vastekommatformules. Deze is momenteel uitgeschakeld en zal verder worden gedocumenteerd zodra deze beschikbaar komt (adressen $D780 - $D7E3).

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D70F	55055	DIVBUSY	MULBUSY	–					
D768	55144	DIVOUT							
D769	55145	DIVOUT							
D76A	55146	DIVOUT							
D76B	55147	DIVOUT							
D76C	55148	DIVOUT							
D76D	55149	DIVOUT							
D76E	55150	DIVOUT							
D76F	55151	DIVOUT							
D770	55152	MULTINA							
D771	55153	MULTINA							
D772	55154	MULTINA							
D773	55155	MULTINA							
D774	55156	MULTINB							
D775	55157	MULTINB							
D776	55158	MULTINB							
D777	55159	MULTINB							
D778	55160	MULTOUT							
D779	55161	MULTOUT							
D77A	55162	MULTOUT							
D77B	55163	MULTOUT							
D77C	55164	MULTOUT							
D77D	55165	MULTOUT							
D77E	55166	MULTOUT							
D77F	55167	MULTOUT							
D780	55168	MATHIN0							
D781	55169	MATHIN0							
D782	55170	MATHIN0							
D783	55171	MATHIN0							
D784	55172	MATHIN1							
D785	55173	MATHIN1							
D786	55174	MATHIN1							
D787	55175	MATHIN1							
D788	55176	MATHIN2							
D789	55177	MATHIN2							
D78A	55178	MATHIN2							
D78B	55179	MATHIN2							
D78C	55180	MATHIN3							
D78D	55181	MATHIN3							
D78E	55182	MATHIN3							
D78F	55183	MATHIN3							
D790	55184	MATHIN4							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D791	55185	MATHIN4							
D792	55186	MATHIN4							
D793	55187	MATHIN4							
D794	55188	MATHIN5							
D795	55189	MATHIN5							
D796	55190	MATHIN5							
D797	55191	MATHIN5							
D798	55192	MATHIN6							
D799	55193	MATHIN6							
D79A	55194	MATHIN6							
D79B	55195	MATHIN6							
D79C	55196	MATHIN7							
D79D	55197	MATHIN7							
D79E	55198	MATHIN7							
D79F	55199	MATHIN7							
D7A0	55200	MATHIN8							
D7A1	55201	MATHIN8							
D7A2	55202	MATHIN8							
D7A3	55203	MATHIN8							
D7A4	55204	MATHIN9							
D7A5	55205	MATHIN9							
D7A6	55206	MATHIN9							
D7A7	55207	MATHIN9							
D7A8	55208	MATHINA							
D7A9	55209	MATHINA							
D7AA	55210	MATHINA							
D7AB	55211	MATHINA							
D7AC	55212	MATHINB							
D7AD	55213	MATHINB							
D7AE	55214	MATHINB							
D7AF	55215	MATHINB							
D7B0	55216	MATHINC							
D7B1	55217	MATHINC							
D7B2	55218	MATHINC							
D7B3	55219	MATHINC							
D7B4	55220	MATHIND							
D7B5	55221	MATHIND							
D7B6	55222	MATHIND							
D7B7	55223	MATHIND							
D7B8	55224	MATHINE							
D7B9	55225	MATHINE							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D7BA	55226	MATHINE							
D7BB	55227	MATHINE							
D7BC	55228	MATHINF							
D7BD	55229	MATHINF							
D7BE	55230	MATHINF							
D7BF	55231	MATHINF							
D7C0	55232	UNIT0INB				UNIT0INA			
D7C1	55233	UNIT1INB				UNIT1INA			
D7C2	55234	UNIT2INB				UNIT2INA			
D7C3	55235	UNIT3INB				UNIT3INA			
D7C4	55236	UNIT4INB				UNIT4INA			
D7C5	55237	UNIT5INB				UNIT5INA			
D7C6	55238	UNIT6INB				UNIT6INA			
D7C7	55239	UNIT7INB				UNIT7INA			
D7C8	55240	UNIT8INB				UNIT8INA			
D7C9	55241	UNIT9INB				UNIT9INA			
D7CA	55242	UNITAINB				UNITAINA			
D7CB	55243	UNITBINB				UNITBINA			
D7CC	55244	UNITCINB				UNITCINA			
D7CD	55245	UNITDINB				UNITDINA			
D7CE	55246	UNITEINB				UNITEINA			
D7CF	55247	UNITFINB				UNITFINA			
D7D0	55248	U0LATCH	U0MLADD	U0HIOUT	U0-LOWOUT	UNIT0OUT			
D7D1	55249	U1LATCH	U1MLADD	U1HIOUT	U1-LOWOUT	UNIT1OUT			
D7D2	55250	U2LATCH	U2MLADD	U2HIOUT	U2-LOWOUT	UNIT2OUT			
D7D3	55251	U3LATCH	U3MLADD	U3HIOUT	U3-LOWOUT	UNIT3OUT			
D7D4	55252	U4LATCH	U4MLADD	U4HIOUT	U4-LOWOUT	UNIT4OUT			
D7D5	55253	U5LATCH	U5MLADD	U5HIOUT	U5-LOWOUT	UNIT5OUT			
D7D6	55254	U6LATCH	U6MLADD	U6HIOUT	U6-LOWOUT	UNIT6OUT			
D7D7	55255	U7LATCH	U7MLADD	U7HIOUT	U7-LOWOUT	UNIT7OUT			
D7D8	55256	U8LATCH	U8BSADD	U8HIOUT	U8-LOWOUT	UNIT8OUT			
D7D9	55257	U9LATCH	U9BSADD	U9HIOUT	U9-LOWOUT	UNIT9OUT			
D7DA	55258	UALATCH	UABSADD	UAHIOUT	UA-LOWOUT	UNITAOUT			

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D7DB	55259	UBLATCH	UBBSADD	UBHIOUT	UB-LOWOUT	UNITBOUT			
D7DC	55260	UCLATCH	UCDVADD	UCHIOUT	UC-LOWOUT	UNITCOUT			
D7DD	55261	UDLATCH	UDDVADD	UDHIOUT	UD-LOWOUT	UNITDOUT			
D7DE	55262	UELATCH	UEDVADD	UEHIOUT	UE-LOWOUT	UNITEOUT			
D7DF	55263	UFLATCH	UFDVADD	UFHIOUT	UF-LOWOUT	UNITFOUT			
D7E0	55264	LATCHINT							
D7E1	55265	-						CALCEN	WREN
D7E2	55266	RESERVED							
D7E3	55267	RESERVED							

- **CALCEN** Maakt het vastleggen van uitvoerwaarden van rekenunits in de wiskunderegisters mogelijk (uitwissen pauzeert effectief iteratieve formules)
- **DIVBUSY** Gezet indien de hardwaredeler bezig is
- **DIVOUT** 64-bit-uitvoer van MULTINA ÷ MULTINB
- **LATCHINT** Vastleginterval voor vastgelegde uitvoer (in CPU-cycli)
- **MATHINX** 32-bit-invoer X van de rekenunit
- **MULBUSY** Gezet indien de hardwarevermenigvuldiger bezig is
- **MULTINA** Vermenigvuldiger-invoer A / teller van de deler (32 bit)
- **MULTINB** Vermenigvuldiger-invoer B / noemer van de deler (32 bit)
- **MULTOUT** 64-bit-uitvoer van MULTINA × MULTINB
- **RESERVED** Gereserveerd
- **UNITXINA** Selecteert welke van de 16 32-bit-wiskunderegisters invoer A is voor rekenfunctieunit X.
- **UNITXINB** Selecteert welke van de 16 32-bit-wiskunderegisters invoer B is voor rekenfunctieunit X.
- **UNITXOUT** Selecteert welke van de 16 32-bit-wiskunderegisters de uitvoer van rekenfunctieunit X ontvangt
- **UXBSADD** Indien gezet, treedt rekenfunctieunit Y op als een 32-bit-opteller in plaats van een 32-bit-barrelshifter.
- **UXDVADD** Indien gezet, treedt rekenfunctieunit X op als een 32-bit-opteller in plaats van een 32-bit-deler.

- **UXHIOUT** Indien gezet, wordt de hoge helft van de uitvoer van rekenfunctieunit X naar het wiskunderegister UNITXOUT geschreven.
- **UXLATCH** Indien gezet, wordt de uitvoer van rekenfunctieunit X vastgelegd.
- **UXLOWOUT** Indien gezet, wordt de lage helft van de uitvoer van rekenfunctieunit X naar het wiskunderegister UNITXOUT geschreven.
- **UXMLADD** Indien gezet, treedt rekenfunctieunit X op als een 32-bit-opteller in plaats van een 32-bit-vermenigvuldiger.
- **WREN** Maakt het instellen van de wiskunderegisters mogelijk (moet normaal gezet zijn)

MEGA65-HYPERVISORMODUS

Reset

Bij inschakeling of reset start de MEGA65 in de hypervisormodus op en verwacht een programma in het 16KB-hypervisorgeheugen, en begint het uitvoeren van instructies op adres $8100. Normaal bevindt zich op dit adres een JMP-instructie die naar een routine voor reset springt. Dat wil zeggen, de 45GS02 gebruikt niet de normale resetvector van de 6502. De functie daarvan wordt geëmuleerd door het Hyppo-hypervisorprogramma, dat het adres ophaalt uit de resetvector van de 6502 in het geladen clientbesturingssysteem wanneer het de hypervisormodus verlaat.

Het hypervisorgeheugen wordt bij reset automatisch op $8000 - $BFFF gemapt. Dit speciale geheugen kan niet worden gemapt of op enige andere manier worden benaderd, behalve wanneer de processor zich in de hypervisormodus bevindt. Het kan echter altijd worden benaderd vanuit de seriële monitor-/debuggerinterface via zijn 28-bit-adres, $FFF8000 - $FFFBFFF. Dit beschermt het tegen onbedoelde of kwaadwillende toegang vanuit een gastbesturingssysteem.

De hypervisormodus in- en uitstappen

Het betreden van de hypervisor vindt plaats wanneer een van de volgende gebeurtenissen optreedt:

- **Inschakelen** Wanneer de MEGA65 voor het eerst wordt ingeschakeld.
- **Reset** Wanneer de resetlijn laag wordt gemaakt of een reset door een watchdog wordt geactiveerd.
- **SYSCALL-register benaderd** De registers $D640 - $D67F in de MEGA65 I/O-modus activeren SYSCALLs wanneer ze worden benaderd. Dit is bedoeld als het mechanisme waarmee een clientbesturingssysteem of -proces de aandacht van de hypervisor of het besturingssysteem verzoekt.

- **Paginafout** Op MEGA65's met virtueel geheugen veroorzaakt een paginafout een trap naar de hypervisormodus.
- **Bepaalde toetsenbordgebeurtenissen** Het langer dan 0.5 seconden ingedrukt houden van RESTORE, of de combinatie van de  - en TAB-toets, activeert een trap naar de hypervisor. Doorgaans wordt het eerste gebruikt om het Freeze-menu te starten en het tweede om de weergave van de debuginterface te wisselen.
- **Benaderen van virtueel gemaakte I/O-apparaten** Indien bijvoorbeeld de F011 (interne 3.5"-schijfstationcontroller) is gevirtualiseerd, zal het pogingen om sectoren met dit apparaat te lezen of schrijven activeren tot trappen naar de hypervisor.
- **Uitvoeren van een instructie die de CPU zou blokkeren** Een aantal niet-gedocumenteerde opcodes op de 6502 zal de CPU blokkeren. Op de MEGA65 zal de computer in plaats daarvan een trap naar de hypervisor activeren. Dit kan worden gebruikt om alternatieve instructiegedragingen te implementeren, of eenvoudigweg om de gebruiker te melden dat er iets fout is gegaan.
- **Bepaalde speciale gebeurtenissen** Sommige apparaten kunnen hypervisor-niveau-onderbrekingen genereren. Deze worden geïmplementeerd als trappen naar de hypervisor.

De 45GS02 verwerkt al deze gebeurtenissen op een vergelijkbare manier intern:

1. Het SYSCALL- of trap-adres wordt berekend, op basis van de gebeurtenis.
2. De inhoud van alle CPU-registers wordt opgeslagen in de virtualisatiebesturings-registers.
3. De geheugenindeling van de hypervisormodus wordt geactiveerd, de decimale vlag van de CPU en de special purpose-registers worden allemaal op geschikte waarden gezet. De inhoud van de A-, X-, Y- en Z-registers en de meeste andere CPU-vlaggen blijven behouden, zodat die benaderd kunnen worden vanuit de SYSCALL/trap-handlerroutine van de hypervisor, zonder ze te moeten laden, wat een paar cycli per aanroep bespaart.
4. De hypervisormodus-vlag wordt gezet en de programmateller (PC) wordt op het berekende adres gezet.

Al het bovenstaande gebeurt in één CPU-cyclus, d.w.z. in 25 nanoseconden. Terugkeren van een SYSCALL of trap bestaat eenvoudigweg uit het schrijven naar $D67F, wat 125 nanoseconden vereist, voor een totale overhead van 150 nanoseconden. Dit geeft de SYSCALL-prestaties van de MEGA65 die wedijveren met - zelfs sneller zijn dan - zelfs de snelste moderne computers, waar de systeemaanroeplatentie doorgaans honderden tot tienduizenden cycli bedraagt [1].

Hypervisor-geheugenindeling

Het hypervisorgeheugen heeft een grootte van 16KB. De eerste 512 bytes zijn gereserveerd voor SYSCALL- en systeem-trap-ingangspunten, met vier bytes voor elk. Het reset-ingangspunt is bijvoorbeeld $8100 - $8100 + 3 = $8100 - $8103. Dit laat 4 bytes toe voor een instructie, doorgaans een JMP-instructie, gevolgd door een NOP om deze tot 4 bytes op te vullen.

De volledige lijst van SYSCALLs en traps is:

HEX	DEC	Naam	Omschrijving
8000	32768	SYSCALL00	SYSCALL 0 ingangspunt
8004	32772	SYSCALL01	SYSCALL 1 ingangspunt
8008	32776	SYSCALL02	SYSCALL 2 ingangspunt
800C	32780	SYSCALL03	SYSCALL 3 ingangspunt
8010	32784	SYSCALL04	SYSCALL 4 ingangspunt
8014	32788	SYSCALL05	SYSCALL 5 ingangspunt
8018	32792	SYSCALL06	SYSCALL 6 ingangspunt
801C	32796	SYSCALL07	SYSCALL 7 ingangspunt
8020	32800	SYSCALL08	SYSCALL 8 ingangspunt
8024	32804	SYSCALL09	SYSCALL 9 ingangspunt
8028	32808	SYSCALL0A	SYSCALL 10 ingangspunt
802C	32812	SYSCALL0B	SYSCALL 11 ingangspunt
8030	32816	SYSCALL0C	SYSCALL 12 ingangspunt
8034	32820	SYSCALL0D	SYSCALL 13 ingangspunt
8038	32824	SYSCALL0E	SYSCALL 14 ingangspunt
803C	32828	SYSCALL0F	SYSCALL 15 ingangspunt
8040	32832	SYSCALL10	SYSCALL 16 ingangspunt
8044	32836	SECURENTR	Ingangspunt van de trap naar de beveiligde container
8048	32840	SECUREXIT	Verlaten van de beveiligde container trap-ingangspunt.
804C	32844	SYSCALL13	SYSCALL 19 ingangspunt
8050	32848	SYSCALL14	SYSCALL 20 ingangspunt
8054	32852	SYSCALL15	SYSCALL 21 ingangspunt
8058	32856	SYSCALL16	SYSCALL 22 ingangspunt
805C	32860	SYSCALL17	SYSCALL 23 ingangspunt
8060	32864	SYSCALL18	SYSCALL 24 ingangspunt
8064	32868	SYSCALL19	SYSCALL 25 ingangspunt
8068	32872	SYSCALL1A	SYSCALL 26 ingangspunt
806C	32876	SYSCALL1B	SYSCALL 27 ingangspunt
8070	32880	SYSCALL1C	SYSCALL 28 ingangspunt
8074	32884	SYSCALL1D	SYSCALL 29 ingangspunt
8078	32888	SYSCALL1E	SYSCALL 30 ingangspunt
807C	32892	SYSCALL1F	SYSCALL 31 ingangspunt

Vervolgd ...

...Vervolgd

HEX	DEC	Naam	Omschrijving
8080	32896	SYSCALL20	SYSCALL 32 ingangspunt
8084	32900	SYSCALL21	SYSCALL 33 ingangspunt
8088	32904	SYSCALL22	SYSCALL 34 ingangspunt
808C	32908	SYSCALL23	SYSCALL 35 ingangspunt
8090	32912	SYSCALL24	SYSCALL 36 ingangspunt
8094	32916	SYSCALL25	SYSCALL 37 ingangspunt
8098	32920	SYSCALL26	SYSCALL 38 ingangspunt
809C	32924	SYSCALL27	SYSCALL 39 ingangspunt
80A0	32928	SYSCALL28	SYSCALL 40 ingangspunt
80A4	32932	SYSCALL29	SYSCALL 41 ingangspunt
80A8	32936	SYSCALL2A	SYSCALL 42 ingangspunt
80AC	32940	SYSCALL2B	SYSCALL 43 ingangspunt
80B0	32944	SYSCALL2C	SYSCALL 44 ingangspunt
80B4	32948	SYSCALL2D	SYSCALL 45 ingangspunt
80B8	32952	SYSCALL2E	SYSCALL 46 ingangspunt
80BC	32956	SYSCALL2F	SYSCALL 47 ingangspunt
80C0	32960	SYSCALL30	SYSCALL 48 ingangspunt
80C4	32964	SYSCALL31	SYSCALL 49 ingangspunt
80C8	32968	SYSCALL32	SYSCALL 50 ingangspunt
80CC	32972	SYSCALL33	SYSCALL 51 ingangspunt
80D0	32976	SYSCALL34	SYSCALL 52 ingangspunt
80D4	32980	SYSCALL35	SYSCALL 53 ingangspunt
80D8	32984	SYSCALL36	SYSCALL 54 ingangspunt
80DC	32988	SYSCALL37	SYSCALL 55 ingangspunt
80E0	32992	SYSCALL38	SYSCALL 56 ingangspunt
80E4	32996	SYSCALL39	SYSCALL 57 ingangspunt
80E8	33000	SYSCALL3A	SYSCALL 58 ingangspunt
80EC	33004	SYSCALL3B	SYSCALL 59 ingangspunt
80F0	33008	SYSCALL3C	SYSCALL 60 ingangspunt
80F4	33012	SYSCALL3D	SYSCALL 61 ingangspunt
80F8	33016	SYSCALL3E	SYSCALL 62 ingangspunt
80FC	33020	SYSCALL3F	SYSCALL 63 ingangspunt
8100	33024	RESET	Inschakel-/reset-ingangspunt
8104	33028	PAGFAULT	Paginafout-ingangspunt (momenteel niet gebruikt)
8108	33032	RESTORKEY	Trap-ingangspunt voor lang indrukken van de RESTORE-toets
810C	33036	ALTTABKEY	ALT+TAB-trap-ingangspunt
8110	33040	VF011RD	Trap-ingangspunt voor virtueel gelezen F011-schijf
8114	33044	VF011WR	Trap-ingangspunt voor virtueel geschreven F011-schijf

Vervolgd ...

...Vervolgd

HEX	DEC	Naam	Omschrijving
8118	33048	BREAKPT	CPU-breukpunt aangetroffen
811C - 81FB	33048 - 33275	RESERVED	Gereserveerd trap-ingangspunt
81FC	33276	CPUKIL	KIL-instructie in 6502-modus trap-ingangspunt

Het resterende deel van het 16KB-hypervisorgeheugen staat ter beschikking van de programmeur, maar doorgaans worden de laatste 512 bytes gebruikt voor de stack en de zero page, wat leidt tot de volgende algemene geheugenindeling:

HEX	DEC	Omschrijving
8000 - 81FF	32768 - 33279	SYSCALL- en trap-ingangspunten
8200 - BDFF	33280 - 48639	Beschikbaar voor hypervisor- of besturingssysteem-programma
8E00 - BEFF	48640 - 48895	Processorstack voor hypervisor of besturingssysteem
8F00 - BFFF	48896 - 49151	Zero page-opslag van de processor voor hypervisor of besturingssysteem

De stack wordt gebruikt om het retouradres van functieaanroepen op te slaan. De zero page-opslag wordt doorgaans gebruikt om variabelen en andere gegevens voor korte termijn op te slaan, zoals gebruikelijk op de 6502.

Hypervisor-virtualisatieregisters

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D640	54848	REGA							
D641	54849	REGX							
D643	54851	REGZ							
D644	54852	REGB							
D645	54853	SPL							
D646	54854	SPH							
D647	54855	PFLAGS							
D648	54856	PCL							
D649	54857	PCH							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D64A	54858	MAPLO0							
D64B	54859	MAPLO1							
D64C	54860	MAPHI0							
D64D	54861	MAPHI1							
D64E	54862	MAPLOMB							
D64F	54863	MAPHIMB							
D650	54864	PORT00							
D651	54865	PORT01							
D652	54866	–					EXSID	VICMODE	
D653	54867	DMASRCMB							
D654	54868	DMADSTMB							
D655	54869	DMALADDR0							
D656	54870	DMALADDR1							
D657	54871	DMALADDR2							
D658	54872	DMALADDR3							
D659	54873	–						VFLOP1	VFLOP0
D65D	54877	VIRTPAGE0		–	VPG-ACTIVE	VPGDIRTY			
D65E	54878	VIRTPAGE1							
D65F	54879	VIRTPAGE2							
D660	54880	VPG0LOG0							
D661	54881	VPG0LOG1							
D662	54882	VPG0PHY0							
D663	54883	VPG0PHY1							
D664	54884	VPG1LOG0							
D665	54885	VPG1LOG1							
D666	54886	VPG1PHY0							
D667	54887	VPG1PHY0							
D668	54888	VPG2LOG0							
D669	54889	VPG2LOG1							
D66A	54890	VPG2PHY0							
D66B	54891	VPG2PHY0							
D66C	54892	VPG3LOG0							
D66D	54893	VPG3LOG1							
D66E	54894	VPG3PHY0							
D66F	54895	VPG3PHY0							
D670	54896	GEORAMBASE							
D671	54897	GEORAMMASK							
D672	54898	–	MATRIXEN	–					
D67C	54908	UARTDATA							
D67D	54909	WATCHDOG							
D67E	54910	HICKED							

Vervolgd ...

...Vervolgd

HEX	DEC	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
D67F	54911	ENTEREXIT							

- **ASCFAST** Hypervisor: schakelt de ASC/DIN-CAPS-LOCK-toets in om de CPU-vertraging in de C64/C128/C65-modus in of uit te schakelen
- **CPUFAST** Hypervisor: dwingt de CPU naar 40.5MHz af voor de userland (de userland kan dit overschrijven via POKE0)
- **DMADSTMB** Hypervisor: DMAgic-bestemmings-MB
- **DMALADDR0** Hypervisor: bits 0-7 van het DMAGic-lijstadres
- **DMALADDR1** Hypervisor: bits 15-8 van het DMAGic-lijstadres
- **DMALADDR2** Hypervisor: bits 23-16 van het DMAGic-lijstadres
- **DMALADDR3** Hypervisor: bits 27-24 van het DMAGic-lijstadres
- **DMASRCMB** Hypervisor: DMAgic-bron-MB
- **ENTEREXIT** Schrijven activeert het terugkeren vanuit de hypervisor
- **EXSID** 0=interne SID's gebruiken, 1=externe(1) SID's gebruiken
- **F4502** Hypervisor: dwingt de CPU naar de 4502-persoonlijkheid af, zelfs in de C64-I/O-modus.
- **GEORAMBASE** Hypervisor: basisadres van GeoRAM (x MB)
- **GEORAMMASK** Hypervisor: adresmasker van GeoRAM (toegepast op het blok-register van GeoRAM)
- **HICKED** Hypervisor: al-geüpgraded-bit (schrijven zet dit permanent)
- **JMP32EN** Hypervisor: schakelt 32-bit JMP/JSR en dergelijke in
- **MAPHI0** Hypervisor: opslag van het MAPHI-register (hoge bits)
- **MAPHI1** Hypervisor: opslag van het MAPHI-register (lage bits)
- **MAPHIMB** Hypervisor: opslag van het megabyte-aantal van het MAPHI-register
- **MAPLO0** Hypervisor: opslag van het MAPLO-register (hoge bits)
- **MAPLO1** Hypervisor: opslag van het MAPLO-register (lage bits)
- **MAPLOMB** Hypervisor: opslag van het megabyte-aantal van het MAPLO-register
- **MATRIXEN** Schakelt de samengestelde Matrix-modus in en schakelt de UART-toegang tot de seriële monitor uit.
- **PCH** Hypervisor: opslag van het hoge PC-register

- **PCL** Hypervisor: opslag van het lage PC-register
- **PFLAGS** Hypervisor: opslag van het P-register
- **PIRQ** Hypervisor: vlag die aangeeft of er een IRQ in behandeling is bij het verlaten van de hypervisor / zet 1 om IRQ/NMI-uitstel voor 1.024 cycli af te dwingen bij het verlaten van de hypervisor.
- **PNMI** Hypervisor: vlag die aangeeft of er een NMI in behandeling is bij het verlaten van de hypervisor.
- **PORT00** Hypervisor: waarde van CPU-port $00
- **PORT01** Hypervisor: waarde van CPU-port $01
- **REGA** Hypervisor: opslag van het A-register
- **REGB** Hypervisor: opslag van het B-register
- **REGX** Hypervisor: opslag van het X-register
- **REGZ** Hypervisor: opslag van het Z-register
- **ROMPROT** Hypervisor: schrijfbeveiliging van de C65-ROM $20000-$3FFFF
- **RSVD** GERESERVEERD
- **SPH** Hypervisor: opslag van het SPH-register
- **SPL** Hypervisor: opslag van het SPL-register
- **UARTDATA** (schrijven) Hypervisor: schrijft seriële uitvoer naar de UART-monitor
- **VFLOP0** 1=SD-/Floppy0-toegang virtualiseren (doorgaans voor toegang via de seriële debuggerinterface)
- **VFLOP1** 1=SD-/Floppy1-toegang virtualiseren (doorgaans voor toegang via de seriële debuggerinterface)
- **VICMODE** VIC-II/VIC-III/VIC-IV-modusselectie
- **VIRTPAGE0** Hypervisor: huidig virtueel paginanummer (bits 0 - 1)
- **VIRTPAGE1** Hypervisor: huidig virtueel paginanummer (bits 2 - 9)
- **VIRTPAGE2** Hypervisor: huidig virtueel paginanummer (bits 10 - 17)
- **VPGACTIVE** Hypervisor: pagina actief
- **VPGDIRTY** Hypervisor: pagina gewijzigd
- **VPGXLOG0** Hypervisor: logische pagina van virtueel geheugenpagina X (bits 0 - 7)
- **VPGXLOG1** Hypervisor: logische pagina van virtueel geheugenpagina X (bits 8 - 15)

- **VPGXPHY0** Hypervisor: fysieke pagina van virtueel geheugenpagina X (bits 0 - 7)
- **VPGXPHY1** Hypervisor: fysieke pagina van virtueel geheugenpagina X (bits 8 - 15)
- **WATCHDOG** Hypervisor: watchdogregister: het schrijven van een willekeurige waarde wist de watchdog

Programmeren in de hypervisormodus

De eenvoudigste manier om een programma voor de hypervisormodus op de MEGA65 te schrijven, is het gebruik van KickC, een speciale versie van C die is gemaakt voor het schrijven van programma's voor 6502-klasse processors. De volgende voorbeeldprogramma's zijn afkomstig uit de meegeleverde voorbeelden van KickC. KickC produceert zeer efficiënte code en ondersteunt de hypervisormodus van de MEGA65 op eenvoudige wijze via een linkerdefinitiebestand met de volgende inhoud:

```
.file [name="%O.bin", type="bin", segments="XMega65Bin"]
.segmentdef XMega65Bin [segments="Syscall, Code, Data, Stack, Zeropage"]
.segmentdef Syscall [start=$8000, max=$81ff]
.segmentdef Code [start=$8200, min=$8200, max=$bdff]
.segmentdef Data [startAfter="Code", min=$8200, max=$bdff]
.segmentdef Stack [min=$be00, max=$beff, fill]
.segmentdef Zeropage [min=$bf00, max=$bfff, fill]
```

Dit bestand draagt de assembler van KickC op om een bestand van 16 KB te maken met het ingangspuntgebied van 512 bytes voor SYSCALL- en trap-verwijzingen aan het begin, gevolgd door code- en data-gebieden, en vervolgens de stack- en zeropage-gebieden. Het dwingt de grootte en locatie van deze velden af en zal tijdens de compilatie een foutmelding geven wanneer iets te groot is om te passen.

