6502 Microprocessor

DavidSolid (LateX document), Unknown (Original)

April 22, 2019

Contents

1 Introduction

Most of the following information has been taking out of the "Commodore 64 Programmers Reference Manual" simply because it was available in electronic form and there appears to be no difference between this documentation and the 6502 documentation, they are both from the 6500 family after all. I've made changes and additions where appropriate.

In theory you should be able to use any code you can find for emulating the 6510 (the C64 processor).

2 Registers

Almost all calculations are done in the microprocessor. Registers are special pieces of memory in the processor which are used to carry out, and store information about calculations. The 6502 has the following registers:

2.1 Accumulator (A)

This is **the most important register** in the microprocessor. Various machine language instructions allow you to copy the contents of a memory location into the accumulator, copy the contents of the accumulator into a memory location, modify the contents of the accumulator or some other register directly, without affecting any memory. And the accumulator is the only register that has instructions for performing math.

2.2 X Index Register

This is a very important register. There are instructions for nearly all of the transformations you can make to the accumulator. But there are other instructions for things that only the X register can do. Various machine language instructions allow you to copy the contents of a memory location into the X register, copy the contents of the X register into a memory location, and modify the contents of the X, or some other register directly.

2.3 Y Index Register

This is a very important register. There are instructions for nearly all of the transformations you can make to the accumulator, and the X register. But there are other instructions for things that only the Y register can do. Various machine language instructions allow you to copy the contents of a memory location into the Y register, copy the contents of the Y register into a memory location, and modify the contents of the Y, or some other register directly.

2.4 Status register

This register consists of eight "flags" (a flag = something that indi- cates whether something has, or has not occurred). Bits of this register are altered depending on the result of arithmetic and logical operations. These bits are described below:

Bit No.	7	6	5	4	3	2	1	0
Simbols	S	V		B	D	I	Z	C

Table 1: Status register layout

- Bit 0 - C - Carry flag: this holds the carry out of the most significant bit in any arithmetic operation. In subtraction operations however, this flag is cleared - set to 0 - if a borrow is required, set to 1 - if no borrow is required. The carry flag is also used in shift and rotate logical operations.
- Bit 1 - Z - Zero flag: this is set to 1 when any arithmetic or logical operation produces a zero result, and is set to 0 if the result is non-zero.
- Bit 2 - I: this is an interrupt enable/disable flag. If it is set, interrupts are disabled. If it is cleared, interrupts are enabled.
- Bit 3 - D: this is the decimal mode status flag. When set, and an Add with Carry or Subtract with Carry instruction is executed, the source values are treated as valid BCD (Binary Coded Decimal, eg. 0x00-0x99 = 0-99) numbers. The result generated is also a BCD number.
- Bit 4 - B: this is set when a software interrupt (BRK instruction) is executed.
- Bit 5: not used. Supposed to be logical 1 at all times.
- Bit 6 - V - Overflow flag: when an arithmetic operation produces a result too large to be represented in a byte, V is set.
- Bit 7 - S - Sign flag: this is set if the result of an operation is negative, cleared if positive.

The most commonly used flags are C, Z, V, S.

2.5 Program Counter

This contains the address of the current machine language instruction being executed. Since the operating system is always "RUN"ning in the Commodore VIC-20 (or, for that matter, any computer), the program counter is always changing. It could only be stopped by halting the microprocessor in some way.

2.6 Stack pointer

This register contains the location of the first empty place on the stack. The stack is used for temporary storage by machine language pro- grams, and by the computer.

3 Addressing Modes

Instructions need operands to work on. There are various ways of indicating where the processor is to get these operands. The different methods used to do this are called addressing modes. The 6502 offers 11 modes, as described below.

3.1 Immediate

In this mode the operand's value is given in the instruction itself. In assembly language this is indicated by "#" before the operand. eg. LDA #$0A - means "load the accumulator with the hex value 0A" In machine code different modes are indicated by different codes. So LDA would be translated into different codes depending on the addressing mode. In this mode, it is: $A9 $0A

3.2 Absolute and Zero-page Absolute

In these modes the operands address is given. eg. LDA $31F6 - (assembler) $AD $31F6 - (machine code) If the address is on zero page - i.e. any address where the high byte is 00 - only 1 byte is needed for the address. The processor automatically fills the 00 high byte. eg. LDA $F4 $A5 $F4 Note the different instruction codes for the different modes. Note also that for 2 byte addresses, the low byte is store first, eg. LDA $31F6 is stored as three bytes in memory, $AD $F6 $31. Zero-page absolute is usually just called zero-page.

3.3 Implied

No operand addresses are required for this mode. They are implied by the instruction. eg. TAX - (transfer accumulator contents to X-register) $AA

3.4 Accumulator

In this mode the instruction operates on data in the accumulator, so no operands are needed. eg. LSR - logical bit shift right $4A

3.5 Indexed and Zero-page Indexed

In these modes the address given is added to the value in either the X or Y index register to give the actual address of the operand. eg. LDA $31F6, Y $D9 $31F6 LDA $31F6, X $DD $31F6 Note that the different operation codes determine the index register used. In the zero-page version, you should note

that the X and Y registers are not interchangeable. Most instructions which can be used with zero-page indexing do so with X only. eg. LDA $20, X $B5 $20

3.6 Indirect

This mode applies only to the JMP instruction - JuMP to new location. It is indicated by parenthesis around the operand. The operand is the address of the bytes whose value is the new location. eg. JMP ($215F) Assume the following -

byte	value
$215F	$76
$2160	$30

This instruction takes the value of bytes $215F, $2160 and uses that as the address to jump to - i.e. $3076 (remember that addresses are stored with low byte first).

