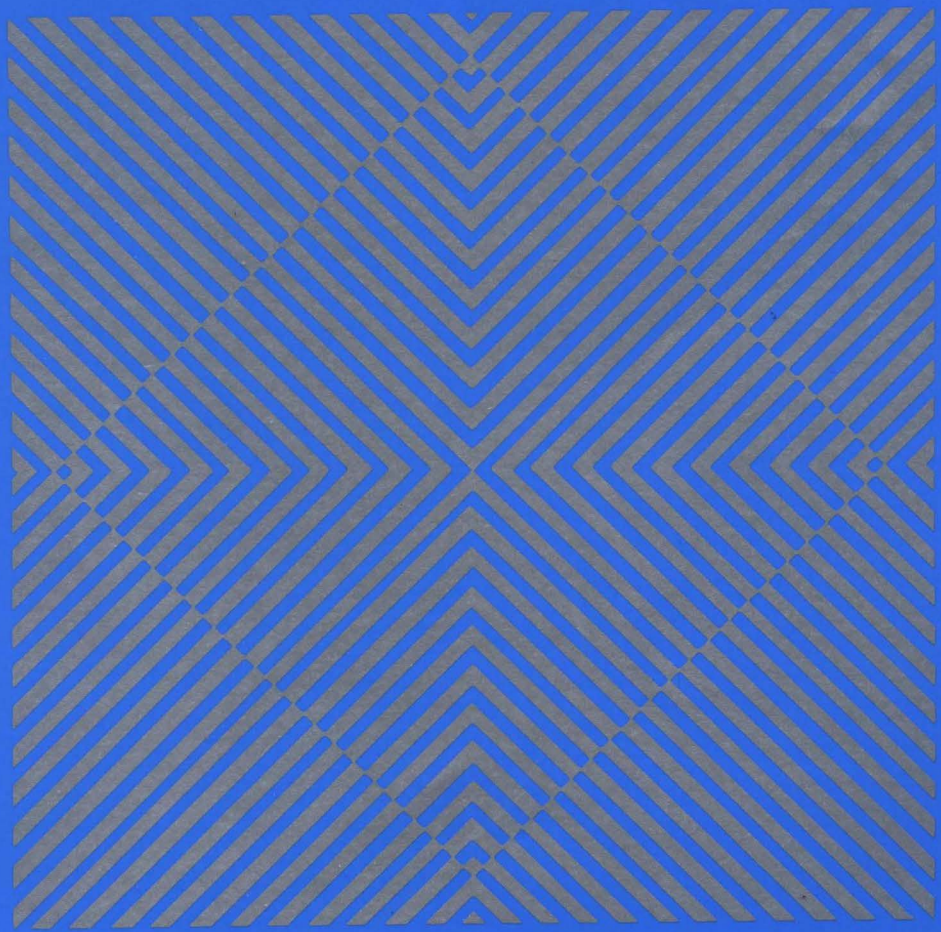
MITSUBISHI SEMICONDUCTORS

USER'S MANUAL

MELPS 7700

SOFTWARE MANUAL



MELPS 7700 <SOFT WARE>

MITSUBISHI ELECTRIC

Foreword

This manual has been prepared to enable the users of the **Series MELPS 7700** CMOS 16-bit microcomputers to better understand the instruction set and the features so that they can utilize the capabilities of the microcomputers to the fullest. This manual presents detailed descriptions of the instructions and addressing modes available for the **Series MELPS 7700** microcomputers.

For the hardware descriptions of the **Series MELPS 7700** microcomputers and descriptions of various development support tools (e.g., assembler, debugger), please refer to the user's manuals and operating guidebooks for the respective hardware and software products.

Contents

1. Introduction of Series MELPS 7700 Software

The software for the **Series MELPS 7700** 16-bit CMOS microcomputers was developed by making is numerous enhancements on the software for the **Series MELPS 740** 8-bit microcomputer which are based on Mitsubishi Electric Corporation's proprietary designs. The enhancements include support of word (16-bit) operations and linear accessing of up to 16M bytes of memory space.

