HuC6280

CMOS 8-bit Microprocessor

SOFTWARE MANUAL

TABLE OF CONTENTS

1. DESCRIPTION

The HuC6280 software provides a set of instructions by which the HuC6280 hardware works. This manual describes the HuC6280 instruction set using the HuC6280 assembly language, addressing modes and instruction formats, and details of the individual instructions.

2. HuC6280 INSTRUCTION SET

The HuC6280 provides a total of 89 basic instructions available to the user. To save memory and to fasten operation, they are optimized for programming, and they are classified into seven categories:

- ALU instructions
- Flag instructions
- Data transfer instructions
- Branch instructions
- Subroutine instructions
- Test instructions
- Control instructions

Tables 2-2-1 through 2-2-3 list all the HuC6280 instructions and the functions assigned to them, arranged under the above classification. Each of the HuC6280 instructions uses the notation and mnemonic shown in the tables when used.
Tables 2-3-1 through 2-3-3 contain the HuC6280 instructions arranged in alphabetical order.

2.1 Definition of Symbols and Terms Used throughout this Manual

- Addressing Mode: The term "addressing mode" means how the CPU gets the address of operand. Addressing modes are described in the notation specified in Table 2-1-1.

Table 2-1-1 Addressing Modes

Addressing mode	Notation	Addressing mode	Notation
Implied	IMPLID	Absolute X-register indexed	ABS, X
Immediate	IMM	Absolute Y-register indexed	ABS, Y
Zero page	ZP	Absolute indirect	(ABS)
Zero page X-register indexed	ZP, X	Absolute indexed indirect	(ABS, X)
Zero page Y-register indexed	ZP, Y	Relative	REL
Zero page relative	ZP, REL	Immediate zero page	IMM ZP
Zero page indirect	(IND)	Immediate zero page indexed	IMM ZP, X
Zero page indexed indirect	(IND, X)	Immediate absolute	IMM ABS
Zero page indirect indexed	(IND), Y	Immediate absolute indexed	IMM ABS, X
Absolute	ABS	Accumulator	ACC

● Mnemonic: The term "mnemonic" means a symbolic representation for each instruction written in an assembly language. For the HuC6280 instructions, the symbol specified in Table 2-1-2 is used to describe each.

Table 2-1-2 Mnemonic

Symbol	Description	Symbol	Description
A	Accumulator	hh	High-order byte (hex) of address on memory
X	X-register		
Y	Y-register	ll	Low-order byte (hex) of address on memory
M	Memory		
Ms	Memory (stack)	ZZ	Low-order byte (hex) of address on zero page
Mi	Any specified bit of memory		
M_6	Bit 6 of memory	∧	AND
M_7	Bit 7 of memory	∨	OR
M(X)	Zero page memory specified by X-register	⊻	Exclusive-OR
		+	Add
IMM	Immediate data	−	Subtract
MPR	Mapping register	#	Indicates immediate data.*
MPRi	Specified mapping register	nn	8 bits of data
N	Negative flag	i	Bit data or mapping register number
V	Overflow flag	␣	Tab or space
T	Memory operation flag	rr	Offset (hex) of relative branch instruction
B	Break command		
D	Decimal flag	SH	High-order byte of source address
I	Interrupt disable	SL	Low-order byte of source address
Z	Zero flag	DH	High-order byte of destination address
C	Carry flag		
$\overline{C}$	Carry not, borrow	DL	Low-order byte of destination address
PC	Program counter	LH	High-order byte of length
PCH	High-order byte of program counter	LL	Low-order byte of length
PCL	Low-order byte of program counter	M_{DD}	Memory specified by DH and DL
S	Stack pointer	M_{SS}	Memory specified by SH and SL
P	Status register	x	Number of block transfer bytes

*) Used with mnemonic and machine code.

● Flag: A flag indicates how the status register changes after an instruction is executed. The symbols for flags are shown in Table 2-1-3.

Table 2-1-3 Flags

Symbol	Description	Symbol	Description
N, V, Z or C	Indicates that each flag changes as a result of instruction execution.	−	The flag remains unchanged.
		M_7	Bit 7 of memory is set.
		M_6	Bit 6 of memory is set.
1	The flag is set.	(RESTORED)	The data in the stack is loaded to the status register.
0	The flag is reset.		

*) N: Negative flag, V: Overflow flag, Z: Zero flag, C: Carry flag

2.2 Classification of HuC6280 Instructions

Table 2-2-1 Mnemonic for Instructions and their Functions (1)

Category	Mnemonic	Function
ALU instruc-tions	ADC	Add with Carry
	AND	AND
	ASL	Shift Left
	CLA	Clear A
	CLX	Clear X
	CLY	Clear Y
	CMP	Compare A with M
	CPX	Compare X with M
	CPY	Compare Y with M
	DEC	Decrement
	DEX	Decrement X
	DEY	Decrement Y
	EOR	Exclusive-OR
	INC	Increment
	INX	Increment X
	INY	Increment Y
	LSR	Shift Right
	ORA	OR
	ROL	Rotate Left
	ROR	Rotate Right
	SBC	Subtract with Carry
Flag instruct-ions	CLC	Clear C
	CLD	Clear D
	CLI	Clear I
	CLV	Clear O
	SEC	Set C
	SED	Set D
	SEI	Set I
	SET	Set T
Data transfer instruct-ions	LDA	Load A
	LDX	Load X
	LDY	Load Y
	SAX	Swap A for X
	SAY	Swap A for Y
	ST0	Store HuC6270 No. 1
	ST1	Store HuC6270 No. 2
	ST2	Store HuC6270 No. 3
	STA	Store A
	STX	Store X
	STY	Store Y
	STZ	Store Zero

Table 2-2-2 Mnemonic for Instructions and their Functions (2)

Category	Mnemonic	Function
Data transfer instructions	SXY	Swap X for Y
	TAI	Transfer Block Data
	TAMi	Transfer A to MPR
	TAX	Transfer A to X
	TAY	Transfer A to Y
	TDD	Transfer Block Data
	TIA	Transfer Block Data
	TII	Transfer Block Data
	TIN	Transfer Block Data
	TMAi	Transfer MPR to A
	TSX	Transfer S to X
	TXA	Transfer X to A
	TXS	Transfer X to S
	TYA	Transfer Y to A
Branch instructions	BBRi	Branch on Bit Reset
	BBSi	Branch on Bit Set
	BCC	Branch on Carry Clear
	BCS	Branch on Carry Set
	BEQ	Branch on Equal
	BMI	Branch on Minus
	BNE	Branch on Not Equal
	BPL	Branch on Plus
	BRA	Branch Always
	BVC	Branch on V Clear
	BVS	Branch on V Set
	JMP	Jump to New Location
Subroutine instructions	BSR	Branch Subroutine
	JSR	Jump to Subroutine
	PHA	Push A
	PHP	Push P
	PHX	Push X
	PHY	Push Y
	PLA	Pull A
	PLP	Pull P
	PLX	Pull X
	PLY	Pull Y
	RTI	Return from Interrupt
	RTS	Return from Subroutine

Table 2-2-3 Mnemonic for Instructions and their Functions (3)

Category	Mnemonic	Function
Test instruct-ions	BIT	Bit Test
	TRB	Test and Reset Bit
	TSB	Test and Set Bit
	TST	Test Memory
Control instruct-ions	BRK	Break
	NOP	No operation
	RMBi	Reset Memory Bit
	SMBi	Set Memory Bit