3.7 Pre-indexed indirect

In this mode a zer0-page address is added to the contents of the X-register to give the address of the bytes holding the address of the operand. The indirection is indicated by parenthesis in assembly language. eg. LDA ($3E, X) $A1 $3E Assume the following -

byte	value
X-reg.	$05
$0043	$15
$0044	$24
$2415	$6E

Then the instruction is executed by:

1. adding $3E and $05 = $0043
2. getting address contained in bytes $0043, $0044 = $2415
3. loading contents of $2415 - i.e. $6E - into accumulator
4. Note
   (a) When adding the 1-byte address and the X-register, wrap around addition is used - i.e. the sum is always a zero-page address. eg. FF + 2 = 0001 not 0101 as you might expect. DON'T FORGET THIS WHEN EMULATING THIS MODE.
   (b) Only the X register is used in this mode.

3.8 Post-indexed indirect

In this mode the contents of a zero-page address (and the following byte) give the indirect addressm which is added to the contents of the Y-register to yield the actual address of the operand. Again, inassembly language, the instruction is indicated by parenthesis. eg. LDA ($4C), Y Note that the parenthesis are only around the 2nd byte of the instruction since it is the part that does the indirection. Assume the following -

byte	value
$004C	$00
$004D	$21
Y-reg.	$05
$2105	$6D

Then the instruction above executes by:

1. getting the address in bytes $4C, $4D = $2100
2. adding the contents of the Y-register = $2105
3. loading the contents of the byte $2105 - i.e. $6D into the accumulator.

Note: only the Y-register is used in this mode.

3.9 Relative

This mode is used with Branch-on-Condition instructions. It is probably the mode you will use most often. A 1 byte value is added to the program counter, and the program continues execution from that address. The 1 byte number is treated as a signed number - i.e. if bit 7 is 1, the number given byt bits 0-6 is negative; if bit 7 is 0, the number is positive. This enables a branch displacement of up to 127 bytes in either direction.

Table 2: Example

bit no.	7	6	5	4	3	2	1	0	signed value	unsigned value
value	1	0	1	0	0	1	1	1	-39	$A7
value	0	0	1	0	0	1	1	1	+39	$27

Instruction example: BEQ $A7 $F0 $A7 This instruction will check the zero status bit. If it is set, 39 decimal will be subtracted from the program counter and execution continues from that address. If the zero status bit is not set, execution continues from the following instruction. Notes:

1. The program counter points to the start of the instruction after the branch instruction before the branch displacement is added. Remember to take this into account when calculating displacements.

2. Branch-on-condition instructions work by checking the relevant status bits in the status register. Make sure that they have been set or unset as you want them. This is often done using a CMP instruction.

3. If you find you need to branch further than 127 bytes, use the opposite branch-on-condition and a JMP.

4 Instruction Set

MCS6502 MICROPROCESSOR INSTRUCTION SET - ALPHABETIC SEQUENCE	
Mnemonic	Description
ADC	Add Memory to Accumulator with Carry
AND	"AND" Memory with Accumulator
ASL	Shift Left One Bit (Memory or Accumulator)
BCC	Branch on Carry Clear
BCS	Branch on Carry Set
BEQ	Branch on Result Zero
BIT	Test Bits in Memory with Accumulator
BMI	Branch on Result Minus
BNE	Branch on Result not Zero
BPL	Branch on Result Plus
BRK	Force Break
BVC	Branch on Overflow Clear
BVS	Branch on Overflow Set
CLC	Clear Carry Flag
CLD	Clear Decimal Mode
CLI	Clear interrupt Disable Bit
CLV	Clear Overflow Flag
CMP	Compare Memory and Accumulator
CPX	Compare Memory and Index X
CPY	Compare Memory and Index Y
DEC	Decrement Memory by One
DEX	Decrement Index X by One
DEY	Decrement Index Y by One
EOR	"Exclusive-Or" Memory with Accumulator
INC	Increment Memory by One
INX	Increment Index X by One
INY	Increment Index Y by One
JMP	Jump to New Location
Continues in next page	

JSR	Jump to New Location Saving Return Address
LDA	Load Accumulator with Memory
LDX	Load Index X with Memory
LDY	Load Index Y with Memory
LSR	Shift Right One Bit (Memory or Accumulator)
NOP	No Operation
ORA	"OR" Memory with Accumulator
PHA	Push Accumulator on Stack
PHP	Push Processor Status on Stack
PLA	Pull Accumulator from Stack
PLP	Pull Processor Status from Stack
ROL	Rotate One Bit Left (Memory or Accumulator)
ROR	Rotate One Bit Right (Memory or Accumulator)
RTI	Return from Interrupt
RTS	Return from Subroutine
SBC	Subtract Memory from Accumulator with Borrow
SEC	Set Carry Flag
SED	Set Decimal Mode
SEI	Set Interrupt Disable Status
STA	Store Accumulator in Memory
STX	Store Index X in Memory
STY	Store Index Y in Memory
TAX	Transfer Accumulator to Index X
TAY	Transfer Accumulator to Index Y
TSX	Transfer Stack Pointer to Index X
TXA	Transfer Index X to Accumulator
TXS	Transfer Index X to Stack Pointer
TYA	Transfer Index Y to Accumulator

Table 3: List of documented opcodes

The following notation applies to this summary:

Note: At the top of each table is located in parentheses a reference number (Ref: XX) which directs the user to that Section in the MCS6500 Microcomputer Family Programming Manual in which the instruction is defined and discussed.