The new software's compact and easy to use instruction set and the support of powerful addressing modes will significantly increase

:The **Series MELPS 7700** microcomputers offer the following features

- Upward compatibility for the **Series MELPS 740**.
- Powerful addressing modes and fast and compact instruction set.
- Direct page mapping function and memory oriented software system by direct paging.
- Byte and word operations can be selected at will by the m flag.
- The usual 64K bytes program memory boundary can be ignored for the practical purposes,and programs can be written to utilize the full 16M bytes of memory space. For data memory, linear as well as bank memory accessing are supported.
- Bit manipulation instructions and bit test and branch instructions can be used for memory and I/O accessing of the entire 16M bytes space.
- Block transfer instruction capable of handling blocks of up to 64K bytes each.
- Improved stack accessing capability.
- Decimal arithmetic instruction execution requiring no software compensation.

The performance of the systems based on the **Series MELPS 7700** microcomputers, whether used as advanced 8-bit microcomputer or next-generation 16-bit one.

2. Register Configuration

The central processing unit (CPU) of each **Series MELPS 7700** microcomputer has 10 internal registers (See Fig.2.1). Each of these registers is described below

2.1 Accumulator (Acc)

(1) Accumulator A (A)

The accumulator A is the main register of the microcomputer, and data processing such as arithmetic calculations, data transfer and input/output operations are executed via this accumulator. It consists of 16-bit register, but it can be used as an 8-bit register by setting the data length selection flag m in the processor status register PS. The flag m is described in detail in a later section. The flag m value of "0" specifies 16-bit data length, and "1" specifies 8-bit data length. When operating under 8-bit data length setting, only the lower 8 bits of the accumulator A are used and the upper 8 bits do not change.

(2) Accumulator B (B)

The accumulator B is a 16-bit register whose function is equivalent to that of the accumulator A. The **Series MELPS 7700** instructions can use the accumulator B instead of the accumulator A. Note, however, that use of the accumulator B requires more instruction bytes and execution cycles than when using the accumulator A.

2.2 Index Register X (X)

The index register X is a 16-bit register, but it can be used as an 8-bit register by setting the index register length selection flag x in the processor status register PS. The flag x is described in detail in a later section. The flag x value of "0" specifies 16-bit index register length, and "1" specifies 8-bit index register length. When operating under 8-bit index register length setting, only the lower 8 bits of the index register X are used and the upper 8 bits do not change.

In an addressing mode in which the index register X is used as the index register, the address obtained by adding the contents of this register is accessed. For the block transfer instructions, MVP and MVN, the contents of the index register X become the lower 16 bits of the transfer-from address and the byte-3 of the instruction becomes the upper 8 bits.

2.3 Index Register Y (Y)

The index register Y is a 16-bit register whose function is equivalent to that of the index register X. As in the case of the index register X, the index register length selection flag x can be used to use only the lower 8 bits of the index register Y. For the block transfer instructions, MVP and MVN, the contents of the index register Y become the lower 16 bits of the transfer-to address and the byte-2 of the instruction become the upper 8 bits.