2.3 HuC6280 Instructions Listed in Alphabetical Order

Table 2-3-1

Mnemonic	Function	Flag								Reference page
		N	V	T	B	D	I	Z	C	
ADC	A ←A+M+C (when T=0) M(x)←M(x)+M+C (when T=1)	N	V	0	—	—	—	Z	C	S8-17
AND	A ←A∧M (when T=0) M(x)←M(X)∧M (when T=1)	N	—	0	—	—	—	Z	—	S8-18
ASL	C← [7 0] ←0	N	—	0	—	—	—	Z	C	S8-19
BBRi	Branch on Mi = 0	—	—	0	—	—	—	—	—	S8-20
BBSi	Branch on Mi = 1	—	—	0	—	—	—	—	—	S8-21
BCC	Branch on C = 0	—	—	0	—	—	—	—	—	S8-22
BCS	Branch on C = 1	—	—	0	—	—	—	—	—	S8-23
BEQ	Branch on Z = 1	—	—	0	—	—	—	—	—	S8-24
BIT	A∧M	M_7	M_6	0	—	—	—	Z	—	S8-25
BMI	Branch on N = 1	—	—	0	—	—	—	—	—	S8-26
BNE	Branch on Z = 0	—	—	0	—	—	—	—	—	S8-27
BPL	Branch on N = 0	—	—	0	—	—	—	—	—	S8-28
BRA	Branch Always	—	—	0	—	—	—	—	—	S8-29
BRk	Break	—	—	0	1	0	1	—	—	S8-30
BSR	Branch Subroutine	—	—	0	—	—	—	—	—	S8-31
BVC	Branch on V = 0	—	—	0	—	—	—	—	—	S8-32
BVS	Branch on V = 1	—	—	0	—	—	—	—	—	S8-33
CLA	A ← 00_{16}	—	—	0	—	—	—	—	—	S8-34
CLC	C ← 0	—	—	0	—	—	—	—	0	S8-35
CLD	D ← 0	—	—	0	—	0	—	—	—	S8-36
CLI	I ← 0	—	—	0	—	—	0	—	—	S8-37
CLV	V ← 0	—	0	0	—	—	—	—	—	S8-38
CLX	X ← 00_{16}	—	—	0	—	—	—	—	—	S8-39
CLY	Y ← 00_{16}	—	—	0	—	—	—	—	—	S8-40
CMP	A-M	N	—	0	—	—	—	Z	C	S8-41
CPX	X-M	N	—	0	—	—	—	Z	C	S8-42
CPY	Y-M	N	—	0	—	—	—	Z	C	S8-43
DEC	M ← M-1 or A ← A-1	N	—	0	—	—	—	Z	—	S8-44
DEX	X ← X-1	N	—	0	—	—	—	Z	—	S8-45
DEY	Y ← Y-1	N	—	0	—	—	—	Z	—	S8-46

Table 2-3-2

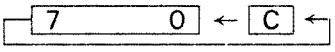
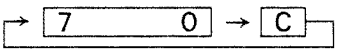
Mnemonic	Function	Flag								Reference page
		N	V	T	B	D	I	Z	C	
EOR	A ← A∀M	N	–	0	–	–	–	Z	–	S8-47
INC	M ← M+1 or A ← A+1	N	–	0	–	–	–	Z	–	S8-58
INX	X ← X+1	N	–	0	–	–	–	Z	–	S8-59
INY	Y ← Y+1	N	–	0	–	–	–	Z	–	S8-50
JMP	Jump to New Location	–	–	0	–	–	–	–	–	S8-51
JSR	Jump to Subroutine	–	–	0	–	–	–	–	–	S8-52
LDA	A ← M	N	–	0	–	–	–	Z	–	S8-53
LDX	X ← M	N	–	0	–	–	–	Z	–	S8-54
LDY	Y ← M	N	–	0	–	–	–	Z	–	S8-55
LSR	0→ [7 0] →C	0	–	0	–	–	–	Z	C	S8-56
NOP	No Operation	–	–	0	–	–	–	–	–	S8-57
ORA	A ← A∨M	N	–	0	–	–	–	Z	–	S8-68
PHA	Ms ← A, S ← S-1	–	–	0	–	–	–	–	–	S8-69
PHP	Ms ← P, S ← S-1	–	–	0	–	–	–	–	–	S8-60
PHX	Ms ← X, S ← S-1	–	–	0	–	–	–	–	–	S8-61
PHY	Ms ← Y, S ← S-1	–	–	0	–	–	–	–	–	S8-62
PLA	S ← S+1, A ← Ms	N	–	0	–	–	–	Z	–	S8-63
PLP	S ← S+1, P ← Ms	(RESTORED)								S8-64
PLX	S ← S+1, X ← Ms	N	–	0	–	–	–	Z	–	S8-65
PLY	S ← S+1, Y ← Ms	N	–	0	–	–	–	Z	–	S8-66
RMBi	Mi ← 0	–	–	0	–	–	–	–	–	S8-67
ROL		N	–	0	–	–	–	Z	C	S8-78
ROR		N	–	0	–	–	–	Z	C	S8-79
RTI	Return from Interrupt	(RESTORED)								S8-70
RTS	Return from Subroutine	–	–	0	–	–	–	–	–	S8-71
SAX	A ↔ X	–	–	0	–	–	–	–	–	S8-72
SAY	A ↔ Y	–	–	0	–	–	–	–	–	S8-73
SBC	A ← A-M-$\overline{C}$	N	V	0	–	–	–	Z	C	S8-74
SEC	C ← 1	–	–	0	–	–	–	–	1	S8-75
SED	D ← 1	–	–	0	–	1	–	–	–	S8-76
SEI	I ← 1	–	–	0	–	–	1	–	–	S8-77
SET	T ← 1	–	–	1	–	–	–	–	–	S8-88
SMBi	Mi ← 1	–	–	0	–	–	–	–	–	S8-89

Table 2-3-3

Mnemonic	Function	Flag								Reference page
		N	V	T	B	D	I	Z	C	
ST0	HuC6270: (A1, A0)=(0,0) ← IM	—	—	0	—	—	—	—	—	S8-80
ST1	HuC6270: (A1, A0)=(1,0) ← IM	—	—	0	—	—	—	—	—	S8-81
ST2	HuC6270: (A1, A0)=(1,1) ← IM	—	—	0	—	—	—	—	—	S8-82
STA	M ← A	—	—	0	—	—	—	—	—	S8-83
STX	M ← X	—	—	0	—	—	—	—	—	S8-84
STY	M ← Y	—	—	0	—	—	—	—	—	S8-85
STZ	M ← 00_{16}	—	—	0	—	—	—	—	—	S8-86
SXY	X ⟷ Y	—	—	0	—	—	—	—	—	S8-87
TAI	Transfer Block Data (INC ← ALT)	—	—	0	—	—	—	—	—	S8-90
TAMi	MPRi ← A	—	—	0	—	—	—	—	—	S8-88
TAX	X ← A	N	—	0	—	—	—	Z	—	S8-89
TAY	Y ← A	N	—	0	—	—	—	Z	—	S8-92
TDD	Transfer Block Data (DEC ← DEC)	—	—	0	—	—	—	—	—	S8-93
TIA	Transfer Block Data (ALT ← INC)	—	—	0	—	—	—	—	—	S8-94
TII	Transfer Block Data (INC ← INC)	—	—	0	—	—	—	—	—	S8-95
TIN	Transfer Block Data (FIX ← INC)	—	—	0	—	—	—	—	—	S8-97
TMAi	A ← MPRi	—	—	0	—	—	—	—	—	S8-99
TRB	M ← $\overline{A} \wedge M$	M_7	M_6	0	—	—	—	Z	—	S8-101
TSB	M ← $A \vee M$	M_7	M_6	0	—	—	—	Z	—	S8-103
TST	$M \wedge IM$	M_7	M_6	0	—	—	—	Z	—	S8-104
TSX	X ← S	N	—	0	—	—	—	Z	—	S8-105
TXA	A ← X	N	—	0	—	—	—	Z	—	S8-106
TXS	S ← X	—	—	0	—	—	—	—	—	S8-107
TYA	A ← Y	N	—	0	—	—	—	Z	—	S8-108

3. ADDRESSING MODES AND INSTRUCTION FORMATS

The HuC6280 provides twenty addressing modes available to the user. The addressing modes and instruction formats are described in this section.

3.1 Implied

Three types (Nos. 1-3) of implied addressing mode are available.

No.1 Standard format

PC	OP code

Description

- Abbreviation for the mode: IMPLID
- 1-byte instruction
- The OP code specifies the source and destination.

No. 2 Special format

PC	OP code
PC+1	Mapping register number

Description

- Abbreviation for the mode: IMPLID
- Applicable to TAMi and TMAi instructions only.
- 2-byte instruction
- The second byte specifies the mapping register number.

Mapping register number	Second byte
MPR0	01_{16}
MPR1	02_{16}
MPR2	04_{16}
MPR3	08_{16}
MPR4	10_{16}
MPR5	20_{16}
MPR6	40_{16}
MPR7	80_{16}

No. 3 Special format

PC	OP code
PC+1	Low-order byte of source address
PC+2	High-order byte of source address
PC+3	Low-order byte of destination address
PC+4	High-order byte of destination address
PC+5	Low-order byte of length
PC+6	High-order byte of length

Description

- Abbreviation for the mode: IMPLID
- Applicable to the following block transfer instructions only:
  - TAI
  - TDD
  - TIA
  - TII
  - TIN
- 7-byte instruction
- The second and third bytes specify the source address. The fourth and fifth bytes specify the destination address. The sixth and seventh bytes specify the length or the number of bytes to be trasnferred.

3.2 Immediate

Format

PC	OP code
PC+1	Immediate data

Description

- Abbreviation for the mode: IMM
- 2-byte instruction
- The second byte contains the immediate data as the operand.