Symbol	Meaning	Symbol	Meaning
A	Accumulator	EOR	Logical Exclusive Or
X, Y	Index Registers	fromS	Transfer from Stack
M	Memory	toS	Transfer to Stack
P	Processor Status Register	$\Rightarrow$	Transfer to
S	Stack Pointer	$\Leftarrow$	Transfer from
/	Change	$\vee$	Logical OR
-	No Change	PC	Program Counter
+	Add	PCH	Program Counter High
$\wedge$	Logical AND	PCL	Program Counter Low
-	Subtract	OPER	OPERAND
		#	IMMEDIATE ADDRESSING MODE

Table 4: Notation table

4.1 ADC (Add memory to accumulator with carry)

Operation: $A + M + C \implies A, C$

N	Z	C	I	D	V
/	/	/	-	-	/

(Ref: 2.2.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	ADC #Oper	69	2	2
Zero Page	ADC Oper	65	2	3
Zero Page,X	ADC Oper,X	75	2	4
Absolute	ADC Oper	60	3	4
Absolute,X	ADC Oper,X	70	3	4*
Absolute,Y	ADC Oper,Y	79	3	4*
(Indirect,X)	ADC (Oper,X)	61	2	6
(Indirect),Y	ADC (Oper),Y	71	2	5*

* Add 1 if page boundary is crossed.

4.2 AND ("AND" memory with accumulator)

Operation: $A \wedge M \implies A$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 2.2.3.0)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	AND #Oper	29	2	2
Zero Page	AND Oper	25	2	3
Zero Page,X	AND Oper,X	35	2	4
Absolute	AND Oper	2D	3	4
Absolute,X	AND Oper,X	3D	3	4*
Absolute,Y	AND Oper,Y	39	3	4*
(Indirect,X)	AND (Oper,X)	21	2	6
(Indirect,Y)	AND (Oper),Y	31	2	5

* Add 1 if page boundary is crossed.

4.3 ASL (ASL Shift Left One Bit (Memory or Accumulator))

Operation: $C \Leftarrow$ | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $\Leftarrow 0$

N	Z	C	I	D	V
/	/	/	-	-	-

(Ref: 10.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Accumulator	ASL A	0A	1	2
Zero Page	ASL Oper	06	2	5
Zero Page,X	ASL Oper,X	16	2	6
Absolute	ASL Oper	0E	3	6
Absolute, X	ASL Oper,X	1E	3	7

4.4 BCC (Branch on Carry Clear)

Operation: Branch on C = 0

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 4.1.1.3)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BCC Oper	90	2	2*

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

4.5 BCS (BCS Branch on carry set)

Operation: Branch on C = 1

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 4.1.1.4)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BCS Oper	B0	2	2*

* Add 1 if branch occurs to same page. * Add 2 if branch occurs to next page.

4.6 BEQ (BEQ Branch on result zero)

Operation: Branch on Z = 1

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 4.1.1.5)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BEQ Oper	F0	2	2*

* Add 1 if branch occurs to same page. * Add 2 if branch occurs to next page.

4.7 BIT (BIT Test bits in memory with accumulator)

Operation: $A \wedge M, M7 \implies N, M6 \implies V$

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

N	Z	C	I	D	V
M7	/	-	-	-	M6

(Ref: 4.2.1.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	BIT Oper	24	2	3
Absolute	BIT Oper	2C	3	4

4.8 BMI (BMI Branch on result minus)

Operation: Branch on N = 1

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 4.1.1.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BMI Oper	30	2	2*

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

4.9 BNE (BNE Branch on result not zero)

Operation: Branch on Z = 0

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 4.1.1.6)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BMI Oper	D0	2	2*

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

4.10 BPL (BPL Branch on result plus)

Operation: Branch on N = 0

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 4.1.1.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BPL Oper	10	2	2*

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

4.11 BRK (BRK Force Break)

Operation: Forced Interrupt PC + 2 toS P toS

N	Z	C	I	D	V
-	-	-	1	-	-

(Ref: 9.11)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	BRK	00	1	7

1. A BRK command cannot be masked by setting I.

4.12 BVC (BVC Branch on overflow clear)

Operation: Branch on V = 0

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 4.1.1.8)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BVC Oper	50	2	2*

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

4.13 BVS (BVS Branch on overflow set)

Operation: Branch on V = 1

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 4.1.1.7)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Relative	BVS Oper	70	2	2*