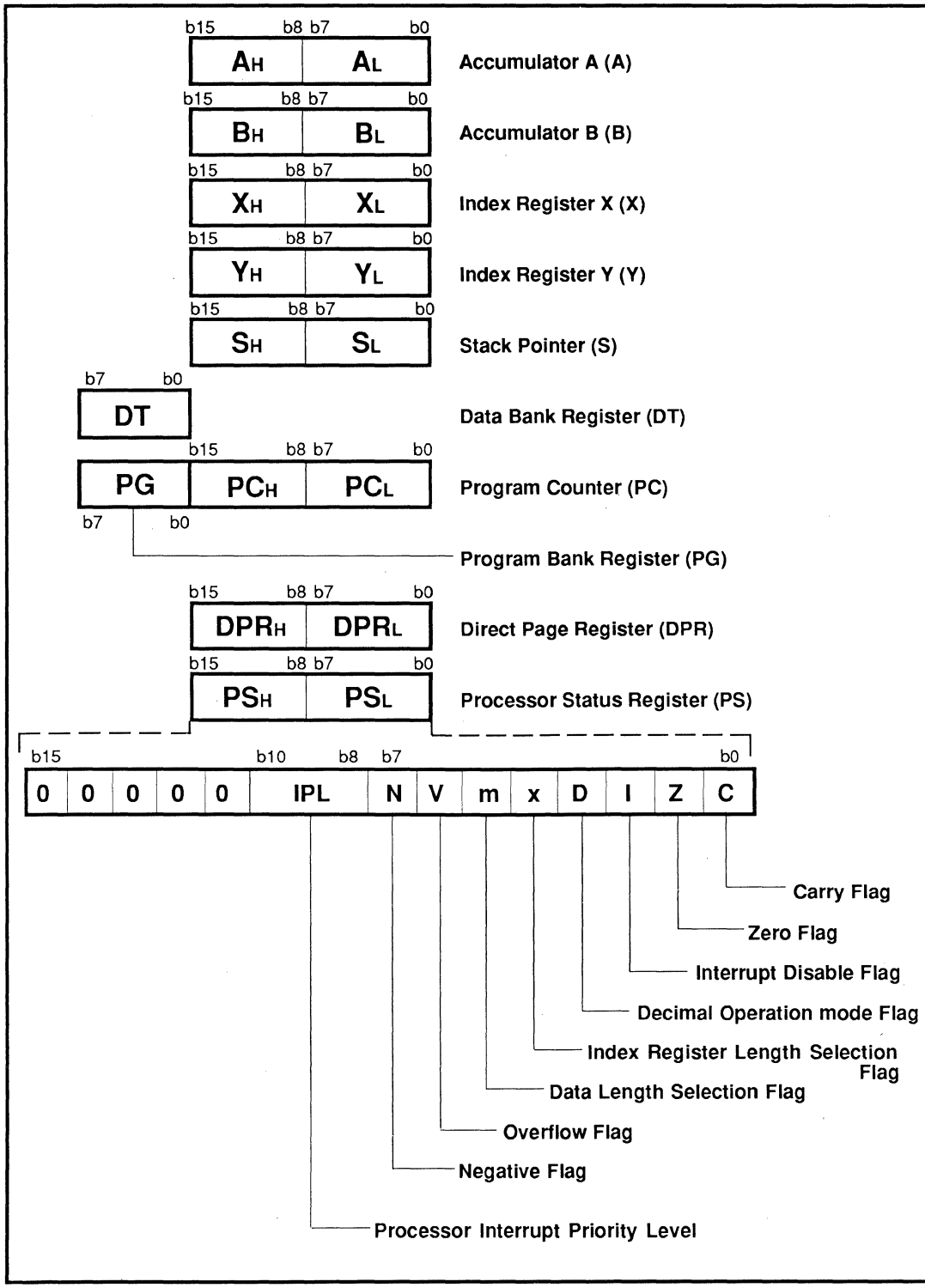


Fig. 2.1 CPU Register Model

2.4 Stack Pointer (S)

The stack pointer (S) is a 16-bit register, and it is used when calling a subroutine, at the time of interrupt processing and when using one of the stack addressing modes. The contents of the stack pointer specifies the address (stack area) where the memory (RAM) registers that must be saved are to be stored.

When an interrupt is received, the contents of the program bank register are saved at the address specified by the stack pointer's value, and the stack pointer's value is decremented by 1. Similarly, the contents of the program counter and the processor status register are saved in the order of lower bytes first (PC_H, PC_L, PS_H, PS_L). Thus, the value of the stack pointer after an interrupt has been accepted will be 5 less than the value before the interrupt acceptance. When the interrupt processing is completed and the control is returned to the original routine, the registers that had been saved to the stack area are restored in the reverse order of the saving operation, and the stack pointer's value is restored to that before the interrupt was accepted. Similar operations are executed when a subroutine is called, except that the processor status register (and the program bank register for some addressing modes) is not saved.

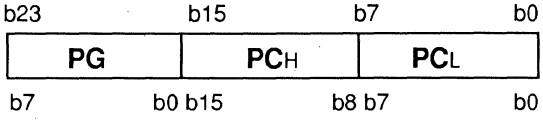
The registers other than those indicated above are not saved when an interrupt is invoked or when a subroutine is called, so that provisions must be made in the application programs to save the registers if necessary. Also note that the stack pointer must be initialized after the microcomputer is reset, because its content is indeterminable after reset operation. Normally, the highest address of the internal RAM is set in the stack pointer. The contents of the stack area will change by nesting of subroutines and acceptance of multiple interrupts, so that the subroutine nesting levels must be chosen carefully so as not to destroy the integrity of RAM data.

2.5 Program Counter (PC)

The program counter (PC) is a 16-bit register that contains the lower 16-bit values of the 24-bit program memory address of the instruction to be executed next.

