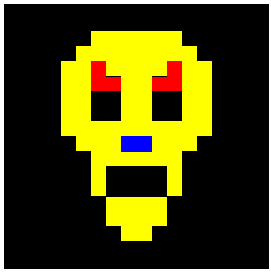


APPENDIX B

Registers

This section gives a short description of all the 6502's registers:

Accumulator - A

8-bit general-purpose register, which forms one operand in all the arithmetic and logical instructions.

Index Register - X

8-bit register used as the offset in indexed and pre-indexed indirect addressing, or as a counter.

Index Register - Y

8-bit register used as the offset in indexed and post-indexed indirect addressing modes.

Status Register - P

8-bit register containing the following:

Bit 0 - Carry flag (C). Set if a carry occurs during an add operation; cleared if a borrow occurs during a subtract operation; used as a ninth bit in the shift and rotate instructions.

Bit 1 - Zero flag (Z). Set if the result of an operation is zero; cleared otherwise.

Bit 2 - Interrupt disable (I). If set, disables the effect of the IRQ interrupt. Is set by the processor during interrupts.

Bit 3 - Decimal mode flag (D). If set, the add and subtract operations work in binary-coded-decimal arithmetic; if clear, the add the subtract operations work in binary arithmetic.

Bit 4 - Break command (B). Set by the processor during a BRK interrupt; otherwise cleared.

Bit 5 - Unused.

Bit 6 - Overflow flag (V). Set if a carry occurred from bit 6 during an add operation; cleared if a borrow occurred to bit 6 in a subtract operation.

Bit 7 - Negative flag (N). Set if bit 7 of the result of an operation is set; otherwise cleared.

Stack Pointer - S

8-bit register which forms the lower byte of the address of the next free stack location; the upper byte of this address is always &01.

Program Counter - PC

16-bit register which always contains the address of the next instruction to be fetched by the processor.

Assembler mnemonics

The following section lists all the instruction mnemonics in alphabetical order. Each instruction is accompanied by a description of the instruction,

a symbolic representation of the action performed by the instruction, a diagram showing the status-register flags affected by the instruction, and a list of the permitted addressing modes for the instruction.

The following symbols are used in this section:

Symbol:	Definition:
+	Addition
-	Subtraction
&	Logical AND
|	Logical OR
:	Logical Exclusive-OR
!	Push onto hardware stack
~	Pull from hardware stack
=	Assignment
M	Memory location
(PC+1)	Contents of location after op-code
#	Immediate addressing mode
~	No change to flag
%	Change to flag
1	Set
0	Cleared
A	Accumulator
X	X Index Register
Y	Y Index Register
PC	Program Counter
PCL	Low byte of Program Counter
PCH	High byte of Program Counter

ADC Add memory to accumulator with carry ADC

A,C=A+M+C

			N	Z	C	I	D	V
			%	%	%	~	~	%

Addressing	Assembler	Format	Bytes	Cycles
Immediate	ADC #	Oper	2	2
Zero Page	ADC	Oper	2	3
Zero Page,X	ADC	Oper,X	2	4
Absolute	ADC	Oper	3	4
Absolute,X	ADC	Oper,X	3	4*
Absolute,Y	ADC	Oper,Y	3	4*
(Indirect,X)	ADC	(Oper,X)	2	6
(Indirect),Y	ADC	(Oper),Y	2	5*

* Add 1 if page boundary crossed.

AND AND memory with accumulator AND

A=A&M

```
                                        N Z C I D V
                                        % % ~ ~ ~ ~
```

Addressing	Assembler	Format	Bytes	Cycles
Immediate	AND #	Oper	2	2
Zero Page	AND	Oper	2	3
Zero Page,X	AND	Oper,X	2	4
Absolute	AND	Oper	3	4
Absolute,X	AND	Oper,X	3	4*
Absolute,Y	AND	Oper,Y	3	4*
(Indirect,X)	AND	(Oper,X)	2	6
(Indirect),Y	AND	(Oper),Y	2	5*

