The Complete Machine Code Tutor

HANDLEIDING

ATARI

LAAD INSTRUCTIES VOOR DE ATARI COMPUTERS

Schakel de computer uit en verwijder eventuele cartridges. Plaats de cassette met de goede zijde in de recorder. Schakel, terwijl u de START toets ingedrukt houdt, de computer in en start de cassette recorder (bij XL computers dient ook de OPTION toets te worden ingedrukt). Er verschijnt een mededeling in beeld tijdens het laden, en wanneer het laden voltooid is start het programma automatisch.

Herhaal deze procedure wanneer u een volgende serie lessen wilt laden.

HET MENU SCHERM

Nadat het programma is geladen verschijnt dit scherm dat u een aantal keuze mogelijkheden biedt. Om tijdens het runnen van het programma naar dit scherm terug te keren drukt u op de SYSTEM RESET toets.Druk nooit op de SYSTEM RESET toets wanneer de computer bezig is een nieuwe serie lessen te laden.

U maakt uw keuze uit de opties van dit menu door de RETURN toets in te drukken tot uw keuze wordt aangegeven. Druk vervolgens de spatie balk in.

Nadat u een les heeft gekozen verschijnt een tekst pagina die op deze les betrekking heeft. Na het lezen hiervan toetst u de spatie balk weer in, waarna een volgende pagina of het menu verschijnt.

Wanneer u een voorbeeld programma wilt zien wordt dit automatisch geassembleerd en gerund. Door het indrukken van de RETURN toets wordt het programma regel voor regel gerund. Daarbij wordt de beschrijving van iedere regel aan de onderzijde van het scherm afgebeeld. Wanneer u de RETURN toets indrukt wordt de aangegeven instructie uitgevoerd. De vlaggen en registers worden overeenkomstig gewijzigd, waarna de volgende instructie kan worden gerund. Wanneer het runnen van het programma wordt beeindigd door een 'BRK' instructie, aan het einde van het programma of door een fout, komt de MC tutor terug in de EDIT mode. Nu kunt u het programma wijzigen, het assembleren (om het opnieuw te runnen), terugkeren naar het hoofd menu of het programma wissen om een nieuw programma in te typen. De volgende toetsen kunt u in de EDIT mode gebruiken:

Toets	Funktie
START	Om het programma te assembleren, waarna het gerund kan worden.
SELECT	Om terug te keren naar het hoofd menu voor een

volgende les.
OPTION Om het programma uit het geheugen te wissen.

In EDIT mode kunt u elk alfanumeriek karakter gebruiken bij het intypen van het programma. Daarnaast kunnen de volgende karakters worden gebruikt:

#	Om een getal aan te geven.
$	Om een hexadecimaal getal aan te geven.
()	Om adresserings modes aan te geven.
,	Idem.

Wanneer het programma in de EDIT mode komt, of wanneer u in de EDIT mode de RETURN toets indrukt komt u in het LABEL gebied. Wanneer u een regel wilt labelen kunt u dit label nu intoetsen (tot een maximum van 6 karakters) en de spatie balk indrukken. In het andere geval drukt u alleen de spatie balk in. Vervolgens kunt u een mnemonic intoetsen en weer op de spatie balk drukken (een mnemonic bestaat altijd uit 3 letters). Vervolgens voert u de operand in, bestaande uit 0 tot 7 karakters.

U kunt de cursor besturings toetsen gebruiken om het programma te editten. Door de toets met de betreffende pijl in te drukken beweegt u de cursor in die richting. op deze wijze kunt u ieder deel van het programma snel en eenvoudig wijzigen.

Wanneer u in de EDIT mode de START toets indrukt komt u in de RUN mode. Ook in deze mode hebben een aantal toetsen een speciale funktie.

Toets	Funktie
OPTION	Wisselen tussen decimale en hexadecimale weergave.
SELECT	Wisselen tussen onder en zij weergave. Automatisch wordt de zij weergave gekozen. In de onder weergave worden de adressen en de OP-codes van het programma afgebeeld.
-	Overgaan in EDIT mode.

In het menu is steeds een mogelijkheid om een volgende serie lessen te laden. Dit gebruikt u om de tweede, derde en vierde serie lessen te laden. De eerste serie wordt tezamen met het hoofdprogramma geladen. Voor dat de lessen worden geladen vraagt het programma om een bevestiging.
N.B. Druk tijdens het laden niet op SYSTEM RESET.

INHOUDSOPGAVE VAN DE CASSETTE

Kant 1

Hoofdprogramma Eerste fase van de lessen:

Instructies voor het gebruik van het programma
Les 1 : The registers (De registers).
Les 2 : Memory locations (Geheugen locaties).
Les 3 : Load and store (Laden en opslaan).

Oefening 1 voor les 3.

Les 4 : What's a flag? (Wat is een vlag?).
Les 5 : Addition (Optellen).

Oefening 1 voor les 5
Oefening 2 voor les 5

Les 6 : Substraction (Aftrekken).

Oefening 1 voor les 6
Oefening 2 voor les 6

Les 7 : Increments and decrements (Verhogen en verlagen).

Oefening 1 voor les 7

Tweede fase van de lessen
Les 8 : Transfers (Overbrengen).

Oefening 1 voor les 8

Les 9 : Binary notation (Binaire notatie).
Les 10 : Shifts and rotates (Schuiven en roteren).

Oefening 1 voor les 10

Les 11 : Two more flags (Nog twee vlaggen).

Oefening 1 voor les 11

Les 12 : Logical instructions (Logische instructies).

