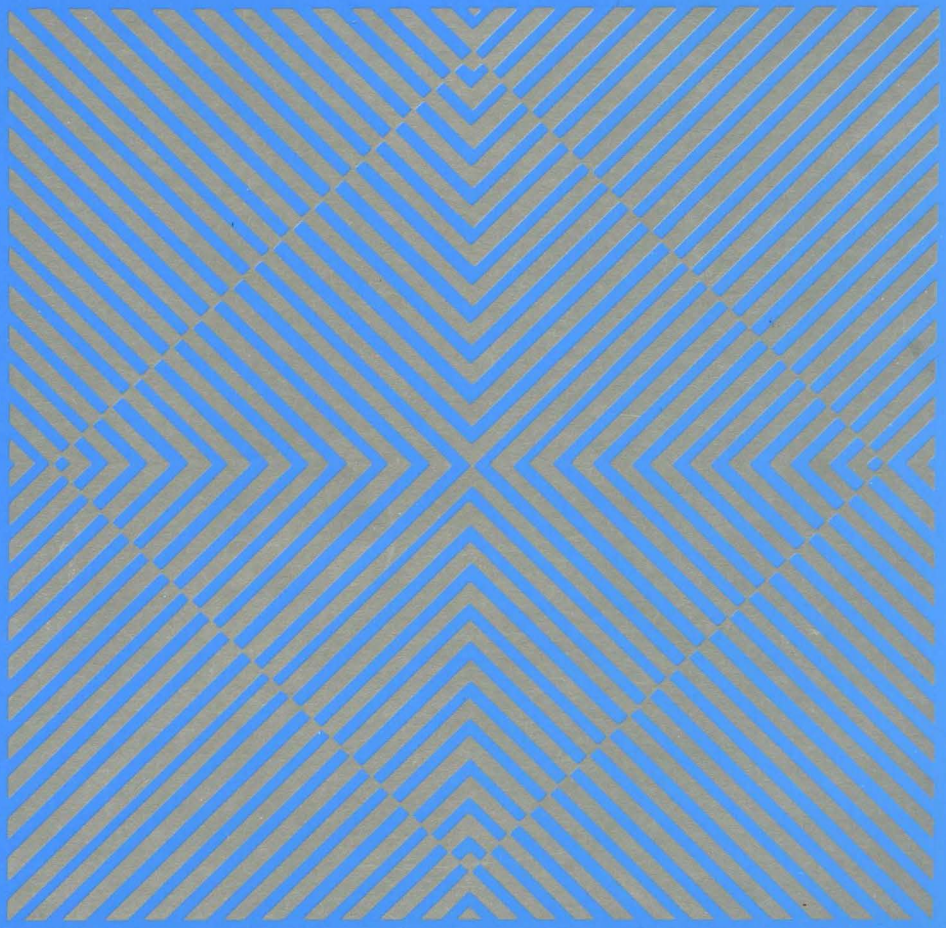
MITSUBISHI SEMICONDUCTORS

SERIES 740 USER'S MANUAL

SOFTWARE



MITSUBISHI ELECTRIC

USER'S MANUAL

SERIES 740 〈SOFTWARE〉

FOREWORD

This software manual is for users of the MELPS 740 series.
Register structure, addressing mode and instructions are introduced in each section. The enhanced instruction set with enhanced data and memory operations enable efficient programming.
Please refer to the "USER'S MANUAL" appropriate for the hardware device or the development support tools used.

CONTENTS

CONTENTS

INSTRUCTIONS

1.OVERVIEW

The software of MELPS 740 series CMOS microcomputers includes the complex bit processing functions of the 4-bit series along with byte calculations and manipulations. The software has been designed to make use of the distinctive hardware features.

The distinctive features of the MELPS 740 series are described below.

1) An efficient instruction set and many addressing modes allow the effective use of ROM.
2) The same bit management, test, and branch instructions can be performed on the accumulator, memory, or I/O area.
3) Multiple interrupts with separate interrupt vectors allow servicing of different non-periodic events.
4) Byte processing and table referencing can be easily performed using the index addressing mode.
5) Decimal mode needs no software correction for proper decimal operation.
6) The accumulator does not need to be used in operations using memory or I/O, or memory and I/O.

2. CENTRAL PROCESSING UNIT

Six main registers are built into the CPU of the MELPS 740.
The program counter (PC) is a 16-bit register but the accumulator (A), index register X (X), index register Y (Y), stack pointer (S) and processor status register (PS) are 8-bit registers.

☞ Except for the I flag, the contents of these registers are indeterminate after a hardware reset. Therefore, initialization is required with some programs.

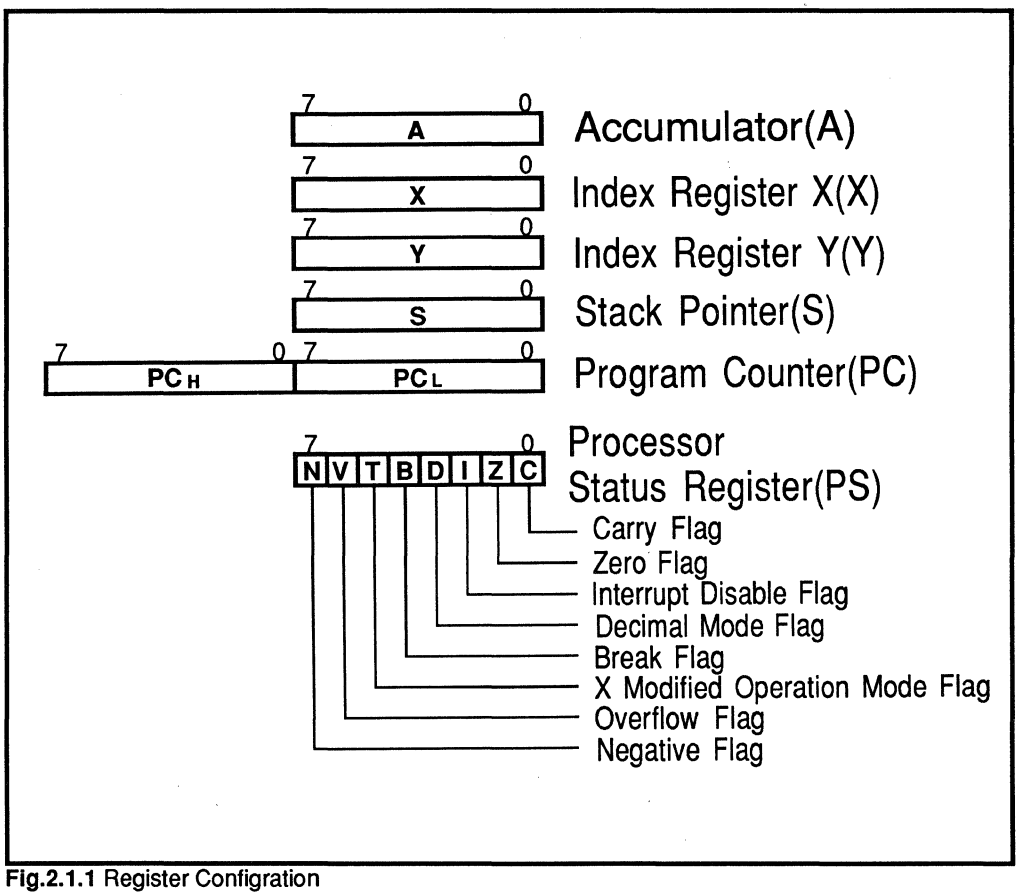


Fig.2.1.1 Register Configration

2.1 Accumulator (A)

The accumulator is the central register of the microcomputer and is an 8-bit register.
This general-purpose register is used the most frequently for arithmetic operations, data transfer, temporary memory, conditional judgments, etc.

2.2 Index Register X (X), Index Register Y (Y)

The MELPS 740 has an index register X and an index register Y, both of which are 8-bit registers.
In addressing modes which use these index registers, the register contents are added to the specified address and this becomes the actual address. These modes are extremely effective for referencing subroutine and memory tables.
The index registers also have increment, decrement, compare, and data transfer functions and can therefore be used as simple accumulators.

2.3 Stack Pointer (S)

The stack pointer is an 8-bit register used for generating interrupts and calling subroutines. When an interrupt is received, the following procedure is performed automatically in the indicated sequence.

(1) The contents of the higher 8 bits of the program counter (PCH) are saved to a zero page address using the stack pointer contents for the lower 8 address bits.
(2) The stack pointer contents are decremented by 1.
(3) The contents of the lower 8 bits of the program counter (PCL) are saved to a zero page address using the stack pointer contents for the lower 8 address bits.
(4) The stack pointer contents are decremented by 1.
(5) The contents of the processor status register (PS) are saved to a zero page address using the stack pointer contents for the lower 8 address bits.
(6) The stack pointer contents are decremented by 1.

The process status register is not saved when calling subroutines (items (5) and (6) above are not executed). The processor status register is saved by executing the PHP instruction in software.

To prevent data loss when generating interrupts and calling subroutines, it is necessary to save other registers as well. This is done by executing the proper instruction in software while in the service routine or subroutine.

For example, the PHA instruction is executed to save the contents of the accumulator. Executing the PHA instruction saves the accumulator contents to a zero page address using the stack pointer contents as the lower 8 address bits.

The RTI instruction is executed to return from an interrupt routine.

When the RTI instruction is executed, the following procedure is performed automatically in sequence.

(1) The stack pointer contents are incremented by 1.
(2) The contents of the zero page addressing the stack pointer contents as the lower 8 address bits is returned to the process status register(PS).
(3) The stack pointer contents are incremented by 1.
(4) The contents of the zero page address using the stack pointer as the lower 8 address bits is returned to the lower 8 bits of the program counter(PCL).
(5) The stack pointer contents are incremented by 1.
(6) The contents of the zero page address using the stack pointer as the lower 8 address bits is returned to the higher 8 bits of the program counter.

Steps (1) and (2) are not performed when returning from a subroutine using the RTS instruction. The processor status register should be restored before returning from a subroutine by using the PLP instruction. The accumulator should be restored before returning from a subroutine or an interrupt servicing routine by using the PLA instruction.

The PLA and PLP instructions increment the stack pointer by 1 and return to the contents of the zero page address stored in the stack pointer to the accumulator or processor status register, respectively.

☞ Saving data in the stack area gradually fills the RAM area with saved data; therefore caution must exercised concerning the depth of interrupt levels and subroutine nesting.

2.4 Program Counter (PC)

The program counter is a 16-bit counter consisting of PC_H and PC_L, which are each 8-bit. The program counter indicates the address of the next program instruction.
The MELPS 740 uses stored programs system; to start a new operation it is necessary to transfer the instruction and relevant data from memory to the CPU.
Normally the program counter is used to indicate the next memory address. After each instruction is executed, the next instruction required is read. This cycle is repeated until the program is finished.

☞ The control of the program counter of the MELPS 740 is almost fully automatic. However, caution must be exercised to avoid differences between program flow and program counter contents when using the stack pointer or directly altering the contents of the program counter.

2.5 Processor Status Register (PS)

The processor status register is an 8-bit register consisting of 5 flags which indicate the status of arithmetic operations and 3 flags which determine operation.
Each of these flags is described below. Refer to Appendix 3 "MACHINE LANGUAGE INSTRUCTION TABLE" or Chapter 3 "INSTRUCTIONS" concerning the factors which change these flags.

[Carry flag C]----------Bit 0
This flag stores any carry or borrow from the ALU after an arithmetic operation and is also changed by the Shift or Rotate instruction.
This flag is set by the SEC instruction and is cleared by the CLC instruction.

[Zero flag Z]----------Bit 1
This flag is set when the result of an arithmetic operation or data transfer is "0" and is cleared by any other result.

[Interrupt disable flag I]----------Bit 2
This flag disables interrupts when it is set to "1." This flag immediately becomes "1" when an interrupt is received. This flag is set by the SEI instruction and is cleared by the CLI instruction.

[Decimal mode flag D]----------Bit 3
This flag determines whether additions are performed in binary or decimal notation. Additions are performed in binary notation when this flag is set to "0" and as a 2-digit, 1-word decimal numeral when set to "1." Decimal notation correction is performed automatically at this time. This flag is set by the SED instruction and is cleared by the CLD instruction. Only the ADC and SBC instructions are used for decimal arithmetic operations.

[Break flag B]----------Bit 4
This flag determines whether an interrupt was generated with the BRK instruction. When a BRK instruction interrupt occurs, the flag B is set to "1"; for all other interrupts the flag is set to "0" and saved to the stack.

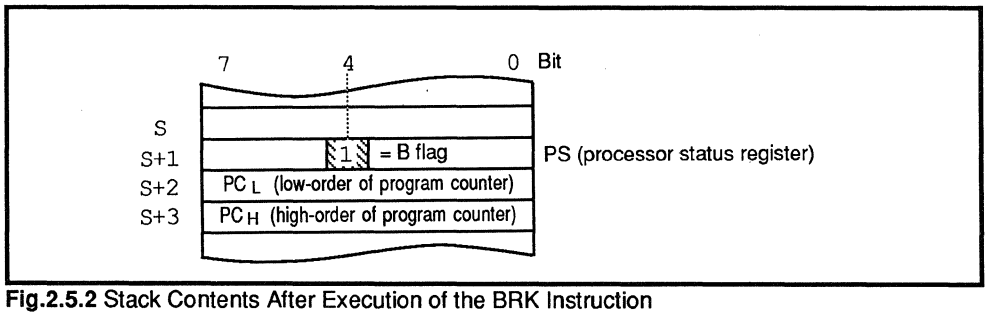


Fig.2.5.2 Stack Contents After Execution of the BRK Instruction

[X modified operation mode flag T]----------Bit 5

This flag determines whether arithmetic operations are performed via the accumulator or directly in memory. When the flag is set to "0", arithmetic operations are performed between the accumulator and memory. When "1", arithmetic operations are performed directly in memory.

This flag is set by the SET instruction and is cleared by the CLT instruction.

(1) When the T flag = 0

A ← A * M2

* : indicates an arithmetic operation

A : accumulator contents

M2: contents of the memory specified by the addressing of the arithmetic operation

(2) When the T flag = 1

M1 ← M1 * M2

* : indicates arithmetic operation

M1: memory contents specified directly by index register X

M2: contents of the memory specified by the addressing of the arithmetic operation

[Overflow flag V] ----------Bit 6

This flag is set to "1" when an overflow occurs as a result of a signed arithmetic operation. An overflow occurs when the result of an addition or subtraction exceeds +127 ($7F_{16}$) or |–128 (80_{16})|.

The CLV instruction clears the overflow flag. There is no set instruction.

The overflow flag is also set during the BIT instruction when bit 6 of the value being tested is 1.

☞ Overflows do not occur when the result of an addition or subtraction is equal to or smaller than the above numerical values, or for additions involving values with different signs.

[Negative flag N] ----------Bit 7

This flag is set to match the sign bit (bit 7) of the result of a data or arithmetic operation. This flag can be used to determine whether the results of arithmetic operations are positive or negative, and also to perform a simple bit test.

3. INSTRUCTIONS

3.1 Addressing Mode

The series MELPS 740 has 17 addressing modes and a powerful memory access capability. When extracting data required for arithmetic and logic operations from memory or when storing the results of such operations in memory, a memory address must be specified. The specification of the memory address is called addressing. The data required for addressing and the registers involved are described below. The series MELPS 740 instructions can be classified into three kinds, by the number of bytes required in program memory for the instruction: 1-byte, 2-byte and 3-byte instructions. In each case, the first byte is known as the "operation code" which forms the basis of the instruction. The second or third byte is called the "operand" which affects the addressing. The contents of index registers X and Y can also effect the addressing.

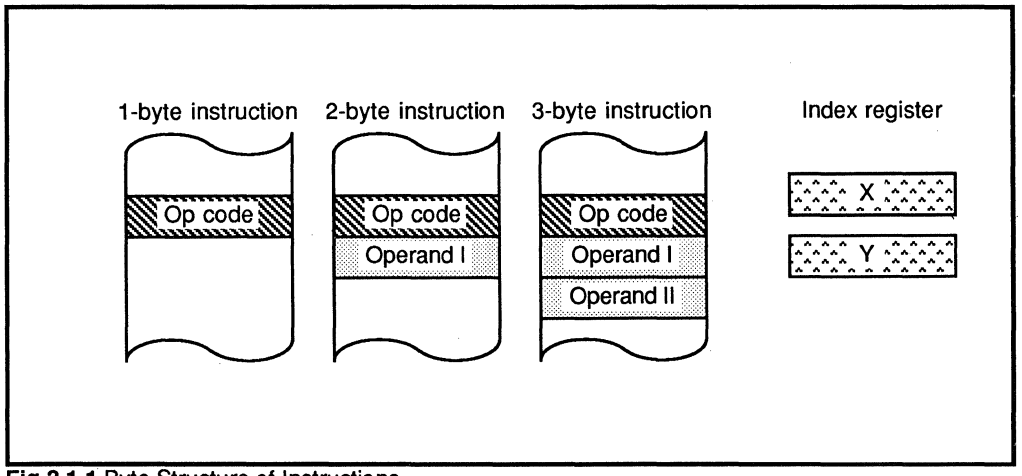


Fig.3.1.1 Byte Structure of Instructions

Although there are many addressing modes, there is always a particular memory location specified. What differs is whether the operand, or the index register contents, or a combination of both should be used to specify the memory or jump destination. Based on these 3 types of instructions, the range of variation is increased and operation is enhanced by combinations of the bit operation instructions, jump instruction, and arithmetic instructions.
Actual addressing modes are now described by type.

Immediate

Addressing mode : **Immediate**

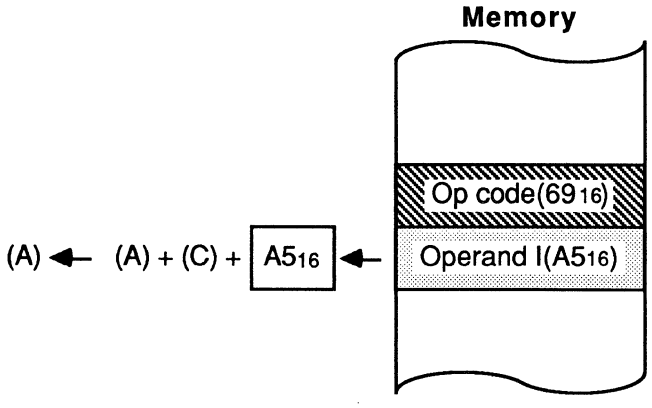
Function : **Specifies the operand as an operation data.**

Instructions : **ADC, AND, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA, SBC**

Example : Mnemonic — Machine code

ADC #$A5 — **$69_{16}$ $A5_{16}$**

This symbol(#) indicates the Immediate addressing mode.



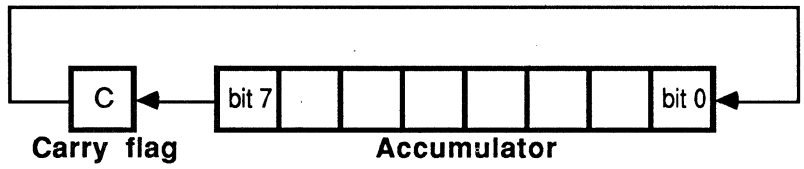
Accumulator

Addressing mode : **Accumulator**

Function : **Specifies the accumulator contents as an operation data.**

Instructions : **ASL, DEC, INC, LSR, ROL, ROR**

Example : Mnemonic **ROL A** Machine code **$2A_{16}$**



Zero Page

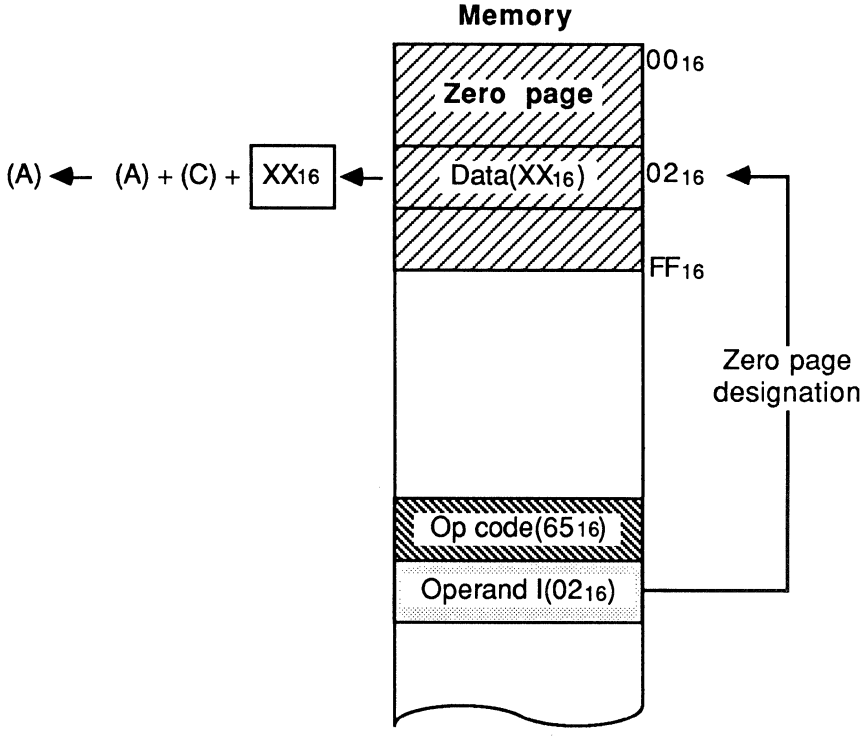
Addressing mode : **Zero page**

Function : **Specifies the Zero page memory location appointed by the operand as an operation data (00_{16} ~ FF_{16}).**

Instructions : **ADC, AND, ASL, BIT, CMP, COM, CPX, CPY, DEC, EOR, INC, LDA, LDM, LDX, LDY, LSR, ORA, ROL, ROR, RRF, SBC, STA, STX, STY, TST**

Example :

Mnemonic	Machine code
ADC $02	**65_{16} 02_{16}**



Zero page X

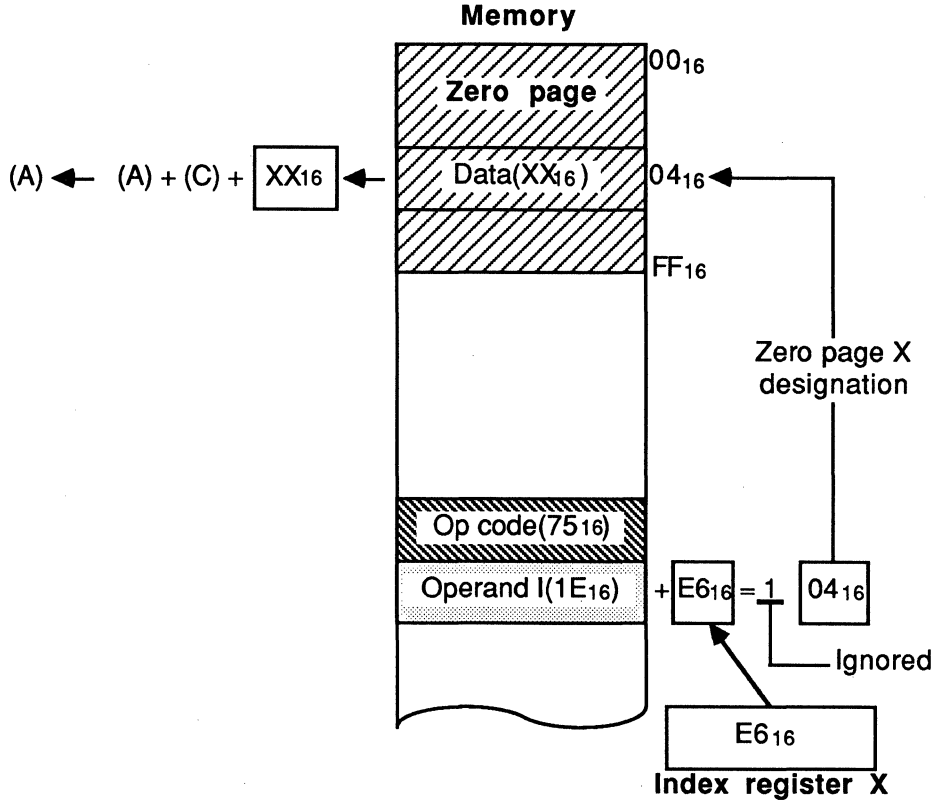
Addressing mode : **Zero page X**

Function : **Specifies the contents of the memory address in zero page consisting of the added value of operand and index register, as an operation data.**

Instructions : **ADC, AND, ASL, CMP, DEC, DIV, EOR, INC, LDA, LDY, LSR, MUL, ORA, ROL, ROR, SBC, STA, STY**

Example : Mnemonic — **ADC $1E,X**

Machine code — **75_{16} $1E_{16}$**



Zero page Y

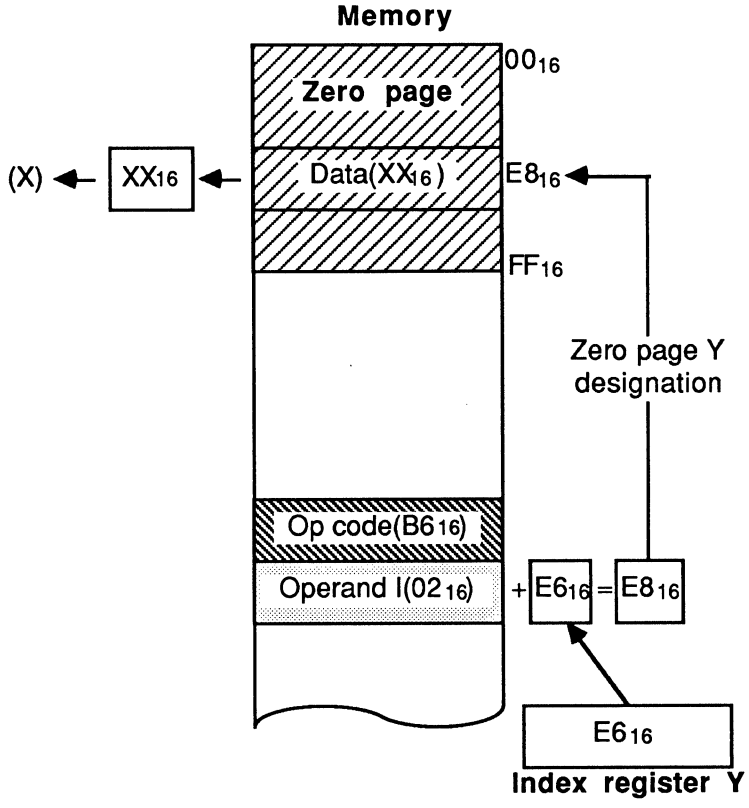
Addressing mode : **Zero page Y**

Function : **Specifies the contents of the memory address in zero page consisting of the added value of operand and index register, as an operation data.**

Instructions : **LDX, STX**

Example : Mnemonic — **LDX $02,Y**

Machine code — **$B6_{16}$ 02_{16}**



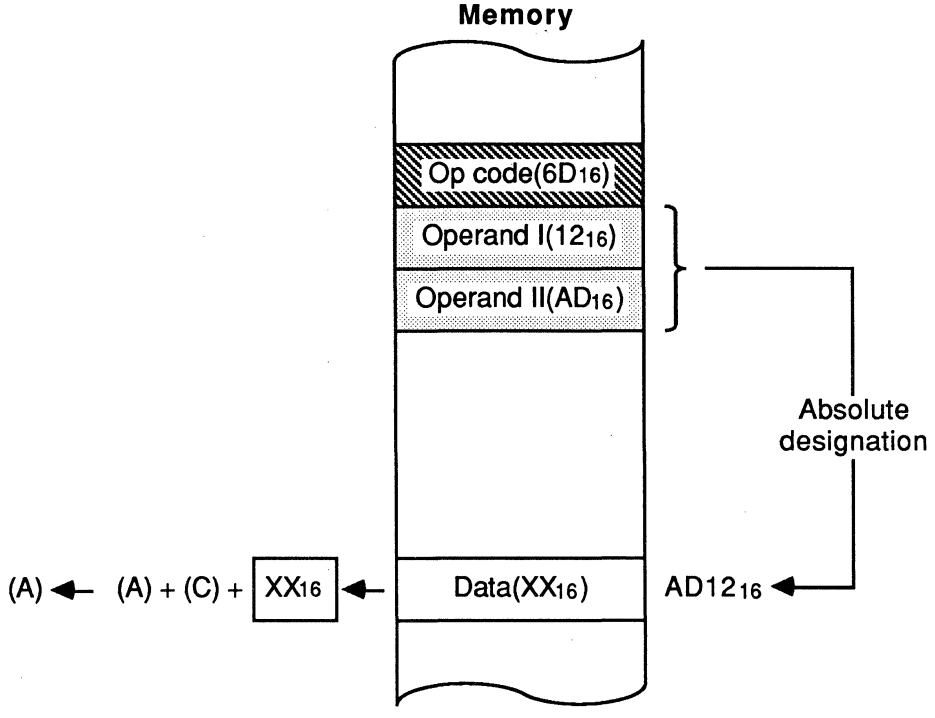
Absolute

Addressing mode : **Absolute**

Function : **Specifies the memory content appointed by operand I and II as an operation data.**

Instructions : **ADC, AND, ASL, BIT, CMP, CPX, CPY, DEC, EOR, INC, JMP, JSR, LDA, LDX, LDY, LSR, ORA, ROL, ROR, SBC, STA, STX, STY**

Example : Mnemonic **ADC $AD12** Machine code **$6D_{16}$ 12_{16} AD_{16}**



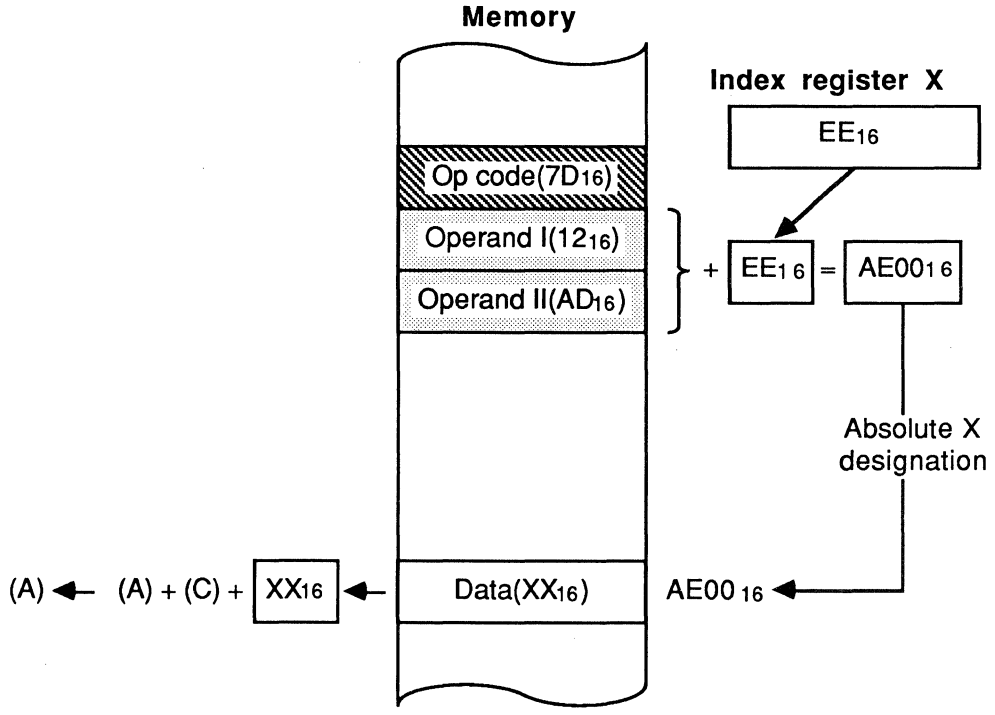
Absolute X

Addressing mode : **Absolute X**

Function : **Specifies the memory content appointed by the added value of operand I, II and register X as an operation data.**

Instructions : **ADC, AND, ASL, CMP, DEC, EOR, INC, LDA, LDY, LSR, ORA, ROL, ROR, SBC, STA**

Example : Mnemonic **ADC $AD12,X** Machine code **$7D$_{16}$ 12_{16} AD_{16}**



Absolute Y

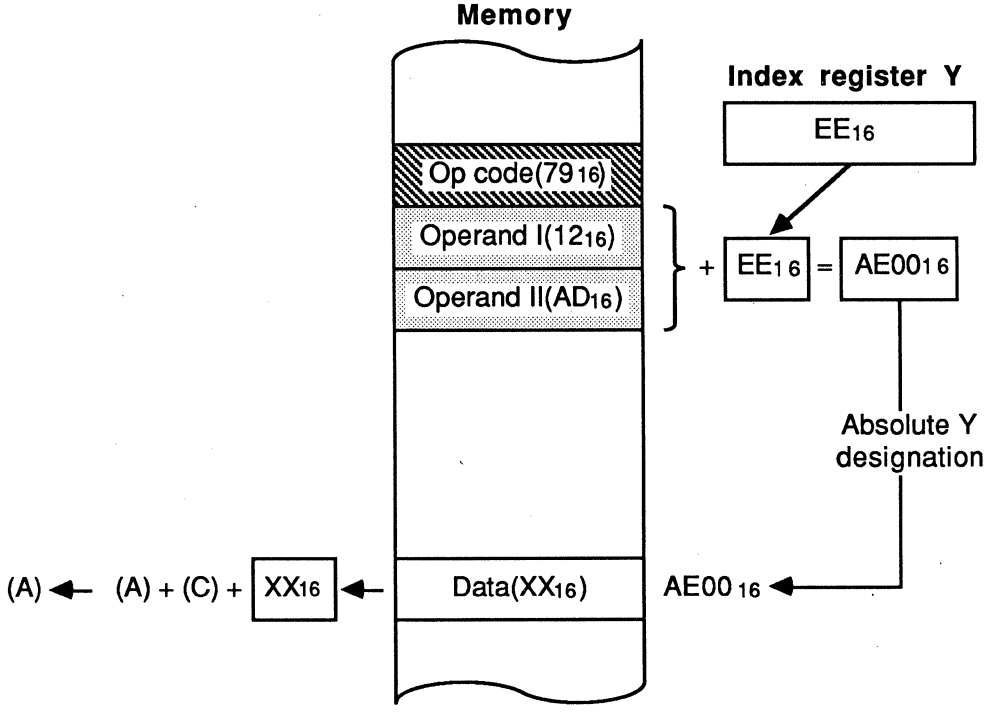
Addressing mode : **Absolute Y**

Function : **Specifies the memory content appointed by the added value of operand I, II, and register Y as data.**

Instructions : **ADC, AND, CMP, EOR, LDA, LDX, ORA, SBC, STA**

Example : Mnemonics
ADC $AD12,Y

Machine code
79_{16} 12_{16} AD_{16}



Implied

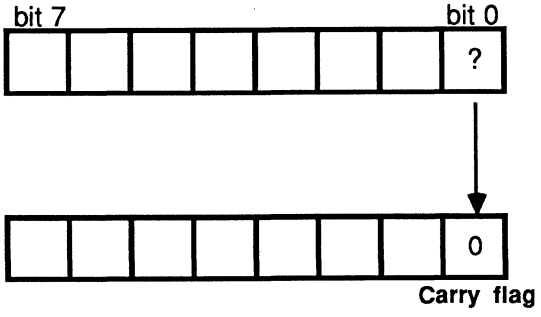
Addressing mode : **Implied**

Function : **Operates on a given register or the accumulator, but the address is always inherent in the instruction.**

Instructions : **BRK, CLC, CLD, CLI, CLT, CLV, DEX, DEY, INX, INY, NOP, PHA, PHP, PLA, PLP, RTI, RTS, SEC, SED, SEI, SET, STP, TAX, TAY, TSX, TXA, TXS, TYA, WIT**

Example : Mnemonic — **CLC**; Machine code — **18_{16}**

Processor status register



Relative

Addressing mode : **Relative**

Function : **Jumps to the address specified by the addition of the program counter and the operand.**

Instructions : **BCC, BCS, BEQ, BMI, BNE, BPL, BRA, BVC, BVS**

Example : Mnemonic

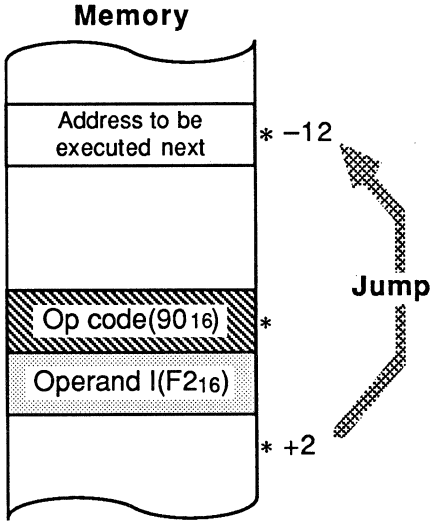
BCC *–12

Decimal

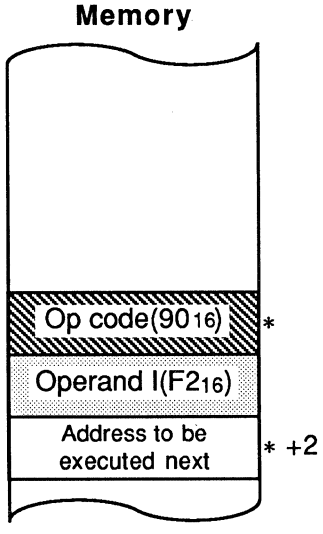
Machine code

90_{16} $F2_{16}$

*When the carry flag is clear, jumps to address * –12.*



*When the carry flag is set, goes to address * +2.*



Indirect X

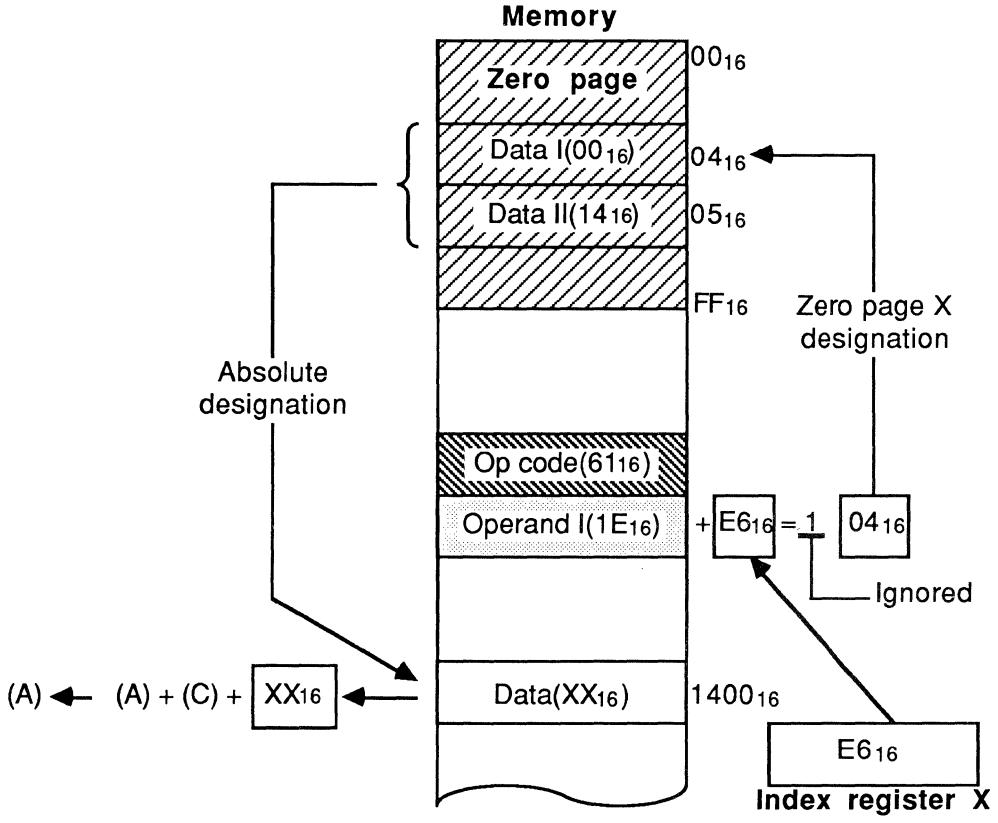
Addressing mode : **Indirect X**

Function : **Takes memory content appointed by continous 2 bytes memory on zero page that is appointed by the added value of operand and index register X as an operation data.**

Instructions : **ADC, AND, CMP, EOR, LDA, ORA, SBC, STA**

Example : Mnemonic — **ADC ($1E,X)**

Machine code — **61_{16} $1E_{16}$**



Indirect Y

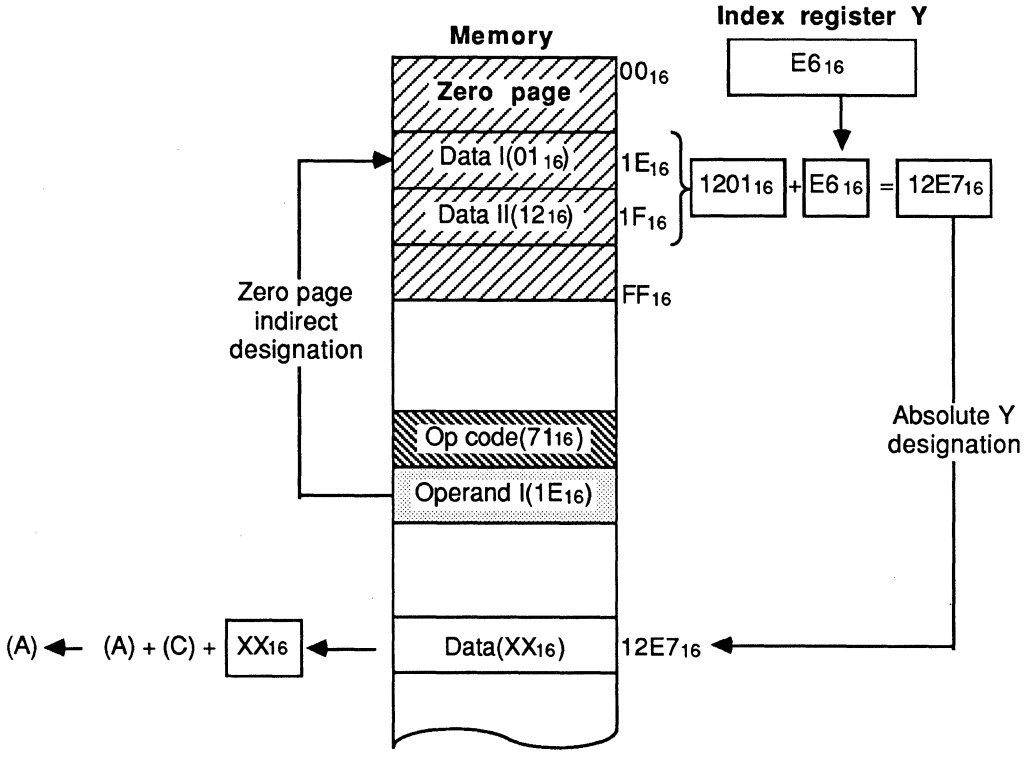
Addressing mode : **Indirect Y**

Function : **Takes memory content appointed by the adder of index register Y and the continuos 2 bytes memory on zero page that is appointed by operand as an operation data.**

Instructions : **ADC, AND, CMP, EOR, LDA, ORA, SBC, STA**

Example : Mnemonic **ADC ($1E),Y**

Machine code **71_{16} $1E_{16}$**



Indirect

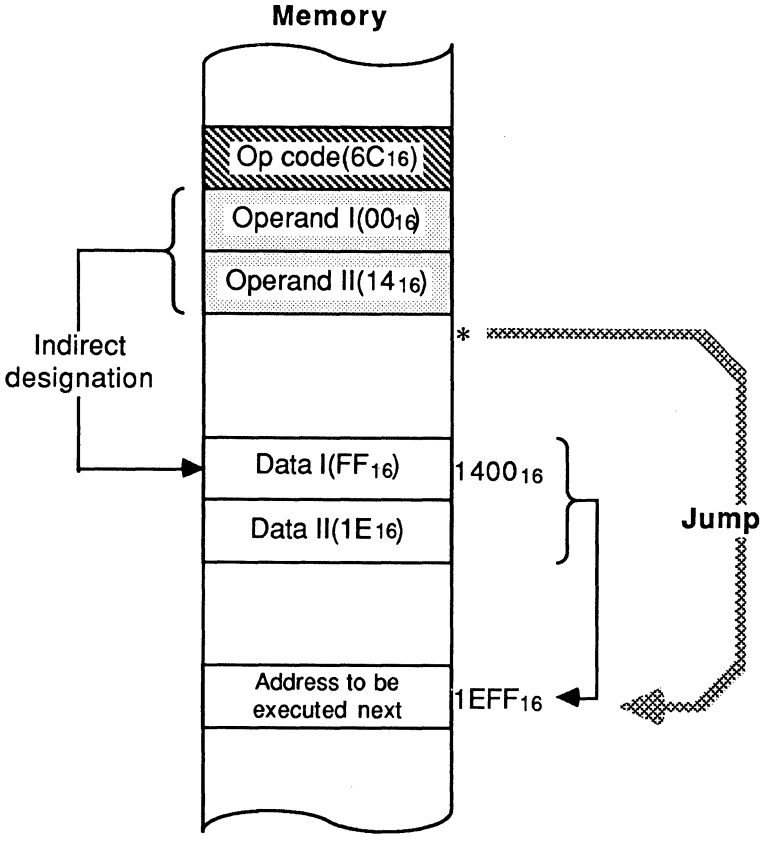
Addressing mode : **Indirect Absolute**

Function : **Specifies the continuous 2 bytes memory selected operand I and II, and jumps to the address specified by the contents of those two locations.**

Instructions : **JMP**

Example : Mnemonic
JMP ($1400)

Machine code
$6C_{16}$ 00_{16} 14_{16}



Note : This addressing mode can not appointed end address ($XXFF_{16}$) as indirect address.