* Add 1 if page boundary crossed

ASL Arithmetic shift left one bit ASL

```
---  -----------------                  N Z C I D V
|C| <-  |7|6|5|4|3|2|1|0| <- 0         % % % ~ ~ ~
---  -----------------
```

Addressing	Assembler	Format	Bytes	Cycles
Accumulator	ASL A		1	2
Zero Page	ASL	Oper	2	5
Zero Page,X	ASL	Oper,X	2	6
Absolute	ASL	Oper	3	6
Absolute,X	ASL	Oper,X	3	7

BCC Branch on carry clear BCC

```
Branch if C=0                    N Z C I D V
                                 ~ ~ ~ ~ ~ ~
```

Addresing	Assembler	Format	Bytes	Cycles
Relative	BCC	Oper	2	2*

* Add 1 if branch occurs to same page
 Add 2 if branch occurs to different page

BCS Branch on carry set BCS

```
Branch if C-1                    N Z C I D V
                                 ~ ~ ~ ~ ~ ~
```

Addressing	Assembler	Format	Bytes	Cycles
Relative	BCS	Oper	2	2*

* Add 1 if branch occurs to same page
 Add 2 if branch occurs to different page

BEQ Branch on result zero **BEQ**

Branch if Z=1

N	Z	C	I	D	V
~	~	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Relative	BCS	Oper	2	2*

* Add 1 if branch occurs to same page
Add 2 if branch occurs to different page

BIT Test bits in memory with accumulator **BIT**

A&M, N=M , V=M

7 6

N	Z	C	I	D	V
M 7	%	~	~	~	M 6

Bit 6 and 7 are transferred to the status register. If the result of A&M is zero then Z=1, otherwise Z=0.

Addressing	Assembler	Format	Bytes	Cycles
Zero Page	BIT	Oper	2	3
Absolute	BIT	Oper	3	4

BMI Branch on result minus **BMI**

Branch if N=1

N	Z	C	I	D	V
~	~	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Relative	BMI	Oper	2	2*

* Add 1 if branch occurs to same page
Add 2 if branch occurs to different page

BNE Branch on result not zero **BNE**

Branch if Z=0

N	Z	C	I	D	V
~	~	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Relative	BNE	Oper	2	2*

* Add 1 if branch occurs to same page
Add 2 if branch occurs to different page

BPL Branch on result plus **BPL**

N	Z	C	I	D	V
~	~	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Relative	BPL	Oper	2	2*

* Add 1 if branch occurs to same page
Add 2 if branch occurs to different page

```
BRK Force break                BRK
Forced interrupt; PC+2 ! P !   N Z C I D V
                               ~ ~ ~ 1 ~ ~
Addressing  Assembler  Format  Bytes  Cycles
Implied       BRK              1      7
A BRK command cannot be masked by setting I.


BVC Branch on overflow clear  BVC
Branch if V=0                  N Z C I D V
                               ~ ~ ~ ~ ~ ~
Addressing  Assembler  Format  Bytes  Cycles
Relative      BVC      Oper    2      2*
* Add 1 if branch occurs to same page
 Add 2 if branch occurs to different page

BVS Branch on overflow set   BVS
Branch if V=1                  N Z C I D V
                               ~ ~ ~ ~ ~ ~
Addressing  Assembler  Format  Bytes  Cycles
Relative      BVS      Oper    2      2*
* Add 1 if branch occurs to same page
 Add 2 if branch occurs to different page


CLC Clear carry flag          CLC
C=0                            N Z C I D V
                               ~ ~ 0 ~ ~ ~
Addressing  Assembler  Format  Bytes  Cycles
Implied       CLC              1      2


CLD Clear decimal mode       CLD
D=0                            N Z C I D V
                               ~ ~ ~ ~ 0 ~
Addressing  Assembler  Format  Bytes  Cycles
Implied       CLD              1       2

CLI Clear interrupt disable bit    CLI
                               N Z C I D V
                               ~ ~ ~ 0 ~ ~
Addressing  Assembler  Format  Bytes  Cycles
Implied       CLI              1       2
```