Oefening 1 voor les 12

Les 13 : Indexed addressing (geindexeerde adressering).

Oefening 1 voor les 13

Kant 2

Derde fase van de lessen

Les 14 : Branches and jumps (Sprongopdrachten).

Oefening 1 voor les 14
Oefening 2 voor les 14
Oefening 3 voor les 14

Les 15 : Compares (Vergelijken).

Oefening 1 voor les 15
Oefening 2 voor les 15

Les 16 Subroutines

Oefening 1 voor les 16

Les 17 : Hexadecimal notation (Hexadecimale notatie).

Oefening 1 voor les 17

Oefening 2 voor les 17

Les 18 : Binary coded decimal (Binair gecodeerd decimaal).

Oefening 1 voor les 18

Vierde fase van de lessen

Les 19 : The stack (De stack).

Oefening 1 voor les 19

Les 20 : More addressing modes (Andere adressering modes).

Oefening 1 voor les 20
Oefening 2 voor les 20

Les 21 : More on the stack (Meer over de stack).

Oefening 1 voor les 21

Les 22 : The BIT instruction (De BIT instructie).

Oefening 1 voor les 22

Les 23 : Interrupts
Les 24 : What now? (Hoe verder?).

Les 1 De registers

Deze les behandelt de registers en hun gebruik. In de 6502 processor, de belangrijkste chip van de ATARI, bevinden zich drie registers. Dit zijn resp. de acumulator, het X-register en het Y-register. De accumulator is een algemeen te gebruiken register en de meeste instrukties kunnen gebruik maken van dit register. De X en Y registers worden index registers genoemd en zijn bestemd voor meer speciale toepassingen.

Wanneer u een programma runt ziet u de inhoud van de registers links op het scherm afgebeeld. De vlaggen worden ook afgebeeld, maar daar hoeft u in deze fase niet op te letten, omdat deze later worden besproken. Een register kan een getal van 0 tot 255 bevatten.

Les 2 Geheugen locaties

Wanneer u bekend bent met BASIC weet u wat een geheugen locatie is. Wanneer u in BASIC PEEK gebruikt leest u een geheugen locatie. En iedere keer dat u POKE gebruikt schrijft u naar een geheugen locatie. De ATARI computers hebben tussen 16 en 64K RAM geheugen en tussen 10 en 24K ROM geheugen. Een geheugen locatie vertoont in die zin overeenkomst met een register dat het een getal van 0 tot 255 kan bergen.

RAM betekent Random Access Memory, wat in feite inhoudt dat RAM geheugen locaties van inhoud kunnen veranderen. Programma's die in RAM zijn opgeslagen gaan verloren wanneer de stroom wordt uitgeschakeld. Dat is de reden dat programma's moetn worden gesaved op cassette of disc wanneer ze later weer gebruikt moeten worden.

ROM staat voor Read Only Memory, wat inhoudt dat ROM geheugen locaties niet van inhoud kunnen veranderen.

Pogingen om de inhoud van ROM te veranderen hebben geen effect. Het ROM geheugen wordt voor de fabricage van de computer geprogrammeerd. Het bevat gewoonlijk machine code programma's die de computer vertellen wat er gebeuren moet na het inschakelen. In de ATARI is BASIC in ROM opgeslagen. U kunt ook ROM uitbreidings 'cartridges' kopen die achter in de computer kunnen worden gestoken. De inhoud van het ROM geheugen gaat niet verloren wanneer de computer wordt uitgeschakeld.

Wanneer u programma's runt met de simulator hebt u de beschikking over 1K geheugen. Dit geheugen gebied loopt van adres 0 tot 1023 (in RAM), en u kunt dit gebruiken zoals u wilt. Het programma dat u met de simulator maakt wordt op een andere plaats in het geheugen opgeslagen en kan niet beschadigd worden.

Les 3 Laden en opslaan

Laad en opslag instructies beinvloeden geheugen locaties en registers. Een laad opdracht verandert de inhoud van een register. Een getal tussen 0 en 1023 wordt aangegeven achter de laad instructie, het adres van de geheugen locatie waarvan de inhoud in het register wordt geplaatst. Ook kunt u een getal (aangegeven met een £) opgeven van 0 tot en met 255 dat in het register wordt geplaatst. De volgende laad instructies zijn beschikbaar op de ATARI:

LDA : Laad accumulator.
LDX : Laad X-register.
LDY : Laad Y-register.

Voorbeelden van het gebruik van laad instructies:

LDA 1000 : Laad accumulator met inhoud van locatie 1000.
LDT #34 : Laad het Y-register met het getal 34.

Een opslag instructie wordt gebruikt om de inhoud van een geheugen locatie te veranderen. De inhoud van ieder van de drie registers kan worden opgeslagen in het geheugen. Het getal achter de opslag instructie geeft de geheugen locatie aan. Voorbeelden van opslag instructies:

STX 1 : Sla de inhoud van het X-register op in geheugen locatie 1.
STA 1019 : Sla de inhoud van de accumulator op in geheugen locatie 1019.

Oefening 1 voor les 3

Dit is een eenvoudig programma om het gebruik van laad en opslag instructies te tonen. Druk de RETURN toets in om een regel te runnen. DE 'BRK' instructie dient om het runnen van het programma te stoppen en terug te gaan in de EDIT mode. In de EDIT mode kunt u door START in te drukken het programma opnieuw runnen. Door indrukken van SELECT keert u terug naar het hoofd memu zodat u een andere les of oefening kunt kiezen. Met behulp van de cursor toetsen kunt u het programma wijzigen.(u hoeft hierbij niet de CONTROL toets te gebruiken). Door intoetsen van OPTION wist u het huidige programma, zodat u een nieuw programma kunt intoetsen.