2.6 Program Bank Register (PG)

The program bank register (PG) is an 8-bit register that contains the upper 8-bit (bank) value of the 24-bit program memory address of the instruction to be executed next. When a carry is generated by incrementing of the program counter's content or when a carry or borrow is generated by addition or subtraction of an offset value to the program counter's content by execution of a branching instruction, for example, the program bank register's content is automatically incremented or decremented by 1 so that the bank boundary needs not be considered for application programming.



2.7 Data Bank Register (DT)

The data bank register (DT) is an 8-bit register. Its contents are interpreted as the upper 8 bits (bank) of a 24-bit memory address under certain addressing modes.

2.8 Direct Page Register (DPR)

The direct page register (DPR) is a 16-bit register, which allows specification of a 256 byte space called a direct page in bank-0. This area can be accessed by 2 bytes in the direct page addressing mode. The contents of the direct page register specify the least-significant (base) address of the direct page area. A value in the range of 0_{16}-$FFFF_{16}$ may be set in the direct page register. When a value of or higher than $FF01_{16}$ is set in the direct page register, the direct page area will cross over the bank-0 and bank-1 boundary. Normally, the lower 8-bit value of the direct page register is set to 00_{16} since that reduces the number of cycles required for address generation.

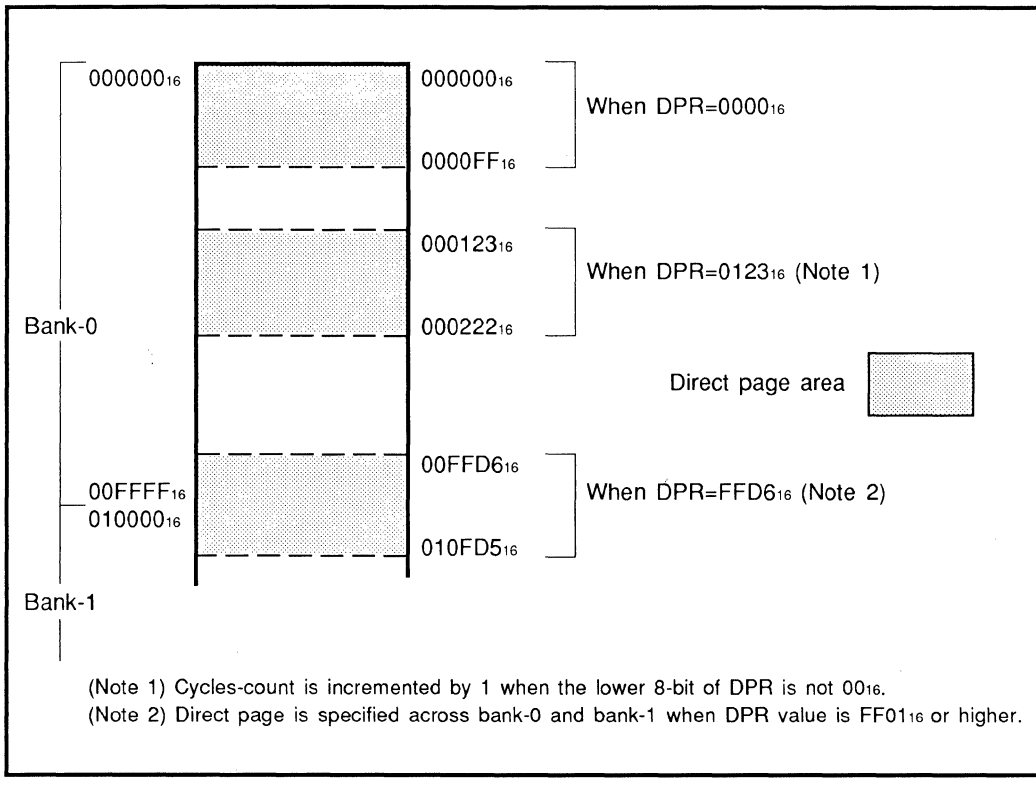
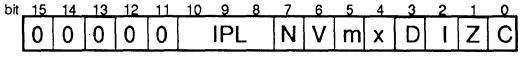


Fig. 2.2 Setting Direct Page by Direct Page Register

2.9 Processor Status Register (PS)

The processor status register (PS) is an 11-bit register, and it consists of flags that specify the status immediately after operation and bits that set the processor interrupt priority level. The C, Z, V and N flags enable execution of branching instructions depending on the flag values. Each bit of the processor status register is explained below.