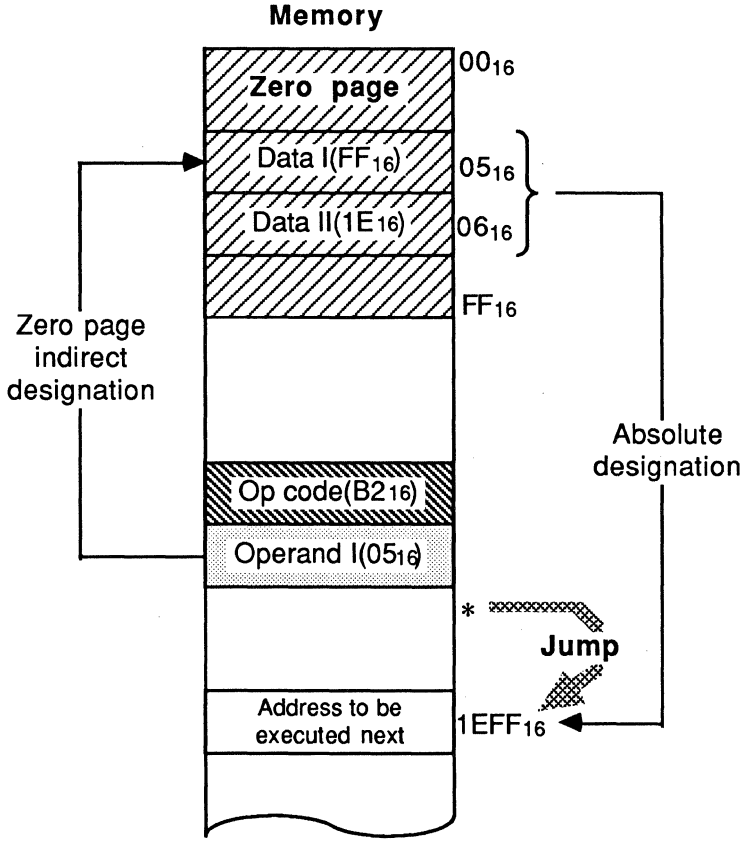
Zero pege Indirect Addressing mode

Addressing mode : **Zero page Indirect Absolute**

Function : **Jumps to the zero page address specified by the operand and uses that data and the next byte as the next address.**

Instructions : **JMP, JSR**

Example : Mnemonic **JMP ($05)** Machine code **B2$_{16}$ 05$_{16}$**



Special page

Addressing mode : **Special page**

Function : **Jumps to the special page address specified by the operand as address 2 of the lower 8-bits.**

Instructions : **JSR**

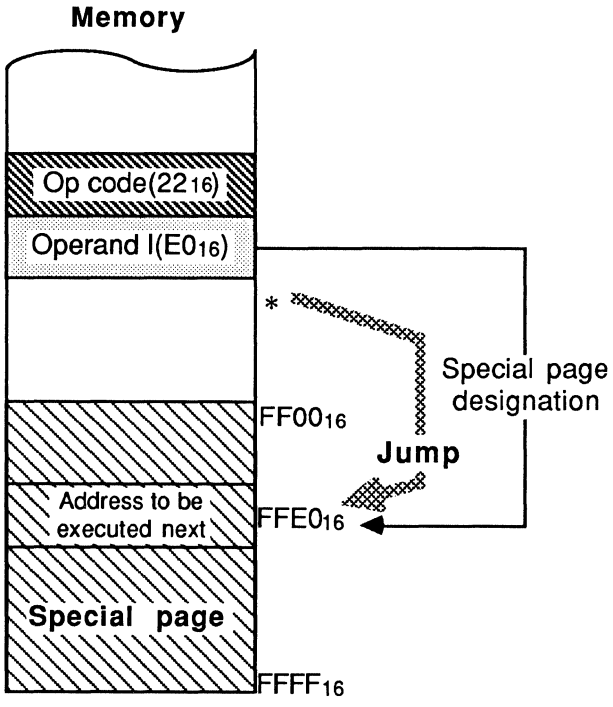
Example : Mnemonic

JSR \$FFE0

Machine code

22_{16} $E0_{16}$

This symbol indicates the Special page mode.



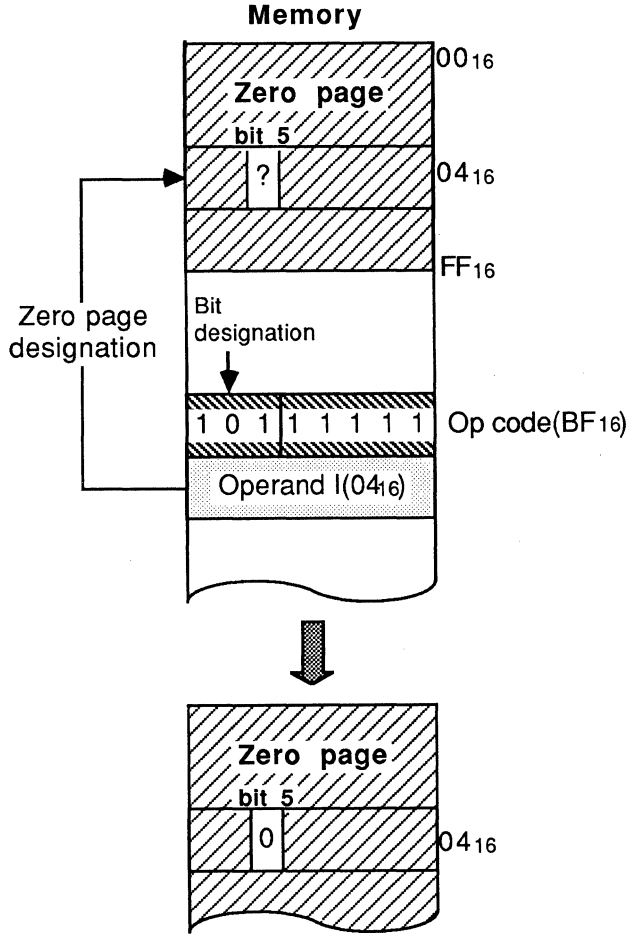
Zero page Bit

Addressing mode : **Zero page Bit**

Function : **Selects the specified bit located at the zero page address specified by the operand. The bit position is appointed by the upper 3-bits of the OP CODE.**

Instructions : **CLB, SEB**

Example : Mnemonic **CLB 5,$04** Machine code **BF16 0416**



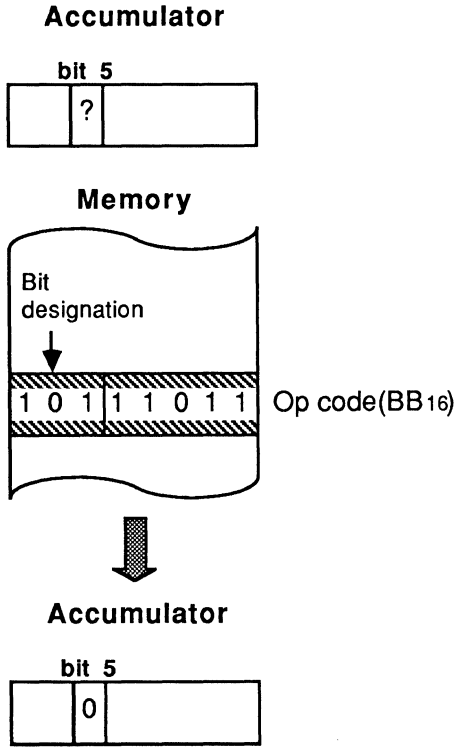
Accumulator Bit Addressing mode

Addressing mode : **Accumulator Bit**

Function : **Selects a specific bit of the accumulator designated by the upper 3-bits of OP CODE.**

Instruction: **CLB, SEB**

Example : Mnemonic **CLB 5,A** Machine code **BB16**



Accumulator Bit Relative Addressing mode

Addressing mode : **Accumulator Bit Relative**

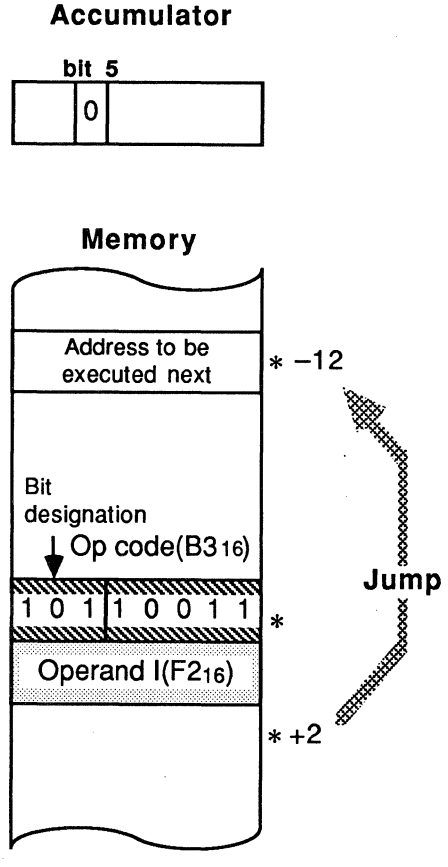
Function : **The upper 3-bits of the OP CODE, designate the bit of the accumulator is to be tested. The branch condition is met if the bit is set (BBS) or cleared (BBC). If a branch is valid, the following OPERAND is added to the program counter to generate the next address.**

Instructions : **BBC, BBS**

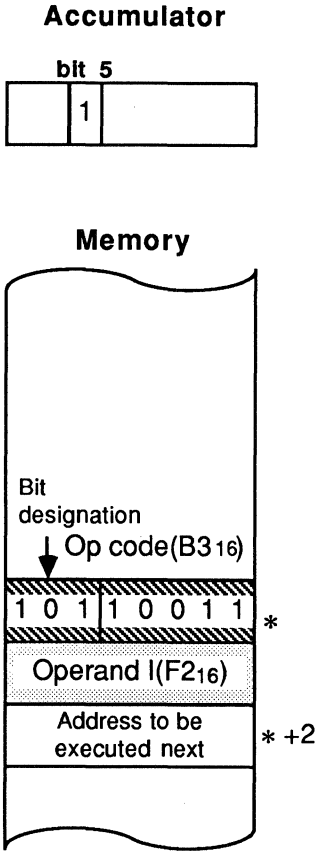
Example : Mnemonic — **BBC 5,A,*–12**

Machine code — **B3$_{16}$ F2$_{16}$**

When the bit 5 of accumulator is clear.



When the bit 5 of accumulator is set.



Zero page Bit Relative Addressing mode

Addressing mode : **Zero page Bit Relative**

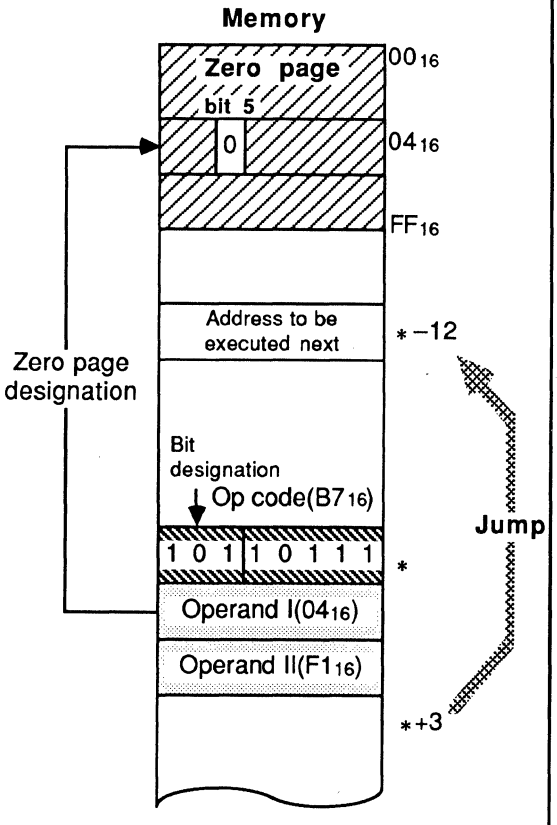
Function : **Appoints the bit at zero page specified by operand I, and jumps to the address where the added value of the program counter and operand II indicates according to the state of this bit.**

Instructions : **BBC, BBS**

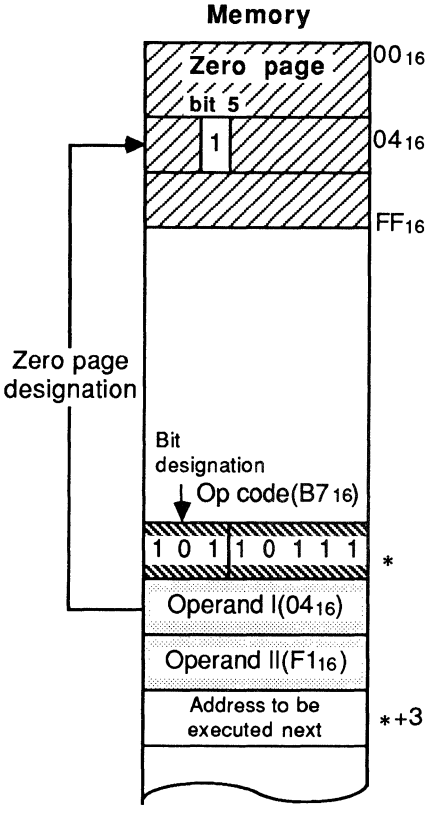
Example : Mnemonic — **BBC 5,$04,*–12**

Machine language — **B7$_{16}$ 04$_{16}$ F1$_{16}$**

*When the bit 5 at address 04$_{16}$ is clear, jumps to address *–12.*



*When the bit 5 at address 04$_{16}$ is set, goes to address *+3.*



3.2 System of Instructions

The series MELPS 740 has 73 kinds of instructions. The detailed explanation is discussed in Section 3.

Note) There are some kind of types that can not use a specific instructions.

3.2.1 Data transfer instructions

These instructions transfer the data between registers, register and memory and memories. The followings are data transfer instructions.

	Instructions	Contents
Load	LDA	Load memory into accumulator or memory where index X indicates
	LDM	Load immediate value to memory
	LDX	Load memory into index X
	LDY	Load memory into index Y
Store	STA	Store accumulator in memory
	STX	Store index X in memory
	STY	Store index Y in memory
Transfer	TAX	Transfer accumulator to index X
	TXA	Transfer index X to accumulator
	TAY	Transfer accumulator to index Y
	TYA	Transfer index Y to accumulator
	TSX	Transfer stack pointer to index X
	TXS	Transfer index X to sack pointer
Stack Operation	PHA	Push accumulator on stack
	PHP	Push processor status on stack
	PLA	Pull accumulator from stack
	PLP	Pull processor status from stack

3.2.2 Operation Instruction

The operating instruction includes the operations of addition and subtraction, logic, comparison, and rotation. Multiplication and division instruction can use in the special page addressing mode(and so M37450).

The operating instructions are as follows.

	Instructions	Contents
Addition & Subtraction	ADC	Add memory and C flag to accumulator or memory where index X indicates
	SBC	Substruction memory and C flag from accumulator or memory where index X indicates
	INC	Increment memory or accumulator by 1
	DEC	Decrement memory or accumulator by 1
	INX	Increment index X by 1
	DEX	Decrement index X by 1
	INY	Increment index Y by 1
	DEY	Decrement index Y by 1
Multiplication & Division	MUL	Multiplies accumulator by memory where index X indicates
	DIV	Divids word data that constituted memory where index X indicated to higher byte and memory where in next address to lower byte by accumulator
Logic Operation	AND	"AND" memory with accumulator
	ORA	"OR" memory with accumulator
	EOR	"Exclusive-OR" memory with accumulator
	COM	Memory one's complement
	BIT	Test bits in memory with accumulator
	TST	Test memory content (=0)
Compari-son	CMP	Compare memory and accumulator or memory where index X indicates
	CPX	Compare memory and index X
	CPY	Compare memory and index Y
Shift & Rotate	ASL	Shift left one bit (memory or accumulator)
	LSR	Shift right one bit (memory or accumulator)
	ROL	Rotate one bit left with carry (memory or accumulator)
	ROR	Rotate one bit right with carry (memory or accumulator)
	RRF	Rotate four bits right (memory)

3.2.3 Bit managing instructions

The bit managing instructions clear (0) or set (1) designated bits in accumulator or memory.

	Instructions	Contents
Bit Managing	CLB	Clear (0) designated bit in accumulator or memory
	SEB	Set (1) designated bit in accumulator or memory

3.2.4 Flag setting instructions

The flag setting instructions clear (0) or set (1) C, D, I, T and V flags.

	Instructions	Contents	
Flag Setting	CLC	Clear C flag	C flag : Carry flag
	SEC	Set C flag	
	CLD	Clear D flag	D flag : Decimal mode flag
	SED	Set D flag	
	CLI	Clear I flag	I flag : Interrupt disable flag
	SEI	Set I flag	
	CLT	Clear T flag	T flag : Index X mode flag
	SET	Set T flag	
	CLV	Clear V flag	V flag : Overflow flag

3.2.5 Jump, Branch and Return instructions

The jump, branch and return instructions as following are used to change program flow.

	Instructions	Contents
Jump	JMP	Jump to new location
	BRA	Jump to new location
	JSR	Jump to new location saving return address
Branch	BBC	Branch on designated bit in accumulator or memory clear
	BBS	Branch on designated bit in accumulator or memory set
	BCC	Branch on C flag clear
	BCS	Branch on C flag set
	BNE	Branch on Z flag clear
	BEQ	Branch on Z flag set
	BPL	Branch on N flag clear
	BMI	Branch on N flag set
	BVC	Branch on V flag clear
	BVS	Branch on V flag set
Return	RTI	Return from interrupt
	RTS	Return from subroutine

3.2.6 Interrupt Instruction (Break Instruction)

This instruction interrupts software

	Instruction	Content
Interrupt	BRK	Force break

3.2.7 Special Instructions

Special instructions control operation speed

	Instructions	Content
Special	FST	High speed operation mode (Note 1)
	SLW	Low speed operation mode (Note 1)
	WIT	Wait CPU mode (Note 1)
	STP	Stop CPU mode (Note 2)

(Note 1) This instruction is only provided for M50740A-XXXSP/FP, M50740ASP, M50741-XXXSP/FP, M50752-XXXSP, M50757-XXXSP and M50758-XXXSP.

(Note 2) This instruction is not provided for M50752-XXXSP, M50757-XXXSP and M50758-XXXSP.

3.2.8 Other Instruction

	Instruction	Content
Other	NOP	No operation

3.3 Instructions

This section has presented discussion of the series MELPS 740 instructions by arranging mnemonics of instructions alphabetically and dividing each instruction in one page basically.
Head of each page is a mnemonic. Operation, explanation, and changes of status flags are written for each instruction. In addition, statement format, machine code, byte number, and list of cycle number on each addressing mode are discussed.
The followings are symbols used in this manual.

Symbol	Means	Symbol	Means
A	Accumulator	hh	Address upper byte data in 0~255
Ai	Bit i of accumulator	ll	Address lower byte data in 0~255
PC	Program counter	zz	Zero page address data in 0~255
PCL	Lower byte of program counter	nn	Data in 0~255
PCH	Upper byte of program counter	i	Data in 0~7
P	Processor status register	*	Contents of the program counter
S	Stack pointer	Δ	Tab or space
X	Index register X	#	Immediate mode
Y	Index register Y	\	Special page mode
M	Memory	$	Hexadecimal symbol
Mi	Bit i of memory	+	Addition
C	Carry flag	−	Subtraction
Z	Zero flag	×	Multiplication
I	Interrupt disable flag	/	Division
D	Decimal operation mode flag	∧	Logical AND
B	Software interrupt (Break) flag	∨	Logical OR
T	Index X register mode flag	∀	Logical exclusive OR
V	Overflow flag	( )	Contents of register or flag
N	Negative flag	←	Direction of data transfer
REL	Relative address		
BADRS	Break address		

ADD WITH CARRY

Operation : When (T) = 0, (A) ← (A) + (M) + (C)
(T) = 1, (M(X)) ← (M(X)) + (M) + (C)

Function : When T flag is zero (T=0), this instruction adds the value of Memory, Carry flag, and register A, and stores the results in the register A and C flag.
When T flag is 1 (T=1), this adds the value of M(X), M and C, and stores the results in the M(X) and C. In this case, no A content is changed except the status flag. M(X), means the content of a memory which is appointed by the index register X.

Status flag
N : N is 1 when bit 7 is 1 after the operation; otherwise it is 0.
V : V is 1 when code bit or bit 7 is changed caused by the operation results that exceed +127 or |–128|; otherwise it is 0.
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the operation result is 0; otherwise it is 0.
C : C is 1 when the result of a binary addition exceeds 255 or when the result of a decimal addition exceeds 99; otherwise it is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔADCΔ#$nn	69_{16}, nn_{16}	2	2
Zero page	ΔADCΔ$zz	65_{16}, zz_{16}	2	3
Zero page X	ΔADCΔ$zz,X	75_{16}, zz_{16}	2	4
Absolute	ΔADCΔ$hhll	$6D_{16}$, ll_{16}, hh_{16}	3	4
Absolute X	ΔADCΔ$hhll,X	$7D_{16}$, ll_{16}, hh_{16}	3	5
Absolute Y	ΔADCΔ$hhll,Y	79_{16}, ll_{16}, hh_{16}	3	5
(Indirect X)	ΔADCΔ($zz,X)	61_{16}, zz_{16}	2	6
(Indirect Y)	ΔADCΔ($zz),Y	71_{16}, zz_{16}	2	6

(Note 1) Add 3 to cycle number when T flag is 1.
(Note 2) When the ADC instruction is used in decimal operation mode the following rule must be applied if a SEC, CLC, SED or CLD instructions are used immediately afterwards: one dummy instruction (e.g. NOP) must be inserted between the ADC and clear/set carry (or clear/set decimal mode) instruction.

LOGICAL AND

Operation : When (T) = 0, (A) ← (A) ∧ (M)
(T) = 1, (M(X)) ← (M(X)) ∧ (M)

Function : When T flag is zero, this instruction transfers the content of A and M to the ALU which performs a bit-by-bit AND operation and stores the result back in the A. When T flag is 1, this transfers M(X) and M to the ALU which performs a bit-by-bit AND operation and stores the results back in the M(X). Content of A will not change except status flag. The M(X) means a memory appointed by the X.

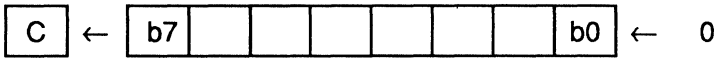
Status flag
N : N is 1 when bit 7 is 1 after the operation; otherwise it is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the operation result is 0; otherwise it is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔANDΔ#$nn	29_{16}, nn_{16}	2	2
Zero page	ΔANDΔ$zz	25_{16}, zz_{16}	2	3
Zero page X	ΔANDΔ$zz,X	35_{16}, zz_{16}	2	4
Absolute	ΔANDΔ$hhll	$2D_{16}$, ll_{16}, hh_{16}	3	4
Absolute X	ΔANDΔ$hhll,X	$3D_{16}$, ll_{16}, hh_{16}	3	5
Absolute Y	ΔANDΔ$hhll,Y	39_{16}, ll_{16}, hh_{16}	3	5
(Indirect X)	ΔANDΔ($zz,X)	21_{16}, zz_{16}	2	6
(Indirect Y)	ΔANDΔ($zz),Y	31_{16}, zz_{16}	2	6

(Note) When T flag is 1, add 3 to a cycle time.

ARITHMETIC SHIFT LEFT

Operation :



Function : This instruction shifts the content of A or M by one bit to the left, with bit 0 always being set to 0 and the bit 7 of the A or the M always being contained in the C.

Status flag

N : N is 1 when the bit 7 is 1 after the execution; otherwise it is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the result of the operation is 0; otherwise it is 0.
C : C is 1 when the bit 7 of the A or the M before this operation is 1; otherwise it is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	ΔASLΔA	$0A_{16}$	1	2
Zero page	ΔASLΔ$zz	06_{16}, zz_{16}	2	5
Zero page X	ΔASLΔ$zz,X	16_{16}, zz_{16}	2	6
Absolute	ΔASLΔ$hhll	$0E_{16}$, ll_{16}, hh_{16}	3	6
Absolute X	ΔASLΔ$hhll,X	$1E_{16}$, ll_{16}, hh_{16}	3	7

BRANCH ON BIT CLEAR

Operation : When (Mi) = 0, (PC) ← (PC) + n + REL
(Mi) = 1, (PC) ← (PC) + n
n: If addressing mode is Zero page, n=3. And if addressing mode is Accumulator, n=2.

Function : This instruction tests the appointed bit i of the M and takes a conditional branch if the bit is 0. If the bit is 1, next instruction is executed. The addressing mode is relative.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator bit Relative	ΔBBCΔi,A,$hhll	(20•i+13)16, rr16	2	4
Zero page bit Relative	ΔBBCΔi,$zz,$hhll	(20•i+17)16, zz16, rr16	3	5

(Note 1) rr16=$hhll–(*+n). The rr16 is a value in a range of –128 ~ +127.
(Note 2) Add 2 to cycle number for a branch execution.
(Note 3) One dummy or other instruction cycle is needed after changing the content of interrupt request bit (bit 1, bit 3, bit 5, bit 7 in address 00FE16 and bit 7 in address 00FF16) to execute this instruction.

BRANCH ON BIT SET

Operation : When (Mi) = 1, (PC) ← (PC) + n + REL
(Mi) = 0, (PC) ← (PC) + n
n : If addressing mode is zero page, n=3. And if addressing mode is accumulator, n=2.

Function : This instruction tests the appointed bit i of the M and takes a conditional branch if the bit is 1. If the bit is 0, next instruction is executed. The addressing mode is relative.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator bit Relative	ΔBBSΔi,A,$hhll	(20•i+3)16, rr16	2	4
Zero page bit Relative	ΔBBSΔi,$zz,$hhll	(20•i+7)16, zz16, rr16	3	5

(Note 1) rr_{16}=$hhll–(*+n). The rr_{16} is a value in a range of –128 ~ +127.
(Note 2) Add 2 to cycle number for a branch execution.
(Note 3) One dummy or other instruction cycle is needed after changing the content of interrupt request bit (bit 1, bit 3, bit 5, bit 7 in address $00FE_{16}$ and bit 7 in address $00FF_{16}$) to execute this instruction.

BRANCH ON CARRY CLEAR

Operation : When (C) = 0, (PC) ← (PC) + 2 + REL
(C) = 1, (PC) ← (PC) + 2

Function : This instruction takes a conditional branch to the appointed address if the C is 0. If the C is 1, the next instruction is executed. The addressing mode is relative.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	ΔBCCΔ$hhll	90_{16}, rr_{16}	2	2

(Note 1) rr_{16}=$hhll–(*+2). The rr16 is a value in a range of –128 ~ +127.
(Note 2) Add 2 to cycle number for a branch execution.

BRANCH ON CARRY SET

Operation : When (C) = 1, (PC) ← (PC) + 2 + REL
(C) = 0, (PC) ← (PC) + 2

Function : This instruction takes a conditional branch to the appointed address if the C is 1. If the C is 0, the address goes to the next address.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	ΔBCSΔ$hhll	B0$_{16}$, rr$_{16}$	2	2

(Note 1) rr$_{16}$=$hhll–(*+2). The rr$_{16}$ is a value in a range of –128 ~ +127
(Note 2) Add 2 to a cycle time for a branch execution.

Operation : When (Z) = 1, (PC) ← (PC) + 2 + REL
(Z) = 0, (PC) ← (PC) + 2

Function : This instruction takes a conditional branch to the appointed address when the Z is 1. If the Z is 0, the next instruction is executed. The addressing mode is relative.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	ΔBEQΔ$hhll	$F0_{16}$,rr_{16}	2	2

(Note 1) rr_{16}=$hhll–(*+2). The rr_{16} is a value in a range of –128 ~ +127.
(Note 2) Add 2 to cycle number for a branch execution.

TEST BIT IN MEMORY WITH ACCUMULATOR

Operation : (A) ∧ (M)

Function : This instruction takes a logical AND of every bit of the A and M contents; however, no result is stored anywhere. The contents of the N, V and the Z flags will be changed but no change for the A and M contents.

Status flag
N : N is 1 when bit 7 of M is 1; otherwise it is 0.
V : V is 1 when bit 6 of M is 1; otherwise it is 0.
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the result of the logical AND is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	ΔBITΔ$zz	24_{16}, zz_{16}	2	3
Absolute	ΔBITΔ$hhll	$2C_{16}$, ll_{16}, hh_{16}	3	4

BRANCH ON RESULT MINUS

Operation : When (N) = 1, (PC) ← (PC) + 2 + REL
(N) = 0, (PC) ← (PC) + 2

Function : This instruction takes a conditional branch to the appointed address when the N is 1. If the N is 0, the next instruction is executed. The addressing mode is relative.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	ΔBMIΔ$hhll	30_{16}, rr_{16}	2	2

(Note 1) rr_{16}=$hhll–(*+2). The rr_{16} is a value in a range of –128 ~ +127.
(Note 2) Add 2 to cycle time for a branch execution.

BRANCH ON NOT EQUAL

Operation : When (Z) = 0, (PC) ← (PC) + 2 + REL
(Z) = 1, (PC) ← (PC) + 2

Function : This instruction takes a conditional branch to the appointed address if the Z is 0. If the Z is 1, the next instruction is executed. The addressing mode is relative.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	ΔBNEΔ$hhll	$D0_{16}$, rr_{16}	2	2

(Note 1) rr_{16}=$hhll–(*+2). The rr_{16} is a value in a range of –128 ~ +127.
(Note 2) Add 2 to cycle number for a branch execution.

BRANCH ON RESULT PLUS

Operation : When (N) = 0, (PC) ← (PC) + 2 + REL
(N) = 1, (PC) ← (PC) + 2

Function : This instruction takes a conditional branch to the appointed address if the N is 0. If the N is 1, the next instruction is executed. The addressing mode is relative.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	ΔBPLΔ$hhll	10_{16}, rr_{16}	2	2

(Note 1) rr_{16}=$hhll–(*+2). The rr16 is a value in a range of –128 ~ +127.
(Note 2) Add 2 to cycle number for a branch execution.

BRANCH ALWAYS

Operation : (PC) ← (PC) + 2 + REL

Function : This instruction jumps to the appointed address. The objective address is expressed in relative addressing mode.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	ΔBRAΔ$hhll	80_{16}, rr_{16}	2	4

(Note) rr_{16}=$hhll–(*+2). The rr_{16} is a value in a range of –128 ~ +127.

Operation : (B) ← 1
(PC) ← (PC) + 2
(M(S)) ← (PC_H)
(S) ← (S) − 1
(M(S)) ← (PC_L)
(S) ← (S) − 1
(M(S)) ← (PS)
(S) ← (S) − 1
(I) ← 1
(PC) ← BADRS (Note 1)

Function : When the BRK instruction is executed, the CPU pushes the current PC contents onto the stack and stores the break address data (BADRS) . The break address data is stored in the interrupt vector table into the PC.

Status flag
N : No change
V : No change
T : No change
B : 1
I : No change
D : 1
Z : No change
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔBRKΔ	00_{16}	1	7

(Note 1) "BADRS" means a break address.
(Note 2) Addresses vary for different microcomputers.
(Note 3) The value of the PC pushed onto the stack by the execution of the BRK instruction is the BRK instruction address plus two. Therefore, the byte following the BRK will not be executed when the value of the PC is returned from the BRK routine.
(Note 4) In some microcomputers, the BRK instruction uses the same vector address as $\overline{INT_2}$.

BRANCH ON OVERFLOW CLEAR

Operation : When (V) = 0, (PC) ← (PC) + 2 + REL
(V) = 1, (PC) ← (PC) + 2

Function : This instruction takes a branch to the appointed address if the V is 0. The addressing mode of the objective address is relative. If the V is 1, the address goes to the next address.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	ΔBVCΔ$hhll	50_{16}, rr_{16}	2	2

(Note 1) rr_{16}=$hhll–(*+2). The rr_{16} is a value in a range of –128 ~ +127.
(Note 2) Add 2 to cycle number for a branch execution.

BRANCH ON OVERFLOW SET

Operation : When (V) = 1, (PC) ← (PC) + 2 + REL
(V) = 0, (PC) ← (PC) + 2

Function : This instruction takes a branch to the appointed address if the V is 1. The addressing mode of the objective address is relative. If the V is 0, the PC goes to the next address.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Relative	ΔBVSΔ$hhll	70_{16}, rr_{16}	2	2

(Note 1) rr_{16}=$hhll–(*+2). The rr_{16} is a value in a range of –128 ~ +127.
(Note 2) Add 2 to cycle number for a branch execution.

CLEAR BIT

Operation : (Ai) ← 0, or
(Mi) ← 0

Function : This instruction resets the appointed bit i of the A or the M to 0.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator bit Relative	ΔCLBΔi,A	$(20 \cdot i+1B)_{16}$	1	2
Zero page bit Relative	ΔCLBΔi,$zz	$(20 \cdot i+1F)_{16}$, ZZ_{16}	2	5

CLEAR CARRY FLAG

Operation : (C) ← 0

Function : This instruction resets the C to 0.

Status flag
N : No change
V : No change
T : No change
B : No change
I : No change
D : No change
Z : No change
C : 0

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔCLC	18_{16}	1	2

CLEAR DECIMAL MODE

Operation : (D) ← 0

Function : This instruction resets the D to 0.

Status flag
N : No change
V : No change
T : No change
B : No change
I : 0
D : No change
Z : No change
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔCLD	$D8_{16}$	1	2

CLEAR INTERRUPT DISABLE STATUS

Operation : (I) ← 0

Function : This instruction resets the I to 0.

Status flag
N : No change
V : No change
T : No change
B : No change
I : No change
D : 0
Z : No change
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔCLI	58_{16}	1	2

CLEAR TRANSFER FLAG

Operation : (T) ← 0

Function : This instruction resets the T to 0.

Status flag
N : No change
V : No change
T : 0
B : No change
I : No change
D : No change
Z : No change
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔCLT	12_{16}	1	2

Operation : (V) ← 0

Function : This instruction resets the V to 0.

Status flag
N : No change
V : 0
T : No change
B : No change
I : No change
D : No change
Z : No change
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔCLV	$B8_{16}$	1	2

Operation : When (T) = 0, (A) − (M)
(T) = 1, (M(X)) − (M)

Function : This instruction subtracts the content of M from the A contents if the T is 0. No result is stored in anywhere and none of the content of A, B or V is changed at this moment. If the T is 1, the CMP subtracts the M contents from the M(X) contents. No result is stored anywhere and the V will not be changed. The M(X) means the contents of the memory where the X indicates.

Status flag
N : N is 1 when the bit 7 is 1 after the subtraction; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the subtraction result is 0; otherwise Z is 0.
C : C is 1 when the subtraction result is equal to or greater than 0; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔCMPΔ#$nn	$C9_{16}$, nn_{16}	2	2
Zero page	ΔCMPΔ$zz	$C5_{16}$, zz_{16}	2	3
Zero page X	ΔCMPΔ$zz,X	$D5_{16}$, zz_{16}	2	4
Absolute	ΔCMPΔ$hhll	CD_{16}, ll_{16}, hh_{16}	3	4
Absolute X	ΔCMPΔ$hhll,X	DD_{16}, ll_{16}, hh_{16}	3	5
Absolute Y	ΔCMPΔ$hhll,Y	$D9_{16}$, ll_{16}, hh_{16}	3	5
(Indirect X)	ΔCMPΔ($zz,X)	$C1_{16}$, zz_{16}	2	6
(Indirect Y)	ΔCMPΔ($zz),Y	$D1_{16}$, zz_{16}	2	6

Operation : (M) ← $\overline{(M)}$

Function : This instruction takes one's complement of the M contents and stores the result in M.

Status flag
N : N is 1 when the bit 7 of the M is 1 after the implement; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the execution result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	ΔCOMΔ$zz	44$_{16}$, zz$_{16}$	2	5

Operation : (X) – (M)

Function : This instruction subtracts the content of M from the X contents; however, no result is stored anywhere and none of X, M or V content is changed at this moment.

Status flag
N : N is 1 when the 7 bit after the subtraction; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the subtracted result is 0; otherwise Z is 0.
C : C is 1 when the subtracted result is equal to or greater than 0; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔCPXΔ#$nn	$E0_{16}$, nn_{16}	2	2
Zero page	ΔCPXΔ$zz	$E4_{16}$, zz_{16}	2	3
Absolute	ΔCPXΔ$hhll	EC_{16}, ll_{16}, hh_{16}	3	4

Operation : (Y) – (M)

Function : This instruction subtracts the content of the M from the Y contents;
however, no result is stored anywhere and none of the content of Y, M or V is changed in this instruction.

Status flag
N : N is 1 when the bit 7 is 1 after the subtraction; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the subtracted result is 0; otherwise Z is 0.
C : C is 1 when the subtracted result is equal to or greater than 0; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔCPYΔ#$nn	$C0_{16}$, nn_{16}	2	2
Zero page	ΔCPYΔ$zz	$C4_{16}$, zz_{16}	2	3
Absolute	ΔCPYΔ$hhll	CC_{16}, ll_{16}, hh_{16}	3	4

DECREMENT BY ONE

Operation : (A) ← (A) − 1, or
(M) ← (M) − 1

Function : This instruction subtracts 1 from the A or the M contents; however, neither of V or C is changed.

Status flag
N : N is 1 when the bit 7 is 1 after the addition; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the added result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	ΔDECΔA	$1A_{16}$	1	2
Zero page	ΔDECΔ$zz	$C6_{16}$, zz_{16}	2	5
Zero page X	ΔDECΔ$zz,X	$D6_{16}$, zz_{16}	2	6
Absolute	ΔDECΔ$hhll	CE_{16}, ll_{16}, hh_{16}	3	6
Absolute X	ΔDECΔ$hhll,X	DE_{16}, ll_{16}, hh_{16}	3	7

Operation : (X) ← (X) − 1

Function : This instruction subtracts one from the current value of the X and stores the result in the X.

Status flag
N : N is 1 when the bit 7 after the addition is 1; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the added result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔDEX	CA_{16}	1	2

DECREMENT INDEX REGISTER Y BY ONE

Operation : (Y) ← (Y) − 1

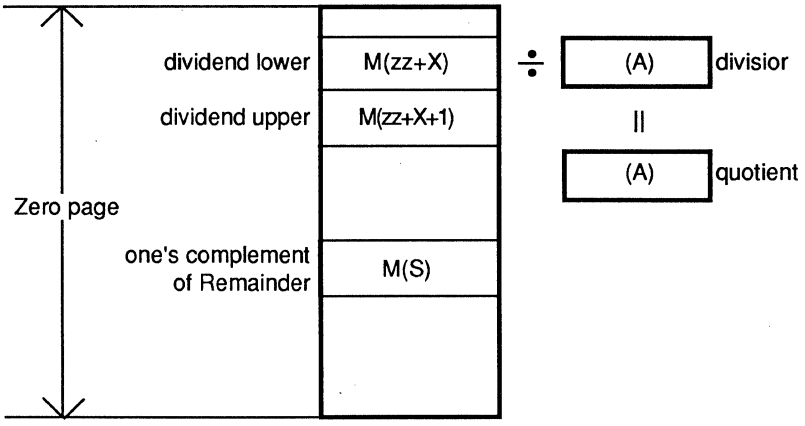
Function : This instruction subtracts one from the current value in the Y and stores the result into the Y; however, neither of the V or C content is changed at this moment.

Status flag
N : N is 1 when the bit 7 is 1 after the subtraction; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the subtracted result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔDEY	88_{16}	1	2

Operation : (A) ← (M(zz+X+1),M(zz+X)) / (A)
M(S) ← one's complement of Remainder
(S) ← (S) − 1

Function : Divides the 16-bit data in M(zz+X) (lower byte) and M(zz+X+1) (higher byte) by the accumulator contents. The quotient is stored in the accumulator and the one's complement of the remainder is pushed onto the stack.



Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page X	ΔDIVΔ$zz,X	E2₁₆, zz₁₆	2	16

(Note 1) This instruction can not detect overflow due to a divide by zero. Please check for a non-zero divisor in software.
(Note 2) This instruction changes the stack pointer and the contents of the accumulator.
(Note 3) This instruction is not available on the entire MELPS 740 series.

EXCLUSIVE OR MEMORY WITH ACCUMULATOR

Operation : When (T) = 0, (A) ← (A) ∀ (M)
(T) = 1, (M(X)) ← (M(X)) ∀ (M)

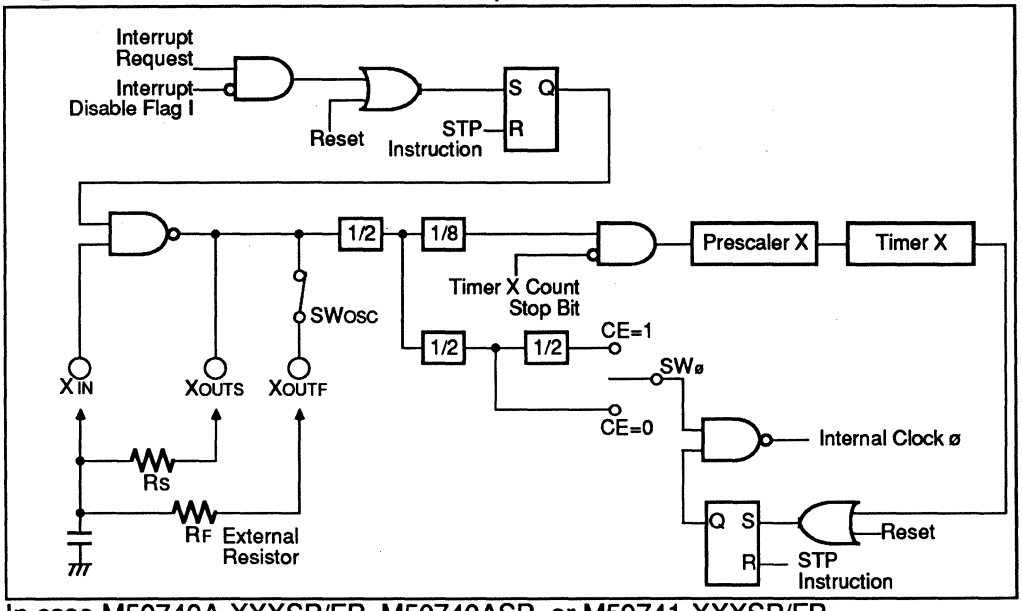
Function : This instruction transfers the contents of the M and the A to the ALU which performs an exclusive OR bit-by-bit, and stores the result into the A when the T is 0. When the T is 1, it transfers the contents of the M(X) and the M in the same way us above. The content of A will not be changed except the status flag contents. The M(X) means a memory content of the address appointed by X.

Status flag
N : N is 1 when the bit 7 is 1 after the operation; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the operation result is 0; otherwise Z is 0.
C : No change

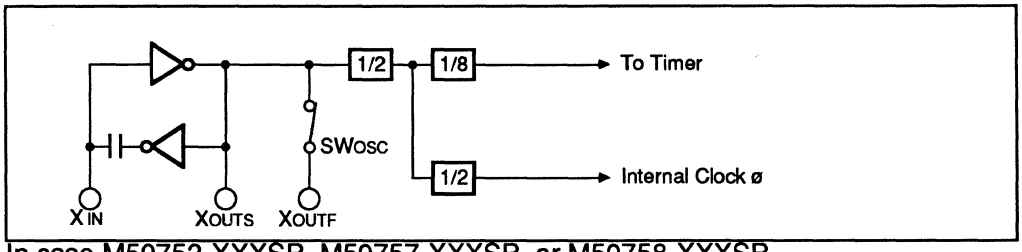
Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔEORΔ#$nn	49_{16}, nn_{16}	2	2
Zero page	ΔEORΔ$zz	45_{16}, zz_{16}	2	3
Zero page X	ΔEORΔ$zz,X	55_{16}, zz_{16}	2	4
Absolute	ΔEORΔ$hhll	$4D_{16}$, ll_{16}, hh_{16}	3	4
Absolute X	ΔEORΔ$hhll,X	$5D_{16}$, ll_{16}, hh_{16}	3	5
Absolute Y	ΔEORΔ$hhll,Y	59_{16}, ll_{16}, hh_{16}	3	5
(Indirect X)	ΔEORΔ($zz,X)	41_{16}, zz_{16}	2	6
(Indirect Y)	ΔEORΔ($zz),Y	51_{16}, zz_{16}	2	6

(Note) Add 3 to cycle number when T is 1.

Operation : Connects oscillator output to XOUTF.



In case M50740A-XXXSP/FP, M50740ASP, or M50741-XXXSP/FP



In case M50752-XXXSP, M50757-XXXSP, or M50758-XXXSP

Function : When CR clock generating circuit is formed, external resistance value became small. Therefore clock oscillating frequency become high.

$$\text{Resistance value} = \frac{RS \cdot RF}{RS + RF}$$

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔFST	$E2_{16}$	1	2

(Note 1) This instruction is not for any other models than M50740A-XXXSP/FP, M50740ASP, M50741-XXXSP/FP, M50752-XXXSP, M50757-XXXSP and M50758-XXXSP.

(Note 2) The SWOSC closes at reset.

INCREMENT BY ONE

Operation : (A) ← (A) + 1, or
(M) ← (M) + 1

Function : This instruction adds one to the A or the M contents.

Status flag
N : N is 1 when the bit 7 is 1 after the addition; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the added result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	ΔINCΔA	$3A_{16}$	1	2
Zero page	ΔINCΔ$zz	$E6_{16}$, zz_{16}	2	5
Zero page X	ΔINCΔ$zz,X	$F6_{16}$, zz_{16}	2	6
Absolute	ΔINCΔ$hhll	EE_{16}, ll_{16}, hh_{16}	3	6
Absolute X	ΔINCΔ$hhll,X	FE_{16}, ll_{16}, hh_{16}	3	7

INCREMENT INDEX REGISTER X BY ONE

Operation : (X) ← (X) + 1

Function : This instruction adds one to X contents.

Status flag
N : N is 1 when the bit 7 is 1 after the addition; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the added result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔINX	$E8_{16}$	1	2

INCREMENT INDEX REGISTER Y BY ONE

Operation : (Y) ← (Y) + 1

Function : This instruction adds one to the Y contents.