Les 4 Wat is een vlag?

Deze les behandeld vlaggen in het algemeen en de nul vlag in het bijzonder. een vlag kent twee toestanden. Hij kan ge"set" zijn, wat inhoudt dat hij de waarde 1 heeft, of hij kan ge"reset" zijn, wat betekent dat hij

de waarde 0 heeft. Een vlag is dus als een schakelaar, waarbij "aan" overeenkomt met 1 en "uit" met 0.

les 5 Optellen

Deze les behandelt eenvoudige optellingen in machine code. De instructie ADC (Add with carry) wordt gebruikt voor optellen in MC. De 6502 heeft geen instructie om op te tellen zonder carry. Daarom is het nodig voor een optelling de carry vlag te "reset"-ten. Hiervoor wordt de instructie CLC (Clear carry) gebruikt. De ADC instructie telt de waarde van carry op bij het resultaat van de optelling, dus zal de uitkomst 1 hoger zijn wanneer de carry vlag ge-"set" is.

De carry vlag wordt overeenkomstig het resultaat van de optelling ingesteld. Wanneer het resultaat groter is dan 255 wordt de carry vlag ge-"set". Wanneer dus de carry vlag ge-"set" is na een optelling moet 256 bij het resultaat worden opgeteld om het juiste antwoord te vinden. Wanneer het resultaat bij voorbeeld 10 is met de carry vlag ge-"set" moet het juiste antwoord 266 zijn.

Oefening 1 bij les 5

Dit is een voorbeeld programma om het gebruik van ADC en CLC te tonen. Merk op dat de ADC instructie wordt gevolgd door een geheugen locatie of een getal, aangegeven door **#**.

Oefening 2 bij les 5

Dit korte programma introduceert de SEC (Set carry vlag) instructie. Dit omdat de SEC instructie de carry vlag 1 maakt, die bij de ADC instructie wordt opgeteld.

Les 6 Aftrekken

Bij de 6502 gaat aftrekken op eenzelfde wijze als optellen, waarbij de carry vlag echter andersom werkt. De instructie hiervoor is SBC (Substract with carry) Bij aftrekken wordt de carry vlag gebruikt om een waarde te "lenen". Daarom moet de carry vlag in plaats van ge-"reset" ge-"set" wordenmet de SEC instructie. Als na het aftrekken de carry vlag nul is betekent dit da het resultaat te klein is, en dat 256 afgetrokken moet worden om het juiste antwoord te vinden. Als voor het aftrekken de carry vlag ge-"reset" is zal het resultaat 1 minder zijn dan wanneer de vlag ge-"set" is.

Oefening 1 voor les 6

Dit voorbeeld programma toont het effect van de carry vlag voor en na de SBC instructie. Let op het resultaat van het aftekken van 0 van 3 wanneer de carry vlag nul is.

Oefening 2 bij les 6

Dit is een kort programma dat toont wat gebeurt wanneer u een aftrekking doet met een negatief resultaat. Let daarbij op de carry vlag.

Les 7 Verhogen en verlagen

Deze instructies zijn eenvoudig te begrijpen en toe

te passen. Het verhogen dient om de inhoud van een register of een geheugen locatie met 1 te vermeerderen. A ls een register 255 bevat voor de verhoging zal het daarna 0 bevatten. Een verlaging dient om de inhoud van een register of een geheugen locatie met 1 te verminderen. Als het register of de geheugen locatie 0 bevat voor de verlaging dan zal die daarna 255 bevatten. De volgende verhoging en verlaging instructies zijn beschikbaar:

DEX : Verlaag X-register
DEY : Verlaag Y-register.
INX : Verhoog X-register.
INY : Verhoog Y-register.
DEC : Verlaag geheugen locatie.
INC : Verhoog geheugen locatie.

Oefening 1 bij les 7

Dit programma gebruikt alle eerder genoemde instructies

Les 8 Transfer (Overbreng) instructies

Een transfer is het overbrengen van een waarde van een register naar een ander.De mnemonics van alle overbreng instructies beginnen met een T. De volgende letter geeft de herkomst aan en de derde de bestemming. De volgende instructies zijn beschikbaar:

TAX : Breng accumulator over naar X.
TXA : Breng X register over naar accumulator.
TAY : Breng accumulator over naar Y.
TYA : Breng Y register over naar accumulator.

Twee andere speciale transfer instructies zullen later worden besproken.

Oefening 1 bij les 8

Dit programma toont de werking van transfer instructies. Laat u niet misleiden door de eerste TXA instructie. Deze brengt het X register over naar de accumulator, dus de accumulator bevat de waarde 0 na de instructie.

Les 9 Binaire notatie

Deze les behandeld een nieuw talstelsel, het binaire stelsel. in dit systeem woden maar twee cijfers gebruikt, nul en een. Een vlag is een goed voorbeeld van binaire notatie, omdat een vlag alleen de waarde een of nul kan hebben. Na het doornemen van de les over binaire getallen kan de volgende tabel van nut zijn:

Waarden van bit 7 tot bit 0:

Bit nummer	7	6	5	4	3	2	1	0
Waarde	128	64	32	16	8	4	2	1

Les 10 Shifts and Rotates (Schuiven en roteren)

Deze instructie houdt direct verband met het veranderen van binaire cijfers. Ale schuif en roteer instructies worden gevolgd door een A als de instructie

op deaccumulator betrekking heeft, of door een getal als de instructie een geheugen locatie betreft. De volgende lijst geeft een overzicht van de schuif en roteer instructies die beschikbaar zijn op de 6502:

ROR Rotate right

Dit is een 9 bit rotatie Alle bits in de accumulator of de geheugen locatie worden 1 bit naar rechts geroteerd. De carry wordt in bit 7 geplaatst en bit 0 in de carry.