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

4.14 CLC (CLC Clear carry flag)

Operation: $0 \implies C$

N	Z	C	I	D	V
-	-	0	-	-	-

(Ref: 3.0.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	CLC	18	1	2

4.15 CLD (CLD Clear decimal mode)

Operation: $0 \implies D$

N	A	C	I	D	V
-	-	-	-	0	-

(Ref: 3.3.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	CLD	D8	1	2

4.16 CLI (CLI Clear interrupt disable bit)

Operation: $0 \implies I$

N	Z	C	I	D	V
-	-	-	0	-	-

(Ref: 3.2.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	CLI	58	1	2

4.17 CLV (CLV Clear overflow flag)

Operation: $0 \implies V$

N	Z	C	I	D	V
-	-	-	-	-	0

(Ref: 3.6.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	CLV	B8	1	2

4.18 CMP (CMP Compare memory and accumulator)

Operation: A - M

N	Z	C	I	D	V
/	/	/	-	-	-

(Ref: 4.2.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	CMP #Oper	C9	2	2
Zero Page	CMP Oper	C5	2	3
Zero Page,X	CMP Oper,X	D5	2	4
Absolute	CMP Oper	CD	3	4
Absolute,X	CMP Oper,X	DD	3	4*
Absolute,Y	CMP Oper,Y	D9	3	4*
(Indirect,X)	CMP (Oper,X)	C1	2	6
(Indirect),Y	CMP (Oper),Y	D1	2	5*

* Add 1 if page boundary is crossed.

4.19 CPX (CPX Compare Memory and Index X)

Operation: X - M

N	Z	C	I	D	V
/	/	/	-	-	-

(Ref: 7.8)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	CPX #Oper	E0	2	2
Zero Page	CPX Oper	E4	2	3
Absolute	CPX Oper	EC	3	4

4.20 CPY (CPY Compare memory and index Y)

Operation: Y - M

N	Z	C	I	D	V
/	/	/	-	-	-

(Ref: 7.9)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	CPY #Oper	C0	2	2
Zero Page	CPY Oper	C4	2	3
Absolute	CPY Oper	CC	3	4

4.21 DEC (DEC Decrement memory by one)

Operation: $M - 1 \implies M$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 10.7)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	DEC Oper	C6	2	5
Zero Page,X	DEC Oper,X	D6	2	6
Absolute	DEC Oper	CE	3	6
Absolute,X	DEC Oper,X	DE	3	7

4.22 DEX (DEX Decrement index X by one)

Operation: $X - 1 \implies X$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.6)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	DEX	CA	1	2

4.23 DEY (DEY Decrement index Y by one)

Operation: $X - 1 \implies Y$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.7)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	DEY	88	1	2

4.24 EOR (EOR "Exclusive-Or" memory with accumulator)

Operation: $AEORM \implies A$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 2.2.3.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	EOR #Oper	49	2	2
Zero Page	EOR Oper	45	2	3
Zero Page,X	EOR Oper,X	55	2	4
Absolute	EOR Oper	40	3	4
Absolute,X	EOR Oper,X	50	3	4*
Absolute,Y	EOR Oper,Y	59	3	4*
(Indirect,X)	EOR (Oper,X)	41	2	6
(Indirect),Y	EOR (Oper),Y	51	2	5*

* Add 1 if page boundary is crossed.

4.25 INC (INC Increment memory by one)

Operation: $M + 1 \implies M$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 10.6)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	INC Oper	E6	2	5
Zero Page,X	INC Oper,X	F6	2	6
Absolute	INC Oper	EE	3	6
Absolute,X	INC Oper,X	FE	3	7

4.26 INX (INX Increment Index X by one)

Operation: $X + 1 \implies X$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.4)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	INX	E8	1	2

4.27 INY (INY Increment Index Y by one)

Operation: $X + 1 \implies X$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.5)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	INY	C8	1	2

4.28 JMP (JMP Jump to new location)

Operation: $(PC + 1) -> PCL\ (PC + 2) -> PCH$

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 4.0.2) (Ref: 9.8.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Absolute	JMP Oper	4C	3	3
Indirect	JMP (Oper)	6C	3	5

4.29 JSR (JSR Jump to new location saving return address)

Operation: $PC + 2toS, (PC + 1) \implies PCL\ (PC + 2) \implies PCH$

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 8.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Absolute	JSR Oper	20	3	6

LDA LDA Load accumulator with memory LDA
Operation: $M \implies A$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 2.1.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	LDA #Oper	A9	2	2
Zero Page	LDA Oper	A5	2	3
Zero Page,X	LDA Oper,X	B5	2	4
Absolute	LDA Oper	AD	3	4
Absolute,X	LDA Oper,X	BD	3	4*
Absolute,Y	LDA Oper,Y	B9	3	4*
(Indirect,X)	LDA (Oper,X)	A1	2	6
(Indirect),Y	LDA (Oper),Y	B1	2	5*

* Add 1 if page boundary is crossed.

4.30 LDX (LDX Load index X with memory)

Operation: $M \implies X$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.0)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	LDX #Oper	A2	2	2
Zero Page	LDX Oper	A6	2	3
Zero Page,Y	LDX Oper,Y	B6	2	4
Absolute	LDX Oper	AE	3	4
Absolute,Y	LDX Oper,Y	BE	3	4*

* Add 1 when page boundary is crossed.