Status flag
N : N is 1 when bit 7 is 1 after the addition; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the added result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔINY	$C8_{16}$	1	2

Operation : When addressing mode is
(a) Absolute, then
(PC) ← hhll
(b) Indirect Absolute, then
(PCL) ← (hhll)
(PCH) ← (hhll+1)
(c) Zero page Indirect Absolute, then
(PCL) ← (zz)
(PCH) ← (zz+1)

Function : This instruction jumps to the address appointed by the following three addressing modes; the absolute addressing mode, Indirect absolute addressing mode or the zero page Indirect absolute addressing mode.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Absolute	ΔJMPΔ$hhll	$4C_{16}$,ll$_{16}$,hh$_{16}$	3	3
Indirect	ΔJMPΔ($hhll)	$6C_{16}$,ll$_{16}$,hh$_{16}$	3	5
Zero page Indirect	ΔJMPΔ($zz)	$B2_{16}$,zz$_{16}$	2	4

JUMP TO SUBROUTINE

Operation : (M(S)) ← (PCH)
(S) ← (S) – 1
After the above operations, if the addressing mode is
(a) Absolute, then
(PC) ← hhll
(b) Special page, then
(PCL) ← ll
(PCH) ← FF16
(c) Zero page Indirect, then
(PCL) ← (zz)
(PCH) ← (zz+1)

Function : This instruction jumps to the address appointed by the Absolute's, Special page's or Zero page Indirect's addressing mode after saving the PC contents into the stack.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Absolute	ΔJSRΔ$hhll	2016, ll16, hh16	3	6
Special page	ΔJSRΔ\$hhll (Note)	2216, ll16	2	5
Zero page Indirect	ΔJSRΔ($zz)	0216, zz16	2	7

(Note) "\" denote special page(At DC16 of the ASCII code). hh16 must be FF16 in the special page addressing mode. However, hh16 must be 1F16 at the M50740A-XXXSP/FP, M50741-XXXSP/FP, M50740ASP, M50752-XXXSP, M50757-XXXSP and M50758-XXXSP in the special page addressing mode.

LOAD ACCUMULATOR WITH MEMORY

Operation : When (T) = 0, (A) ← (M)
(T) = 1, (M(X)) ← (M)

Function : This instruction transfers the content of M to the A when the T is 0. It transfers the M content to the (M(X)) when the T is 1. The content of A is not changed at the time except the status flag. The M(X) is the memory content of the address where the X appointed.

Status flag
N : N is 1 when the bit 7 is 1 after the execution; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the execution result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔLDAΔ#$nn	$A9_{16}$, nn_{16}	2	2
Zero page	ΔLDAΔ$zz	$A5_{16}$, nn_{16}	2	3
Zero page X	ΔLDAΔ$zz,X	$B5_{16}$, nn_{16}	2	4
Absolute	ΔLDAΔ$hhll	AD_{16}, ll_{16}, hh_{16}	3	4
Absolute X	ΔLDAΔ$hhll,X	BD_{16}, ll_{16}, hh_{16}	3	5
Absolute Y	ΔLDAΔ$hhll,Y	$B9_{16}$, ll_{16}, hh_{16}	3	5
(Indirect X)	ΔLDAΔ($zz,X)	Al_{16}, zz_{16}	2	6
(Indirect Y)	ΔLDAΔ($zz),Y	Bl_{16}, zz_{16}	2	6

(Note) Add 2 to cycle number when T is 1.

Operation : (M) ← nn

Function : This instruction loads the immediate value into the M.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	ΔLDMΔ#$nn,$zz	$3C_{16}$, nn_{16}, zz_{16}	3	4

Operation : (X) ← (M)

Function : This instruction loads the M content into the X.

Status flag
N : N is 1 when the bit 7 is 1 after the execution; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the execution result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔLDXΔ#$nn	A2$_{16}$, nn$_{16}$	2	2
Zero page	ΔLDXΔ$zz	A6$_{16}$, zz$_{16}$	2	3
Zero page Y	ΔLDXΔ$zz,Y	B6$_{16}$, zz$_{16}$	2	4
Absolute	ΔLDXΔ$hhll	AE$_{16}$, ll$_{16}$, hh$_{16}$	3	4
Absolute Y	ΔLDXΔ$hhll,Y	BE$_{16}$, ll$_{16}$, hh$_{16}$	3	5

Operation : (Y) ← (M)

Function : This instruction loads the M content into the Y.

Status flag
N : N is 1 when the bit 7 is 1 after the execution; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the execution result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔLDYΔ#$nn	$A0_{16}$, nn_{16}	2	2
Zero page	ΔLDYΔ$zz	$A4_{16}$, zz_{16}	2	3
Zero page X	ΔLDYΔ$zz,X	$B4_{16}$, zz_{16}	2	4
Absolute	ΔLDYΔ$hhll	AC_{16}, ll_{16}, hh_{16}	3	4
Absolute X	ΔLDYΔ$hhll,X	BC_{16}, ll_{16}, hh_{16}	3	5

LOGICAL SHIFT RIGHT

Operation : 0 → [b7 | | | | | | | b0] → [C]

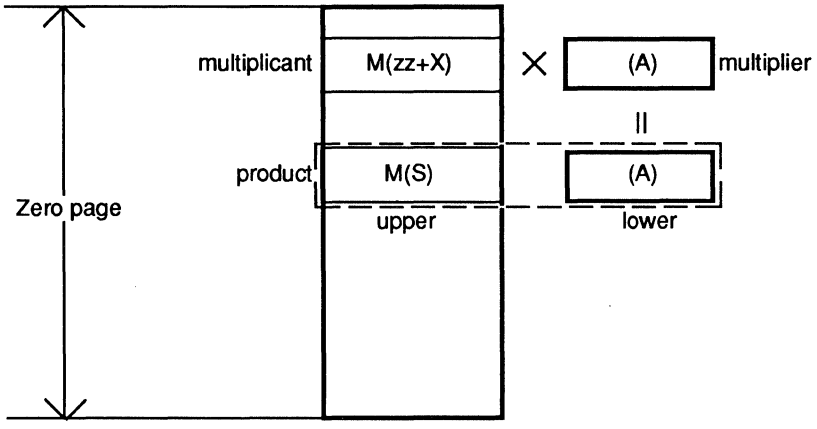
Function : This instruction shifts either the A or the M location one bit to the right with the bit 7 of the result always being set to 0, and the bit 0 being stored in the C.

Status flag
N : 0
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the execution result is 0; otherwise Z is 0.
C : C is 1 when the bit 0 of either the A or the M before the execution is 1; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	ΔLSRΔA	$4A_{16}$	1	2
Zero page	ΔLSRΔ$zz	46_{16}, zz_{16}	2	5
Zero page X	ΔLSRΔ$zz,X	56_{16}, zz_{16}	2	6
Absolute	ΔLSRΔ$hhll	$4E_{16}$, ll_{16}, hh_{16}	3	6
Absolute X	ΔLSRΔ$hhll,X	$5E_{16}$, ll_{16}, hh_{16}	3	7

Operation : M(S) • (A) ← (A) × M(zz+X)
(S) ← (S) − 1

Function : Multiplies accumulator with the memory specified by the zero page X addressing mode and stores the high byte of the result on the stack and the low byte in the accumulator.



Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page X	ΔMULΔ$zz,X	62₁₆, zz₁₆	2	15

(Note 1) Be careful instruction of accumulator and stack pointer changes when this instruction execute.
(Note 2) This instruction can use in special kind (and so M37450).

Operation : (PC) ← (PC) + 1

Function : This instruction adds one to the program counter but does no other operation.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔNOP	EA_{16}	1	2

ORA OR MEMORY WITH ACCUMULATOR ORA

Operation : When (T) = 0, (A) ← (A) ∨ (M)
(T) = 1, (M(X)) ← (M(X)) ∨ (M)

Function : This instruction transfers the A and the M contents to the ALU which performs an "OR" bit-by-bit, and stores the result in the A if flag T is 0. If the T is 1, it operates the M(X) and the M in the same way of above. The content of A is not changed at the time except the status flag. The M(X) means the memory content of the address where the X appoints.

Status flag
N : N is when the bit 7 is 1 after the operation; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the execution result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔORAΔ#$nn	09_{16}, nn_{16}	2	2
Zero page	ΔORAΔ$zz	05_{16}, zz_{16}	2	3
Zero page X	ΔORAΔ$zz,X	15_{16}, zz_{16}	2	4
Absolute	ΔORAΔ$hhll	$0D_{16}$, ll_{16}, hh_{16}	3	4
Absolute X	ΔORAΔ$hhll,X	$1D_{16}$, ll_{16}, hh_{16}	3	5
Absolute Y	ΔORAΔ$hhll,Y	19_{16}, ll_{16}, hh_{16}	3	5
(Indirect X)	ΔORAΔ($zz,X)	01_{16}, zz_{16}	2	6
(Indirect Y)	ΔORAΔ($zz),Y	11_{16}, zz_{16}	2	6

(Note) Add 3 to cycle number when T is 1.

PUSH ACCUMULATOR ON STACK

Operation : (M(S)) ← (A)
(S) ← (S) − 1

Function : This instruction pushes the content of A to the address the stack pointer appoints, and decrements the stack pointer contents by one.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔPHA	48_{16}	1	3

PUSH PROCESSOR STATUS ON STACK

Operation : (M(S)) ← (P)
(S) ← (S) − 1

Function : This instruction pushes the contents of the processor status register to the memory of the address where the stack pointer appoints and decrements the stack pointer contents by one.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔPHP	08_{16}	1	3

PULL ACCUMULATOR FROM STACK

Operation : (S) ← (S) + 1
(A) ← (M(S))

Function : This instruction adds one to the stack pointer and loads the memory content of the address where the stack pointer appoints into the A.

Status flag
N : N is 1 when the bit 7 after the execution is 1; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the execution result is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔPLA	68_{16}	1	4

PULL PROCESSOR STATUS FROM STACK

Operation : (S) ← (S) + 1
(P) ← (M(S))

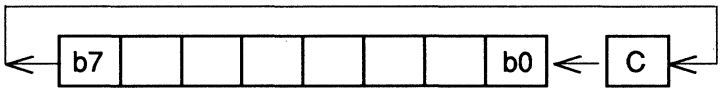
Function : This instruction adds one to the stack pointer and loads the memory content of the address where the stack pointer appoints into the processor status register.

Status flag : Value returns to the original one that was pushed in the stack.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔPLP	28_{16}	1	4

ROTATE ONE BIT LEFT

Operation :



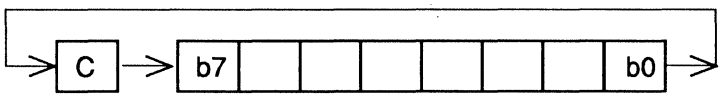
Function : This instruction shifts either the A or the M one bit left combining with the C. The C content is stored in the bit 0 and the bit 7 is stored in the C.

Status flag
N : N is 1 when the bit 6 is 1 before the execution; otherwise N is 0.
V: No change
T: No change
B: No change
I: No change
D: No change
Z: Z is 1 when the execution result is 0; otherwise Z is 0.
C: C is 1 when bit 7 is 1 befor execution; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	ΔROLΔA	$2A_{16}$	1	2
Zero page	ΔROLΔ$zz	26_{16}, zz_{16}	2	5
Zero page X	ΔROLΔ$zz,X	36_{16}, zz_{16}	2	6
Absolute	ΔROLΔ$hhll	$2E_{16}$, ll_{16}, hh_{16}	3	6
Absolute X	ΔROLΔ$hhll,X	$3E_{16}$, ll_{16}, hh_{16}	3	7

ROTATE ONE BIT RIGHT

Operation :



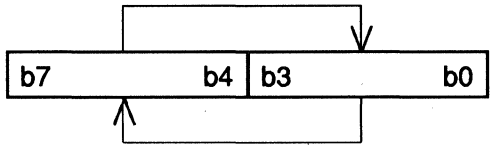
Function : This instruction shifts either the A or the M one bit right combining with the C. The C content is stored in the bit 7 and the bit 0 content is stored in the C.

Status flag
N : N is 1 when the C content before the execution is 1; otherwise N is 0.
V: No change
T: No change
B: No change
I: No change
D: No change
Z: Z is 1 when the execution result is 0; otherwise Z is 0.
C: C is 1 when bit 0 is 1 before execution; otherwise C is 0.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator	ΔRORΔA	$6A_{16}$	1	2
Zero page	ΔRORΔ$zz	66_{16}, zz_{16}	2	5
Zero page X	ΔRORΔ$zz,X	76_{16}, zz_{16}	2	6
Absolute	ΔRORΔ$hhll	$6E_{16}$, ll_{16}, hh_{16}	3	6
Absolute X	ΔRORΔ$hhll,X	$7E_{16}$, ll_{16}, hh_{16}	3	7

ROTATE RIGHT OF FOUR BITS

Operation :



Function : This instruction rotates 4 bits of the M content to the right (It may be called "swap nibbles").

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	ΔRRFΔ$zz	82_{16}, zz_{16}	2	8

RETURN FROM INTERRUPT

Operation : (S) ← (S) + 1
(P) ← (M(S))
(S) ← (S) + 1
(PC_L) ← (M(S))
(S) ← (S) + 1
(PC_H) ← (M(S))

Function : This instruction reload the status flag and the PC content that have been pushed to the stack when they are interrupted and sets them back to their pre-interrupt state. The M(S) means the memory content of the address where the stack pointer appoints.

Status flag : Value returns to the original one that was pushed in the stack.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔRTI	40_{16}	1	6

Operation : (S) ← (S) + 1
(PCL) ← (M(S))
(S) ← (S) + 1
(PCH) ← (M(S))
(PC) ← (PC) + 1

Function : This instruction reloads the content of the PC which has been pushed to the stack when the subroutine is called and increments the PC by one. The PC points the next instruction at this time. The M(S) means the memory content of the address where the stack pointer appoints.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔRTS	60_{16}	1	6

SUBTRACT WITH CARRY

Operation : When (T) = 0, (A) ← (A) − (M) − $\overline{(C)}$
(T) = 1, (M(X)) ← (M(X)) − (M) − $\overline{(C)}$

Function : This instruction subtracts the value of the M and the complement of C flag from the value of the A, and stores the results into the A and the C if the T is 0. If the T is 1, the instruction subtracts the value of the M(X) and the complement of C from the value of the M(X), and stores the results into the M(X) and the C. The content of A is not changed except the status flag. The M(X) is the memory content of the address where the X appoints.

Status flag
N : N is when the bit 7 is 1 after the execution; otherwise N is 0
V : V is 1 when the operation result exceeds +127 or −128; otherwise V is 0.
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the operation result is 0; otherwise Z is 0.
C : C is 1 when the operation result is equal to 0 or greater than 0; otherwise C is 0 and shows the borrow is generated.

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Immediate	ΔSBCΔ#$nn	E9$_{16}$, nn$_{16}$	2	2
Zero page	ΔSBCΔ$zz	E5$_{16}$, zz$_{16}$	2	3
Zero page X	ΔSBCΔ$zz,X	F5$_{16}$, zz$_{16}$	2	4
Absolute	ΔSBCΔ$hhll	ED$_{16}$, ll$_{16}$, hh$_{16}$	3	4
Absolute X	ΔSBCΔ$hhll,X	FD$_{16}$, ll$_{16}$, hh$_{16}$	3	5
Absolute Y	ΔSBCΔ$hhll,Y	F9$_{16}$, ll$_{16}$, hh$_{16}$	3	5
(Indirect X)	ΔSBCΔ($zz,X)	E1$_{16}$, zz$_{16}$	2	6
(Indirect Y)	ΔSBCΔ($zz),Y	F1$_{16}$, zz$_{16}$	2	6

(Note 1) Add 3 to cycle number when T is 1.
(Note 2) When the SBC instruction is used in decimal operation mode the following rule must be applied if a SEC, CLC, SED or CLD instructions are used immediately afterwards: one dummy instruction (e.g. NOP) must be inserted between the ADC and clear/set carry (or clear/set decimal mode) instruction.

Operation : (Ai) ← 1, or
(Mi) ← 1

Function : This instruction sets the appointed bit i of the A or the M to 1.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Accumulator bit	ΔSEBΔi,A	(20•i+B)16	1	2
Zero page bit	ΔSEBΔi,$zz	(20•i+F)16, zz16	2	5

SET CARRY FLAG

Operation : (C) ← 1

Function : This instruction sets the C to 1.

Status flag
N : No change
V : No change
T : No change
B : No change
I : No change
D : No change
Z : No change
C : 1

Addressing mode	Statement	Machine code	Byte number	Cycle number
Implied	ΔSEC	38_{16}	1	2

Operation : (D) ← 1

Function : This instruction set the D to 1.

Status flag
N : No change
V : No change
T : No change
B : No change
I : No change
D : 1
Z : No change
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔSED	$F8_{16}$	1	2

SET INTERRUPT DISABLE FLAG

Operation : (I) ← 1

Function : This instruction sets the I to 1.

Status flag
N : No change
V : No change
T : No change
B : No change
I : 1
D : No change
Z : No change
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔSEI	78_{16}	1	2

SET TRANSFER FLAG

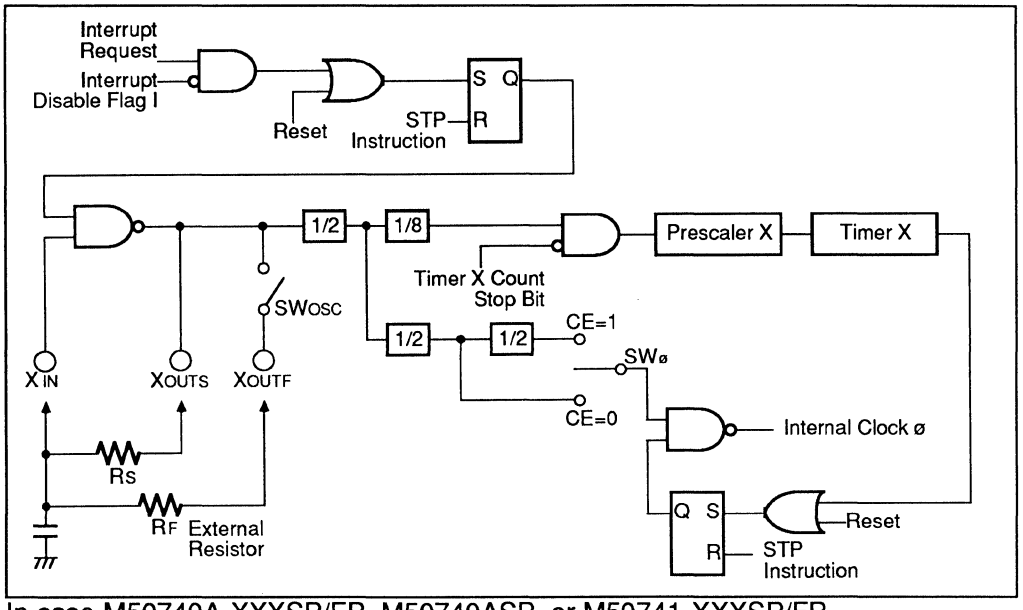
Operation : (T) ← 1

Function : This instruction sets the T to 1.

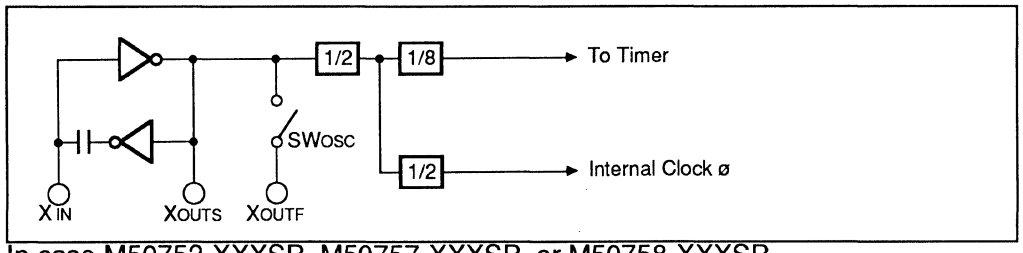
Status flag
N : No change
V : No change
T : 1
B : No change
I : No change
D : No change
Z : No change
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔSET	32_{16}	1	2

Operation : Releases the connection between the oscillator output and pin XOUTF pin.



In case M50740A-XXXSP/FP, M50740ASP, or M50741-XXXSP/FP



In case M50752-XXXSP, M50757-XXXSP, or M50758-XXXSP

Function : When CR clock generating circuit is formed, executing the SLW instruction is lowered the oscillation than FST instruction executing, because of the external resistance value is only Rs.

Status flag : No change

Addressing mode	Statement	Machine number	Byte number	Cycle number
Implied	ΔSLW	$C2_{16}$	1	2

(Note 1) This instruction is not for any other models than M50740A-XXXSP/FP, M50740ASP, M50741-XXXSP/FP, M50752-XXXSP, M50757-XXXSP and M50758-XXXSP.

(Note 2) The SWosc closes at reset.

STORE ACCUMULATOR IN MEMORY

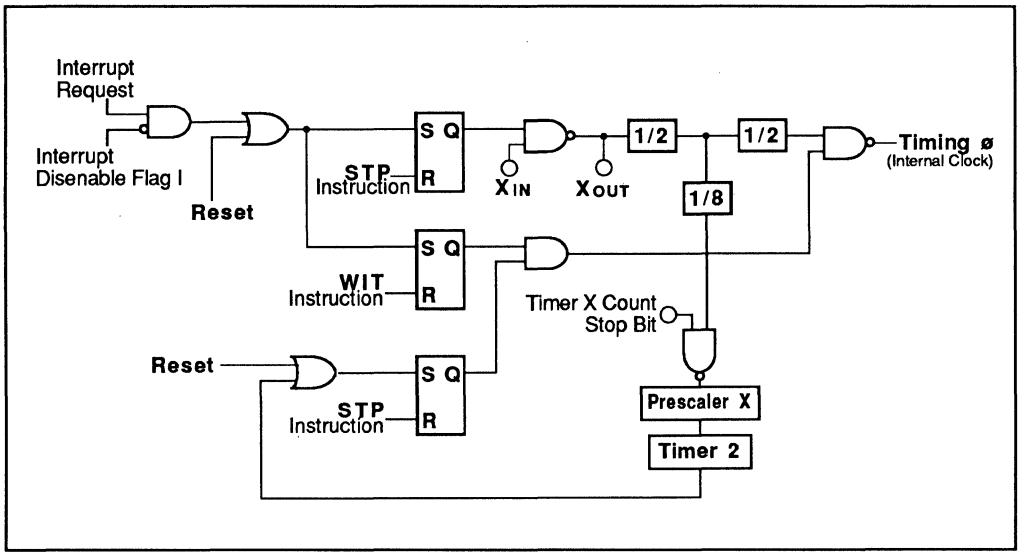
Operation : (M) ← (A)

Function : This instruction stores the A content into the M. The contents of A is not changed at the time.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	ΔSTAΔ$zz	85_{16}, zz_{16}	2	4
Zero page X	ΔSTAΔ$zz,X	95_{16}, zz_{16}	2	5
Absolute	ΔSTAΔ$hhll	$8D_{16}$, ll_{16}, hh_{16}	3	5
Absolute X	ΔSTAΔ$hhll,X	$9D_{16}$, ll_{16}, hh_{16}	3	6
Absolute Y	ΔSTAΔ$hhll,Y	99_{16}, ll_{16}, hh_{16}	3	6
(Indirect X)	ΔSTAΔ($zz,X)	81_{16}, zz_{16}	2	7
(Indirect Y)	ΔSTAΔ($zz),Y	91_{16}, zz_{16}	2	7

Operation : CPU ← Stand-by state (Oscillation stopped)



Function : This instruction resets the oscillation control F/F and the oscillation stops. To wake up from this mode, reset or interrupt input is needed. CPU starts its function after the timer X over flows (comes to the terminal count). All registers or internal memory contents except Timer X will not change during this mode. (Of course needs V_{DD}).

atus flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔSTP	42_{16}	1	2

(Note 1) This instruction is not provided for M50752-XXXSP, M50757-XXXSP, and M50758-XXXSP.

(Note 2) This circuit diagram a little differ by any other models. For more detail, see the "MITSUBISHI SEMICONDUCTORS SINGLE-CHIP MICROCOMPUTERS DATA BOOK".

Operation : (M) ← (X)

Function : This instruction stores the X content into the M; X is not changed.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	ΔSTXΔ$zz	86_{16}, zz_{16}	2	4
Zero page Y	ΔSTXΔ$zz,Y	96_{16}, zz_{16}	2	5
Absolute	ΔSTXΔ$hhll	$8E_{16}$, ll_{16}, hh_{16}	3	5

Operation : (M) ← (Y)

Function : This instruction stores the Y content into the M; Y is not changed.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	ΔSTYΔ$zz	84_{16}, zz_{16}	2	4
Zero page X	ΔSTYΔ$zz,X	94_{16}, zz_{16}	2	5
Absolute	ΔSTYΔ$hhll	$8C_{16}$, ll_{16}, hh_{16}	3	5

TRANSFER ACCUMULATOR TO INDEX REGISTER X

Operation : (X) ← (A)

Function : This instruction transfers the A content to the X; The contents of A are not changed.

Status flag
N : N is 1 when the bit 7 is 1 after the transfer; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the result after the transfer is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔTAX	AA_{16}	1	2

TRANSFER ACCUMULATOR TO INDEX REGISTER Y

Operation : (Y) ← (A)

Function : This instruction transfers the A content to the Y; The content of A are not changed.

Status flag
- **N :** N is 1 when the bit 7 is 1 after the transfer; otherwise N is 0.
- **V :** No change
- **T :** No change
- **B :** No change
- **I :** No change
- **D :** No change
- **Z :** Z is 1 when the result of the transfer is 0; otherwise Z is 0.
- **C :** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔTAY	$A8_{16}$	1	2

TEST FOR NEGATIVE OR ZERO

Operation : (M) = 0 ?

Function : This instruction tests whether or not the M contents are zero and modifies the N and Z.

Status flag
N : N is when the bit 7 of M is 1; otherwise N is 0.
V : No change
T : No change
B : No change
I : No change
D : No change
Z : Z is 1 when the M content is 0; otherwise Z is 0.
C : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Zero page	ΔTSTΔ$zz	64$_{16}$, zz$_{16}$	2	3

TRANSFER STACK POINTER TO INDEX REGISTER X

Operation : (X) ← (S)

Function : This instruction transfers the S contents to X.

Status flag
N : N is 1 when the bit 7 is 1 after the execution otherwise N
V : is 0.
T : No change
B : No change
I : No change
D : No change
Z : No change
C : Z is 1 when the execution result is 0; otherwise Z is 0.
No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔTSX	BA_{16}	1	2

TRANSFER INDEX REGISTER X TO ACCUMULATOR

Operation : (A) ← (X)

Function : This instruction transfers the X contents to A.

Status flag
N : N is 1 when the bit 7 after the execution is 1; otherwise N is 0.
V: No change
T: No change
B: No change
I: No change
D: No change
Z: Z is 1 when the execution result is 0; otherwise Z is 0.
C: No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔTXA	$8A_{16}$	1	2

TXS TXS

TRANSFER INDEX REGISTER X TO STACK POINTER

Operation : (S) ← (X)

Function : This instruction transfers the X contents to S.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔTXS	$9A_{16}$	1	2

TRANSFER INDEX REGISTER Y TO ACCUMULATOR

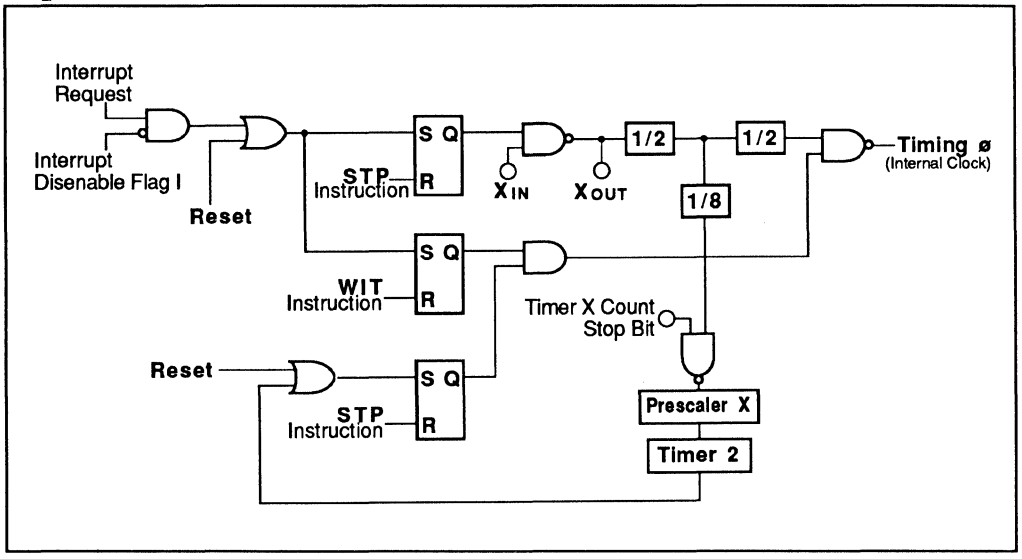
Operation : (A) ← (Y)

Function : This instruction transfers the Y contents to A.

Status flag :
- **N :** N is 1 when the bit 7 after the execution is 1; otherwise N is 0.
- **V:** No change
- **T:** No change
- **B:** No change
- **I:** No change
- **D:** No change
- **Z:** Z is 1 when the execution result is 0; otherwise Z is 0.
- **C:** No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔTYA	98_{16}	1	2

Operation : CPU ← Wait state



Function : This instruction resets the internal clock control F/F and internal clock source is stopped after this instruction. To wake up from this wait mode, reset or interrupt inputs are needed. Oscillation will not stop during this wait mode.

Status flag : No change

Addressing mode	Statement	Machine codes	Byte number	Cycle number
Implied	ΔWIT	$C2_{16}$	1	2

(Note 1) This instruction is not provided for M50740A-XXXSP/FP, M50740ASP, M50741-XXXSP/FP, M50752-XXXSP, M50757-XXXSP and M50758-XXXSP.
(Note 2)This circuit diagram a little differ by any other models. For more detail, see the "MITSUBISHI SEMICONDUCTORS SINGLE-CHIP MICROCOMPUTERS".

4.Notes for Programing

4.1 Note about Processor Status Register

4.1.1 Initialize the processor status register

The contents of processor status register(PS) is indeterminate after reset except the I flag. Therefore the flags that influence on execution of program, are required to initialize.

Especially initialize the T flag, and D flag are influenced on operation directory.

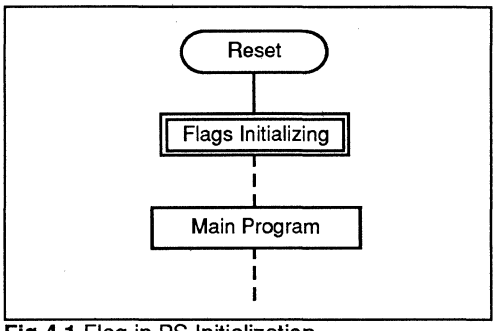


Fig.4.1 Flag in PS Initialization

4.1.2 How to refer to the processor status register

(1)Execute a **PHP** instruction to save the of processor status register(PS) into stack(S+1).

(2)Read from the contents of stack(S+1).

(3)To restore the previous PS from stack, execute a **PLP** instruction. However, a **NOP** instruction is needed after execute the **PLP** instruction.

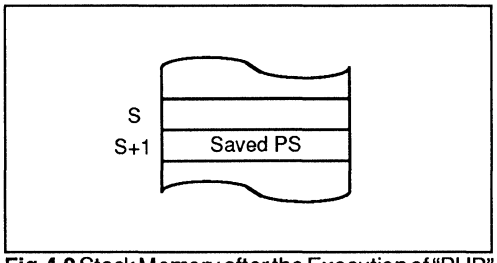


Fig.4.2 Stack Memory after the Execution of "PHP"

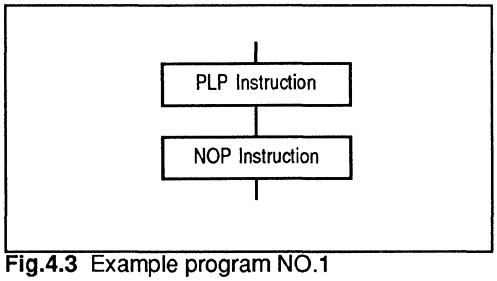


Fig.4.3 Example program NO.1

4.2 Notes for Interrupt Function

(1)More than one instruction cycle is needed to execute the **BBC** or **BBS** instruction after switching the value interrupt request bit.

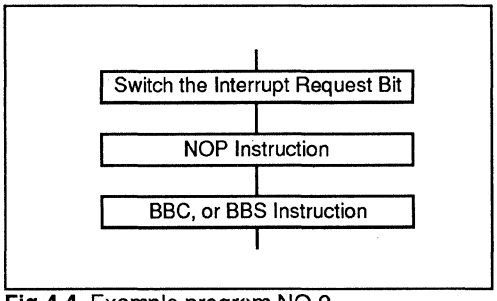


Fig.4.4 Example program NO.2

(2)Even though the **BBC** and **BBS** instructions are executed just after the interrupt request bits are modified (by the program),those instructions are only valid for the contents before the modification.

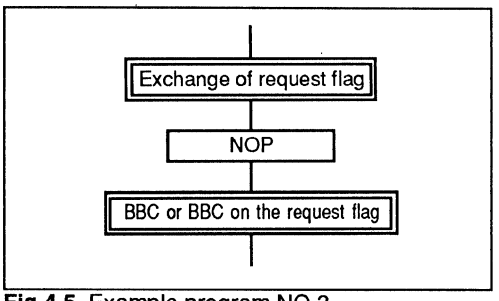


Fig.4.5 Example program NO.3

(3)Checking the condition of B flag in the interrupt routine can be used to discriminate between the BRK interrupt and the lowest priority interrupt.

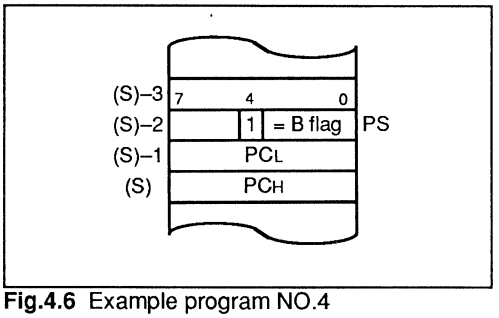


Fig.4.6 Example program NO.4

4.3 Note for Decimal Operation

(1)In decimal operation mode(at D flag="1"), more than one instruction cycle is needed before **SEC**, **CLC**, or **CLD** instruction after the **ADC** or **SBC** instruction.

(2)The N (Negative),V(Overflow),and Z(Zero) flags are ignored during decimal mode.

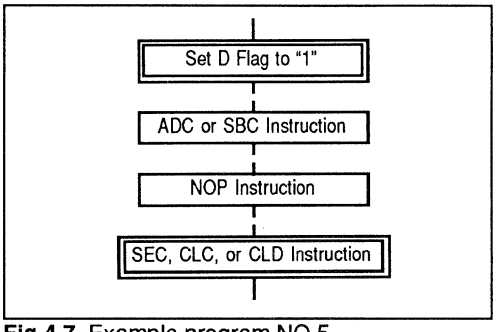


Fig.4.7 Example program NO.5

4. 4 Note for Stop Mode

The time X gate flag (bit 5 at $00FF_{16}$) must be closed before a stop instruction so that the processor can be waked up later (**CLB** 5 $FF before STP).

4.5 Notes for JMP instruction

Don't designate the last address of each page($XXFF_{16}$) as the indirect address when using **JMP** instruction in the indirect addressing mode.

APPENDIX 1. Instruction cycles in each addressing mode

Internal clock ø, which has a frequency one fourth of the f(XIN) (oscillating frequency), controls the system timing of series MELPS 740.
The SYNC signal is output in every instruction fetch cycle.
In this period, value of PC(program counter) is also output.
Op code is fetched during the next half-period of ø.
Instruction decoder of CPU decodes this op code and determines the following procedure.
Instruction timings of all addressing modes are described on the next pages.
In these figures, ø, SYNC, R/$\overline{W}$($\overline{RD}$, $\overline{WR}$), ADDR(ADDRH, ADDRL), and DATA are internal signals of the single-chip microcomputer.
Therefore, these signals can be investigated only in the microprocessor mode.

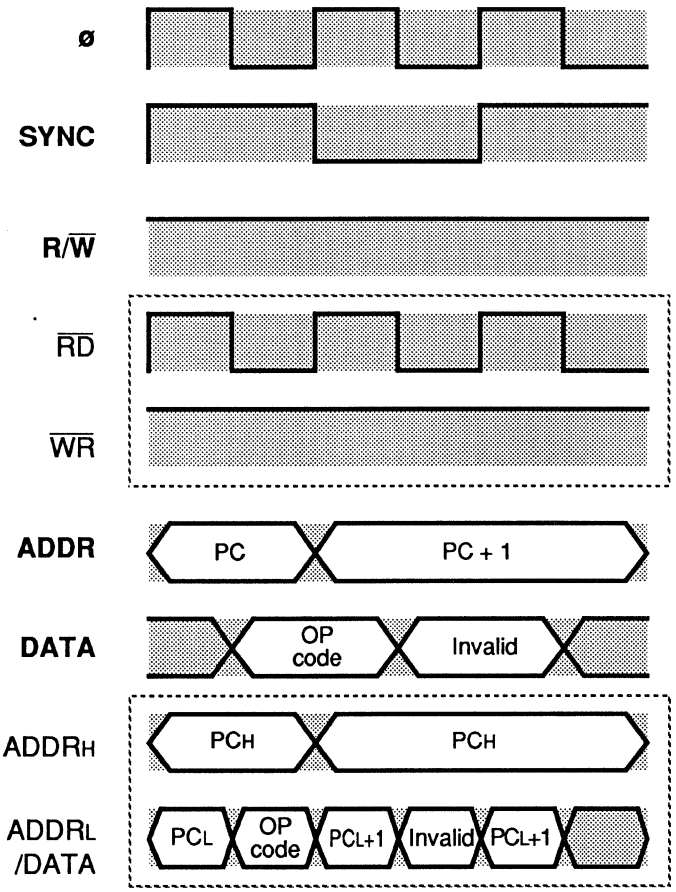
- If some kind of the single-chip microcomputer that outputs the address signals separately (M50734SP is a case in point), the each signal timing are shown in the dashed line box as ADDRH and ADDRL. And in this case, the data signal timing(DATA) is shown with ADDRL(ADDRL/DATA).
- If some kind of the single-chip microcomputer that outputs the read and write signals separately (M37450 is a case in point), the each signal timing are shown in the dashed line box as $\overline{RD}$ and $\overline{WR}$.

IMPLIED

Instructions : **CLC**
CLD
CLI
CLT
CLV
DEX
DEY
INX
INY
NOP
SEC
SED
SEI
SET
TAX
TAY
TSX
TXA
TXS
TYA

Byte length : **1**
Cycle number : **2**

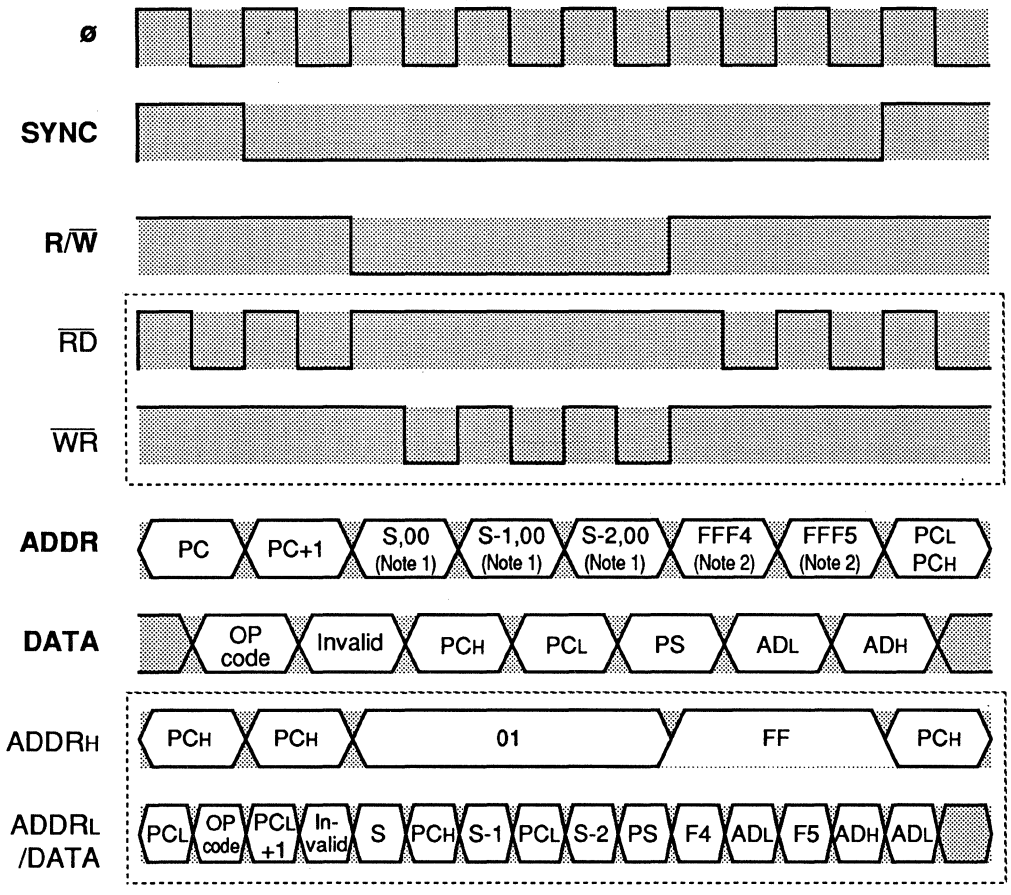
Timing :



IMPLIED

Instruction : **BRK**
Byte length : **1**
Cycle number : **7**

Timing :



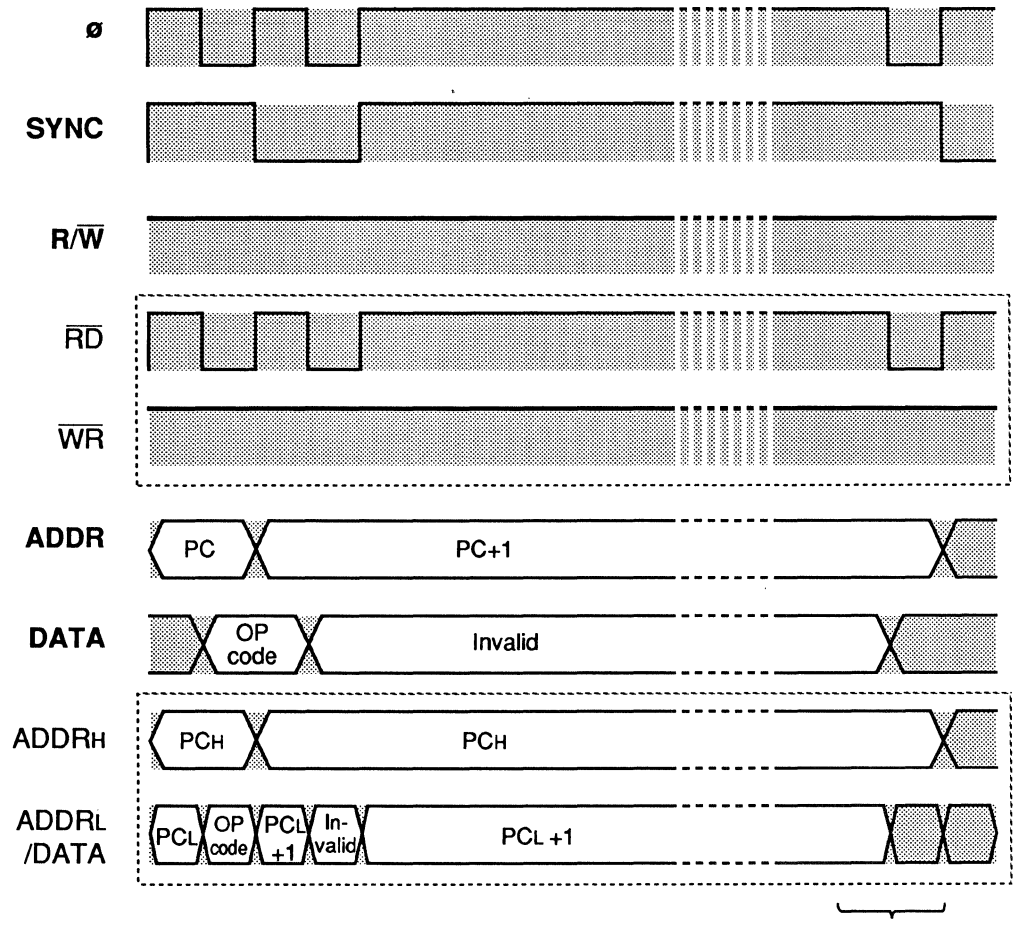
Note 1 : Some kind of types are "01" or content of SPS flag.
Note 2 : Some kind of types differ the address.

IMPLIED

Instructions : **STP**
WIT

Byte length : **1**

Timing :

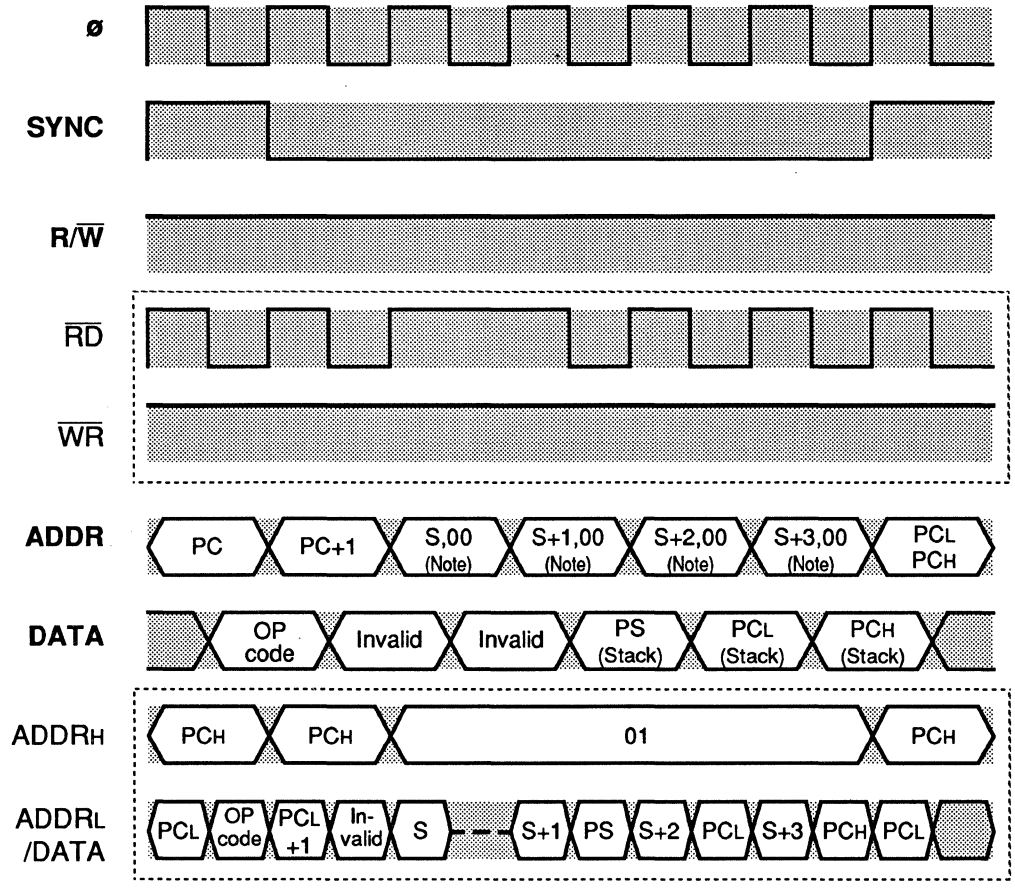


Return from standby state is excuted by external interrupt.
Return from wait state is excuted by internal or external interrupt.