ROL Rotate left

Dit is ook een 9 bit rotatie. Alle bits worden 1 plaats naar links geroteerd, decarry wordt in bit 0 geplaatst en bit 7 in de carry.

LSR Logical shift right

Hierbij wordt een nul in bit 7 geplaatst en schuiven alle andere bits naar rechts. Bit nul wordt in de carry geplaatst. Deze instructie heeft het effect van een deling door 2. De carry geeft aan of er een "rest" van een halve bit is overgebleven.

ASL Arithmetic shift left

Er wordt een nul in bit 0 geplaatst en alle bits schuiven een plaats naar links. De inhoud van bit 7 wordt in de carry geplaatst. Deze instructie heeft het effect van een vermenigvuldiging met 2. De carry geeft aan dat het resultaat te groot was en 256 bij het antwoord meot worden opgeteld.

Oefening 1 bij les 10

Dit is een programma dat alle schuif en roteer instructies van de 6502 gebruikt. Let vooral op het effect op de carry van alle instructies.

Les 11 Two more flags (Nog twee vlaggen)

Deze les behandelt de overflow vlag en de negatief vlag. Vaak kan de overflow vlag na een reken operatie veranderd zijn. De vlag is i als na een instructie het teken verkeerd is, dat wil zeggen een carry van bit 6 naar bit 7. De instructie CLV (clear overflow) wordt gebruikt om de overflow vlag te "reset"-ten. De negatief vlag geeft na een instructie de toestand van bit 7 , de teken bit, weer. Wanneer na een instructie bit 7 een is, is ook de negatief vlag een, en andersom.

Oefening 1 bij les 11

Dit programma demonstreert in welke gevallen de overflow en negatief vlaggen ge-"set" en ge-"reset" worden.

Les 12 Logical instructions (logische instructies)

Deze instructies beinvloeden ook direct binaire getallen. Hier wordt een waarheids tabel gegeven voor alle logische instructies. Bij een logische instructie worden alle bits van de accumulator vergeleken met de bits van een geheugen locatie of een getal, en het resultaat van de vergelijking wordt in de accumulator geplaatst.

ORA Logical or accumulator (logische or op de accumulator)

```
Input 1   0 0 1 1
Input 2   0 1 0 1
          -------
Resultaat 0 1 1 1
```


Een of beide bits 1 geeft als resultaat 1, anders 0.

AND Logical and accumulator (logische and op de accumulator)

```
Input 1   0 0 1 1
Input 2   0 1 0 1
          -------
Resultaat 0 0 0 1
```

Als beide bits 1 zijn is het resultaat 1, anders 0.

EOR Exclusive or accumulator

```
Input 1   0 0 1 1
Input 2   0 1 0 1
          -------
Resultaat 0 1 1 0
```

Als beide bits 1 of 0 zijn is het resultaat 0, anders is het resultaat 1.

Oefening 1 bij les 12

Dit voorbeeld programma gebruikt deze drie logische instructies.

Les 13 Indexed addressing (geindexeerde adressering)

Bij deze adressering wordt de inhoud van het X of Y register opgeteld bij het opgegeven adres om het te

gebruiken adres te vinden. Voorbeelden van geindexeerde adressering:

LDA 1000,X Tel de inhoud van X op bij 1000 en laad de accumulator met de inhoud van dit adres.
EOR 300,Y Tel de inhoud van Y op bij 300. Pas vervolgens een exclusive OR toe op de accumulator en dit adres.

Les 14 Branches and jumps (Sprongopdrachten)

Deze instructies worden gebruikt om de volgorde waarin de instructies van het programma worden uitgevoerd te veranderen. Normaal worden de instructies in volgode van het laagste adres naar het hoogste uitgevoerd. Een sprong instructie is equivalent aan het GOTO statement in BASIC. Een voorwaardelijke sprongopdracht komt overeen met IF...THEN GOTO. Wanneer u met de simulator programma's schrijft moet u voor een sprong een label opgeven. Om een label in te voeren drukt u niet op de spatie balk voor u de instructie intypt. In plaats daarvan typt u een label van ten hoogste 6 karakters in, gevolgd door indrukken van de spatie balk. Wanneer u een sprong naar een label wilt uitvoeren typt u de naam van de label in na de instructie. De volgende sprong instructies zijn beschikbaar:

JMP	Sprong naar een label
BEQ	Sprong indien nul (nul vlag ge-"set")
BNE	Sprong indien niet nul
BCS	Sprong indien carry ge-"set"
BCC	Sprong indien carry niet ge-"set"
BVS	Sprong indien overflow (overflow vlag ge-"set")
BVC	Sprong indien geen overflow

Oefening 1 bij les 14

Dit programma gebruikt het X register om van 10 af te tellen tot 0. BEQ wordt gebruikt om de waarde nul te detecteren.

Oefening 2 bij les 14

Dit programma deelt 58 door 5. Na afloop van de run

bevat het X register het antwoord en het Y register de rest.

Oefening 3 bij les 14

Dit programma vermenigvuldigt de inhoud van X met de inhoud van Y. U kunt het programa wijzigen om andere getallen met elkaar te vermenigvuldigen, wanneer de uitkomst maar niet groter is dan 255.