4.31 LDY (LDY Load index Y with memory)

Operation: $M \implies Y$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	LDY #Oper	A0	2	2
Zero Page	LDY Oper	A4	2	3
Zero Page,X	LDY Oper,X	B4	2	4
Absolute	LDY Oper	AC	3	4
Absolute,X	LDY Oper,X	BC	3	4*

* Add 1 when page boundary is crossed.

4.32 LSR (LSR Shift right one bit (memory or accumulator))

Operation: $0 \Longrightarrow$ | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | $\Longrightarrow C$

N	Z	C	I	D	V
0	/	/	-	-	-

(Ref: 10.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Accumulator	LSR A	4A	1	2
Zero Page	LSR Oper	46	2	5
Zero Page,X	LSR Oper,X	56	2	6
Absolute	LSR Oper	4E	3	6
Absolute,X	LSR Oper,X	5E	3	7

4.33 NOP (NOP No operation)

Operation: No Operation (2 cycles)

N	Z	C	I	D	V
-	-	-	-	-	-

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	NOP	EA	1	2

4.34 ORA (ORA "OR" memory with accumulator)

Operation: $A \vee M \Longrightarrow A$

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 2.2.3.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	ORA #Oper	09	2	2
Zero Page	ORA Oper	05	2	3
Zero Page,X	ORA Oper,X	15	2	4
Absolute	ORA Oper	0D	3	4
Absolute,X	ORA Oper,X	1D	3	4*
Absolute,Y	ORA Oper,Y	19	3	4*
(Indirect,X)	ORA (Oper,X)	01	2	6
(Indirect),Y	ORA (Oper),Y	11	2	5

* Add 1 on page crossing

4.35 PHA (PHA Push accumulator on stack)

Operation: A toS

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 8.5)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	PHA	48	1	3

4.36 PHP (PHP Push processor status on stack)

Operation: P toS

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 8.11)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	PHP	08	1	3

4.37 PLA (PLA Pull accumulator from stack)

Operation: A fromS

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 8.6)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	PLA	68	1	4

4.38 PLP (PLP Pull processor status from stack)

Operation: P fromS

N	Z	C	I	D	V
From Stack					

(Ref: 8.12)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	PLP	28	1	4

4.39 ROL (Rotate one bit left (memory or accumulator))

M or A

Operation: ...⇐ | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ⇐ | C | ⇐...

N	Z	C	I	D	V
/	/	/	-	-	-

(Ref: 10.3)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Accumulator	ROL A	2A	1	2
Zero Page	ROL Oper	26	2	5
Zero Page,X	ROL Oper,X	36	2	6
Absolute	ROL Oper	2E	3	6
Absolute,X	ROL Oper,X	3E	3	7

4.40 ROR (ROR Rotate one bit right (memory or accumulator))

M or A

Operation: ...⇒ | C | ⇒ | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | ⇒...

N	Z	C	I	D	V
/	/	/	-	-	-

(Ref: 10.4)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Accumulator	ROR A	6A	1	2
Zero Page	ROR Oper	66	2	5
Zero Page,X	ROR Oper,X	76	2	6
Absolute	ROR Oper	6E	3	6
Absolute,X	ROR Oper,X	7E	3	7

Note: ROR instruction is available on MCS650X microprocessors after June, 1976.

4.41 RTI (RTI Return from interrupt)

Operation: P fromS PC fromS

N	Z	C	I	D	V
From Stack					

(Ref: 9.6)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	RTI	4D	1	6

4.42 RTS (RTS Return from subroutine)

Operation: PC fromS, PC + 1 $\implies$ PC

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 8.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	RTS	60	1	6

4.43 SBC (SBC Subtract memory from accumulator with borrow)

Operation: A - M - C^B $\implies$ A

N	Z	C	I	D	V
/	/	/	-	-	/

Note:C^B = Borrow
(Ref: 2.2.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Immediate	SBC #Oper	E9	2	2
Zero Page	SBC Oper	E5	2	3
Zero Page,X	SBC Oper,X	F5	2	4
Absolute	SBC Oper	ED	3	4
Absolute,X	SBC Oper,X	FD	3	4*
Absolute,Y	SBC Oper,Y	F9	3	4*
(Indirect,X)	SBC (Oper,X)	E1	2	6
(Indirect),Y	SBC (Oper),Y	F1	2	5

* Add 1 when page boundary is crossed.

4.44 SEC (SEC Set carry flag)

Operation: 1 $\Longrightarrow$ C

N	Z	C	I	D	V
-	-	1	-	-	-

(Ref: 3.0.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	SEC	38	1	2

4.45 SED (SED Set decimal mode)

Operation: 1 $\Longrightarrow$ D

N	Z	C	I	D	V
-	-	-	-	1	-

(Ref: 3.3.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	SED	F8	1	2

4.46 SEI (SEI Set interrupt disable status)

Operation: 1 $\Longrightarrow$ I

N	Z	C	I	D	V
-	-	-	1	-	-

(Ref: 3.2.1)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	SEI	78	1	2

4.47 STA (STA Store accumulator in memory)

Operation: A $\Longrightarrow$ M

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 2.1.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	STA Oper	85	2	3
Zero Page,X	STA Oper,X	95	2	4
Absolute	STA Oper	80	3	4
Absolute,X	STA Oper,X	90	3	5
Absolute,Y	STA Oper, Y	99	3	5
(Indirect,X)	STA (Oper,X)	81	2	6
(Indirect),Y	STA (Oper),Y	91	2	6