Wanneer dit bestand aanwezig is, kunt u een KickC-bronbestand aanmaken dat gegevensstructuren voor de SYSCALL/trap-tabel bevat:

```
// XMega65 KERNAL-ontwikkelingssjabloon
// Elke functie van het KERNAL is een functie zonder argumenten
// De functies worden in de SYSCALLS-tabel geplaatst, omringd door JMP en NOP

import "string"

// Gebruik een linkerdefinitiebestand (plaats de vorige listing in dat bestand)
#pragma link("mega65hyper.ld")

// Enkele definities van adressen en speciale waarden die dit programma gebruikt
const char* RASTER = 0xd012;
const char* VIC+MEMORY = 0xd018;
const char* SCREEN = 0x0400;
const char* BGCOL = 0xd021;
const char* COLS = 0xd800;
const char BLACK = 0;
const char WHITE = 1;

// Enige tekst om weer te geven
char[] MESSAGE = "hello world!";
```

```
void main() {
    // Initialiseer het schermgeheugen en selecteer het juiste lettertype
    *VIC+MEMORY = 0x14;
    // Vul het scherm met spaties
    memset(SCREEN, ' ', 40*25);
    // Stel de kleur van elk teken op het scherm in op wit
    memset(COLS, WHITE, 40*25);
    // Toon de "hello world!"-melding
    char* sc = SCREEN+40;  // Toon het   n regel lager op het scherm
    char* msg = MESSAGE; // De boodschap om weer te geven
    // Een eenvoudige kopieerroutine om de reeks te kopi ren
    while(*msg) {
        *sc++ = *msg++;
    }
    // Laat voor altijd twee witte lijnen als rasterbalken tonen
    while(true) {
        if(*RASTER==54 || *RASTER==66) {
            *BGCOL = WHITE;
        } else {
            *BGCOL = BLACK;
        }
    }
}

// Hier zijn een paar voorbeeld-SYSCALL-handlers die slechts een teken op het scherm
void syscall1() {
    *(SCREEN+79) = ')';
}

void syscall2() {
    *(SCREEN+78) = '(';
}

// Nu selecteren wij het SYSCALL-segment om de SYSCALL/trap-ingangspunttabel te beva
#pragma data+seg(Syscall)

// De structuur van elk ingangspunt is JMP (handleradres) + NOP.
// Wij hebben een char (xjmp) om de opcode voor de JMP-instructie te bevatten,
// en plaatsen daarna het adres van de SYSCALL/trap-handler in de volgende
// twee posities als een aanwijzer, en eindigen met de opcode van de NOP-instructie.
```

```
struct SysCall {
    char xjmp;          // Bevat $4C, de opcode van JMP $nnnn
    void()* syscall;    // Bevat het handleradres, zal het doel van de JMP zijn
    char xnop;          // Bevat $EA, de opcode van NOP
};

// Om niet steeds 0x4C en 0xEA te hoeven schrijven, defini ren wij ze als constanten
const char JMP = 0x4c;
const char NOP = 0xea;

// Nu kunnen wij een mooie tabel van maximaal 64 SYSCALL-handlers uitdrukken
// in een redelijk leesbaar en eenvoudig formaat.
// Elke regel is een instantie van de bovenstaande struct SysCall, met de JMP-
// opcodewaarde, het adres van de handler-routine en de NOP-opcodewaarde.
export struct SysCall[] SYSCALLS = {
    { JMP, &syscall1, NOP },
    { JMP, &syscall2, NOP }
    };

// In dit voorbeeld hadden wij slechts twee SYSCALLs gedefinieerd, dus in plaats van
// 62 regels te hebben, kunnen wij KickC vragen de TRAP-tabel te laten beginnen
// bij het eerstvolgende veelvoud van $100, dat wil zeggen op $8100.
export align(0x100) struct SysCall[] SYSCALL£+RESET = {
    { JMP, &main, NOP }
};
```

Wanneer u de eerste listing opslaat in een bestand met de naam mega65hyper.ld en de tweede in een bestand met de naam mega65hyper.kc, kunt u deze compileren met KickC met een commando als:

```
kickc -a mega65hyper
```

Vervolgens wordt een bestand met de naam mega65hyper.bin geproduceerd, dat u kunt uitproberen op uw MEGA65 of kunt uitvoeren in de XMega65-emulator met een commando als:

```
xmega65 -kickup mega65hyper.bin
```

HOOFDSTUK 3

45GS02 & 6502-instructiesets

- Inleiding
- Stapelbewerkingen
- Adresseermethoden
- 6502-instructieset
- De instructieset van de 4510
- Samengestelde instructies van de 45GS02

INLEIDING

De 45GS02-CPU kan in de native-modus werken, waarbij hij compatibel is met de CSG 4510, en in de 6502-compatibiliteitsmodus, waarbij niet-gedocumenteerde 6502-instructies, ook wel illegale instructies genoemd, worden ondersteund voor compatibiliteit.

> **WAARSCHUWING:** Deze functie is onvolledig en niet getest. De meeste niet-gedocumenteerde 6502-opcodes werken niet correct wanneer de 6502-persoonlijkheid is ingeschakeld.

Bij 4510-compatibiliteit ondersteunt de 45GS02 ook een aantal uitbreidingen via *samengestelde instructies*. Dit werkt door de opcode van de gewenste instructie te laten voorafgaan door een of meer *prefix-bytes*, welke instructievolgordes representeren die normaal gesproken niet voorkomen. Twee **NEG**-instructies achter elkaar functioneren bijvoorbeeld als een prefix om de 45GS02 te vertellen dat de volgende instructie op 32 bits aan gegevens werkt, in plaats van de gebruikelijke 8 bits. Dit betekent dat een 45GS02-instructiestroom eenvoudig gedecodeerd of gedassembleerd kan worden, zonder dat speciale instructielengtevlaggen ingesteld hoeven te worden, zoals het geval is bij de 65816-familie van microprocessors. Het nadeel is een langere uitvoeringstijd, omdat de 45GS02 de prefix-bytes moet overslaan.

De rest van dit hoofdstuk introduceert de adresseermethoden, instructies, opcodes en instructietijding van de 45GS02, te beginnen met de 6502-compatibiliteitsmodus, waarna de 4510-compatibiliteitsmodus en de 45GS02-uitbreidingen worden behandeld.

STAPELBEWERKINGEN

De stapel (Engels: *stack*) is een gebied in het geheugen waar u gegevens op kunt plaatsen of de laatste gegevens van kunt ophalen (pop). Elke stapelbewerking die gegevens op de stapel plaatst (push) verlaagt de stapelaanwijzer (SP, *stack pointer*) en elke stapelbewerking die gegevens van de stapel afneemt (pop) verhoogt de SP. De stapel begint namelijk aan de bovenkant van het gebied en groeit naar onderen.

Wanneer u iets tegenkomt als

STAPEL ← WAARDE

betekent dat WAARDE op de STAPEL wordt geplaatst en de SP wordt verlaagd met het aantal bytes dat WAARDE groot is. Bij het plaatsen van meer dan één byte wordt eerst de meest significante byte (MSB) geplaatst, gevolgd door de minst significante byte (LSB). (**PHW** is een uitzondering: die plaatst de LSB eerst.)

In de tegenovergestelde richting geldt voor een bewerking als

MEMORY of REGISTER ← STAPEL

dat gegevens van de STAPEL worden opgehaald en de SP met elke verwijderde byte wordt verhoogd.

ADRESSEERMETHODEN

De 45GS02 ondersteunt 34 verschillende adresseermethoden, welke hieronder worden uitgelegd. Velen hiervan lijken sterk op elkaar en zijn variaties op de standaard 6502- of 65CE02-adresseermethoden, met als uitzondering dat ze ofwel 32-bits pointers accepteren, op 32 bits aan gegevens werken, of beide.

Impliciet

Bij deze methode zijn er geen operanden, omdat de exacte functie van de instructie impliciet door de instructie zelf wordt bepaald. De `INX`-instructie verhoogt bijvoorbeeld het X-register.

Accumulator

Bij deze methode is de Accumulator de operand. Dit wordt doorgaans gebruikt om de waarde van het accumulatorregister te verschuiven, te roteren of op een andere manier aan te passen. `INC A` verhoogt bijvoorbeeld de waarde in het accumulatorregister.

Het pseudoregister Q

Bij deze methode is het pseudoregister Q de operand. Dit wordt doorgaans gebruikt om de waarde van het pseudoregister Q te verschuiven, te roteren of op een andere manier aan te passen. `ASLQ` verschuift bijvoorbeeld de waarde in het pseudoregister Q één bit naar links.

Onthoud dat het pseudoregister Q eenvoudigweg de A-, X-, Y- en Z-registers zijn die samenwerken als een virtueel 32-bits register, waarbij A de minst significante bits bevat en Z de meest significante bits. Wanneer u Q aanpast, past u de daadwerkelijke registers aan, en op dezelfde manier verandert het aanpassen van een daadwerkelijk register het desbetreffende deel van het Q-register.

Er zijn enkele gevallen waarin het gebruik van een Q-modusinstructie nuttig kan zijn voor het bewerken van de vier daadwerkelijke registers, bijvoorbeeld om alle vier de registers snel te laden of op te slaan.

Immediate-modus

In deze modus is het argument van de instructie een waarde die rechtstreeks wordt gebruikt. Dit wordt aangegeven door de waarde te laten voorafgaan door een #-teken. De meeste assemblers staan toe dat waarden worden ingevoerd in decimaal,

of in hexadecimaal door de waarde te laten voorafgaan door een $-teken, of in binair door de waarde te laten voorafgaan door een %-teken. Om bijvoorbeeld het accumulatorregister op de waarde 5 te zetten, kunt u het volgende gebruiken:

```
LDA #5
```

Het immediate-argument wordt gecodeerd als een enkel byte dat op de instructie volgt. In het bovenstaande geval zou de instructiestroom $A9 bevatten, de opcode voor LDA in immediate-modus, gevolgd door $05, de immediate-operand.

Immediate Word-modus

In deze modus is het argument een 16-bits waarde die direct wordt gebruikt. Er is slechts één instructie die deze adresseermethode gebruikt: `PHW`. Om bijvoorbeeld het woord $1234 op de stapel te plaatsen, kunt u het volgende gebruiken:

```
PHW #$1234
```

De lage byte van de immediate-waarde volgt op de opcode van de instructie. De hoge byte van de immediate-waarde volgt daarop. In het bovenstaande voorbeeld zou de instructiestroom dus $F4 $34 $12 zijn.

Base-Page-modus

In deze modus is het argument een 8-bits adres. De bovenste 8 bits van het adres worden overgenomen uit het Base-Page-register. Op 6502-processors bestaat er geen Base-Page-register; in plaats daarvan worden de bovenste 8 bits altijd op nul gezet - vandaar de naam van deze modus op de 6502: Zero-Page. Op de 45GS02 is het mogelijk dit "Zero-Page" naar elke pagina in het 64KB-geheugenbeeld van de processor te verplaatsen door het Base-Page-register met behulp van de `TAB`-instructie in te stellen. De Base-Page-modus biedt snellere toegang tot een geheugengebied van 256 bytes en gebruikt daarvoor minder instructiebytes.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
LDA $12
```

zou de waarde lezen die is opgeslagen in locatie $12 in de Base-Page en deze in het accumulatorregister plaatsen. De instructiestroom hiervoor zou $85 $12 zijn.

Base-Page Quad-modus

Deze modus is identiek aan de Base-Page-modus, behalve dat deze een 32-bits woord leest dat begint op het opgegeven adres.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
LDQ $12
```

zou de waarden lezen die zijn opgeslagen in de locaties $12 - $15 in de Base-Page en deze in het pseudoregister Q plaatsen. De instructiestroom hiervoor zou $42 $42 $85 $12 zijn. Merk op dat dit hetzelfde is als bij de Base-Page-modus (Zero-Page), met als toevoeging de twee $42-prefixbytes. Opcode $42 is normaal gesproken NEG (het negatief maken van de waarde in het A-register). Indien deze twee keer achter elkaar wordt uitgevoerd, keert de A-waarde terug naar zijn oorspronkelijke waarde. De 45GS02-processor beschikt over speciale logica om deze reeks te herkennen, zodat hij weet dat de volgende instructie met behulp van het Q Pseudo Register moet worden uitgevoerd.

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page.

Base-Page X-Indexed-modus

Deze modus is identiek aan de Base-Page-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het X-register bij op te tellen. In 6502-modus ligt het resultaat altijd in de Base-Page; dat wil zeggen dat eventuele carry als gevolg van de optelling van de lage byte naar de hoge byte van het adres wordt genegeerd. De codering van deze adresseermethode is identiek aan die van de Base-Page-modus.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
LDA $12,X
```

zou de waarde lezen die is opgeslagen in locatie ($12 + X) in de Base-Page en deze in het A-register plaatsen. De instructiestroom hiervoor zou $B5 $12 zijn.

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page.

Base-Page Quad X-Indexed-modus

Deze modus is identiek aan de Base-Page Quad-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het X-register bij op te tellen. In 6502-modus ligt het resultaat altijd in de Base-Page; dat wil zeggen dat eventuele carry als gevolg van de optelling van de lage byte naar de hoge byte van het adres wordt genegeerd. De codering van deze adresseermethode is identiek aan die van de Base-Page Quad-modus.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
DEQ $12,X
```

zou het 32-bits woord dat is opgeslagen in ($12 + X) tot en met ($15 + X) in de Base-Page met één verhogen en het in het X-register plaatsen. De instructiestroom hiervoor zou $42 $42 $D6 $12 zijn.

Merk op dat LDQ niet beschikbaar is in deze adresseermethode.

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page.
Zie de opmerking op pagina 55 voor meer informatie over Quad-modusinstructies.

Base-Page Y-Indexed-modus

Deze modus is identiek aan de Base-Page-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het Y-register bij op te tellen. In 6502-modus ligt het resultaat altijd in de Base-Page; dat wil zeggen dat eventuele carry als gevolg van de optelling van de lage byte naar de hoge byte van het adres wordt genegeerd. De codering van deze adresseermethode is identiek aan die van de Base-Page-modus.

Het argument wordt gecodeerd als een enkel byte dat direct na de opcode van de instructie komt. Bijvoorbeeld:

```
LDX $12,Y
```

zou de waarde lezen die is opgeslagen in locatie ($12 + Y) in de Base-Page en deze in het X-register plaatsen. De instructiestroom hiervoor zou $B6 $12 zijn.

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page.

Absolute modus

In deze modus is het argument een 16-bits adres. De lage 8 bits van het adres worden genomen uit het byte dat direct na de opcode van de instructie komt. De hoge 8 bits worden genomen uit het byte dat daarop volgt. Bijvoorbeeld de instructie

```
LDA $1234
```

zou de geheugenlocatie $1234 lezen en de gelezen waarde in het accumulatorregister plaatsen. Dit zou worden gecodeerd als $AD $34 $12.

Absolute Quad-modus

In deze modus is het argument een 16-bits adres. De lage 8 bits van het adres worden genomen uit het byte dat direct na de opcode van de instructie komt. De hoge 8 bits worden genomen uit het byte dat daarop volgt. Bijvoorbeeld de instructie

```
LDQ $1234
```

zou de geheugenlocaties $1234 - $1237 lezen en de gelezen waarden in het Q Pseudo Register plaatsen. Dit zou worden gecodeerd als $42 $42 $AD $34 $12.

Zie de opmerking op pagina 55 voor meer informatie over Quad-modusinstructies.

Absolute X-Indexed-modus

Deze modus is identiek aan de Absolute-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het X-register bij op te tellen. Indien de indexering het adres een paginagrens doet overschrijden, dat wil zeggen indien de hoge byte van het adres verandert, kan dit een extra cyclus met zich meebrengen, afhankelijk van de processormodus en de snelheidsinstelling. De codering van deze adresseermethode is identiek aan die van de Absolute-modus. Bijvoorbeeld de instructie

```
LDA $1234,X
```

zou de geheugenlocatie ($1234 + X) lezen en de daar gelezen waarde in het A-register plaatsen. Dit zou worden gecodeerd als $BD $34 $12.

Absolute Quad X-Indexed-modus

Deze modus is identiek aan de Absolute Quad-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het X-register bij op te tellen. Indien de indexering het adres een paginagrens doet overschrijden, dat wil zeggen indien de hoge byte van het adres verandert, kan dit een extra cyclus met zich meebrengen, afhankelijk van de processormodus en de snelheidsinstelling. De codering van deze adresseermethode is identiek aan die van de Absolute Quad-modus.

Bijvoorbeeld de instructie

```
ROLQ $1234,X
```

zou de 32-bits waarde op de geheugenlocaties ($1234+X) - ($1237+X) naar links roteren en het resultaat terugschrijven naar dezezelfde geheugenlocaties. Dit zou worden gecodeerd als $42 $42 $3E $34 $12.

Zie de opmerking op pagina 55 voor meer informatie over Quad-modusinstructies.

Absolute Y-Indexed-modus

Deze modus is identiek aan de Absolute-modus, behalve dat het adres wordt gevormd door het argument te nemen en daar de waarde van het Y-register bij op te tellen. Indien de indexering het adres een paginagrens doet overschrijden, dat wil zeggen indien de hoge byte van het adres verandert, kan dit een extra cyclus met zich meebrengen, afhankelijk van de processormodus en de snelheidsinstelling. De codering van deze adresseermethode is identiek aan die van de Absolute-modus. Bijvoorbeeld de instructie

```
LDA $1234,Y
```

zou de geheugenlocatie ($1234 + Y) lezen en de daar gelezen waarde in het A-register plaatsen. Dit zou worden gecodeerd als $B9 $34 $12.

Absolute Indirect-modus

In deze modus is het 16-bits argument het adres dat naar het daadwerkelijk te lezen byte verwijst, dat wil zeggen de waarde van dat byte bevat. Indien geheugenlocatie $1234 bijvoorbeeld de waarde $78 bevat en geheugenlocatie $1235 de waarde $56, dan

```
JMP ($1234)
```

zou naar adres $5678 springen. De codering van deze adresseermethode is identiek aan die van de Absolute-modus en deze instructie zou dus worden gecodeerd als $6C $34 $12.

Absolute Indirect X-Indexed-modus

In deze modus is het 16-bits argument het adres dat naar het daadwerkelijk te lezen byte verwijst, dat wil zeggen de waarde van dat byte bevat. Deze modus is identiek aan de Absolute Indirect-modus, behalve dat de waarde van het X-register aan het pointeradres wordt toegevoegd. Indien het X-register bijvoorbeeld de waarde $04 bevat, geheugenlocatie $1238 de waarde $78 en geheugenlocatie $1239 de waarde $56, dan

```
JMP ($1234,X)
```

zou naar adres $5678 springen. De codering van deze adresseermethode is identiek aan die van de Absolute-modus en deze instructie zou dus worden gecodeerd als $7C $34 $12.

Base-Page Indirect X-Indexed-modus

Deze adresseermethode is identiek aan de Absolute Indirect X-Indexed-modus, behalve dat het adres van de pointer wordt gevormd uit het Base-Page-register (hoge byte) en de 8-bits operand (lage byte). De codering van deze adresseermethode is identiek aan die van de Base-Page-modus.

Indien het X-register bijvoorbeeld de waarde $04 bevat en de geheugenlocaties $16 en $17 in de huidige Base-Page respectievelijk $34 en $12 bevatten, dan

```
LDA ($12,X)
```

zou de inhoud van geheugenlocatie $1234 lezen en het resultaat opslaan in het A-register. Deze instructie wordt gecodeerd als $A1 $12.

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page.

Base-Page indirect Y-geïndexeerde modus

Deze adresseermethode verschilt van de X-geïndexeerde indirecte modi doordat de waarde van het Y-register wordt opgeteld bij het adres dat van de pointer wordt gelezen, in plaats van bij de pointer zelf. Dit is een zeer nuttige modus die vaak wordt gebruikt omdat deze feitelijk toegang biedt tot "de byte met index Y in het geheugen op het adres waarnaar de operand verwijst." Met andere woorden, het derefereert een pointer. De codering voor deze adresseermethode is identiek aan de Base-Page-modus.

Bijvoorbeeld, als de Y-register de waarde $04 bevat en de geheugenlocaties $12 en $13 in de huidige Base-Page respectievelijk $78 en $56 bevatten, dan

```
LDA ($12),Y
```

zou de inhoud van geheugenlocatie $567C (d.w.z. $5678 + Y) lezen en het resultaat opslaan in het A-register. Deze instructie wordt gecodeerd als $B1 $12.

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page.

Base-Page indirect Z-geïndexeerde modus

Deze adresseermethode verschilt van de X-geïndexeerde indirecte modi doordat de waarde van het Z-register wordt opgeteld bij het adres dat van de pointer wordt gelezen, in plaats van bij de pointer zelf. Dit is een zeer nuttige modus die vaak wordt gebruikt omdat deze feitelijk toegang biedt tot "de byte met index Z in het geheugen op het adres waarnaar de operand verwijst." Met andere woorden, het derefereert een pointer. De codering voor deze adresseermethode is identiek aan de Base-Page-modus.

Bijvoorbeeld, als de Z-register de waarde $04 bevat en de geheugenlocaties $12 en $13 in de huidige Base-Page respectievelijk $78 en $56 bevatten, dan

```
LDA ($12),Z
```

zou de inhoud van geheugenlocatie $567C (d.w.z. $5678 + Z) lezen en het resultaat opslaan in het A-register. Deze instructie wordt gecodeerd als $B2 $12.

Met andere woorden, het is equivalent aan de Base-Page indirect Y-geïndexeerde modus, maar met de Z-register in plaats van de Y-register om de offset te berekenen.

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page.

Base-Page Quad indirect Z-geïndexeerde modus

Deze adresseermethode is identiek aan de Base-Page indirect Z-geïndexeerde modus, met als uitzondering dat er met 32 bits aan data wordt gewerkt. De codering

voor deze adresseermethode is identiek aan de Base-Page-modus, met als uitzondering dat deze wordt voorafgegaan door $42, $42. Bijvoorbeeld, als de Z-register de waarde $04 bevat en de geheugenlocaties $12 en $13 in de huidige Base-Page respectievelijk $CD en $AB bevatten, dan

```
LDQ ($12),Z
```

zou de inhoud van geheugenlocatie $ABD1 (d.w.z. $ABCD + Z) tot en met $ABD4 lezen en het resultaat opslaan in het Q Pseudo Register. Deze instructie wordt gecodeerd als $42 $42 $B2 $12.

Momenteel is LDQ de enige instructie die deze modus ondersteunt.

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page. Zie de opmerking op pagina 55 voor meer informatie over instructies in de Quad-modus.

32-bits Base-Page-indirecte Z-geïndexeerde modus

Deze modus ontstaat door een Base-Page-indirecte Z-geïndexeerde instructie te laten voorafgaan door de NOP-instructie (opcode $EA). Hierdoor leest de 45GS02 in plaats van een 16-bits adres een 32-bits adres uit het Base-Page-adres dat door de operand wordt aangegeven. De Z-index wordt bij die pointer opgeteld. Belangrijk is dat het 32-bits adres niet verwijst naar het huidige 64KB-geheugenbeeld van de processor, maar naar de werkelijke 28-bits adresruimte van de 45GS02. Hierdoor is elk geheugengebied eenvoudig toegankelijk, zonder dat complexe bank-switching- of DMA-operaties nodig zijn.

Indien adressen $12 tot $15 de bytes $20, $30, $FD, $0F bevatten, wat het 32-bits adres $FFD3020 weergeeft, d.w.z. het natuurlijke adres van het randkleurregister van de VIC-IV, en de Z-index de waarde $01 bevat, zou de volgende instructiesequentie de schermkleur op blauw zetten, omdat zich het schermkleurregister op $FFD3021 bevindt, dus $FFD3020 + Z:

```
LDA #$06
LDZ #$01
STA [$12],Z
```

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page.

32-bits Base-Page-(Zero-Page)-indirecte quad Z-geïndexeerde modus

Deze adresseermethode is identiek aan de 32-bits Base-Page-indirecte Z-geïndexeerde modus, met als uitzondering dat deze op 32 bits gegevens werkt op het 32-bits adres dat door het argument wordt gevormd, in tegenstelling tot de 32-bits Base-Page-indirecte Z-geïndexeerde modus die slechts op 8 bits gegevens werkt. De

codering van deze adresseermethode is $42, $42, $EA, gevolgd door de natuurlijke 6502-opcode van de instructie die wordt uitgevoerd.

Het is ook belangrijk op te merken dat Z in werkelijkheid meestal *niet* bij de pointer wordt opgeteld, omdat het deel uitmaakt van het Q Pseudo Register. Alleen `LDQ ($nn),Z` telt Z op. Bijvoorbeeld,

```
LDQ [$12],Z
```

zou de geheugenlocaties $12 tot $15 uit de Base-Page lezen en die waarden gebruiken om het 32-bits adres te vormen waaruit het Q Pseudo Register wordt geladen. De Z-index wordt bij het 32-bits adres opgeteld. Deze instructie wordt gecodeerd als $42 $42 $EA $B2 $12.

Zie de opmerking op pagina 55 voor meer informatie over Base-Page en Zero-Page. Zie de opmerking op pagina 55 voor meer informatie over instructies in de Quad-modus. Zie de opmerking op pagina 61 voor meer informatie over 32-bits Base-Page-indirecte adressering.

Stapelrelatieve indirecte Y-geïndexeerde modus

Deze adresseermethode lijkt op de Base-Page-indirecte Y-geïndexeerde modus, met als uitzondering dat de operand in plaats van het adres van de pointer in de Base-Page de offset in de stapel aangeeft om de pointer te vinden. Deze adresseermethode derefereert effectief een pointer die op de stapel is geplaatst, bijv. als onderdeel van een functieaanroep vanuit een hogere programmeertaal. De codering is identiek aan die van de Base-Page-modus.

Bijvoorbeeld,

```
LDA ($12,SP),Y
```

Dit zou de inhoud van het geheugen op de huidige stapelaanwijzer plus $12 gebruiken om het adres van de pointer te berekenen. Aan deze pointer wordt vervolgens de waarde van het Y-register opgeteld om het uiteindelijke adres te verkrijgen waaruit wordt gelezen en om de waarde in het accumulatorregister op te slaan. De bytestroom van deze instructie is $E2 $12.

Indien de stapelaanwijzer momenteel naar $01E0 wijst, wordt het pointeradres gelezen van adressen $1F2 en $1F3, dus de twee bytes op $1E0 + $12 en $1E0 + $13. Indien de locaties $1F2 en $1F3 respectievelijk $78 en $56 bevatten en het Y-register de waarde $34 bevat, zou de uiteindelijke geheugenlocatie die wordt gelezen $5678 + $34 = $56AC zijn en wordt de inhoud van die geheugenlocatie in het accumulatorregister geladen.

Relatieve adresseermethode

Bij deze adresseermethode is de operand een 8-bits teken-offset ten opzichte van de huidige waarde van de programmateller (PC). Hiermee kunnen vertakkingen het

nabije adres coderen waar de uitvoering wordt voortgezet indien de vertakking wordt gevolgd.

Bijvoorbeeld,

```
BNE $2003
```

zou naar $2003 springen indien de Z-vlag van de processor niet gezet is. Indien deze instructie zich op adres $2000 bevindt, wordt deze gecodeerd als $D0 $01, dus een vertakking naar +1 bytes na de PC. Vertakkingsoffsets groter dan $7F vertakken achterwaarts, waarbij $FD naar de byte direct voor de vertakkinstructie vertakt en lagere waarden geleidelijk verder achterwaarts vertakken. Op deze manier kan effectief een vertaking worden gemaakt tussen -125 en +127 bytes vanaf de opcode-byte van de vertakinstructie. Voor langere vertaktingen ondersteunt de 45GS02 de relatieve word-adresseermethode, waarbij de offset met 2 bytes in plaats van 1 wordt gecodeerd.

Relatieve word-adresseermethode

Deze adresseermethode is identiek aan de relatieve adresseermethode, met als uitzondering dat de adresoffset een 16-bits waarde is. Hiermee kan een relatieve vertakking of sprong naar elke locatie in het huidige 64KB-geheugenbeeld worden gemaakt. Dit maakt het mogelijk volledig verplaatsbare software te schrijven, omdat absolute adressen bij het aanroepen van routines niet nodig zijn.

Bijvoorbeeld,

```
BNE $3000
```

zou naar $3000 springen indien de Z-vlag van de processor niet gezet is. Indien deze instructie zich op adres $2002 bevindt, wordt deze gecodeerd als $D3 $FC $0F, dus een vertakking naar +$FFC = 4092 bytes na de tweede byte van de instructie. Het feit dat de instructie 3 bytes lang is, wordt bij deze berekening buiten beschouwing gelaten.