3.3 Zero Page

Format

PC	OP code
PC+1	Low-order byte of zero-page address

Description

- Abbreviation for the mode: ZP
- 2-byte instruction
- The second byte equals the low-order byte of a zero-page address. Its high-order byte always contains the logical address 20_{16}.

3.4 Zero Page X-Register Indexed

Format

PC	OP code
PC+1	Low-order byte of zero-page address

Description

- Abbreviations for the mode: ZP, X
- 2-byte instruction
- The X-register is added to the second byte to generate the low-order byte of a zero-page address. Its high-order byte always contains the logical address 20_{16}.

3.5 Zero Page Y-Register Indexed

Format

PC	OP code
PC+1	Low-order byte of zero-page address

Description

- Abbreviation for the mode: ZP, Y
- 2-byte instruction
- The Y-register is added to the second byte to generate the low-order byte of a zero-page address. Its high-order byte always contains the logical address 20_{16}.

3.6 Zero Page Relative

Format

PC	OP code
PC+1	Low-order byte of zero-page address
PC+2	Offset to destination address

Description

- Abbreviation for the mode: ZP, REL
- 3-byte instruction
- Applicable to BBSi and BBRi (i=0-7) instructions only.
- The second byte equals the low-order byte of a zero-page address. Its high-order byte contains the logical address 20_{16}.
- The third byte contains an offset to the destination address.

Offset to destination address = Destination address − (PC+3)

3.7 Zero Page Indirect

Format

PC	OP code
PC+1	Low-order byte of zero-page address

Description

- Abbreviation for the mode: (IND)
- 2-byte instruction
- The memory address is contained in the zero page in such an order that the low-order byte precedes the high-order byte. The second byte of the instruction generates the address of the low-order byte in the zero page.

3.8 Zero Page Indexed Indirect

Format

PC	OP code
PC+1	Low-order byte of zero-page address

Description

- Abbreviation for the mode: (IND, X)
- 2-byte instruction
- The memory address is contained in the zero page in such an order that the low-order byte precedes the high-order byte. The second byte of the instruction generates the address of the low-order byte specified in the zero page (X-register indexed).

3.9 Zero Page Indirect Indexed

Format

PC	OP code
PC+1	Low-order byte of zero-page address

Description

- Abbreviation for the mode: (IND), Y
- 2-byte instruction
- The zero page contains 16-bit data in such an order that the low-order byte precedes the high-order byte. The Y register is added to the 16-bit data to generate a memory address. The second byte of the instruction generates the address of the low-order byte in the zero page.

3.10 Absolute

Format

PC	OP code
PC+1	Low-order byte of memory address
PC+2	High-order byte of memory address

Description

- Abbreviation for the mode: ABS
- 3-byte instruction
- The second and third bytes specify the memory address.

3.11 Absolute X-Register Indexed

Format

PC	OP code
PC+1	Low-order byte of memory address
PC+2	High-order byte of memory address

Description

- Abbreviation for the mode: ABS, X
- 3-byte instruction
- The X-register is added to the address specified by the second and third bytes to generate an address.

3.12 Absolute Y-Register Indexed

Format

PC	OP code
PC+1	Low-order byte of memory address
PC+2	High-order byte of memory address

Description

- Abbreviation for the mode: ABS, Y
- 3-byte instruction
- The Y-register is added to the address specified by the second and third bytes to generate an address.

3.13 Absolute Indirect

Format

PC	OP code
PC+1	Low-order byte of memory address
PC+2	High-order byte of memory address

Description

- Abbreviation for the mode: (ABS)
- 3-byte instruction
- The memory address is contained in the memory in such an order that the low-order byte precedes the high-order byte. The address of the low-order byte is specified in the absolute mode.

3.14 Absolute Indexed Indirect

Format

PC	OP code
PC+1	Low-order byte of memory address
PC+2	High-order byte of memory address

Description

- Abbreviation for the mode: (ABS, X)
- 3-byte instruction
- The memory address is contained in the memory in such an order that the low-order byte precedes the high-order byte. The X-register is added to the 16 bits of data generated by the second and third bytes of the instruction to specify the address of the above low-order byte.

3.15 Relative

Format

PC	OP code
PC+1	Offset to destination address

Description

- Abbreviation for the mode: REL
- 2-byte instruction
- Applicable to relative branch instructions
- Offset to destination address
  = Destination address–(PC+2)

3.16 Immediate Zero Page

Format

PC	OP code
PC+1	Immediate data
PC+2	Low-order byte of zero-page address

Description

- Abbreviation for the mode: IMM ZP
- 3-byte instruction
- Applicable to the TST instruction only.
- The immediate data is ANDed with the zero-page data in order to change the status register.

3.17 Immediate Zero Page Indexed

Format

PC	OP code
PC+1	Immediate data
PC+2	Low-order byte of zero-page address

Description

- Abbreviation for the mode: IMM ZP, X
- 3-byte instruction
- Applicable to the TST instruction only.
- The immediate data is ANDed with the zero-page data indexed by the X-register in order to change the status register.

3.18 Immediate Absolute

Format

PC	OP code
PC+1	Immediate data
PC+2	Low-order byte of memory address
PC+3	High-order byte of memory address

Description

- Abbreviation for the mode: IMM ABS
- 4-byte instruction
- Applicable to the TST instruction only.
- The immediate data is ANDed with the memory data in order to change the status register.

3.19 Immediate Absolute Indexed

Format

PC	OP code
PC+1	Immediate data
PC+2	Low-order byte of memory address
PC+3	High-order byte of memory address

Description

- Abbreviation for the mode: IMM ABS, X
- 4-byte instruction
- Applicable to the TST instruction only.
- The immediate data is ANDed with the memory data indexed by the X-register in order to change the status register.

3.20 Accumulator

Format

PC	OP code

Description

- Abbreviation for the mode: ACC
- 1-byte instruction.

4. DESCRIPTION OF INSTRUCTIONS

This section describes what function each of the 89 HuC6280 instructions provides, how it is described in the assembly language, and what effect it has on the status register.
The format of and the meaning of each header for the information given for each description are as follows:

Function: — This part describes the function of each instruction. The description consists of two parts:

- Detailed function description
- Brief function description

Instruction: — This part describes the applicable addressing modes, mnemonic, machine code, byte count, and cycle count of the instruction.

- Addressing mode: The addressing modes applicable to the instruction are given, using the abbreviations listed in Table 2-1-1.
- Mnemonic: How to describes the instruction in the assemble language is indicated. The symbols used in the description are listed in Table 2-1-2.
- Machine code: The hexadecimal machine code for the instruction is given. It has an OP code at its beginning, followed by operands if necessary.
- Bytes: The number of bytes the instruction has is given.
- Cycles: The number of cycles the instruction requires for its execution is given. One cycle consists of one bus cycle (read, write, or dummy cycle).

Flags: — This part indicates how the status register changes as a result of instruction execution. Symbols listed in Table 2-1-3 are used in this description.

ADC (Add with Carry)

Function: The ADC instruction operates in either of two different ways depending on T flag.

i) When T=1 (a SET instruction was executed immediately before the ADC):
M(x), M, and C are added and the result is stored in M(x). The number of cycles given in the table below is increased by 3.
M(x) ← M+M(x)+C

ii) When T=0 (a SET instruction was not executed immediately before the ADC):
A, M, and C are added and the result is stored in A.
A ← A+M+C

The ADC instruction also operates in either of two different ways depending on the D flag.

i) When D=1
A decimal add operation is performed. The number of cycles given in the table below is increased by 1. V is unaffected.

ii) When D=0
A binary add operation is performed.

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣ADC␣#nn	69, nn	2	2
ZP	␣ADC␣ZZ	65, ZZ	2	4
ZP, X	␣ADC␣ZZ, X	75, ZZ	2	4
(IND)	␣ADC␣(ZZ)	72, ZZ	2	7
(IND, X)	␣ADC␣(ZZ, X)	61, ZZ	2	7
(IND), Y	␣ADC␣(ZZ), Y	71, ZZ	2	7
ABS	␣ADC␣hhll	6D, ll, hh	3	5
ABS, X	␣ADC␣hhll, X	7D, ll, hh	3	5
ABS, Y	␣ADC␣hhll, Y	79, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	V	0	–	–	–	Z	C

AND (And)

Function: The AND instruction operates in either of two different ways depending on the T flag.

i) When T = 1 (a SET instruction was executed immediately before the AND)
M(x) is ANDed with M and the result is stored in M(x). The number of cycles given in the table below is increased by 3.
M(x) ← M(x)∧M

ii) When T = 0 (a SET instruction was not executed immediately before the AND)
A is ANDed with M and the result is stored in A.
A ← A∧M

Instruction:

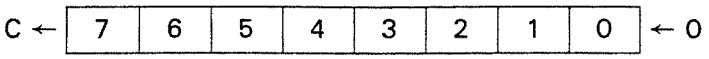
Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣AND␣#nn	29, nn	2	2
ZP	␣AND␣ZZ	25, ZZ	2	4
ZP, X	␣AND␣ZZ, X	35, ZZ	2	4
(IND)	␣AND␣(ZZ)	32, ZZ	2	7
(IND, X)	␣AND␣(ZZ, X)	21, ZZ	2	7
(IND), Y	␣AND␣(ZZ), Y	31, ZZ	2	7
ABS	␣AND␣hhll	2D, ll, hh	3	5
ABS, X	␣AND␣hhll, X	3D, ll, hh	3	5
ABS, Y	␣AND␣hhll, Y	39, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

ASL (Shift Left)

Function: The content of memory or the accumulator is shifted left by one bit. 0 is set in M0 or A0, and M7 or A7 in C.



Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣ASL␣ZZ	06, ZZ	2	6
ZP, X	␣ASL␣ZZ, X	16, ZZ	2	6
ABS	␣ASL␣hhll	0E, ll, hh	3	7
ABS, X	␣ASL␣hhll, X	1E, ll, hh	3	7
ACC	␣ASL␣A	0A	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	C

BBRi (Branch on Bit Reset)

Function: If the specified zero-page bit is "0", the CPU branches to the specified relative address. The number of cycles given in the table below is increases by 2. If the specified zero-page bit is "1", the program counter is increased by 3 and the BBRi instruction has no effect.

PC ← PC+3+rr if Mi=0

PC ← PC+3 if Mi=1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP, REL	␣BBRi␣ZZ, hhll	10i+F, ZZ, rr	3	6

NOTE: rr=hhll−(PC+3)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

BBSi (Branch on Bit Set)

Function: If the specified zero-page bit is "1", the CPU branches to the specified relative address. The number of cycles given in the table below is increases by 2. If the specified zero-page bit is "0", the program counter is increased by 3 and the BBSi instruction has no effect.

PC ← PC+3+rr if Mi=1

PC ← PC+3 if Mi=0

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP, REL	␣BBSi␣ZZ, hhll	10i+8F, ZZ, rr	3	6

NOTE: rr=hhll−(PC+3)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

BCC (Branch on Carry Clear)

Function: If the C flag is "0", the CPU branches to the specified relative address. The number of cycles given in the table below is increased by 2. If the C flag is "1", the program counter is increased by 2 and the BBRi instruction has no effect.

PC ← PC+2+rr if C=0

PC ← PC+2 if C=1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BBC␣hhll	90, rr	2	2

NOTE: rr=hhll−(PC+2)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

BCS (Branch on Carry Set)

Function: If the C flag is "1", the CPU branches to the specified relative address. The number of cycles given in the table below is increased by 2. If the C flag is "0", the program counter is increased by 2 and the BCS instruction has no effect.

PC ← PC+2+rr if C=1

PC ← PC+2 if C=0

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BCS␣hhll	B0, rr	2	2

NOTE: rr=hhll−(PC+2)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

BEQ (Branch on Equal)

Function: If the Z flag is "1", the CPU branches to the specified relative address. The number of cycles given in the table below is increased by 2. If the Z flag is "0", the program counter is increased by 2 and the BCS instruction has no effect.

PC ← PC+2+rr if Z=1

PC ← PC+2 if Z=0

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BEQ␣hhll	F0, rr	2	2

NOTE: rr=hhll−(PC+2)

−128≦rr≦127

Flags:

Status Register							
N	V	T	B	D	I	Z	C
−	−	0	−	−	−	−	−

BIT (Bit Test)

Function: An AND operation is performed between the accumulator and memory. The result is not stored. Memory bit 7 is saved in the negative flag and bit 6 in the overflow flag. A∧M

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣BIT␣#nn	89, nn	2	2
ZP	␣BIT␣ZZ	24, ZZ	2	4
ZP, X	␣BIT␣ZZ, X	34, ZZ	2	4
ABS	␣BIT␣hhll	2C, ll, hh	3	5
ABS, X	␣BIT␣hhll, X	3C, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
M_7	M_6	0	—	—	—	Z	—

BMI (Branch on Minus)

Function: If the N flag is "1", the CPU branches to the specified relative address. The number of cycles given in the table below is increased by 2. If the N flag is "0", the program counter is increased by 2 and the BCS instruction has no effect.

PC ← PC+2+rr if N=1

PC ← PC+2 if N=0

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BMI␣hhll	30, rr	2	2

NOTE: rr=hhll−(PC+2)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

BNE (Branch on Not Equal)

Function: If the Z flag is "0", the CPU branches to the specified relative address. The number of cycles given in the table below is increased by 2. If the Z flag is "1", the program counter is increased by 2 and the BNE instruction has no effect.

PC ← PC+2+rr if Z=0

PC ← PC+2 if Z=1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BNE␣hhll	D0, rr	2	2

NOTE: rr=hhll−(PC+2)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

BPL (Branch on Plus)

Function: If the N flag is "0", the CPU branches to the specified relative address. The number of cycles given in the table below is increased by 2. If the N flag is "1", the program counter is increased by 2 and the BPL instruction has no effect.

PC ← PC+2+rr if N=1

PC ← PC+2 if N=0

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BPL␣hhll	10, rr	2	2

NOTE: rr=hhll−(PC+2)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

BRA (Branch Always)

Function: The CPU branches to the specified relative address.

PC ← PC+2+rr

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BRA␣hhll	80, rr	2	4

NOTE: rr=hhll−(PC+2)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

BRK (Break)

Function: The program counter and the content of the status register are pushed into the stack. In this case, PCH is first pushed in, followed by PCL and P. Then the instruction reads the low-order byte at logical address FFF6(hex) and the high-order byte at logical address FFF7(hex), and the CPU branches to the interrupt handling subroutine. The B flag in the status register which is pushed into the stack is set to "1". The value of the program counter which is pushed into the stack is the address of (BRK +2).

PC ← PC+2
Ms ← PCH , S ← S − 1
Ms ← PCL , S ← S − 1
Ms ← P , S ← S − 1
PCL ← ($FFF6_{16}$)
PCH ← ($FFF7_{16}$)

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣BRK	00	1	8

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	1	0	1	—	—

BSR (Branch Subroutine)

Function: The program counter is pushed into the stack. In this case, PCH precedes PCL. The value of the program counter which is pushed into the stack is the address of the last byte of the BSR instruction.

PC ← PC+1

Ms ← PCH , S ← S − 1

Ms ← PCL , S ← S − 1

PC ← PC+2+rr

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BSR␣hhll	44, rr	2	8

NOTE: rr=hll−(PC+2)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
−	−	0	−	−	−	−	−

BVC (Branch on V Clear)

Function: If the V flag is "0" the CPU branches to the specified relative address. The number of cycles given in the table below is incresed by 2. If the V flag is "1", the program counter is increased by 2 and the BVC instruction has no effect.

PC ← PC+2+rr if V = 0

PC ← PC+2 if V=1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BVC␣hhll	50, rr	2	2

NOTE: rr=hll−(PC+2)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

BVS (Branch on V Set)

Function: If the V flag is "1", the CPU branches to the specified relative address. The number of cycles given in the tabel below is increased by 2. If the V flag is "0", the program counter is increased by 2 and the BVS instruction has no effect.

PC ← PC+2+rr if V=1

PC ← PC+2 if V=0

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
REL	␣BVS␣hhll	70, rr	2	2

NOTE: rr=hhll−(PC+2)

$-128 \leqq rr \leqq 127$

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

CLA (Clear A)

Function: The accumulator is cleared.

$A \leftarrow 00_{16}$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣CLA	62	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

CLC (Clear C)

Function: The carry flag is cleared.

$C \leftarrow 0$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣CLC	18	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	0

CLD (Clear D)

Function: The decimal flag is cleared.

D ← 0

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣CLD	D8	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	0	—	—	—

CLI (Clear I)

Function: The interrupt disable is cleared.

I ← 0

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⎵CLI	58	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	0	—	—

CLV (Clear V)

Function: The overflow flag is cleared.

$V \leftarrow 0$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣CLV	B8	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	0	0	–	–	–	–	–

CLX (Clear X)

Function: The X-register is cleared.

$X \leftarrow 00_{16}$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣CLX	82	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

CLY (Clear Y)

Function: The Y-register is cleared.

$Y \leftarrow 00_{16}$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⌴CLY	C2	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

CMP (Compare A with M)

Function: Memory is subtracted from the accumulator. The result is not stored.

