

**APPENDIX B**

**INSTRUCTION LIST**

**ALPHABETIC BY MNEMONIC**

**WITH OP CODES, EXECUTION CYCLES**

**AND MEMORY REQUIREMENTS**

The following notation applies to this summary:

A	Accumulator
X, Y	Index Registers
M	Memory
P	Processor Status Register
S	Stack Pointer
✓	Change
—	No Change
+	Add
∧	Logical AND
−	Subtract
⊕	Logical Exclusive Or
↑	Transfer from Stack
↓	Transfer to Stack
→	Transfer to
←	Transfer to
V	Logical OR
PC	Program Counter
PCH	Program Counter High
PCL	Program Counter Low
OPER	OPERAND
#	IMMEDIATE ADDRESSING MODE

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.

**ADC***Add memory to accumulator with carry***ADC**Operation:  $A + M + C \rightarrow A, C$ 

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	6D	3	4
Absolute, X	ADC Oper, X	7D	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.

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

Operation: C ← 

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

 + 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

**BCC****BCC** *Branch on Carry Clear***BCC**

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.

**BCS****BCS** *Branch on carry set***BCS**

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	BØ	2	2*

\* Add 1 if branch occurs to same page.

\* Add 2 if branch occurs to next page.

**BEQ****BEQ** *Branch on result zero***BEQ**

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	FØ	2	2*

\* Add 1 if branch occurs to same page.

\* Add 2 if branch occurs to next page.

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

Operation:  $A \wedge M, M_7 \rightarrow N, M_6 \rightarrow V$

Bit 6 and 7 are transferred to the status register.  $N \ Z \ C \ I \ D \ V$

If the result of  $A \wedge M$  is zero then  $Z = 1$ , otherwise  $M_7 \checkmark \text{ --- } M_6$

$Z = 0$

(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

**BMI****BMI** *Branch on result minus***BMI**

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 2 if branch occurs to different page.

**CMP***CMP Compare memory and accumulator***CMP**

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.

**CPX****CPX** *Compare Memory and Index X***CPX**

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

**CPY****CPY** *Compare memory and index Y***CPY**

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



## JMP

**JMP** *Jump to new location*

## JMP

Operation: (PC + 1) → PCL  
(PC + 2) → PCH

(Ref: 4.0.2)  
(Ref: 9.8.1)

N Z C I D V  
- - - - -

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

# JSR

JSR *Jump to new location saving return address*

# JSR

Operation: PC + 2 ↓, (PC + 1) → PCL

N Z C I D V

(PC + 2) → PCH

(Ref: 8.1)

— — — — —

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

# LSR

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

# LSR

Operation:  $\emptyset \rightarrow$ 

7	6	5	4	3	2	1	0
---	---	---	---	---	---	---	---

 $\rightarrow C$

N Z C I D V

$\emptyset \checkmark \checkmark - - -$

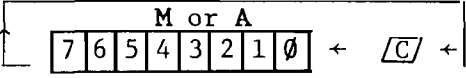
(Ref: 10.1)

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

## ROL

ROL Rotate one bit left (memory or accumulator)

## ROL

Operation:  N Z C I D V  
✓ ✓ ✓ — — —

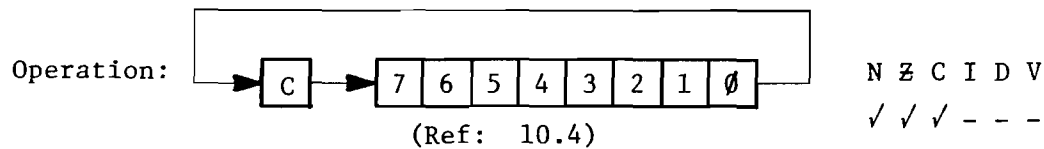
(Ref: 10.3)

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

## ROR

ROR Rotate one bit right (memory or accumulator)

## ROR



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 will be available on MCS650X micro-processors after June, 1976.

## RTI

RTI Return from interrupt

## RTI

Operation: P↑ PC↑

N Z C I D V

(Ref: 9.6)

From Stack

Addressing Mode	Assembly Language Form	OP CODE	No. Bytes	No. Cycles
Implied	RTI	40	1	6

## RTS

RTS Return from subroutine

## RTS

Operation: PC↑, PC + 1 → 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

**SBC****SBC** *Subtract memory from accumulator with borrow***SBC**Operation:  $A \leftarrow M - \overline{C} \rightarrow A$ 

N Z C I D V

Note:  $\overline{C}$  = 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.