6502-INSTRUCTIESET

Opmerking: de mechanismen voor het overschakelen van de 4510-CPU naar de 6502-CPU-persoonlijkheid zijn nog niet vastgelegd.

Officiële en onbedoelde instructies

De opcode-matrix van de 6502 heeft een grootte van 16 x 16 = 256 mogelijke opcodes. De opcodes die officieel gedocumenteerd zijn, vormen de verzameling van de legal instructies. Alle instructies van deze geldige verzameling worden voorafgegaan door een blauw gekleurde mnemoniek.

De overige opcodes vormen de verzameling van de onbedoelde instructies (soms worden deze "illegal"-instructies genoemd). Volledigheidshalve worden ook deze gedocumenteerd. Alle instructies van de onbedoelde verzameling worden voorafgegaan door een rood gekleurde mnemoniek.

OPMERKING: De onbedoelde instructies zijn momenteel niet geïmplementeerd en er wordt niet gegarandeerd dat zij exact dezelfde resultaten produceren als op andere CPU's van de 65xx-familie. Het is bekend dat veel van deze instructies instabiel zijn, zelfs wanneer zij op oude hardware draaien.

Opcodetabel

De opcodetabel vermeldt alle mogelijke opcodes, hun grootte, cycli en adresseermethode in een beknopt formaat.

Een cel met een lichtrode achtergrond duidt een onbedoelde instructie aan.

	$x0	$x1
$0x	size *cyc* OPC mode	size *cyc* OPC mode

De letters die aan het aantal cycli zijn toegevoegd, hebben de volgende betekenis:

	Betekenis
b	Voeg één cyclus toe als een branch wordt genomen. Voeg één cyclus extra toe als een genomen branch een paginagrens overschrijdt.
p	Voeg één cyclus toe als indexering een paginagrens overschrijdt.

Opcode-tabel 6502

	$x0	$x1	$x2	$x3	$x4	$x5	$x6	$x7
$0x	1 7 BRK imp	2 6 ORA bpiX	1 9 KIL imp	2 8 SLO bpiX	2 3 NOP bp	2 3 ORA bp	2 5 ASL bp	2 5 SLO bp
$1x	2 2b BPL rel	2 5p ORA ibpY	1 9 KIL imp	2 8 SLO ibpY	2 4 NOP bpX	2 4p ORA bpX	2 6 ASL bpX	2 6 SLO bpX
$2x	3 6 JSR abs	2 6 AND bpiX	1 9 KIL imp	2 8 RLA bpiX	2 3 BIT bp	2 3 AND bp	2 5 ROL bp	2 5 RLA bp
$3x	2 2b BMI rel	2 5p AND ibpY	1 9 KIL imp	2 8 RLA ibpY	2 4 NOP bpX	2 4 AND bpX	2 6 ROL bpX	2 6 RLA bpX
$4x	1 6 RTI imp	2 6 EOR bpiX	1 9 KIL imp	2 8 SRE bpiX	2 3 NOP bp	2 3 EOR bp	2 5 LSR bp	2 5 SRE bp
$5x	2 2b BVC rel	2 5p EOR ibpY	1 9 KIL imp	2 8 SRE ibpY	2 4 NOP bpX	2 4 EOR bpX	2 6 LSR bpX	2 6 SRE bpX
$6x	1 6 RTS imp	2 6 ADC bpiX	1 9 KIL imp	2 8 RRA bpiX	2 3 NOP bp	2 3 ADC bp	2 5 ROR bp	2 5 RRA bp
$7x	2 2b BVS rel	2 5p ADC ibpY	1 9 KIL imp	2 8 RRA ibpY	2 4 NOP bpX	2 4 ADC bpX	2 6 ROR bpX	2 6 RRA bpX
$8x	2 2 NOP imm	2 6 STA bpiX	2 2 NOP imm	2 6 SAX bpiX	2 3 STY bp	2 3 STA bp	2 3 STX bp	2 3 SAX bp
$9x	2 2b BCC rel	2 6 STA ibpY	1 9 KIL imp	2 6 SHA ibpY	2 4 STY bpX	2 4 STA bpX	2 4 STX bpY	2 4 SAX bpY
$Ax	2 2 LDY imm	2 6 LDA bpiX	2 2 LDX imm	2 6 LAX bpiX	2 3 LDY bp	2 3 LDA bp	2 3 LDX bp	2 3 LAX bp
$Bx	2 2b BCS rel	2 5p LDA ibpY	1 9 KIL imp	2 5p LAX ibpY	2 4 LDY bpX	2 4 LDA bpX	2 4 LDX bpY	2 4 LAX bpY
$Cx	2 2 CPY imm	2 6 CMP bpiX	2 2 NOP imm	2 8 DCP bpiX	2 3 CPY bp	2 3 CMP bp	2 5 DEC bp	2 5 DCP bp
$Dx	2 2b BNE rel	2 5p CMP ibpY	1 9 KIL imp	2 8 DCP ibpY	2 4 NOP bpX	2 4 CMP bpX	2 6 DEC bpX	2 6 DCP bpX
$Ex	2 2 CPX imm	2 6 SBC bpiX	2 2 NOP imm	2 8 ISC bpiX	2 3 CPX bp	2 3 SBC bp	2 5 INC bp	2 5 ISC bp
$Fx	2 2b BEQ rel	2 5p SBC ibpY	1 9 KIL imp	2 8 ISC ibpY	2 4 NOP bpX	2 4 SBC bpX	2 6 INC bpX	2 6 ISC bpX

Opcode-tabel 6502

$x8	$x9	$xA	$xB	$xC	$xD	$xE	$xF	
1 PHP 3 imp	2 ORA 2 imm	1 ASL 2 acc	2 ANC 2 imm	3 NOP 4 abs	3 ORA 4 abs	3 ASL 6 abs	3 SLO 6 abs	$0x
1 CLC 2 imp	3 ORA 4p absY	1 NOP 2 imp	3 SLO 7 absY	3 NOP 4p absX	3 ORA 4p absX	3 ASL 7 absX	3 SLO 7 absX	$1x
1 PLP 4 imp	2 AND 2 imm	1 ROL 2 acc	2 ANC 2 imm	3 BIT 4 abs	3 AND 4 abs	3 ROL 6 abs	3 RLA 6 abs	$2x
1 SEC 2 imp	3 AND 4p absY	1 NOP 2 imp	3 RLA 7 absY	3 NOP 4p absX	3 AND 4p absX	3 ROL 7 absX	3 RLA 7 absX	$3x
1 PHA 3 imp	2 EOR 2 imm	1 LSR 2 acc	2 ALR 2 imm	3 JMP 3 abs	3 EOR 4 abs	3 LSR 6 abs	3 SRE 6 abs	$4x
1 CLI 2 imp	3 EOR 4p absY	1 NOP 2 imp	3 SRE 7 absY	3 NOP 4p absX	3 EOR 4p absX	3 LSR 7 absX	3 SRE 7 absX	$5x
1 PLA 4 imp	2 ADC 2 imm	1 ROR 2 acc	2 ARR 2 imm	3 JMP 5 ind	3 ADC 4 abs	3 ROR 6 abs	3 RRA 6 abs	$6x
1 SEI 2 imp	3 ADC 4p absY	1 NOP 2 imp	3 RRA 7 absY	3 NOP 4p absX	3 ADC 4p absX	3 ROR 7 absX	3 RRA 7 absX	$7x
1 DEY 2 imp	2 NOP 2 imm	1 TXA 2 imp	2 XAA 2 imm	3 STY 4 abs	3 STA 4 abs	3 STX 4 abs	3 SAX 4 abs	$8x
1 TYA 2 imp	3 STA 5 absY	1 TXS 2 imp	3 TAS 5 absY	3 SHY 5 absX	3 STA 5 absX	3 SHX 5 absY	3 SHA 5 absY	$9x
1 TAY 2 imp	2 LDA 2 imm	1 TAX 2 imp	2 LAX 2 imm	3 LDY 4 abs	3 LDA 4 abs	3 LDX 4 abs	3 LAX 4 abs	$Ax
1 CLV 2 imp	3 LDA 4p absY	1 TSX 2 imp	3 LAS 4p absY	3 LDY 4p absX	3 LDA 4p absX	3 LDX 4p absY	3 LAX 4p absY	$Bx
1 INY 2 imp	2 CMP 2 imm	1 DEX 2 imp	2 SBX 2 imm	3 CPY 4 abs	3 CMP 4 abs	3 DEC 6 abs	3 DCP 6 abs	$Cx
1 CLD 2 imp	3 CMP 4p absY	1 NOP 2 imp	3 DCP 7 absY	3 NOP 4p absX	3 CMP 4p absX	3 DEC 7 absX	3 DCP 7 absX	$Dx
1 INX 2 imp	2 SBC 2 imm	1 NOP 2 imp	2 SBC 2 imm	3 CPX 4 abs	3 SBC 4 abs	3 INC 6 abs	3 ISC 6 abs	$Ex
1 SED 2 imp	3 SBC 4p absY	1 NOP 2 imp	3 ISC 7 absY	3 NOP 4p absX	3 SBC 4p absX	3 INC 7 absX	3 ISC 7 absX	$Fx

ADC

Deze instructie telt het argument en de carry-vlag op bij de inhoud van het accumulatorregister. Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten, anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De C-vlag wordt gezet als het ongetekende resultaat >255 is, of >99 indien de D-vlag is gezet.

ADC : Add with carry **6502**

$A \leftarrow A + M + C$

			N	Z	I	C	D	V	E	
			+	+	·	+	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect,X)	ADC ($nn,X)	61	2	6						
zero-page	ADC $nn	65	2	3						
immediate 8bit	ADC #$nn	69	2	2						
absolute	ADC $nnnn	6D	3	4						
(indirect),Y	ADC ($nn),Y	71	2	5						p
zero-page,X	ADC $nn,X	75	2	4						
absolute,Y	ADC $nnnn,Y	79	3	4						p
absolute,X	ADC $nnnn,X	7D	3	4						p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

ALR [onbedoeld]

Deze instructie schuift het accumulatorregister één bit naar rechts na het uitvoeren van een bitgewijze EN van het accumulatorregister en het immediate-mode argument. Bit 7 wordt op nul gezet en bit 0 wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

- De C-vlag wordt gezet op bit 0 van het AND-resultaat, voor het verschuiven.

ALR : Binary AND and Logical Shift Right **6502**

A ← A *AND* M, C ← A(0), A ← A≫1, A(7) ← 0

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	ALR #$nn	4B	2	2

ANC [onbedoeld]

Deze instructie voert een bitgewijze EN-operatie uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister en die ook in het argument staan, worden na voltooiing ingesteld in het accumulatorregister. In tegenstelling tot de AND-instructie wordt de carry-vlag gezet alsof het resultaat één bit naar links is verschoven. Dat wil zeggen, de carry-vlag wordt op dezelfde manier gezet als de N-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

ANC : Binary AND, and Set Carry **6502**

A ← A *AND* M, C ← A(7)

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	ANC #$nn	0B	2	2
immediate 8bit	ANC #$nn	2B	2	2

AND

Deze instructie voert een bitgewijze EN-operatie uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister en die ook in het argument staan, worden na voltooiing ingesteld in het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

AND : Binary AND **6502**

A ← A *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	AND ($nn,X)	21	2	6	
zero-page	AND $nn	25	2	3	
immediate 8bit	AND #$nn	29	2	2	
absolute	AND $nnnn	2D	3	4	
(indirect),Y	AND ($nn),Y	31	2	5	p
zero-page,X	AND $nn,X	35	2	4	
absolute,Y	AND $nnnn,Y	39	3	4	p
absolute,X	AND $nnnn,X	3D	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

ARR [onbedoeld]

Deze instructie verschuift het accumulatorregister één bit naar rechts na het uitvoeren van een bitgewijze EN-operatie van het accumulatorregister en het argument M in de immediate-addreringswijze. Bit 7 wordt uitgewisseld met de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De V-vlag wordt op een of andere manier beïnvloed.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ARR : Binary AND and Rotate Right **6502**

A ← A *AND* M, A ← A$\gg$1, C ↔ A(7)

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	ARR #$nn	6B	2	2

ASL

Deze instructie verschuift het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt op nul gezet en bit 7 wordt verschoven naar de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ASL : Arithmetic Shift Left Memory or Accumulator **6502**

$A \leftarrow A \ll 1$ of $M \leftarrow M \ll 1$

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	ASL $nn	06	2	5
accumulator	ASL A	0A	1	2
absolute	ASL $nnnn	0E	3	6
zero-page,X	ASL $nn,X	16	2	6
absolute,X	ASL $nnnn,X	1E	3	7

BCC

Deze instructie vertakt naar het aangegeven adres als de carry-vlag gewist is.

BCC : Branch on Carry Flag Clear **6502**

$C{=}0 \Longrightarrow PC \leftarrow PC + R8$

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BCC $rr	90	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BCS

Deze instructie vertakt naar het aangegeven adres als de carry-vlag gezet is.

BCS : Branch on Carry Flag Set **6502**

C=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BCS $rr	B0	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BEQ

Deze instructie vertakt naar het aangegeven adres als de zero-vlag gezet is. BEQ staat voor branch if equal, omdat een CMP ertoe leidt dat de zero-vlag gezet wordt als de operanden gelijk zijn.

BEQ : Branch on Zero Flag Set **6502**

Z=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BEQ $rr	F0	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BIT

Deze instructie wordt gebruikt om de bits te testen die zijn opgeslagen op een geheugenlocatie of in het immediate-argument van de opcode.

Bits 6 en 7 van de inhoud van de geheugenlocatie worden rechtstreeks gekopieerd naar de overflow-vlag en de negatieve-vlag. De zero-vlag wordt gezet of gewist op basis van het resultaat van de binaire AND van het accumulatorregister en de inhoud van de aangegeven geheugenlocatie.

De immediate-test zet de N- en V-vlaggen met geldige toestanden, waarbij het argument wordt behandeld als de geheugenwaarde, wat bij de eerdere 65C02-implementatie niet het geval was.

Neveneffecten

- De N-vlag wordt gezet als bit 7 van de geheugenlocatie gezet is, anders wordt hij gewist.
- De V-vlag wordt gezet als bit 6 van de geheugenlocatie gezet is, anders wordt hij gewist.

- De Z-vlag wordt gezet als het resultaat van A *AND* M nul is, anders wordt hij gewist.

BIT : Perform Bit Test **6502**

N ← M(7), V ← M(6), Z ← A *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	BIT $nn	24	2	3
absolute	BIT $nnnn	2C	3	4

BMI

Deze instructie vertakt naar het aangegeven adres als de negatieve-vlag gezet is. BMI staat voor branch on minus.

BMI : Branch on Negative Flag Set **6502**

N=1 $\Longrightarrow$ PC ← PC + R8

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BMI $rr	30	2	2 ^{b}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BNE

Deze instructie vertakt naar het aangegeven adres als de zero-vlag gewist is. BNE staat voor branch if not equal, omdat een CMP ertoe leidt dat de zero-vlag gewist wordt als de operanden niet gelijk zijn.

BNE : Branch on Zero Flag Clear **6502**

Z=0 $\Longrightarrow$ PC ← PC + R8

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BNE $rr	D0	2	2 ^{b}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BPL

Deze instructie vertakt naar het aangegeven adres als de negatieve-vlag gewist is. BPL staat voor branch on plus.

BPL : Branch on Negative Flag Clear **6502**

N=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BPL $rr	10	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BRK

terruptserviceroutine onderscheid maken tussen IRQ-gebeurtenissen en BRK-gebeurtenissen. Bijvoorbeeld:

```
PLA           ; laad het statusregister
PHA           ; herstel de stapel
AND #$10      ; maskeer de break-vlag
BNE DO_BREAK  ; -> het was een BRK
...           ; anders door met de IRQ-server
```

Geciteerd uit: MCS6500 Microcomputer Family Programming Manual, januari 1976, tweede editie, MOS Technology Inc., blz. 144:

"De BRK is een instructie van één byte en de adresseermethode ervan is Implied."

Er wordt gediscussieerd over de vraag of BRK kan worden beschouwd als een instructie van twee bytes met de immediate-adresseermethode, waarbij het operand-byte wordt weggegooid. Het byte na de BRK zou dan als oproepargument voor de break-handler kunnen dienen. Commodore gebruikte echter de BRK, zoals in de handleiding vermeld, als een instructie van één byte, die in de ML-monitor breekt als deze aanwezig is. Deze ingebouwde monitors verlaagden de op de stapel geplaatste PC, zodat deze kon worden gebruikt om direct terug te keren naar of te springen naar het code-byte na de BRK.

BRK : SP $\leftarrow$ SP $-$ 2, STAPEL $\leftarrow$ PC + 2, PC $\leftarrow$ ($FFFE) **6502**

M+

N	Z	I	C	D	V	E
.	.	.	.	**e**	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	BRK	00	1	7

BVC

Deze instructie vertakt naar het aangegeven adres als de overflow-vlag (V) gewist is.

BVC : Branch on Overflow Flag Clear **6502**

V=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BVC $rr	50	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BVS

Deze instructie vertakt naar het aangegeven adres als de overflow-vlag (V) gezet is.

BVS : Branch on Overflow Flag Set **6502**

V=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
relative	BVS $rr	70	2	2 [b]

[b] Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

CLC

Deze instructie wist de carry-vlag.

Neveneffecten

- De C-vlag wordt gewist.

CLC : Clear Carry Flag **6502**

C $\leftarrow$ 0

N	Z	I	C	D	V	E
.	.	.	+	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	CLC	18	1	2

CLD

Deze instructie wist de decimalenvlag. Rekenkundige bewerkingen maken dan gebruik van gewoon binair rekenen, in plaats van met BCD-gewogen decimale getallen (BCD).

Neveneffecten

- De D-vlag wordt gewist.

CLD : Clear Decimal Flag				**6502**
D ← 0				
			N Z I C D V E	
			· · · · + · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	CLD	D8	1	2

CLI

Deze instructie wist de interruptblokkeervlag. Interrupts kunnen nu plaatsvinden.

Neveneffecten

- De I-vlag wordt gewist.

CLI : Clear Interrupt Disable Flag				**6502**
I ← 0				
			N Z I C D V E	
			· · + · · · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	CLI	58	1	2

CLV

Deze instructie wist de overflow-vlag.

Neveneffecten

- De V-vlag wordt gewist.

CLV : Clear Overflow Flag				**6502**
V ← 0				
			N Z I C D V E	
			· · · · · + ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	CLV	B8	1	2

CMP

Deze instructie voert A − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt deze gewist.

CMP : Compare Accumulator **6502**

N,C,Z ⇐ [A − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	CMP ($nn,X)	C1	2	6	
zero-page	CMP $nn	C5	2	3	
immediate 8bit	CMP #$nn	C9	2	2	
absolute	CMP $nnnn	CD	3	4	
(indirect),Y	CMP ($nn),Y	D1	2	5	p
zero-page,X	CMP $nn,X	D5	2	4	
absolute,Y	CMP $nnnn,Y	D9	3	4	p
absolute,X	CMP $nnnn,X	DD	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

CPX

Deze instructie voert X − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van X − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van X − M nul of positief is, d.w.z. als X niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van X − M nul is; anders wordt deze gewist.

CPX : Compare X Register **6502**

N,C,Z ⇐ [X − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	CPX #$nn	E0	2	2
zero-page	CPX $nn	E4	2	3
absolute	CPX $nnnn	EC	3	4

CPY

Deze instructie voert Y − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Y − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van Y − M nul of positief is, d.w.z. als Y niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van Y − M nul is; anders wordt deze gewist.

CPY : Compare Y Register **6502**

N,C,Z ⇐ [Y − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	CPY #$nn	C0	2	2
zero-page	CPY $nn	C4	2	3
absolute	CPY $nnnn	CC	3	4

DCP [onbedoeld]

Deze instructie verlaagt de inhoud van de aangegeven geheugenlocatie met één en voert daarna A − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.

- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt deze gewist.

DCP : Decrement and Compare Accumulator **6502**

M ← M − 1, N,C,Z ⇐ [A − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	DCP ($nn,X)	C3	2	8
zero-page	DCP $nn	C7	2	5
absolute	DCP $nnnn	CF	3	6
(indirect),Y	DCP ($nn),Y	D3	2	8
zero-page,X	DCP $nn,X	D7	2	6
absolute,Y	DCP $nnnn,Y	DB	3	7
absolute,X	DCP $nnnn,X	DF	3	7

DEC

Deze instructie verlaagt het accumulatorregister of de aangegeven geheugenlocatie met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEC : Decrement **6502**

A ← A − 1 *of* M ← M − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	DEC $nn	C6	2	5
absolute	DEC $nnnn	CE	3	6
zero-page,X	DEC $nn,X	D6	2	6
absolute,X	DEC $nnnn,X	DE	3	7

DEX

Deze instructie verlaagt het X-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEX : Decrement X Register **6502**

X ← X − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	DEX	CA	1	2

DEY

Deze instructie verlaagt het Y-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEY : Decrement Y Register **6502**

Y ← Y − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	DEY	88	1	2

EOR

Deze instructie voert een bitgewijze uitsluitende-OF-operatie (XOR) uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister of die ingesteld staan in het argument, worden na voltooiing ingesteld in het accumulatorregister, maar niet allebei.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

EOR : Binary Exclusive OR **6502**

A ← A *XOR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	EOR ($nn,X)	41	2	6	
zero-page	EOR $nn	45	2	3	
immediate 8bit	EOR #$nn	49	2	2	
absolute	EOR $nnnn	4D	3	4	
(indirect),Y	EOR ($nn),Y	51	2	5	p
zero-page,X	EOR $nn,X	55	2	4	
absolute,Y	EOR $nnnn,Y	59	3	4	p
absolute,X	EOR $nnnn,X	5D	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

INC

Deze instructie verhoogt het accumulatorregister of de aangegeven geheugenlocatie met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INC : Increment Memory or Accumulator **6502**

A ← A + 1 of M ← M + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	INC $nn	E6	2	5
absolute	INC $nnnn	EE	3	6
zero-page,X	INC $nn,X	F6	2	6
absolute,X	INC $nnnn,X	FE	3	7

INX

Deze instructie verhoogt het X-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.

- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INX : Increment X Register				**6502**
X ← X + 1				
			N Z I C D V E	
			+ + · · · · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	INX	E8	1	2

INY

Deze instructie verhoogt het Y-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INY : Increment Y Register				**6502**
Y ← Y + 1				
			N Z I C D V E	
			+ + · · · · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	INY	C8	1	2

ISC [onbedoeld]

Deze instructie verhoogt de aangegeven geheugenlocatie en voert vervolgens A − M − 1 + C uit en zet de processorvlaggen dienovereenkomstig. Het resultaat wordt opgeslagen in het accumulatorregister.

OPMERKING: Deze instructie wordt beïnvloed door de status van de decimalenvlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M, anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten, anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.

- De Z-vlag wordt gezet als het resultaat van A − M nul is, anders wordt de vlag gewist.

ISC : Increment Memory, Subtract With Carry **6502**

M ← M + 1, A ← − M − 1 + C

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	ISC ($nn,X)	E3	2	8
zero-page	ISC $nn	E7	2	5
absolute	ISC $nnnn	EF	3	6
(indirect),Y	ISC ($nn),Y	F3	2	8
zero-page,X	ISC $nn,X	F7	2	6
absolute,Y	ISC $nnnn,Y	FB	3	7
absolute,X	ISC $nnnn,X	FF	3	7

JMP

Deze instructie stelt het Program Counter (PC)-register in op het door de instructie aangegeven adres, waardoor de uitvoering vanaf dat adres wordt voortgezet.

JMP : Jump to Address **6502**

PC ← M2:M1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute	JMP $nnnn	4C	3	3
(indirect)	JMP ($nnnn)	6C	3	5

JSR

Deze instructie slaat het adres van de instructie na de JSR-instructie op in de stapel en stelt vervolgens het Program Counter (PC)-register in op het door de instructie aangegeven adres, waardoor de uitvoering vanaf dat adres wordt voortgezet. Omdat het terugkeeradres in de stapel is opgeslagen, kan de RTS-instructie worden gebruikt om vanuit de aangeroepen subroutine terug te keren en de uitvoering na de JSR-instructie voort te zetten.

OPMERKING: deze instructie plaatst in feite het adres van de laatste byte van de JSR-instructie op de stapel. De RTS-instructie weet dit uiteraard en verhoogt het adres wanneer het van de stapel wordt gehaald, voordat het Program Counter-register (PC) wordt ingesteld.

JSR : Jump to Sub-Routine **6502**

SP ← SP − 2, STAPEL ← PCH:PCL, PC ← M2:M1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute	JSR $nnnn	20	3	6

KIL [onbedoeld]

Op een 6502 zorgen deze instructies ervoor dat de processor in een oneindige lus in zijn interne logica terechtkomt, welke alleen onderbroken kan worden door de computer te resetten. Op de 45GS02 veroorzaken deze instructies Hypervisor Traps, zodra deze functionaliteit is geïmplementeerd. Zo kunnen zij worden gebruikt om vast te stellen of er een 6502 of een 45GS02 draait: op een 6502-processor keert de instructie nooit terug, terwijl zij op een 45GS02 een uitzondering veroorzaakt, waardoor het aanroepende programma waarschijnlijk wordt afgebroken of crasht.

KIL : Lock-up 6502 Processor **6502**

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	KIL	02	1	9
implied	KIL	12	1	9
implied	KIL	22	1	9
implied	KIL	32	1	9
implied	KIL	42	1	9
implied	KIL	52	1	9
implied	KIL	62	1	9
implied	KIL	72	1	9
implied	KIL	92	1	9
implied	KIL	B2	1	9
implied	KIL	D2	1	9
implied	KIL	F2	1	9

LAS [onbedoeld]

OPMERKING: dit monster van een instructie is, afgezien van het feit dat het elk denkbaar nuttig doel ontbeert, onstabiel op veel 6502-processoren en dient daarom ook om die reden te worden vermeden, indien u niet al was afgeschrikt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.
- Een gevoel van holle voldoening, wanneer u daadwerkelijk een nuttig doel voor deze instructie ontdekt.

LAS : Set A, X and SPL Register With Useless Value **6502**

SP, A, X ← SP *AND* M

N	Z	I	C	D	V	E
+	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
absolute,Y	LAS $nnnn,Y	BB	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LAX [onbedoeld]

Deze instructie laadt zowel het accumulatorregister als het X-register met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

OPMERKING: de LAX-instructie staat erom bekend onstabiel te zijn op veel 6502-processoren en dient niet te worden gebruikt. Niet-immediate adresseermethoden zijn mogelijk stabiel genoeg om te gebruiken, maar dienen in het algemeen te worden vermeden.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.

LAX : Load Accumulator and X Registers **6502**

A, X ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	LAX ($nn,X)	A3	2	6	
zero-page	LAX $nn	A7	2	3	
immediate 8bit	LAX #$nn	AB	2	2	
absolute	LAX $nnnn	AF	3	4	
(indirect),Y	LAX ($nn),Y	B3	2	5	p
zero-page,Y	LAX $nn,Y	B7	2	4	
absolute,Y	LAX $nnnn,Y	BF	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LDA

Deze instructie laadt het accumulatorregister met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.

LDA : Load Accumulator **6502**

A ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	LDA ($nn,X)	A1	2	6	
zero-page	LDA $nn	A5	2	3	
immediate 8bit	LDA #$nn	A9	2	2	
absolute	LDA $nnnn	AD	3	4	
(indirect),Y	LDA ($nn),Y	B1	2	5	p
zero-page,X	LDA $nn,X	B5	2	4	
absolute,Y	LDA $nnnn,Y	B9	3	4	p
absolute,X	LDA $nnnn,X	BD	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LDX

Deze instructie laadt het X-register met de opgegeven waarde, of met de inhoud van de opgegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

LDX : Load X Register **6502**

X ← M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
immediate 8bit	LDX #$nn	A2	2	2						
zero-page	LDX $nn	A6	2	3						
absolute	LDX $nnnn	AE	3	4						
zero-page,Y	LDX $nn,Y	B6	2	4						
absolute,Y	LDX $nnnn,Y	BE	3	4						p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LDY

Deze instructie laadt het Y-register met de opgegeven waarde, of met de inhoud van de opgegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

LDY : Load Y Register **6502**

Y ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	LDY #$nn	A0	2	2	
zero-page	LDY $nn	A4	2	3	
absolute	LDY $nnnn	AC	3	4	
zero-page,X	LDY $nn,X	B4	2	4	
absolute,X	LDY $nnnn,X	BC	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

LSR

Deze instructie verschuift ofwel het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt op nul gezet en bit 0 wordt naar de carry-vlag uitgeschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.
- De C-vlag wordt gezet als bit 0 van de waarde gezet was voordat de waarde is verschoven.