CLV Clear overflow flag **CLV**

V=0

N	Z	C	I	D	V
~	~	~	~	~	0

Addressing	Assembler	Format	Bytes	Cycles
Implied	CLV		1	2

CMP Compare memory and accumulator **CMP**

A-M

N	Z	C	I	D	V
%	%	%	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Immediate	CMP #	Oper	2	2
Zero Page	CMP	Oper	2	4
Zero Page,X	CMP	Oper,X	2	4
Absolute	CMP	Oper	3	4
Absolute,X	CMP	Oper,X	3	4*
Absolute,Y	CMP	Oper,Y	3	4*
(Indirect,X)	CMP	(Oper,X)	2	6
(Indirect),Y	CMP	(Oper),Y	2	5*

* Add 1 if page boundary crossed.

CPX Compare memory and index register X **CPX**

X-M

N	Z	C	I	D	V
%	%	%	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Immediate	CPX #	Oper	2	2
Zero Page	CPX	Oper	2	3
Absolute	CPX	Oper	3	4

CPY Compare memory and index register **CPY**

N	Z	C	I	D	V
%	%	%	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Immediate	CPY #	Oper	2	2
Zero Page	CPY	Oper	2	3
Absolute	CPY	Oper	3	4

DEC Decrement memory by one **DEC**

M=M-1

N	Z	C	I	D	V
%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Zero Page	DEC	Oper	2	5
Zero Page,X	DEC	Oper,X	2	6
Absolute	DEC	Oper	3	6
Absolute,X	DEC	Oper,X	3	7

DEX Decrement index register X by one **DEX**

X=X-1

N	Z	C	I	D	V
%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	BRK		1	2

DEY Decrement index register Y by one **DEY**

Y=Y-1

N	Z	C	I	D	V
%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	DEY		1	2

EOR Exclusive-OR memory with accumulator **EOR**

A=A:M

N	Z	C	I	D	V
%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Immediate	EOR #	Oper	2	2
Zero Page	EOR	Oper	2	3
Zero Page,X	EOR	Oper,X	2	4
Absolute	EOR	Oper	3	4
Absolute,X	EOR	Oper,X	3	4*
Absolute,Y	EOR	Oper,Y	3	4*
(Indirect,X)	EOR	(Oper,X)	2	6
(Indirect),Y	EOR	(Oper),Y	2	5*

* Add 1 if page boundary crossed.

INC Increment memory by one **INC**

M=M+1

N	Z	C	I	D	V
%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Zero Page	INC	Oper	2	5
Zero Page,X	INC	Oper,X	2	6
Absolute	INC	Oper	3	6
Absolute,X	INC	Oper,X	3	7

```
INX Increment index register X by one   INX
X=X+1                              N Z C I D V
                                   % % ~ ~ ~ ~
Addressing  Assembler  Format  Bytes  Cycles
Implied       INX               1      2


INY Increment index register Y by one   INY
Y=Y+1                              N Z C I D V
                                   % % ~ ~ ~ ~
Addressing  Assembler  Format  Bytes  Cycles
Implied       INY               1      2


JMP Jump to new location     JMP
PCL=(PC+1)                      N C Z I D V
PCH=(PC+2)                      ~ ~ ~ ~ ~ ~

Addressing  Assembler  Format  Bytes  Cycles
Absolute      JMP      Oper     3      3
Indirect      JMP      (Oper)   3      5


JSR Jump to subroutine saving return address  JSR
PC+2 !, PCL=(PC+1)              N Z C I D V
PCH=(PC+2)                      ~ ~ ~ ~ ~ ~

Addressing  Assembler  Format  Bytes  Cycles
Absolute      JSR      Oper     3      6


LDA Load accumulator with memory    LDA
A=M                             N Z C I D V
                                % % ~ ~ ~ ~
Addressing  Assembler  Format  Bytes  Cycles
Immediate     LDA #    Oper     2      2
Zero Page     LDA      Oper     2      3
Zero Page,X   LDA      Oper,X   2      4
Absolute      LDA      Oper     3      4
Absolute,X    LDA      Oper,X   3      4*
Absolute,Y    LDA      Oper,Y   3      4*
(Indirect,X)  LDA      (Oper,X) 2      6
(Indirect),Y  LDA      (Oper),Y 2      5*
* Add 1 if page boundary crossed.
```

LDX Load index register X with memory **LDX**

X=M

N	Z	C	I	D	V
%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Immediate	LDX #	Oper	2	2
Zero Page	LDX	Oper	2	3
Zero Page,Y	LDX	Oper,Y	2	4
Absolute	LDX	Oper	3	4
Absolute,Y	LDX	Oper,Y	3	4*

* Add 1 if page boundary crossed.