## **APPENDIX C**

### **INSTRUCTION ADDRESSING**

### **MODES AND**

### **RELATED EXECUTION TIMES**



# INSTRUCTION ADDRESSING MODES AND RELATED EXECUTION TIMES (in clock cycles)

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect, Y)	Absolute Indirect
✓ ADC	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
✓ AND	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
ASL	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
BCC	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
BCS	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
BEQ	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
BIT	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
BMI	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
BNE	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
✓ BPL	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
✓ BRK	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
BVC	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
BVS	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
✓ CLC	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
CLD	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
CLI	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
CLV	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
✓ CMP	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
CPX	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
CPY	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
DEC	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
DEX	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
DEY	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
EOR	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
INC	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
INX	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
INY	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
JMP	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
JSR	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
LDA	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
LDX	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
LDY	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
LSR	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
NOP	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
ORA	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
PHA	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
PHP	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
PLA	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
PLP	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
ROL	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
ROR	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
RTI	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
RTS	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
SBC	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
SEC	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
SED	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
SEI	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
STA	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
STX*	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
STY**	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
TAX	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
TAY	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
TSX	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
TXA	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
TXS	2	2	3	4	4	4	4*	4*	2	2	6	5*	5*
TYA	2	2	3	4	4	4	4*	4*	2	2	6	5*	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



## **APPENDIX E**

### **SUMMARY OF ADDRESSING MODES**

This appendix is to serve the user in providing a reference for the MCS650X addressing modes. Each mode of address is shown with a symbolic illustration of the bus status at each cycle during the instruction fetch and execution. The example number as found in the text is provided for reference purposes.

### *E.1 IMPLIED ADDRESSING*

#### Example 5.3: Illustration of implied addressing

<u>Clock Cycle</u>	<u>Address Bus</u>	<u>Program Counter</u>	<u>Data Bus</u>	<u>Comments</u>
1	PC	PC + 1	OP CODE	Fetch OP CODE
2	PC + 1	PC + 1	New OP CODE	Ignore New OP CODE; Decode Old OP CODE
3	PC + 1	PC + 2	New OP CODE	Fetch New OP CODE; Execute Old OP CODE

## *E.2 IMMEDIATE ADDRESSING*

### Example 5.4: Illustration of immediate addressing

<u>Clock Cycle</u>	<u>Address Bus</u>	<u>Program Counter</u>	<u>Data Bus</u>	<u>Comments</u>
1	PC	PC + 1	OP CODE	Fetch OP CODE
2	PC + 1	PC + 2	Data	Fetch Data, Decode OP CODE
3	PC + 2	PC + 3	New OP CODE	Fetch New OP CODE, Execute Old OP CODE

## *E.3 ABSOLUTE ADDRESSING*

### Example 5.5: Illustration of absolute addressing

<u>Clock Cycle</u>	<u>Address Bus</u>	<u>Program Counter</u>	<u>Data Bus</u>	<u>Comments</u>
1	PC	PC + 1	OP CODE	Fetch OP CODE
2	PC + 1	PC + 2	ADL	Fetch ADL, Decode OP CODE
3	PC + 2	PC + 3	ADH	Fetch ADH, Retail ADL
4	ADH, ADL	PC + 3	Data	Fetch Data
5	PC + 3	PC + 4	New OP CODE	Fetch New OP CODE, Execute Old OP CODE

#### E.4 ZERO PAGE ADDRESSING

Example 5.6: Illustration of zero page addressing

<u>Clock Cycle</u>	<u>Address Bus</u>	<u>Program Counter</u>	<u>Data Bus</u>	<u>Comments</u>
1	PC	PC + 1	OP CODE	Fetch OP CODE
2	PC + 1	PC + 2	ADL	Fetch ADL, De- code OP CODE
3	00, ADL	PC + 2	Data	Fetch Data
4	PC + 2	PC + 3	New OP CODE	Fetch New OP CODE, Exe- cute Old OP CODE

#### E.5 RELATIVE ADDRESSING – (Branch Positive, no crossing of page boundaries)

Example 5.8: Illustration of relative addressing--branch positive taken, no crossing of page boundaries

<u>Cycle</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>External Operation</u>	<u>Internal Operation</u>
1	0100	OP CODE	Fetch OP CODE	Finish Previous Oper- ation, Increment Pro- gram Counter to 101
2	0101	+50	Fetch Offset	Interpret Instruction, Increment Program Counter to 102
3	0102	Next OP CODE	Fetch Next OP CODE	Check Flags, Add Rela- tive to PCL, Increment Program Counter to 103
4	0152	Next OP CODE	Fetch Next OP CODE	Transfer Results to PCL, Increment Program Counter to 153

*E.6 ABSOLUTE INDEXED ADDRESSING – (with page crossing)*

*Step 5 is deleted and the data in step 4 is valid when no page crossing occurs.*

Example 6.7: Absolute Indexed; With Page Crossing

<u>Cycle</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>External Operation</u>	<u>Internal Operation</u>
1	0100	OP CODE	Fetch OP CODE	Finish Previous Operation Increment PC to 101
2	0101	BAL	Fetch BAL	Interpret Instruction Increment PC to 102
3	0102	BAH	Fetch BAH	Add BAL + Index Increment PC to 103
4	BAH, BAL +X	Data (Ignore)	Fetch Data (Data is ignored)	Add BAH + Carry
5	BAH+1, BAL+X	Data	Fetch Data	
6	0103	Next OP CODE	Fetch Next OP CODE	Finish Operation