4.48 STX (STX Store index X in memory)

Operation: X $\Longrightarrow$ M

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 7.2)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	STX Oper	86	2	3
Zero Page,Y	STX Oper,Y	96	2	4
Absolute	STX Oper	8E	3	4

4.49 STY (STY Store index Y in memory)

Operation: Y $\Longrightarrow$ M

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 7.3)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Zero Page	STY Oper	84	2	3
Zero Page,X	STY Oper,X	94	2	4
Absolute	STY Oper	8C	3	4

4.50 TAX (TAX Transfer accumulator to index X)

Operation: A $\Longrightarrow$ X

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.11)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TAX	AA	1	2

4.51 TAY (TAY Transfer accumulator to index Y)

Operation: A $\Longrightarrow$ Y

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.13)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TAY	A8	1	2

4.52 TSX (TSX Transfer stack pointer to index X)

Operation: S $\Longrightarrow$ X

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 8.9)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TSX	BA	1	2

4.53 TXA (TXA Transfer index X to accumulator)

Operation: X $\Longrightarrow$ A

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.12)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TXA	8A	1	2

4.54 TXS (TXS Transfer index X to stack pointer)

Operation: X $\Longrightarrow$ S

N	Z	C	I	D	V
-	-	-	-	-	-

(Ref: 8.8)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TXS	9A	1	2

4.55 TYA (TYA Transfer index Y to accumulator)

Operation: Y $\Longrightarrow$ A

N	Z	C	I	D	V
/	/	-	-	-	-

(Ref: 7.14)

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	TYA	98	1	2

5 General instruction set overview

5.1 Instruction addressing modes and related execution times (in clock cycles)

	ADC	AND	ASL	BCC	BCS	BEQ	BIT	BMI	BNE	BPL	BRK	BVC	BVS	CLC
Accumulator	.	.	2	.	.	.	.	.	.	.	.	.	.	.
Immediate	2	2		.	.	.	.	.	.	.	.	.	.	.
Zero Page	3	3	5	.	.	.	3	.	.	.	.	.	.	.
Zero Page,X	4	4	6	.	.	.	.	.	.	.	.	.	.	.
Zero Page,Y	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Absolute	4	4	6	.	.	.	4	.	.	.	.	.	.	.
Absolute,X	4*	4*	7	.	.	.	.	.	.	.	.	.	.	.
Absolute,Y	4*	4*	.	.	.	.	.	.	.	.	.	.	.	.
Implied	.	.	.	.	.	.	.	.	.	.	.	.	.	2
Relative	.	.	.	2**	2**	2**	.	2**	2**	2**	7	2**	2**	.
(Indirect,X)	6	6	.	.	.	.	.	.	.	.	.	.	.	.
(Indirect),Y	5*	5*	.	.	.	.	.	.	.	.	.	.	.	.
Abs. Indirect	.	.	.	.	.	.	.	.	.	.	.	.	.	.

	CLD	CLI	CLV	CMP	CPX	CPY	DEC	DEX	DEY	EOR	INC	INX	INY	JMP
Accumulator	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Immediate	.	.	.	2	2	2	.	.	.	2	.	.	.	.
Zero Page	.	.	.	3	3	3	5	.	.	3	5	.	.	.
Zero Page,X	.	.	.	4	.	.	6	.	.	4	6	.	.	.
Zero Page,Y	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Absolute	.	.	.	4	4	4	6	.	.	4	6	.	.	3
Absolute,X	.	.	.	4*	.	.	7	.	.	4*	7	.	.	.
Absolute,Y	.	.	.	4*	.	.	.	.	.	4*	.	.	.	.
Implied	2	2	2	.	.	.	.	2	2	.	.	2	2	.
Relative	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(Indirect,X)	.	.	.	6	.	.	.	.	.	6	.	.	.	.
(Indirect),Y	.	.	.	5*	.	.	.	.	.	5*	.	.	.	.
Abs. Indirect	.	.	.	.	.	.	.	.	.	.	.	.	.	5

* Add one cycle if indexing across page boundary ** Add one cycle if branch is taken, Add one additional if branching operation crosses page boundary

J S R	L D A	L D X	L D Y	L S R	N O P	O R A	P H A	P H P	P L A	P L P	R O L	R O R	R T I	
Accumulator	.	.	.	.	2	.	.	.	.	.	.	2	2	.
Immediate	.	2	2	2	.	.	2	.	.	.	.	.	.	.
Zero Page	.	3	3	3	5	.	3	.	.	.	.	5	5	.
Zero Page,X	.	4	.	4	6	.	4	.	.	.	.	6	6	.
Zero Page,Y	.	.	4	.	.	.	.	.	.	.	.	.	.	.
Absolute	6	4	4	4	6	.	4	.	.	.	.	6	6	.
Absolute,X	.	4*	.	4*	7	.	4*	.	.	.	.	7	7	.
Absolute,Y	.	4*	4*	.	.	.	4*	.	.	.	.	.	.	.
Implied	.	.	.	.	.	2	.	3	3	4	4	.	.	6
Relative	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(Indirect,X)	.	6	.	.	.	.	6	.	.	.	.	.	.	.
(Indirect),Y	.	5*	.	.	.	.	5*	.	.	.	.	.	.	.
Abs. Indirect	.	.	.	.	.	.	.	.	.	.	.	.	.	.