Les 15 Compares (Vergelijkingen)

Een vergelijking is een aftrekking die de accumulator niet verandert. Hij beinvloed de vlaggen op dezelfde wijze als een aftrekking. Een vergelijk instructie ka gebruikt worden om het X of het Y register te vergelijken, zowel als de accumulator. Het is echter niet nodig de carry vlag 1 te maken, omdat we niet te maken hebben met een echte aftrekking. Vergelijk instructies in combinatie met sprongopdrachten bieden de programmeur vele mogelijkheden. De volgende vergelijk instructies zijn beschikbaar:

CPM Vergelijk accumulator met een getal of een geheugen locatie.
CPX Vergelijk X register met een getal of een geheugen locatie.
CPY Vergelijk Y register met een getal of een geheugen locatie.

Oefeningen 1 en 2 bij les 15

Deze voorbeeld programma's tonen de werking van vergelijk instructies. Let op het veranderen van de vlaggen en verander het programma om te zien hoe het werkt.

Les 16 Subroutines

Een subroutine is als een GOSUB commando in BASIC. Om een subroutine aan te roepen gebruikt u JSR (Jump to subroutine). De instructie wordt gevolgd door een label, zoals bij JMP. Deze instructie werkt exact gelijk als JMP, echter wanneer na een JSR instructie een RTS (return from subroutine) wordt uitgevoerd springt de computer terug naar de instructie na de JSR instructie. De RTS instructie komt dus overeen met RETURN in BASIC.

Oefening 1 bij les 16

Dit programma gebruikt subroutines om 5 met 8 te vermenigvuldigen.

Les 17 Hexadecimal

Dit is een tal stelsel zoals het decimale stelsel en het binaire stelsel. De volgende tabel kan van nut zijn:

Dec	Bin	Hex	Dec	Bin	Hex
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F

Oefening 1 bij les 17

Dit programma gebruikt hexadecimale getallen, voorafgegaan door een $ karakter. Steeds als u dit karakter tegen komt betekent het dat het gegeven getal in HEX is. U kunt het in uw eigen programma's gebruiken om hexadecimale getallen aan te geven. Tijdens het runnen van het programma kunt u wisselen tussen decimale en hexadecimale notatie door OPT in te drukken. Druk tijdens het runnen van dit programma op toets OPT om dit

te proberen.

Les 18 Binary coded decimal (binair gecodeerd decimaal, BCD)

In dit systeem wordt een byte opgedeeld in 2 delen die ieder een decimaal cijfer aangeven. Om over te gaan op BCS wordt de instructie SED (Set decimal mode flag) gebruikt. Om terug te gaan dient de instructie CLD (clear decimal mode flag). Om met een getal in BCD te werken wordt het omgezet in HEX. Daarom is het noodakelijk om HEX mode in te schakelen wanneer in BCD wordt gewerkt. Denk eraan dat hoewel BCD hexadecimale getallen gebruikt, de cijfers A - F niet worden gebruikt.

Oefening 1 bij les 18

Dit programma demonstreert BCD. Kies zodra het programma wordt gerund de HEX mode, anders zal de uitkomst geen betekenins hebben. De ADC en SBC instructies kunnen in BCD gebruikt worden, de carry vlag wordt ge-"set" als de uitkomst groter dan 99 is en ge-"reset" als de uitkomst kleiner dan 00 is.

Les 19 The stack

De stack is 1 pagina van het geheugen, page 1. De stack gebruikt dus 256 bytes (HEX 100) van het geheugen, vanaf locatie 256 tot 512 (HEX: 100 tot 200). De stack pointer staat oorspronkelijk op 255 (HEX: FF). Hier wordt 100 HEX opgeteld, zodat de eerste invoer van de stack gebeurt op 1FF. Volgende invoer verlaagt de stack pointer, het weghalen van invoer verhoogt de pointer. De volgende instructies hebben invloed op de stack:

PHA	Breng accumulator naar de stack
PLA	Breng inhoud stack over naar accumulator
TXS	Breng de inhoud van het X register over naar

de stack
TSX Breng de inhoud van de stack over naar het X register

Oefening bij les 19

Dit programma toont het gebruik van de hier boven vermelde instructies. Denk eraan dat ook de instructies JSR en RTS de stack beinvloeden.

Les 20 More addressing modes (andere adresserings modes)

De wijze van adressering bepaald waar de 6502 zijn gegevens vandaan zal halen. De verschillende modes worden uitgebreid besproken in de les. Hier volgt een lijst van alle instructies die in de verchillende modes kunnen worden gebruikt:

Implied:

BRK	CLC	CLD	CLI	CLV	DEX	DEY	INK
INY	NOP	PHA	PHP	PLA	PLP	RTS	SEC
SED	SEI	TAX	TAY	TSX	TXA	TXS	TYA

Merk op dat NOP staat voor No Operation en niets doet. De instructie RTI (return from interrupt) wordt hier niet vermeld, omdat deze niet door de simulator wordt verwerkt.

Accumulator addressing

Deze adressering wordt aangegeven door de letter A achter de instructie, en geeft aan dat de instructie op de accumulator moet worden uitgevoord. De volgende instructies zijn beschikbaar:

ASL A LSR A ROL A ROR A

Absolute addressing

De volgende instructies kunnen worden uitgevoerd op een absoluut geheugen adres:

ADC	AND	ASL	BIT	CMP	CPX	CPY	DEC
EOR	INC	LDA	LDX	LDY	LSR	ORA	ROL
ROR	SBC	STA	STX	STY			

Denk er aan dat de instructies JMP en JSR absolute instructies zijn, die met de simulator echter alleen met labels kunnen worden gebruikt.