LSR : Logical Shift Right **6502**

A ← A≫1, C ← A(0) of M ← M≫1, C ← M(0)

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	LSR $nn	46	2	5
accumulator	LSR A	4A	1	2
absolute	LSR $nnnn	4E	3	6
zero-page,X	LSR $nn,X	56	2	6
absolute,X	LSR $nnnn,X	5E	3	7

NOP

Deze instructies fungeren als noopinstructies: zij voeren de busoperaties uit alsof het echte instructies zijn, maar doen vervolgens niets met de opgehaalde waarde. Zij kunnen derhalve worden gebruikt als vertragingsinstructies, of om registers te lezen die bij het lezen neveneffecten hebben, zonder een register te beschadigen.

Op de 6502 is alleen $EA een opcode die voor NOP bedoeld is. Alle andere zijn uitsluitend beschikbaar op NMOS-versies van de processor, of op de 45GS02 in de 6502-modus.

NOP : No-Operation (some are unintended opcodes) **6502**

			N Z I C D V E		
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**	
zero-page	NOP $nn	04	2	3	
absolute	NOP $nnnn	0C	3	4	
zero-page,X	NOP $nn,X	14	2	4	
implied	NOP	1A	1	2	
absolute,X	NOP $nnnn,X	1C	3	4	*p*
zero-page,X	NOP $nn,X	34	2	4	
implied	NOP	3A	1	2	
absolute,X	NOP $nnnn,X	3C	3	4	*p*
zero-page	NOP $nn	44	2	3	
zero-page,X	NOP $nn,X	54	2	4	
implied	NOP	5A	1	2	
absolute,X	NOP $nnnn,X	5C	3	4	*p*
zero-page	NOP $nn	64	2	3	
zero-page,X	NOP $nn,X	74	2	4	
implied	NOP	7A	1	2	
absolute,X	NOP $nnnn,X	7C	3	4	*p*
immediate 8bit	NOP #$nn	80	2	2	
immediate 8bit	NOP #$nn	82	2	2	
immediate 8bit	NOP #$nn	89	2	2	
immediate 8bit	NOP #$nn	C2	2	2	
zero-page,X	NOP $nn,X	D4	2	4	
implied	NOP	DA	1	2	
absolute,X	NOP $nnnn,X	DC	3	4	*p*
immediate 8bit	NOP #$nn	E2	2	2	
implied	NOP	EA	1	2	
zero-page,X	NOP $nn,X	F4	2	4	
implied	NOP	FA	1	2	
absolute,X	NOP $nnnn,X	FC	3	4	*p*

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

ORA

Deze instructie voert een binaire OR-bewerking uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al

in het accumulatorregister gezet waren, of die in het argument gezet waren, worden bij voltooiing in het accumulatorregister gezet, of beide.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

ORA : Binary OR **6502**

A ← A *OR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	ORA ($nn,X)	01	2	6	
zero-page	ORA $nn	05	2	3	
immediate 8bit	ORA #$nn	09	2	2	
absolute	ORA $nnnn	0D	3	4	
(indirect),Y	ORA ($nn),Y	11	2	5	p
zero-page,X	ORA $nn,X	15	2	4	p
absolute,Y	ORA $nnnn,Y	19	3	4	p
absolute,X	ORA $nnnn,X	1D	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

PHA

Deze instructie duwt de inhoud van het accumulatorregister op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHA : Push Accumulator Register onto the Stack **6502**

SP ← SP − 1, STAPEL ← A

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHA	48	1	3

PHP

Deze instructie duwt de inhoud van de processorvlaggen op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHP : Push Processor Flags onto the Stack **6502**

SP ← SP − 1, STAPEL ← P

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHP	08	1	3

PLA

Deze instructie vervangt de inhoud van het accumulatorregister door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLA : Pull Accumulator Register from the Stack **6502**

A ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PLA	68	1	4

PLP

Deze instructie vervangt de inhoud van de processorvlaggen door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

OPMERKING: deze instructie vervangt NIET de 'Extended Stack Disable'-vlag (de E-vlag) of de software-interruptvlag (de B-vlag)

PLP : Pull Processor Flags from the Stack **6502**

P ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	+	+	+	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PLP	28	1	4

RLA [onbedoeld]

Deze instructie verschuift de inhoud van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt ingesteld op de huidige waarde van de carry-vlag en bit 7 wordt uitgeschoven in de carry-vlag. Het resultaat wordt vervolgens via een logische AND-bewerking met het accumulatorregister gecombineerd.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

RLA : Rotate Left Memory, and AND with Accumulator **6502**

$M \leftarrow M \ll 1$, $M(0) \leftarrow C$, $C \leftarrow M(7)$, $A \leftarrow A$ *AND* M

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	RLA ($nn,X)	23	2	8
zero-page	RLA $nn	27	2	5
absolute	RLA $nnnn	2F	3	6
(indirect),Y	RLA ($nn),Y	33	2	8
zero-page,X	RLA $nn,X	37	2	6
absolute,Y	RLA $nnnn,Y	3B	3	7
absolute,X	RLA $nnnn,X	3F	3	7

ROL

Deze instructie verschuift de inhoud van het accumulatorregister of van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt ingesteld op de huidige waarde van de carry-vlag en bit 7 wordt uitgeschoven in de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ROL : Rotate Left Memory or Accumulator **6502**

$M \leftarrow M \ll 1, C \leftarrow M(7), M(0) \leftarrow C$

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	ROL $nn	26	2	5
accumulator	ROL A	2A	1	2
absolute	ROL $nnnn	2E	3	6
zero-page,X	ROL $nn,X	36	2	6
absolute,X	ROL $nnnn,X	3E	3	7

ROR

Deze instructie verschuift ofwel het accumulatorregister ofwel de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt ingesteld op de huidige waarde van de carry-vlag, en bit 0 wordt in de carry-vlag verschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was gezet, voordat er werd verschoven.

ROR : Rotate Right Memory or Accumulator **6502**

$M \leftarrow M \gg 1, C \leftarrow M(0), M(7) \leftarrow C$

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	ROR $nn	66	2	5
accumulator	ROR A	6A	1	2
absolute	ROR $nnnn	6E	3	6
zero-page,X	ROR $nn,X	76	2	6
absolute,X	ROR $nnnn,X	7E	3	7

RRA [onbedoeld]

Deze instructie verschuift de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt ingesteld op de huidige waarde van de carry-vlag, en bit 0 wordt in de carry-vlag verschoven. Het resultaat wordt aan het accumulatorregister toegevoegd.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als de optelling in het accumulatorregister een overflow oplevert.

RRA : Rotate Right Memory, and Add to Accumulator **6502**

$M \leftarrow M \gg 1$, $C \leftarrow M(0)$, $M(7) \leftarrow C$, $A \leftarrow A + M$

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	RRA ($nn,X)	63	2	8
zero-page	RRA $nn	67	2	5
absolute	RRA $nnnn	6F	3	6
(indirect),Y	RRA ($nn),Y	73	2	8
zero-page,X	RRA $nn,X	77	2	6
absolute,Y	RRA $nnnn,Y	7B	3	7
absolute,X	RRA $nnnn,X	7F	3	7

RTI

Deze instructie haalt de processorvlaggen van de stapel en vervolgens het Program Counter (PC)-register van de stapel, waardoor een onderbroken programma kan worden hervat.

- De 6502-processorvlaggen worden hersteld van de stapel.
- Door deze instructie wordt noch de B-vlag (software-interrupt) noch de E-vlag (uitgebreide stapel) gezet.

RTI : Return From Interrupt **6502**

$P \leftarrow$ STAPEL, PCH:PCL $\leftarrow$ STAPEL, $SP \leftarrow SP - 3$

N	Z	I	C	D	V	E
+	+	+	+	+	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	RTI	40	1	6

RTS

Deze instructie haalt het Program Counter (PC)-register van de stapel en telt vervolgens een optioneel byte-argument op bij de stapelaanwijzer, waardoor een subroutine naar haar aanroeper kan terugkeren.

Het optionele argument is bedoeld om een aanroeper in staat te stellen argumenten voor de subroutine op de stapel te plaatsen. De subroutine leest de argumenten uit het stapelgeheugen en keert vervolgens terug met het aantal argumentbytes. De CPU haalt naast het retouradres ook de argumenten van de stapel, zodat de aanroeper de argumenten niet expliciet hoeft te verwijderen.

RTS : Return From Subroutine **6502**

PCH:PCL ← STAPEL, SP ← SP + 2 (+ M)

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	RTS	60	1	6

SAX [onbedoeld]

Deze instructie werkt als een combinatie van AND en CMP. Het resultaat wordt in het X-register opgeslagen. Omdat de instructie functionaliteit van CMP bevat in plaats van SBC, wordt de carry-vlag niet gebruikt bij de aftrekking, hoewel zij door de instructie wordt gewijzigd.

OPMERKING: deze instructie wordt beïnvloed door de status van de decimalenvlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt de vlag gewist. Als de vlag is gezet, geeft dit aan dat een getekende overflow is opgetreden.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt de vlag gewist.

SAX : AND Accumulator and X, and Subtract Without Carry **6502**

X ← (A *AND* X) − M

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	SAX ($nn,X)	83	2	6
zero-page	SAX $nn	87	2	3
absolute	SAX $nnnn	8F	3	4
zero-page,Y	SAX $nn,Y	97	2	4

SBC

Deze instructie voert A − M − 1 + C uit en zet de processorvlaggen dienovereenkomstig. Het resultaat wordt in het accumulatorregister opgeslagen.

OPMERKING: Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt de vlag gewist.

SBC : Subtract With Carry **6502**

A ← − M − 1 + C

			N	Z	I	C	D	V	E	
			+	+	·	+	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect,X)	SBC ($nn,X)	E1	2	6						
zero-page	SBC $nn	E5	2	3						
immediate 8bit	SBC #$nn	E9	2	2						
immediate 8bit	SBC #$nn	EB	2	2						
absolute	SBC $nnnn	ED	3	4						
(indirect),Y	SBC ($nn),Y	F1	2	5						p
zero-page,X	SBC $nn,X	F5	2	4						
absolute,Y	SBC $nnnn,Y	F9	3	4						p
absolute,X	SBC $nnnn,X	FD	3	4						p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

SBX [onbedoeld]

Deze instructie laadt het X-register met de binaire EN van het accumulatorregister en het X-register, verminderd met het immediate-argument.

OPMERKING: het aftrekkingseffect in deze instructie is te danken aan CMP , niet aan . De negatieve-vlag wordt derhalve gezet volgens de werking van CMP, niet van SBC. Dat wil zeggen dat de carry-vlag niet in de berekening wordt gebruikt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat nul of positief is; anders wordt de vlag gewist.

SBX : AND and Subtract **6502**

X ← (A *AND* X) − M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	SBX #$nn	CB	2	2

SEC

Deze instructie zet de carry-vlag.

Neveneffecten

- De C-vlag wordt gezet.

SEC : Set Carry Flag **6502**

C ← 1

N	Z	I	C	D	V	E
·	·	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	SEC	38	1	2

SED

Deze instructie zet de decimalenvlag. Rekenkundige bewerkingen maken vanaf nu gebruik van de BCD-modus (Binary Coded Decimal).

OPMERKING: de interrupt-handler van de C64 wist de decimalenvlag niet, waardoor het gevaarlijk is de decimalenvlag te zetten zonder eerst de interruptblokkeervlag te zetten.

Neveneffecten

- De D-vlag wordt gezet.

SED : Set Decimal Flag **6502**

D ← 1

N	Z	I	C	D	V	E
·	·	·	·	+	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	SED	F8	1	2

SEI

Deze instructie zet de interruptblokkeervlag. Gewone (IRQ-)interrupts kunnen niet langer optreden. Niet-maskeerbare interrupts (NMI) blijven optreden, zoals hun naam al aangeeft.

Neveneffecten

- De I-vlag wordt gezet.

SEI : Set Interrupt Disable Flag **6502**

I ← 1

N	Z	I	C	D	V	E
·	·	+	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	SEI	78	1	2

SHA [onbedoeld]

OPMERKING: deze instructie is onstabiel op veel 6502-processoren en dient te worden vermeden.

Deze instructie slaat de bitgewijze EN van de inhoud van het accumulatorregister, het X-register en de derde byte van de instructie op in de aangegeven geheugenlocatie.

SHA : Store binary AND of A, X and 3rd Instruction Byte **6502**

M ← A *AND* X AND B3

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect),Y	SHA ($nn),Y	93	2	6
absolute,Y	SHA $nnnn,Y	9F	3	5

SHX [onbedoeld]

OPMERKING: deze instructie is onstabiel op veel 6502-processoren en dient te worden vermeden.

Deze instructie slaat de bitgewijze EN van de inhoud van het X-register en de derde byte van de instructie op in de aangegeven geheugenlocatie.

SHX : Store Binary AND of X Register and 3rd Instruction Byte **6502**

M ← X *AND* B3

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute,Y	SHX $nnnn,Y	9E	3	5

SHY [onbedoeld]

OPMERKING: deze instructie is onstabiel op veel 6502-processoren en dient te worden vermeden.

Deze instructie slaat de bitgewijze EN van de inhoud van het Y-register en de derde byte van de instructie op in de aangegeven geheugenlocatie.

SHY : Store Binary AND of Y Register and 3rd Instruction Byte **6502**

M ← Y *AND* B3

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute,X	SHY $nnnn,X	9C	3	5

SLO [onbedoeld]

Deze instructie schuift de inhoud van de opgegeven geheugenlocatie één bit naar links en voert daarna een OR-bewerking uit van het resultaat met het accumulatorregister en slaat het resultaat op in het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het accumulatorregister na voltooiing van de instructie $00 bevat, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de geheugeninhoud was ingesteld, voor het verschuiven.

SLO : Shift Left and OR **6502**

C ← M(7), M ← M≪1, A ← A *OR* M

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	SLO ($nn,X)	03	2	8
zero-page	SLO $nn	07	2	5
absolute	SLO $nnnn	0F	3	6
(indirect),Y	SLO ($nn),Y	13	2	8
zero-page,X	SLO $nn,X	17	2	6
absolute,Y	SLO $nnnn,Y	1B	3	7
absolute,X	SLO $nnnn,X	1F	3	7

SRE [onbedoeld]

Deze instructie schuift de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt op nul gezet en bit 0 wordt naar de carry-vlag uitgeschoven. Het resultaat wordt vervolgens met behulp van een uitsluitende-OF-bewerking (XOR) met het accumulatorregister gecombineerd en het eindresultaat wordt in het accumulatorregister opgeslagen.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 0 van de waarde was ingesteld, voor het verschuiven.

SRE : Logical Shift Right and Exclusive OR with Accumulator **6502**

C ← M(0), M ← M≫1, A ← A *XOR* M

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	SRE ($nn,X)	43	2	8
zero-page	SRE $nn	47	2	5
absolute	SRE $nnnn	4F	3	6
(indirect),Y	SRE ($nn),Y	53	2	8
zero-page,X	SRE $nn,X	57	2	6
absolute,Y	SRE $nnnn,Y	5B	3	7
absolute,X	SRE $nnnn,X	5F	3	7

STA

Deze instructie slaat de inhoud van het accumulatorregister op in de aangeduide locatie.

STA : Store Accumulator **6502**

M ← A

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
(indirect,X)	STA ($nn,X)	81	2	6
zero-page	STA $nn	85	2	3
absolute	STA $nnnn	8D	3	4
(indirect),Y	STA ($nn),Y	91	2	6
zero-page,X	STA $nn,X	95	2	4
absolute,Y	STA $nnnn,Y	99	3	5
absolute,X	STA $nnnn,X	9D	3	5

STX

Deze instructie slaat de inhoud van het X-register op in de aangeduide locatie.

STX : Store X Register **6502**

M ← X

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	STX $nn	86	2	3
absolute	STX $nnnn	8E	3	4
zero-page,Y	STX $nn,Y	96	2	4

STY

Deze instructie slaat de inhoud van het Y-register op in de aangeduide locatie.

STY : Store Y Register **6502**

M ← Y

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
zero-page	STY $nn	84	2	3
absolute	STY $nnnn	8C	3	4
zero-page,X	STY $nn,X	94	2	4

TAS [onbedoeld]

OPMERKING: Dit monster van een instructie, behalve dat deze van elk denkbaar nut is ontdaan, is onstabiel op veel 6502-processoren en zou om die reden eveneens vermeden moeten worden, indien u er nog niet van afgeschrikt was.

Neveneffecten

- Opmerkelijk genoeg wijzigt het, ondanks de overgecompliceerde bewerking die het uitvoert, geen van de processorvlaggen.
- Verlies van verstand als u het probeert te gebruiken, of zelfs maar precies probeert uit te vogelen hoe het werkt.

TAS : Munge X Register and Stack Pointer **6502**

SP ← A *AND* X, M ← SP AND B3

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
absolute,Y	TAS $nnnn,Y	9B	3	5

TAX

Deze instructie laadt het X-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet is, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

TAX : Transfer Accumulator Register into the X Register **6502**

X ← A

N	Z	I	C	D	V	E
+	+	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TAX	AA	1	2

TAY

Deze instructie laadt het Y-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet is, anders wordt de vlag gewist.

- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

TAY : Transfer Accumulator Register into the Y Register **6502**

Y ← A

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TAY	A8	1	2

TSX

Deze instructie laadt het X-register met de inhoud van het Stack Pointer Low (SPL)-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TSX : Transfer Stack Pointer Low Register into the X Register **6502**

X ← SPL

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TSX	BA	1	2

TXA

Deze instructie laadt het accumulatorregister met de inhoud van het X-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TXA : Transfer X Register into the Accumulator Register **6502**

A ← X

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TXA	8A	1	2

TXS

Deze instructie zet de lage byte van het stapelaanwijzerregister (SPL) op de inhoud van het X-register.

TXS : Transfer X Register into Stack Pointer Low Register **6502**

SPL ← X

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TXS	9A	1	2

TYA

Deze instructie laadt het accumulatorregister met de inhoud van het Y-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TYA : Transfer Y Register into the Accumulator Register **6502**

A ← Y

N	Z	I	C	D	V	E
+	+	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	TYA	98	1	2

XAA [onbedoeld]

Deze instructie laadt het accumulatorregister met de binaire AND van het X-register en het immediate-modusargument.

OPMERKING: Deze instructie is op veel 6502-processors instabiel en mag niet gebruikt worden.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

XAA : Transfer X into A and AND with operand **6502**

A ← X *AND* VALUE

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
immediate 8bit	XAA #$nn	8B	2	2

Instructietiming

Merk op dat het aantal cycli afhangt van de snelheidsinstelling van de processor: sommige instructies nemen meer of minder cycli wanneer de processor op volle snelheid draait, op een C65-compatibele 3.5 MHz-snelheid of op een C64-compatibele 1 MHz/2 MHz-snelheid. Meer gedetailleerde informatie hierover staat vermeld bij de beschrijving van elke instructie, maar het overzicht op hoofdlijnen is:

- Wanneer de processor op 1 MHz draait, duren alle instructies ten minste twee cycli en worden er dummy-cycli opnieuw ingevoegd in de Read-Modify-Write-instructies, zodat alle instructies exact hetzelfde aantal cycli duren als op een 6502.
- De Read-Modify-Write-instructies en alle instructies die een waarde uit het geheugen lezen, vereisen allemaal een extra cyclus wanneer zij op volle snelheid draaien, om de signalen de gelegenheid te geven zich binnen de processor te verplaatsen.
- De Read-Modify-Write-instructies vereisen een extra cyclus wanneer de operand $D019 is, omdat de dummy-schrijfoperatie in dat geval wordt uitgevoerd. Dit dient om de compatibiliteit te verbeteren met C64-software die deze "bug" van de 6502 vaak gebruikt om VIC-II-interrupts sneller te bevestigen.
- De strafcycli voor paginagrensoverschrijding en het nemen van branches zijn niet van toepassing wanneer de processor op volle snelheid draait.
- Veel instructies vereisen minder cycli wanneer de processor op volle snelheid draait, omdat de niet-bus-cycli over het algemeen worden verwijderd. Het plaatsen op en halen van waarden van en naar de stapel vereist slechts 2 cycli in plaats van de 4 die de 6502 voor dergelijke instructies nodig heeft.

Opcodetabel

De gekleurde cellen duiden een uitgebreide 45GS02-opcode aan. Een opcode met het pseudoregister Q is blauw gemarkeerd en een Base-Page-indirecte, Z-geïndexeerde opcode die gebruik kan maken van 32-bits pointers is cyaan.

	$x0	$x1	$x2
$0x	bytes *cyc* OPC modus	bytes *cyc* QOP Qmodus	bytes *cyc* FARQ Q lbpZ

De letters die aan het aantal cycli zijn toegevoegd, hebben de volgende betekenis:

	Betekenis
b	Voeg één cyclus toe als een branch wordt genomen. Voeg één cyclus extra toe als een genomen branch een paginagrens overschrijdt.
d	Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
i	Voeg één cyclus toe wanneer de kloksnelheid 40 MHz bedraagt.
m	Trek de niet-bus-cycli af wanneer de CPU op 40 MHz draait.
p	Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r	Voeg één cyclus toe wanneer de kloksnelheid 40 MHz bedraagt.
s	De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

Opcode-tabel 4510/45GS02

	$x0	$x1	$x2	$x3	$x4	$x5	$x6	$x7
$0x	1 7 BRK imp	2 6rp ORA bpiX	1 1 CLE imp	1 1 SEE imp	2 3r TSB bp	2 3r Q ORA bp	2 4r Q ASL bp	2 4rb RMB0 bp
$1x	2 2b BPL rel	2 5rp ORA ibpY	2 5rp Q ORA ibpZ	3 3b BPL relfar	2 5r TRB bp	2 3r ORA bpX	2 4r Q ASL bpX	2 4rb RMB1 bp
$2x	3 5s JSR abs	2 5rp AND bpiX	3 5r JSR ind	3 5rp JSR indX	2 3r Q BIT bp	2 3r AND bp	2 4r Q ROL bp	2 4r RMB2 bp
$3x	2 2r BMI rel	2 5rp AND ibpY	2 5rp Q AND ibpZ	3 3b BMI relfar	2 3rp BIT bpX	2 4rp AND bpX	2 5rp Q ROL bpX	2 4r RMB3 bp
$4x	1 6m RTI imp	2 5r EOR bpiX	1 1 NEG acc	1 1 Q ASR acc	2 4r ASR bp	2 3r EOR bp	2 4r Q LSR bp	2 4r RMB4 bp
$5x	2 2b BVC rel	2 5rp EOR ibpY	2 5rp Q EOR ibpZ	3 3b BVC relfar	2 5rp Q ASR bpX	2 3p EOR bpX	2 3rp Q LSR bpX	2 4r RMB5 bp
$6x	1 6m RTS imp	2 5r ADC bpiX	2 4 RTS imm	3 3b BSR relfar	2 3 STZ bp	2 3r Q ADC bp	2 5r Q ROR bp	2 5r RMB6 bp
$7x	2 2b BVS rel	2 5rp ADC ibpY	2 5rp Q ADC ibpZ	3 3b BVS relfar	2 3 STZ bpX	2 3r ADC bpX	2 5rmdp Q ROR bpX	2 4r RMB7 bp
$8x	2 2b BRA rel	2 5p STA bpiX	2 6p STA ispY	3 3b BRA relfar	2 3 STY bp	2 3 Q STA bp	2 3 STX bp	2 4r SMB0 bp
$9x	2 2b BCC rel	2 5p STA ibpY	2 5p Q STA ibpZ	3 3b BCC relfar	2 3p STY bpX	2 3p STA bpX	2 3p STX bpY	2 4r SMB1 bp
$Ax	2 2 LDY imm	2 5rp LDA bpiX	2 2 LDX imm	2 2 LDZ imm	2 3r LDY bp	2 3r Q LDA bp	2 3r LDX bp	2 4r SMB2 bp
$Bx	2 2b BCS rel	2 5rp LDA ibpY	2 5rp Q LDA ibpZ	3 3b BCS relfar	2 3rp LDY bpX	2 3rp LDA bpX	2 5rp LDX bpY	2 4r SMB3 bp
$Cx	2 2 CPY imm	2 5rp CMP bpiX	2 2 CPZ imm	2 7mdr DEW bp	2 3r CPY bp	2 3r Q CMP bp	2 5mdr Q DEC bp	2 4r SMB4 bp
$Dx	2 2b BNE rel	2 5rp CMP ibpY	2 5rp Q CMP ibpZ	3 3b BNE relfar	2 3r CPZ bp	2 3rp CMP bpX	2 5mdrp Q DEC bpX	2 4r SMB5 bp
$Ex	2 2 CPX imm	2 3pm SBC bpiX	2 6rmp LDA ispY	2 7mdr INW bp	2 3r CPX bp	2 3r Q SBC bp	2 5mdr Q INC bp	2 4r SMB6 bp
$Fx	2 2b BEQ rel	2 3rp SBC ibpY	2 5rp Q SBC ibpZ	3 3b BEQ relfar	3 5m PHW im16	2 3rp SBC bpX	2 5dmrp Q INC bpX	2 4r SMB7 bp

Opcode-tabel 4510/45GS02

$x8	$x9	$xA	$xB	$xC	$xD	$xE	$xF	
1 PHP 2 imp	2 ORA 2 imm	1 ASL 1 Q acc	1 TSY 1 imp	3 TSB 5r abs	3 ORA 4r Q abs	3 ASL 5r Q abs	3 BBR0 0rb bpr8	$0x
1 CLC 1 imp	3 ORA 4r absY	1 INC 1 Q acc	1 INZ 1 imp	3 TRB 4r abs	3 ORA 4rp absX	3 ASL 5rp Q absX	3 BBR1 5b bpr8	$1x
1 PLP 4m imp	2 AND 2 imm	1 ROL 1 Q acc	1 TYS 1 imp	3 BIT 4r Q abs	3 AND 4r Q abs	3 ROL 5r Q abs	3 BBR2 5b bpr8	$2x
1 SEC 1 imp	3 AND 4r absY	1 DEC 1 Q acc	1 DEZ 1 imp	3 BIT 4rp absX	3 AND 4rp absX	3 ROL 5rp Q absX	3 BBR3 4b bpr8	$3x
1 PHA 2 imp	2 EOR 2 imm	1 LSR 1 Q acc	1 TAZ 1 imp	3 JMP 3 abs	3 EOR 4r Q abs	3 LSR 5r Q abs	3 BBR4 4rb bpr8	$4x
1 CLI 1 imp	3 EOR 4rp absY	1 PHY 2 imp	1 TAB 1 imp	1 MAP 1 imp	3 EOR 4rp absX	3 LSR 5rp Q absX	3 BBR5 4rb bpr8	$5x
1 PLA 4m imp	2 ADC 2 imm	1 ROR 1 Q acc	1 TZA 1 imp	3 JMP 5r ind	3 ADC 4r Q abs	3 ROR 6r Q abs	3 BBR6 4rb bpr8	$6x
1 SEI 1s imp	3 ADC 4r absY	1 PLY 4m imp	1 TBA 1 imp	3 JMP 6mp indX	3 ADC 4r absX	3 ROR 5rmdp Q absX	3 BBR7 4br bpr8	$7x
1 DEY 1s imp	2 BIT 2 imm	1 TXA 1 imp	3 STY 4p absX	3 STY 4 abs	3 STA 4 Q abs	3 STX 4 abs	3 BBS0 4br bpr8	$8x
1 TYA 1 imp	3 STA 4p absY	1 TXS 1 imp	3 STX 4p absY	3 STZ 4 abs	3 STA 4p absX	3 STZ 4p absX	3 BBS1 4br bpr8	$9x
1 TAY 1 imp	2 LDA 2 imm	1 TAX 1 imp	3 LDZ 4r abs	3 LDY 4r abs	3 LDA 4r Q abs	3 LDX 4r abs	3 BBS2 4br bpr8	$Ax
1 CLV 1 imp	3 LDA 4rp absY	1 TSX 1 imp	3 LDZ 4rp absX	3 LDY 4rp absX	3 LDA 4rp absX	3 LDX 4rp absY	3 BBS3 4br bpr8	$Bx
1 INY 1s imp	2 CMP 2 imm	1 DEX 1s imp	3 ASW 7rmd abs	3 CPY 4r abs	3 CMP 4r Q abs	3 DEC 6mdr Q abs	3 BBS4 4br bpr8	$Cx
1 CLD 1 imp	3 CMP 4rp absY	1 PHX 3m imp	1 PHZ 3m imp	3 CPZ 4r abs	3 CMP 4rp absX	3 DEC 6mdrp Q absX	3 BBS5 4br bpr8	$Dx
1 INX 1s imp	2 SBC 2 imm	1 EOM 1 imp	3 ROW 5rmd abs	3 CPX 4r abs	3 SBC 4r Q abs	3 INC 6dmr Q abs	3 BBS6 4br bpr8	$Ex
1 SED 1s imp	3 SBC 4rp absY	1 PLX 4m imp	1 PLZ 4m imp	3 PHW 7m abs	3 SBC 4rp absX	3 INC 6drp Q absX	3 BBS7 4br bpr8	$Fx

ADC

Deze instructie telt het argument en de carry-vlag op bij de inhoud van het accumulatorregister. Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten, anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De C-vlag wordt gezet als het ongetekende resultaat >255 is, of >99 indien de D-vlag is gezet.