LDY Load index register Y with memory **LDY**

X=M

N	Z	C	I	D	V
%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Immediate	LDY #	Oper	2	2
Zero Page	LDY	Oper	2	3
Zero Page,X	LDY	Oper,X	2	4
Absolute	LDY	Oper	3	4
Absolute,X	LDY	Oper,X	3	4*

* Add 1 if page boundary crossed.

LSR Logical shift right one bit **LSR**

```
     ----------------- ---
0 -> |7|6|5|4|3|2|1|0| -> |C|
     ----------------- ---
```

N	Z	C	I	D	V
%	%	%	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Accumulator	LSR A		1	2
Zero Page	LSR	Oper	2	5
Zero Page,X	LSR	Oper,X	2	6
Absolute	LSR	Oper	3	6
Absolute,X	LSR	Oper,X	3	7

NOP No operation **NOP**

N	Z	C	I	D	V
~	~	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	NOP		1	2

ORA OR memory with accumulator ORA

A=A|M

			N	Z	C	I	D	V
			%	%	~	~	~	~
Addressing	**Assembler**	**Format**	**Bytes**	**Cycles**				
Immediate	ORA #	Oper	2	2				
Zero Page	ORA	Oper	2	3				
Zero Page,X	ORA	Oper,X	2	4				
Absolute	ORA	Oper	3	4				
Absolute,X	ORA	Oper,X	3	4*				
Absolute,Y	ORA	Oper,Y	3	4*				
(Indirect,X)	ORA	(Oper,X)	2	6				
(Indirect),Y	ORA	(Oper),Y	2	5*				

* Add 1 if page boundary crossed.

PHA Push accumulator on stack PHA

A !

			N	Z	C	I	D	V
			~	~	~	~	~	~
Addressing	**Assembler**	**Format**	**Bytes**	**Cycles**				
Implied	PHA		1	3				

PHP Push processor status on stack PHP

P !

			N	Z	C	I	D	V
			~	~	~	~	~	~
Addressing	**Assembler**	**Format**	**Bytes**	**Cycles**				
Implied	PHP		1	3				

PLA Pull accumulator from stack PLA

A ^

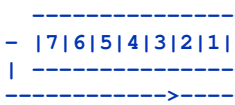
			N	Z	C	I	D	V
			%	%	~	~	~	~
Addressing	**Assembler**	**Format**	**Bytes**	**Cycles**				
Implied	PLA		1	4				

PLP Pull processor status from stack PLP

A ^

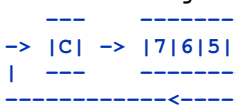
			N	Z	C	I	D	V
			from			stack		
Addressing	**Assembler**	**Format**	**Bytes**	**Cycles**				
Implied	PLP		1	4				

ROL Rotate left one bit **ROL**



Addressing	Assembler	Format	Bytes	Cycles
Accumulator	ROL A		1	2
Zero Page	ROL	Oper	2	5
Zero Page,X	ROL	Oper,X	2	6
Absolute	ROL	Oper	3	6
Absolute,X	ROL	Oper,X	3	7

ROR Rotate right one bit **ROR**



Addressing	Assembler	Format	Bytes	Cycles
Accumulator	ROR A		1	2
Zero Page	ROR	Oper	2	5
Zero Page,X	ROR	Oper,X	2	6
Absolute	ROR	Oper	3	6
Absolute,X	ROR	Oper,X	3	7