IMPLIED

Instruction : **RTI**
Byte length : **1**
Cycle number : **6**

Timing :

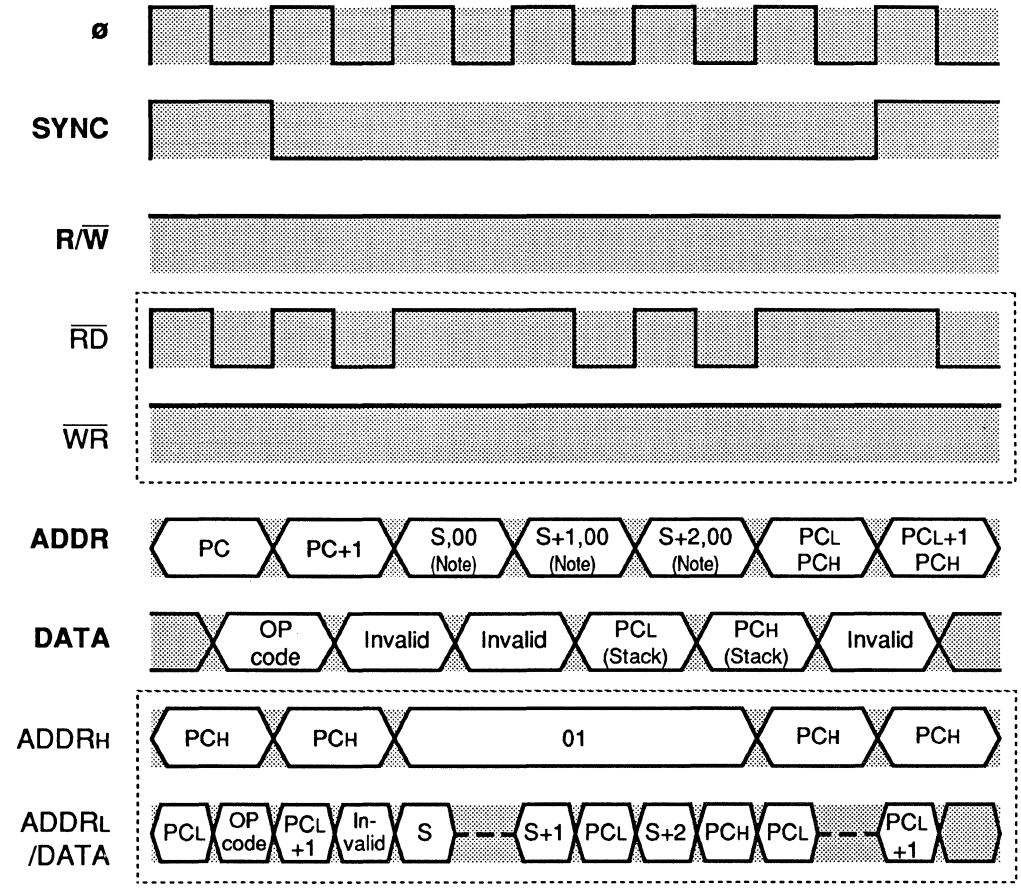


Note : Some kind of types are "01" or content of SPS flag.

IMPLIED

Instruction : **RTS**
Byte length : **1**
Cycle number : **6**

Timing :

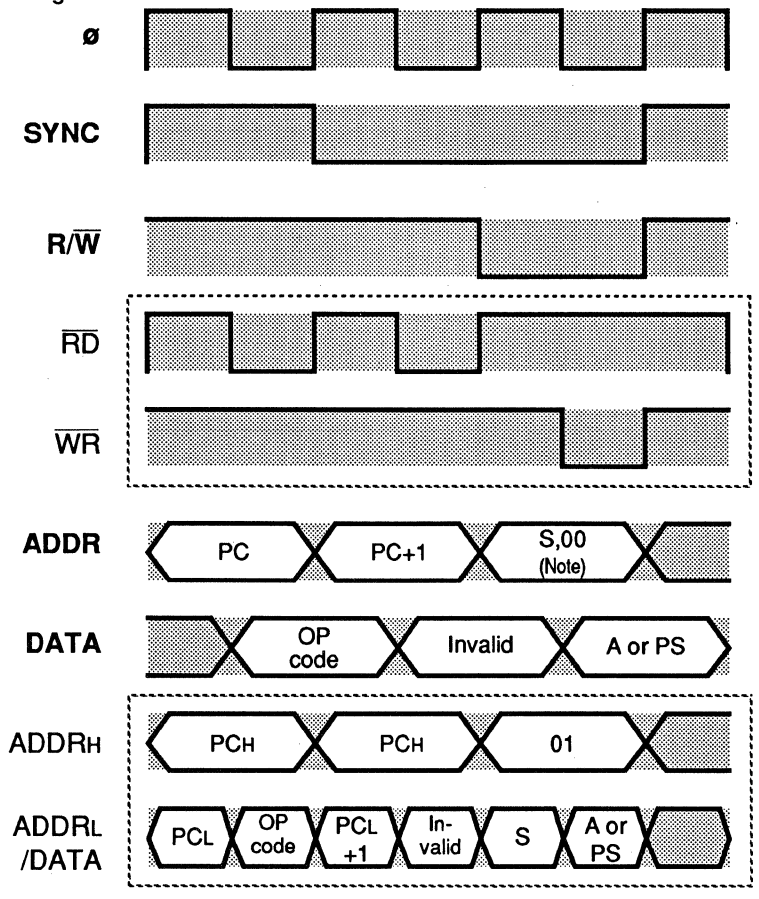


Note : Some kind of types are "01" or SPS flag.

IMPLIED

Instructions : **PHA**
PHP
Byte length : **1**
Cycle number : **3**

Timing :

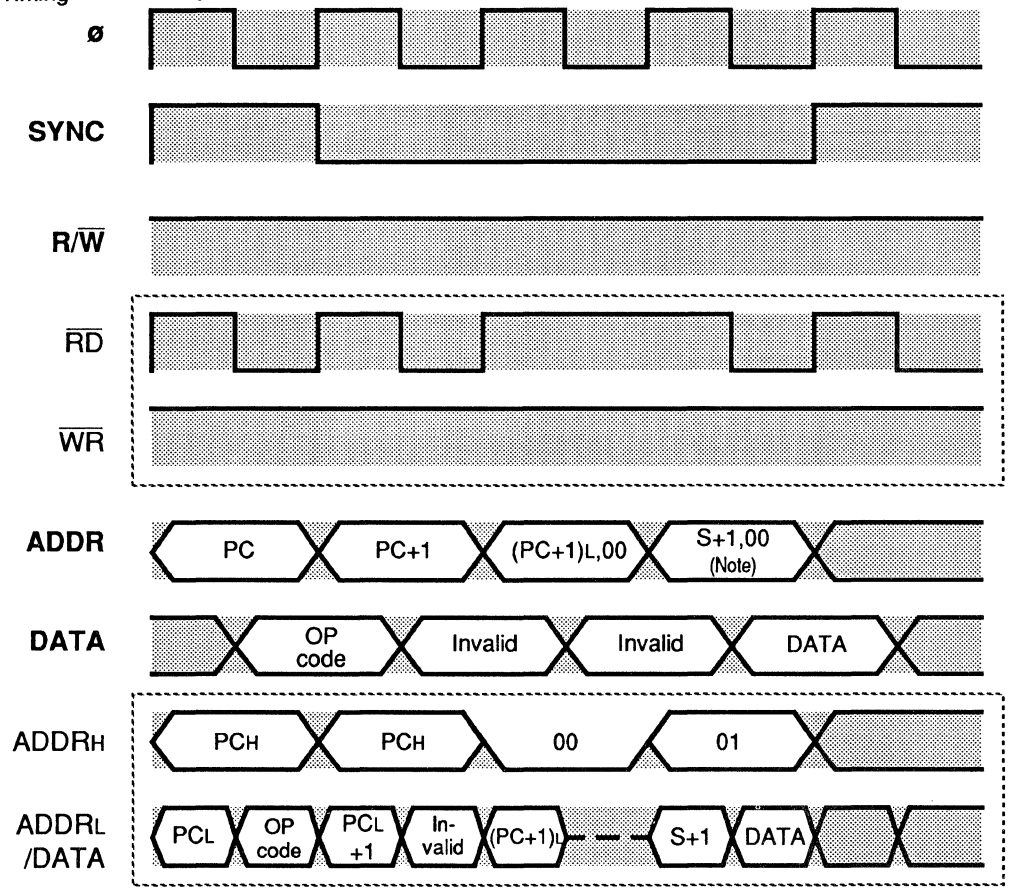


Note : Some kind of types are "01" or content of SPS flag.

IMPLIED

Instructions : **PLA**
PLP
Byte length : **1**
Cycle number : **4**

Timing :



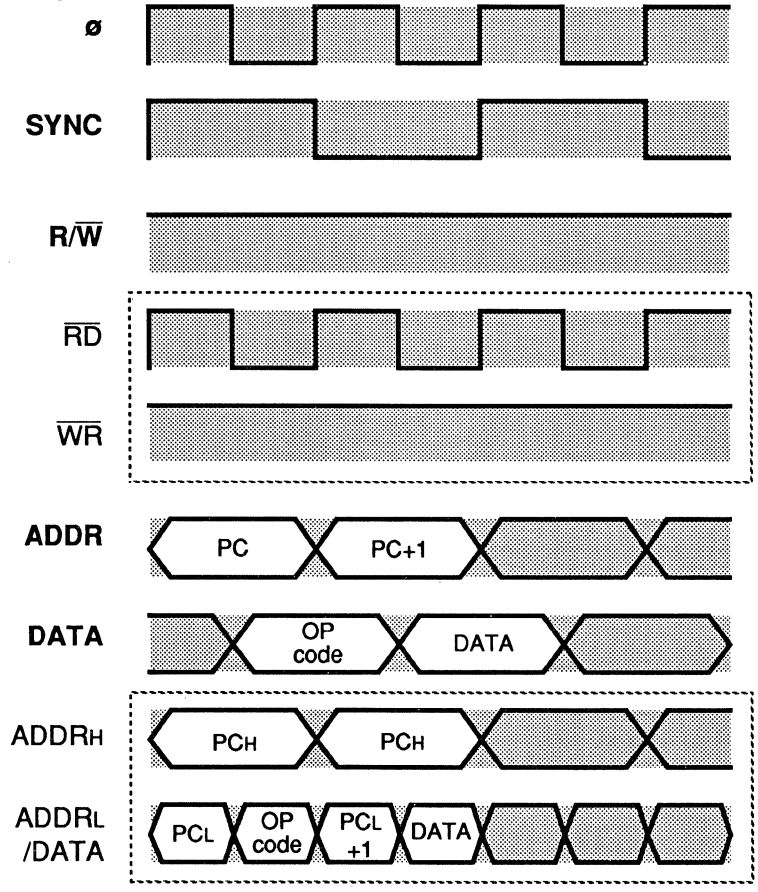
Note : Some kind of types are "01" or content of SPS flag.

IMMEDIATE

Instructions : **ADCΔ#$nn (T=0)**
ANDΔ#$nn (T=0)
CMPΔ#$nn (T=0)
CPXΔ#$nn
CPYΔ#$nn
EORΔ#$nn (T=0)
LDAΔ#$nn (T=0)
LDXΔ#$nn
LDYΔ#$nn
ORAΔ#$nn (T=0)
SBCΔ#$nn (T=0)

Byte length : **2**
Cycle number : **2**

Timing :



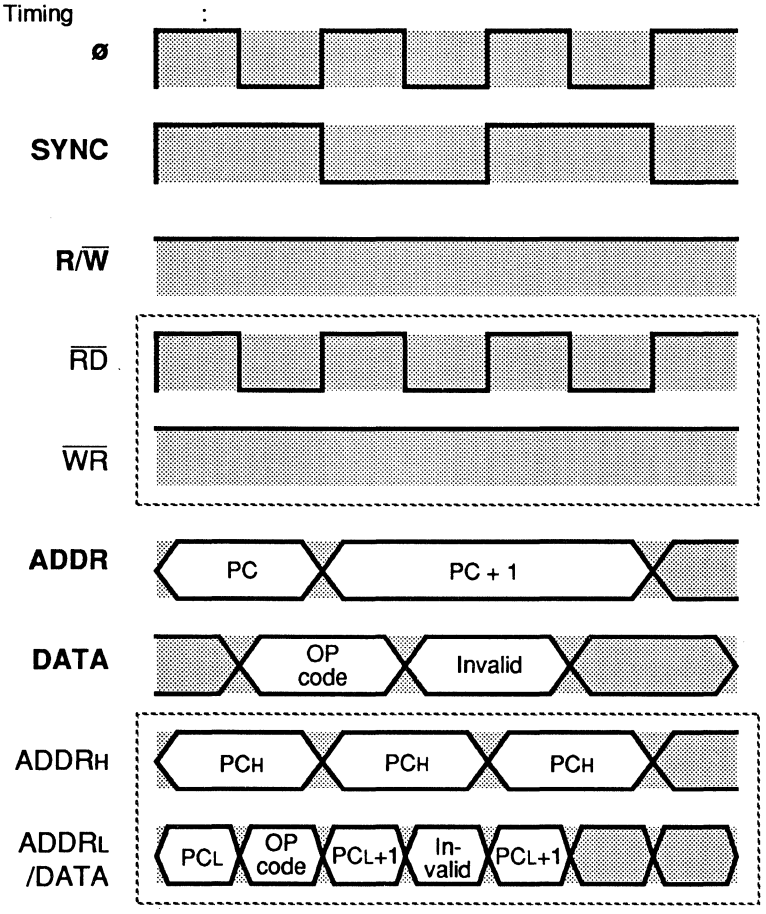
ACCUMULATOR

Instructions : **ASLΔA**
DECΔA
INCΔA
LSRΔA
ROLΔA
RORΔA

Byte length : **1**

Cycle number : **2**

Timing :

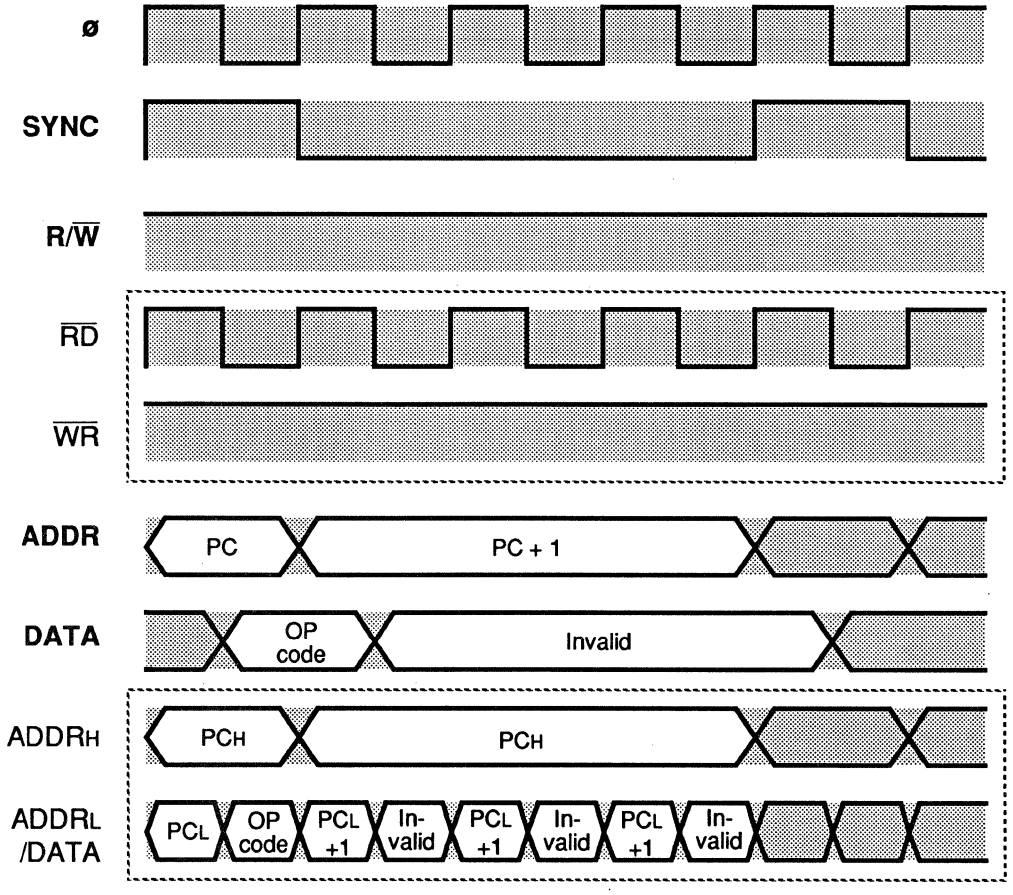


ACCUMULATOR BIT RELATIVE

Instructions : **BBCΔI,A,$hhll**
BBSΔI,A,$hhll
Byte length : **2**

(1) With no branch
Cycle number : **4**

Timing :

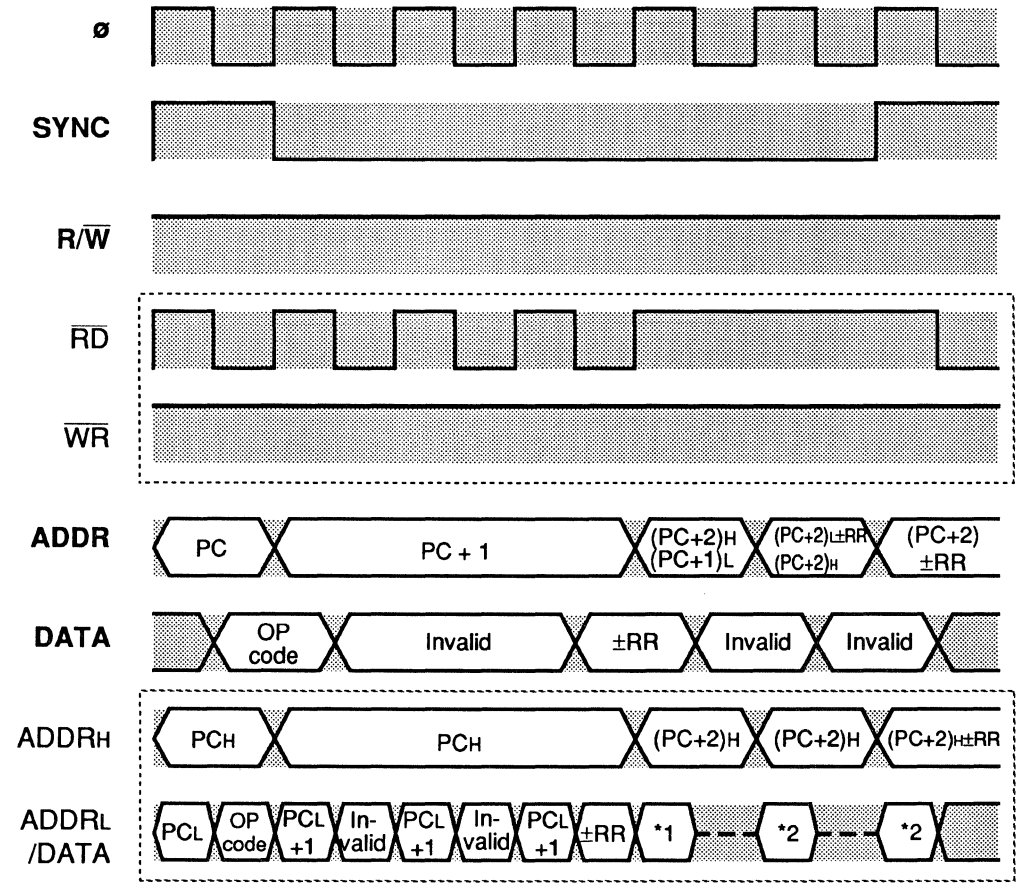


ACCUMULATOR BIT RELATIVE

Instructions : **BBCΔi,A,$hhll**
BBSΔi,A,$hhll
Byte length : **2**

(2) With branch
Cycle number : **6**

Timing :

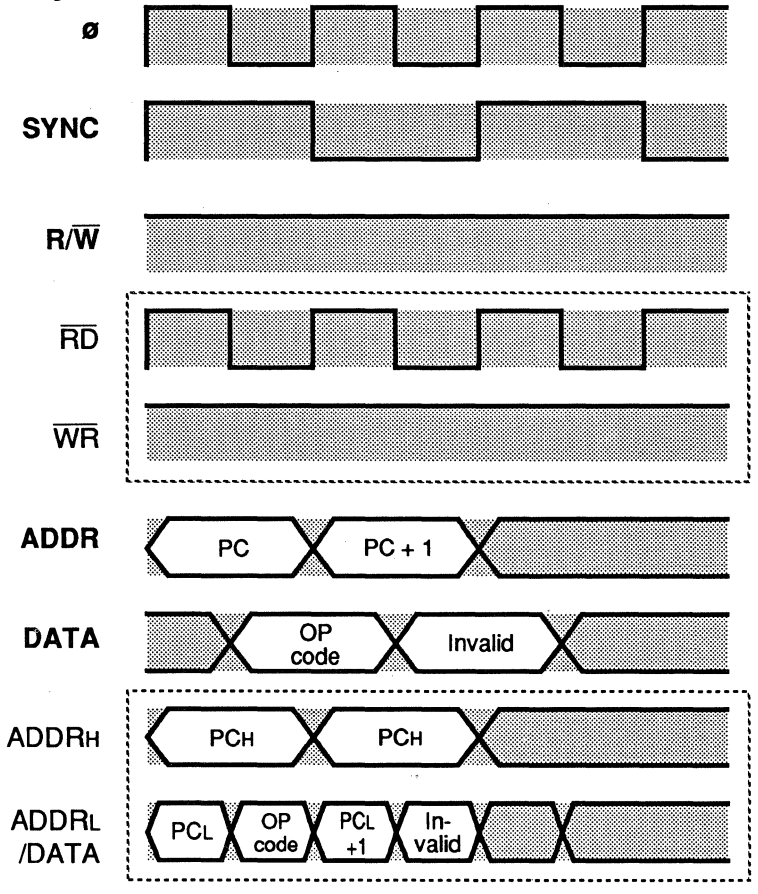


RR: Offset value
*1: (PC+1)L
*2: (PC+2)L±RR

ACCUMULATOR BIT

Instructions : **CLBΔI,A**
SEBΔI,A
Byte length : **1**
Cycle number : **2**

Timing :

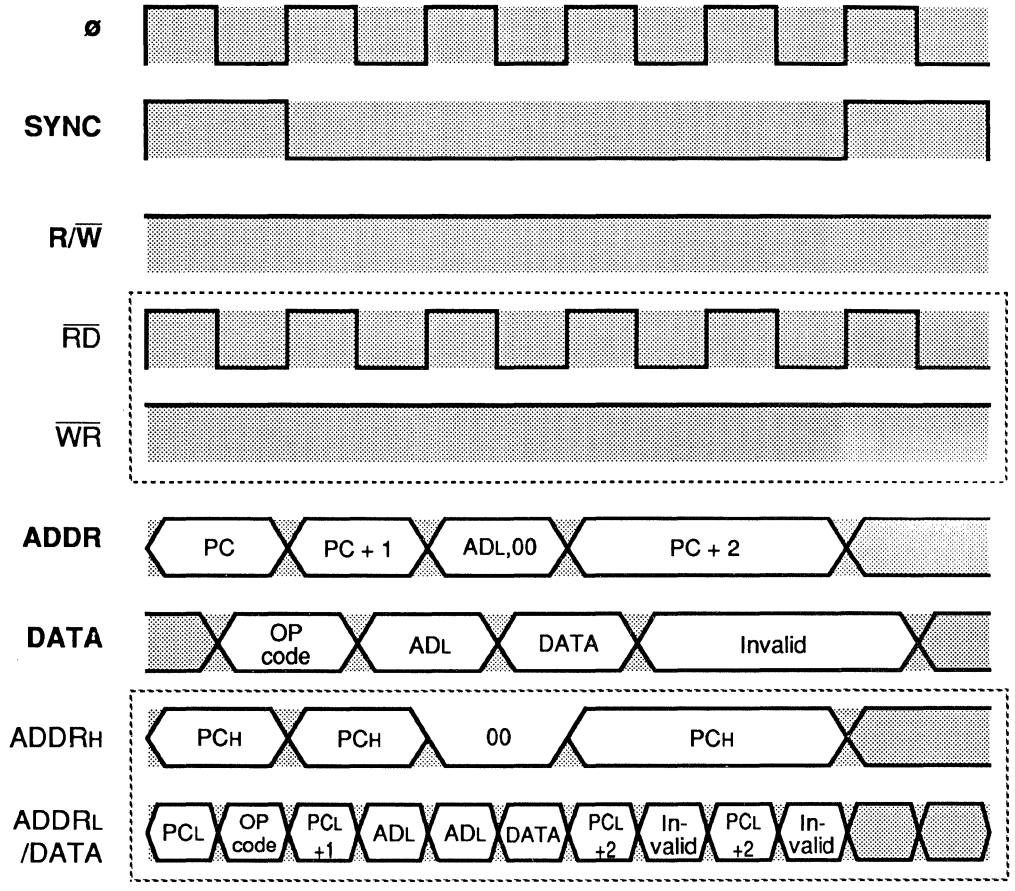


BIT RELATIVE

Instructions : **BBCΔI,$zz,$hhll**
BBSΔI,$zz,$hhll
Byte length : **3**

(1) With no branch
Cycle number : **5**

Timing :

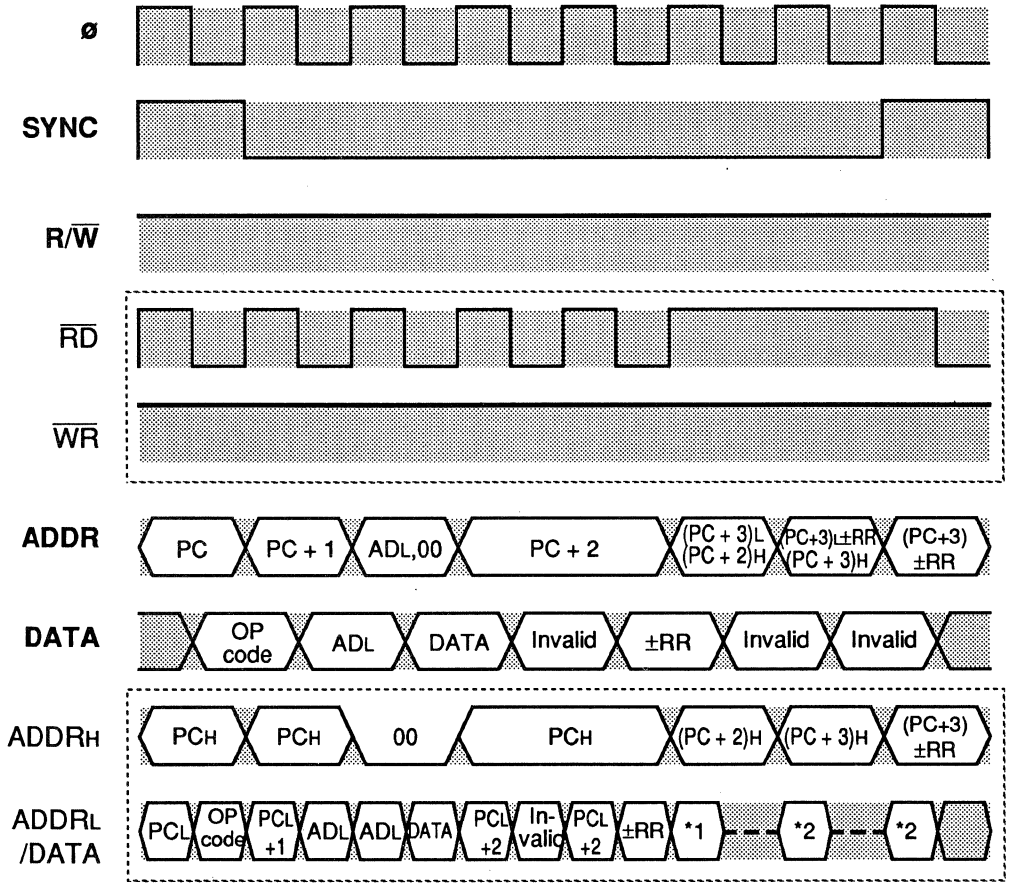


BIT RELATIVE

Instructions : **BBCΔi,$zz,$hhll**
BBSΔi,$zz,$hhll
Byte length : **3**

(2) With branch
Cycle number : **7**

Timing :

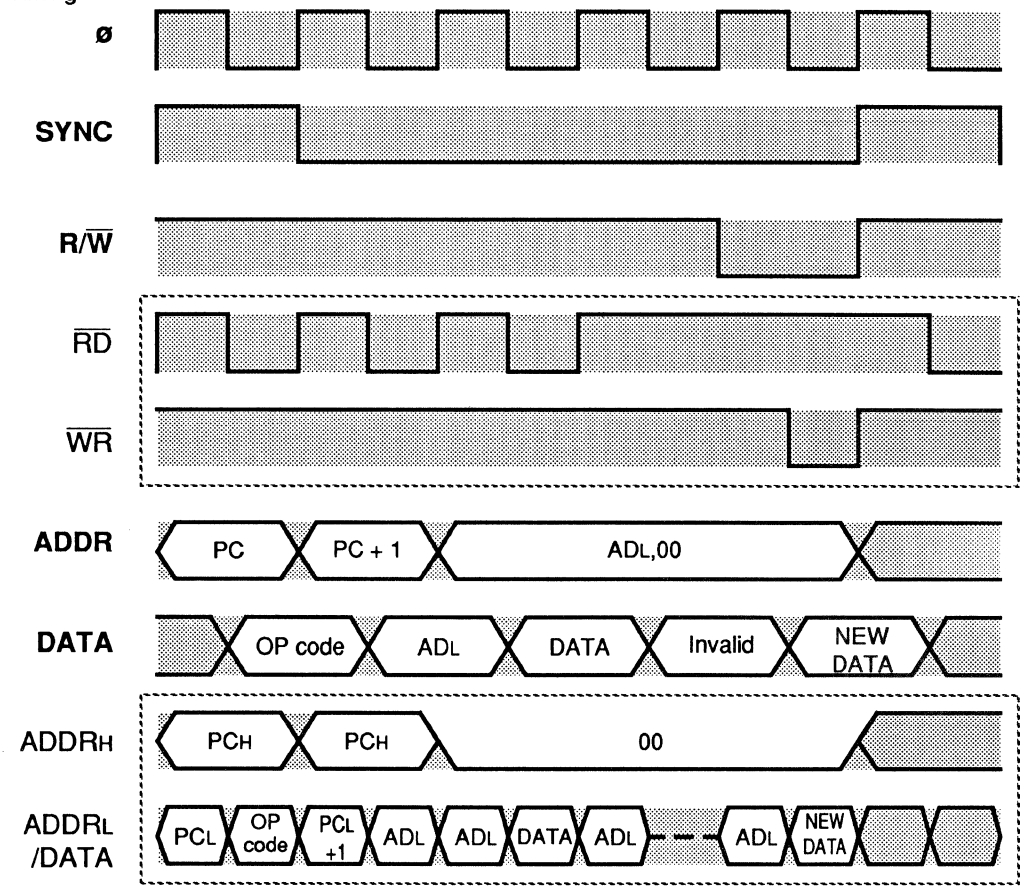


RR : Offset address
*1 : (PC+3)L
*2 : (PC+3)L±RR

ZERO PAGE BIT

Instructions : **CLBΔI,$zz**
SEBΔI,$zz
Byte length : **2**
Cycle number : **5**

Timing :

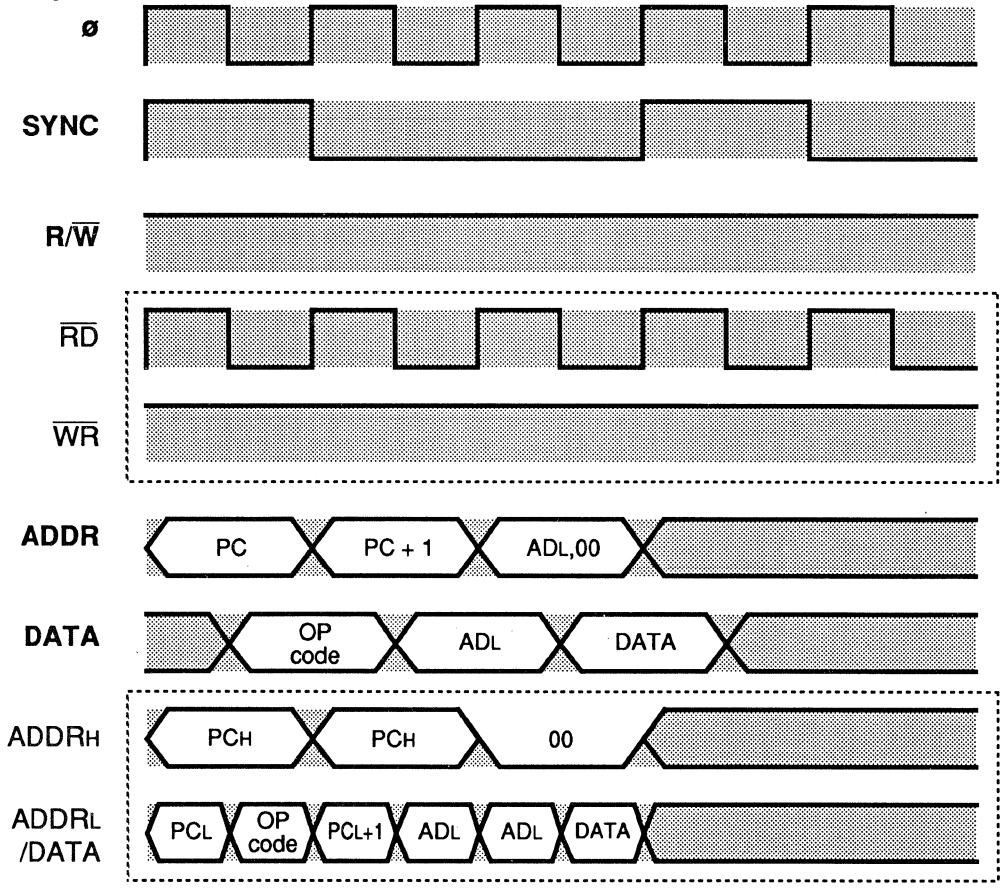


ZERO PAGE

Instructions : **ADCΔ$zz (T=0)**
ANDΔ$zz (T=0)
BITΔ$zz
CMPΔ$zz (T=0)
CPXΔ$zz
CPYΔ$zz
EORΔ$zz (T=0)
LDAΔ$zz (T=0)
LDXΔ$zz
LDYΔ$zz
ORAΔ$zz (T=0)
SBCΔ$zz (T=0)
TSTΔ$zz

Byte length : **2**
Cycle number : **3**

Timing :



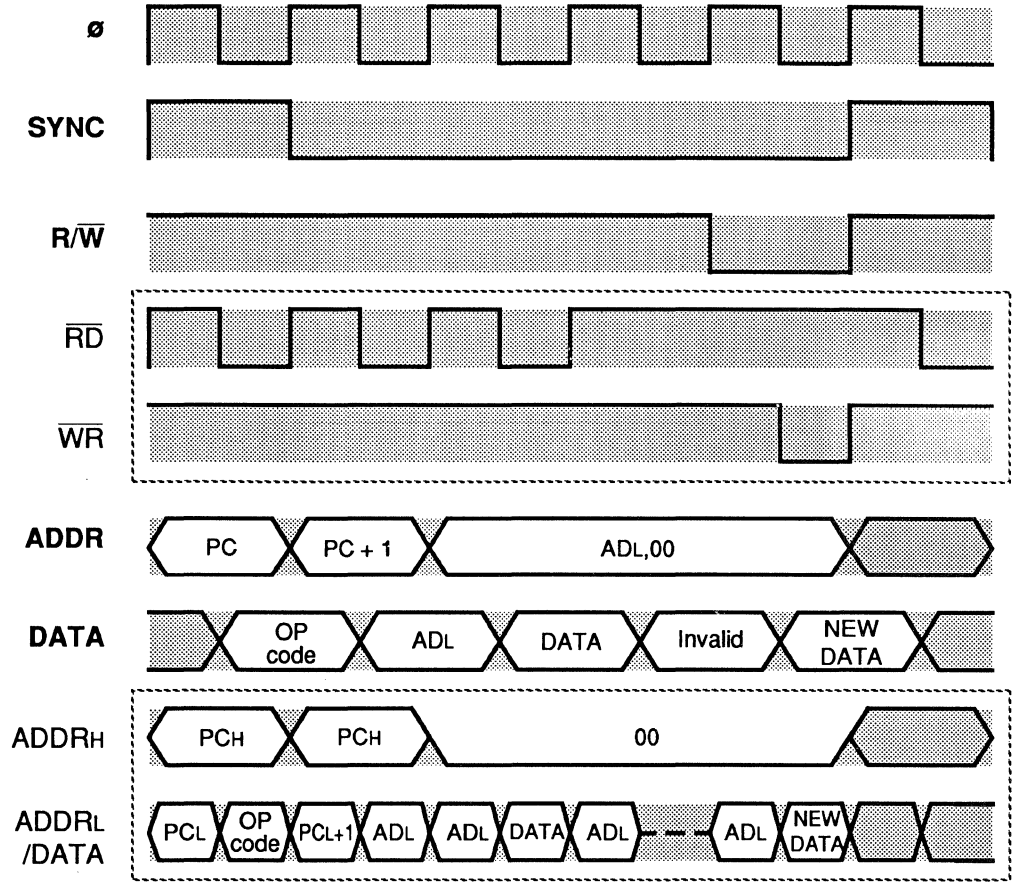
ZERO PAGE

Instructions : **ASLΔ$zz**
COMΔ$zz
DECΔ$zz
INCΔ$zz
LSRΔ$zz
ROLΔ$zz
RORΔ$zz

Byte length : **2**

Cycle number : **5**

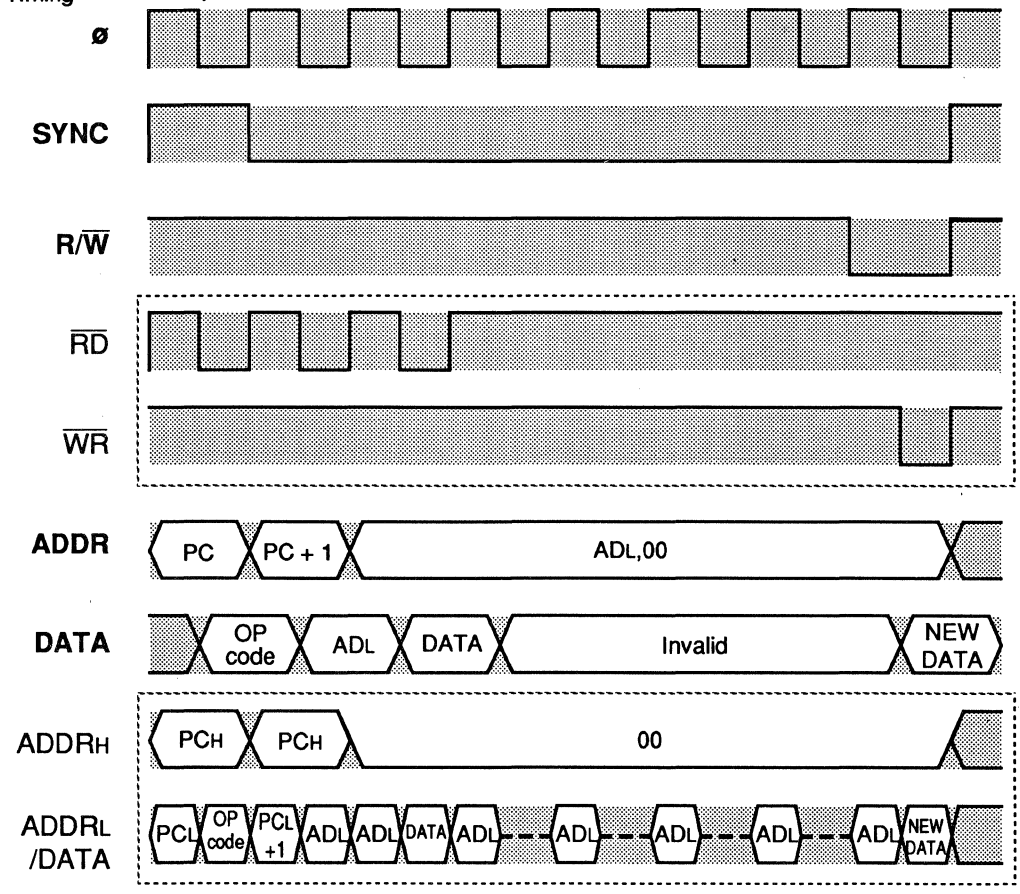
Timing :



ZERO PAGE

Instruction : **RRFΔ$zz**
Byte length : **2**
Cycle number : **8**

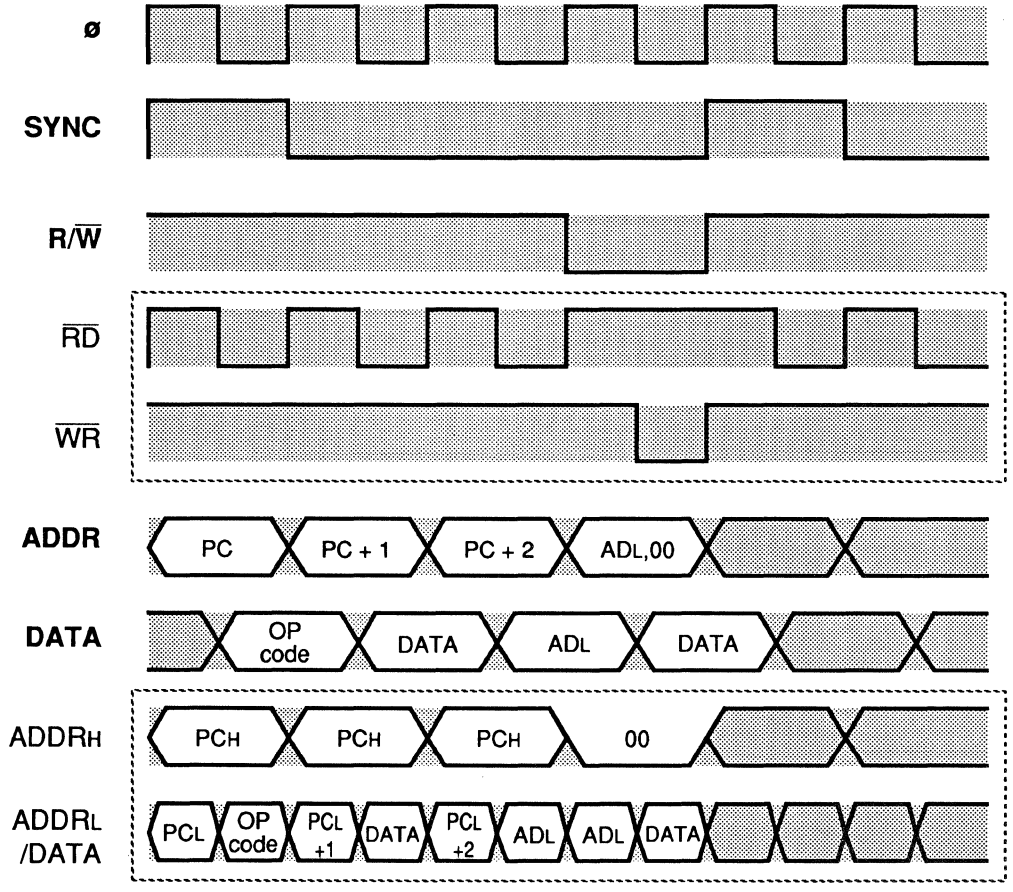
Timing :



ZERO PAGE

Instruction : **LDMΔ#$nn,$zz**
Byte length : **3**
Cycle number : **4**

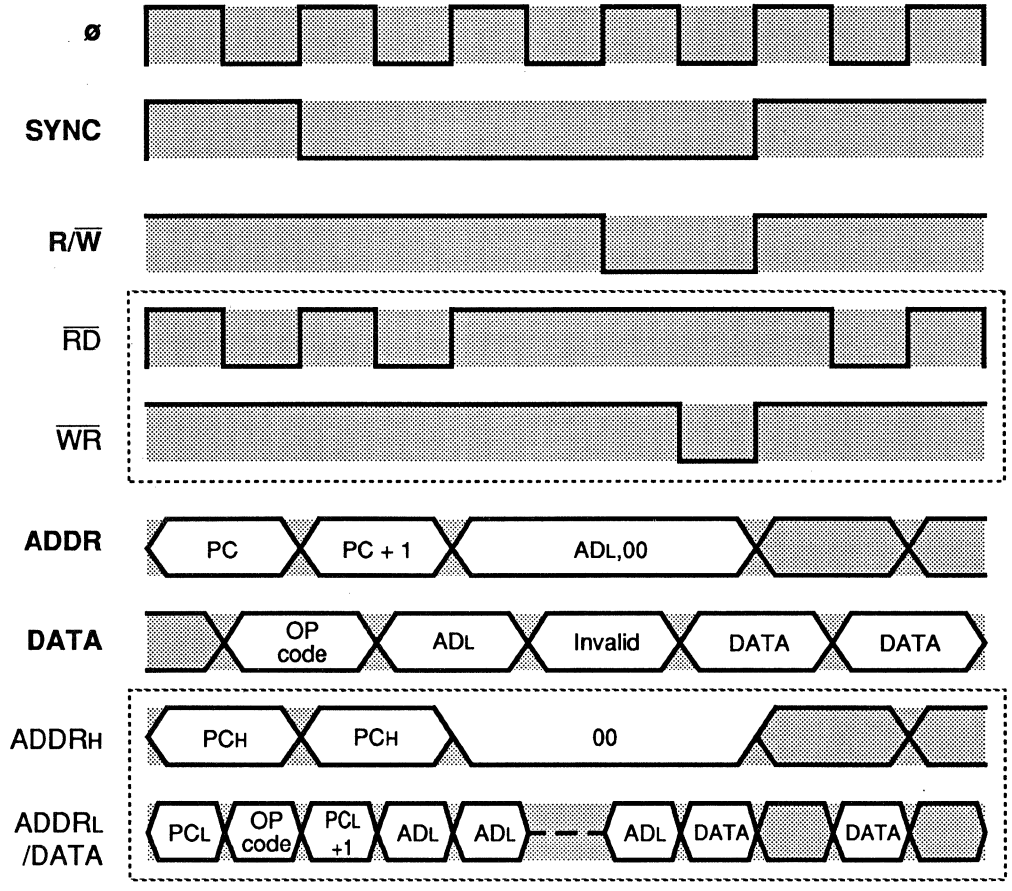
Timing :



ZERO PAGE

Instructions : **STAΔ$zz**
STXΔ$zz
STYΔ$zz
Byte length : **2**
Cycle number : **4**

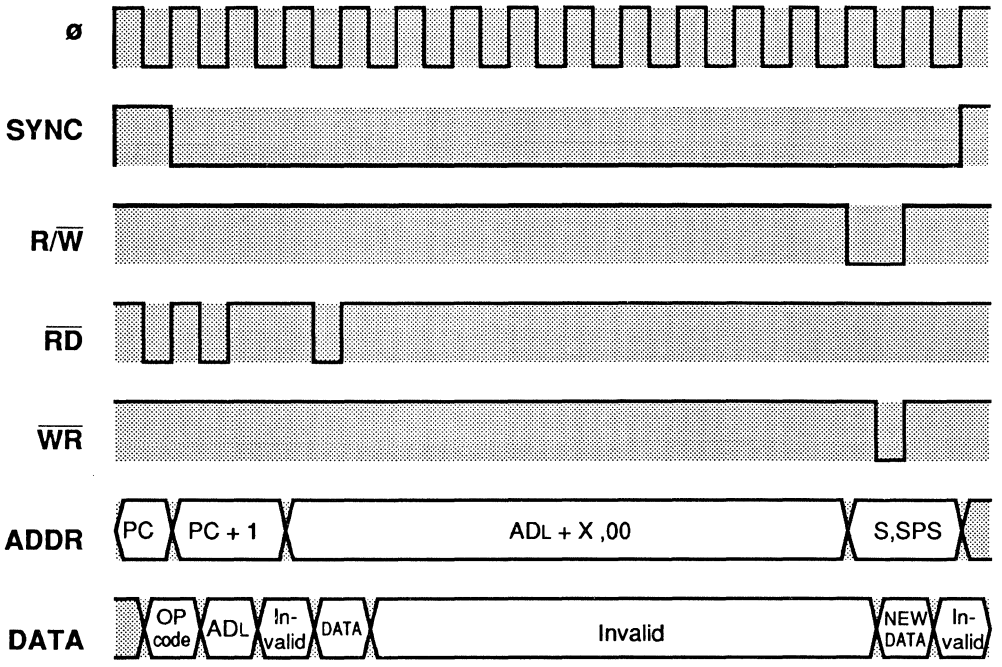
Timing :



ZERO PAGE X

Instruction : **MULΔ$zz,X** (Note)
Byte length : **2**
Cycle number : **15**

Timing :

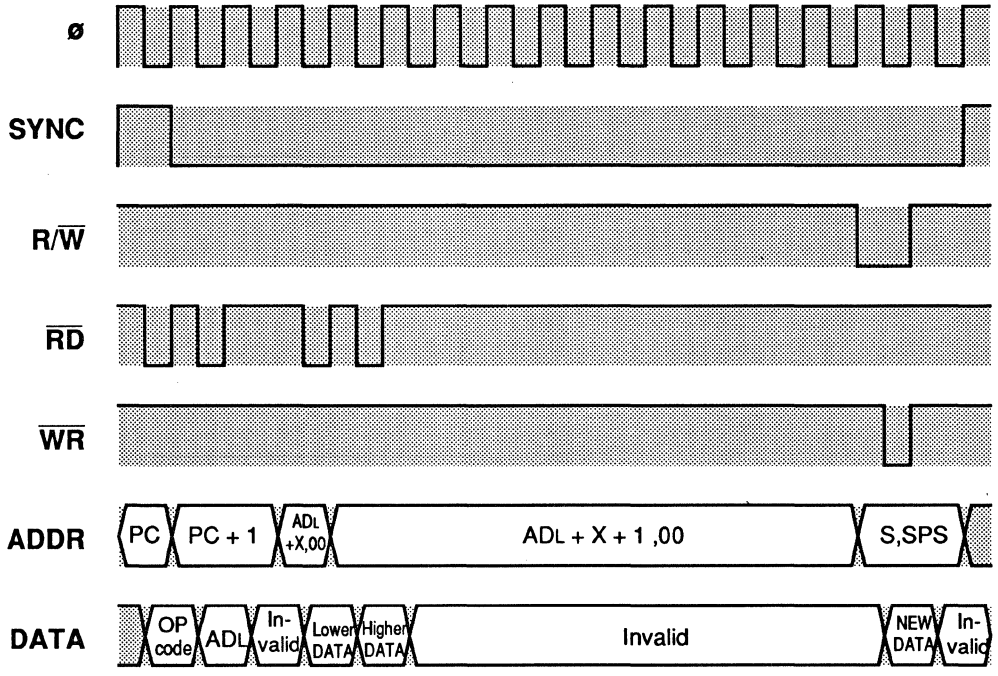


Note : This instruction can be used special types (M37450 is a case in point).