ADC : Add with carry **4510**

$A \leftarrow A + M + C$

			N	Z	I	C	D	V	E	
			+	+	·	+	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect,X)	ADC ($nn,X)	61	2	5	r
base-page	ADC $nn	65	2	3	r
immediate 8bit	ADC #$nn	69	2	2	
absolute	ADC $nnnn	6D	3	4	r
(indirect),Y	ADC ($nn),Y	71	2	5	pr
(indirect),Z	ADC ($nn),Z	72	2	5	pr
base-page,X	ADC $nn,X	75	2	3	r
absolute,Y	ADC $nnnn,Y	79	3	4	r
absolute,X	ADC $nnnn,X	7D	3	4	r

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

AND

Deze instructie voert een bitgewijze EN-operatie uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister en die ook in het argument staan, worden na voltooiing ingesteld in het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

AND : Binary AND **4510**

A ← A *AND* M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect,X)	AND ($nn,X)	21	2	5	pr
base-page	AND $nn	25	2	3	r
immediate 8bit	AND #$nn	29	2	2	
absolute	AND $nnnn	2D	3	4	r
(indirect),Y	AND ($nn),Y	31	2	5	pr
(indirect),Z	AND ($nn),Z	32	2	5	pr
base-page,X	AND $nn,X	35	2	4	pr
absolute,Y	AND $nnnn,Y	39	3	4	r
absolute,X	AND $nnnn,X	3D	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ASL

Deze instructie verschuift het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt op nul gezet en bit 7 wordt verschoven naar de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ASL : Arithmetic Shift Left Memory or Accumulator **4510**

A ← A≪1 of M ← M≪1

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	ASL $nn	06	2	4	r
accumulator	ASL A	0A	1	1	s
absolute	ASL $nnnn	0E	3	5	r
base-page,X	ASL $nn,X	16	2	4	r
absolute,X	ASL $nnnn,X	1E	3	5	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ASR

Deze instructie schuift de inhoud van het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt als tekenbit beschouwd en behouden. De inhoud van bit 0 wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 0 van de waarde voor het schuiven was gezet.

ASR : Arithmetic Shift Right **4510**

A ← A>>1 of M ← M>>1

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
accumulator	ASR A	43	1	1	s
base-page	ASR $nn	44	2	4	r
base-page,X	ASR $nn,X	54	2	5	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ASW

Deze instructie schuift een 16-bits waarde in het geheugen één bit naar links.

Indien adres $1234 de waarde $87 bevat en adres $1235 de waarde $A9 bevat, geeft ASW $1234 als resultaat dat adres $1234 de waarde $0E bevat en adres $1235 de waarde $53, en wordt de carry-vlag gezet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de hoge byte voor het verschuiven gezet was; anders wordt de vlag gewist.

ASW : Arithmetic Shift Word Left **4510**

M ← M≪1

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
absolute	ASW $nnnn	CB	3	7	dmr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR0

Deze instructie vertakt naar het opgegeven adres indien bit 0 in de opgegeven basispagina-locatie nul is.

BBR0 : Branch on Bit 0 Reset **4510**

M(0)=0 ⟹ PC ← PC + R8

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR0 $nn,$rr	0F	3	0	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR1

Deze instructie vertakt naar het opgegeven adres indien bit 1 in de opgegeven basispagina-locatie nul is.

BBR1 : Branch on Bit 1 Reset **4510**

M(1)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR1 $nn,$rr	1F	3	5	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BBR2

Deze instructie vertakt naar het opgegeven adres indien bit 2 in de opgegeven basispagina-locatie nul is.

BBR2 : Branch on Bit 2 Reset **4510**

M(2)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR2 $nn,$rr	2F	3	5	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BBR3

Deze instructie vertakt naar het opgegeven adres indien bit 3 in de opgegeven basispagina-locatie nul is.

BBR3 : Branch on Bit 3 Reset **4510**

M(3)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR3 $nn,$rr	3F	3	4	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BBR4

Deze instructie vertakt naar het opgegeven adres indien bit 4 in de opgegeven basispagina-locatie nul is.

BBR4 : Branch on Bit 4 Reset **4510**

M(4)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR4 $nn,$rr	4F	3	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR5

Deze instructie vertakt naar het opgegeven adres indien bit 5 in de opgegeven basispagina-locatie nul is.

BBR5 : Branch on Bit 5 Reset **4510**

M(5)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR5 $nn,$rr	5F	3	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR6

Deze instructie vertakt naar het opgegeven adres indien bit 6 in de opgegeven basispagina-locatie nul is.

BBR6 : Branch on Bit 6 Reset **4510**

M(6)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR6 $nn,$rr	6F	3	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBR7

Deze instructie vertakt naar het opgegeven adres indien bit 7 in de opgegeven basispagina-locatie nul is.

BBR7 : Branch on Bit 7 Reset **4510**

M(7)=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBR7 $nn,$rr	7F	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS0

Deze instructie vertakt naar het opgegeven adres indien bit 0 in de opgegeven basispagina-locatie één is.

BBS0 : Branch on Bit 0 Set **4510**

M(0)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS0 $nn,$rr	8F	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS1

Deze instructie vertakt naar het opgegeven adres indien bit 1 in de opgegeven basispagina-locatie één is.

BBS1 : Branch on Bit 1 Set **4510**

M(1)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS1 $nn,$rr	9F	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS2

Deze instructie vertakt naar het opgegeven adres indien bit 2 in de opgegeven basispagina-locatie één is.

BBS2 : Branch on Bit 2 Set **4510**

M(2)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS2 $nn,$rr	AF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS3

Deze instructie vertakt naar het opgegeven adres indien bit 3 in de opgegeven basispagina-locatie één is.

BBS3 : Branch on Bit 3 Set **4510**

M(3)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS3 $nn,$rr	BF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS4

Deze instructie vertakt naar het opgegeven adres indien bit 4 in de opgegeven basispagina-locatie één is.

BBS4 : Branch on Bit 4 Set **4510**

M(4)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS4 $nn,$rr	CF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS5

Deze instructie vertakt naar het opgegeven adres indien bit 5 in de opgegeven basispagina-locatie één is.

BBS5 : Branch on Bit 5 Set **4510**

M(5)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS5 $nn,$rr	DF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS6

Deze instructie vertakt naar het opgegeven adres indien bit 6 in de opgegeven basispagina-locatie één is.

BBS6 : Branch on Bit 6 Set **4510**

M(6)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS6 $nn,$rr	EF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BBS7

Deze instructie vertakt naar het opgegeven adres indien bit 7 in de opgegeven basispagina-locatie één is.

BBS7 : Branch on Bit 7 Set **4510**

M(7)=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page+rel	BBS7 $nn,$rr	FF	3	4	br

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BCC

Deze instructie vertakt naar het aangegeven adres als de carry-vlag gewist is.

BCC : Branch on Carry Flag Clear **4510**

C=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BCC $rr	90	2	2	b
16-bit relative	BCC $rrrr	93	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BCS

Deze instructie vertakt naar het aangegeven adres als de carry-vlag gezet is.

BCS : Branch on Carry Flag Set **4510**

C=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BCS $rr	B0	2	2	b
16-bit relative	BCS $rrrr	B3	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BEQ

Deze instructie vertakt naar het aangegeven adres als de zero-vlag gezet is. BEQ staat voor branch if equal, omdat een CMP ertoe leidt dat de zero-vlag gezet wordt als de operanden gelijk zijn.

BEQ : Branch on Zero Flag Set **4510**

Z=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BEQ $rr	F0	2	2	b
16-bit relative	BEQ $rrrr	F3	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BIT

Deze instructie wordt gebruikt om de bits te testen die zijn opgeslagen op een geheugenlocatie of in het immediate-argument van de opcode.

Bits 6 en 7 van de inhoud van de geheugenlocatie worden rechtstreeks gekopieerd naar de overflow-vlag en de negatieve-vlag. De zero-vlag wordt gezet of gewist op basis van het resultaat van de binaire AND van het accumulatorregister en de inhoud van de aangegeven geheugenlocatie.

De immediate-test zet de N- en V-vlaggen met geldige toestanden, waarbij het argument wordt behandeld als de geheugenwaarde, wat bij de eerdere 65C02-implementatie niet het geval was.

Neveneffecten

- De N-vlag wordt gezet als bit 7 van de geheugenlocatie gezet is, anders wordt hij gewist.
- De V-vlag wordt gezet als bit 6 van de geheugenlocatie gezet is, anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat van A *AND* M nul is, anders wordt hij gewist.

BIT : Perform Bit Test **4510**

N ← M(7), V ← M(6), Z ← A *AND* M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page	BIT $nn	24	2	3	r
absolute	BIT $nnnn	2C	3	4	r
base-page,X	BIT $nn,X	34	2	3	pr
absolute,X	BIT $nnnn,X	3C	3	4	pr
immediate 8bit	BIT #$nn	89	2	2	

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BMI

Deze instructie vertakt naar het aangegeven adres als de negatieve-vlag gezet is. BMI staat voor branch on minus.

BMI : Branch on Negative Flag Set **4510**

N=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BMI $rr	30	2	2	r
16-bit relative	BMI $rrrr	33	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BNE

Deze instructie vertakt naar het aangegeven adres als de zero-vlag gewist is. BNE staat voor branch if not equal, omdat een CMP ertoe leidt dat de zero-vlag gewist wordt als de operanden niet gelijk zijn.

BNE : Branch on Zero Flag Clear **4510**

Z=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BNE $rr	D0	2	2	b
16-bit relative	BNE $rrrr	D3	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BPL

Deze instructie vertakt naar het aangegeven adres als de negatieve-vlag gewist is. BPL staat voor branch on plus.

BPL : Branch on Negative Flag Clear **4510**

N=0 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BPL $rr	10	2	2	b
16-bit relative	BPL $rrrr	13	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BRA

Deze instructie vertakt naar het opgegeven adres.

BRA : Branch Unconditionally **4510**

PC ← PC + R8 *of* PC ← PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BRA $rr	80	2	2	*b*
16-bit relative	BRA $rrrr	83	3	3	*b*

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BRK

terruptserviceroutine onderscheid maken tussen IRQ-gebeurtenissen en BRK-gebeurtenissen. Bijvoorbeeld:

```
PLA           ; laad het statusregister
PHA           ; herstel de stapel
AND #$10      ; maskeer de break-vlag
BNE DO_BREAK  ; -> het was een BRK
...           ; anders door met de IRQ-server
```

Geciteerd uit: MCS6500 Microcomputer Family Programming Manual, januari 1976, tweede editie, MOS Technology Inc., blz. 144:

"De BRK is een instructie van één byte en de adresseermethode ervan is Implied."

Er wordt gediscussieerd over de vraag of BRK kan worden beschouwd als een instructie van twee bytes met de immediate-adresseermethode, waarbij het operand-byte wordt weggegooid. Het byte na de BRK zou dan als oproepargument voor de break-handler kunnen dienen. Commodore gebruikte echter de BRK, zoals in de handleiding vermeld, als een instructie van één byte, die in de ML-monitor breekt als deze aanwezig is. Deze ingebouwde monitors verlaagden de op de stapel geplaatste PC, zodat deze kon worden gebruikt om direct terug te keren naar of te springen naar het code-byte na de BRK.

BRK : SP ← SP − 2, STAPEL ← PC + 2, PC ← ($FFFE) **4510**

M+

N	Z	I	C	D	V	E
.	.	.	.	**e**	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	BRK	00	1	7

BSR

Deze instructie vertakt naar het opgegeven adres en slaat het adres van de volgende instructie op de stapel op, zodat met een RTS-instructie uit de routine kan worden teruggekeerd.

Deze instructie is nuttig bij het gebruik van verplaatsbare code, omdat het een relatief geadresseerd alternatief voor JSR biedt.

BSR : Branch Sub-Routine **4510**

SP ← SP − 2, STAPEL ← PC + len(V), PC ← PC + V

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
16-bit relative	BSR $rrrr	63	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BVC

Deze instructie vertakt naar het aangegeven adres als de overflow-vlag (V) gewist is.

BVC : Branch on Overflow Flag Clear **4510**

V=0 $\Longrightarrow$ PC ← PC + R8 of PC ← PC + R16

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BVC $rr	50	2	2	b
16-bit relative	BVC $rrrr	53	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

BVS

Deze instructie vertakt naar het aangegeven adres als de overflow-vlag (V) gezet is.

BVS : Branch on Overflow Flag Set **4510**

V=1 $\Longrightarrow$ PC $\leftarrow$ PC + R8 of PC $\leftarrow$ PC + R16

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
relative	BVS $rr	70	2	2	b
16-bit relative	BVS $rrrr	73	3	3	b

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.

CLC

Deze instructie wist de carry-vlag.

Neveneffecten

- De C-vlag wordt gewist.

CLC : Clear Carry Flag **4510**

C $\leftarrow$ 0

N	Z	I	C	D	V	E
·	·	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLC	18	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CLD

Deze instructie wist de decimalenvlag. Rekenkundige bewerkingen maken dan gebruik van gewoon binair rekenen, in plaats van met BCD-gewogen decimale getallen (BCD).

Neveneffecten

- De D-vlag wordt gewist.

CLD : Clear Decimal Flag **4510**

D $\leftarrow$ 0

N	Z	I	C	D	V	E
·	·	·	·	+	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLD	D8	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CLE

Deze instructie wist de Extended Stack Disable-vlag. Hierdoor kan de stapel in lengte meer dan 256 bytes omvatten, doordat de processor de waarde van de hoge byte van het stapeladres (SPH) kan wijzigen.

Neveneffecten

- De E-vlag wordt gewist.

CLE : Clear Extended Stack Disable Flag **4510**

E ← 0

N	Z	I	C	D	V	E
·	·	·	·	·	·	+

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLE	02	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CLI

Deze instructie wist de interruptblokkeervlag. Interrupts kunnen nu plaatsvinden.

Neveneffecten

- De I-vlag wordt gewist.

CLI : Clear Interrupt Disable Flag **4510**

I ← 0

N	Z	I	C	D	V	E
·	·	+	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLI	58	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CLV

Deze instructie wist de overflow-vlag.

Neveneffecten

- De V-vlag wordt gewist.

CLV : Clear Overflow Flag **4510**

V ← 0

N	Z	I	C	D	V	E
·	·	·	·	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	CLV	B8	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

CMP

Deze instructie voert A − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt deze gewist.

CMP : Compare Accumulator **4510**

N,C,Z ⇐ [A − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	CMP ($nn,X)	C1	2	5	pr
base-page	CMP $nn	C5	2	3	r
immediate 8bit	CMP #$nn	C9	2	2	
absolute	CMP $nnnn	CD	3	4	r
(indirect),Y	CMP ($nn),Y	D1	2	5	pr
(indirect),Z	CMP ($nn),Z	D2	2	5	pr
base-page,X	CMP $nn,X	D5	2	3	pr
absolute,Y	CMP $nnnn,Y	D9	3	4	pr
absolute,X	CMP $nnnn,X	DD	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CPX

Deze instructie voert X − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van X − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van X − M nul of positief is, d.w.z. als X niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van X − M nul is; anders wordt deze gewist.

CPX : Compare X Register **4510**

N,C,Z ⇐ [X − M]

			N	Z	I	C	D	V	E	
			+	+	·	+	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
immediate 8bit	CPX #$nn	E0	2	2						
base-page	CPX $nn	E4	2	3	r					
absolute	CPX $nnnn	EC	3	4	r					

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CPY

Deze instructie voert Y − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Y − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van Y − M nul of positief is, d.w.z. als Y niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van Y − M nul is; anders wordt deze gewist.

CPY : Compare Y Register **4510**

N,C,Z ⇐ [Y − M]

			N	Z	I	C	D	V	E	
			+	+	·	+	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
immediate 8bit	CPY #$nn	C0	2	2						
base-page	CPY $nn	C4	2	3	r					
absolute	CPY $nnnn	CC	3	4	r					

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CPZ

Deze instructie voert Z − M uit en zet de processorvlaggen overeenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Z − M negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van Z − M nul of positief is, d.w.z. als Z niet kleiner is dan M; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat van Z − M nul is; anders wordt de vlag gewist.

CPZ : Compare Z Register **4510**

N,C,Z ⇐ [Z − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	CPZ #$nn	C2	2	2	
base-page	CPZ $nn	D4	2	3	*r*
absolute	CPZ $nnnn	DC	3	4	*r*

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

DEC

Deze instructie verlaagt het accumulatorregister of de aangegeven geheugenlocatie met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEC : Decrement **4510**

A ← A − 1 *of* M ← M − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
accumulator	DEC A	3A	1	1	*s*
base-page	DEC $nn	C6	2	5	*dmr*
absolute	DEC $nnnn	CE	3	6	*dmr*
base-page,X	DEC $nn,X	D6	2	5	*dmpr*
absolute,X	DEC $nnnn,X	DE	3	6	*dmpr*

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

DEW

Deze instructie verlaagt het aangegeven geheugenwoord in de basispagina met één. Het laagst genummerde adres bevat de minst significante bits. Wanneer geheugenlocatie $12 bijvoorbeeld $78 bevat en geheugenlocatie $13 $56, zorgt de instructie DEW $12 ervoor dat geheugenlocatie $12 de waarde $77 krijgt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

DEW : Decrement Memory Word **4510**

M16 ← M16 − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	DEW $nn	C3	2	7	*dmr*

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

DEX

Deze instructie verlaagt het X-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEX : Decrement X Register **4510**

$X \leftarrow X - 1$

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	DEX	CA	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

DEY

Deze instructie verlaagt het Y-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEY : Decrement Y Register **4510**

$Y \leftarrow Y - 1$

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	DEY	88	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

DEZ

Deze instructie verlaagt het Z-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

DEZ : Decrement Z Register **4510**

$Z \leftarrow Z - 1$

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	DEZ	3B	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

EOM

Anders dan op de 6502 vervult de NOP-instructie op de 45GS02 twee extra rollen wanneer de processor in de 4502-modus draait.

Ten eerste geeft de instructie het einde aan van een geheugenmappingreeks die door een MAP-instructie is veroorzaakt, waardoor interrupts opnieuw kunnen optreden.

Ten tweede instrueert hij de processor om een 32-bits pointer te gebruiken in plaats van een 16-bits pointer in 6502-stijl, indien de volgende instructie de adresseermethode Base-Page Indirect Z Indexed gebruikt. Dergelijke 32-bits adressen worden niet beïnvloed door de geheugenbanken van de C64, C65 of MEGA65. Dit maakt snelle en eenvoudige toegang tot de volledige adresruimte van de MEGA65 mogelijk, zonder dat u enige vorm van banking hoeft uit te voeren, daarvan bewust hoeft te zijn of de DMA-controller hoeft te gebruiken. Deze adresseermethode brengt twee extra cycli met zich mee, vanwege de tijd die nodig is om de twee extra bytes van de pointer te lezen.

OPMERKING: wees voorzichtig wanneer u EOM/NOP na een Hypervisor Call gebruikt voor een vertraging, omdat dit uw volgende instructie zou kunnen wijzigen. CLV kan als alternatief worden gebruikt.

Neveneffecten

- Heft het door de MAP-instructie opgelegde verbod op alle interrupts op, waardoor niet-maskeerbare interrupts weer kunnen optreden en tevens IRQ-interrupts, indien de interruptblokkeervlag niet gezet is.

EOM : End of Mapping Sequence / No-Operation **4510**

Speciaal

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	EOM	EA	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

EOR

Deze instructie voert een bitgewijze uitsluitende-OF-operatie (XOR) uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister of die ingesteld staan in het argument, worden na voltooiing ingesteld in het accumulatorregister, maar niet allebei.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

EOR : Binary Exclusive OR **4510**

A ← A *XOR* M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**		**Cycli**					
(indirect,X)	EOR ($nn,X)	41	2		5	r
base-page	EOR $nn	45	2		3	r
immediate 8bit	EOR #$nn	49	2		2	
absolute	EOR $nnnn	4D	3		4	r
(indirect),Y	EOR ($nn),Y	51	2		5	pr
(indirect),Z	EOR ($nn),Z	52	2		5	pr
base-page,X	EOR $nn,X	55	2		3	p
absolute,Y	EOR $nnnn,Y	59	3		4	pr
absolute,X	EOR $nnnn,X	5D	3		4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

INC

Deze instructie verhoogt het accumulatorregister of de aangegeven geheugenlocatie met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INC : Increment Memory or Accumulator **4510**

A ← A + 1 *of* M ← M + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
accumulator	INC A	1A	1	1	s
base-page	INC $nn	E6	2	5	dmr
absolute	INC $nnnn	EE	3	6	dmr
base-page,X	INC $nn,X	F6	2	5	$dmpr$
absolute,X	INC $nnnn,X	FE	3	6	dpr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

INW

Deze instructie verhoogt het aangegeven geheugenwoord in de basispagina met één. Het laagst genummerde adres bevat de minst significante bits. Wanneer geheugenlocatie $12 bijvoorbeeld $78 bevat en geheugenlocatie $13 $56, zorgt de instructie INW $12 ervoor dat geheugenlocatie $12 de waarde $79 krijgt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

INW : Increment Memory Word **4510**

M16 ← M16 + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	INW $nn	E3	2	7	dmr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

INX

Deze instructie verhoogt het X-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INX : Increment X Register **4510**

X ← X + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	INX	E8	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

INY

Deze instructie verhoogt het Y-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

INY : Increment Y Register **4510**

Y ← Y + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	INY	C8	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

INZ

Deze instructie verhoogt het Z-register met één.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

INZ : Increment Z Register **4510**

Z ← Y + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	INZ	1B	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

JMP

Deze instructie stelt het Program Counter (PC)-register in op het door de instructie aangegeven adres, waardoor de uitvoering vanaf dat adres wordt voortgezet.

JMP : Jump to Address **4510**

PC ← M2:M1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
absolute	JMP $nnnn	4C	3	3	
(indirect)	JMP ($nnnn)	6C	3	5	r
(indirect,X)	JMP ($nnnn,X)	7C	3	6	mp

m Trek de niet-buscycli af bij 40 MHz.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

JSR

Deze instructie slaat het adres van de instructie na de JSR-instructie op in de stapel en stelt vervolgens het Program Counter (PC)-register in op het door de instructie aangegeven adres, waardoor de uitvoering vanaf dat adres wordt voortgezet. Omdat het terugkeeradres in de stapel is opgeslagen, kan de RTS-instructie worden gebruikt om vanuit de aangeroepen subroutine terug te keren en de uitvoering na de JSR-instructie voort te zetten.

OPMERKING: deze instructie plaatst in feite het adres van de laatste byte van de JSR-instructie op de stapel. De RTS-instructie weet dit uiteraard en verhoogt het adres wanneer het van de stapel wordt gehaald, voordat het Program Counter-register (PC) wordt ingesteld.

JSR : Jump to Sub-Routine **4510**

SP ← SP − 2, STAPEL ← PCH:PCL, PC ← M2:M1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
absolute	JSR $nnnn	20	3	5	
(indirect)	JSR ($nnnn)	22	3	5	r
(indirect,X)	JSR ($nnnn,X)	23	3	5	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDA

Deze instructie laadt het accumulatorregister met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.

LDA : Load Accumulator **4510**

A ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	LDA ($nn,X)	A1	2	5	pr
base-page	LDA $nn	A5	2	3	r
immediate 8bit	LDA #$nn	A9	2	2	
absolute	LDA $nnnn	AD	3	4	r
(indirect),Y	LDA ($nn),Y	B1	2	5	pr
(indirect),Z	LDA ($nn),Z	B2	2	5	pr
base-page,X	LDA $nn,X	B5	2	3	pr
absolute,Y	LDA $nnnn,Y	B9	3	4	pr
absolute,X	LDA $nnnn,X	BD	3	4	pr
(immediate,SP),Y	LDA ($nn,SP),Y	E2	2	6	mpr

m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDX

Deze instructie laadt het X-register met de opgegeven waarde, of met de inhoud van de opgegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

LDX : Load X Register **4510**

X ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	LDX #$nn	A2	2	2	
base-page	LDX $nn	A6	2	3	r
absolute	LDX $nnnn	AE	3	4	r
base-page,Y	LDX $nn,Y	B6	2	5	pr
absolute,Y	LDX $nnnn,Y	BE	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDY

Deze instructie laadt het Y-register met de opgegeven waarde, of met de inhoud van de opgegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

LDY : Load Y Register **4510**

Y ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	LDY #$nn	A0	2	2	
base-page	LDY $nn	A4	2	3	r
absolute	LDY $nnnn	AC	3	4	r
base-page,X	LDY $nn,X	B4	2	3	pr
absolute,X	LDY $nnnn,X	BC	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDZ

Deze instructie laadt het Z-register met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

LDZ : Load Z Register **4510**

Z ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 8bit	LDZ #$nn	A3	2	2	
absolute	LDZ $nnnn	AB	3	4	r
absolute,X	LDZ $nnnn,X	BB	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LSR

Deze instructie verschuift ofwel het accumulatorregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt op nul gezet en bit 0 wordt naar de carry-vlag uitgeschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.
- De C-vlag wordt gezet als bit 0 van de waarde gezet was voordat de waarde is verschoven.

LSR : Logical Shift Right **4510**

A ← A≫1, C ← A(0) of M ← M≫1, C ← M(0)

			N	Z	I	C	D	V	E	
			+	+	·	+	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page	LSR $nn	46	2	4	r					
accumulator	LSR A	4A	1	1	s					
absolute	LSR $nnnn	4E	3	5	r					
base-page,X	LSR $nn,X	56	2	3	pr					
absolute,X	LSR $nnnn,X	5E	3	5	pr					

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

MAP

Deze instructie stelt de geheugenindeling in de C65- of MEGA65-stijl in, afhankelijk van de waarden in het accumulatorregister en de X-, Y- en Z-registers.

Het is een goede gewoonte om geen MAP-instructie uit te voeren in een geheugenregio waarvan de geheugenindeling verandert. De CPU wijzigt de geheugeninstellingen onmiddellijk wanneer de MAP-instructie wordt uitgevoerd. De programmateller gebruikt de nieuwe geheugenindeling om de volgende instructie te vinden.

OPMERKING: De MAP-instructie schakelt interrupts uit, ongeacht de I-vlag, tot aan de volgende EOM-instructie. EOM (byte EA) wordt daarnaast gebruikt als samengestelde instructie-prefix voor 32-bit indirect adresseren. Een dergelijke instructie gedraagt zich na een MAP-instructie als een EOM-instructie.

Meer informatie over deze instructie vindt u in bijlage 2.

Neveneffecten

- De geheugenindeling wordt onmiddellijk gewijzigd in de aangevraagde.
- Alle interrupts, waaronder niet-maskeerbare interrupts (NMI's), worden geblokkeerd totdat een EOM-instructie (NOP) wordt aangetroffen.

MAP : Set Memory Map **4510**

Speciaal

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	MAP	5C	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

NEG

Deze instructie vervangt de inhoud van het accumulatorregister door het tweecomplement van de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

NEG : Negate Accumulator **4510**

A $\leftarrow$ (A *XOR* $FF) + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
accumulator	NEG A	42	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ORA

Deze instructie voert een binaire OR-bewerking uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al in het accumulatorregister gezet waren, of die in het argument gezet waren, worden bij voltooiing in het accumulatorregister gezet, of beide.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

ORA : Binary OR **4510**

A ← A *OR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	ORA ($nn,X)	01	2	6	pr
base-page	ORA $nn	05	2	3	r
immediate 8bit	ORA #$nn	09	2	2	
absolute	ORA $nnnn	0D	3	4	r
(indirect),Y	ORA ($nn),Y	11	2	5	pr
(indirect),Z	ORA ($nn),Z	12	2	5	pr
base-page,X	ORA $nn,X	15	2	3	r
absolute,Y	ORA $nnnn,Y	19	3	4	r
absolute,X	ORA $nnnn,X	1D	3	4	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

PHA

Deze instructie duwt de inhoud van het accumulatorregister op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHA : Push Accumulator Register onto the Stack **4510**

SP ← SP − 1, STAPEL ← A

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHA	48	1	2

PHP

Deze instructie duwt de inhoud van de processorvlaggen op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHP : Push Processor Flags onto the Stack **4510**

SP ← SP − 1, STAPEL ← P

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHP	08	1	2

PHW

Deze instructie schuift een 16-bits letterlijke waarde of het aangegeven geheugenwoord op de stapel en verlaagt de waarde van de stapelaanwijzer met 2.