RTI Return from interrupt **RTI**

P^ PC^ N Z C I D V
from stack

Addressing	Assembler	Format	Bytes	Cycles
Implied	RTI		1	6

RTS Return from subroutine **RTS**

PC^ N Z C I D V
~ ~ ~ ~ ~ ~

Addressing	Assembler	Format	Bytes	Cycles
Implied	RTS		1	6

SBC Subtract memory from accumulator with carry SBC

A,C=A-M-(1-C)

			N	Z	C	I	D	V
			%	%	%	~	~	%

Addressing	Assembler	Format	Bytes	Cycles
Immediate	SBC #	Oper	2	2
Zero Page	SBC	Oper	2	3
Zero Page,X	SBC	Oper,X	2	4
Absolute	SBC	Oper	3	4
Absolute,X	SBC	Oper,X	3	4*
Absolute,Y	SBC	Oper,Y	3	4*
(Indirect,X)	SBC	(Oper,X)	2	6
(Indirect),Y	SBC	(Oper),Y	2	5*

* Add 1 if page boundary crossed.

SEC Set carry flag SEC

C=1

N	Z	C	I	D	V
~	~	1	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	SEC		1	2

SED Set decimal mode SED

D=1

N	Z	C	I	D	V
~	~	~	~	1	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	SEI		1	2

STA Store accumulator in memory SBC

M=A

N	Z	C	I	D	V
~	~	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Immediate	STA #	Oper	2	3
Zero Page	STA	Oper	2	4
Zero Page,X	STA	Oper,X	3	4
Absolute	STA	Oper	3	5
Absolute,X	STA	Oper,X	3	4
Absolute,Y	STA	Oper,Y	3	4
(Indirect,X)	STA	(Oper,X)	2	6
(Indirect),Y	STA	(Oper),Y	2	6

* Add 1 if page boundary crossed.

STX Store index register X in memory STX

M=X			N	Z	C	I	D	V
			~	~	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Zero Page	STX	Oper	2	3
Zero Page,Y	STX	Oper,Y	2	4
Absolute	STX	Oper	3	4

STY Store index register Y in memory STY

M=Y			N	Z	C	I	D	V
			~	~	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Zero Page	STX	Oper	2	3
Zero Page,X	STX	Oper,X	2	4
Absolute	STX	Oper	3	4

TAX Transfer accumulator to index register X TAX

X=A			N	Z	C	I	D	V
			%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	TAX		1	2

TAY Transfer accumulator to index register Y TAY

Y=A			N	Z	C	I	D	V
			%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	TAY		1	2

TSX Transfer accumulator to index register X TSX

X=S			N	Z	C	I	D	V
			%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	TSX		1	2

TXA Transfer index register X to accumulator TXA

A=X	N	Z	C	I	D	V
	%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	TXA		1	2

TXS Transfer index register X to stack pointer TAX

S=X	N	Z	C	I	D	V
	%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	TXS		1	2

TXA Transfer index register X to accumulator TXA

X=A	N	Z	C	I	D	V
	%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	TXS		1	2

TYA Transfer accumulator to index register Y TYA

A=Y	N	Z	C	I	D	V
	%	%	~	~	~	~

Addressing	Assembler	Format	Bytes	Cycles
Implied	TYA		1	2

The Acorn Guide to the Electron
Neil and Pat Cryer

The Acorn Electron (described in Which Micro? as 'a winner') is probably, for the price, the most advanced personal computer on the market. This guide, published with the full cooperation of the manufactuers, describes and explains everything a non-technical owner needs to know in order to get the most from this versatile and amazing new machine. The Electron is designed to be fun, useful and, above all, the best introduction to the new Age of Computers. It has been developed by the people responsible for the BBC Micro -- the machine that is part of the syllabus of over 80 per cent of our schools. Both computers understand the same language and both have been designed to grow with your understanding of their capability and you needs.
You may be thinking of buying this book because you have just bought the new Acorn Eletron or maybe you have bought the new Acorn Electron or maybe you have bought the machine as a present for a friend or relative. Whichever is the case, The Acorn Guide to the Electron is the indispensable companion to your machine.