ZERO PAGE X

Instruction : **DIVΔ$zz,X** (Note)
Byte length : **2**
Cycle number : **16**

Timing :



Note : This instruction can be used special types (M37450 is a case in point).

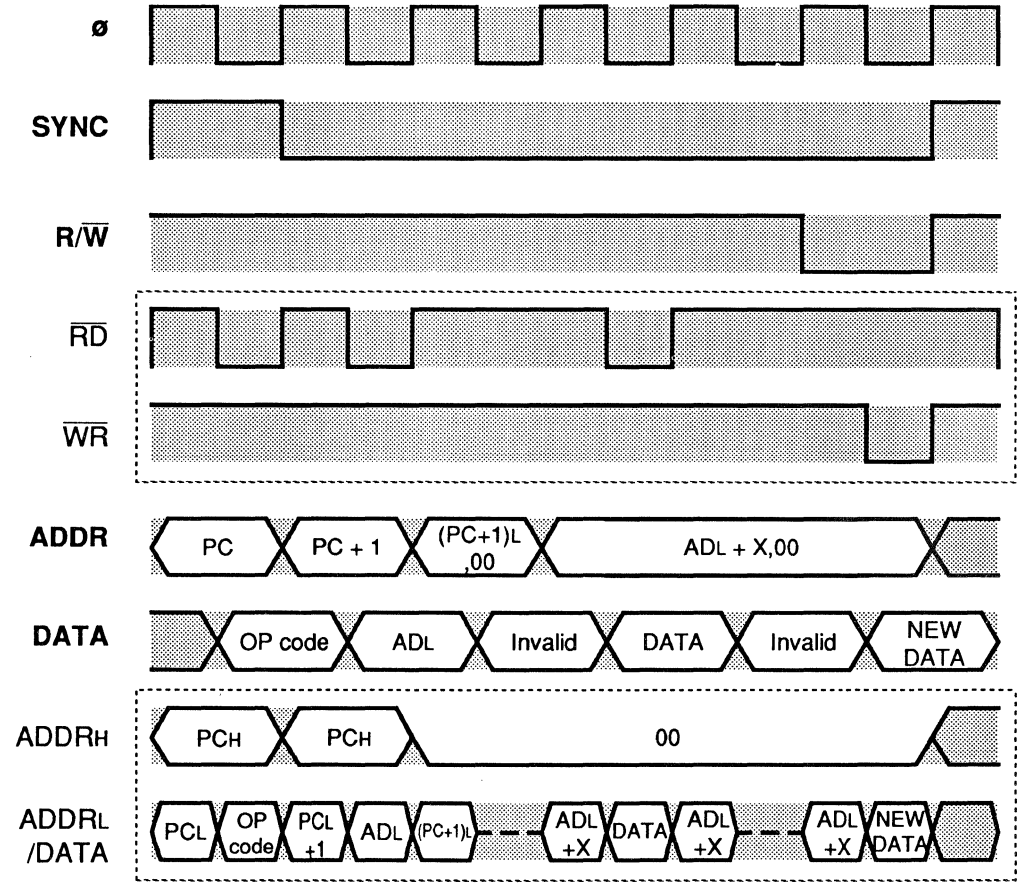
ZERO PAGE X

Instructions : **ASLΔ$zz,X**
DECΔ$zz,X
INCΔ$zz,X
LSRΔ$zz,X
ROLΔ$zz,X
RORΔ$zz,X

Byte length : **2**

Cycle number : **6**

Timing :

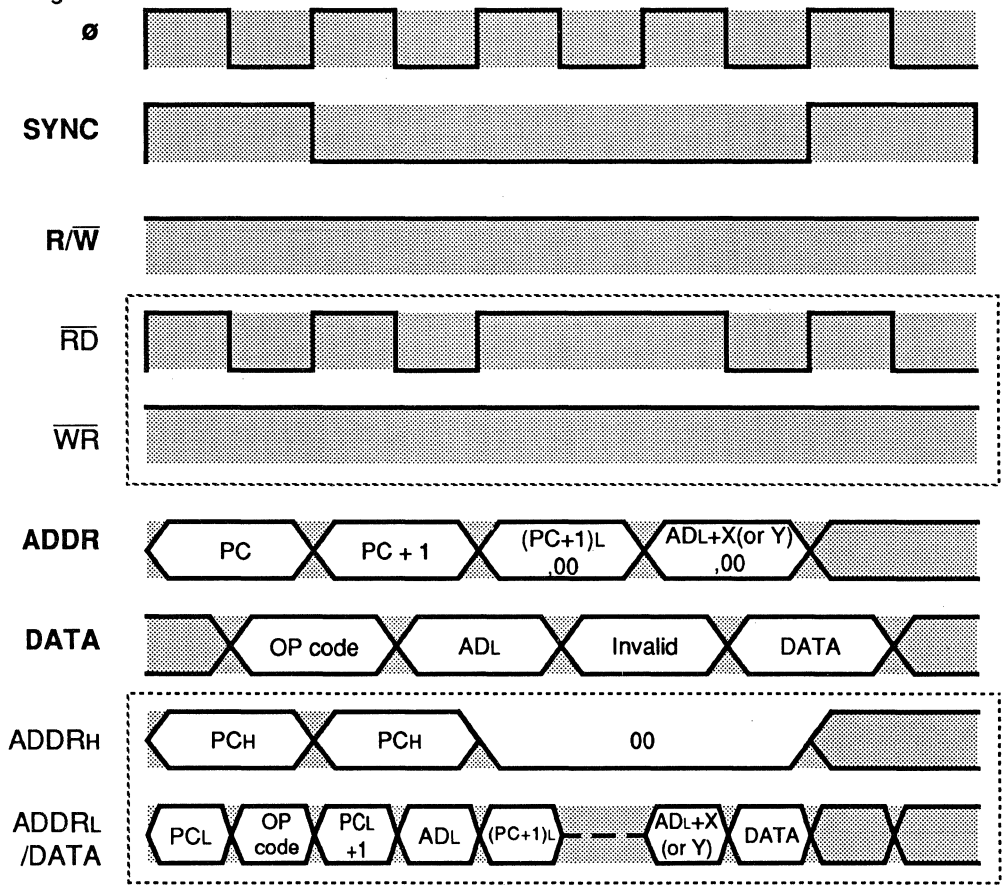


ZERO PAGE X
ZERO PAGE Y

Instructions : **ADCΔ$zz,X (T=0)**
ANDΔ$zz,X (T=0)
CMPΔ$zz,X (T=0)
EORΔ$zz,X (T=0)
LDAΔ$zz,X (T=0)
LDXΔ$zz,Y
LDYΔ$zz,X
ORAΔ$zz,X (T=0)
SBCΔ$zz,X (T=0)

Byte length : **2**
Cycle number : **4**

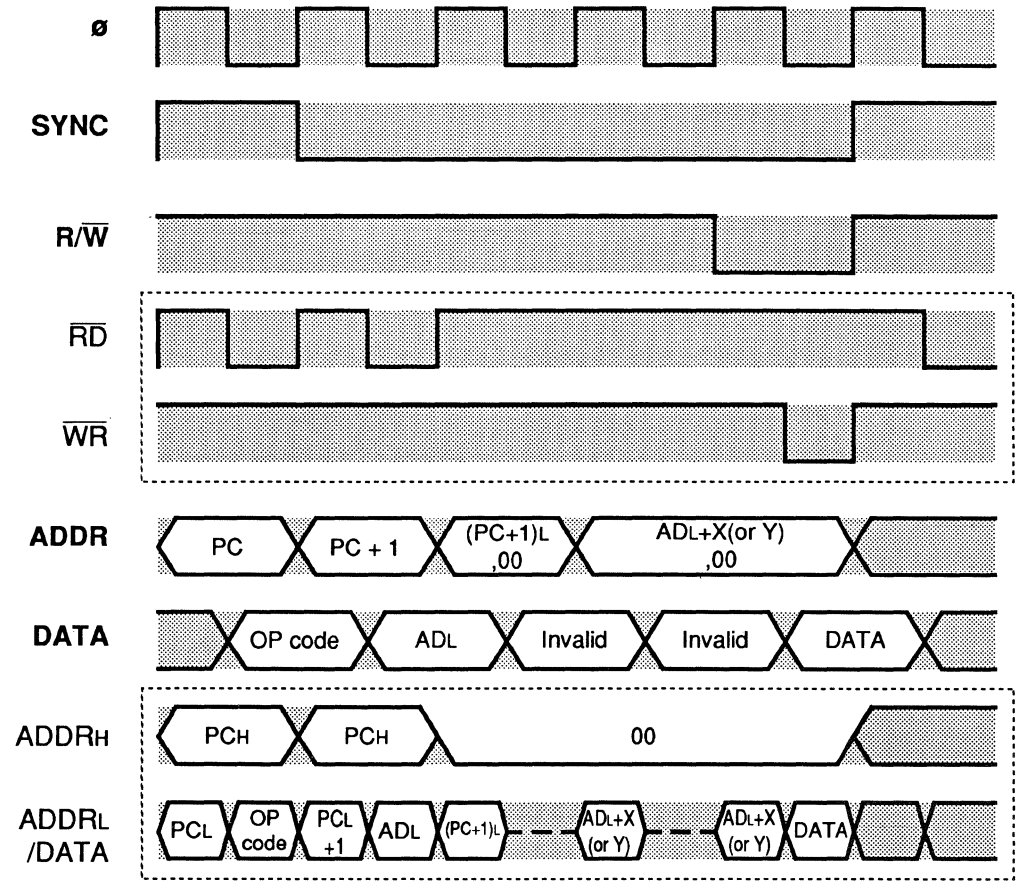
Timing :



ZERO PAGE X
ZERO PAGE Y

Instructions : **STAΔ\$zz,X**
STXΔ\$zz,Y
STYΔ\$zz,X
Byte length : **2**
Cycle number : **5**

Timing :

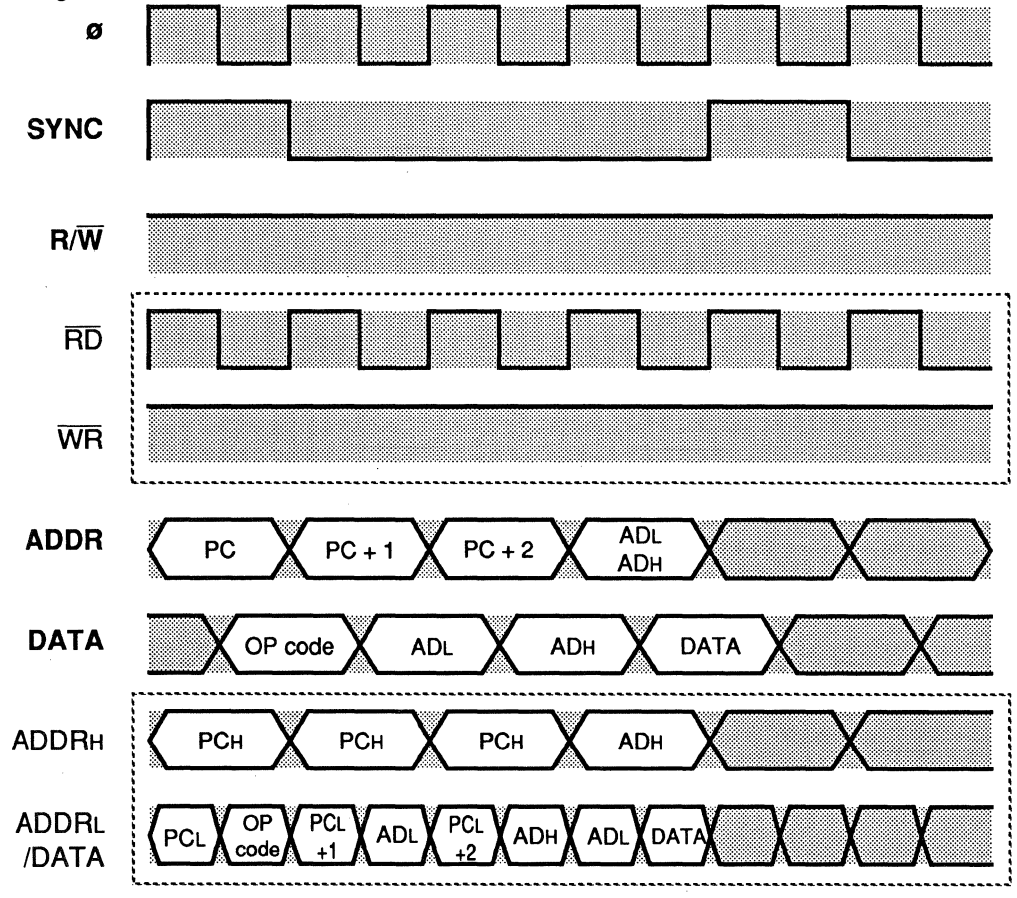


ABSOLUTE

Instructions : **ADCΔ$hhll (T=0)**
ANDΔ$hhll (T=0)
BITΔ$hhll
CMPΔ$hhll (T=0)
CPXΔ$hhll
CPYΔ$hhll
EORΔ$hhll (T=0)
LDAΔ$hhll (T=0)
LDXΔ$hhll
LDYΔ$hhll
ORAΔ$hhll (T=0)
SBCΔ$hhll (T=0)

Byte length : **3**
Cycle number : **4**

Timing :



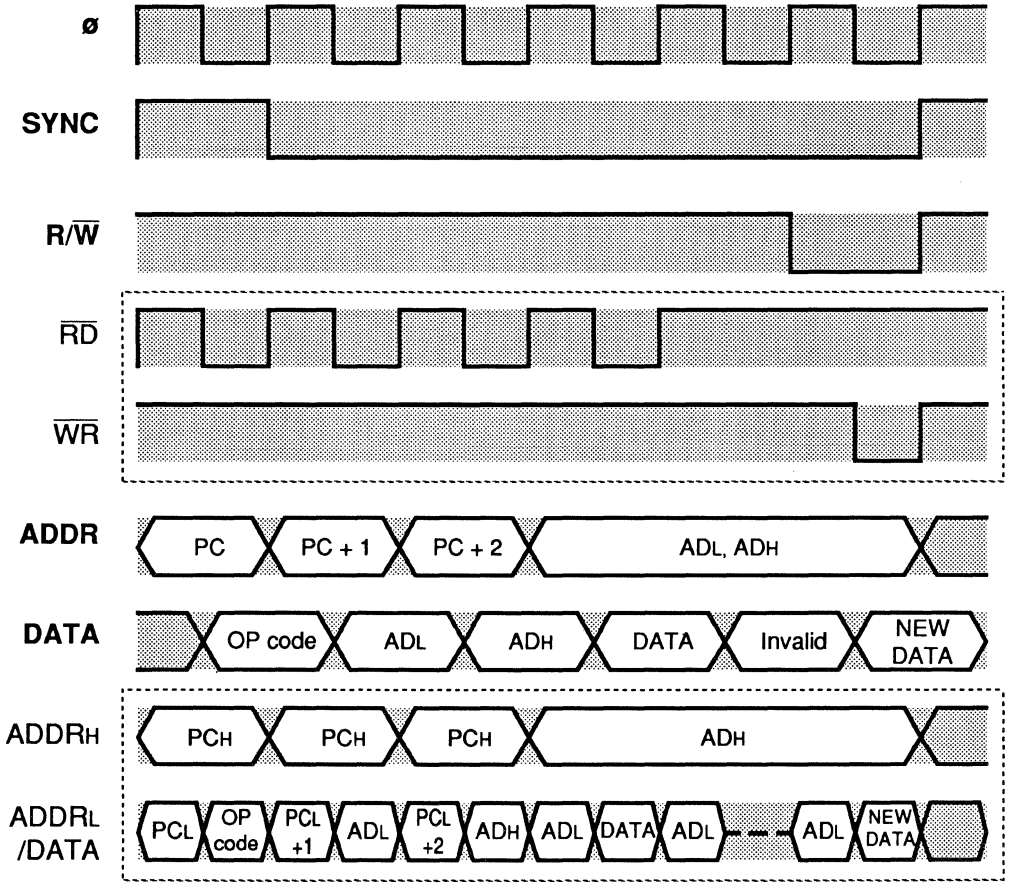
ABSOLUTE

Instructions : **ASLΔ$hhll**
DECΔ$hhll
INCΔ$hhll
LSRΔ$hhll
ROLΔ$hhll
RORΔ$hhll

Byte length : **3**

Cycle number : **6**

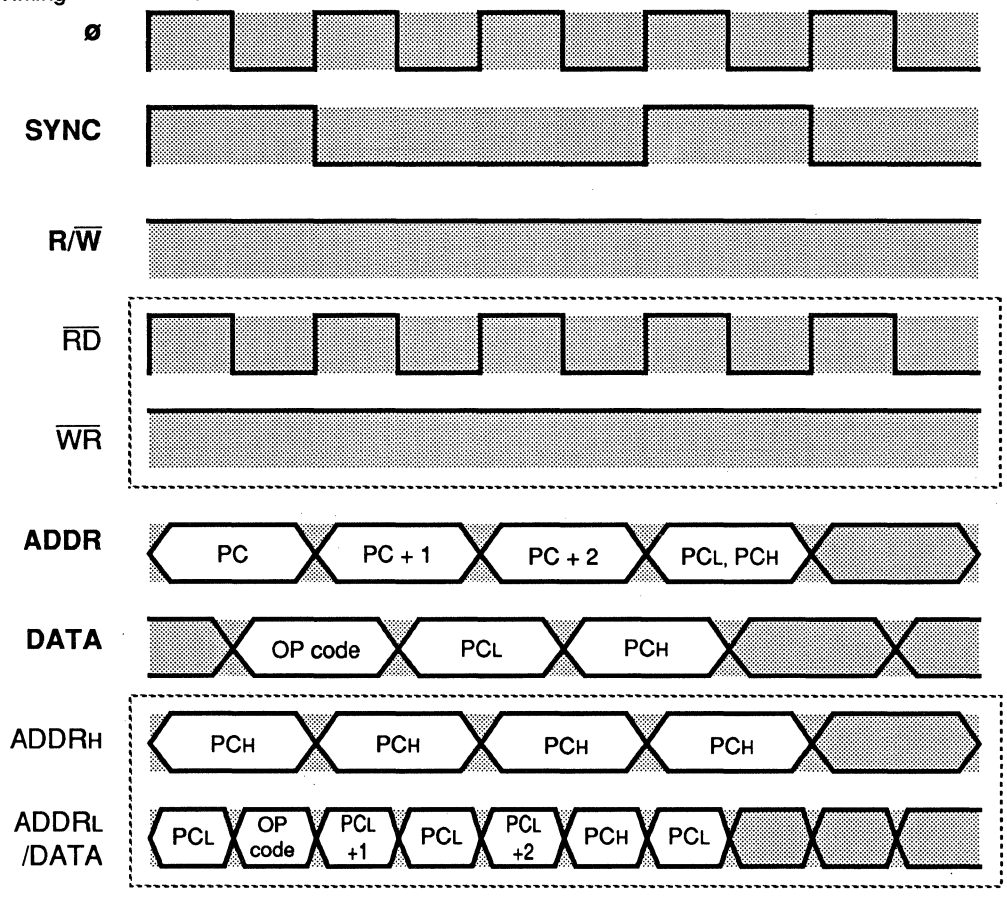
Timing :



ABSOLUTE

Instruction : **JMPΔ$hhll**
Byte length : **3**
Cycle number : **3**

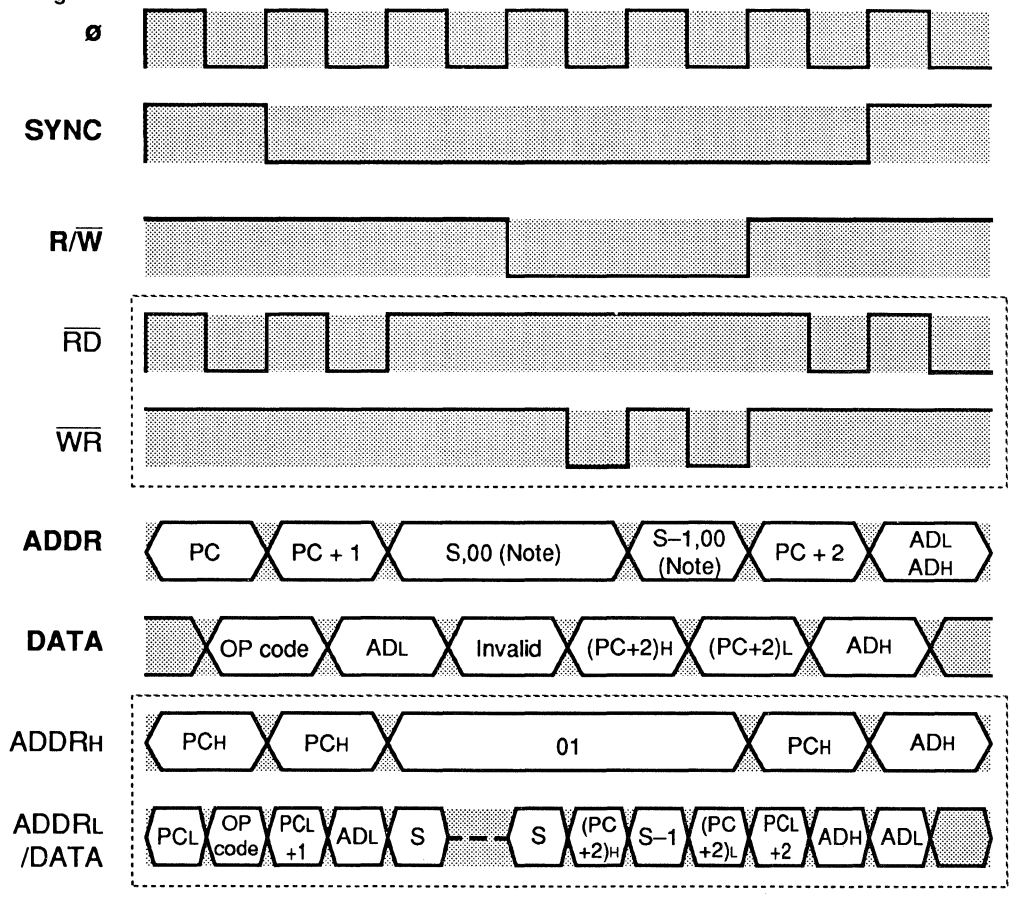
Timing :



ABSOLUTE

Instruction : **JSRΔ$hhll**
Byte length : **3**
Cycle number : **6**

Timing :

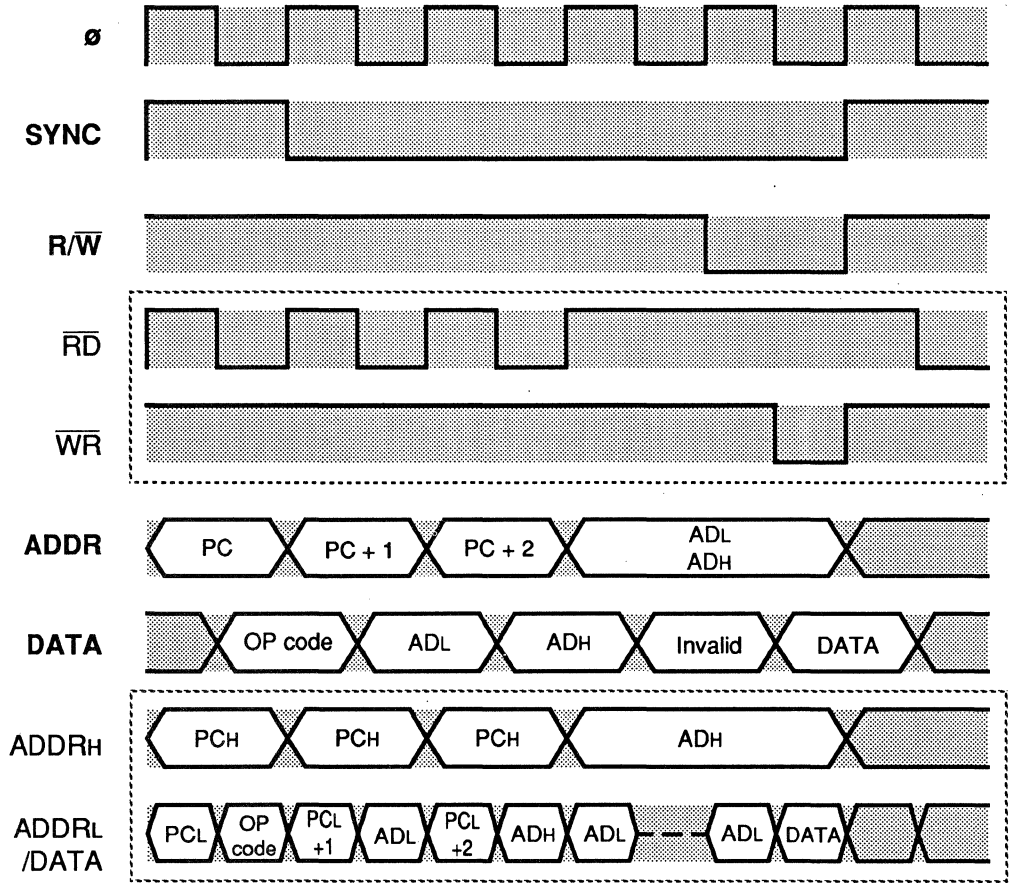


Note : Some kind of types are "01" or content of SPS flag.

ABSOLUTE

Instructions : **STAΔ$hhll**
STXΔ$hhll
STYΔ$hhll
Byte length : **3**
Cycle number : **5**

Timing :



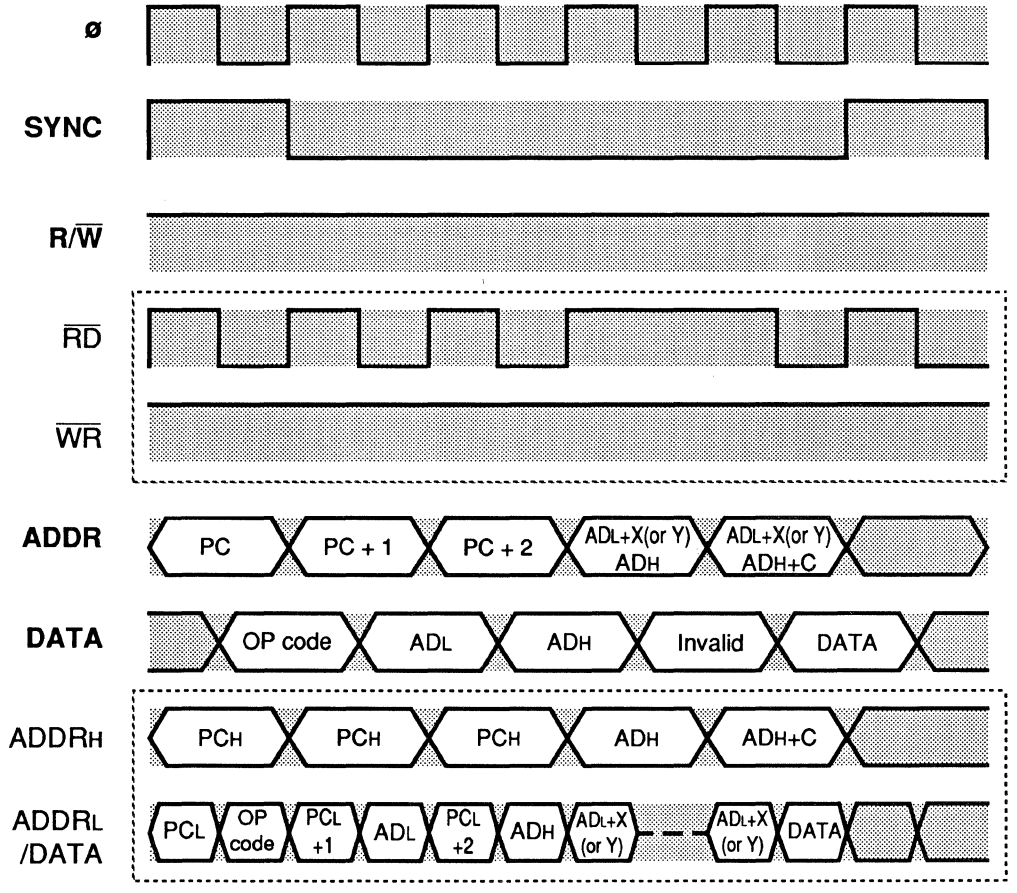
[T=0]

ABSOLUTE X
ABSOLUTE Y

Instructions : **ADCΔ$hhll,X or Y (T=0)**
ANDΔ$hhll,X or Y (T=0)
CMPΔ$hhll,X or Y (T=0)
EORΔ$hhll,X or Y (T=0)
LDAΔ$hhll,X or Y (T=0)
LDXΔ$hhll,Y
LDYΔ$hhll,X
ORAΔ$hhll,X or Y (T=0)
SBCΔ$hhll,X or Y (T=0)

Byte length : **3**
Cycle number : **5**

Timing :



C:Carry of ADL+X or Y

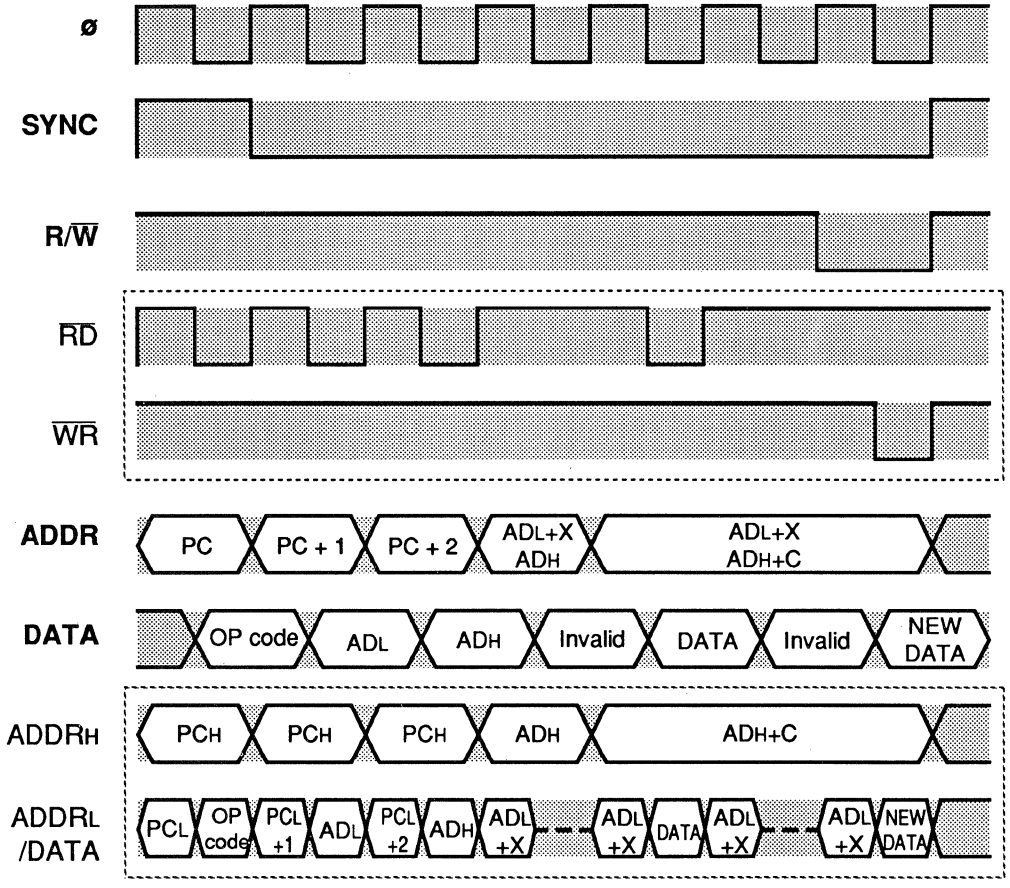
ABSOLUTE X

Instructions : **ASLΔ$hhll,X**
DECΔ$hhll,X
INCΔ$hhll,X
LSRΔ$hhll,X
ROLΔ$hhll,X
RORΔ$hhll,X

Byte length : **3**

Cycle number : **7**

Timing :

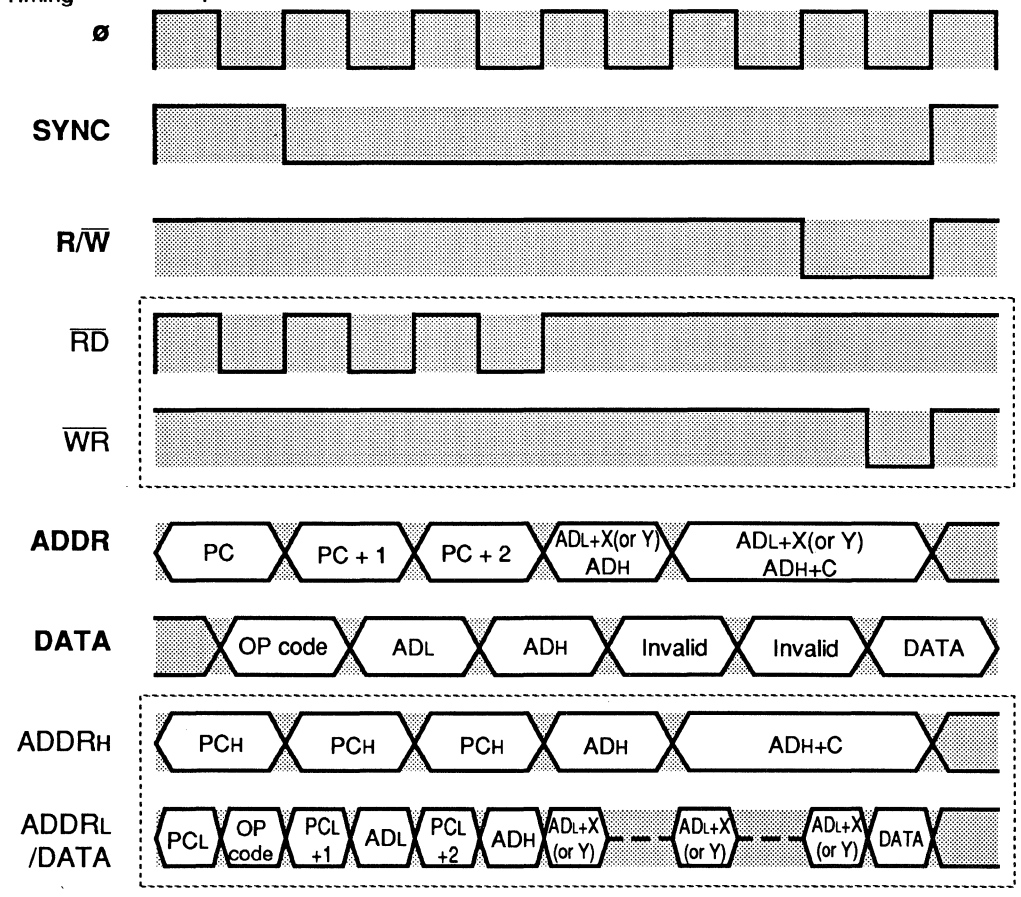


C:Carry of ADL+X

ABSOLUTE X
ABSOLUTE Y

Instruction : **STAΔ$hhll,X or Y**
Byte length : **3**
Cycle number : **6**

Timing :

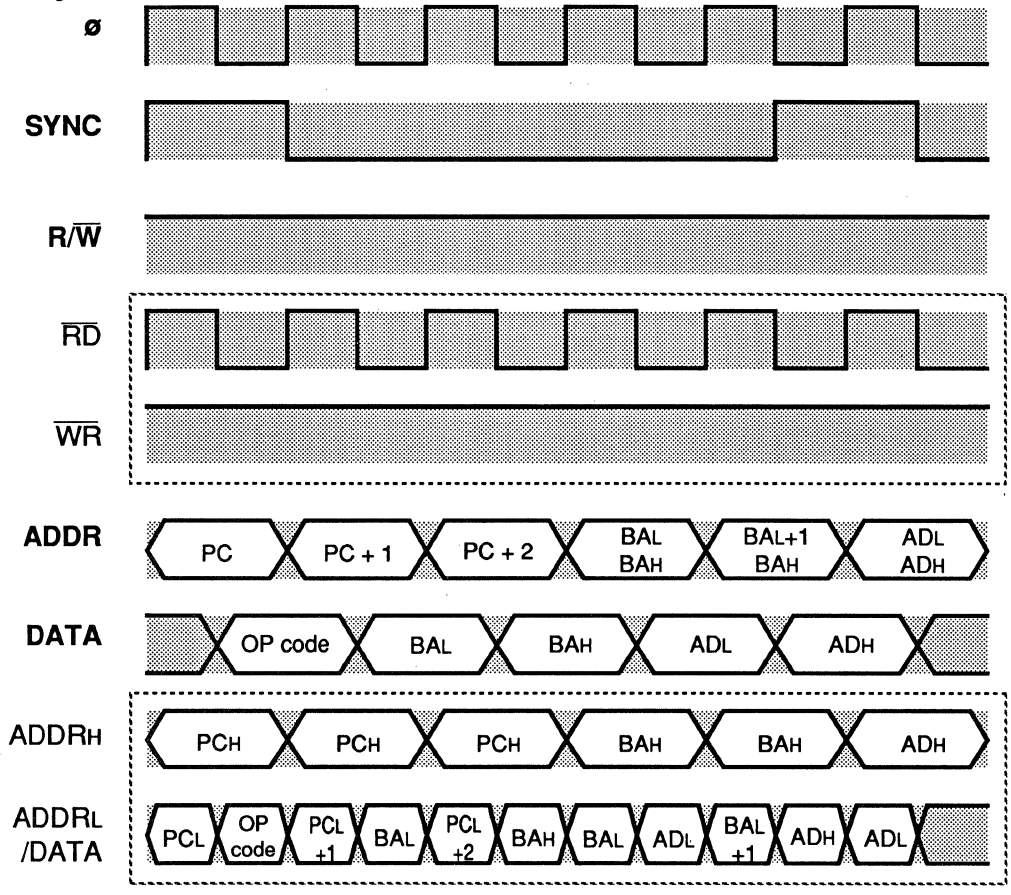


C:Carry of ADL+X or Y

INDIRECT

Instructions : **JMPΔ($hhll)**
Byte length : **3**
Cycle number : **5**

Timing :

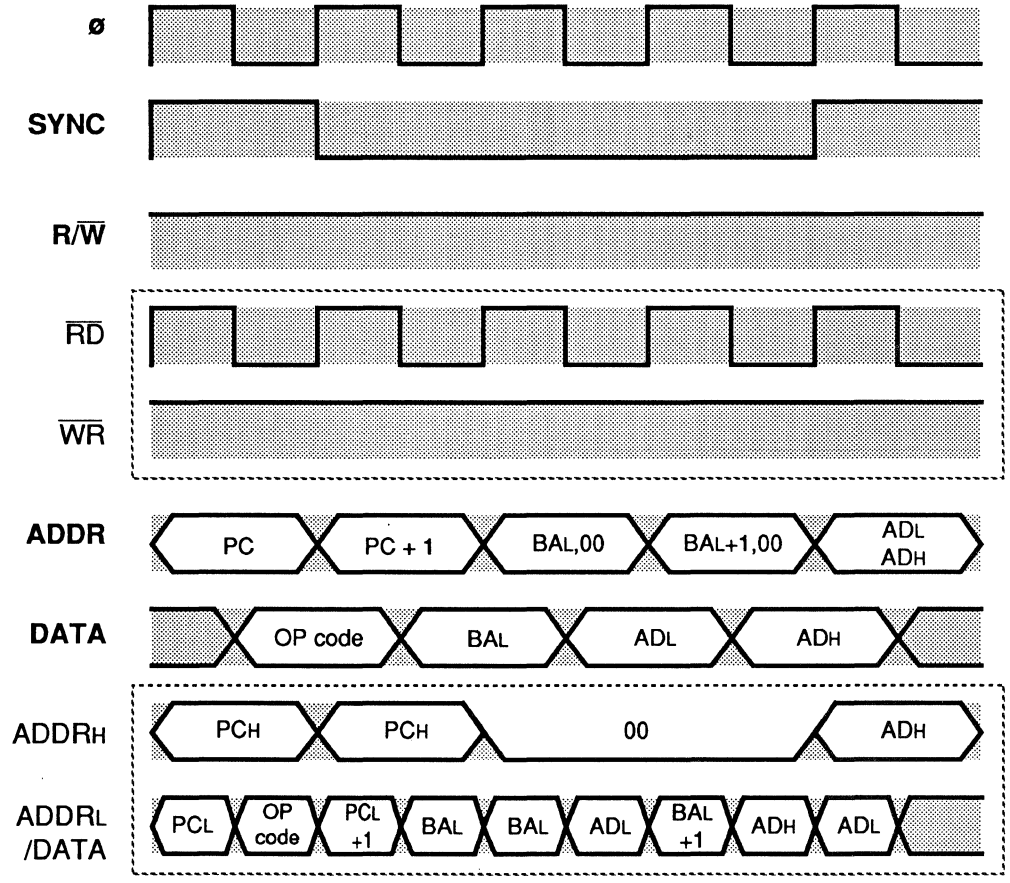


BA: Basic address

ZERO PAGE INDIRECT

Instruction : **JMPΔ($zz)**
Byte length : **2**
Cycle number : **4**

Timing :

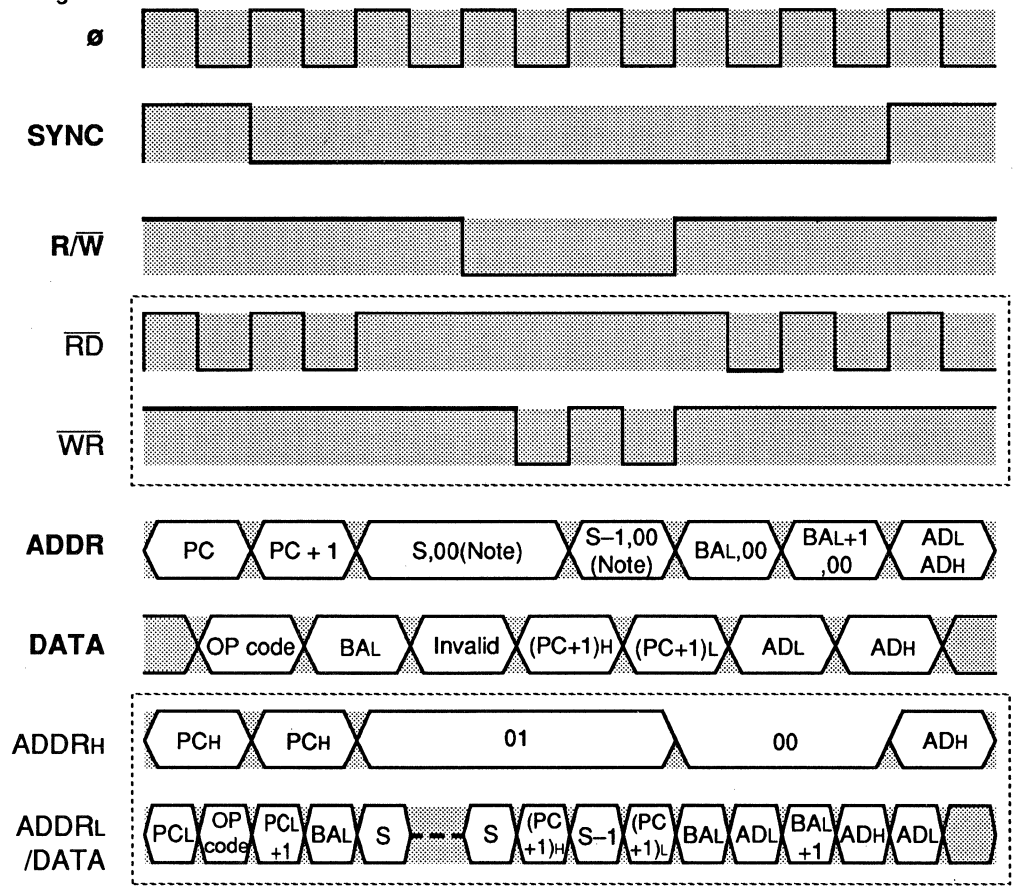


BA: Basic address

ZERO PAGE INDIRECT

Instruction : **JSRΔ($zz)**
Byte length : **2**
Cycle number : **7**

Timing :



BA : Basic address
Note : Some kind of types are "01" or content of SPS flag.