Zero page addressing

De volgende instructies kunnen met zero page addressing worden gebruikt:

ADC	AND	ASL	BIT	CMP	CPX	CPY	DEC
EOR	INC	LDA	LDX	LDY	LSR	ORA	ROL
ROR	SBC	STA	STX	STY			

Immediate addressing

De immediate adresserings wijze wordt aangegeven door een (#) voor een getal. De instructies die bij deze wijze van adresseren kunnen worden gebruikt zijn:

ADC	AND	CMP	CPX	CPY	EOR	LDA	LDX
LDY	ORA	SBC					

Absolute,Y addressing

Het Y register wordt opgeteld bij het adres wat een nieuw adres oplevert. Dit nieuwe adres wordt gebruikt voor de instructie. De volgende instructies kunnen worden gebruikt:

ADC	AND	CMP	EOR	LDA	LDX	ORA	SBC	STA

Absolute,X addressing

Dit komt overeen met absolute,Y echter het X register wordt gebruikt in plaats van het Y register. De volgende instructies zijn beschikbaar:

ADC AND ASL CMP DEC EOR INC LDA
LDY LSR ORA ROL ROR SBC STA

(Indirect,X) addressing

Bij deze wijze van adresseren wordt de inhoud van het X register opgeteld bij de inhoud van een zero page adres, wat een ander zero page adres oplevert. De inhoud van deze twee byte zero page pointer wijst naar een ander adres, waarop de instructie wordt uitgevoerd. De volgende instructies zijn beschikbaar:

ADC AND CMP EOR LDA ORA SBC STA

(Indirect),Y addressing

Bij deze mode wordt een zero page adres opgegeven. Dit adres is een twee byte pointer naar een ander adres. De inhoud van het Y register wordt vervolgens opgeteld bij dit adres, wat een ander adres oplevert waarop de instructie wordt uitgevoerd. De volgende instructies kunnen worden gebruikt:

ADC AND CMP EOR LDA ORA SBC STA

Zero page,X addressing

Deze adressering komt overeen met absolute,X adressering, echter wordt een zero page adres opgegeven in plaats van een absoluut adres. Deze adressering gebruikt daarom maar twee bytes per instructie in plaats

van drie. De volgende instructies kunnen worden gebruikt:

ADC AND ASL CMP DEC EOR INC LDA
LDY LSR ORA ROL ROR SBC STA STY

Zero page,Y addressing

Deze adressering komt overeen met zero page,X, echter het Y register wordt gebruikt in plaats van het X register. De volgende instructies kunnen worden gebruikt:

LDX STX

Relative addressing

In de simulator moet deze adressering worden gebruikt met een label. Een relatieve sprong wordt alleen dan gemaakt als de voorwaarde in de instructie waar is. In dat geval wordt een sprong byte bij de programma teller opgeteld. De sprong byte wordt opgeslagen als "two's complement". Daarom kan de programma teller met niet meer dat +127 of -128 worden veranderd. Let er daarom op als u relatieve sprongen gebruikt dat de sprong byte niet te groot is. Een relatieve sproninstructie gebruikt maar twee bytes, terwijl een absolute sprong er drie nodig heeft. De volgende instructies kunnen worden gebruikt:

BCC BCS BEQ BMI BNE BPL BVC BVS

(Indirect) addressing

Deze adressering kan niet met de simulator worden gebruikt. Een gewone 6502 assembler kan deze adressering echter wel verwerken. De inhoud van een adres levert een nieuw adres op, dat voor de instructie wordt gebruikt. De enige instructie die deze adressering gebruikt is JMP

(xxxx).

Oefening 1 en 2 bij les 20

Deze voorbeeld programma's gebruiken de meest gecompliceerde wijzen van adressering. Bestudeer deze programma's zorgvuldig omdat ze u veel over programmeren kunnen leren.

Les 21 More on the stack (meer over de stack)

Deze les bespreekt hoe vlaggen in de 6502 processor worden opgeslagen in het Processor Status Register (PSR). De vlaggen worden op de volgende wijze opgeslagen:

```
Bit : 7  6  5  4  3  2  1  0
Vlag: N  V  -  B  D  I  Z  C
```

N : Negatief vlag
V : Overflow vlag
\- : Niet gebruikt
B : Break vlag (wordt niet gebruikt door de simulator)
I : "Interrupt geblokkeerd" vlag
Z : Nul vlag
C : Carry vlag

Slechts twee instructies beinvloeden direct het PSR. Deze saven en herstellen de waarden van de verschillende vlaggen door het PSR op de stack op te slaan en weer terug te halen. De instructie PHP (Push PSR onto stack) slaat alle vlaggen op, de instructie PLP (Pull PSR from stack) brengt de vlaggen weer terug in het PSR.

Oefening 1 bij les 21

Dit programma demonstreert wat er gedaan kan worden met de instructies PHP en PLP.

Les 22 The BIT instruction

Deze instructie is in feite een AND instructie die de accumulator intact laat. De BIT instructie wordt gevolgd door een geheugen adres, wat in zero page of absolute addressing kan worden opgegeven. De inhoud van de accumulator wordt ge-AND met de geheugen locatie en de vlaggen worden overeenkomstig ingesteld. Bit 7 van de geheugen locatie wordt in de negatief vlag en bit 6 in de overflow vlag geplaatst. De nul vlag wordt ge-"set" als geen enkele bit in de geheugen locatie overeenkomt met de bits van de accumulator.