	R T S	S B C	S E C	S E D	S E I	S T A	S T X	S T Y	T A X	T A Y	T S X	T X A	T X S	T Y A
Accumulator	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Immediate	.	2	.	.	.	.	.	.	.	.	.	.	.	.
Zero Page	.	3	.	.	.	3	3	3	.	.	.	.	.	.
Zero Page,X	.	4	.	.	.	4	.	4	.	.	.	.	.	.
Zero Page,Y	.	.	.	.	.	.	4	.	.	.	.	.	.	.
Absolute	.	4	.	.	.	4	4	4	.	.	.	.	.	.
Absolute,X	.	4*	.	.	.	5	.	.	.	.	.	.	.	.
Absolute,Y	.	4*	.	.	.	5	.	.	.	.	.	.	.	.
Implied	6	.	2	2	2	.	.	.	2	2	2	2	2	2
Relative	.	.	.	.	.	.	.	.	.	.	.	.	.	.
(Indirect,X)	.	6	.	.	.	6	.	.	.	.	.	.	.	.
(Indirect),Y	.	5*	.	.	.	6	.	.	.	.	.	.	.	.
Abs. Indirect	.	.	.	.	.	.	.	.	.	.	.	.	.	.

* Add one cycle if indexing across page boundary ** Add one cycle if branch is taken, Add one additional if branching operation crosses page boundary

5.2 Instruction set ordered by opcode

00 00 - BRK

01 01 - ORA - (Indirect,X)

02 02 - Future Expansion

03 03 - Future Expansion

04 04 - Future Expansion

05 05 - ORA - Zero Page

06 06 - ASL - Zero Page

07 07 - Future Expansion

08 08 - PHP

09 09 - ORA - Immediate

0A 0A - ASL - Accumulator

0B 0B - Future Expansion

0C 0C - Future Expansion

0D 0D - ORA - Absolute

0E 0E - ASL - Absolute

0F 0F - Future Expansion

10 10 - BPL

11 11 - ORA - (Indirect),Y

12 12 - Future Expansion

13 13 - Future Expansion

14 14 - Future Expansion

15 15 - ORA - Zero Page,X

16 16 - ASL - Zero Page,X

17 17 - Future Expansion

18 18 - CLC

19 19 - ORA - Absolute,Y

1A 1A - Future Expansion

1B 1B - Future Expansion

1C 1C - Future Expansion

1D 1D - ORA - Absolute,X

1E 1E - ASL - Absolute,X

1F 1F - Future Expansion

20 20 - JSR

21 21 - AND - (Indirect,X)

22 22 - Future Expansion

23 23 - Future Expansion

24 24 - BIT - Zero Page

25 25 - AND - Zero Page

26 26 - ROL - Zero Page

27 27 - Future Expansion

28 28 - PLP

29 29 - AND - Immediate

2A 2A - ROL - Accumulator

2B 2B - Future Expansion

2C 2C - BIT - Absolute

2D 2D - AND - Absolute

2E 2E - ROL - Absolute

2F 2F - Future Expansion

30 30 - BMI

31 31 - AND - (Indirect),Y

32 32 - Future Expansion

33 33 - Future Expansion

34 34 - Future Expansion

35 35 - AND - Zero Page,X

36 36 - ROL - Zero Page,X

37 37 - Future Expansion

38 38 - SEC

39 39 - AND - Absolute,Y

3A 3A - Future Expansion

3B 3B - Future Expansion

3C 3C - Future Expansion

3D 3D - AND - Absolute,X

3E 3E - ROL - Absolute,X

3F 3F - Future Expansion

40 40 - RTI

41 41 - EOR - (Indirect,X)

42 42 - Future Expansion

43 43 - Future Expansion

44 44 - Future Expansion

45 45 - EOR - Zero Page

46 46 - LSR - Zero Page

47 47 - Future Expansion

48 48 - PHA

49 49 - EOR - Immediate

4A 4A - LSR - Accumulator

4B 4B - Future Expansion

4C 4C - JMP - Absolute

4D 4D - EOR - Absolute

4E 4E - LSR - Absolute

4F 4F - Future Expansion

50 50 - BVC

51 51 - EOR - (Indirect),Y

52 52 - Future Expansion

53 53 - Future Expansion

54 54 - Future Expansion

55 55 - EOR - Zero Page,X

56 56 - LSR - Zero Page,X

57 57 - Future Expansion

58 58 - CLI

59 59 - EOR - Absolute,Y

5A 5A - Future Expansion

5B 5B - Future Expansion

5C 5C - Future Expansion

5D 50 - EOR - Absolute,X

5E 5E - LSR - Absolute,X

5F 5F - Future Expansion

60 60 - RTS

61 61 - ADC - (Indirect,X)