[T=0]

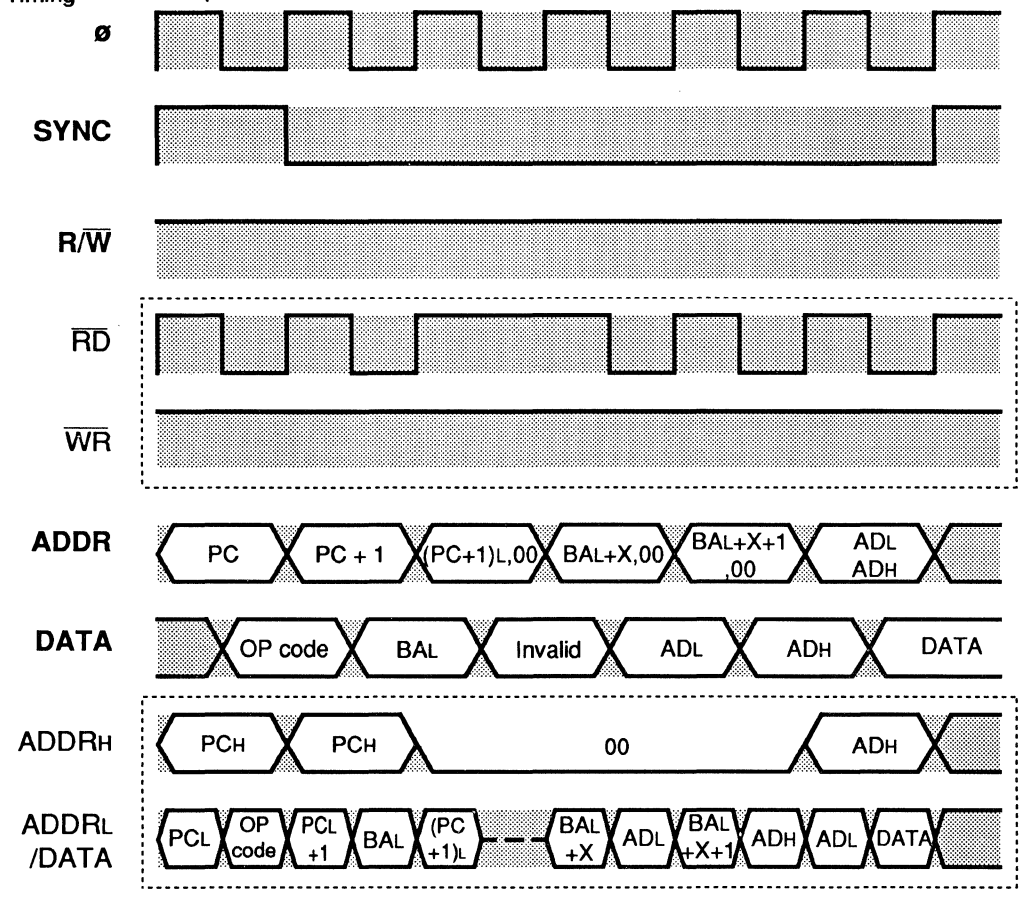
INDIRECT X

Instructions : **ADCΔ($zz,X) (T=0)**
ANDΔ($zz,X) (T=0)
CMPΔ($zz,X) (T=0)
EORΔ($zz,X) (T=0)
LDAΔ($zz,X) (T=0)
ORAΔ($zz,X) (T=0)
SBCΔ($zz,X) (T=0)

Byte length : **2**

Cycle number : **6**

Timing :

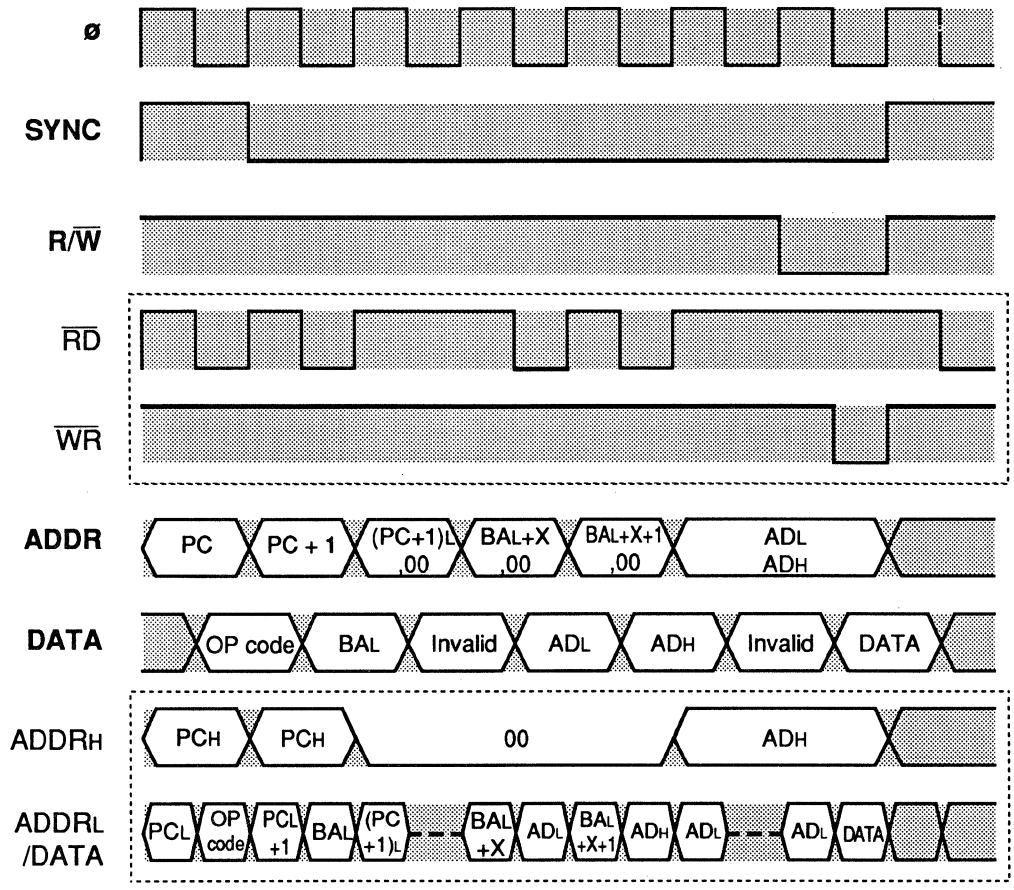


BA : Basic address

INDIRECT X

Instruction : **STAΔ($zz,X)**
Byte length : **2**
Cycle number : **7**

Timing :



BA : Basic address

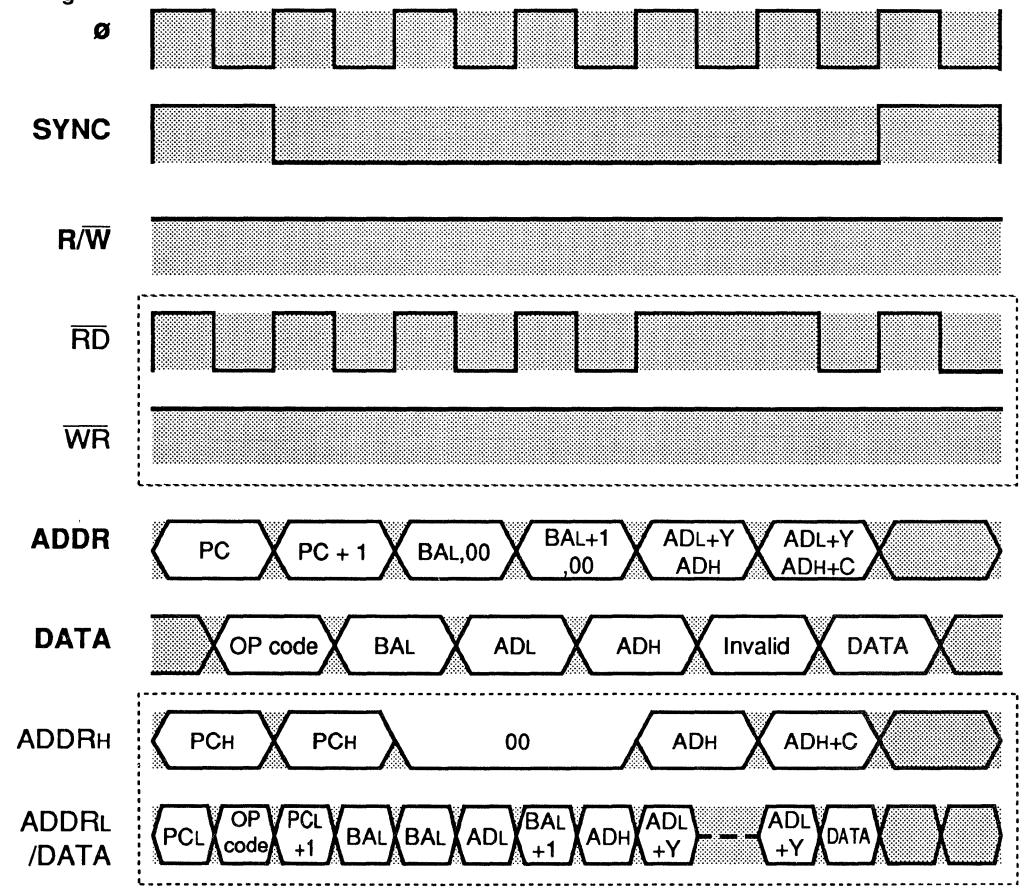
[T=0]

INDIRECT Y

Instructions : **ADCΔ($zz),Y (T=0)**
ANDΔ($zz),Y (T=0)
CMPΔ($zz),Y (T=0)
EORΔ($zz),Y (T=0)
LDAΔ($zz),Y (T=0)
ORAΔ($zz),Y (T=0)
SBCΔ($zz),Y (T=0)

Byte length : **2**
Cycle number : **6**

Timing :

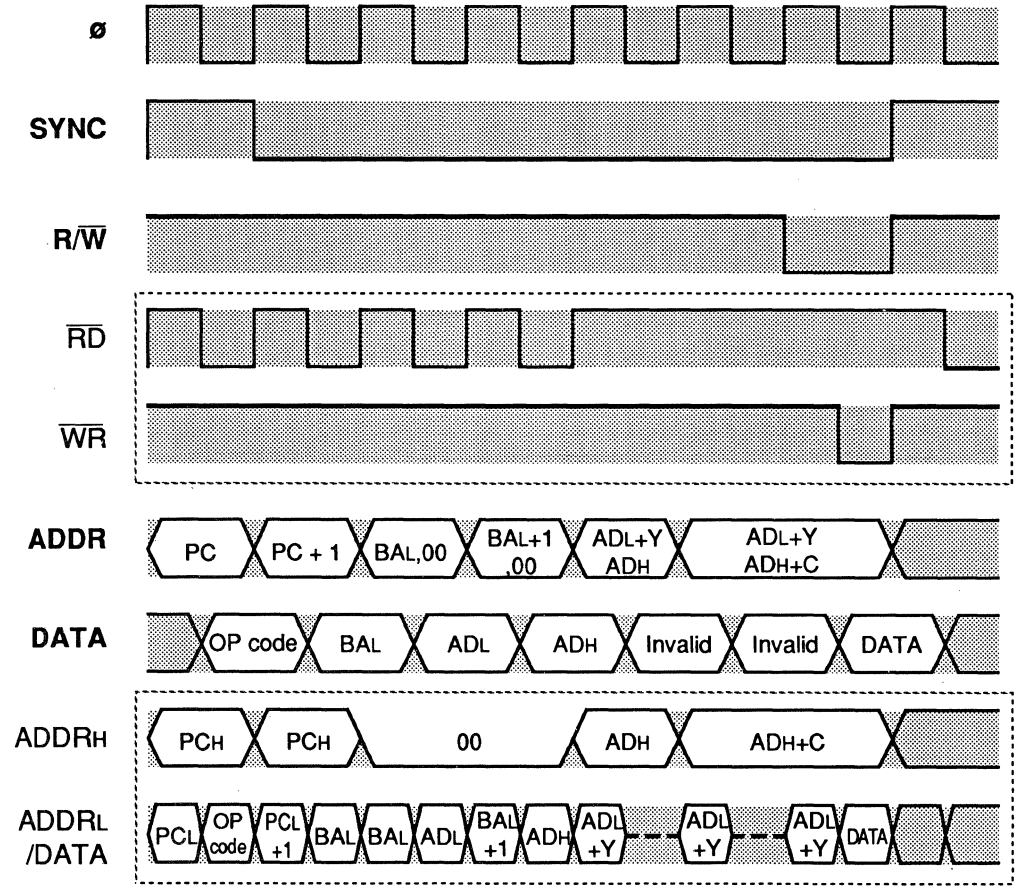


BA: Basic address
C : Carry of ADL+Y

INDIRECT Y

Instruction : **STAΔ($zz),Y**
Byte length : **2**
Cycle number : **7**

Timing :



BA: Basic address
C : Carry of ADL+Y

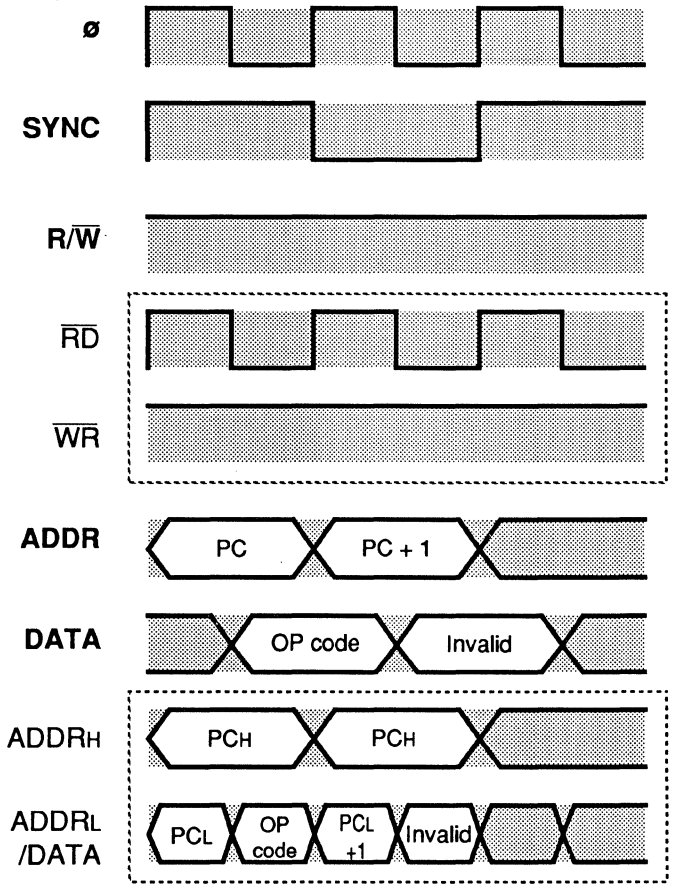
RELATIVE

Instructions : **BCCΔ$hhll**
BCSΔ$hhll
BEQΔ$hhll
BMIΔ$hhll
BNEΔ$hhll
BPLΔ$hhll
BVCΔ$hhll
BVSΔ$hhll

Byte length : **2**

(1)With no branch
Cycle number : **2**

Timing :



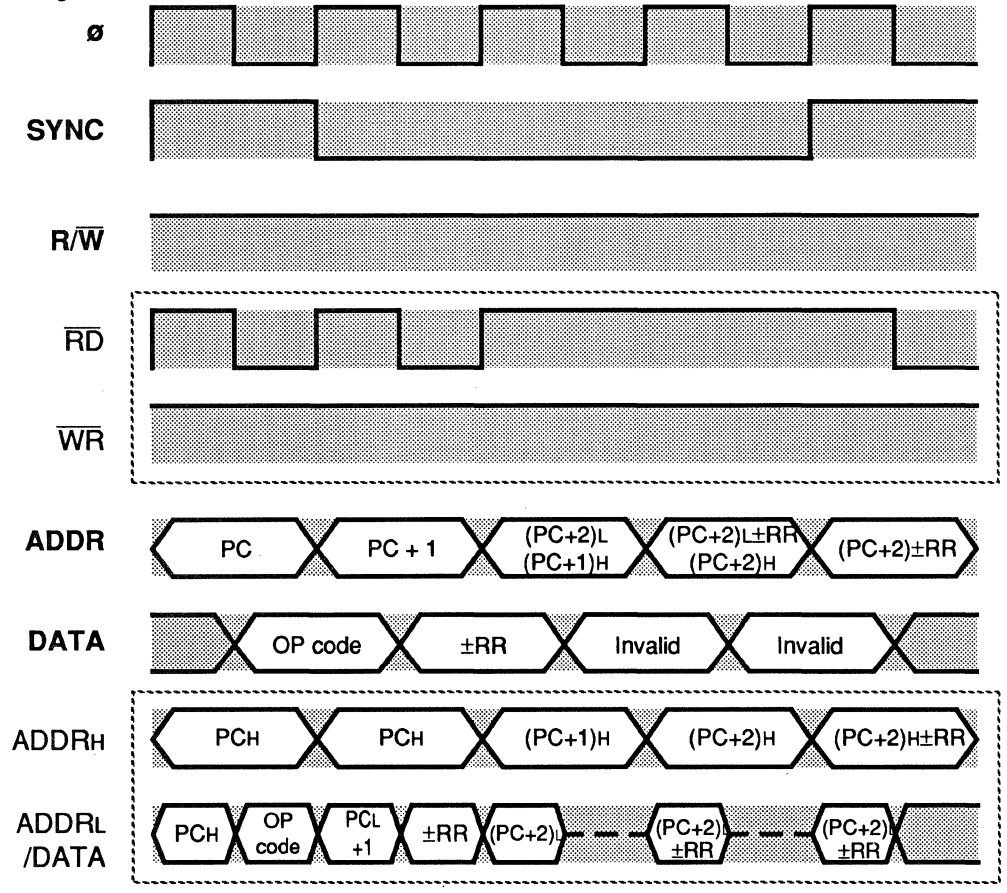
RELATIVE

Instructions : **BCCΔ$hhll**
BCSΔ$hhll
BEQΔ$hhll
BMIΔ$hhll
BNEΔ$hhll
BPLΔ$hhll
BVCΔ$hhll
BVSΔ$hhll

Byte length : **2**

(2)With branch
Cycle number : **4**

Timing :

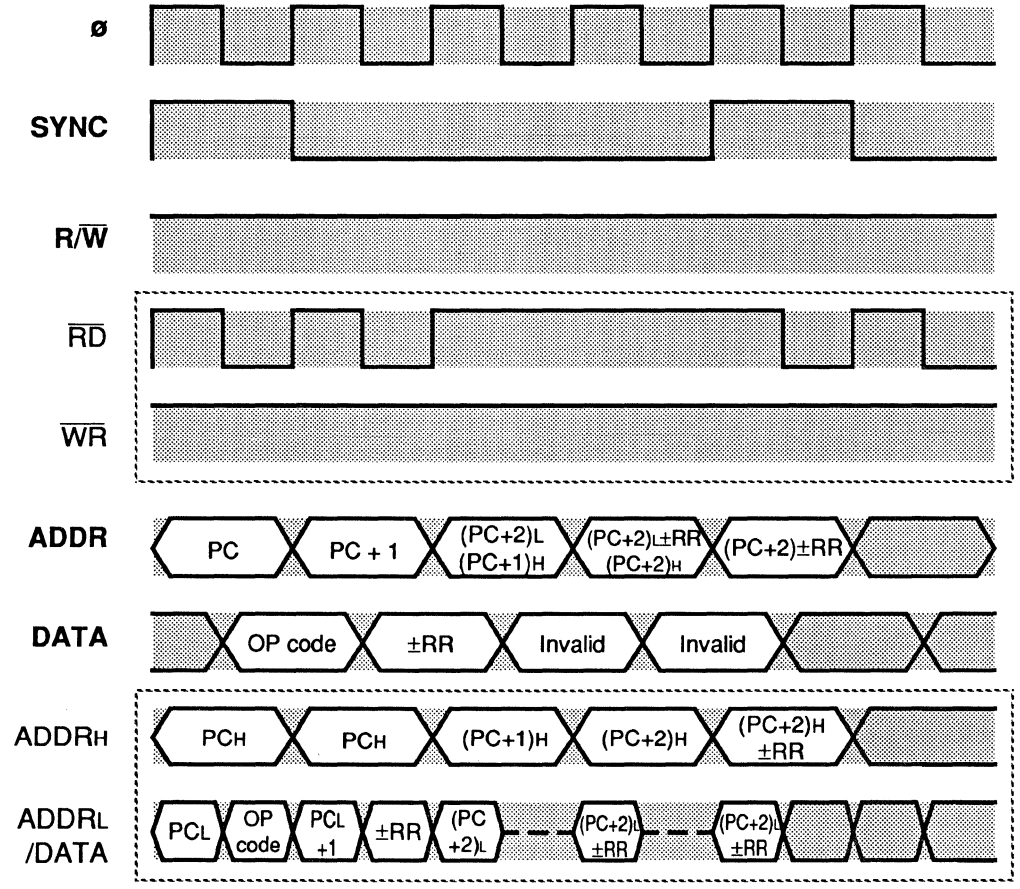


RR : Offset value

RELATIVE

Instruction : **BRAΔ$hhll**
Byte length : **2**
Cycle number : **4**

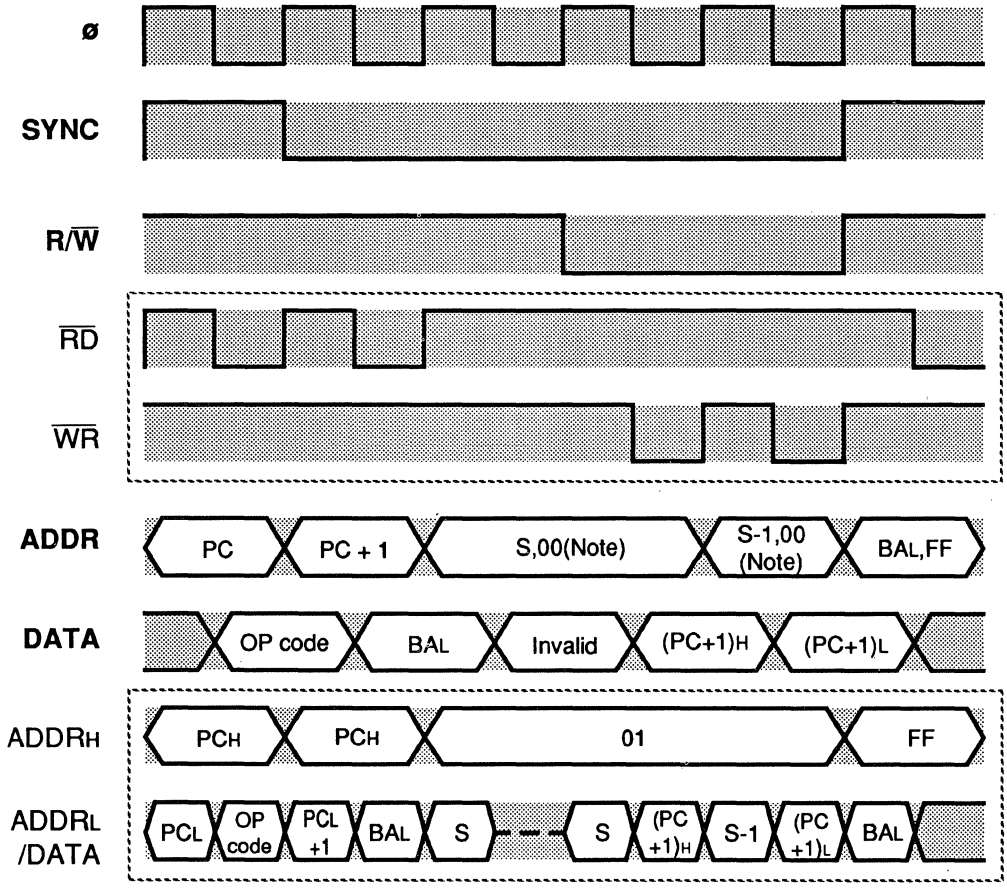
Timing :



RR : Offset value

Instruction : **JSRΔ\$hhll**
Byte length : **2**
Cycle number : **5**

Timing :



BA : Basic address
Note : Some kind of types are "01" or content of SPS flag.

[T=1]

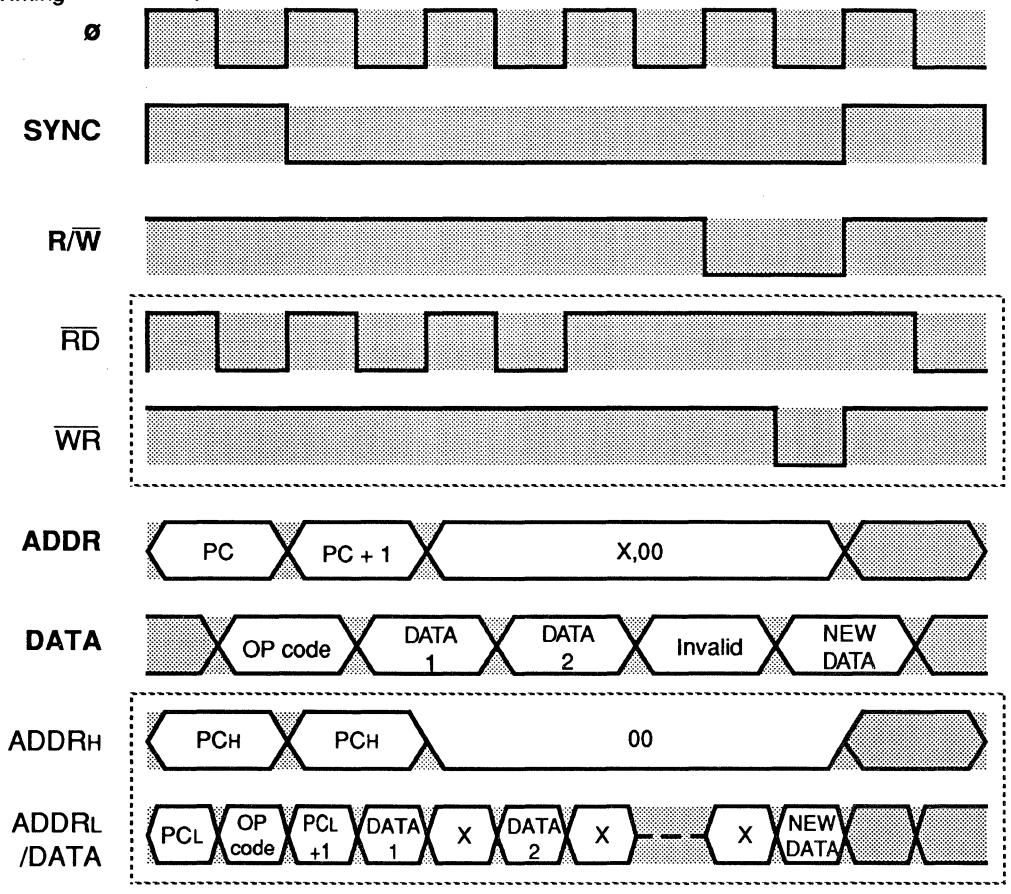
IMMEDIATE

Instructions : **ADCΔ#$nn (T=1)**
ANDΔ#$nn (T=1)
EORΔ#$nn (T=1)
ORAΔ#$nn (T=1)
SBCΔ#$nn (T=1)

Byte length : **2**

Cycle number : **5**

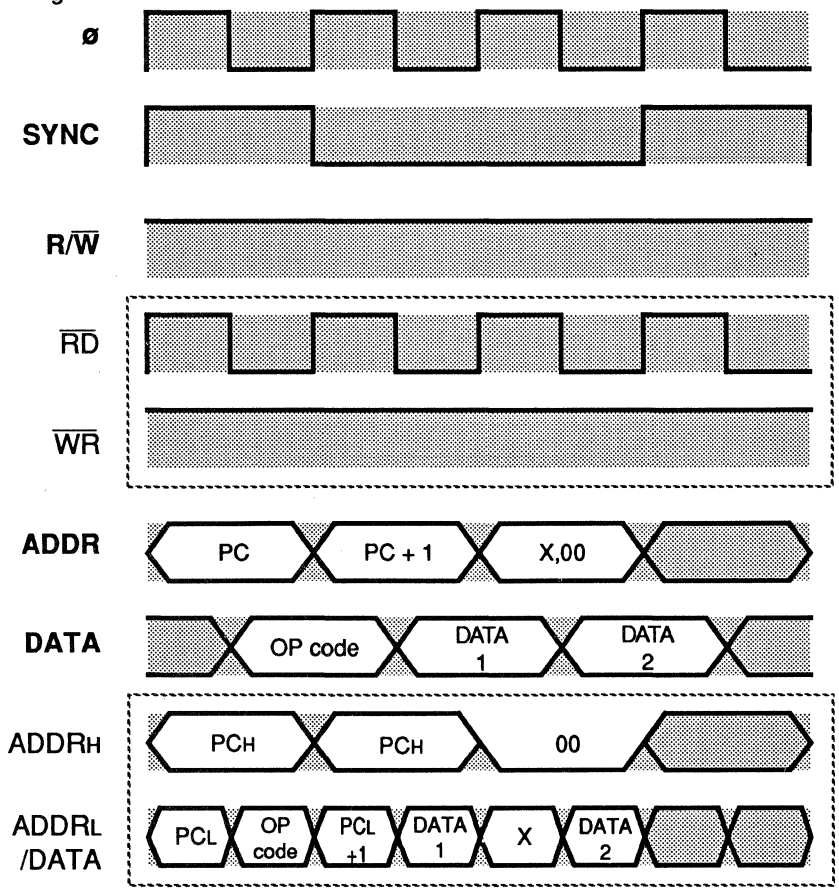
Timing :



IMMEDIATE

Instruction : **CMPΔ#$nn (T=1)**
Byte length : **2**
Cycle number : **3**

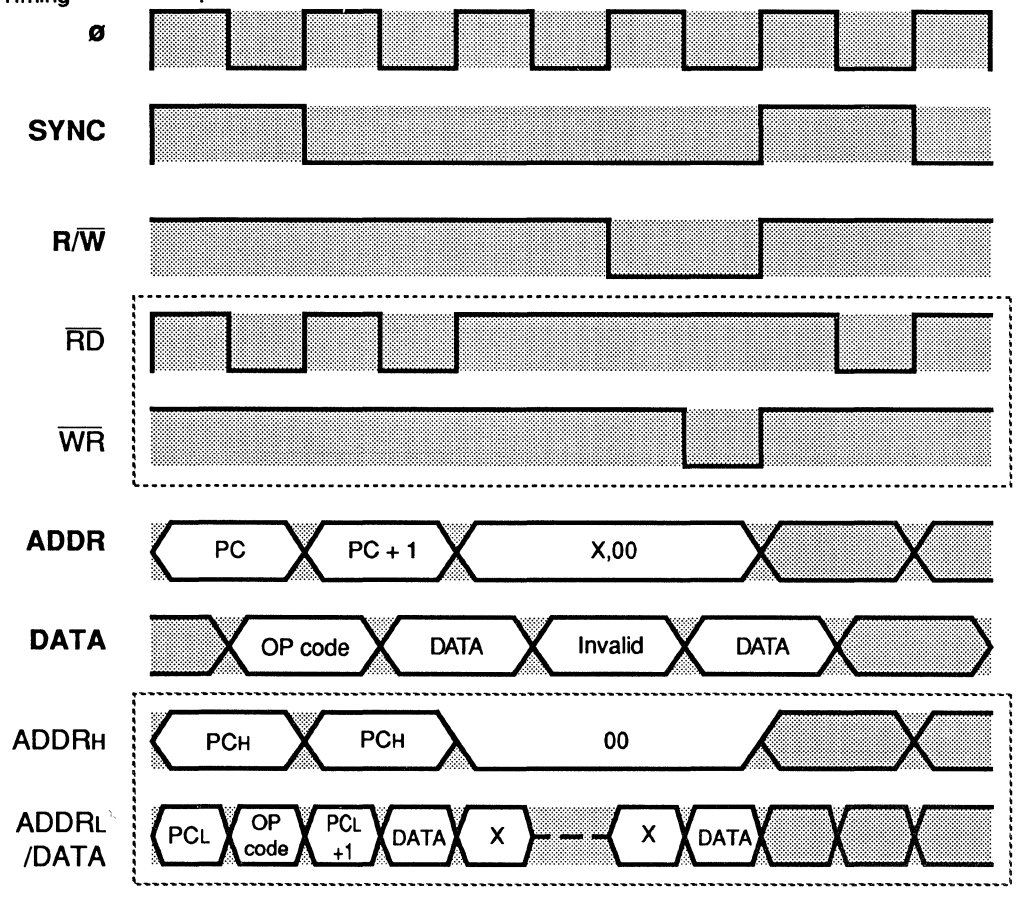
Timing :



IMMEDIATE

Instruction : **LDAΔ#$nn (T=1)**
Byte length : **2**
Cycle number : **4**

Timing :



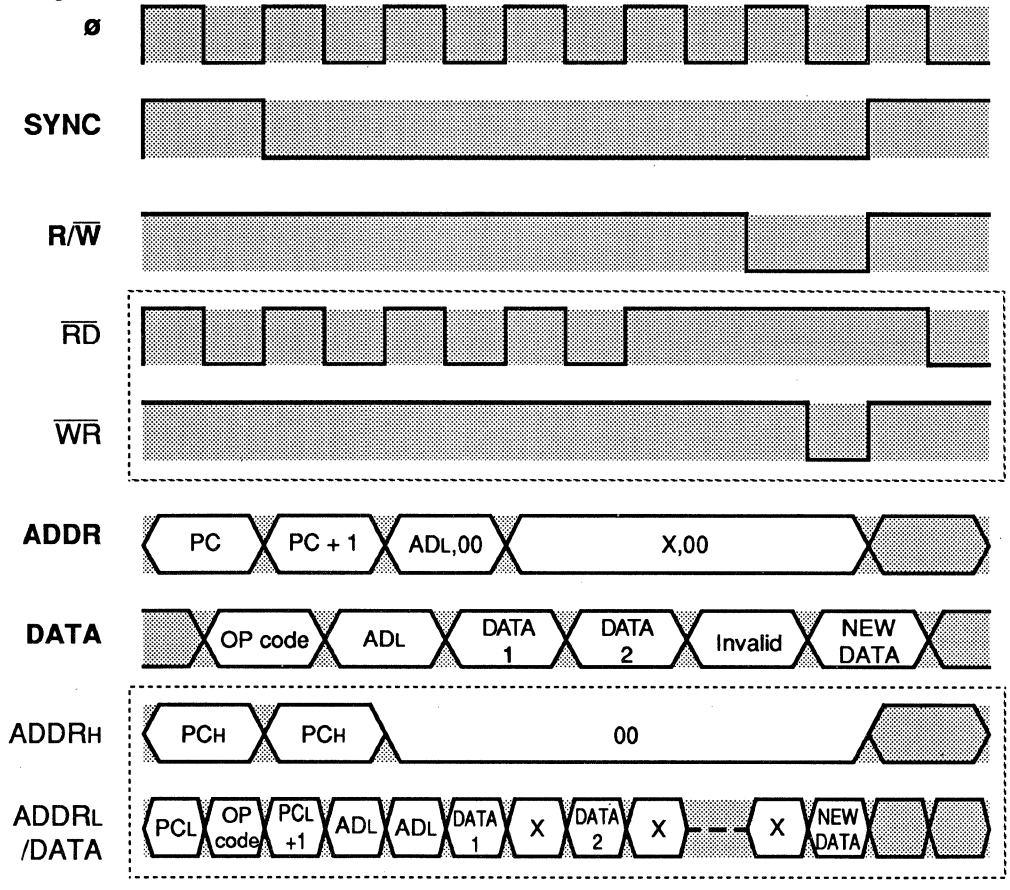
ZERO PAGE

Instructions : **ADCΔ$zz (T=1)**
ANDΔ$zz (T=1)
EORΔ$zz (T=1)
ORAΔ$zz (T=1)
SBCΔ$zz (T=1)

Byte length : **2**

Cycle number : **6**

Timing :

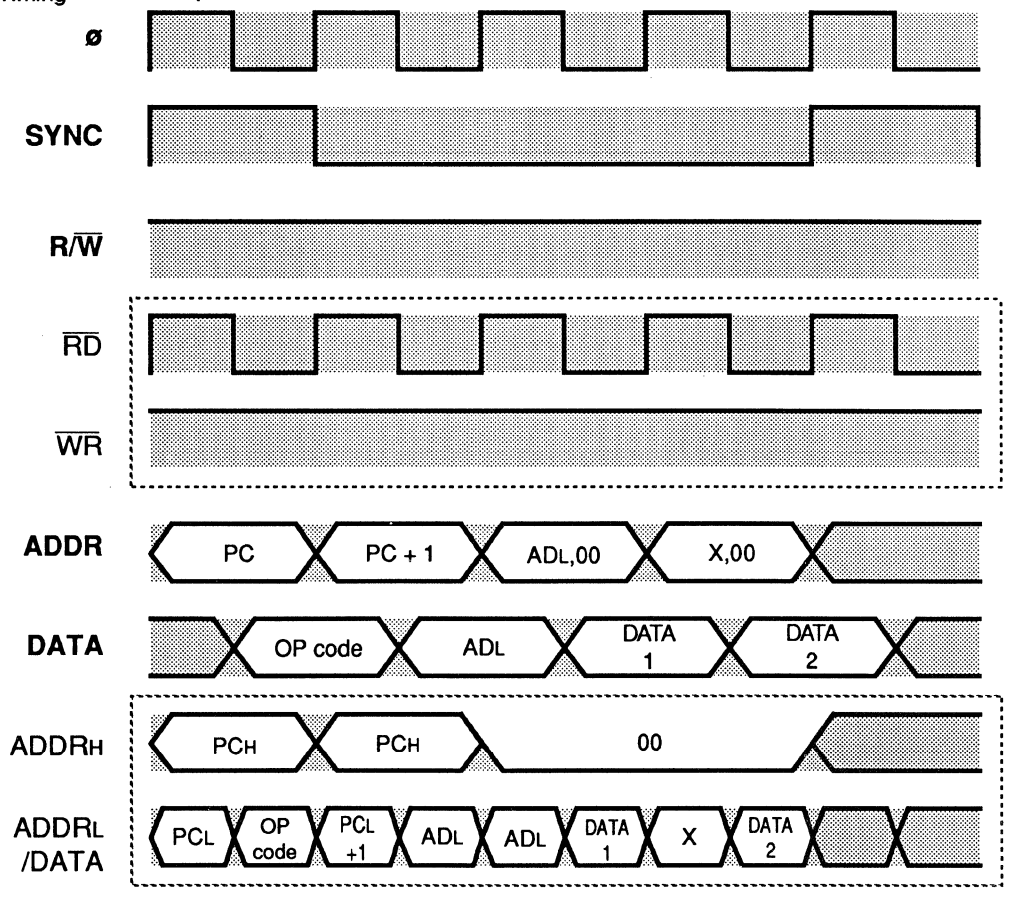


[T=1]

ZERO PAGE

Instruction : **CMPΔ$zz (T=1)**
Byte length : **2**
Cycle number : **4**

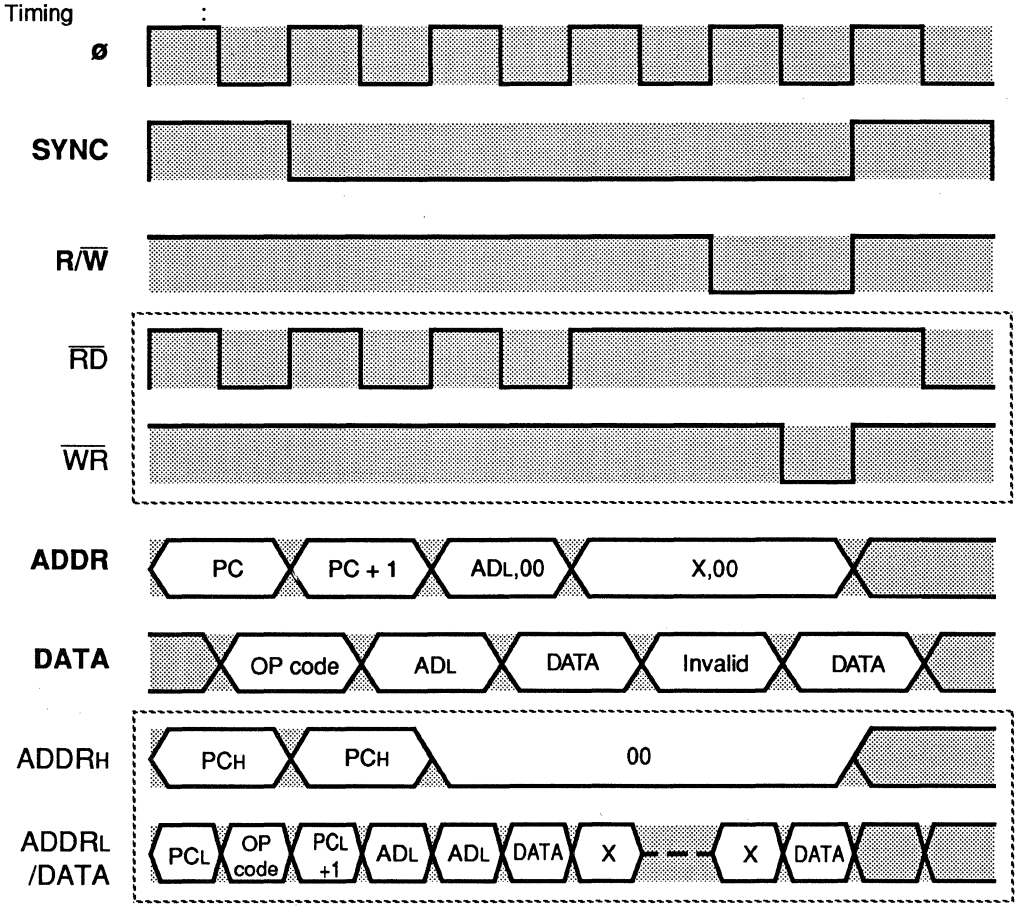
Timing :