OPMERKING: anders dan JSR, waarbij de meest significante byte op het hogere stapeladres achterblijft, plaatst PHW eerst de minst significante byte en daarna de meest significante byte. Het woord ligt dus op de stack met de meest significante byte op het lagere adres (big-endian) en kan daarom niet rechtstreeks worden gebruikt als little-endian-pointer met stapelrelatieve adressering. Zo laat PHW #$1234 $12 op SP+1 en $34 op SP+2 achter.

PHW : Push Word onto the Stack **4510**

SP ← SP − 2, STAPEL ← M1:M2

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
immediate 16bit	PHW #$nnnn	F4	3	5	m
absolute	PHW $nnnn	FC	3	7	m

m Trek de niet-buscycli af bij 40 MHz.

PHX

Deze instructie schuift de inhoud van het X-register op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHX : Push X Register onto the Stack **4510**

SP ← SP − 1, STAPEL ← X

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PHX	DA	1	3	m

m Trek de niet-buscycli af bij 40 MHz.

PHY

Deze instructie schuift de inhoud van het Y-register op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHY : Push Y Register onto the Stack **4510**

SP ← SP − 1, STAPEL ← Y

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli
implied	PHY	5A	1	2

PHZ

Deze instructie schuift de inhoud van het Z-register op de stapel en verlaagt de waarde van de stapelaanwijzer met 1.

PHZ : Push Z Register onto the Stack **4510**

SP $\leftarrow$ SP $-$ 1, STAPEL $\leftarrow$ Z

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PHZ	DB	1	3	m

m Trek de niet-buscycli af bij 40 MHz.

PLA

Deze instructie vervangt de inhoud van het accumulatorregister door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLA : Pull Accumulator Register from the Stack **4510**

A $\leftarrow$ STAPEL, SP $\leftarrow$ SP $+$ 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLA	68	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

PLP

Deze instructie vervangt de inhoud van de processorvlaggen door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

OPMERKING: deze instructie vervangt NIET de 'Extended Stack Disable'-vlag (de E-vlag) of de software-interruptvlag (de B-vlag)

PLP : Pull Processor Flags from the Stack **4510**

P $\leftarrow$ STAPEL, SP $\leftarrow$ SP $+$ 1

N	Z	I	C	D	V	E
+	+	+	+	+	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLP	28	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

PLX

Deze instructie vervangt de inhoud van het X-register door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLX : Pull X Register from the Stack **4510**

X ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLX	FA	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

PLY

Deze instructie vervangt de inhoud van het Y-register door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLY : Pull Y Register from the Stack **4510**

Y ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLY	7A	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

PLZ

Deze instructie vervangt de inhoud van het Z-register door de bovenste waarde van de stapel en verhoogt de waarde van de stapelaanwijzer met 1.

PLZ : Pull Z Register from the Stack **4510**

Z ← STAPEL, SP ← SP + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	PLZ	FB	1	4	m

m Trek de niet-buscycli af bij 40 MHz.

RMB0

Deze instructie wist bit nul van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB0 : Reset Bit 0 in Base Page **4510**

M(0) ← 0

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	RMB0 $nn	07	2	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB1

Deze instructie wist bit 1 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB1 : Reset Bit 1 in Base Page **4510**

M(1) ← 0

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	RMB1 $nn	17	2	4	^{br}

b Voeg één cyclus toe als er een vertakking plaatsvindt.
Voeg nog één cyclus toe als de genomen vertakking een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB2

Deze instructie wist bit 2 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB2 : Reset Bit 2 in Base Page **4510**

M(2) ← 0

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	RMB2 $nn	27	2	4	^{r}

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB3

Deze instructie wist bit 3 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB3 : Reset Bit 3 in Base Page					**4510**
M(3) ← 0					
			N Z I C D V E		
			. . . .		
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**	
base-page	RMB3 $nn	37	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB4

Deze instructie wist bit 4 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB4 : Reset Bit 4 in Base Page					**4510**
M(4) ← 0					
			N Z I C D V E		
			. . . .		
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**	
base-page	RMB4 $nn	47	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB5

Deze instructie wist bit 5 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB5 : Reset Bit 5 in Base Page					**4510**
M(5) ← 0					
			N Z I C D V E		
			. . . .		
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**	
base-page	RMB5 $nn	57	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB6

Deze instructie wist bit 6 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB6 : Reset Bit 6 in Base Page				**4510**
M(6) ← 0				
			N Z I C D V E	
			. . . .	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
base-page	RMB6 $nn	67	2	5 ^{r}

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RMB7

Deze instructie wist bit 7 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

RMB7 : Reset Bit 7 in Base Page				**4510**
M(7) ← 0				
			N Z I C D V E	
			. . . .	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
base-page	RMB7 $nn	77	2	4 ^{r}

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ROL

Deze instructie verschuift de inhoud van het accumulatorregister of van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt ingesteld op de huidige waarde van de carry-vlag en bit 7 wordt uitgeschoven in de carry-vlag.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was ingesteld, voor het verschuiven.

ROL : Rotate Left Memory or Accumulator **4510**

M ← M≪1, C ← M(7), M(0) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	ROL $nn	26	2	4	r
accumulator	ROL A	2A	1	1	s
absolute	ROL $nnnn	2E	3	5	r
base-page,X	ROL $nn,X	36	2	5	pr
absolute,X	ROL $nnnn,X	3E	3	5	pr

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ROR

Deze instructie verschuift ofwel het accumulatorregister ofwel de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 7 wordt ingesteld op de huidige waarde van de carry-vlag, en bit 0 wordt in de carry-vlag verschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de waarde was gezet, voordat er werd verschoven.

ROR : Rotate Right Memory or Accumulator **4510**

M ← M≫1, C ← M(0), M(7) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	ROR $nn	66	2	5	r
accumulator	ROR A	6A	1	1	s
absolute	ROR $nnnn	6E	3	6	r
base-page,X	ROR $nn,X	76	2	5	$dmpr$
absolute,X	ROR $nnnn,X	7E	3	5	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.
s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

ROW

Deze instructie roteert de inhoud van het opgegeven geheugenwoord één bit naar links. Bit 0 van de lage byte wordt ingesteld op de huidige waarde van de carry-vlag en bit 7 van de hoge byte wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit is gezet; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.
- De C-vlag wordt gezet als bit 7 van de hoge byte voor het verschuiven gezet was.

ROW : Rotate Word Left **4510**

M2:M1 ← M2:M1≪1, C ← M2(7), M1(0) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
absolute	ROW $nnnn	EB	3	5	dmr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RTI

Deze instructie haalt de processorvlaggen van de stapel en vervolgens het Program Counter (PC)-register van de stapel, waardoor een onderbroken programma kan worden hervat.

- De 6502-processorvlaggen worden hersteld van de stapel.
- Door deze instructie wordt noch de B-vlag (software-interrupt) noch de E-vlag (uitgebreide stapel) gezet.

RTI : Return From Interrupt **4510**

P ← STAPEL, PCH:PCL ← STAPEL, SP ← SP − 3

N	Z	I	C	D	V	E
+	+	+	+	+	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	RTI	40	1	6	m

m Trek de niet-buscycli af bij 40 MHz.

RTS

Deze instructie haalt het Program Counter (PC)-register van de stapel en telt vervolgens een optioneel byte-argument op bij de stapelaanwijzer, waardoor een subroutine naar haar aanroeper kan terugkeren.

Het optionele argument is bedoeld om een aanroeper in staat te stellen argumenten voor de subroutine op de stapel te plaatsen. De subroutine leest de argumenten uit het stapelgeheugen en keert vervolgens terug met het aantal argumentbytes. De CPU haalt naast het retouradres ook de argumenten van de stapel, zodat de aanroeper de argumenten niet expliciet hoeft te verwijderen.

RTS : Return From Subroutine **4510**

PCH:PCL ← STAPEL, SP ← SP + 2 (+ M)

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	RTS	60	1	6	m
immediate 8bit	RTS #$nn	62	2	4	

m Trek de niet-buscycli af bij 40 MHz.

SBC

Deze instructie voert A − M − 1 + C uit en zet de processorvlaggen dienovereenkomstig. Het resultaat wordt in het accumulatorregister opgeslagen.

OPMERKING: Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt de vlag gewist.

SBC : Subtract With Carry **4510**

A ← − M − 1 + C

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	SBC ($nn,X)	E1	2	3	mp
base-page	SBC $nn	E5	2	3	r
immediate 8bit	SBC #$nn	E9	2	2	
absolute	SBC $nnnn	ED	3	4	r
(indirect),Y	SBC ($nn),Y	F1	2	3	pr
(indirect),Z	SBC ($nn),Z	F2	2	5	pr
base-page,X	SBC $nn,X	F5	2	3	pr
absolute,Y	SBC $nnnn,Y	F9	3	4	pr
absolute,X	SBC $nnnn,X	FD	3	4	pr

m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SEC

Deze instructie zet de carry-vlag.

Neveneffecten

- De C-vlag wordt gezet.

SEC : Set Carry Flag **4510**

C ← 1

N	Z	I	C	D	V	E
·	·	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	SEC	38	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SED

Deze instructie zet de decimalenvlag. Rekenkundige bewerkingen maken vanaf nu gebruik van de BCD-modus (Binary Coded Decimal).

OPMERKING: de interrupt-handler van de C64 wist de decimalenvlag niet, waardoor het gevaarlijk is de decimalenvlag te zetten zonder eerst de interruptblokkeervlag te zetten.

Neveneffecten

- De D-vlag wordt gezet.

SED : Set Decimal Flag				**4510**
D ← 1				
			N Z I C D V E	
			· · · · + · ·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	SED	F8	1	1 *s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SEE

Deze instructie zet de Extended Stack Disable Flag. Hierdoor werkt de stapel zoals op de 6502, d.w.z. beperkt tot één geheugenpagina. De geheugenpagina waarin de stapel zich bevindt, kan nog steeds worden gewijzigd door het Stack Pointer High (SPH)-register in te stellen.

Neveneffecten

- De E-vlag wordt gezet.

SEE : Set Extended Stack Disable Flag				**4510**
E ← 1				
			N Z I C D V E	
			· · · · · · +	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**
implied	SEE	03	1	1 *s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SEI

Deze instructie zet de interruptblokkeervlag. Gewone (IRQ-)interrupts kunnen niet langer optreden. Niet-maskeerbare interrupts (NMI) blijven optreden, zoals hun naam al aangeeft.

Neveneffecten

- De I-vlag wordt gezet.

SEI : Set Interrupt Disable Flag **4510**

I ← 1

N	Z	I	C	D	V	E
·	·	+	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	SEI	78	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SMB0

Deze instructie zet bit nul van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB0 : Set Bit 0 in Base Page **4510**

M(0) ← 1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB0 $nn	87	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB1

Deze instructie zet bit 1 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB1 : Set Bit 1 in Base Page **4510**

M(1) ← 1

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB1 $nn	97	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB2

Deze instructie zet bit 2 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB2 : Set Bit 2 in Base Page **4510**

M(2) ← 1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB2 $nn	A7	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB3

Deze instructie zet bit 3 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB3 : Set Bit 3 in Base Page **4510**

M(3) ← 1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB3 $nn	B7	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB4

Deze instructie zet bit 4 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB4 : Set Bit 4 in Base Page **4510**

M(4) ← 1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB4 $nn	C7	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB5

Deze instructie zet bit 5 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB5 : Set Bit 5 in Base Page **4510**

M(5) ← 1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB5 $nn	D7	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB6

Deze instructie zet bit 6 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB6 : Set Bit 6 in Base Page **4510**

M(6) ← 1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB6 $nn	E7	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SMB7

Deze instructie zet bit 7 van het opgegeven adres. Er worden geen vlaggen gewijzigd, ongeacht het resultaat.

SMB7 : Set Bit 7 in Base Page **4510**

M(7) ← 1

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	SMB7 $nn	F7	2	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

STA

Deze instructie slaat de inhoud van het accumulatorregister op in de aangeduide locatie.

STA : Store Accumulator **4510**

M ← A

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
(indirect,X)	STA ($nn,X)	81	2	5	p
(immediate,SP),Y	STA ($nn,SP),Y	82	2	6	p
base-page	STA $nn	85	2	3	
absolute	STA $nnnn	8D	3	4	
(indirect),Y	STA ($nn),Y	91	2	5	p
(indirect),Z	STA ($nn),Z	92	2	5	p
base-page,X	STA $nn,X	95	2	3	p
absolute,Y	STA $nnnn,Y	99	3	4	p
absolute,X	STA $nnnn,X	9D	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

STX

Deze instructie slaat de inhoud van het X-register op in de aangeduide locatie.

STX : Store X Register **4510**

M ← X

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	STX $nn	86	2	3	
absolute	STX $nnnn	8E	3	4	
base-page,Y	STX $nn,Y	96	2	3	p
absolute,Y	STX $nnnn,Y	9B	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

STY

Deze instructie slaat de inhoud van het Y-register op in de aangeduide locatie.

STY : Store Y Register **4510**

M ← Y

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	STY $nn	84	2	3	
absolute,X	STY $nnnn,X	8B	3	4	p
absolute	STY $nnnn	8C	3	4	
base-page,X	STY $nn,X	94	2	3	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

STZ

Deze instructie slaat de inhoud van het Z-register op in het opgegeven geheugenadres.

STZ : Store Z Register **4510**

M ← Z

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	STZ $nn	64	2	3	
base-page,X	STZ $nn,X	74	2	3	
absolute	STZ $nnnn	9C	3	4	
absolute,X	STZ $nnnn,X	9E	3	4	p

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

TAB

Deze instructie stelt het Base Page-register in op de inhoud van het accumulatorregister. Hierdoor kan de zero page van de 6502 naar elke pagina in het geheugen worden verplaatst.

TAB : Transfer Accumulator into Base Page Register **4510**

B ← A

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TAB	5B	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TAX

Deze instructie laadt het X-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet is, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

TAX : Transfer Accumulator Register into the X Register **4510**

X ← A

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TAX	AA	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TAY

Deze instructie laadt het Y-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet is, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

TAY : Transfer Accumulator Register into the Y Register **4510**

Y ← A

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TAY	A8	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TAZ

Deze instructie laadt het Z-register met de inhoud van het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit gezet is; anders wordt de vlag gewist.

- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

TAZ : Transfer Accumulator Register into the Z Register **4510**

Z ← A

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TAZ	4B	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TBA

Deze instructie laadt het accumulatorregister met de inhoud van het Base Page-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel het meest significante bit gezet is; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

TBA : Transfer Base Page Register into the Accumulator **4510**

A ← B

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TBA	7B	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TRB

Deze instructie voert een binair EN op de ontkrachte versie van het accumulatorregister en de aangegeven geheugenlocatie uit en slaat het resultaat daar op. Dat wil zeggen dat elk bit dat in het accumulatorregister gezet is, wordt gereset in de aangegeven geheugenlocatie.

Deze instructie voert ook een test uit op bits die zowel in het accumulatorregister als in de aangegeven geheugenlocatie voorkomen. Hiermee kunnen eenvoudige gedeeld-geheugen multiprocessorsystemen worden gebouwd, doordat een atomair mechanisme wordt geboden om een semafoor te zetten of een vergrendeling te verkrijgen.

Neveneffecten

- De Z-vlag wordt gezet als het binaire EN van het accumulatorregister en de inhoud van de aangegeven geheugenlocatie voor de uitvoering van de instructie nul is.

TRB : Test and Reset Bit **4510**

M ← M *AND* (*NOT* A)

N	Z	I	C	D	V	E
·	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	TRB $nn	14	2	5	r
absolute	TRB $nnnn	1C	3	4	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

TSB

Deze instructie voert een binair OF op het accumulatorregister en de aangegeven geheugenlocatie uit en slaat het resultaat daar op. Dat wil zeggen dat elk bit dat in het accumulatorregister gezet is, wordt gezet in de aangegeven geheugenlocatie.

Deze instructie voert ook een test uit op bits die zowel in het accumulatorregister als in de aangegeven geheugenlocatie voorkomen. Hiermee kunnen eenvoudige gedeeld-geheugen multiprocessorsystemen worden gebouwd, doordat een atomair mechanisme wordt geboden om een semafoor te zetten of een vergrendeling te verkrijgen.

Neveneffecten

- De Z-vlag wordt gezet als het binaire EN van het accumulatorregister en de inhoud van de aangegeven geheugenlocatie voor de uitvoering van de instructie nul is.

TSB : Test and Set Bit **4510**

M ← M *OR* A

N	Z	I	C	D	V	E
·	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page	TSB $nn	04	2	3	r
absolute	TSB $nnnn	0C	3	5	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

TSX

Deze instructie laadt het X-register met de inhoud van het Stack Pointer Low (SPL)-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TSX : Transfer Stack Pointer Low Register into the X Register **4510**

X ← SPL

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TSX	BA	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TSY

Deze instructie laadt het Y-register met de inhoud van het Stapelpointer High (SPH)-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit ingesteld is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

TSY : Transfer Stack Pointer High Register into the Y Register **4510**

Y ← SPH

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TSY	0B	1	1	s

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TXA

Deze instructie laadt het accumulatorregister met de inhoud van het X-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TXA : Transfer X Register into the Accumulator Register **4510**

A ← X

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TXA	8A	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TXS

Deze instructie zet de lage byte van het stapelaanwijzerregister (SPL) op de inhoud van het X-register.

TXS : Transfer X Register into Stack Pointer Low Register **4510**

SPL ← X

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TXS	9A	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TYA

Deze instructie laadt het accumulatorregister met de inhoud van het Y-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, dat wil zeggen als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

TYA : Transfer Y Register into the Accumulator Register **4510**

A ← Y

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TYA	98	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TYS

Deze instructie zet het hoge byte van de stapelaanwijzerregister (SPH) op de inhoud van het Y-register. Hierdoor kan de geheugenpagina waar de stapel zich bevindt,

worden gewijzigd (indien de “Extended Stack Disable”-vlag (E) gezet is), of kan de huidige stapelpointer op elke pagina in het geheugen worden ingesteld, indien de “Extended Stack Disable”-vlag (E) gereset is.

TYS : Transfer Y Register into Stack Pointer High Register **4510**

SPH ← Y

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TYS	2B	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

TZA

Deze instructie laadt het accumulatorregister met de inhoud van het Z-register.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, oftewel als het meest significante bit gezet is; anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt de vlag gewist.

TZA : Transfer Z Register into the Accumulator Register **4510**

A ← Z

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
implied	TZA	6B	1	1	*s*

s De instructie vereist 2 cycli wanneer de CPU op 1 MHz of 2 MHz draait.

SAMENGESTELDE INSTRUCTIES VAN DE 45GS02

Omdat de 4510 geen ongebruikte opcodes heeft, gebruikt de 45GS02 samengestelde instructies om de uitbreiding te implementeren. Deze samengestelde instructies bestaan uit één of meer instructies van één byte, welke onmiddellijk vóór een conventionele instructie worden geplaatst. Deze prefixes geven de 45GS02 de opdracht de volgende instructie anders te behandelen, zoals beschreven in Hoofdstuk/Bijlage 2 op bladzijde 17.

OPMERKING: Prefix-instructies worden door de CPU uitgevoerd alsof het zelfstandige instructies zijn, nog voordat de samengestelde instructie wordt herkend. Dit kan

van invloed zijn op de vlagstatus. De instructiebytes 42 42 8D voeren bijvoorbeeld twee NEG-instructies uit, gevolgd door de samengestelde STQ-instructie. De NEG-instructies kunnen van invloed zijn op de N- en Z-vlaggen, afhankelijk van de inhoud van het accumulatorregister.

U vindt de samengestelde instructies gemarkeerd in de 4510-opcodetabel. Zie Hoofdstuk/Bijlage 3 op bladzijde 108.

ADC

Deze instructie telt het argument en de carry-vlag op bij de inhoud van het accumulatorregister. Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten, anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.
- De C-vlag wordt gezet als het ongetekende resultaat >255 is, of >99 indien de D-vlag is gezet.

ADC : Add with carry **45GS02**

$A \leftarrow A + M + C$

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	ADC [$nn],Z	EA 72	3	7	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ADCQ

Deze instructie telt het argument en de carry-vlag op bij de inhoud van het 32-bits Q-pseudoregister.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Opmerking: indien de D-vlag gezet is, is de bewerking ongedefinieerd en kan deze nog wijzigen.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt deze gewist. Indien de vlag gezet is, geeft dit aan dat een tekenoverflow heeft plaatsgevonden.
- De C-vlag wordt gezet als het ongetekende resultaat $\geq 2^{32}$ is.

ADCQ : Add with carry Quad **45GS02**

Q ← Q + M + C

			N	Z	I	C	D	V	E	
			+	+	·	+	·	+	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page quad	ADCQ $nn	42 42 65	4	8	r					
absolute quad	ADCQ $nnnn	42 42 6D	5	9	r					
(indirect quad)	ADCQ ($nn)	42 42 72	4	10	ipr					
[indirect quad]	ADCQ [$nn]	42 42 EA 72	5	13	ipr					

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

AND

Deze instructie voert een bitgewijze EN-operatie uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister en die ook in het argument staan, worden na voltooiing ingesteld in het accumulatorregister.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit is ingesteld, anders wordt de vlag gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt de vlag gewist.

AND : Binary AND **45GS02**

A ← A *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	AND [$nn],Z	EA 32	3	7	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ANDQ

Deze instructie voert een binair EN-bewerking uit van het argument met het Q-pseudoregister en slaat het resultaat op in het accumulatorregister. Alleen bits die reeds in het Q-pseudoregister stonden en die ook in het argument zijn ingesteld, blijven in het Q-pseudoregister ingesteld na afloop.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

ANDQ : Binary AND Quad **45GS02**

Q ← Q *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	ANDQ $nn	42 42 25	4	8	r
absolute quad	ANDQ $nnnn	42 42 2D	5	9	r
(indirect quad)	ANDQ ($nn)	42 42 32	4	10	ipr
[indirect quad]	ANDQ [$nn]	42 42 EA 32	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ASLQ

Deze instructie verschuift het Q-pseudoregister, of de inhoud van de opgegeven geheugenlocatie en de drie daaropvolgende, één bit naar links, waarbij deze als een

little-endian 32-bits waarde worden behandeld. Bit 0 wordt op nul gezet en bit 31 wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De C-vlag wordt gezet als bit 31 van de waarde is ingesteld vóórdat deze werd verschoven; anders wordt deze gewist.

ASLQ : Arithmetic Shift Left Quad **45GS02**

Q ← Q<<1 *of* M ← M<<1

			N	Z	I	C	D	V	E	
			+	+	·	+	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page quad	ASLQ $nn	42 42 06	4	12	dmr					
pseudoregister Q	ASLQ Q	42 42 0A	3	3						
absolute quad	ASLQ $nnnn	42 42 0E	5	13	dmr					
base-page quad,X	ASLQ $nn,X	42 42 16	4	12	$dmpr$					
absolute quad,X	ASLQ $nnnn,X	42 42 1E	5	13	$dmpr$					

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ASRQ

Deze instructie verschuift het Q-pseudoregister, of de inhoud van de opgegeven geheugenlocatie en de drie daaropvolgende, één bit naar rechts, waarbij deze als een little-endian 32-bits waarde worden behandeld. Bit 31 wordt als tekenbit beschouwd en blijft behouden. De inhoud van bit 0 wordt in de carry-vlag geschoven.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De C-vlag wordt gezet als bit 0 van de waarde is ingesteld vóórdat deze werd verschoven; anders wordt deze gewist.

ASRQ : Arithmetic Shift Right Quad **45GS02**

Q ← Q>>1 *of* M ← M>>1

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
pseudoregister Q	ASRQ Q	42 42 43	3	3	
base-page quad	ASRQ $nn	42 42 44	4	12	dmr
base-page quad,X	ASRQ $nn,X	42 42 54	4	12	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.

m Trek de niet-buscycli af bij 40 MHz.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

BITQ

Deze instructie wordt gebruikt om de bits te testen die zijn opgeslagen in een geheugenlocatie en de drie daaropvolgende, waarbij deze als een little-endian 32-bits waarde worden behandeld. Bits 30 en 31 van de inhoud van de geheugenlocatie worden rechtstreeks gekopieerd naar de Overflow-vlag en de Negativiteit-vlag. De Zero-vlag wordt gezet of gereset op basis van het resultaat van het uitvoeren van de binaire EN-bewerking van het Q-register en de inhoud van de aangegeven geheugenlocatie.

Neveneffecten

- De N-vlag wordt gezet als bit 31 van de geheugenlocatie is ingesteld; anders wordt deze gewist.
- De V-vlag wordt gezet als bit 30 van de geheugenlocatie is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van Q *AND* M nul is; anders wordt deze gewist.

BITQ : Perform Bit Test Quad **45GS02**

N ← M(31), V ← M(30), Z ← Q *AND* M

N	Z	I	C	D	V	E
+	+	·	·	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	BITQ $nn	42 42 24	4	8	r
absolute quad	BITQ $nnnn	42 42 2C	5	9	r

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CMP

Deze instructie voert A − M uit en zet de processorvlaggen dienovereenkomstig, maar wijzigt de inhoud van het accumulatorregister niet.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A − M negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van A − M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt deze gewist.

CMP : Compare Accumulator **45GS02**

N,C,Z ⇐ [A − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	CMP [$nn],Z	EA D2	3	7	^{ipr}

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

CMPQ

Deze instructie voert Q − M uit en stelt de processorvlaggen dienovereenkomstig in, maar wijzigt de inhoud van het Q-register niet.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Q − M negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De C-vlag wordt gezet als het resultaat van Q − M nul of positief is, d.w.z. als Q niet kleiner is dan M; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat van Q − M nul is; anders wordt deze gewist.

CMPQ : Compare Q Pseudo Register **45GS02**

N,C,Z $\Leftarrow$ [Q − M]

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	CMPQ $nn	42 42 C5	4	8	r
absolute quad	CMPQ $nnnn	42 42 CD	5	9	r
(indirect quad)	CMPQ ($nn)	42 42 D2	4	10	ipr
[indirect quad]	CMPQ [$nn]	42 42 EA D2	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

DEQ

Deze instructie verlaagt het Q-pseudoregister of de opgegeven geheugenlocatie met één.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

DEQ : Decrement Quad **45GS02**

Q $\leftarrow$ Q − 1 of M $\leftarrow$ M − 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
pseudoregister Q	DEQ Q	42 42 3A	3	3	
base-page quad	DEQ $nn	42 42 C6	4	12	dmr
absolute quad	DEQ $nnnn	42 42 CE	5	13	dmr
base-page quad,X	DEQ $nn,X	42 42 D6	4	12	$dmpr$
absolute quad,X	DEQ $nnnn,X	42 42 DE	5	13	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

EOR

Deze instructie voert een bitgewijze uitsluitende-OF-operatie (XOR) uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al ingesteld stonden in het accumulatorregister of die ingesteld staan in het argument, worden na voltooiing ingesteld in het accumulatorregister, maar niet allebei.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

EOR : Binary Exclusive OR **45GS02**

A ← A *XOR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	EOR [$nn],Z	EA 52	3	7	^{ipr}

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

EORQ

Deze instructie voert een binaire EXCLUSIEF-OF-bewerking uit van het argument met het Q-pseudoregister en slaat het resultaat op in het Q-pseudoregister. Alleen bits die reeds in het Q-pseudoregister stonden of die in het argument zijn ingesteld, worden bij afronding in het Q-pseudoregister ingesteld, maar niet bits die in beide stonden.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

EORQ : Binary Exclusive OR Quad **45GS02**

Q ← Q *XOR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	EORQ $nn	42 42 45	4	8	r
absolute quad	EORQ $nnnn	42 42 4D	5	9	r
(indirect quad)	EORQ ($nn)	42 42 52	4	10	ipr
[indirect quad]	EORQ [$nn]	42 42 EA 52	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

INQ

Deze instructie verhoogt het Q-pseudoregister of de opgegeven geheugenlocatie.