### *E.7 ZERO PAGE INDEXED ADDRESSING*

#### Example 6.8: Illustration of Zero Page Indexing

<u>Cycle</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>External Operation</u>	<u>Internal Operation</u>
1	0100	OP CODE	Fetch OP CODE	Finish Previous Operation
2	0101	BAL	Fetch Base Address Low (BAL)	Interpret Instruct- ion
3	00, BAL	Data (Dis- carded	Fetch Discarded Data	Add: BAL + X
4	00, BAL +X	Data	Fetch Data	
5	0102	Next OP CODE	Fetch Next OP CODE	Finish Operation



## *E.8 INDEXED INDIRECT ADDRESSING*

### Example 6.10: Illustration of Indexed Indirect Addressing

<u>Cycle</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>External Operation</u>	<u>Internal Operation</u>
1	0100	OP CODE	Fetch OP CODE	Finish Previous Operation
2	0101	BAL	Fetch BAL	Interpret Instruction
3	00,BAL	DATA (Discarded)	Fetch Discarded DATA	Add BAL + X
4	00,BAL + X	ADL	Fetch ADL	Add 1 to BAL + X
5	00,BAL + X + 1		Fetch ADH	Hold ADL
6	ADH,ADL	DATA	Fetch DATA	
7	0102	Next OP	Fetch Next OP CODE	Finish Operation

*E.9 INDIRECT INDEXED ADDRESSING (with page crossing)*

*Step 6 is deleted and the data in step 5 is valid when no page crossing occurs.*

Example 6.12: Indirect Indexed Addressing (With Page Crossing)

<u>Cycle</u>	<u>Address Bus</u>	<u>Data Bus</u>	<u>External Operation</u>	<u>Internal Operation</u>
1	0100	OP CODE	Load OP CODE	Finish Previous Operation
2	0101	IAL	Fetch IAL	Interpret Instruction
3	00, IAL	BAL	Fetch BAL	Add 1 to IAL
4	00, IAL + 1	BAH	Fetch BAH	Add BAL to Y
5	BAH, BAL + Y	DATA (Discarded)	Fetch DATA (Discarded)	Add 1 to BAH
6	BAH + 1 BAL + Y	DATA	Fetch Data	
7	0102	Next OP CODE	Fetch Next OP CODE	Finish This Operation



**APPENDIX H**

**REVIEW OF BINARY**

**AND**

**BINARY CODED DECIMAL**

**ARITHMETIC**

The microprocessor automatically takes this into account and corrects for the fact that

<u>Decimal</u>		<u>BCD</u>		<u>Hex</u>
79	=	01111001		79 = 01111001
+12	=	00010010		12 = 00010010
91	=	10010001		88 = 10001011

The only difference between Hex and BCD representation is that the microprocessor automatically adjusts for the fact that BCD does not allow for Hex values A - F during add and subtract operations.

The offset which follows a branch instruction is in signed two's complement form which means that

$$\begin{aligned}
 \$+50 &= +80 = 01010000 \\
 \text{and } \$-50 &= -80 = 10110000 \\
 \text{Proof} &= 00000000
 \end{aligned}$$

The sign for this operation is in bit 7 where an 0 equals positive and a 1 equals negative.

This bit is correct for the two's complement representation but also flags the microprocessor whether to carry or borrow from the address high byte.

The following 4 examples represent the combinations of offsets which might occur (all notations are in hexadecimal):

Example H.4.1: Forward reference, no page crossing

0105	INE
0106	+55
0107	Next CP CODE

To calculate next instruction if the branch is taken

Offset	+55	01010101
Address Low		
for next		
OP CODE	07	00000111
	5C	01011100

with no carry, giving 015C as the result.

Example H.4.2: Backward reference, no page crossing

015A	BNE
015B	-55
015C	Next OP CODE

To calculate if branch is taken,

Offset	-55 = AB = 10101011
+ Address Low for	
Next OP CODE	$\frac{+5C}{07} = \frac{5C}{07} = \frac{01011100}{00000111}$

The carry is expected because of the negative offset and is ignored, thus giving 0107 as the result.

Example H.4.3: Backward reference if page boundary crossed

0105	BNE
0106	-55
0107	Next OP CODE

To calculate if branch is taken, first calculate a low byte

Offset	-55 = AB = 10101011
Address Low for	
Next OP CODE	$\frac{07}{B2} = \frac{07}{B2} = \frac{00000111}{10110010}$

There is no carry from a negative offset; therefore, a carry must be made:

-1 =	-1 = FF = 11111111
+ Address High	$= \frac{01}{00} = \frac{01}{00} = \frac{00000001}{00000000}$

This gives 00 B2 as a result.

Example H.4.4: Forward reference across page boundary

00B0	BNE
00B1	+55
00B2	Next OP CODE

To calculate next instruction if branch is taken,

Offset	55 = 01010101
Address Low for Next	
OP CODE	B2 = 10110010
	07 = 0000111

with carry on positive number.

	+1	1 = 00000001
Address High	00 = 00000000	
	1 = 00000001	

which gives 0107.