Oefening 1 bij les 22

Dit programma toont de werking van de BIT instructie.

Les 23 Interrupts

In deze les wordt een poging gedaan een zeer gecompliceerd onderwerp op heldere wijze te bespreken: Interrupts. Met de simulator kunnen geen interrupt routines geschreven worden. De instructies SEI (Set interrupt disable vlag) en CLI (Clear interrupt disable vlag) kunnen met de simulator worden gebruikt om het effect ervan op de vlaggen te tonen.

Wat nu?

Hiermee bent u aan het einde gekomen van de cursus 6502 machine code programmeren. U kunt nu uw eigen MC programma's schrijven met behulp van een normale 6502 assembler. Deze assemblers zijn minder gebruikers vriendelijk, maar bieden meer mogelijkheden. Een fout in uw programma kan er echter voor zorgen dat uw computer "crasht" of vast loopt. Daarom is het aan te raden ieder machine code programma te saven voor het wordt gerund.

Denk erom dat een normaal MC programma bliksemsnel wordt uitgevoerd, en niet op een toetsindruk wacht om de volgende instructie uit te voeren, zoals de simulator!

Als u niet zeker weet welk effect een instructie zal hebben kunt u deze altijd uitproberen met de simulator. Typ het programma in, controleer de werking en kijk of de vlaggen worden veranderd als bedoeld. Wanneer u tevreden bent over de werking kunt u het programma in een echte assembler intoetsen.

VEEL SUCCES!

APPENDIX A: TE GEBRUIKEN INSTRUCTIES

De instructies die de simulator kan verwerken zijn:

Instructie	Functie
ADC	Add with carry (optellen met carry)
AND	Logical AND accumulator (logische AND)
ASL	Arithemetic shift left (schuiven naar links)
BCC	Branch on carry clear (sprong als carry=0)
BCS	Branch on carry set (sprong als carry=1)
BEQ	Branch on zero (sprong als 0)
BIT	Test memory bits against accumulator (test bits van geheugen tegen accumulator)
BMI	Branch on minus (sprongindien neg.)
BNE	Branch on not equal zero (sprong indien niet 0)
BPL	Branch on positive (sprong indien pos.)
BRK	Break (stop programma)
BVC	Branch on overflow clear (sprong indien geen overflow)
BVS	Branch on overflow set (sprong indien overflow)
CLC	Clear carry flag (stelt carry vlag op 0)
CLD	Clear decimal mode flag (beeindig decimale mode)
CLI	Clear interrupt disable flag (stelt interrupt disable vlag op 0)
CLV	Clear overflow flag (oerflow vlag = 0)
CMP	Compare accumulator (vergelijk met accum.)

CPX	Compare X register (idem X register)
CPY	Compare Y register (idem Y register)
DEC	Decrement memory location (verminder geheugen locatie)
DEX	Decrement X register (verminder X register)
DEY	Decrement Y register (verminder Y register)
EOR	Exclusive OR accumulator (Exclusief OR operatie op accumulator)
INC	Increment memory location (verhoog geheugen locatie)
INX	Increment X register (verhoog X register)
INY	Increment Y register (verhoog y register)
JMP	Unconditional jump (onvoorwaardelijke sprong)
JSR	Jump to subroutine (sprong naar subrout.)
LDA	Load accumulator (laad accumulator)
LDX	Load X register (laad X register)
LDY	Load Y register (laad Y register)
LSR	Logical shift right (schuif naar rechts)
NOP	No operation (geen functie)
ORA	Logical OR accumulator (OR operatie op accum.)
PHA	Push accumulator onto stack (breng accum. inhoud naar stack)
PHP	Push processor status register onto stack (breng PSR over naar de stack)
PLP	Pull PSR from stack (haal PSR van de stack)
ROL	Rotate left (roteer naar links)
ROR	Rotate right (roteer naar rechts)
RTI	Return from subroutine (terugkeren van sub.)

SBC Substract with carry (aftrekken met carry)

SEC Set carry flag (Maak carry vlag 1)
SED Set decimal mode flag (overgaan op decimaal)

SEI Set interrupt disable flag
(interrupts niet toegestaan)
STA Store accumulator (sla accumulator inhoud op)

STX Store X register (idem x register)
STY Store Y register (idem Y register)
TAX Transfer accumulator to X register
(breng accumulator over naar X register)

TAY Transfer accumulator to Y register
(breng accumulator over naar Y register)
TSX Transfer stack pointer to X register
(breng stack pointer over naar X reg.)
TXA Transfer X register to accumulator
(breng X register over naar accum.)
TXS Transfer X register to stack pointer
(breng X register over naar stack pointer)

TYA Transfer y register to accumulator
(breng Y register over naar accum.)

APPENDIX B: SAMENVATTING FOUTMELDINGEN

Soms verschijnt een foutmelding aan de onderzijde van het beeld. Gewoonlijk zullen fouten aan het licht treden bij het assembleren of runnen van een programma. Wanneer een foutmelding verschijnt kunt u RETURN indrukken om terug te keren naar de EDIT mode om de fout te herstellen.

Duplicate label error: Deze melding geeft aan dat een label meer dan eens voorkomt. Wanneer u bij voorbeeld twee maal het label 'FRED' heeft gebruikt weet het programma niet welk label u bedoelt. Verander de label namen, zodat iedere naam slechts 1 maal voorkomt.