62 62 - Future Expansion

63 63 - Future Expansion

64 64 - Future Expansion

65 65 - ADC - Zero Page

66 66 - ROR - Zero Page

67 67 - Future Expansion

68 68 - PLA

69 69 - ADC - Immediate

6A 6A - ROR - Accumulator

6B 6B - Future Expansion

6C 6C - JMP - Indirect

6D 6D - ADC - Absolute

6E 6E - ROR - Absolute

6F 6F - Future Expansion

70 70 - BVS

71 71 - ADC - (Indirect),Y

72 72 - Future Expansion

73 73 - Future Expansion

74 74 - Future Expansion

75 75 - ADC - Zero Page,X

76 76 - ROR - Zero Page,X

77 77 - Future Expansion

78 78 - SEI

79 79 - ADC - Absolute,Y

7A 7A - Future Expansion

7B 7B - Future Expansion

7C 7C - Future Expansion

7D 70 - ADC - Absolute,X

7E 7E - ROR - Absolute,X

7F 7F - Future Expansion

80 80 - Future Expansion

81 81 - STA - (Indirect,X)

82 82 - Future Expansion

83 83 - Future Expansion

84 84 - STY - Zero Page

85 85 - STA - Zero Page

86 86 - STX - Zero Page

87 87 - Future Expansion

88 88 - DEY

89 89 - Future Expansion

8A 8A - TXA

8B 8B - Future Expansion

8C 8C - STY - Absolute

8D 80 - STA - Absolute

8E 8E - STX - Absolute

8F 8F - Future Expansion

90 90 - BCC

91 91 - STA - (Indirect),Y

92 92 - Future Expansion

93 93 - Future Expansion

94 94 - STY - Zero Page,X

95 95 - STA - Zero Page,X

96 96 - STX - Zero Page,Y

97 97 - Future Expansion

98 98 - TYA

99 99 - STA - Absolute,Y

9A 9A - TXS

9B 9B - Future Expansion

9C 9C - Future Expansion

9D 90 - STA - Absolute,X

9E 9E - Future Expansion

9F 9F - Future Expansion

A0 A0 - LDY - Immediate

A1 A1 - LDA - (Indirect,X)

A2 A2 - LDX - Immediate

A3 A3 - Future Expansion

A4 A4 - LDY - Zero Page

A5 A5 - LDA - Zero Page

A6 A6 - LDX - Zero Page

A7 A7 - Future Expansion

A8 A8 - TAY

A9 A9 - LDA - Immediate

AA AA - TAX

AB AB - Future Expansion

AC AC - LDY - Absolute

AD AD - LDA - Absolute

AE AE - LDX - Absolute

AF AF - Future Expansion

B0 B0 - BCS

B1 B1 - LDA - (Indirect),Y

B2 B2 - Future Expansion

B3 B3 - Future Expansion

B4 B4 - LDY - Zero Page,X

B5 BS - LDA - Zero Page,X

B6 B6 - LDX - Zero Page,Y

B7 B7 - Future Expansion

B8 B8 - CLV

B9 B9 - LDA - Absolute,Y

BA BA - TSX

BB BB - Future Expansion

BC BC - LDY - Absolute,X

BD BD - LDA - Absolute,X

BE BE - LDX - Absolute,Y

BF BF - Future Expansion

C0 C0 - Cpy - Immediate

C1 C1 - CMP - (Indirect,X)

C2 C2 - Future Expansion

C3 C3 - Future Expansion

C4 C4 - CPY - Zero Page

C5 C5 - CMP - Zero Page

C6 C6 - DEC - Zero Page

C7 C7 - Future Expansion

C8 C8 - INY

C9 C9 - CMP - Immediate

CA CA - DEX

CB CB - Future Expansion

CC CC - CPY - Absolute

CD CD - CMP - Absolute

CE CE - DEC - Absolute

CF CF - Future Expansion

D0 D0 - BNE

D1 D1 - CMP (Indirect@,Y

D2 D2 - Future Expansion

D3 D3 - Future Expansion

D4 D4 - Future Expansion

D5 D5 - CMP - Zero Page,X

D6 D6 - DEC - Zero Page,X

D7 D7 - Future Expansion

D8 D8 - CLD

D9 D9 - CMP - Absolute,Y

DA DA - Future Expansion

DB DB - Future Expansion

DC DC - Future Expansion

DD DD - CMP - Absolute,X

DE DE - DEC - Absolute,X

DF DF - Future Expansion

E0 E0 - CPX - Immediate

E1 E1 - SBC - (Indirect,X)

E2 E2 - Future Expansion

E3 E3 - Future Expansion

E4 E4 - CPX - Zero Page

E5 E5 - SBC - Zero Page

E6 E6 - INC - Zero Page

E7 E7 - Future Expansion

E8 E8 - INX

E9 E9 - SBC - Immediate

EA EA - NOP

EB EB - Future Expansion

EC EC - CPX - Absolute

ED ED - SBC - Absolute

EE EE - INC - Absolute

EF EF - Future Expansion

F0 F0 - BEQ

F1 F1 - SBC - (Indirect),Y

F2 F2 - Future Expansion

F3 F3 - Future Expansion

F4 F4 - Future Expansion

F5 F5 - SBC - Zero Page,X

F6 F6 - INC - Zero Page,X

F7 F7 - Future Expansion

F8 F8 - SED

F9 F9 - SBC - Absolute,Y

FA FA - Future Expansion

FB FB - Future Expansion

FC FC - Future Expansion

FD FD - SBC - Absolute,X

FE FE - INC - Absolute,X

FF FF - Future Expansion