Opmerking: de opgegeven geheugenlocatie wordt beschouwd als het eerste byte van een 32-bits little-endian-waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

INQ : Increment Memory or Accumulator **45GS02**

Q ← Q + 1 of M ← M + 1

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
pseudoregister Q	INQ Q	42 42 1A	3	3	
base-page quad	INQ $nn	42 42 E6	4	13	dmr
absolute quad	INQ $nnnn	42 42 EE	5	14	dmr
base-page quad,X	INQ $nn,X	42 42 F6	4	13	$dmpr$
absolute quad,X	INQ $nnnn,X	42 42 FE	5	14	dpr

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDA

Deze instructie laadt het accumulatorregister met de aangegeven waarde, of met de inhoud van de aangegeven locatie.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit gezet is; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is, anders wordt hij gewist.

LDA : Load Accumulator **45GS02**

A ← M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**		**Cycli**					
[indirect],Z	LDA [$nn],Z	EA B2	3		7					^{ipr}

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LDQ

Deze instructie laadt het Q-pseudoregister met de opgegeven waarde of met de inhoud van de opgegeven locatie. Omdat het Q-register een alias is voor A, X, Y en Z tezamen, zet deze bewerking die vier registers. A bevat de minst significante bits, X de volgende minst significante bits, vervolgens Y, en Z bevat de meest significante bits.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

LDQ : Load Q Pseudo Register **45GS02**

Q ← M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	LDQ $nn	42 42 A5	4	8	r
absolute quad	LDQ $nnnn	42 42 AD	5	9	r
(indirect quad),Z	LDQ ($nn),Z	42 42 B2	4	10	ipr
[indirect quad],Z	LDQ [$nn],Z	42 42 EA B2	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

LSRQ

Deze instructie verschuift het Q-pseudoregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 31 wordt op nul gezet en bit 0 wordt in de carry-vlag verschoven.

Opmerking: het geheugenadres wordt beschouwd als het eerste adres van een little-endian 32-bits waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit staat ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De C-vlag wordt gezet als bit 0 van de waarde voor het verschuiven gezet was; anders wordt deze gewist.

LSRQ : Logical Shift Right Quad **45GS02**

Q ← Q≫1, C ← A(0) of M ← M≫1

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	LSRQ $nn	42 42 46	4	12	dmr
pseudoregister Q	LSRQ Q	42 42 4A	3	3	
absolute quad	LSRQ $nnnn	42 42 4E	5	13	dmr
base-page quad,X	LSRQ $nn,X	42 42 56	4	12	$dmpr$
absolute quad,X	LSRQ $nnnn,X	42 42 5E	5	13	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ORA

Deze instructie voert een binaire OR-bewerking uit van het argument met het accumulatorregister en slaat het resultaat op in het accumulatorregister. Alleen bits die al in het accumulatorregister gezet waren, of die in het argument gezet waren, worden bij voltooiing in het accumulatorregister gezet, of beide.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit gezet heeft; anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt hij gewist.

ORA : Binary OR **45GS02**

A ← A *OR* M

N	Z	I	C	D	V	E
+	+	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	ORA [$nn],Z	EA 12	3	7	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ORQ

Deze instructie voert een binaire OR-bewerking uit van het argument met het Q-pseudoregister en slaat het resultaat op in het Q-pseudoregister. Alleen bits die reeds

in het Q-pseudoregister stonden of die in het argument zijn ingesteld, of beide, worden bij afronding in het Q-pseudoregister ingesteld.

Opmerking: deze bewerking behandelt het geheugenadres als het eerste adres van een 32-bits little-endian-waarde. Dat wil zeggen dat het geheugenadres en de drie daaropvolgende worden gebruikt.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.
- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.

ORQ : Binary OR Quad **45GS02**

Q ← Q *OR* M

			N	Z	I	C	D	V	E	
			+	+	·	·	·	·	·	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
base-page quad	ORQ $nn	42 42 05	4	8	r					
absolute quad	ORQ $nnnn	42 42 0D	5	9	r					
(indirect quad)	ORQ ($nn)	42 42 12	4	10	pr					
[indirect quad]	ORQ [$nn]	42 42 EA 12	5	13	pr					

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RESQ

Deze uitgebreide opcodes zijn gereserveerd en hun functie is niet gedefinieerd en kan in toekomstige revisies van de 45GS02 nog wijzigen. Ze mogen derhalve in geen enkel programma worden gebruikt.

RESQ : Reserved extended opcode **45GS02**

UNDEFINED

			N	Z	I	C	D	V	E	
			.	.	.	.	.	.	.	
Adresseermethode	**Instructie**	**Opcode**		**Bytes**		**Cycli**				
(indirect quad,X)	RESQ ($nn,X)	42 42 01		4		10	ipr			
(indirect quad),Y	RESQ ($nn),Y	42 42 11		4		10	ipr			
base-page quad,X	RESQ $nn,X	42 42 15		4		8	pr			
absolute quad,Y	RESQ $nnnn,Y	42 42 19		5		9	pr			
absolute quad,X	RESQ $nnnn,X	42 42 1D		5		9	pr			
(indirect quad,X)	RESQ ($nn,X)	42 42 21		4		10	ir			
(indirect quad),Y	RESQ ($nn),Y	42 42 31		4		10	ipr			
base-page quad,X	RESQ $nn,X	42 42 34		4		8	pr			
base-page quad,X	RESQ $nn,X	42 42 35		4		8	pr			
absolute quad,Y	RESQ $nnnn,Y	42 42 39		5		10	pr			
absolute quad,X	RESQ $nnnn,X	42 42 3C		5		9	pr			
absolute quad,X	RESQ $nnnn,X	42 42 3D		5		10	pr			
(indirect quad,X)	RESQ ($nn,X)	42 42 41		4		10	ipr			
(indirect quad),Y	RESQ ($nn),Y	42 42 51		4		10	ipr			
base-page quad,X	RESQ $nn,X	42 42 55		4		8	pr			
absolute quad,Y	RESQ $nnnn,Y	42 42 59		5		9	pr			
absolute quad,X	RESQ $nnnn,X	42 42 5D		5		9	pr			
(indirect quad,X)	RESQ ($nn,X)	42 42 61		4		10	ir			
(indirect quad),Y	RESQ ($nn),Y	42 42 71		4		10	ipr			
base-page quad,X	RESQ $nn,X	42 42 75		4		8	pr			
absolute quad,Y	RESQ $nnnn,Y	42 42 79		5		10	pr			
absolute quad,X	RESQ $nnnn,X	42 42 7D		5		10	pr			

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

ROLQ

Deze instructie verschuift het Q-pseudoregister of de inhoud van de opgegeven geheugenlocatie één bit naar links. Bit 0 wordt ingesteld op de huidige waarde van de carry-vlag en bit 31 wordt in de carry-vlag geschoven.

OPMERKING: het geheugenadres wordt beschouwd als het eerste adres van een little-endian gecodeerde 32-bits waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. het meest significante bit is ingesteld; anders wordt deze gewist.

- De Z-vlag wordt gezet als het resultaat nul is; anders wordt deze gewist.
- De C-vlag wordt gezet als bit 31 van de waarde is ingesteld vóórdat deze werd verschoven; anders wordt deze gewist.

ROLQ : Rotate Left Quad **45GS02**

M ← M≪1, C ← M(31), M(0) ← C or Q ← Q≪1, C ← Q(31), Q(0) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	ROLQ $nn	42 42 26	4	12	dmr
pseudoregister Q	ROLQ Q	42 42 2A	3	3	
absolute quad	ROLQ $nnnn	42 42 2E	5	13	dmr
base-page quad,X	ROLQ $nn,X	42 42 36	4	12	$dmpr$
absolute quad,X	ROLQ $nnnn,X	42 42 3E	5	13	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RORQ

Deze instructie verschuift het Q-pseudoregister of de inhoud van de opgegeven geheugenlocatie één bit naar rechts. Bit 31 wordt ingesteld op de huidige waarde van de carry-vlag en bit 0 wordt in de carry-vlag geschoven

Opmerking: het adres wordt beschouwd als het eerste adres van een little-endian 32-bits waarde.

Neveneffecten

- De N-vlag wordt gezet als het resultaat negatief is, d.w.z. als het meest significante bit ervan gezet is en anders wordt hij gewist.
- De Z-vlag wordt gezet als het resultaat nul is en anders wordt hij gewist.
- De C-vlag wordt gezet als bit 31 van de waarde gezet was, vóór deze verschoven werd en anders wordt hij gewist.

RORQ : Rotate Right Quad **45GS02**

M ← M≫1, C ← M(0), M(31) ← C or Q ← Q≫1, C ← Q(0), Q(31) ← C

N	Z	I	C	D	V	E
+	+	·	+	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	RORQ $nn	42 42 66	4	12	dmr
pseudoregister Q	RORQ Q	42 42 6A	3	3	
absolute quad	RORQ $nnnn	42 42 6E	5	13	dmr
base-page quad,X	RORQ $nn,X	42 42 76	4	12	$dmpr$
absolute quad,X	RORQ $nnnn,X	42 42 7E	5	13	$dmpr$

d Trek één cyclus af wanneer de CPU op 3.5 MHz draait.
m Trek de niet-buscycli af bij 40 MHz.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

RSVQ

Deze uitgebreide opcodes zijn gereserveerd en hun functie is niet gedefinieerd en kan in toekomstige revisies van de 45GS02 nog wijzigen. Ze mogen derhalve in geen enkel programma worden gebruikt.

RSVQ : Reserved extended opcode **45GS02**

UNDEFINED

			N	Z	I	C	D	V	E	
			.	.	.	.	.	.	.	
Adresseermethode	**Instructie**	**Opcode**	**Bytes**	**Cycli**						
(indirect quad,X)	RSVQ ($nn,X)	42 42 81	4	10	ip
(indirect quad,SP),Y	RSVQ ($nn,SP),Y	42 42 82	4	10	ip
(indirect quad),Y	RSVQ ($nn),Y	42 42 91	4	10	ip
base-page quad,X	RSVQ $nn,X	42 42 95	4	8	p
absolute quad,Y	RSVQ $nnnn,Y	42 42 99	5	9	p
absolute quad,X	RSVQ $nnnn,X	42 42 9D	5	9	p
(indirect quad,X)	RSVQ ($nn,X)	42 42 A1	4	10	ipr
(indirect quad,X)	RSVQ ($nn,X)	42 42 C1	4	10	ipr
(indirect quad),Y	RSVQ ($nn),Y	42 42 D1	4	10	ipr
base-page quad,X	RSVQ $nn,X	42 42 D5	4	8	pr
absolute quad,Y	RSVQ $nnnn,Y	42 42 D9	5	9	pr
absolute quad,X	RSVQ $nnnn,X	42 42 DD	5	9	pr
(indirect quad,X)	RSVQ ($nn,X)	42 42 E1	4	10	ipr
(indirect quad),Y	RSVQ ($nn),Y	42 42 F1	4	10	ipr
base-page quad,X	RSVQ $nn,X	42 42 F5	4	8	pr
absolute quad,Y	RSVQ $nnnn,Y	42 42 F9	5	8	pr
absolute quad,X	RSVQ $nnnn,X	42 42 FD	5	9	pr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

SBC

Deze instructie voert A $-$ M $-$ 1 $+$ C uit en zet de processorvlaggen dienovereenkomstig. Het resultaat wordt in het accumulatorregister opgeslagen.

OPMERKING: Indien de D-vlag is gezet, wordt de optelling uitgevoerd met behulp van BCD (Binary Coded Decimal).

Neveneffecten

- De N-vlag wordt gezet als het resultaat van A $-$ M negatief is, d.w.z. als het meest significante bit is gezet; anders wordt de vlag gewist.
- De C-vlag wordt gezet als het resultaat van A $-$ M nul of positief is, d.w.z. als A niet kleiner is dan M; anders wordt de vlag gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten; anders wordt de vlag gewist. Indien de vlag is gezet, geeft dit aan dat er een getekende overflow is opgetreden.

- De Z-vlag wordt gezet als het resultaat van A − M nul is; anders wordt de vlag gewist.

SBC : Subtract With Carry **45GS02**

A ← − M − 1 + C

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli
[indirect],Z	SBC [$nn],Z	EA F2	3	0

SBCQ

Deze instructie voert Q − M − 1 + C uit en stelt de processorvlaggen overeenkomstig in. Het resultaat wordt opgeslagen in het Q-pseudoregister.

Let op: de aangegeven geheugenplaats wordt behandeld als het eerste byte van een 32-bits little-endian waarde.

Let op: als de D-vlag gezet is, is de bewerking ongedefinieerd en onderhevig aan verandering.

Neveneffecten

- De N-vlag wordt gezet als het resultaat van Q − M negatief is, d.w.z. als het meest significante bit ervan gezet is en anders wordt hij gewist.
- De C-vlag wordt gezet als het resultaat van Q − M nul of positief is, d.w.z. als A niet kleiner is dan M en anders wordt hij gewist.
- De V-vlag wordt gezet als het resultaat een ander teken heeft dan beide argumenten en anders wordt hij gewist. Als de vlag gezet is, geeft dit aan dat er een getekende overflow is opgetreden.
- De Z-vlag wordt gezet als het resultaat van Q − M nul is en anders wordt hij gewist.

SBCQ : Subtract With Carry Quad **45GS02**

Q ← Q − M − 1 + C

N	Z	I	C	D	V	E
+	+	·	+	·	+	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	SBCQ $nn	42 42 E5	4	8	r
absolute quad	SBCQ $nnnn	42 42 ED	5	9	r
(indirect quad)	SBCQ ($nn)	42 42 F2	4	10	ipr
[indirect quad]	SBCQ [$nn]	42 42 EA F2	5	13	ipr

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.
r Voeg één cyclus toe als de kloksnelheid 40 MHz is.

STA

Deze instructie slaat de inhoud van het accumulatorregister op in de aangeduide locatie.

STA : Store Accumulator **45GS02**

M ← A

N	Z	I	C	D	V	E
·	·	·	·	·	·	·

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
[indirect],Z	STA [$nn],Z	EA 92	3	8	ip

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.
p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

STQ

Deze instructie slaat de inhoud van het Q-pseudoregister op in de opgegeven geheugenlocatie.

Aangezien Q is opgebouwd uit A, X, Y en Z, betekent dit dat deze vier registers respectievelijk worden geschreven naar de opgegeven geheugenlocatie en de drie daaropvolgende geheugenlocaties.

Opmerking: De CPU voert het 42 42-voorvoegsel van de STQ-instructie uit als twee NEG-instructies, voordat hij de STQ-instructie herkent en uitvoert. Dit beïnvloedt de N- en Z-vlag, afhankelijk van de inhoud van het accumulatorregister.

STQ : Store Q **45GS02**

M ← A, M+1 ← X, M+2 ← Y, M+3 ← Z

N	Z	I	C	D	V	E
.	.	.	.	.	.	.

Adresseermethode	Instructie	Opcode	Bytes	Cycli	
base-page quad	STQ $nn	42 42 85	4	8	
absolute quad	STQ $nnnn	42 42 8D	5	9	
(indirect quad)	STQ ($nn)	42 42 92	4	10	ip
[indirect quad]	STQ [$nn]	42 42 EA 92	5	13	ip

i Voeg één cyclus toe als de kloksnelheid 40 MHz is.

p Voeg één cyclus toe als indexering een paginagrens overschrijdt.

HOOFDSTUK 4

Naslagtabellen

- Opslageenheden
- Getalstelselomrekening

OPSLAGEENHEDEN

Eenheid	Is gelijk aan	Afkorting
1 Bit		
1 Nibble	4 Bits	
1 Byte	8 bits	B
1 Kilobyte	1024 B	KB
1 Megabyte	1024KB of 1048576 B	MB

GETALSTELSELOMREKENING

Decimaal	Binair	Hexadecimaal
0	%0	$0
1	%1	$1
2	%10	$2
3	%11	$3
4	%100	$4
5	%101	$5
6	%110	$6
7	%111	$7
8	%1000	$8
9	%1001	$9
10	%1010	$A
11	%1011	$B
12	%1100	$C
13	%1101	$D
14	%1110	$E
15	%1111	$F
16	%10000	$10
17	%10001	$11
18	%10010	$12
19	%10011	$13
20	%10100	$14
21	%10101	$15
22	%10110	$16
23	%10111	$17
24	%11000	$18
25	%11001	$19
26	%11010	$1A
27	%11011	$1B
28	%11100	$1C
29	%11101	$1D
30	%11110	$1E
31	%11111	$1F

Decimaal	Binair	Hexadecimaal
32	%100000	$20
33	%100001	$21
34	%100010	$22
35	%100011	$23
36	%100100	$24
37	%100101	$25
38	%100110	$26
39	%100111	$27
40	%101000	$28
41	%101001	$29
42	%101010	$2A
43	%101011	$2B
44	%101100	$2C
45	%101101	$2D
46	%101110	$2E
47	%101111	$2F
48	%110000	$30
49	%110001	$31
50	%110010	$32
51	%110011	$33
52	%110100	$34
53	%110101	$35
54	%110110	$36
55	%110111	$37
56	%111000	$38
57	%111001	$39
58	%111010	$3A
59	%111011	$3B
60	%111100	$3C
61	%111101	$3D
62	%111110	$3E
63	%111111	$3F

Decimaal	Binair	Hexadecimaal
64	%1000000	$40
65	%1000001	$41
66	%1000010	$42
67	%1000011	$43
68	%1000100	$44
69	%1000101	$45
70	%1000110	$46
71	%1000111	$47
72	%1001000	$48
73	%1001001	$49
74	%1001010	$4A
75	%1001011	$4B
76	%1001100	$4C
77	%1001101	$4D
78	%1001110	$4E
79	%1001111	$4F
80	%1010000	$50
81	%1010001	$51
82	%1010010	$52
83	%1010011	$53
84	%1010100	$54
85	%1010101	$55
86	%1010110	$56
87	%1010111	$57
88	%1011000	$58
89	%1011001	$59
90	%1011010	$5A
91	%1011011	$5B
92	%1011100	$5C
93	%1011101	$5D
94	%1011110	$5E
95	%1011111	$5F

Decimaal	Binair	Hexadecimaal
96	%1100000	$60
97	%1100001	$61
98	%1100010	$62
99	%1100011	$63
100	%1100100	$64
101	%1100101	$65
102	%1100110	$66
103	%1100111	$67
104	%1101000	$68
105	%1101001	$69
106	%1101010	$6A
107	%1101011	$6B
108	%1101100	$6C
109	%1101101	$6D
110	%1101110	$6E
111	%1101111	$6F
112	%1110000	$70
113	%1110001	$71
114	%1110010	$72
115	%1110011	$73
116	%1110100	$74
117	%1110101	$75
118	%1110110	$76
119	%1110111	$77
120	%1111000	$78
121	%1111001	$79
122	%1111010	$7A
123	%1111011	$7B
124	%1111100	$7C
125	%1111101	$7D
126	%1111110	$7E
127	%1111111	$7F

Decimaal	Binair	Hexadecimaal
128	%10000000	$80
129	%10000001	$81
130	%10000010	$82
131	%10000011	$83
132	%10000100	$84
133	%10000101	$85
134	%10000110	$86
135	%10000111	$87
136	%10001000	$88
137	%10001001	$89
138	%10001010	$8A
139	%10001011	$8B
140	%10001100	$8C
141	%10001101	$8D
142	%10001110	$8E
143	%10001111	$8F
144	%10010000	$90
145	%10010001	$91
146	%10010010	$92
147	%10010011	$93
148	%10010100	$94
149	%10010101	$95
150	%10010110	$96
151	%10010111	$97
152	%10011000	$98
153	%10011001	$99
154	%10011010	$9A
155	%10011011	$9B
156	%10011100	$9C
157	%10011101	$9D
158	%10011110	$9E
159	%10011111	$9F

Decimaal	Binair	Hexadecimaal
160	%10100000	$A0
161	%10100001	$A1
162	%10100010	$A2
163	%10100011	$A3
164	%10100100	$A4
165	%10100101	$A5
166	%10100110	$A6
167	%10100111	$A7
168	%10101000	$A8
169	%10101001	$A9
170	%10101010	$AA
171	%10101011	$AB
172	%10101100	$AC
173	%10101101	$AD
174	%10101110	$AE
175	%10101111	$AF
176	%10110000	$B0
177	%10110001	$B1
178	%10110010	$B2
179	%10110011	$B3
180	%10110100	$B4
181	%10110101	$B5
182	%10110110	$B6
183	%10110111	$B7
184	%10111000	$B8
185	%10111001	$B9
186	%10111010	$BA
187	%10111011	$BB
188	%10111100	$BC
189	%10111101	$BD
190	%10111110	$BE
191	%10111111	$BF

Decimaal	Binair	Hexadecimaal
192	%11000000	$C0
193	%11000001	$C1
194	%11000010	$C2
195	%11000011	$C3
196	%11000100	$C4
197	%11000101	$C5
198	%11000110	$C6
199	%11000111	$C7
200	%11001000	$C8
201	%11001001	$C9
202	%11001010	$CA
203	%11001011	$CB
204	%11001100	$CC
205	%11001101	$CD
206	%11001110	$CE
207	%11001111	$CF
208	%11010000	$D0
209	%11010001	$D1
210	%11010010	$D2
211	%11010011	$D3
212	%11010100	$D4
213	%11010101	$D5
214	%11010110	$D6
215	%11010111	$D7
216	%11011000	$D8
217	%11011001	$D9
218	%11011010	$DA
219	%11011011	$DB
220	%11011100	$DC
221	%11011101	$DD
222	%11011110	$DE
223	%11011111	$DF

Decimaal	Binair	Hexadecimaal
224	%11100000	$E0
225	%11100001	$E1
226	%11100010	$E2
227	%11100011	$E3
228	%11100100	$E4
229	%11100101	$E5
230	%11100110	$E6
231	%11100111	$E7
232	%11101000	$E8
233	%11101001	$E9
234	%11101010	$EA
235	%11101011	$EB
236	%11101100	$EC
237	%11101101	$ED
238	%11101110	$EE
239	%11101111	$EF
240	%11110000	$F0
241	%11110001	$F1
242	%11110010	$F2
243	%11110011	$F3
244	%11110100	$F4
245	%11110101	$F5
246	%11110110	$F6
247	%11110111	$F7
248	%11111000	$F8
249	%11111001	$F9
250	%11111010	$FA
251	%11111011	$FB
252	%11111100	$FC
253	%11111101	$FD
254	%11111110	$FE
255	%11111111	$FF

HOOFDSTUK 5

Steuners & donateurs

- Organisaties
- Bijdragers
- Steuners

De MEGA65 zou niet tot stand zijn gekomen zonder de gulle steun van veel organisaties en individuen.

Wij stellen deze lijsten nog steeds samen; onze excuses indien u er nog niet in voorkomt. Indien u iemand kent die wij over het hoofd hebben gezien, laat het ons dan weten. Zo kunnen wij de bijdrage erkennen van iedereen die de MEGA65 mogelijk heeft gemaakt en die er het geweldige retrocomputingproject van heeft gemaakt dat de computer inmiddels is.

ORGANISATIES

Het MEGA Museum of Electronic Games & Art e.V. Duitsland
ALLES

Trenz Electronic, Duitsland
MOEDERBORD **FABRICAGE** **VERKOOP**

Hintsteiner, Oostenrijk
BEHUIZING

GMK, Duitsland
TOETSENBORD

KEVAG Telekom, Duitsland
WEBHOSTING

BIJDRAGERS

Andreas Liebeskind
(libi in paradize)
Financieel directeur MEGA eV

Thomas Hertzler
(grumpyninja)
Woordvoerder voor de VS

Russell Peake
(rdpeake)
Bugherder

Alexander Nik Petra
(n0d)
Vroeg ontwerp van de behuizing

Ralph Egas
(0-limits)
Zakelijk adviseur

Lucas Moss
PCB-ontwerp voor de MEGAphone

Daren Klamer
(Impakt)
Proeflezen van de handleiding

Daniël Mantione
(dmantione)
C64-hardwareguru

Dr. Canan Hastik
(indica)
Voorzitter MEGA eV

Simon Jameson
(Shallan)
Platformverbeteringen

Stephan Kleinert
(ubik)
Vernietiger van BASIC 10

Wayne Johnson
(sausage)
Aanvullingen op de handleiding

Raven L. A. Kleiss
(LAK132)
MegaWAT-presentatiesoftware

Maurice van Gils
(Maurice)
BASIC 65-voorbeeldprogramma's

Andrew Owen
(Cheveron)
Toetsenbord, Sinclair-ondersteuning

Adam Barnes
(amb5l)
HDMI-expert en printplaatherziening

Wayne Rittimann, Jr.
(johnwayner)
Bugverplettering op alle niveaus

STEUNERS

3c74ce64
8-Bit Classics
@11110110100
Aaron Smith
Achim Mrotzek
Adolf Nefischer
Adrian Esdaile
Adrien Guichard
Ahmed Kablaoui
Alan Bastian Witkowski
Alan Field
Alastair Paulin-Campbell
Alberto Mercuri
Alexander Haering
Alexander Kaufmann
Alexander Niedermeier
Alexander Soppart
Alfonso Ardire
Amiga On The Lake
André Kudra
André Simeit
André Wösten
Andrea Farolfi
Andrea Minutello
Andreas Behr
Andreas Freier
Andreas Grabski
Andreas Millinger
Andreas Nopper
Andreas Ochs
Andreas Wendel Manufaktur
Andreas Zschunke
Andrew Bingham
Andrew Dixon
Andrew Mondt
Andrzej Hłuchyj
Andrzej Sawiniec
Andrzej Śliwa
Anthony W. Leal
Arkadiusz Bronowicki
Arkadiusz Kwasny
Arnaud Léandre
Arne Drews
Arne Neumann
Arne Richard Tyarks
Axel Klahr
Balaz Ondrej
Barry Thompson
Bartol Filipovic
Benjamin Maas
Bernard Alaiz
Bernhard Zorn
Bieno Marti-Braitmaier
Bigby
Bill LaGrue
Bjoerg Stojalowski
Björn Johannesson
Bjørn Melbøe
Bo Goeran Kvamme
Boerge Noest
Bolko Beutner
Brett Hallen
Brian Gajewski
Brian Green
Brian Juul Nielsen
Brian Reiter
Bryan Pope
Burkhard Franke
Byron Goodman
Cameron Roberton (KONG)
Carl Angervall
Carl Danowski
Carl Stock
Carl Wall
Carlo Pastore
Carlos Silva
Carsten Sørensen
Cenk Miroglu Miroglu
Chang sik Park
Charles A. Hutchins Jr.
Chris Guthrey
Chris Hooper
Chris Stringer
Christian Boettcher
Christian Eick
Christian Gleinser
Christian Gräfe
Christian Heffner
Christian Kersting
Christian Schiller
Christian Streck
Christian Weyer
Christian Wyk
Christoph Haug
Christoph Huck
Christoph Pross
Christopher Christopher
Christopher Kalk
Christopher Kohlert
Christopher Nelson
Christopher Taylor
Christopher Whillock
Claudio Piccinini
Claus Skrepek
Collen Blijenberg
Constantine Lignos
Crnjaninja
Daniel Auger
Daniel Julien
Daniel Lobitz
Daniel O'Connor
Daniel Teicher
Daniel Tootill
Daniel Wedin
Daniele Benetti
Daniele Gaetano Capursi
Dariusz Szczesniak
Darrell Westbury
David Asenjo Raposo
David Dillard
David Gorgon
David Norwood
David Raulo
David Ross
de voughn accooe
Dean Scully
Dennis Jeschke
Dennis Schaffers
Dennis Schierholz