Label not found error: Dit betekent dat bij een instructie waarin een label werd gebruikt geen overeenkomstige label werd gevonden. Deze fout zou bij voorbeeld optreden bij de instructie JMP JOHN wanneer de computer het label JOHN nergens kan vinden.

Number bigger than 255 error: tijdens het assembleren werd een getal gevonden groter dan 255, terwijl de computer een getal kleiner of gelijk aan 255 verwachtte.De instructie LDA £400 zou deze foutmelding opleveren.

Resulting memory address greater than 1024 error: Deze fout kan alleen tijdens het runnen optreden. Het betekent dat een geheugen locatie groter dan 1024 werd gebruikt. Wijzig het programma zodat het alleen de adressen 0 tot 1024 gebruikt.

Error 7 (check manual): deze foutmelding mag nooit optreden. Wanneer deze fout optreedt betekent dat dat het programma niet goed geladen is. Spoel de tape terug en laadt opnieuw.

Instruction not recognized error: Deze melding verschijnt wanneer tijdens het assembleren een instructie niet wordt begrepen, gewoonlijk veroorzaakt door een fout in de syntaxis van de instructie. Er behoort een spatie te zijn tussen de mnemonic en een label en tussen de mnemonic en de operand (indien aanwezig). In een label, mnemonic of operand behoren geen spaties te zijn. Kontroleer ook of u het juiste aantal komma's, haakjes e.d. heeft gebruikt.

Assembly language: Taal die mnemonics gebruikt als weergave van MC instructies. Een assembly language programma kan niet worden gerund wanneer het niet is geassembleerd.

Binary: Twee. In het binaire stelsel worden de cijfers 0 en 1 gebruikt om getallen weer te geven.

Binary coded decimal (BCD): Een systeem waarin een "nybble" een decimaal cijfer weergeeft. Een byte kan daarom twee cijfers weergeven.

Bit: Een binaire eenheid, is ofwel 1 of 0.

Bug: Een fout of een ongewenste eigenschap van een programma, die maakt dat het programma niet of niet goed werkt.

Character: Een element van een verzameling van symbolen, zoals een letter, cijfer of teken.

Chip: Gebruikelijke benaming voor geintegreerde schakeling, afgeleid van het silicium schijfje waarop de schakeling is aangebracht.

Computer: Een machine die gegevens verwerkt volgens bepaalde instructies en de resultaten van de verwerking naar buiten brengt. Benaming voor het geheel van processor en I/O apparatuur.

Crash: Term die aanduidt dat de computer geen invoer van het toetsenbord meer accepteert. Oplossing is het uit en aan schakelen van de computer.

Cursor: Een Teken dat op het TV scherm aangeeft waar

gegevens worden ingevoerd.

DATA: Een gegeven dat de computer kan verwerken.

Editing: Het proces van het wijzigen van gegevens voor men ze door de processor laat verwerken.

Execute: Het uitvoeren van de instructies van een programma. Een micro processor voert een programma uit door het lezen en vervolgens verwerken van de instructies.

Graphics: Term die aanduidt dat gegevens in de vorm van beelden worden gepresenteerd. Beelden op het TV scherm worden afgebeeld in "pixels".

Hardware: De vaste delen van de computer.

Hexadecimal: Getal stelsel gebaseerd op 16 verschillende tekens als cijfers. Gewoonlijk worden de tekens 0-9 en A-F gebruikt.

Instruction: Een bepaalde opdracht aan de processor. Een MC programma bestaat uit instructies.

Machine Code (MC); Binaire weergave van de instructies aan de micro processor. Machine code kan door de micro processor worden verwerkt zonder verdere bewerking.

Memory: Verzameling van geintegreerde schakelingen waarin gegevens worden opgeslagen. Iedere bit wordt opgeslagen als een elektrisch signaal in de IC. Geheugens worden onderverdeeld in RAM en ROM, hun grootte wordt aangegeven in K (kilobytes).

Micro processor: Een geintegreerde schakeling die alle componenten bevat voor het verwerken van gegevens. Een micro processor moet verbonden zijn met I/O

apparatuur en geheugen om te kunnen werken.

Mnemonic: een groep van 3 of 4 karakters die een MC instructie voorstellen. Iedere mnemonic wordt omgezet in een MC instructie door een assembler.

Nybble: Een groep van 4 bits. Een byte bestaat uit twee nybbles.

Object program: Een programma in MC. Een "source program", dat niet door de processor kan worden uitgevoerd, wordt geassembleerd door een assembler, die het "object program" genereert. Dit "object program" bevindt zich in het geheugen en kan worden uitgevoerd door de processor.

Operating system: Een machine code programma, onderdeel van de systeem software, dat de processor in staat stelt te functioneren.

Page: Gebruikt in verband met geheugen: 265 bytes.

Program: Een verzameling opdrachten die de micro processor een bepaalde taak laten uitvoeren.

RAM: Random Access Memory. Naar dit geheugen kan worden geschreven, zowel als van gelezen. In dit geheugen wordt het programma opgeslagen. Bij uitschakelen van de computer gaan alle in RAM opgeslagen gegevens verloren.

ROM: Read Only Memory. Dit geheugen wordt in de fabriek waar de computer gefabriceerd wordt geprogrammeerd. Het bevat gewoonlijk het operating system en andere programma's die de computer nodig heeft bij het inschakelen. Het in- en uitschakelen van de computer heeft geen effect op het ROM geheugen.

Software: De computer programmatuur.

Source program: Programma bestaande uit mnemonics die voor mensen begrijpelijk zijn. Dit programma moet worden geassembleerd voor het kan worden uitgevoerd.