A − M

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣CMP␣#nn	C9, nn	2	2
ZP	␣CMP␣ZZ	C5, ZZ	2	4
ZP, X	␣CMP␣ZZ, X	D5, ZZ	2	4
(IND)	␣CMP␣(ZZ)	D2, ZZ	2	7
(IND, X)	␣CMP␣(ZZ, X)	C1, ZZ	2	7
(IND), Y	␣CMP␣(ZZ), Y	D1, ZZ	2	7
ABS	␣CMP␣hhll	CD, ll, hh	3	5
ABS, X	␣CMP␣hhll, X	DD, ll, hh	3	5
ABS, Y	␣CMP␣hhll, Y	D9, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	−	0	−	−	−	Z	C

CPX (Compare X with M)

Function: Memory is subtracted from the X-register. The result is not stored.

X − M

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣CPX␣#nn	E0, nn	2	2
ZP	␣CPX␣ZZ	E4, ZZ	2	4
ABS	␣CPX␣hhll	EC, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	C

CPY (Compare Y with M)

Function: Memory is subtracted from the Y-register. The result is not stored.

Y − M

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣CPY␣#nn	C0, nn	2	2
ZP	␣CPY␣ZZ	C4, ZZ	2	4
ABS	␣CPY␣hhll	CC, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	C

DEC (Decrement)

Function: The content of memory or the accumulator is decremented by 1.

$M \leftarrow M - 1$

or

$A \leftarrow A - 1$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣DEC␣ZZ	C6, ZZ	2	6
ZP, X	␣DEC␣ZZ, X	D6, ZZ	2	6
ABS	␣DEC␣hhll	CE, ll, hh	3	7
ABS, X	␣DEC␣hhll, X	DE, ll, hh	3	7
ACC	␣DEC␣A	3A	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	–

DEX (Decrement X)

Function: The content of the Y-register is decremented by 1.

$Y \leftarrow X - 1$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⎵DEX	CA	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	–

DEY (Decrement Y)

Function: The content of the Y-register is decremented by 1.

$Y \leftarrow Y - 1$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⌴DEY	88	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

EOR (Exclusive OR)

Function: The EOR instruction operates in either of two different ways depending on the T flag.

i) When T=1 (a SET instruction was executed immediately before the EOR)
M(x) is exclusive-ORed with M and the result is stored in M(x). The number of cycles given in the table below is increased by 3.
M(x) ← M(x)∀M

ii) When T=0 (a SET instruction was not executed immediately before the EOR)
A is exclusive-ORed with M and the result is stored in A.
A ← A∀M

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣EOR␣#nn	49, nn	2	2
ZP	␣EOR␣ZZ	45, ZZ	2	4
ZP, X	␣EOR␣ZZ, X	55, ZZ	2	4
(IND)	␣EOR␣(ZZ)	52, ZZ	2	7
(IND, X)	␣EOR␣(ZZ, X)	41, ZZ	2	7
(IND), Y	␣EOR␣(ZZ), Y	51, ZZ	2	7
ABS	␣EOR␣hhll	4D, ll, hh	3	5
ABS, X	␣AND␣hhll, X	5D, ll, hh	3	5
ABS, Y	␣EOR␣hhll, Y	59, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

INC (Increment)

Function: The content of memory or the accumulator is incremented by 1.

$M \leftarrow X + 1$

or

$A \leftarrow A + 1$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣INC␣ZZ	E6, ZZ	2	6
ZP, X	␣INC␣ZZ, X	F6, ZZ	2	6
ABS	␣INC␣hhll	EE, ll, hh	3	7
ABS, X	␣INC␣hhll, X	FE, ll, hh	3	7
ACC	␣INC␣A	1A	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	–

INX (Increment X)

Function: The content of the X-register is incremented by 1.

$X \leftarrow X + 1$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣INX	E8	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

INY (Increment Y)

Function: The content of the Y-register is incremented by 1.

$Y \leftarrow Y + 1$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⎵INY	C8	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

JMP (Jump to New Location)

Function: The CPU branches to the specified address.

i) Addressing mode: ABS

PCL ← ll

PCH ← hh

ii) Addressing mode: (ABS)

PCL ← (hhll)

PCH ← (hhll + 1)

iii) Addressing mode: (ABS, X)

PCL ← (hhll + X)

PCH ← (hhll + X + 1)

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ABS	␣JMP␣hhll	4C, ll, hh	3	4
(ABS)	␣JMP␣(hhll)	6C, ll, hh	3	7
(ABS, X)	␣JMP␣(hhll, X)	7C, ll, hh	3	7

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

JSR (Jump to Subroutine)

Function: The program counter is pushed into the stack. In this case, PCH precedes PCL. Next, the CPU branches to the specified address. The value of the program counter which is pushed into the stack is the address of the last byte of the JSR instruction.

PC ← PC+2
Ms ← PCH , S ← S − 1
Ms ← PCL , S ← S − 1
PC ← hhll

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ABS	␣JSR␣hhll	20, ll, hh	3	7

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

LDA (Load A)

Function: The content of memory is loaded to the accumulator.

A ← M

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣LDA␣#nn	A9, nn	2	2
ZP	␣LDA␣ZZ	A5, ZZ	2	4
ZP, X	␣LDA␣ZZ, X	B5, ZZ	2	4
(IND)	␣LDA␣(ZZ)	B2, ZZ	2	7
(IND, X)	␣LDA␣(ZZ, X)	A1, ZZ	2	7
(IND), Y	␣LDA␣(ZZ), Y	B1, ZZ	2	7
ABS	␣LDA␣hhll	AD, ll, hh	3	5
ABS, X	␣LDA␣hhll, X	BD, ll, hh	3	5
ABS, Y	␣LDA␣hhll, Y	B9, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	–

LDX (Load X)

Function: The content of memory is loaded to the X-register.

$X \leftarrow M$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣LDX␣#nn	A2, nn	2	2
ZP	␣LDX␣ZZ	A6, ZZ	2	4
ZP, Y	␣LDX␣ZZ, Y	B6, ZZ	2	4
ABS	␣LDX␣hhll	AE, ll, hh	3	5
ABS, Y	␣LDX␣hhll, Y	BE, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	–

LDY (Load Y)

Function: The content of memory is loaded to the Y-register.

$Y \leftarrow M$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣LDY␣#nn	A0, nn	2	2
ZP	␣LDY␣ZZ	A4, ZZ	2	4
ZP, X	␣LDY␣ZZ, X	B4, ZZ	2	4
ABS	␣LDY␣hhll	AC, ll, hh	3	5
ABS, X	␣LDY␣hhll, X	BC, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	–

LSR (Logical Shift Right)

Function: The content of memory or the accumulator is shifted right by one bit. 0 is set in M7 or A7, and M0 or A0 in C.

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣LSR␣ZZ	46, ZZ	2	6
ZP, X	␣LSR␣ZZ, X	56, ZZ	2	6
ABS	␣LSR␣hhll	4E, ll, hh	3	7
ABS, X	␣LSR␣hhll, X	5E, ll, hh	3	7
ACC	␣LSR␣A	4A	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	C

NOP (No Operation)

Function: The program counter is incremented by 1.

$PC \leftarrow PC + 1$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣NOP	EA	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

ORA (OR)

Function: The ORA instruction operates in either of two different ways depending on the T flag.

i) When T=1 (a SET instruction was executed immediately before the ORA)
M(x) is ORed with M and the result is stored in M(x). The number of cycles given in the table below is increased by 3.
M(x) ← M(x)∨M

ii) When T=0 (a SET instruction was not executed immediately before the ORA)
A is ORed with M and the result is stored in A.
A ← A∨M

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣ORA␣#nn	09, nn	2	2
ZP	␣ORA␣ZZ	05, ZZ	2	4
ZP, X	␣ORA␣ZZ, X	15, ZZ	2	4
(IND)	␣ORA␣(ZZ)	12, ZZ	2	7
(IND, X)	␣ORA␣(ZZ, X)	01, ZZ	2	7
(IND), Y	␣ORA␣(ZZ), Y	11, ZZ	2	7
ABS	␣ORA␣hhll	0D, ll, hh	3	5
ABS, X	␣ORA␣hhll, X	1D, ll, hh	3	5
ABS, Y	␣ORA␣hhll, Y	19, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

PHA (Push A)

Function: The content of the accumulator is pushed into stack.

Ms ← A

S ← S − 1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⌴PHA	48	1	3

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

PHP (Push P)

Function: The content of the status register is pushed into stack.

Ms ← P

S ← S − 1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣PHP	08	1	3

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

PHX (Push X)

Function: The content of the X-register is pushed into stack.

Ms ← X

S ← S − 1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣PHX	DA	1	3

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

PHY (Push Y)

Function: The content of the Y-register is pushed into stack.

Ms ← Y

S ← S − 1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣PHY	5A	1	3

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

PLA (Pull A)

Function: The data in the stack is pulled to the accumulator.