ZERO PAGE

Instruction : **LDAΔ$zz (T=1)**
Byte length : **2**
Cycle number : **5**

Timing :



[T=1]

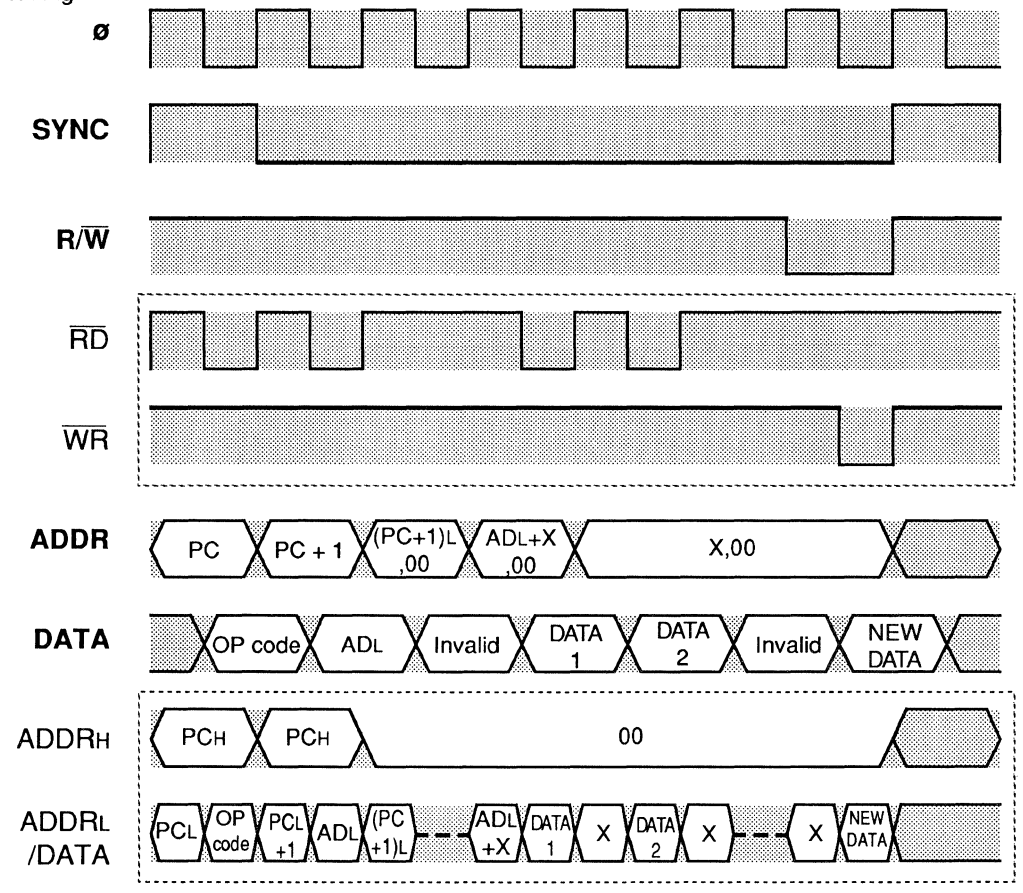
ZERO PAGE X

Instructions : **ADCΔ$zz,X (T=1)**
ANDΔ$zz,X (T=1)
EORΔ$zz,X (T=1)
ORAΔ$zz,X (T=1)
SBCΔ$zz,X (T=1)

Byte length : **2**

Cycle number : **7**

Timing :

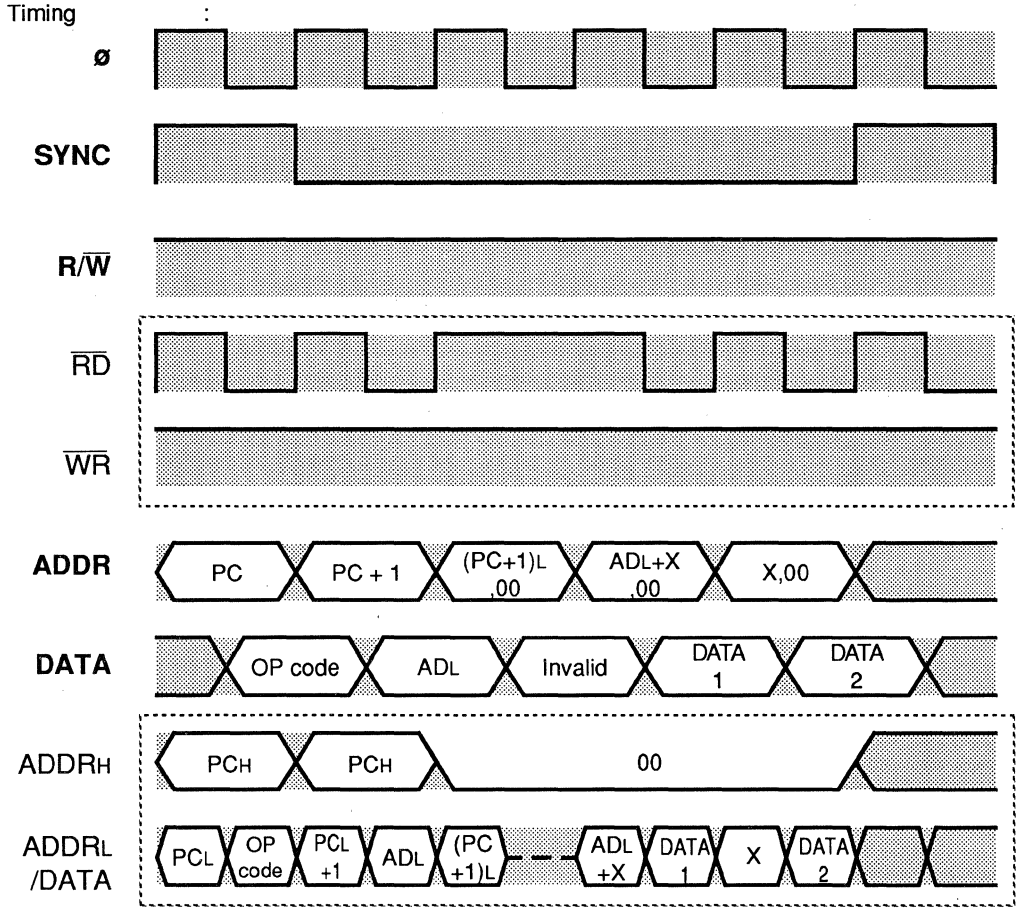


[T=1]

ZERO PAGE X

Instruction : **CMPΔ$zz,X (T=1)**
Byte length : **2**
Cycle number : **5**

Timing :

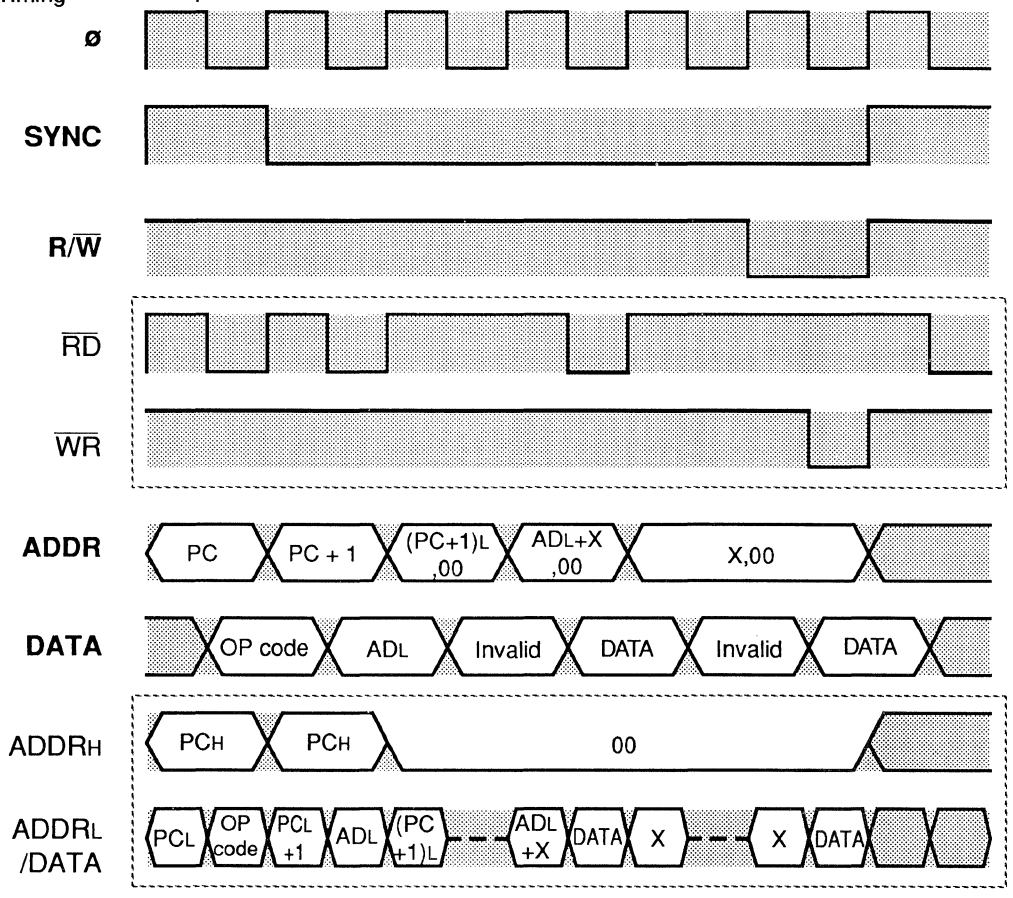


[T=1]

ZERO PAGE X

Instruction : **LDAΔ$zz,X (T=1)**
Byte length : **2**
Cycle number : **6**

Timing :



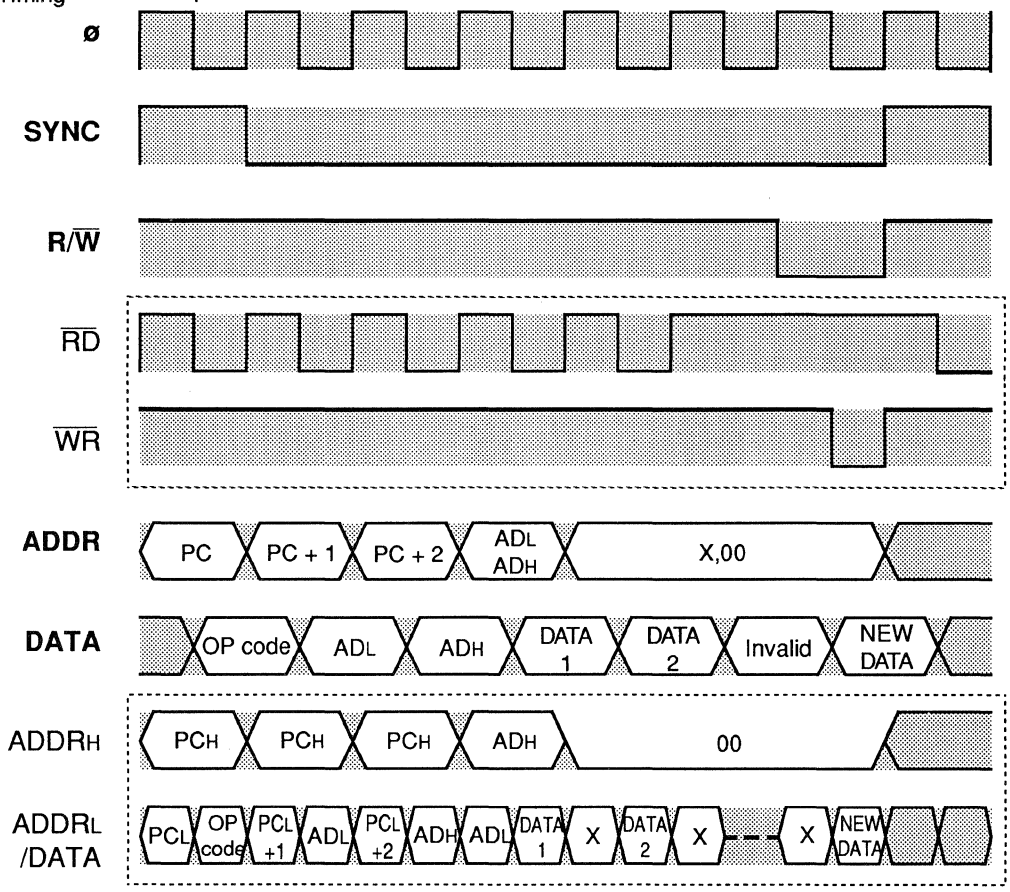
ABSOLUTE

Instruction : **ADCΔ$hhll (T=1)**
ANDΔ$hhll (T=1)
EORΔ$hhll (T=1)
ORAΔ$hhll (T=1)
SBCΔ$hhll (T=1)

Byte length : **3**

Cycle number : **7**

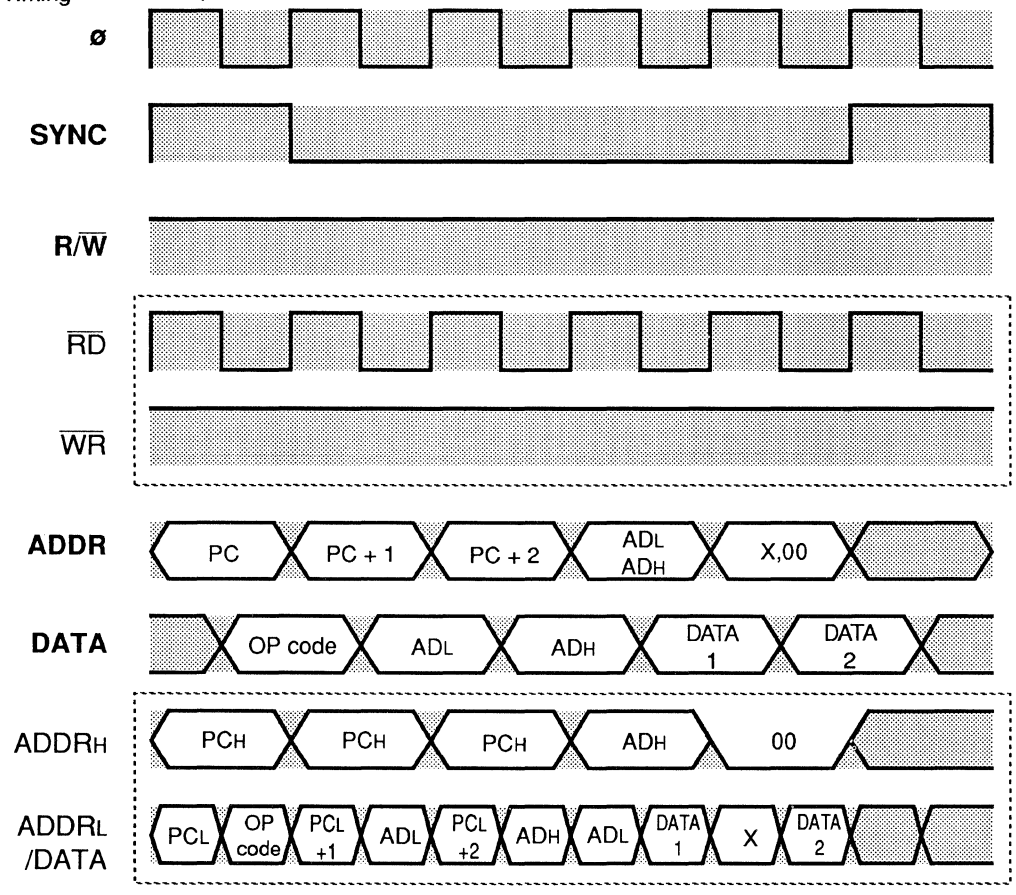
Timing :



ABSOLUTE

Instruction : **CMPΔ$hhll (T=1)**
Byte length : **3**
Cycle number : **5**

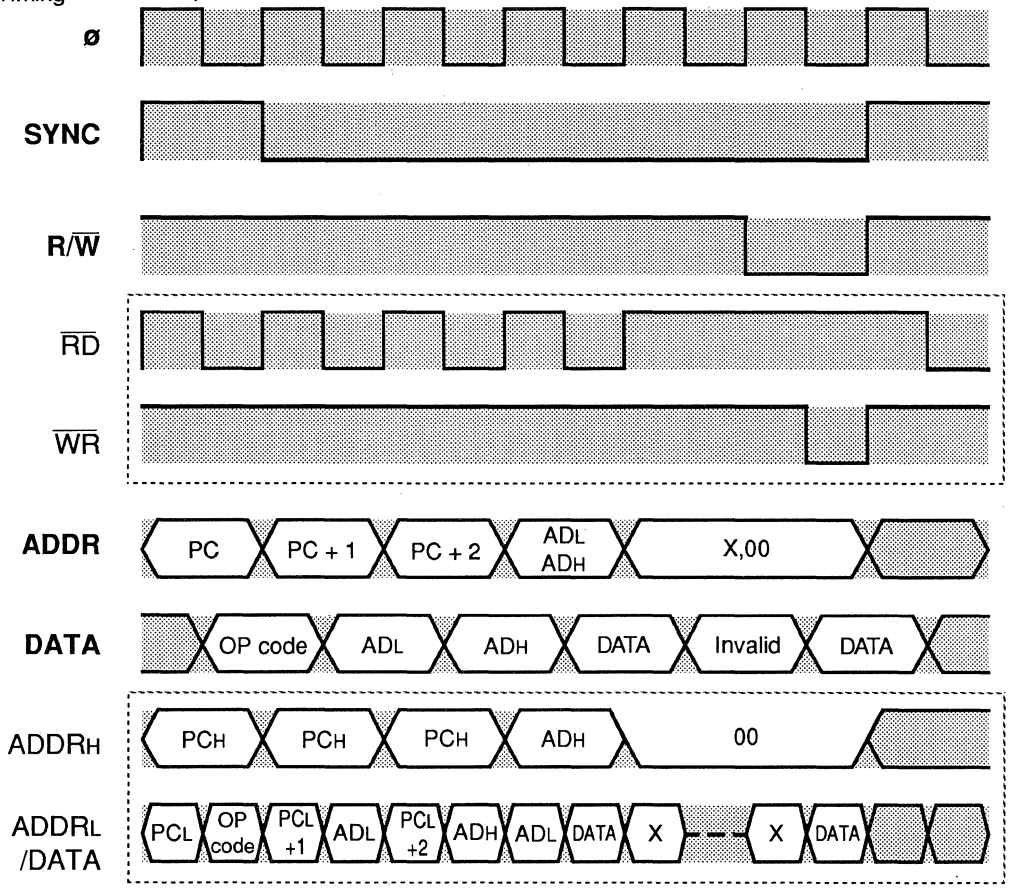
Timing :



ABSOLUTE

Instruction : **LDAΔ$hhll (T=1)**
Byte length : **3**
Cycle number : **6**

Timing :



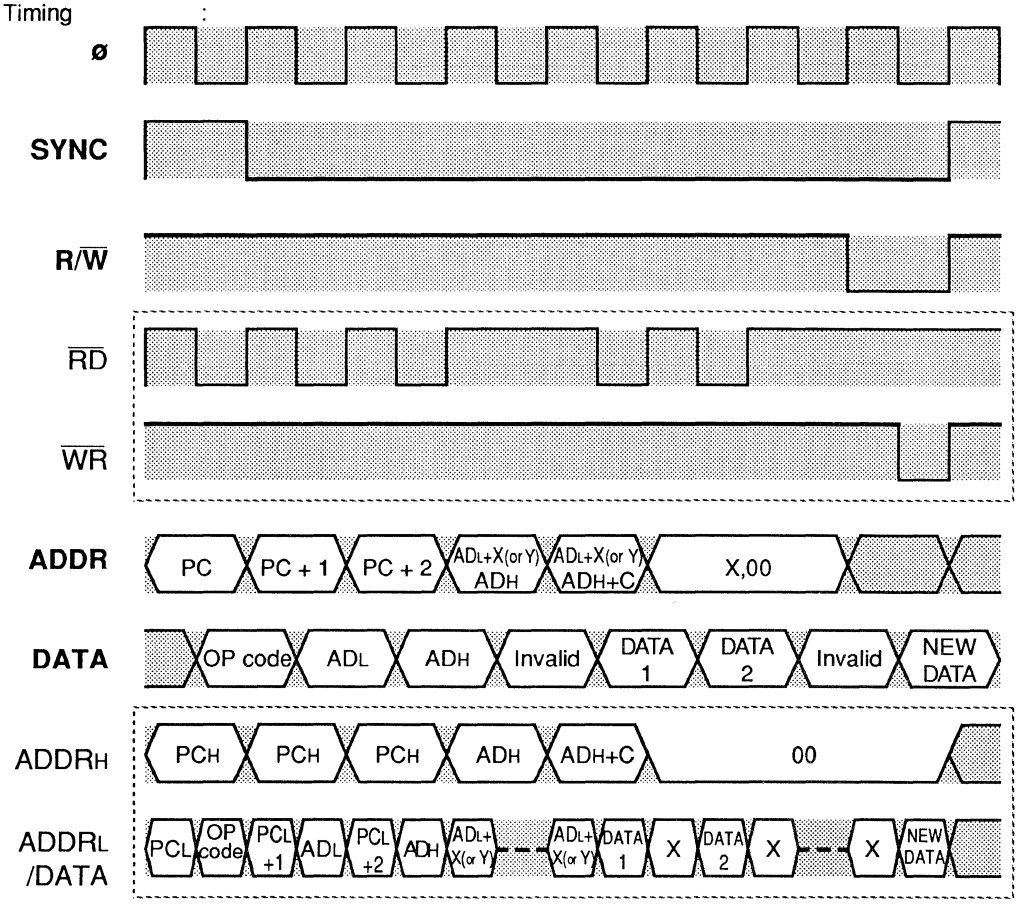
ABSOLUTE X
ABSOLUTE Y

Instructions : **ADCΔ$hhll,X or Y (T=1)**
ANDΔ$hhll,X or Y (T=1)
EORΔ$hhll,X or Y (T=1)
ORAΔ$hhll,X or Y (T=1)
SBCΔ$hhll,X or Y (T=1)

Byte length : **3**

Cycle number : **8**

Timing :

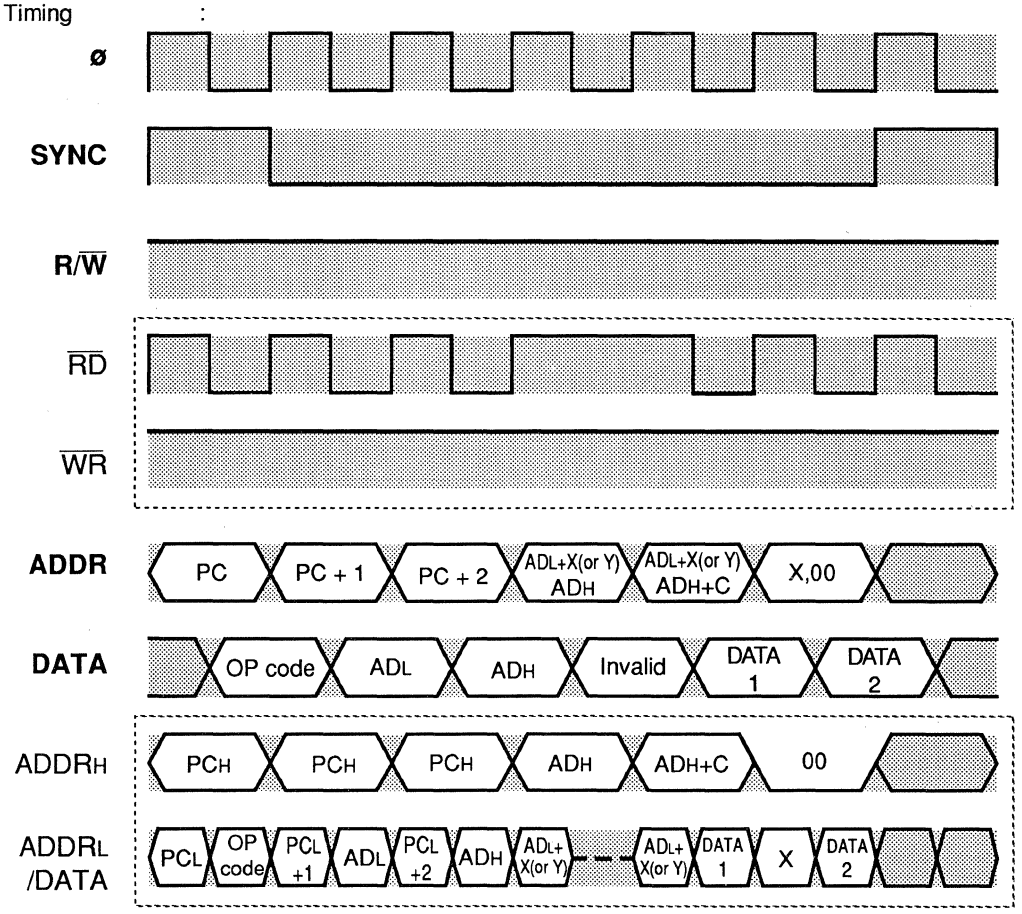


C : Carry of ADL+X or Y

ABSOLUTE X
ABSOLUTE Y

Instruction : **CMPΔ$hhll,X or Y (T=1)**
Byte length : **3**
Cycle number : **6**

Timing :



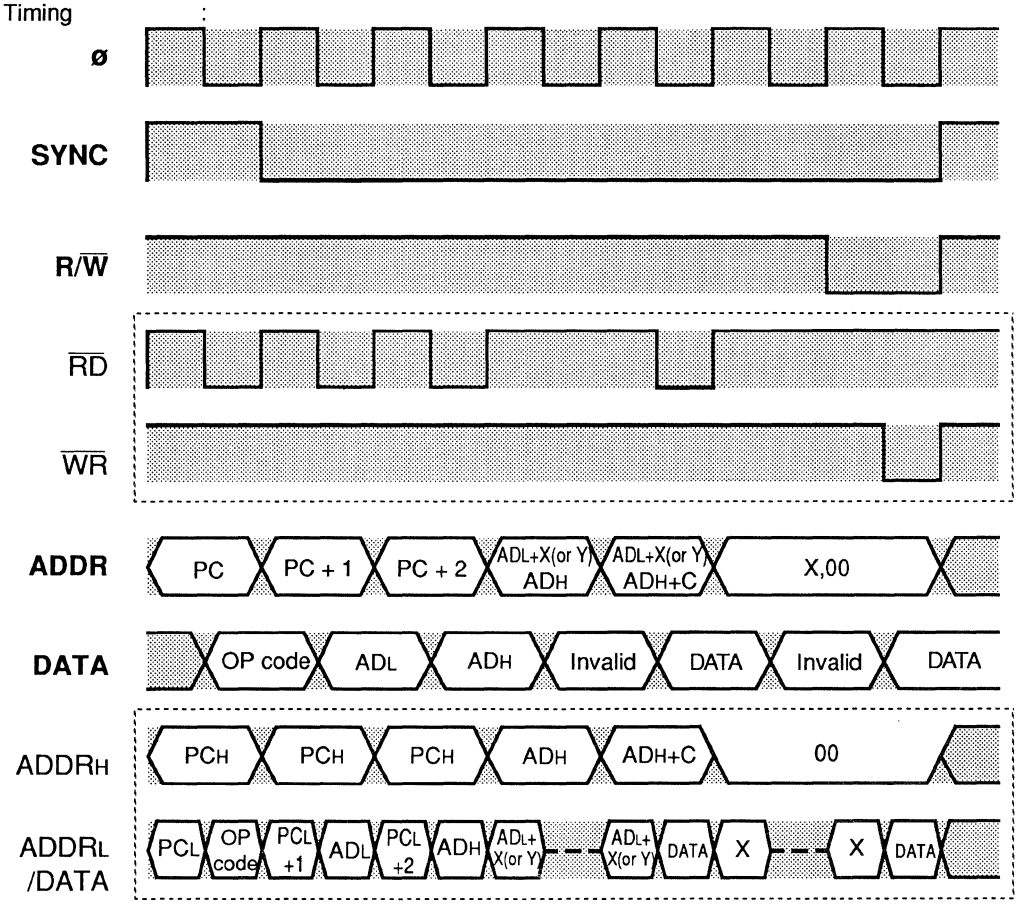
C : Carry of ADL+X or Y

[T=1]

ABSOLUTE X
ABSOLUTE Y

Instruction : **LDAΔ$hhll,X or Y (T=1)**
Byte length : **3**
Cycle number : **7**

Timing :



C : Carry of ADL+X or Y

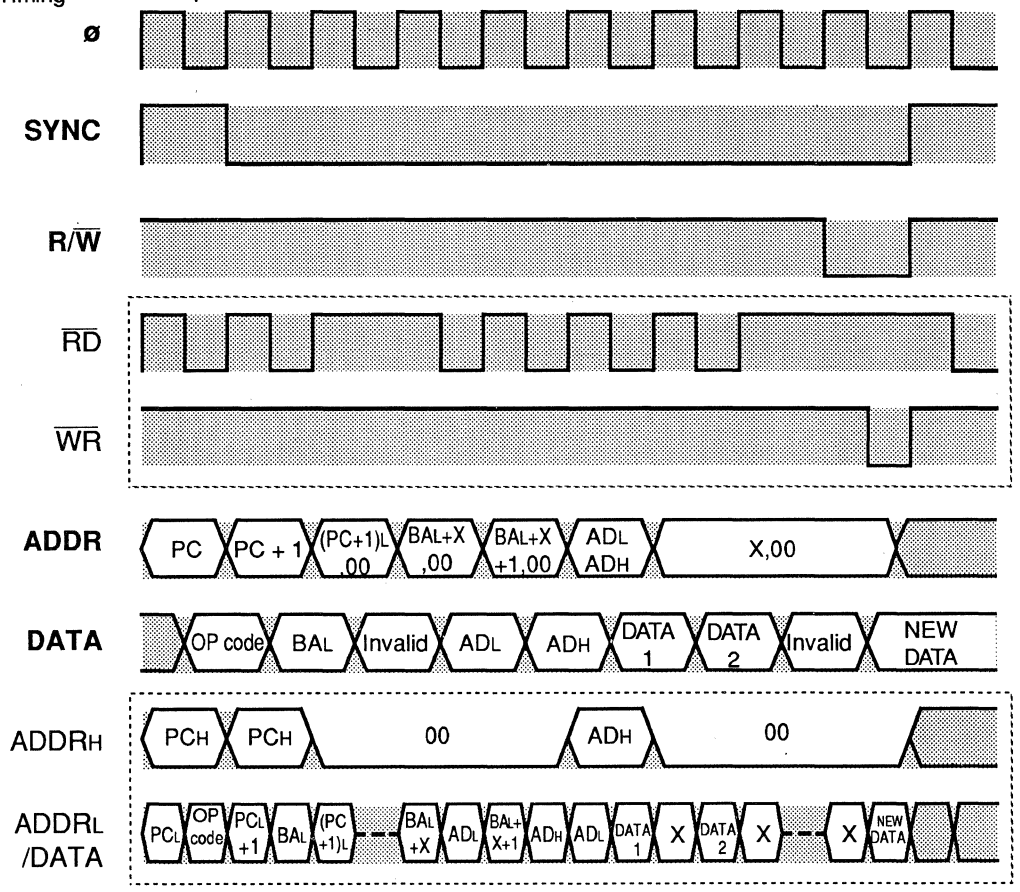
INDIRECT X

Instructions : **ADCΔ($zz,X) (T=1)**
ANDΔ($zz,X) (T=1)
EORΔ($zz,X) (T=1)
ORAΔ($zz,X) (T=1)
SBCΔ($zz,X) (T=1)

Byte length : **2**

Cycle number : **9**

Timing :



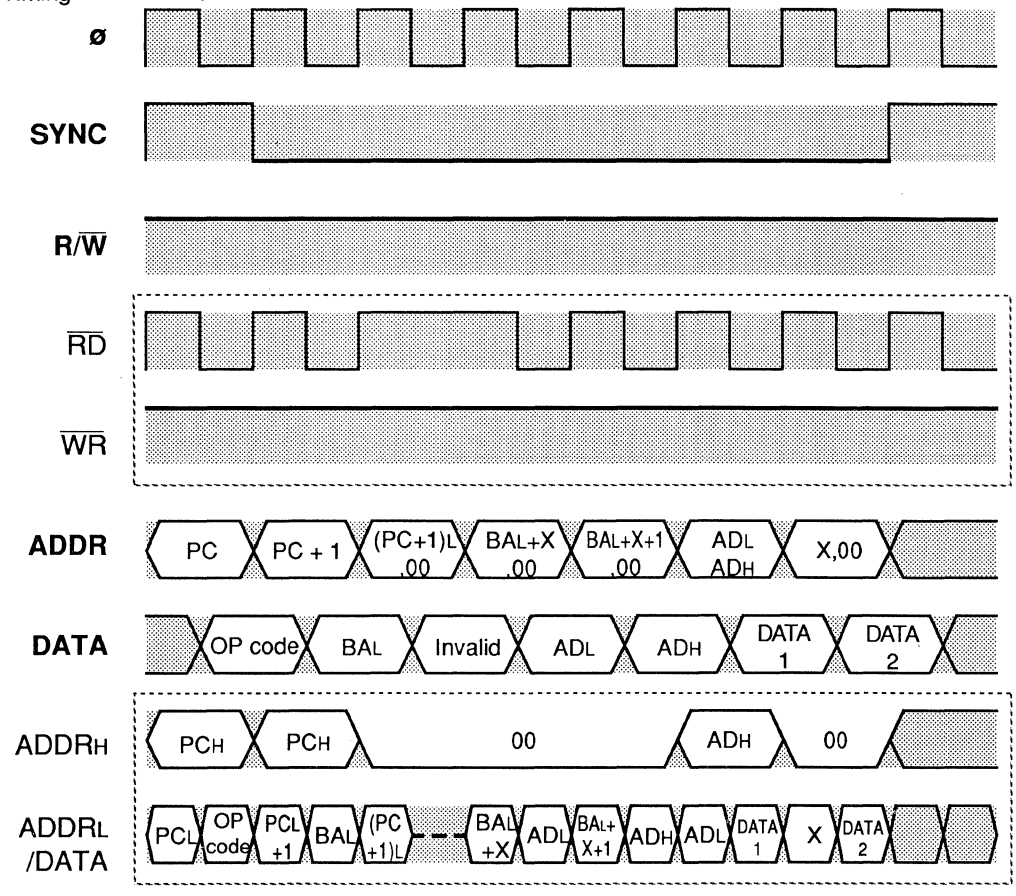
BA : Basic address

[T=1]

INDIRECT X

Instruction : **CMPΔ($zz,X) (T=1)**
Byte length : **2**
Cycle number : **7**

Timing :



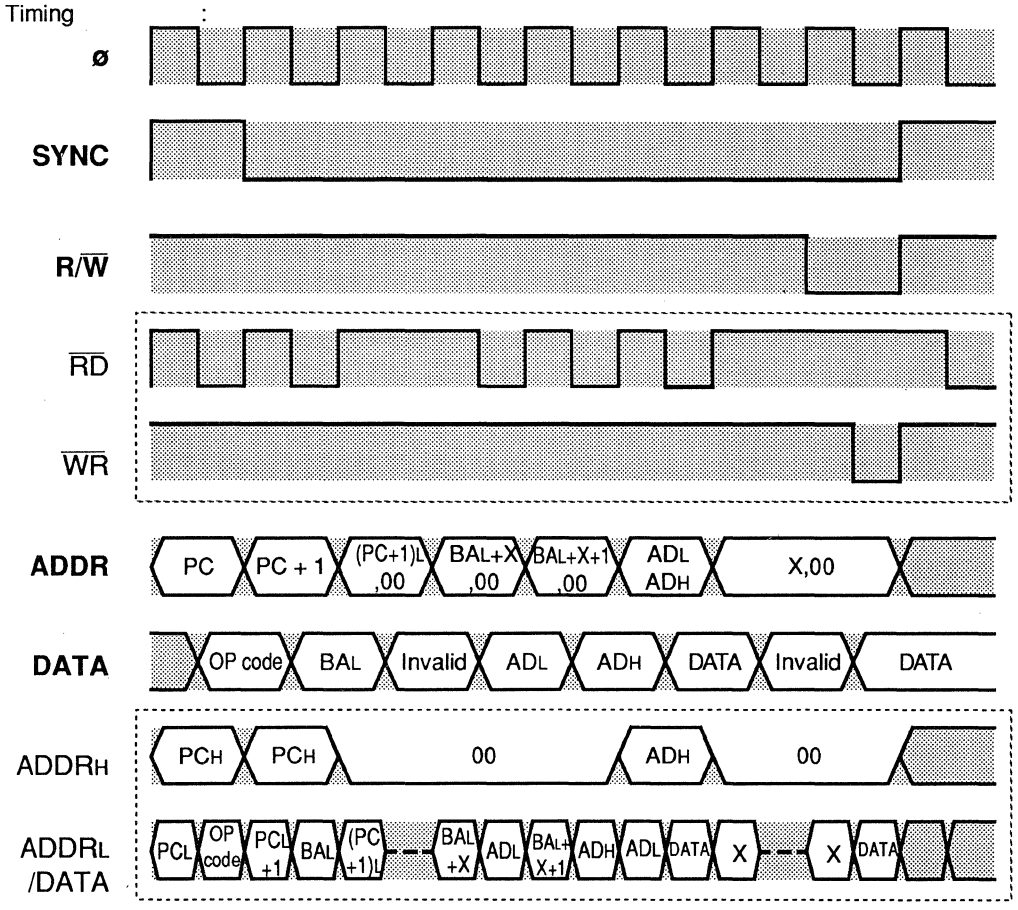
BA : Basic address

[T=1]

INDIRECT X

Instructions : **LDAΔ($zz,X) (T=1)**
Byte length : **2**
Cycle number : **8**

Timing :



BA : Basic address

[T=1]

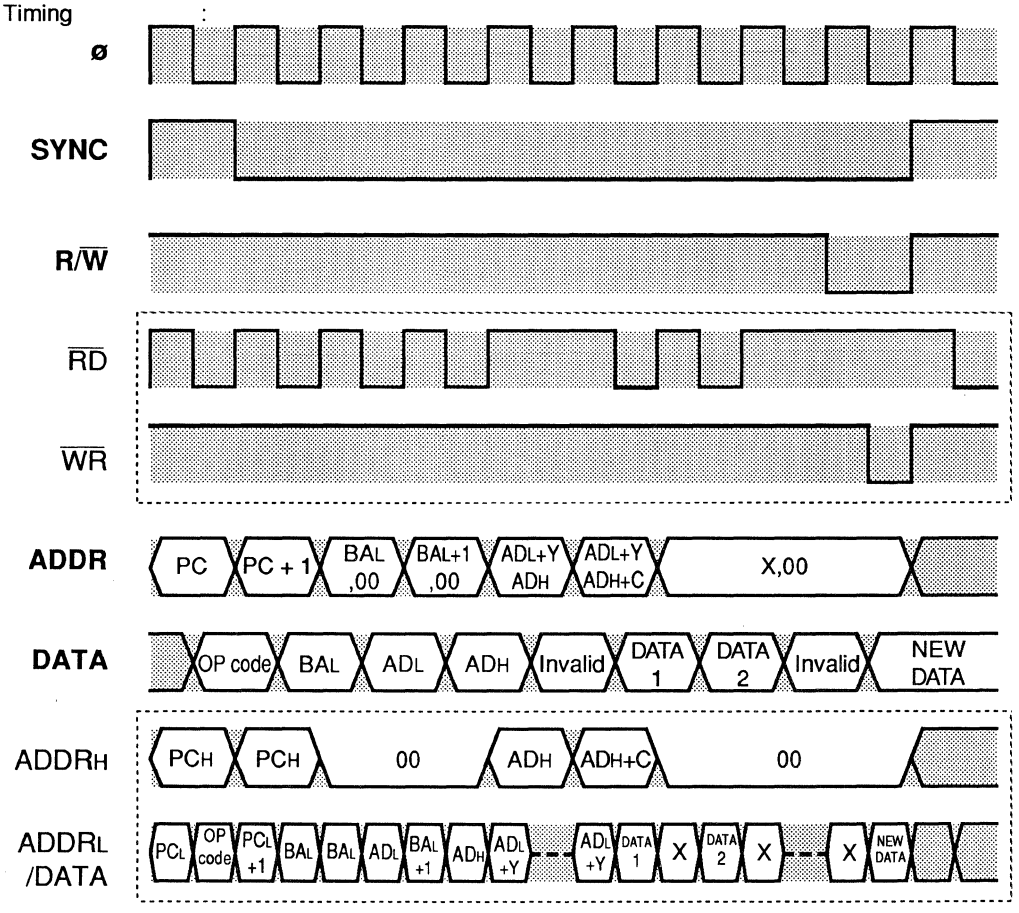
INDIRECT Y

Instructions : **ADCΔ($zz),Y (T=1)**
ANDΔ($zz),Y (T=1)
EORΔ($zz),Y (T=1)
ORAΔ($zz),Y (T=1)
SBCΔ($zz),Y (T=1)

Byte length : **2**

Cycle number : **9**

Timing :



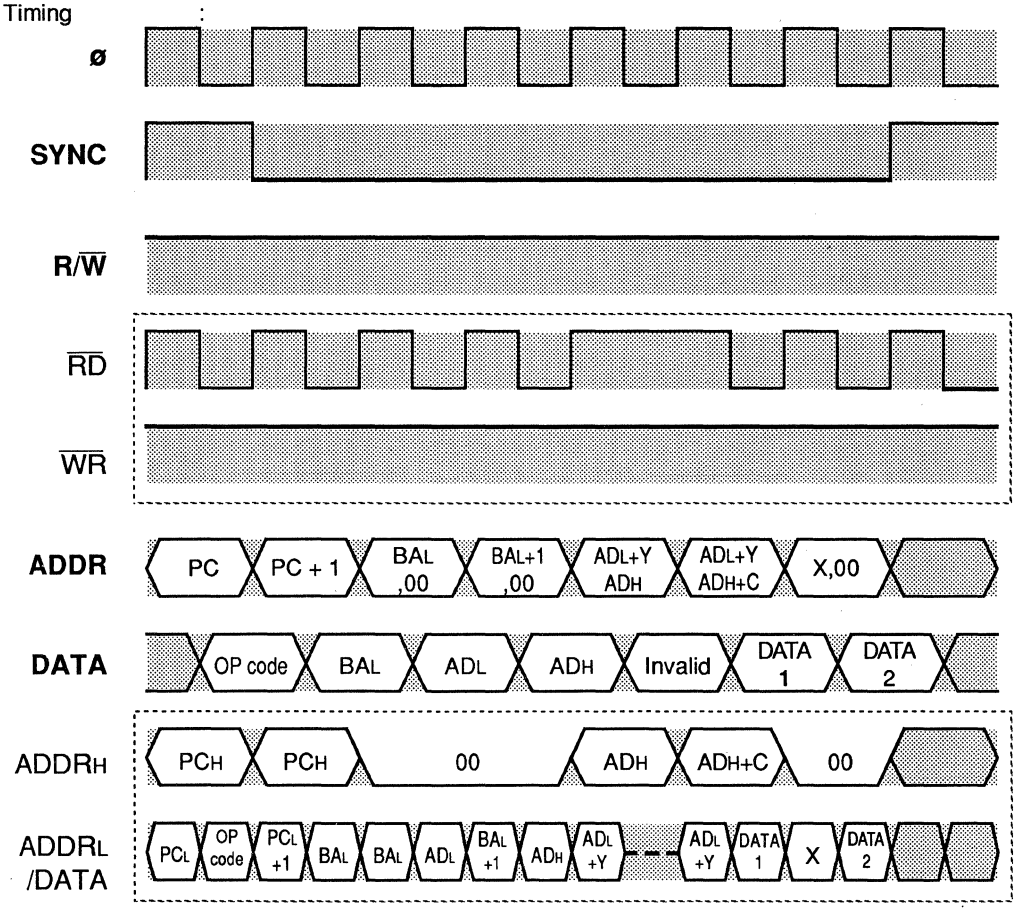
BA : Basic address
C : Carry of ADL+Y

[T=1]

INDIRECT Y

Instruction : **CMPΔ($zz),Y (T=1)**
Byte length : **2**
Cycle number : **7**

Timing :

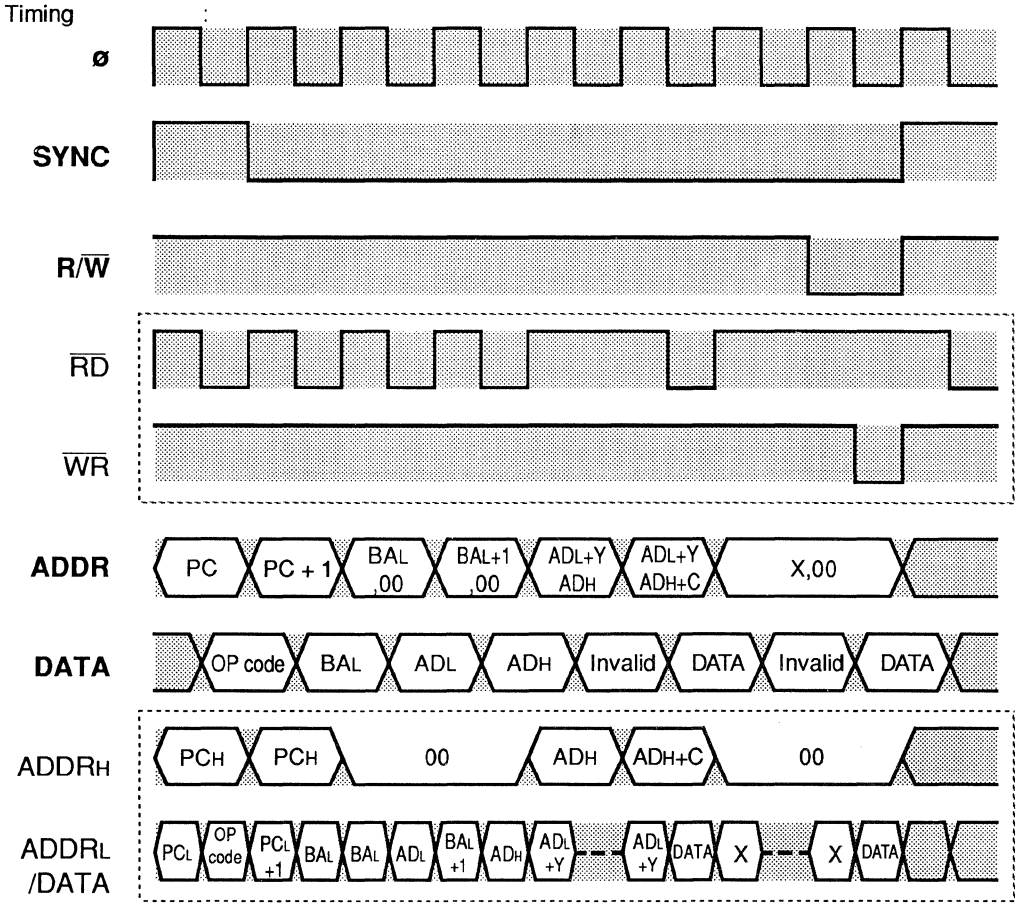


BA : Basic address
C : Carry of ADL+Y

INDIRECT Y

Instructions : **LDAΔ(\$zz),Y (T=1)**
Byte length : **2**
Cycle number : **8**

Timing :