Dennis Schneck
denti
Dick van Ginkel
Diego Barzon
Dierk Schneider
Dietmar Krueger
Dietmar Schinnerl
Dirk Becker
Dirk Wouters
Domingo Fivoli
DonChaos
Donn Lasher
Douglas Johnson
Dr. Leopold Winter
Dusan Sobotka
Earl Woodman
Ed Reilly
Edoardo Auteri
Eduardo Gallardo
Eduardo Luis Arana
Eirik Juliussen Olsen
Emilio Monelli
EP Technical Services
Epic Sound
Erasmus Kuhlmann
ergoGnomik
Eric Hilaire
Eric Hildebrandt
Eric Hill
Eric Jutrzenka
Erwin Reichel
Espen Skog
Evangelos Mpouras
Ewan Curtis
Fabio Zanicotti
Fabrizio Di Dio
Fabrizio Lodi
FARA Gießen GmbH
FeralChild
First Choice Auto's
Florian Rienhardt
Forum64. de
Francesco Baldassarri
Frank Fechner
Frank Glaush
Frank Gulasch
Frank Haaland
Frank Hempel
Frank Koschel
Frank Linhares
Frank Sleeuwaert
Frank Wolf
FranticFreddie
Fredrik Ramsberg
Fridun Nazaradeh
Friedel Kropp
Garrick West
Gary Lake-Schaal
Gary Pearson
Gavin Jones
Geir Sigmund Straume
Gerd Mitlaender
Giampietro Albiero
Giancarlo Valente
Gianluca Girelli
Giovanni Medina
Glen Fraser
Glen R Perye III
Glenn Main
Gordon Rimac
GRANT BYERS
Grant Louth
Gregor Bubek
Gregor Gramlich
Guido Ling
Guido von Gösseln
Guillaume Serge
Gunnar Hemmerling
Günter Hummel
Guy Simmons
Guybrush Threepwood
Hakan Blomqvist
Hans Pronk
Hans-Jörg Nett
Hans-Martin Zedlitz
Harald Dosch
Harri Salokorpi
Harry Culpan
Harry Venema
Heath Gallimore
Heinz Roesner
Heinz Stampfli
Helge Förster
Hendrik Fensch
Henning Harperath
Henri Parfait
Henrik Kühn
Holger Burmester
Holger Sturk
Howard Knibbs
Hubert de Hollain
Huberto Kusters
Hugo Maria Gerardus v.d. Aa
Humberto Castaneda
Ian Cross
IDE64 Staff
Igor Ianov
Igor Kurtes
Immo Beutler
Ingo Katte
Ingo Keck
Insanely Interested Publishing
IT-Dienstleistungen Obsieger
Ivan Elwood
Jaap HUIJSMAN
Jace Courville
Jack Wattenhofer
Jakob Schönpflug
Jakub Tyszko
James Hart
James Marshburn
James McClanahan
James Sutcliffe
Jan Bitruff
Jan Hildebrandt
Jan Iemhoff
Jan Kösters
Jan Peter Borsje
Jan Schulze
Jan Stoltenberg-Lerche
Janne Tompuri
Jannis Schulte
Jari Loukasmäki
Jason Smith
Javier Gonzalez Gonzalez
Jean-Paul Lauque
Jeffrey van der Schilden
Jens Schneider

Jens-Uwe Wessling
Jesse DiSimone
Jett Adams
Johan Arneklev
Johan Berntsson
Johan Svensson
Johannes Fitz
John Cook
John Deane
John Dupuis
John Nagi
John Rorland
John Sargeant
John Traeholt
Jon Sandelin
Jonas Bernemann
Jonathan Prosise
Joost Honig
Jordi Pakey-Rodriguez
Jöre Weber
Jörg Jungermann
Jörg Schaeffer
Jörg Weese
Josef Hesse
Josef Soucek
Josef Stohwasser
Joseph Clifford
Joseph Gerth
Jovan Crnjanin
Juan Pablo Schisano
Juan S. Cardona Iguina
JudgeBeeb
Juliussen Olsen
Juna Luis Fernandez Garcia
Jürgen Endras
Jürgen Herm Stapelberg
Jyrki Laurila
Kai Pernau
Kalle Pöyhönen
Karl Lamford
Karl-Heinz Blum
Karsten Engstler
Karsten Westebbe
katarakt
Keith McComb
Kenneth Dyke
Kenneth Joensson
Kevin Edwards
Kevin Thomasson
Kim Jorgensen
Kim Rene Jensen
Kimmo Hamalainen
Konrad Buryło
Kosmas Einbrodt
Kurt Klemm
Lachlan Glaskin
Large bits collider
Lars Becker
Lars Edelmann
Lars Slivsgaard
Lasse Lambrecht
Lau Olivier
Lee Chatt
Loan Leray
Lorenzo Quadri
Lorenzo Travagli
Lorin Millsap
Lothar James Foss
Lothar Serra Mari
Luca Papinutti
Ludek Smetana
Lukas Burger
Lutz-Peter Buchholz
Luuk Spaetgens
Mad Web Skills
MaDCz
Magnus Wiklander
Maik Diekmann
Malte Mundt
Manfred Wittemann
Manuel Beckmann
Manzano Mérida
Marc "3D-vice" Schmitt
Marc Bartel
Marc Jensen
Marc Schmidt
Marc Theunissen
Marc Tutor
Marc Wink
Marcel Buchtmann
Marcel Kante
Marco Beckers
Marco Cappellari
Marco Rivela
Marco van de Water
Marcus Gerards
Marcus Herbert
Marcus Linkert
Marek Pernicky
Mario Esposito
Mario Fetka
Mario Teschke
Mariusz Tymków
Mark Adams
Mark Anderson
Mark Green
Mark Hucker
Mark Leitiger
Mark Spezzano
Mark Watkin
Marko Rizvic
Markus Bieler
Markus Bonet
Markus Dauberschmidt
Markus Fehr
Markus Fuchs
Markus Guenther-Hirn
Markus Liukka
Markus Merz
Markus Roesgen
Markus Uttenweiler
Martin Bauhuber
Martin Benke
Martin Gendera
Martin Groß
Martin Gutenbrunner
Martin Johansen
Martin Marbach
Martin Sonnleitner
Martin Steffen
Marvin Hardy
Massimo Villani
Mathias Dellacherie
Mathieu Chouinard
Matthew Adams
Matthew Browne
Matthew Carnevale
Matthew Palmer

Matthew Santos
Matthias Barthel
Matthias Dolenc
Matthias Fischer
Matthias Frey
Matthias Grandis
Matthias Guth
Matthias Lampe
Matthias Meier
Matthias Mueller
Matthias Nofer
Matthias Schonder
Maurice Al-Khaliedy
Max Ihlenfeldt
Meeso Kim
Michael Dailly
Michael Dötsch
Michael Dreßel
Michael Fichtner
Michael Fong
Michael Geoffrey Stone
Michael Gertner
Michael Grün
Michael Habel
Michael Härtig
Michael Haynes
Michael J Burkett
Michael Jensen
Michael Jurisch
Michael Kappelgaard
Michael Kleinschmidt
Michael Lorenz
Michael Mayerhofer
Michael Nurney
Michael Rasmussen
Michael Richmond
Michael Sachse
Michael Sarbak
Michael Schneider
Michael Scholz
Michael Timm
Michael Traynor
Michael Whipp
Michal Ursiny
Michele Chiti
Michele Perini
Michele Porcu
Miguel Angel Rodriguez Jodar
Mikael Lund
Mike Betz
Mike Kastrantas
Mike Pikowski
Mikko Hämäläinen
Mikko Suontausta
Mirko Roller
Miroslav Karkus
Morgan Antonsson
Moritz
Morten Nielsen
MUBIQUO APPS,SL
Myles Cameron-Smith
Neil Moore
Nelson
neoman
Nicholas Melnick
Nikolaj Brinch Jørgensen
Nils Andreas
Nils Eilers
Nils Hammerich
Nils77
Norah Smith
Norman King
Normen Zoch
Olaf Grunert
Ole Eitels
Oliver Boerner
Oliver Brüggmann
Oliver Graf
Oliver Smith
Olivier Bori
ONEPSI LLC
oRdYNe
Osaühing Trioflex
OSHA-PROS USA
Padawer
Patrick Becher
Patrick Bürckstümmer
Patrick de Zoete
Patrick Toal
Patrick Vogt
Paul Alexander Warren
Paul Gerhardt (KONG)
Paul Jackson
Paul Johnson
Paul Kuhnast (mindrail)
Paul Massay
Paul Westlake
Paul Woegerer
Pauline Brasch
Paulo Apolonia
Pete Collin
Pete of Retrohax.net
Peter Eliades
Peter Gries
Peter Habura
Peter Herklotz
Peter Huyoff
Peter Knörzer
Peter Leswell
Peter Weile
Petri Alvinen
Philip Marien
Philip Timmermann
Philipp Rudin
Pierre Kressmann
Pieter Labie
Piotr Kmiecik
Power-on.at
Przemysław Safonow
Que Labs
R Welbourn
R-Flux
Rafał Michno
Rainer Kappler
Rainer Kopp
Rainer Weninger
Ralf Griewel
Ralf Pöscha
Ralf Reinhardt
Ralf Schenden
Ralf Smolarek
Ralf Zenker
Ralph Bauer
Ralph Wernecke
Rédl Károly
Reiner Lanowski
Remi Veilleux
Riccardo Bianchi

Richard Englert
Richard Good
Richard Menedetter
Richard Sopuch
Rick Reynolds
Rico Gruninger
Rob Dean
Robert Bernardo
Robert Eaglestone
Robert Grasböck
Robert Miles
Robert Schwan
Robert Shively
Robert Tangmar
Robert Trangmar
Rodney Xerri
Roger Olsen
Roger Pugh
Roland Attila Kett
Roland Evers
Roland Schatz
Rolf Hass
Ronald Cooper
Ronald Hunn
Ronny Hamida
Ronny Preiß
Roy van Zundert
Rüdiger Wohlfromm
Ruediger Schlenter
Rutger WIllemsen
Sampo Peltonen
Sarmad Gilani
SAS74
Sascha Hesse
Scott Halman
Scott Hollier
Scott Robison
Sebastian Baranski
Sebastian Bölling
Sebastian Felzmann
Sebastian Lipp
Sebastian Rakel
Şemseddin Moldibi
Śeth Morabito
Shawn McKee
Siegfried Hartmann
Zytex Online Store
Sigurbjorn Larusson
Sigurdur Finnsson
Simon Lawrence
Simon Wolf
spreen.digital
Stefan Haberl
Stefan Kramperth
Stefan Richter
Stefan Schultze
Stefan Sonnek
Stefan Theil
Stefan Vrampe
Stefano Canali
Stefano Mozzi
Steffen Reiersen
Stephan Bielmann
Stephen Jones
Stephen Kew
Steve Gray
Steve Kurlin
Steve Lemieux
Steven Combs
Stewart Dunn
Stuart Marsh
Sven Neumann
Sven Stache
Sven Sternberger
Sven Wiegand
Szabolcs Bence
Tantrumedia Limited
Techvana Operations Ltd.
Teddy Turmeaux
Teemu Korvenpää
The Games Foundation
Thierry Supplisson
Thieu-Duy Thai
Thomas Bierschenk
Thomas Edmister
Thomas Frauenknecht
Thomas Gitzen
Thomas Gruber
Thomas Haidler
Thomas Jager
Thomas Karlsen
Thomas Laskowski
Thomas Marschall
Thomas Niemann
Thomas Scheelen
Thomas Schilling
Thomas Tahsin-Bey
Thomas Walter
Thomas Wirtzmann
Thorsten Knoll
Thorsten Nolte
Tim Krome
Tim Waite
Timo Weirich
Timothy Blanks
Timothy Henson
Timothy Prater
Tobias Butter
Tobias Heim
Tobias Köck
Tobias Lüthi
Tommi Vasarainen
Toni Ammer
Tore Olsen
Torleif Strand
Torsten Schröder
Tuan Nguyen
Uffe Jakobsen
Ulrich Hintermeier
Ulrich Nieland
Ulrik Kruse
Urban Lindeskog
Ursula Förstle
Uwe Anfang
Uwe Boschanski
Vedran Vrbanc
Verm Project
Wayne Rittimann
Wayne Sander
Wayne Steele
Who Knows
Winfried Falkenhahn
Wolfgang Becker
Wolfgang Stabla
Worblehat
www.patop69.net
Yan B
Zoltan Markus
Zsolt Zsila

Bibliografie

[1] L. Soares and M. Stumm, "Flexsc: Flexible system call scheduling with exception-less system calls." in *Osdi*, vol. 10, 2010, pp. 1–8.

[2] N. Montfort, P. Baudoin, J. Bell, I. Bogost, J. Douglass, M. C. Marino, M. Mateas, C. Reas, M. Sample, and N. Vawter, *10 PRINT CHR $(205.5+ RND (1));: GOTO 10*. MIT Press, 2012.

[3] Actraiser, "Vic-ii for beginners: Screen modes, cheaper by the dozen," 2013. [Online]. Available: http://dustlayer.com/vic-ii/2013/4/26/vic-ii-for-beginners-screen-modes-cheaper-by-the-dozen

Index

$00 (PORTDDR), 30
$01 (C64MAP), 30

ADC, 68, 110, 164
ADCQ, 28, 164
ALR, 68
ANC, 69
AND, 69, 110, 165
ANDQ, 28, 166
ARR, 70
ASL, 71, 111
ASLQ, 28, 166
ASR, 112
ASRQ, 28, 167
ASW, 112

BASIC 65-opdrachten
    BANK, 4
    DMA, 4
BBR0, 113
BBR1, 113
BBR2, 114
BBR3, 114
BBR4, 114
BBR5, 115
BBR6, 115
BBR7, 116
BBS0, 116
BBS1, 116
BBS2, 117
BBS3, 117
BBS4, 117
BBS5, 118
BBS6, 118
BBS7, 118
BCC, 71, 119
BCS, 71, 119
BEQ, 72, 119
BIT, 72, 120
BITQ, 28, 168
BMI, 73, 120
BNE, 73, 121
BPL, 73, 121
BRA, 122
BRK, 74, 122
BSR, 123
BVC, 74, 123
BVS, 75, 123

C64MAP, 30
CLC, 75, 124
CLD, 75, 124
CLE, 125
CLI, 76, 125
CLV, 76, 125
CMP, 76, 96, 126, 169
CMPQ, 169
CPQ, 28
CPX, 77, 126
CPY, 78, 127
CPZ, 128

DCP, 78
DEC, 79, 128
DEQ, 28, 170
DEW, 129
DEX, 79, 129
DEY, 80, 130
DEZ, 130

EOM, 131
EOR, 80, 132, 171
EORQ, 28, 171

Geheugenbanking, 21–23

INC, 81, 132
INQ, 28, 172
INW, 133
INX, 81, 133
INY, 82, 134
INZ, 134
ISC, 82

JMP, 83, 135
JSR, 83, 135

KIL, 84

LAS, 84
LAX, 85
LDA, 86, 136, 173
LDQ, 28, 173
LDX, 87, 137
LDY, 87, 137
LDZ, 138
LSR, 88, 138
LSRQ, 28, 174

MAP, 21–23, 139

NEG, 140
NOP, 88

ORA, 89, 140, 175
ORQ, 28, 175

PHA, 90, 141
PHP, 90, 141
PHW, 142
PHX, 142
PHY, 142
PHZ, 143
PLA, 91, 143
PLP, 91, 143
PLX, 144
PLY, 144
PLZ, 144
PORTDDR, 30

Registers
- $D640, 30, 42
- $D641, 30, 42
- $D642, 30
- $D643, 30, 42
- $D644, 30, 42
- $D645, 30, 42
- $D646, 30, 42
- $D647, 30, 42
- $D648, 30, 42
- $D649, 30, 42
- $D64A, 31, 43
- $D64B, 31, 43
- $D64C, 31, 43
- $D64D, 31, 43
- $D64E, 31, 43
- $D64F, 31, 43
- $D650, 31, 43
- $D651, 31, 43
- $D652, 31, 43
- $D653, 31, 43
- $D654, 31, 43
- $D655, 31, 43
- $D656, 31, 43
- $D657, 31, 43
- $D658, 31, 43
- $D659, 31, 43
- $D65A, 31
- $D65B, 31
- $D65C, 31
- $D65D, 31, 43
- $D65E, 31, 43
- $D65F, 31, 43
- $D660, 31, 43
- $D661, 31, 43
- $D662, 31, 43
- $D663, 31, 43
- $D664, 31, 43
- $D665, 31, 43
- $D666, 31, 43
- $D667, 31, 43
- $D668, 31, 43
- $D669, 31, 43
- $D66A, 31, 43
- $D66B, 31, 43
- $D66C, 31, 43
- $D66D, 31, 43
- $D66E, 31, 43
- $D66F, 31, 43
- $D670, 31, 43
- $D671, 31, 43
- $D672, 31, 43
- $D673, 32
- $D674, 32

$D675, 32
$D676, 32
$D677, 32
$D678, 32
$D679, 32
$D67A, 32
$D67B, 32
$D67C, 32, 43
$D67D, 32, 43
$D67E, 32, 43
$D67F, 32, 44
$D70F, 34
$D710, 32
$D768, 34
$D769, 34
$D76A, 34
$D76B, 34
$D76C, 34
$D76D, 34
$D76E, 34
$D76F, 34
$D770, 34
$D771, 34
$D772, 34
$D773, 34
$D774, 34
$D775, 34
$D776, 34
$D777, 34
$D778, 34
$D779, 34
$D77A, 34
$D77B, 34
$D77C, 34
$D77D, 34
$D77E, 34
$D77F, 34
$D780, 34
$D781, 34
$D782, 34
$D783, 34
$D784, 34
$D785, 34
$D786, 34
$D787, 34
$D788, 34
$D789, 34
$D78A, 34
$D78B, 34
$D78C, 34
$D78D, 34
$D78E, 34
$D78F, 34
$D790, 34
$D791, 35
$D792, 35
$D793, 35
$D794, 35
$D795, 35
$D796, 35
$D797, 35
$D798, 35
$D799, 35
$D79A, 35
$D79B, 35
$D79C, 35
$D79D, 35
$D79E, 35
$D79F, 35
$D7A0, 35
$D7A1, 35
$D7A2, 35
$D7A3, 35
$D7A4, 35
$D7A5, 35
$D7A6, 35
$D7A7, 35
$D7A8, 35
$D7A9, 35
$D7AA, 35
$D7AB, 35
$D7AC, 35
$D7AD, 35
$D7AE, 35

$D7AF, 35
$D7B0, 35
$D7B1, 35
$D7B2, 35
$D7B3, 35
$D7B4, 35
$D7B5, 35
$D7B6, 35
$D7B7, 35
$D7B8, 35
$D7B9, 35
$D7BA, 36
$D7BB, 36
$D7BC, 36
$D7BD, 36
$D7BE, 36
$D7BF, 36
$D7C0, 36
$D7C1, 36
$D7C2, 36
$D7C3, 36
$D7C4, 36
$D7C5, 36
$D7C6, 36
$D7C7, 36
$D7C8, 36
$D7C9, 36
$D7CA, 36
$D7CB, 36
$D7CC, 36
$D7CD, 36
$D7CE, 36
$D7CF, 36
$D7D0, 36
$D7D1, 36
$D7D2, 36
$D7D3, 36
$D7D4, 36
$D7D5, 36
$D7D6, 36
$D7D7, 36
$D7D8, 36
$D7D9, 36
$D7DA, 36
$D7DB, 37
$D7DC, 37
$D7DD, 37
$D7DE, 37
$D7DF, 37
$D7E0, 37
$D7E1, 37
$D7E2, 37
$D7E3, 37
$D7EF, 32
$D7FA, 32
$D7FB, 32
$D7FD, 32
$D7FE, 32
54848, 30, 42
54849, 30, 42
54850, 30
54851, 30, 42
54852, 30, 42
54853, 30, 42
54854, 30, 42
54855, 30, 42
54856, 30, 42
54857, 30, 42
54858, 31, 43
54859, 31, 43
54860, 31, 43
54861, 31, 43
54862, 31, 43
54863, 31, 43
54864, 31, 43
54865, 31, 43
54866, 31, 43
54867, 31, 43
54868, 31, 43
54869, 31, 43
54870, 31, 43
54871, 31, 43
54872, 31, 43
54873, 31, 43

54874, 31
54875, 31
54876, 31
54877, 31, 43
54878, 31, 43
54879, 31, 43
54880, 31, 43
54881, 31, 43
54882, 31, 43
54883, 31, 43
54884, 31, 43
54885, 31, 43
54886, 31, 43
54887, 31, 43
54888, 31, 43
54889, 31, 43
54890, 31, 43
54891, 31, 43
54892, 31, 43
54893, 31, 43
54894, 31, 43
54895, 31, 43
54896, 31, 43
54897, 31, 43
54898, 31, 43
54899, 32
54900, 32
54901, 32
54902, 32
54903, 32
54904, 32
54905, 32
54906, 32
54907, 32
54908, 32, 43
54909, 32, 43
54910, 32, 43
54911, 32, 44
55055, 34
55056, 32
55144, 34
55145, 34
55146, 34
55147, 34
55148, 34
55149, 34
55150, 34
55151, 34
55152, 34
55153, 34
55154, 34
55155, 34
55156, 34
55157, 34
55158, 34
55159, 34
55160, 34
55161, 34
55162, 34
55163, 34
55164, 34
55165, 34
55166, 34
55167, 34
55168, 34
55169, 34
55170, 34
55171, 34
55172, 34
55173, 34
55174, 34
55175, 34
55176, 34
55177, 34
55178, 34
55179, 34
55180, 34
55181, 34
55182, 34
55183, 34
55184, 34
55185, 35
55186, 35
55187, 35

55188, 35
55189, 35
55190, 35
55191, 35
55192, 35
55193, 35
55194, 35
55195, 35
55196, 35
55197, 35
55198, 35
55199, 35
55200, 35
55201, 35
55202, 35
55203, 35
55204, 35
55205, 35
55206, 35
55207, 35
55208, 35
55209, 35
55210, 35
55211, 35
55212, 35
55213, 35
55214, 35
55215, 35
55216, 35
55217, 35
55218, 35
55219, 35
55220, 35
55221, 35
55222, 35
55223, 35
55224, 35
55225, 35
55226, 36
55227, 36
55228, 36
55229, 36
55230, 36
55231, 36
55232, 36
55233, 36
55234, 36
55235, 36
55236, 36
55237, 36
55238, 36
55239, 36
55240, 36
55241, 36
55242, 36
55243, 36
55244, 36
55245, 36
55246, 36
55247, 36
55248, 36
55249, 36
55250, 36
55251, 36
55252, 36
55253, 36
55254, 36
55255, 36
55256, 36
55257, 36
55258, 36
55259, 37
55260, 37
55261, 37
55262, 37
55263, 37
55264, 37
55265, 37
55266, 37
55267, 37
55279, 32
55290, 32
55291, 32
55293, 32

55294, 32
ASCFAST, 44
BADEXTRA, 32
BADLEN, 32
BRCOST, 32
C64MAP, 30
CALCEN, 37
CARTEN, 32
CPUFAST, 44
DIVBUSY, 37
DIVOUT, 34, 37
DMADSTMB, 43, 44
DMALADDR0, 43, 44
DMALADDR1, 43, 44
DMALADDR2, 43, 44
DMALADDR3, 43, 44
DMASRCMB, 43, 44
ENTEREXIT, 44
EXSID, 44
F4502, 44
FRAMECOUNT, 32
GEORAMBASE, 43, 44
GEORAMMASK, 43, 44
HICKED, 43, 44
HTRAP01, 30
HTRAP02, 30
HTRAP03, 30
HTRAP04, 30
HTRAP05, 30
HTRAP06, 30
HTRAP07, 30
HTRAP08, 30
HTRAP09, 30
HTRAP0A, 31
HTRAP0B, 31
HTRAP0C, 31
HTRAP0D, 31
HTRAP0E, 31
HTRAP0F, 31
HTRAP10, 31
HTRAP11, 31
HTRAP12, 31
HTRAP13, 31
HTRAP14, 31
HTRAP15, 31
HTRAP16, 31
HTRAP17, 31
HTRAP18, 31
HTRAP19, 31
HTRAP1A, 31
HTRAP1B, 31
HTRAP1C, 31
HTRAP1D, 31
HTRAP1E, 31
HTRAP1F, 31
HTRAP20, 31
HTRAP21, 31
HTRAP22, 31
HTRAP23, 31
HTRAP24, 31
HTRAP25, 31
HTRAP26, 31
HTRAP27, 31
HTRAP28, 31
HTRAP29, 31
HTRAP2A, 31
HTRAP2B, 31
HTRAP2C, 31
HTRAP2D, 31
HTRAP2E, 31
HTRAP2F, 31
HTRAP30, 31
HTRAP31, 31
HTRAP32, 31
HTRAP33, 32
HTRAP34, 32
HTRAP35, 32
HTRAP36, 32
HTRAP37, 32
HTRAP38, 32
HTRAP39, 32
HTRAP3A, 32
HTRAP3B, 32
HTRAP3C, 32

HTRAP3D, 32
HTRAP3E, 32
HTRAP3F, 32
HTRAPXX, 30, 32
HWRNGNOTRDY, 32
HWRNGRAND, 32
JMP32EN, 44
LATCHINT, 37
MAPHI0, 43, 44
MAPHI1, 43, 44
MAPHIMB, 43, 44
MAPLO0, 43, 44
MAPLO1, 43, 44
MAPLOMB, 43, 44
MATHIN0, 34
MATHIN1, 34
MATHIN2, 34
MATHIN3, 34
MATHIN4, 34, 35
MATHIN5, 35
MATHIN6, 35
MATHIN7, 35
MATHIN8, 35
MATHIN9, 35
MATHINA, 35
MATHINB, 35
MATHINC, 35
MATHIND, 35
MATHINE, 35, 36
MATHINF, 36
MATHINX, 34, 37
MATRIXEN, 44
MULBUSY, 37
MULTINA, 34, 37
MULTINB, 34, 37
MULTOUT, 34, 37
NOEXROM, 32
NOGAME, 32
OCEANA, 32
PCH, 42, 44
PCL, 42, 45
PFLAGS, 42, 45
PIRQ, 45
PNMI, 45
PORT, 30
PORT00, 43, 45
PORT01, 43, 45
PORTDDR, 30
POWEREN, 33
PREFETCH, 33
REGA, 42, 45
REGB, 42, 45
REGX, 42, 45
REGZ, 42, 45
RESERVED, 37
ROMPROT, 45
RSVD, 45
SELSDRAM, 33
SLIEN, 33
SLOWCACHE, 33
SLOWCACHEADV, 33
SLOWSDRAM, 33
SPH, 42, 45
SPL, 42, 45
UARTDATA, 43, 45
UNIT1INA, 36
UNIT1INB, 36
UNIT1OUT, 36
UNIT2INA, 36
UNIT2INB, 36
UNIT2OUT, 36
UNIT3INA, 36
UNIT3INB, 36
UNIT3OUT, 36
UNIT4INA, 36
UNIT4INB, 36
UNIT4OUT, 36
UNIT5INA, 36
UNIT5INB, 36
UNIT5OUT, 36
UNIT6INA, 36
UNIT6INB, 36
UNIT6OUT, 36
UNIT7INA, 36

UNIT7INB, 36
UNIT7OUT, 36
UNIT8INA, 36
UNIT8INB, 36
UNIT8OUT, 36
UNIT9INA, 36
UNIT9INB, 36
UNIT9OUT, 36
UNITAINA, 36
UNITAINB, 36
UNITAOUT, 36
UNITBINA, 36
UNITBINB, 36
UNITBOUT, 37
UNITCINA, 36
UNITCINB, 36
UNITCOUT, 37
UNITDINA, 36
UNITDINB, 36
UNITDOUT, 37
UNITEINA, 36
UNITEINB, 36
UNITEOUT, 37
UNITFINA, 36
UNITFINB, 36
UNITFOUT, 37
UNITXINA, 36, 37
UNITXINB, 36, 37
UNITXOUT, 36, 37
UXBSADD, 37
UXDVADD, 37
UXHIOUT, 38
UXLATCH, 38
UXLOWOUT, 38
UXMLADD, 38
VFLOP0, 45
VFLOP1, 45
VICMODE, 43, 45
VIRTPAGE0, 43, 45
VIRTPAGE1, 43, 45
VIRTPAGE2, 43, 45
VPG1LOG0, 43
VPG1LOG1, 43
VPG1PHY0, 43
VPG2LOG0, 43
VPG2LOG1, 43
VPG2PHY0, 43
VPG3LOG0, 43
VPG3LOG1, 43
VPG3PHY0, 43
VPGACTIVE, 45
VPGDIRTY, 43, 45
VPGXLOG0, 43, 45
VPGXLOG1, 43, 45
VPGXPHY0, 43, 46
VPGXPHY1, 43, 46
WATCHDOG, 43, 46
WREN, 38
RESQ, 176
RLA, 91
RMB0, 144
RMB1, 145
RMB2, 145
RMB3, 145
RMB4, 146
RMB5, 146
RMB6, 146
RMB7, 147
ROL, 92, 147
ROLQ, 28, 177
ROR, 93, 148
RORQ, 28, 178
ROW, 149
RRA, 93
RSVQ, 179
RTI, 94, 149
RTS, 94, 150

SAX, 95
SBC, 96, 150, 180
SBCQ, 28, 181
SBX, 96
SEC, 97, 151
SED, 97, 151
SEE, 152

SEI, 98, 152
SHA, 98
SHX, 98
SHY, 99
SLO, 99
SMB0, 153
SMB1, 153
SMB2, 153
SMB3, 154
SMB4, 154
SMB5, 154
SMB6, 155
SMB7, 155
SRE, 100
STA, 101, 155, 182
STQ, 28, 182
STX, 101, 156
STY, 101, 156
STZ, 157

TAB, 157
TAS, 102
TAX, 102, 158
TAY, 102, 158
TAZ, 158
TBA, 159
TRB, 159
TSB, 160
TSX, 103, 160
TSY, 161
TXA, 103, 161
TXS, 104, 162
TYA, 104, 162
TYS, 162
TZA, 163

XAA, 104