$S \leftarrow S + 1$

$A \leftarrow Ms$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⌴PLA	68	1	4

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

PLP (Pull P)

Function: The data in the stack is pulled to the status register.

S ← S + 1

P ← Ms

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣PLP	28	1	4

Flags:

Status Register							
N	V	T	B	D	I	Z	C
(R	E	S	T	O	R	E	D)

PLX (Pull X)

Function: The data in the stack is pulled to the X-register.

$S \leftarrow S + 1$

$X \leftarrow Ms$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣PLX	FA	1	4

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

PLY (Pull Y)

Function: The data in the stack is pulled to the Y-register.

$S \leftarrow S + 1$

$Y \leftarrow Ms$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣PLY	7A	1	4

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

RMBi (Reset Memory Bit)

Function: The specified bit of memory in the zero page is reset.

Mi ← 0

Instruction:

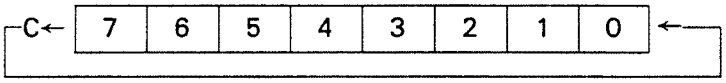
Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣RMBi␣ZZ	10i+7, ZZ	2	7

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

ROL (Rotate Left)

Function: The content of memory or the accumulator, concatenated with the carry flag is rotated left by one bit.



Instruction:

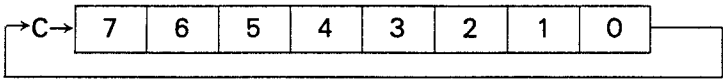
Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣ROL␣ZZ	26, ZZ	2	6
ZP, X	␣ROL␣ZZ, X	36, ZZ	2	6
ABS	␣ROL␣hhll	2E, ll, hh	3	7
ABS, X	␣ROL␣hhll, X	3E, ll, hh	3	7
ACC	␣ROL␣A	2A	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	C

ROR (Rotate Right)

Function: The content of memory or the accumulator, concatenated with the carry flag is rotated right by one bit.



Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣ROR␣ZZ	66, ZZ	2	6
ZP, X	␣ROR␣ZZ, X	76, ZZ	2	6
ABS	␣ROR␣hhll	6E, ll, hh	3	7
ABS, X	␣ROR␣hhll, X	7E, ll, hh	3	7
ACC	␣ROR␣A	6A	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	C

RTI (Return from Interrupt)

Function: The data in the stack is pulled. In this case, the status register is first pulled, followed by the low-order byte and the high-order byte of the program counter. The CPU branches to the address specified by the program counter.

S ← S + 1 , P ← Ms
S ← S + 1 , PCL ← Ms
S ← S + 1 , PCH ← Ms

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣RTI	40	1	7

Flags:

Status Register							
N	V	T	B	D	I	Z	C
(R	E	S	T	O	R	E	D)

RTS (Return from Subrutione)

Function: The data in the stack is pulled. In this case, the low-order byte of the program counter is first pulled, followed by the high-order byte. Then the program counter is incremented by 1, and the CPU branches to the address specified by the program couter.

S ← S + 1 , PCL ← Ms

S ← S + 1 , PCH ← Ms

PC ← PC + 1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣RTS	60	1	7

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

SAX (Swap A forX)

Function: The content of accumulator is swapped for the content of the the X-register.

$A \leftrightarrow X$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣SAX	22	1	3

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

SAY (Swap A for Y)

Function: The content of accumulator is swapped for the content of the the Y-register.

A ⟷ Y

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣SAY	42	1	3

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

SBC (Subtract with Carry)

Function: M and $\overline{C}$ are subtracted from A. The result is stored in A. The SBC instruction operates in either of two different ways depending on the D flag.

$A \leftarrow A - M - \overline{C}$

i) When D = 1
A decimal subtract operation is performed. The number of cycles given in the table below is increased by 1. The V flag is unaffected.

ii) When D = 0
A binary subtract operation is performed.

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣SBC␣#nn	E9, nn	2	2
ZP	␣SBC␣ZZ	E5, ZZ	2	4
ZP, X	␣SBC␣ZZ, X	F5, ZZ	2	4
(IND)	␣SBC␣(ZZ)	F2, ZZ	2	7
(IND, X)	␣SBC␣(ZZ, X)	E1, ZZ	2	7
(IND), Y	␣BSC␣(ZZ), Y	F1, ZZ	2	7
ABS	␣SBC␣hhll	ED, ll, hh	3	5
ABS, X	␣SBC␣hhll, X	FD, ll, hh	3	5
ABS, Y	␣SBC␣hhll, Y	F9, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	V	0	—	—	—	Z	C

SEC (Set C)

Function: The carry flag is set.

C ← 1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣SEC	38	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	1

SED (Set D)

Function: The decimal flag is set.

D ← 1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣SED	F8	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	1	–	–	–

SEI (Set I)

Function: The interrupt disable is set.

$I \leftarrow 1$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⌴SEI	78	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	1	–	–

SET (Set T)

Function: The memory operation flage is set.

$T \leftarrow 1$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⌴SET	F4	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	1	–	–	–	–	–

SMBi (Set Memory Bit)

Function: The specified bit of memory in the zero page is set.

Mi ← 1

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣SMBi␣ZZ	10i +87, ZZ	2	7

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

ST0 (Store HuC6270 No. 1)

Function: Immediate data is transferred to the HuC6270. In this execution cycle, the signal levels are:

$\overline{CE7}$ = "L" level
A1 = "L" level
A0 = "L" level

HuC6270: (A1, A0) = (0, 0) ← IM

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣ST0␣#nn	03, nn	2	4

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

ST1 (Store HuC6270 No. 2)

Function: Immediate data is transferred to the HuC6270. In this execution cycle, the signal levels are:

$\overline{CE7}$ = "L" level
A1 = "H" level
A0 = "L" level

HuC6270: (A1, A0) = (1, 0) ← IM

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣ST1␣#nn	13, nn	2	4

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

ST2 (Store HuC6270 No. 3)

Function: Immediate data is transferred to the HuC6270. In this execution cycle, the signal levels are:

$\overline{CE7}$ = "L" level
A1 = "H" level
A0 = "H" level

HuC6270: (A1, A0) = (1, 1) ← IM

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM	␣ST2␣#nn	23, nn	2	4

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

STA (Store A)

Function: The content of the accumulator is stored in memory.

M ← A

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣STA␣ZZ	85, ZZ	2	4
ZP, X	␣STA␣ZZ, X	95, ZZ	2	4
(IND)	␣STA␣(ZZ)	92, ZZ	2	7
(IND, X)	␣STA␣(ZZ, X)	81, ZZ	2	7
(IND), Y	␣STA␣(ZZ), Y	91, ZZ	2	7
ABS	␣STA␣hhll	8D, ll, hh	3	5
ABS, X	␣STA␣hhll, X	9D, ll, hh	3	5
ABS, Y	␣STA␣hhll, Y	99, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

STX (Store X)

Function: The content of the X-register is stored in memory.

$M \leftarrow X$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣STX␣ZZ	86, ZZ	2	4
ZP, Y	␣STX␣ZZ, Y	96, ZZ	2	4
ABS	␣STX␣hhll	8E, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

STY (Store Y)

Function: The content of the Y-register is stored in memory.

M ← Y

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣STY␣ZZ	84, ZZ	2	4
ZP, X	␣STY␣ZZ, X	94, ZZ	2	4
ABS	␣STY␣hhll	8C, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

STZ (Store Zero)

Function: "00_{16}" is stored in memory.

$M \leftarrow 00_{16}$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣STZ␣ZZ	64, ZZ	2	4
ZP, X	␣STZ␣ZZ, X	74, ZZ	2	4
ABS	␣STZ␣hhll	9C, ll, hh	3	5
ABS,X	␣STZ␣hhll,X	9E, ll, hh	3	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

SXY (Swap X for Y)

Function: The content of the X-register is swapped for the content of the Y-register.