BA : Basic address
C : Carry of ADL+Y

APPENDIX 2

MELPS 740 Machine Language Instruction Table

Parameter Classification		SYMBOL	FUNCTION	FLAG N V T B D I Z C	INSTRUCTION CODE D7 D6 D5 D4 D3 D2 D1 D0	HEX	BYTE NUMBER	CYCLE NUMBER	NOTE
Data Transfer	Load	LDA #$nn	(A) ← nn	OXXXXXOX	1 0 1 0 <B2> 1 0 0 1	A9	2	2	2
		LDA $zz	(A) ← (M) where M=(zz)	OXXXXXOX	1 0 1 0 <B2> 0 1 0 1	A5	2	3	2
		LDA $zz,X	(A) ← (M) where M=(zz+(X))	OXXXXXOX	1 0 1 1 <B2> 0 1 0 1	B5	2	4	2
		LDA $hhll	(A) ← (M) where M=(hhll)	OXXXXXOX	1 0 1 0 <B2> 1 1 0 1 <B3>	AD	3	4	2
		LDA $hhll,X	(A) ← (M) where M=(hhll+(X))	OXXXXXOX	1 0 1 1 <B2> 1 1 0 1 <B3>	BD	3	5	2
		LDA $hhll,Y	(A) ← (M) where M=(hhll+(Y))	OXXXXXOX	1 0 1 1 <B2> 1 0 0 1 <B3>	B9	3	5	2
		LDA ($zz,X)	(A) ← (M) where M=((zz+(X)+1)(zz+(X)))	OXXXXXOX	1 0 1 0 <B2> 0 0 0 1	A1	2	6	2
		LDA ($zz),Y	(A) ← (M) where M=((zz+1)(zz)+(Y))	OXXXXXOX	1 0 1 1 <B2> 0 0 0 1	B1	2	6	2
		LDX #$nn	(X) ← nn	OXXXXXOX	1 0 1 0 <B2> 0 0 1 0	A2	2	2	
		LDX $zz	(X) ← (M) where M=(zz)	OXXXXXOX	1 0 1 0 <B2> 0 1 1 0	A6	2	3	
		LDX $zz,Y	(X) ← (M) where M=(zz+(Y))	OXXXXXOX	1 0 1 1 <B2> 0 1 1 0	B6	2	4	
		LDX $hhll	(X) ← (M) where M=(hhll)	OXXXXXOX	1 0 1 0 <B2> 1 1 1 0 <B3>	AE	3	4	
		LDX $hhll,Y	(X) ← (M) where M=(hhll+(Y))	OXXXXXOX	1 0 1 1 <B2> 1 1 1 0 <B3>	BE	3	5	
		LDY #$nn	(Y) ← nn	OXXXXXOX	1 0 1 0 <B2> 0 0 0 0	A0	2	2	
		LDY $zz	(Y) ← (M) where M=(zz)	OXXXXXOX	1 0 1 0 <B2> 0 1 0 0	A4	2	3	
		LDY $zz,X	(Y) ← (M) where M=(zz+(X))	OXXXXXOX	1 0 1 1 <B2> 0 1 0 0	B4	2	4	
		LDY $hhll	(Y) ← (M) where M=(hhll)	OXXXXXOX	1 0 1 0 <B2> 1 1 0 0 <B3>	AC	3	4	
		LDY $hhll,X	(Y) ← (M) where M=(hhll+(X))	OXXXXXOX	1 0 1 1 <B2> 1 1 0 0 <B3>	BC	3	5	
		LDM #$nn,$zz	(M) ← nn where M=(zz)	XXXXXXXX	0 0 1 1 <B2> 1 1 0 0 <B3>	3C	3	4	
	Store	STA $zz	(M) ← (A) where M=(zz)	XXXXXXXX	1 0 0 0 <B2> 0 1 0 1	85	2	4	
		STA $zz,X	(M) ← (A) where M=(zz+(X))	XXXXXXXX	1 0 0 1 <B2> 0 1 0 1	95	2	5	
		STA $hhll	(M) ← (A) where M=(hhll)	XXXXXXXX	1 0 0 0 <B2> 1 1 0 1 <B3>	8D	3	5	
		STA $hhll,X	(M) ← (A) where M=(hhll+(X))	XXXXXXXX	1 0 0 1 <B2> 1 1 0 1 <B3>	9D	3	6	
		STA $hhll,Y	(M) ← (A) where M=(hhll+(Y))	XXXXXXXX	1 0 0 1 <B2> 1 0 0 1 <B3>	99	3	6	
		STA ($zz,X)	(M) ← (A) where M=((zz+(X)+1)(zz+(X)))	XXXXXXXX	1 0 0 0 <B2> 0 0 0 1	81	2	7	
		STA ($zz),Y	(M) ← (A) where M=((zz+1)(zz)+(Y))	XXXXXXXX	1 0 0 1 <B2> 0 0 0 1	91	2	7	
		STX $zz	(M) ← (X) where M=(zz)	XXXXXXXX	1 0 0 0 <B2> 0 1 1 0	86	2	4	
		STX $zz,Y	(M) ← (X) where M=(zz+(Y))	XXXXXXXX	1 0 0 1 <B2> 0 1 1 0	96	2	5	
		STX $hhll	(M) ← (X) where M=(hhll)	XXXXXXXX	1 0 0 0 <B2> 1 1 1 0 <B3>	8E	3	5	
		STY $zz	(M) ← (Y) where M=(zz)	XXXXXXXX	1 0 0 0 <B2> 0 1 0 0	84	2	4	
		STY $zz,X	(M) ← (Y) where M=(zz+(X))	XXXXXXXX	1 0 0 1 <B2> 0 1 0 0	94	2	5	
		STY $hhll	(M) ← (Y) where M=(hhll)	XXXXXXXX	1 0 0 0 <B2> 1 1 0 0 <B3>	8C	3	6	
	Transfer	TAX	(X) ← (A)	OXXXXXOX	1 0 1 0 1 0 1 0	AA	1	2	
		TXA	(A) ← (X)	OXXXXXOX	1 0 0 0 1 0 1 0	8A	1	2	
		TAY	(Y) ← (A)	OXXXXXOX	1 0 1 0 1 0 0 0	A8	1	2	
		TYA	(A) ← (Y)	OXXXXXOX	1 0 0 1 1 0 0 0	98	1	2	
		TSX	(X) ← (S)	OXXXXXOX	1 0 1 1 1 0 1 0	BA	1	2	
		TXS	(S) ← (X)	XXXXXXXX	1 0 0 1 1 0 1 0	9A	1	2	
	Stack Operation	PHA	(M(S)) ← (A) , (S) ← (S) — 1	XXXXXXXX	0 1 0 0 1 0 0 0	48	1	3	
		PHP	(M(S)) ← (PS) , (S) ← (S) — 1	XXXXXXXX	0 0 0 0 1 0 0 0	08	1	3	
		PLA	(S) ← (S) + 1 , (A) ← (M(S))	OXXXXXOX	0 1 1 0 1 0 0 0	68	1	4	
		PLP	(S) ← (S) + 1 , (PS) ← (M(S))	Previous status in stack	0 0 1 0 1 0 0 0	28	1	4	

Parameter Classification		SYMBOL	FUNCTION	FLAG N V T B D I Z C	INSTRUCTION CODE D7 D6 D5 D4 D3 D2 D1 D0	HEX	BYTE NUMBER	CYCLE NUMBER	NOTE
Operation	Add and Substruct	ADC #$nn	(A) ← (A) + nn + (C)	ooxxxxoo	0 1 1 0 <B2>1 0 0 1	69	2	2	1
		ADC $zz	(A) ← (A) + (M) + (C) where M=(zz)	ooxxxxoo	0 1 1 0 <B2>0 1 0 1	65	2	3	1
		ADC $zz,X	(A) ← (A) + (M) + (C) where M=(zz+(X))	ooxxxxoo	0 1 1 1 <B2>0 1 0 1	75	2	4	1
		ADC $hhll	(A) ← (A) + (M) + (C) where M=(hhll)	ooxxxxoo	0 1 1 0 <B2> <B3>1 1 0 1	6D	3	4	1
		ADC $hhll,X	(A) ← (A) + (M) + (C) where M=(hhll+(X))	ooxxxxoo	0 1 1 1 <B2> <B3>1 1 0 1	7D	3	5	1
		ADC $hhll,Y	(A) ← (A) + (M) + (C) where M=(hhll+(Y))	ooxxxxoo	0 1 1 1 <B2> <B3>1 0 0 1	79	3	5	1
		ADC ($zz,X)	(A) ← (A) + (M) + (C) where M=((zz+(X)+1)(zz+(X)))	ooxxxxoo	0 1 1 0 <B2>0 0 0 1	61	2	6	1
		ADC ($zz),Y	(A) ← (A) + (M) + (C) where M=((zz+1)(zz)+(Y))	ooxxxxoo	0 1 1 1 <B2>0 0 0 1	71	2	6	1
		SBC #$nn	(A) ← (A) — nn — $\overline{(C)}$	ooxxxxoo	1 1 1 0 <B2>1 0 0 1	E9	2	2	1
		SBC $zz	(A) ← (A) — (M) — $\overline{(C)}$ where M=(zz)	ooxxxxoo	1 1 1 0 <B2>0 1 0 1	E5	2	3	1
		SBC $zz,X	(A) ← (A) — (M) — $\overline{(C)}$ where M=(zz+(X))	ooxxxxoo	1 1 1 1 <B2>0 1 0 1	F5	2	4	1
		SBC $hhll	(A) ← (A) — (M) — $\overline{(C)}$ where M=(hhll)	ooxxxxoo	1 1 1 0 <B2> <B3>1 1 0 1	ED	3	4	1
		SBC $hhll,X	(A) ← (A) — (M) — $\overline{(C)}$ where M=(hhll+(X))	ooxxxxoo	1 1 1 1 <B2> <B3>1 1 0 1	FD	3	5	1
		SBC $hhll,Y	(A) ← (A) — (M) — $\overline{(C)}$ where M=(hhll+(Y))	ooxxxxoo	1 1 1 1 <B2> <B3>1 0 0 1	F9	3	5	1
		SBC ($zz,X)	(A) ← (A) — (M) — $\overline{(C)}$ where M=((zz+(X)+1)(zz+(X)))	ooxxxxoo	1 1 1 0 <B2>0 0 0 1	E1	2	6	1
		SBC ($zz),Y	(A) ← (A) — (M) — $\overline{(C)}$ where M=((zz+1)(zz)+(Y))	ooxxxxoo	1 1 1 1 <B2>0 0 0 1	F1	2	6	1
		INC A	(A) ← (A) + 1	oxxxxxox	0 0 1 1 1 0 1 0	3A	1	2	
		INC $zz	(M) ← (M) + 1 where M=(zz)	oxxxxxox	1 1 1 0 <B2>0 1 1 0	E6	2	5	
		INC $zz,X	(M) ← (M) + 1 where M=(zz+(X))	oxxxxxox	1 1 1 1 <B2>0 1 1 0	F6	2	6	
		INC $hhll	(M) ← (M) + 1 where M=(hhll)	oxxxxxox	1 1 1 0 <B2> <B3>1 1 1 0	EE	3	6	
		INC $hhll,X	(M) ← (M) + 1 where M=(hhll+(X))	oxxxxxox	1 1 1 1 <B2> <B3>1 1 1 0	FE	3	7	
		DEC A	(A) ← (A) — 1	oxxxxxox	0 0 0 1 1 0 1 0	1A	1	2	
		DEC $zz	(M) ← (M) — 1 where M=(zz)	oxxxxxox	1 1 0 0 <B2>0 1 1 0	C6	2	5	
		DEC $zz,X	(M) ← (M) — 1 where M=(zz+(X))	oxxxxxox	1 1 0 1 <B2>0 1 1 0	D6	2	6	
		DEC $hhll	(M) ← (M) — 1 where M=(hhll)	oxxxxxox	1 1 0 0 <B2> <B3>1 1 1 0	CE	3	6	
		DEC $hhll,X	(M) ← (M) — 1 where M=(hhll+(X))	oxxxxxox	1 1 0 1 <B2> <B3>1 1 1 0	DE	3	7	
		INX	(X) ← (X) + 1	oxxxxxox	1 1 1 0 1 0 0 0	E8	1	2	
		DEX	(X) ← (X) — 1	oxxxxxox	1 1 0 0 1 0 1 0	CA	1	2	
		INY	(Y) ← (Y) + 1	oxxxxxox	1 1 0 0 1 0 0 0	C8	1	2	
		DEY	(Y) ← (Y) — 1	oxxxxxox	1 0 0 0 1 0 0 0	88	1	2	
	Multiply / Divide	MUL $zz,X	M(S), (A) ← (A)× M(zz+X) (S) ← (S) — 1	xxxxxxxx	0 1 1 0 0 0 1 0	62	2	15	8
		DIV $zz,X	(A) ← (M(zz+X+1), M(zz+X))÷(A) M(S) ← One's complement of remainder (S) ← (S) — 1	xxxxxxxx	1 1 1 0 0 0 1 0	E2	2	16	8

Parameter / Classification		SYMBOL	FUNCTION	FLAG N V T B D I Z C	INSTRUCTION CODE D7 D6 D5 D4 D3 D2 D1 D0	HEX	BYTE NUMBER	CYCLE NUMBER	NOTE
Operation	Logic Operation	AND #$nn	(A) ← (A) ∧ nn	OXXXXXOX	0 0 1 0 <B2>1 0 0 1	29	2	2	1
		AND $zz	(A) ← (A) ∧ (M) where M=(zz)	OXXXXXOX	0 0 1 0 <B2>0 1 0 1	25	2	3	1
		AND $zz,X	(A) ← (A) ∧ (M) where M=(zz+(X))	OXXXXXOX	0 0 1 1 <B2>0 1 0 1	35	2	4	1
		AND $hhll	(A) ← (A) ∧ (M) where M=(hhll)	OXXXXXOX	0 0 1 0 <B2><B3>1 1 0 1	2D	3	4	1
		AND $hhll,X	(A) ← (A) ∧ (M) where M=(hhll+(X))	OXXXXXOX	0 0 1 1 <B2><B3>1 1 0 1	3D	3	5	1
		AND $hhll,Y	(A) ← (A) ∧ (M) where M=(hhll+(Y))	OXXXXXOX	0 0 1 1 <B2><B3>1 0 0 1	39	3	5	1
		AND ($zz,X)	(A) ← (A) ∧ (M) where M=((zz+(X)+1)(zz+(X)))	OXXXXXOX	0 0 1 0 <B2>0 0 0 1	21	2	6	1
		AND ($zz),Y	(A) ← (A) ∧ (M) where M=((zz+1)(zz)+(Y))	OXXXXXOX	0 0 1 1 <B2>0 0 0 1	31	2	6	1
		ORA #$nn	(A) ← (A) ∨ nn	OXXXXXOX	0 0 0 0 <B2>1 0 0 1	09	2	2	1
		ORA $zz	(A) ← (A) ∨ (M) where M=(zz)	OXXXXXOX	0 0 0 0 <B2>0 1 0 1	05	2	3	1
		ORA $zz,X	(A) ← (A) ∨ (M) where M=(zz+(X))	OXXXXXOX	0 0 0 1 <B2>0 1 0 1	15	2	4	1
		ORA $hhll	(A) ← (A) ∨ (M) where M=(hhll)	OXXXXXOX	0 0 0 0 <B2><B3>1 1 0 1	0D	3	4	1
		ORA $hhll,X	(A) ← (A) ∨ (M) where M=(hhll+(X))	OXXXXXOX	0 0 0 1 <B2><B3>1 1 0 1	1D	3	5	1
		ORA $hhll,Y	(A) ← (A) ∨ (M) where M=(hhll+(Y))	OXXXXXOX	0 0 0 1 <B2><B3>1 0 0 1	19	3	5	1
		ORA ($zz,X)	(A) ← (A) ∨ (M) where M=((zz+(X)+1)(zz+(X)))	OXXXXXOX	0 0 0 0 <B2>0 0 0 1	01	2	6	1
		ORA ($zz),Y	(A) ← (A) ∨ (M) where M=((zz+1)(zz)+(Y))	OXXXXXOX	0 0 0 1 <B2>0 0 0 1	11	2	6	1
		EOR #$nn	(A) ← (A) ∀ nn	OXXXXXOX	0 1 0 0 <B2>1 0 0 1	49	2	2	1
		EOR $zz	(A) ← (A) ∀ (M) where M=(zz)	OXXXXXOX	0 1 0 0 <B2>0 1 0 1	45	2	3	1
		EOR $zz,X	(A) ← (A) ∀ (M) where M=(zz+(X))	OXXXXXOX	0 1 0 1 <B2>0 1 0 1	55	2	4	1
		EOR $hhll	(A) ← (A) ∀ (M) where M=(hhll)	OXXXXXOX	0 1 0 0 <B2><B3>1 1 0 1	4D	3	4	1
		EOR $hhll,X	(A) ← (A) ∀ (M) where M=(hhll+(X))	OXXXXXOX	0 1 0 1 <B2><B3>1 1 0 1	5D	3	5	1
		EOR $hhll,Y	(A) ← (A) ∀ (M) where M=(hhll+(Y))	OXXXXXOX	0 1 0 1 <B2><B3>1 0 0 1	59	3	5	1
		EOR ($zz,X)	(A) ← (A) ∀ (M) where M=((zz+(X)+1)(zz+(X)))	OXXXXXOX	0 1 0 0 <B2>0 0 0 1	41	2	6	1
		EOR ($zz),Y	(A) ← (A) ∀ (M) where M=((zz+1)(zz)+(Y))	OXXXXXOX	0 1 0 1 <B2>0 0 0 1	51	2	6	1
		COM $zz	(M) ← $\overline{(M)}$ where M=(zz)	OXXXXXOX	0 1 0 0 <B2>0 1 0 0	44	2	5	
		BIT $zz	(A) ∧ (M) where M=(zz)	M_7 M_6 XXXXXOX	0 0 1 0 <B2>0 1 0 0	24	2	3	
		BIT $hhll	(A) ∧ (M) where M=(hhll)	M_7 M_6 XXXXXOX	0 0 1 0 <B2><B3>1 1 0 0	2C	3	4	
		TST $zz	(M)=0 ? where M=(zz)	OXXXXXOX	0 1 1 0 <B2>0 1 0 0	64	2	3	

MELPS 740 Machine Language Instruction Table

Parameter Classification		SYMBOL	FUNCTION	FLAG NVTBDIZC	INSTRUCTION CODE D7 D6 D5 D4 D3 D2 D1 D0	HEX	BYTE NUMBER	CYCLE NUMBER	NOTE
Operation	Comparison	CMP #$nn	(A) — nn	oxxxxxoo	1 1 0 0 <B2> 1 0 0 1	C9	2	2	3
		CMP $zz	(A) — (M) where M=(zz)	oxxxxxoo	1 1 0 0 <B2> 0 1 0 1	C5	2	3	3
		CMP $zz,X	(A) — (M) where M=(zz+(X))	oxxxxxoo	1 1 0 1 <B2> 0 1 0 1	D5	2	4	3
		CMP $hhll	(A) — (M) where M=(hhll)	oxxxxxoo	1 1 0 0 <B2> <B3> 1 1 0 1	CD	3	4	3
		CMP $hhll,X	(A) — (M) where M=(hhll+(X))	oxxxxxoo	1 1 0 1 <B2> <B3> 1 1 0 1	DD	3	5	3
		CMP $hhll,Y	(A) — (M) where M=(hhll+(Y))	oxxxxxoo	1 1 0 1 <B2> <B3> 1 0 0 1	D9	3	5	3
		CMP ($zz,X)	(A) — (M) where M=((zz+(X)+1)(zz+(X)))	oxxxxxoo	1 1 0 0 <B2> 0 0 0 1	C1	2	6	3
		CMP ($zz),Y	(A) — (M) where M=((zz+1)(zz)+(Y))	oxxxxxoo	1 1 0 1 <B2> 0 0 0 1	D1	2	6	3
		CPX #$nn	(X) — nn	oxxxxxoo	1 1 1 0 <B2> 0 0 0 0	E0	2	2	
		CPX $zz	(X) — (M) where M=(zz)	oxxxxxoo	1 1 1 0 <B2> 0 1 0 0	E4	2	3	
		CPX $hhll	(X) — (M) where M=(hhll)	oxxxxxoo	1 1 1 0 <B2> <B3> 1 1 0 0	EC	3	4	
		CPY #$nn	(Y) — nn	oxxxxxoo	1 1 0 0 <B2> 0 0 0 0	C0	2	2	
		CPY $zz	(Y) — (M) where M=(zz)	oxxxxxoo	1 1 0 0 <B2> 0 1 0 0	C4	2	3	
		CPY $hhll	(Y) — (M) where M=(hhll)	oxxxxxoo	1 1 0 0 <B2> <B3> 1 1 0 0	CC	3	4	
	Rotate and Shift	ASL A	Left Shift C ← A7 A6 … A1 A0 ← 0	oxxxxxoo	0 0 0 0 1 0 1 0	0A	1	2	
		ASL $zz	where M=(zz)	oxxxxxoo	0 0 0 0 <B2> 0 1 1 0	06	2	5	
		ASL $zz,X	Left Shift C ← M7 M6 … M1 M0 ← 0 where M=(zz+(X))	oxxxxxoo	0 0 0 1 <B2> 0 1 1 0	16	2	6	
		ASL $hhll	where M=(hhll)	oxxxxxoo	0 0 0 0 <B2> <B3> 1 1 1 0	0E	3	6	
		ASL $hhll,X	where M=(hhll+(X))	oxxxxxoo	0 0 0 1 <B2> <B3> 1 1 1 0	1E	3	7	
		LSR A	Right Shift 0 → A7 A6 … A1 A0 → C	oxxxxxoo	0 1 0 0 1 0 1 0	4A	1	2	
		LSR $zz	where M=(zz)	oxxxxxoo	0 1 0 0 <B2> 0 1 1 0	46	2	5	
		LSR $zz,X	Right Shift 0 → M7 M6 … M1 M0 → C where M=(zz+(X))	oxxxxxoo	0 1 0 1 <B2> 0 1 1 0	56	2	6	
		LSR $hhll	where M=(hhll)	oxxxxxoo	0 1 0 0 <B2> <B3> 1 1 1 0	4E	3	6	
		LSR $hhll,X	where M=(hhll+(X))	oxxxxxoo	0 1 0 1 <B2> <B3> 1 1 1 0	5E	3	7	
		ROL A	Left Shift ← A7 A6 … A1 A0 ← C	oxxxxxoo	0 0 1 0 1 0 1 0	2A	1	2	
		ROL $zz	where M=(zz)	oxxxxxoo	0 0 1 0 <B2> 0 1 1 0	26	2	5	
		ROL $zz,X	Left Shift ← M7 M6 … M1 M0 ← C where M=(zz+(X))	oxxxxxoo	0 0 1 1 <B2> 0 1 1 0	36	2	6	
		ROL $hhll	where M=(hhll)	oxxxxxoo	0 0 1 0 <B2> <B3> 1 1 1 0	2E	3	6	
		ROL $hhll,X	where M=(hhll+(X))	oxxxxxoo	0 0 1 1 <B2> <B3> 1 1 1 0	3E	3	7	
		ROR A	Right Shift → C → A7 A6 … A1 A0	oxxxxxoo	0 1 1 0 1 0 1 0	6A	1	2	
		ROR $zz	where M=(zz)	oxxxxxoo	0 1 1 0 <B2> 0 1 1 0	66	2	5	
		ROR $zz,X	Right Shift → C → M7 M6 … M1 M0 → where M=(zz+(X))	oxxxxxoo	0 1 1 1 <B2> 0 1 1 0	76	2	6	
		ROR $hhll	where M=(hhll)	oxxxxxoo	0 1 1 0 <B2> <B3> 1 1 1 0	6E	3	6	
		ROR $hhll,X	where M=(hhll+(X))	oxxxxxoo	0 1 1 1 <B2> <B3> 1 1 1 0	7E	3	7	
		RRF $zz	M7 M4 M3 M0 where M=(zz)	xxxxxxxx	1 0 0 0 <B2> 0 0 1 0	82	2	8	
Bit Management		CLB i,A	(Ai) ← 0 where i=0 ~ 7	xxxxxxxx	i i i 1 1 0 1 1	(2i+1)×10+B	1	2	
		CLB i,$zz	(Mi) ← 0 where i=0 ~ 7, M=(zz)	xxxxxxxx	i i i 1 <B2> 1 1 1 1	(2i+1)×10+F	2	5	
		SEB i,A	(Ai) ← 1 where i=0 ~ 7	xxxxxxxx	i i i 0 1 0 1 1	2i×10+B	1	2	
		SEB i,$zz	(Mi) ← 1 where i=0 ~ 7, M=(zz)	xxxxxxxx	i i i 0 <B2> 1 1 1 1	2i×10+F	2	5	

Parameter Classification		SYMBOL	FUNCTION	FLAG N V T B D I Z C	INSTRUCTION CODE D7 D6 D5 D4 D3 D2 D1 D0	HEX	BYTE NUMBER	CYCLE NUMBER	NOTE
Flag Setting		CLC	(C) ← 0	xxxxxxx0	0 0 0 1 1 0 0 0	18	1	2	
		SEC	(C) ← 1	xxxxxxx1	0 0 1 1 1 0 0 0	38	1	2	
		CLD	(D) ← 0	xxxx0xxx	1 1 0 1 1 0 0 0	D8	1	2	
		SED	(D) ← 1	xxxx1xxx	1 1 1 1 1 0 0 0	F8	1	2	
		CLI	(I) ← 0	xxxxx0xx	0 1 0 1 1 0 0 0	58	1	2	
		SEI	(I) ← 1	xxxxx1xx	0 1 1 1 1 0 0 0	78	1	2	
		CLT	(T) ← 0	xx0xxxxx	0 0 0 1 0 0 1 0	12	1	2	
		SET	(T) ← 1	xx1xxxxx	0 0 1 1 0 0 1 0	32	1	2	
		CLV	(V) ← 0	x0xxxxxx	1 0 1 1 1 0 0 0	B8	1	2	
Branch and Return	Jump	BRA $hhll	(PC) ← (PC) + 2 + Rel	xxxxxxxx	1 0 0 0 <B2>0 0 0 0	80	2	4	
		JMP $hhll	(PC) ←hhll	xxxxxxxx	0 1 0 0 <B2>1 <B3>1 0 0	4C	3	3	
		JMP ($hhll)	(PCL) ←(hhll), (PCH) ←(hhll+1)	xxxxxxxx	0 1 1 0 <B2>1 <B3>1 0 0	6C	3	5	
		JMP ($zz)	(PCL) ←(zz), (PCH) ←(zz+1)	xxxxxxxx	1 0 1 1 <B2>0 0 1 0	B2	2	4	
		JSR $hhll	(M(S)) ←(PCH), (S) ←(S)—1, (M(S)) ←(PCL), (S) ←(S)—1, and (PC) ← hhll	xxxxxxxx	0 0 1 0 <B2>1 <B3>1 0 0	20	3	6	
		JSR ($zz)	(M(S)) ←(PCH), (S) ←(S)—1, (M(S)) ←(PCL), (S) ←(S)—1, (PCL) ← (zz), and (PCH) ← (zz+1)	xxxxxxxx	0 0 0 0 <B2>0 0 1 0	02	2	7	
		JSR \$hhll	(M(S)) ←(PCH), (S) ←(S)—1, (M(S)) ←(PCL), (S) ←(S)—1, (PCL) ← ll, and (PCH) ←FF	xxxxxxxx	0 0 1 0 <B2>0 0 1 0	22	2	5	
	Branch	BBC i,A,$hhll	When(Ai)=0 (PC) ← (PC) + 2 + Rel Where i=0~7 When(Ai)=1 (PC) ← (PC) + 2	xxxxxxxx	i i i 1 <B2>0 0 0 0	(2i+1)×10+3	2	4	4
		BBC i,$zz,$hhll	When(Mi)=0 (PC) ← (PC) + 3 + Rel Where i=0~7 When(Mi)=1 (PC) ← (PC) + 3	xxxxxxxx	i i i 1 <B2>1 <B3>1 0 0	(2i+1)×10+7	3	5	4
		BBS i,A,$hhll	When(Ai)=1 (PC) ← (PC) + 2 + Rel Where i=0~7 When(Ai)=0 (PC) ← (PC) + 2	xxxxxxxx	i i i 0 <B2>0 0 0 0	2i×10+3	2	4	4
		BBS i,$zz,$hhll	When(Mi)=1 (PC) ← (PC) + 3 + Rel Where i=0~7 When(Mi)=0 (PC) ← (PC) + 3	xxxxxxxx	i i i 0 <B2>1 <B3>1 0 0	2i×10+7	3	5	4
		BCC $hhll	When(C)=0 (PC) ← (PC) + 2 + Rel When(C)=1 (PC) ← (PC) + 2	xxxxxxxx	1 0 0 1 <B2>0 0 0 0	90	2	2	4
		BCS $hhll	When(C)=1 (PC) ← (PC) + 2 + Rel When(C)=0 (PC) ← (PC) + 2	xxxxxxxx	1 0 1 1 <B2>0 0 0 0	B0	2	2	4
		BNE $hhll	When(Z)=0 (PC) ← (PC) + 2 + Rel When(Z)=1 (PC) ← (PC) + 2	xxxxxxxx	1 1 0 1 <B2>0 0 0 0	D0	2	2	4
		BEQ $hhll	When(Z)=1 (PC) ← (PC) + 2 + Rel When(Z)=0 (PC) ← (PC) + 2	xxxxxxxx	1 1 1 1 <B2>0 0 0 0	F0	2	2	4
		BPL $hhll	When(N)=0 (PC) ← (PC) + 2 + Rel When(N)=1 (PC) ← (PC) + 2	xxxxxxxx	0 0 0 1 <B2>0 0 0 0	10	2	2	4
		BMI $hhll	When(N)=1 (PC) ← (PC) + 2 + Rel When(N)=0 (PC) ← (PC) + 2	xxxxxxxx	0 0 1 1 <B2>0 0 0 0	30	2	2	4
		BVC $hhll	When(V)=0 (PC) ← (PC) + 2 + Rel When(V)=1 (PC) ← (PC) + 2	xxxxxxxx	0 1 0 1 <B2>0 0 0 0	50	2	2	4
		BVS $hhll	When(V)=1 (PC) ← (PC) + 2 + Rel When(V)=0 (PC) ← (PC) + 2	xxxxxxxx	0 1 1 1 <B2>0 0 0 0	70	2	2	4
	Return	RTI	(S) ←(S) + 1, (PS) ←(M(S)), (S) ←(S) + 1, (PCL) ←(M(S)), (S) ←(S) + 1, and (PCH) ← (M(S))	Previous status in stack	0 1 0 0 0 0 0 0	40	1	6	
		RTS	(S) ←(S) + 1, (PCL) ←(M(S)), (S) ←(S) + 1, (PCH) ←(M(S)), and (PC) ← (PC) +1	xxxxxxxx	0 1 1 0 0 0 0 0	60	1	6	
Interrupt		BRK	(B) ←1, (PC) ←(PC) + 2, (M(S)) ←(PCH), S ← S—1, (M(S)) ← (PCL), S ← S—1, (M(S)) ←(PS), S ←S—1, (I) ←1, and (PC) ← BADRS	xxx1x1xx	0 0 0 0 0 0 0 0	00	1	7	
Other		NOP	(PC) ← (PC) + 1	xxxxxxxx	1 1 1 0 1 0 1 0	EA	1	2	
Special		FST	Connect XOUTF pin with internal oscillation circuit.	xxxxxxxx	1 1 1 0 0 0 1 0	E2	1	2	5
		SLW	Cut XOUTF pin with internal oscillation circuit.	xxxxxxxx	1 1 0 0 0 0 1 0	C2	1	2	5
		WIT	Internal clock source is stopped.	xxxxxxxx	1 1 0 0 0 0 1 0	C2	1	2	6
		STP	Oscillation is stopped.	xxxxxxxx	0 1 0 0 0 0 1 0	42	1	2	7

Symbol	Means
A	**Accumulater**
Ai	**Bit i of accumulator**
X	**Index register X**
Y	**Index register Y**
M	**Memory**
Mi	**Bit i of memory**
PS	**Processor status register**
S	**Stack pointer**
PC	**Program counter**
PC_L	**Lower byte of program counter**
PC_H	**Higher byte of program counter**
N	**Negative flag**
V	**Overflow flag**
T	**X modified operation mode flag**
B	**Break flag**
D	**Decimal mode flag**
I	**Interrupt disable flag**
Z	**Zero flag**
C	**Carry flag**
#	**Immediate mode**
$	**Hexadecimal**

Symbol	Means
\	**Special page mode**
hh	**Higher byte of address (0~255)**
ll	**Lower byte of address (0~255)**
zz	**Zero Page address (0~255)**
nn	**Data at 0~255**
i	**Data at 0~7**
iii	**Data at 0~7**
<B2>	**Second byte of instruction**
<B3>	**Third byte of instruction**
Rel	**Relative address**
BADRS	**Break address**
←	**Direction of data transfer**
( )	**Contents of register or memory**
+	**Add**
—	**Substract**
∨	**Logical OR**
∧	**Logical AND**
∀	**Logical Exclusive OR**
¯	**Negative**
×	**Stable flag after execution**
o	**Variable flag after execution**

Note 1 : Listed function is when (T) = 0.
When (T) = 1, (M(X)) is entered instead of (A) and the cycle number is increased by 3.
Note 2 : Ditto. The cycle number is increased by 2.
Note 3 : Ditto. The cycle number is increased by 1.
Note 4 : The cycle number is increased by 2 when a branch is occurred.
Note 5 : This instruction is provided for M50740A-XXXSP/FP, 50740ASP, M50741-XXXSP/FP, M50752-XXXSP, M50757-XXXSP and M50758-XXXSP.
Note 6 : This instruction is not provided for M50740A-XXXSP/FP, 50740ASP, M50741-XXXSP/FP, M50752-XXXSP, M50757-XXXSP and M50758-XXXSP.
Note 7 : This instruction is not provided for M50752-XXXSP, M50757-XXXSP and M50758-XXXSP.
Note 8 : This instruction is provided for special type only (for example M37450 family).
Note 9 : The "CYCLE NUMBER" means the cycle number of the internal clock ø.

APPENDIX 3

MELPS 740 Instruction Hexadecimal Code Table

D_7~D_4 \ D_3~D_0	Hexadecimal notation	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0000	0	BRK	ORA IND,X	JSR ZP,IND	BBS 0,A	—	ORA ZP	ASL ZP	BBS 0,ZP	PHP	ORA IMM	ASL A	SEB 0,A	—	ORA ABS	ASL ABS	SEB 0,ZP
0001	1	BPL	ORA IND,Y	CLT	BBC 0,A	—	ORA ZP,X	ASL ZP,X	BBC 0,ZP	CLC	ORA ABS,Y	DEC A	CLB 0,A	—	ORA ABS,X	ASL ABS,X	CLB 0,ZP
0010	2	JSR ABS	AND IND,X	JSR SP	BBS 1,A	BIT ZP	AND ZP	ROL ZP	BBS 1,ZP	PLP	AND IMM	ROL A	SEB 1,A	BIT ABS	AND ABS	ROL ABS	SEB 1,ZP
0011	3	BMI	AND IND,Y	SET	BBC 1,A	—	AND ZP,X	ROL ZP,X	BBC 1,ZP	SEC	AND ABS,Y	INC A	CLB 1,A	LDM ZP	AND ABS,X	ROL ABS,X	CLB 1,ZP
0100	4	RTI	EOR IND,X	STP	BBS 2,A	COM ZP	EOR ZP	LSR ZP	BBS 2,ZP	PHA	EOR IMM	LSR A	SEB 2,A	JMP ABS	EOR ABS	LSR ABS	SEB 2,ZP
0101	5	BVC	EOR IND,Y	—	BBC 2,A	—	EOR ZP,X	LSR ZP,X	BBC 2,ZP	CLI	EOR ABS,Y	—	CLB 2,A	—	EOR ABS,X	LSR ABS,X	CLB 2,ZP
0110	6	RTS	ADC IND,X	MUL ZP,X *1	BBS 3,A	TST ZP	ADC ZP	ROR ZP	BBS 3,ZP	PLA	ADC IMM	ROR A	SEB 3,A	JMP IND	ADC ABS	ROR ABS	SEB 3,ZP
0111	7	BVS	ADC IND,Y	—	BBC 3,A	—	ADC ZP,X	ROR ZP,X	BBC 3,ZP	SEI	ADC ABS,Y	—	CLB 3,A	—	ADC ABS,X	ROR ABS,X	CLB 3,ZP
1000	8	BRA	STA IND,X	RRF ZP	BBS 4,A	STY ZP	STA ZP	STX ZP	BBS 4,ZP	DEY	—	TXA	SEB 4,A	STY ABS	STA ABS	STX ABS	SEB 4,ZP
1001	9	BCC	STA IND,Y	—	BBC 4,A	STY ZP,X	STA ZP,X	STX ZP,X	BBC 4,ZP	TYA	STA ABS,Y	TXS	CLB 4,A	—	STA ABS,X	—	CLB 4,ZP
1010	A	LDY IMM	LDA IND,X	LDX IMM	BBS 5,A	LDY ZP	LDA ZP	LDX ZP	BBS 5,ZP	TAY	LDA IMM	TAX	SEB 5,A	LDY ABS	LDA ABS	LDX ABS	SEB 5,ZP
1011	B	BCS	LDA IND,Y	JMP ZP,IND	BBC 5,A	LDY ZP,X	LDA ZP,X	LDX ZP,Y	BBC 5,ZP	CLV	LDA ABS,Y	TSX	CLB 5,A	LDY ABS,X	LDA ABS,X	LDX ABS,Y	CLB 5,ZP
1100	C	CPY IMM	CMP IND,X	WIT	BBS 6,A	CPY ZP	CMP ZP	DEC ZP	BBS 6,ZP	INY	CMP IMM	DEX	SEB 6,A	CPY ABS	CMP ABS	DEC ABS	SEB 6,ZP
1101	D	BNE	CMP IND,Y	—	BBC 6,A	—	CMP ZP,X	DEC ZP,X	BBC 6,ZP	CLD	CMP ABS,Y	—	CLB 6,A	—	CMP ABS,X	DEC ABS,X	CLB 6,ZP
1110	E	CPX IMM	SBC IND,X	DIV ZP,X *1	BBS 7,A	CPX ZP	SBC ZP	INC ZP	BBS 7,ZP	INX	SBC IMM	NOP	SEB 7,A	CPX ABS	SBC ABS	INC ABS	SEB 7,ZP
1111	F	BEQ	SBC IND,Y	—	BBC 7,A	—	SBC ZP,X	INC ZP,X	BBC 7,ZP	SED	SBC ABS,Y	—	CLB 7,A	—	SBC ABS,X	INC ABS,X	CLB 7,ZP

Note *1: This instruction is provided for special type only (for example M37450 family).

APPENDIX 4

HEX	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	\	l	|
D	CR	GS	–	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

CONTACT ADDRESSES FOR FURTHER INFORMATION

JAPAN

Semiconductor Marketing Division
Mitsubishi Electric Corporation
2-3, Marunouchi 2-chome
Chiyoda-ku, Tokyo 100, Japan
Telex: 24532 MELCO J
Telephone: (03) 218-3473
(03) 218-3499
Facsimile: (03) 214-5570

Overseas Marketing Manager
Kita-Itami Works
4-1, Mizuhara, Itami-shi,
Hyogo-ken 664, Japan
Telex: 526408 KMELCO J
Telephone: (0727) 82-5131
Facsimile: (0727) 72-2329

HONG KONG

MITSUBISHI ELECTRIC (H.K.) LTD.
25 Floor, Leighton Centre,
77, Leighton Road. Causeway Bay.
Hong Kong
Telex: 60800 MELCO HX
Telephone: (5) 773901-3
Facsimile: (5) 895-3104

SINGAPORE

MELCO SALES SINGAPORE PTE. LTD.
230 Upper Bukit Timah Road # 03-01/15
Hock Soon Industrial Complex
Singapore 2158
Telex: RS 20845 MELCO
Telephone: 4695255
Facsimile: 4695347

TAIWAN

MELCO-TAIWAN CO., Ltd.
1st fl., Chung-Ling Bldg.,
363, Sec. 2, Fu-Hsing S Road,
Taipei R.O.C.
Telephone: (02) 735-3030
Facsimile: (02) 735-6771
Telex: 25433 CHURYO "MELCO-TAIWAN"

U.S.A.

NORTHWEST
Mitsubishi Electronics America, Inc.
1050 East Arques Avenue
Sunnyvale, CA 94086
Telephone: (408) 730-5900
Facsimile: (408) 730-4972

SAN DIEGO
Mitsubishi Electronics America, Inc.
11545 West Bernardo Court
Suite 100
San Diego, CA 92128
Telephone: (619) 592-1445
Facsimile: (619) 592-0242

DENVER
Mitsubishi Electronics America, Inc.
4600 South Ulster Street
Metropoint Building, 7th Floor
Denver, CO 80237
Telephone: (303) 740-6775
Facsimile: (303) 694-0613

SOUTHWEST
Mitsubishi Electronics America, Inc.
991 Knox Street
Torrance, CA 90502
Telephone: (213) 515-3993
Facsimile: (213) 217-5781

SOUTH CENTRAL
Mitsubishi Electronics America, Inc.
1501 Luna Road, Suite 124
Carrollton, TX 75006
Telephone: (214) 484-1919
Facsimile: (214) 243-0207

NORTHERN
Mitsubishi Electronics America, Inc.
15612 Highway 7 #243
Minnetonka, MN 55345
Telephone: (612) 938-7779
Facsimile: (612) 938-5125

NORTH CENTRAL
Mitsubishi Electronics America, Inc.
800 N. Bierman Circle
Mt. Prospect, IL 60056
Telephone: (312) 298-9223
Facsimile: (312) 298-0567

NORTHEAST
Mitsubishi Electronics America, Inc.
200 Unicorn Park Drive
Woburn, MA 01801
Telephone: (617) 932-5700
Facsimile: (617) 938-1075

MID-ATLANTIC
Mitsubishi Electronics America, Inc.
800 Cottontail Lane
Somerset, NJ 08873
Telephone: (201) 469-8833
Facsimile: (201) 469-1909

SOUTH ATLANTIC
Mitsubishi Electronics America, Inc.
2500 Gateway Center Blvd., Suite 300
Morrisville. NC 27560
Telephone: (404) 368-4850
Facsimile: (404) 662-5208

SOUTHEAST
Mitsubishi Electronics America, Inc.
Town Executive Center
6100 Glades Road #210
Boca Raton, FL 33433
Telephone: (407) 487-7747
Facsimile: (407) 487-2046

CANADA
Mitsubishi Electronics America, Inc.
6185 Ordan Drive, Unit #110
Mississauga, Ontario, Canada L5T 2E1
Telephone: (416) 670-8711
Facsimile: (416) 670-8715

Mitsubishi Electronics America, Inc.
300 March Road, Suite 302
Kanata, Ontario, Canada K2K 2E2
Telephone: (416) 670-8711
Facsimile: (416) 670-8715

WEST GERMANY

Mitsubishi Electric Europe GmbH
Headquarters:
Gothear Str. 8
4030 Ratingen 1, West Germany
Telex: 8585070 MED D
Telephone: (02102) 4860
Facsimile: (02102) 486-115

Munich Office:
Arabellastraße 31
8000 München 81, West Germany
Telex: 5214820
Telephone: (089) 919006-09
Facsimile: (089) 9101399

FRANCE

Mitsubishi Electric Europe GmbH
55, Avenue de Colmar
92563 Rueil Malmaison Cedex
Telex: 632326
Telephone: 47087871
Facsimile: 47513622

ITALY

Mitsubishi Electric Europe GmbH
Centro Direzionale Colleoni
Palazzo Cassiopea 1
20041 Agrate Brianza I-Milano
Telephone: (039) 636011
Facsimile: (039) 6360120

SWEDEN

Mitsubishi Electric Europe GmbH
Lastbilsvägen 6B
5-19149 Sollentuna, Sweden
Telex: 10877 (meab S)
Telephone: (08) 960468
Facsimile: (08) 966877

U.K.

Mitsubishi Electric (U.K.) Ltd.
Travellers Lane
Hatfield
Herts AL10 8XB, England, U.K.
Telephone: (0044) 7072 76100
Facsimile: (0044) 7072 78692

AUSTRALIA

Mitsubishi Electric Australia Pty. Ltd.
73-75, Epping Road, North Ryde,
P.O. Box 1567, Macquarie Centre,
N.S.W., 2113, Australia
Telex: MESYD AA 26614
Telephone: (02) (888) 5777
Facsimile: (02) (887) 3635

MITSUBISHI SEMICONDUCTORS
MELPS 740 〈SOFTWARE〉 USER'S MANUAL

November. First Edition 1989
Editioned by
Committee of editing of Mitsubishi Semiconductor USER'S MANUAL

Published by
Mitsubishi Electric Corp., Semiconductor Marketing Division

MITSUBISHI SEMICONDUCTORS
SHINGLE-CHIP 8-BIT MICROCOMPUTERS SERIES 740

MITSUBISHI ELECTRIC CORPORATION
HEAD OFFICE: MITSUBISHI DENKI BLDG., MARUNOUCHI, TOKYO 100. TELEX: J24532 CABLE: MELCO TOKYO

H-E0204-A(ROD)

Printed in Japan