$X \leftrightarrow Y$

Instruction:

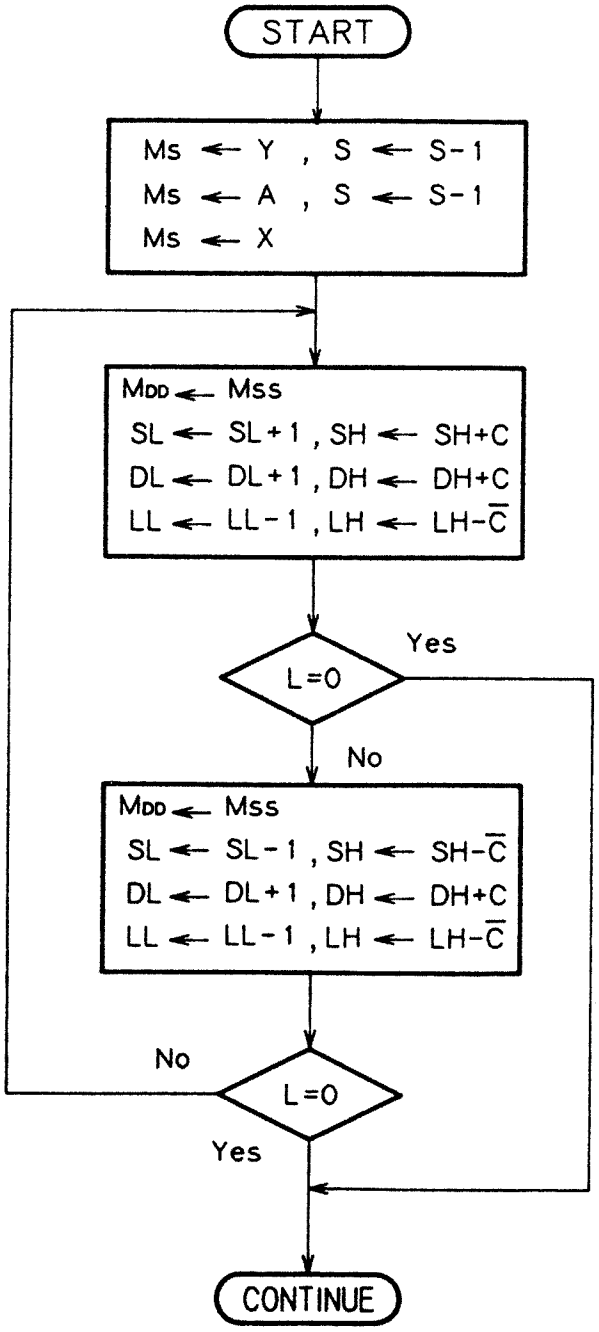
Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣SXY	02	1	3

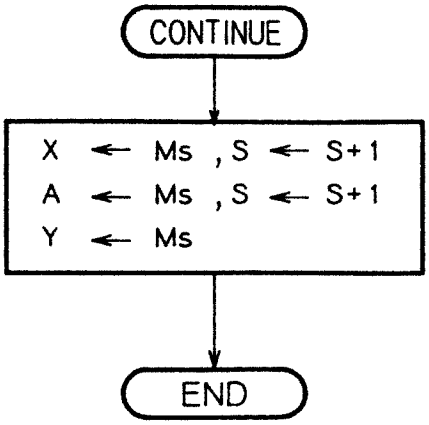
Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

TAI (Transfer Brock Data)

Function: Data is consecutively transferred from the source memory to the destination memory. The source memory address is incremented and decremented alternately each time one byte is sent. The destination memory address is incremented each time one byte is received. The number of bytes of the data to be transferred is specified by the 'length'. If the 'length' is 0, 65,536 bytes of data are transferred.
When executed, the TAI instruction uses three levels of stack for retention of the data in the internal registers (A, X, and Y).





Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣TAI␣SHSL, DHDL, LHLL	F3, SL, SH, DL, DH, LL, LH	7	17+6x

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

TAMi (Transfer A to MPR)

Function: The content of the accumulator is transferred to a mapping register whose number is specified by 'i' (=0~7).
MPRi ← A

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⎵TAMi	53, 2^i	2	5

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

TAX (Transfer A to X)

Function: The content of the accumulator is transferred to the X-register.

$X \leftarrow A$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⎵TAX	AA	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

TAY (Transfer A to Y)

Function: The content of the accumulator is transferred to the Y-register.

$Y \leftarrow A$

Instruction:

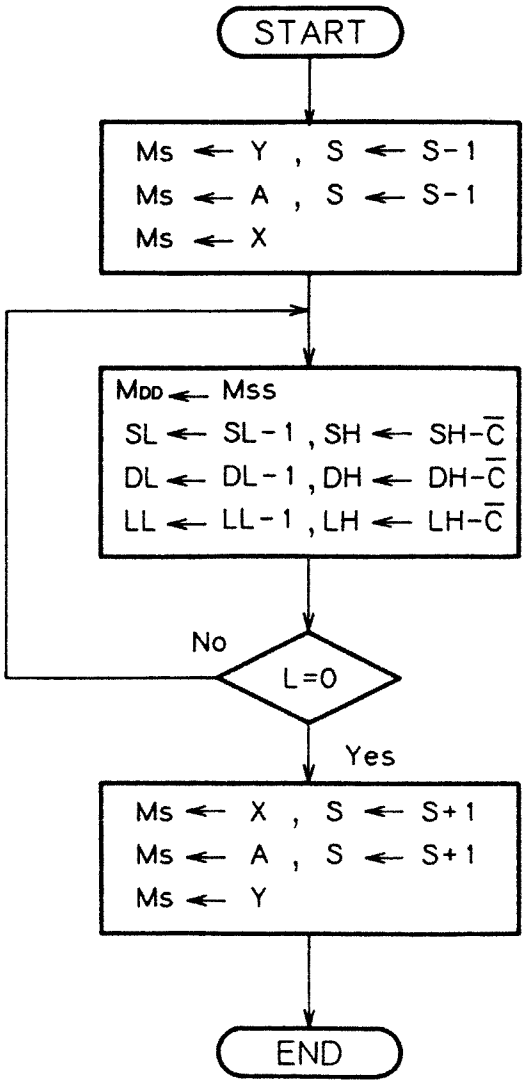
Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣TAY	A8	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	–

TDD (Transfer Block Data)

Function: Data is consecutively transferred from the source memory to the destination memory. The source memory address is incremented each time one byte is sent. The destination memory address is also decremented each time one byte is received. The number of bytes of the data to be transferred is specified by the 'length'. If the 'length' is 0, 65,536 bytes of data are transferred.
When executed, the TDD instruction uses three levels of stack for retention of the data in the internal registers (A, X, and Y).



Instruction:

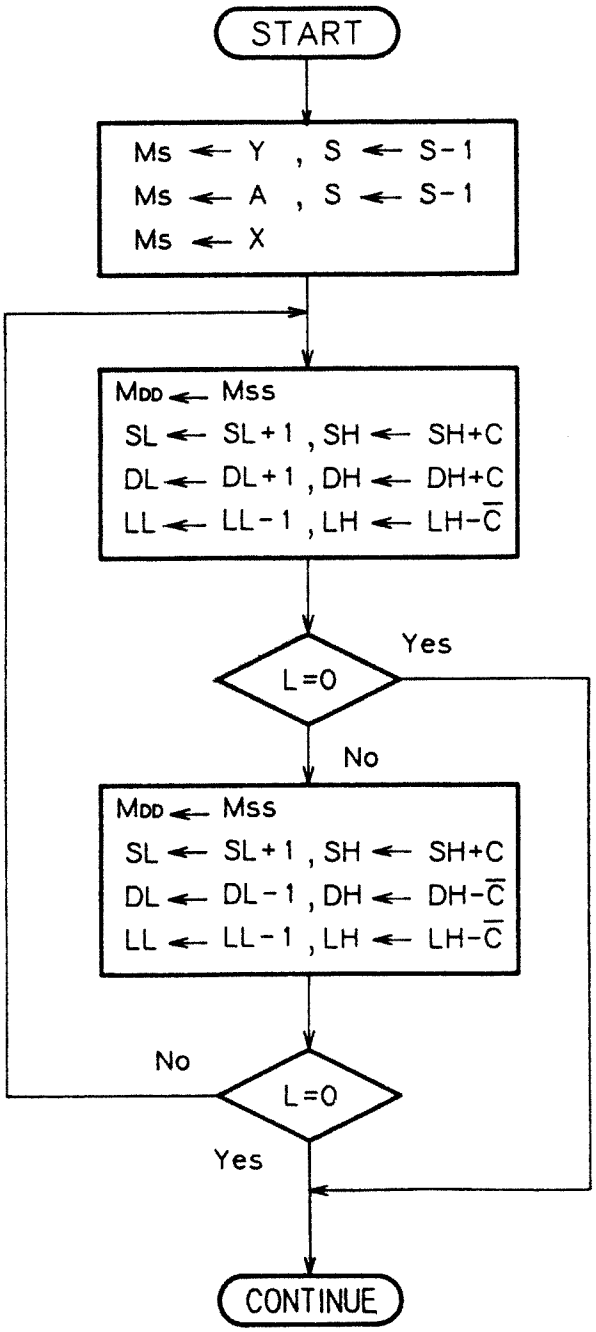
Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⌴TDD⌴SHSL, DHDL, LHLL	C3, SL, SH, DL, DH, LL, LH	7	17+6X

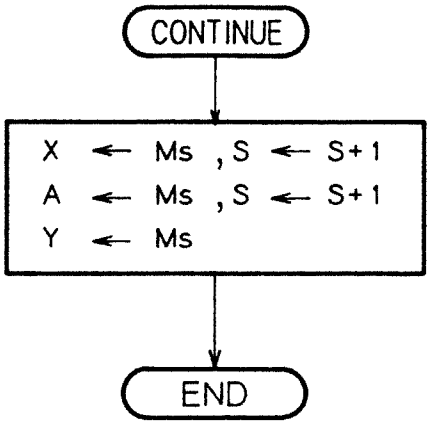
Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

TIA (Transfer Block Data)

Function: Data is consecutively transferred from the source memory to the destination memory. The source memory address is incremented each time one byte is sent. The destination memory address is incremented and decremented alternately each time one byte is received. The number of bytes of the data to be transferred is specified by the 'length'. If the 'length' is 0, 65,536 bytes of data are transferred.
When executed, the TIA instruction uses three levels of stack for retention of the data in the internal registers (A, X, and Y).





Instruction:

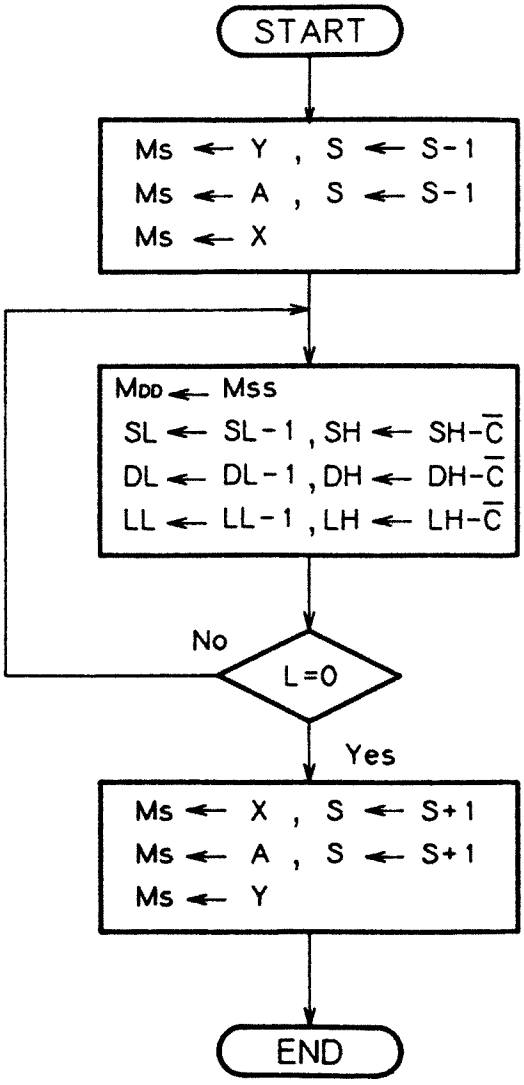
Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣TAI␣SHSL, DHDL, LHLL	E3, SL, SH, DL, DH, LL, LH	7	17+6x

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

TII (Transfer Block Data)

Function: Data is consecutively transferred from the source memory to the destination memory. The source memory address is incremented each time one byte is sent. The destination memory address is incremented and decremented alternately each time one byte is received. The number of bytes of the data to be transferred is specified by the 'length'. If the 'length' is 0, 65.536 bytes of data are transferred.
When executed, the TI I instruction uses three levels of stack for retention of the data in the intenal registers (A, X, and Y).



Instruction:

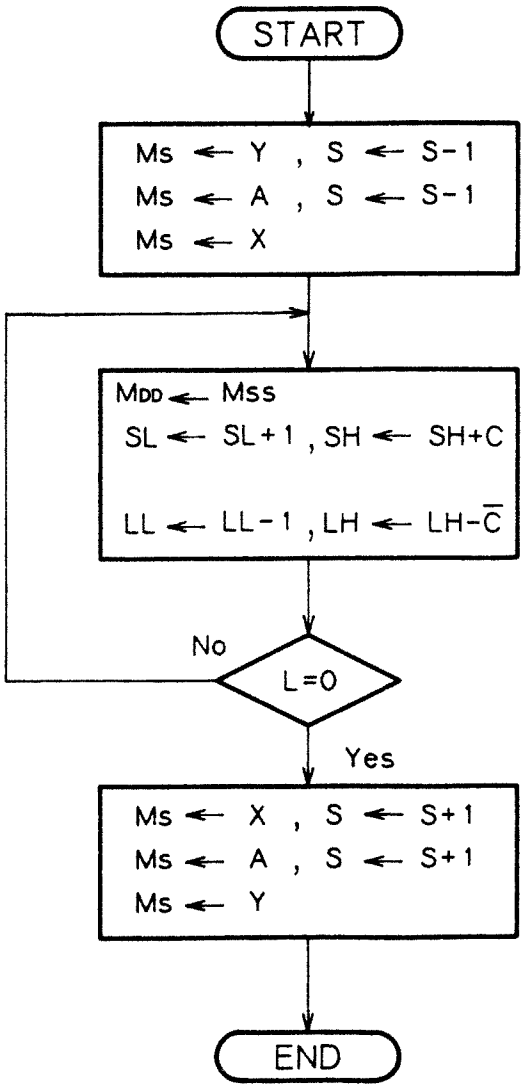
Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣TII␣SHSL, DHDL, LHLL	73, SL, SH, DL, DH, LL, LH	7	17+6X

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

TIN (Transfer Block Data)

Function: Data is consecutively transferred from the source memory to the destination memory. The source memory address is incremented each time one byte is sent. The destination memory address is fixed. The number of bytes of the data to be transferred is specified by the 'length'. If the 'length' is 0, 65,536 bytes of data are transferred. When executed, the TIN instruction uses three levels of stack for retention of the data in the internal registers (A, X, and Y).



Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣TIN␣SHSL, DHDL, LHLL	D3, SL, SH, DL, DH, LL, LH	7	17+6X

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

TMAi (Transfer MPR to A)

Function: The contents of a mapping register whose number is specified by 'i' (=0~7) is transferred the accumulator.

A ← MPRi

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣TMAi	43, 2^i	2	4

Flags:

Status Register							
N	V	T	B	D	I	Z	C
–	–	0	–	–	–	–	–

TRB (Test and Reset Bit)

Function: The negated data of the accumulator is ANDed with the content of memory. The result is not stored in the memory. Memory bit 7 is saved in the negative flag and bit 6 in the overflow flag.

$M \leftarrow \overline{A} \wedge M$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣TRB␣ZZ	14, ZZ	2	6
ABS	␣TRB␣hhll	1C, ll, hh	3	7

Flags:

Status Register							
N	V	T	B	D	I	Z	C
M_7	M_6	0	—	—	—	Z	—

TSB (Test and Set Bit)

Function: The content of the accumulator is ORed with the content of memory The result is stored in the memory. Memory bit 7 is saved in the negative flag and bit 6 in the overflow flag.
M ← A∨M

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
ZP	␣TSB␣ZZ	04, ZZ	2	6
ABS	␣TSB␣hhll	0C, ll, hh	3	7

Flags:

Status Register							
N	V	T	B	D	I	Z	C
M_7	M_6	0	—	—	—	Z	—

TST (Test Memory)

Function: The content of memory in the zero page is ANDed with immediate data. The result is not stored. Memory bit 7 is saved in the negative flag and bit 6 in the overflow flag.
A∧M

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMM ZP	␣TST␣#nn,ZZ	83, nn, ZZ	3	7
IMM ZP,X	␣TST␣#nn,ZZ,X	A3, nn, ZZ	3	7
IMM ABS	␣TST␣#nn,hhll	93, nn, ll, hh	4	8
IMM ABS,X	␣TST␣#nn,hhll,X	B3, nn, ll, hh	4	8

Flags:

Status Register							
N	V	T	B	D	I	Z	C
M_7	M_6	0	—	—	—	Z	—

TSX (Transfer S to X)

Function: The content of the stack pointer is transferred to the X-register.

$X \leftarrow S$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣TSX	BA	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	—	0	—	—	—	Z	—

TXA (Transfer X to A)

Function: The content of the X-register is transferred to the accumulator.

$A \leftarrow X$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⎵TXA	8A	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	–

TXS (Transfer X to S)

Function: The content of the X-register is transferred to the stack pointer.

S ← X

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	⌴TXS	9A	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
—	—	0	—	—	—	—	—

TYA (Transfer Y to A)

Function: The content of the Y-register is transferred to the accumulator.

$A \leftarrow Y$

Instruction:

Addressing	Mnemonic	Machine Code	Number of Bytes	Number of Cycles
IMPLID	␣TYA	98	1	2

Flags:

Status Register							
N	V	T	B	D	I	Z	C
N	–	0	–	–	–	Z	–