Processor Status Register (PS)

(Note) Bits 11-15 are fixed at 0.

[Bit-0] Carry Flag (C)

This bit is the carry flag which holds the carry or borrow from the arithmetic logic unit (ALU) after arithmetic operation. It is also affected by the shift and rotate instructions. This flag can be directly set by the SECand SEP, and cleared by CLC and CLP instructions.

[Bit-1] Zero Flag (Z)

This bit is set 1 when the arithmetic operation or data transfer result is 0, and it is set 0 when such result is not "0". This flag is invalid for addition (ADC) instruction in the decimal-operation mode. This flag can be directly set by SEP and cleared by CLP instructions.

[Bit-2] Interrupt Disable Flag (I)

This is the flag that is used to disable all interrupts (except the interrupts by the watchdog timer, BRK instruction and division by zero). When this flag is "1", interrupts are disabled. This flag is set to "1" automatically when an interrupt is accepted, inhibiting multiple interrupt acceptance. This flag can be set using the SEI and SEP, and cleared using the CLI and CLP instructions.

[Bit-3] Decimal Operation Mode Flag (D)

This flag is used to determine whether to execute addition and subtraction in the binary-mode or in the decimal-mode. "0" specifies the ordinary binary mode. When this flag is set to "1", addition/subtraction is executed with 1 word as a 2- or 4-digit decimal value (2- or 4-digit selection is made by the data length selection flag m). Decimal alignment is performed automatically.

Note that decimal-mode can be used only by the ADC and SBC instructions.

This flag can be set by the SEP and cleared by the CLP instructions.

Because this flag directly affects arithmetic operation, it must be initialized whenever the microcomputer is reset.

[Bit-4] Index Register Length Selection Flag (x)

This flag specifies whether to use the index register X or Y in the 16-bit index register length or in the 8-bit index register length. "0" specifies the 16-bit length mode, and "1" specifies the 8-bit length mode. This flag can be set by the SEP, and cleared by the CLP instructions.

[Bit-5] Data Length Selection Flag (m)

This flag specifies whether to use the 16-bit data length or the 8-bit data length. "0" specifies 16-bit, and "1" specifies 8-bit data length. This flag can be set by the SEM and SEP, and cleared by the CLM and CLP instructions.

[Bit-6] Overflow Flag (V)

The overflow flag has a meaning when adding or subtracting 1 word as a signed binary number. This flag is set 1 when the flag m is set to "0" and the result of addition or subtraction is outside the range -32768~+32767, and it is set 0 otherwise. When the flag m is set to "1", this flag is set 1 if the result of addition or subtraction is outside the range -128~+127 and set 0otherwise. This flag can be directly set by the SEP, and cleared by the CLV and CLP instructions. This flag is meaningless in the decimal operation mode.

[Bit-7] Negative flag (N)

The negative flag (N) is set 1 when the result of data transfer is negative (bit-15 of data is "1" when the flag m is "0", or bit-7 of data is "1" when the flag m is "1"), and it is set 0 otherwise. This flag can be directly set by the SEP, and cleared by the CLP instructions. This flag is meaningless in the decimal operation mode.

[Bit-8~Bit-10] Processor interrupt priority level (IPL_0~IPL_2)

The processor interrupt priority level (IPL) consists of 3 bits, and these 3 bits enable determination of 8 processor interrupt priority levels (level-0 ~ level-7). An interrupt is allowed only when its interrupt priority level is higher than the IPL value. When an interrupt is generated, IPL is saved to the stack area, and the priority level of the allowed interrupt is set in IPL.

There is no instruction that can directly set or clear IPL_0~IPL_2. Therefore, in order to alter the IPL contents, the desired value must be first stored in the stack and then the processor status register contents altered using the PUL or PLP instruction.

3. Addressing Modes

3.1 Addressing Mode

When executing an instruction, the address of the memory location from which the data required for arithmetic operation is to be retrieved or to which the result of arithmetic operation is to be stored must be specified in advance. Address specification is also necessary when the control is to jump to a certain memory address during program execution. Addressing refers to the method of specifying the memory address.

The **Series MELPS 7700** microcomputers support 28 different addressing modes, offering extremely versatile and powerful memory accessing capability.

3.2 Explanation of Addressing Modes

Each of the 28 addressing modes is explained on the pages indicated below:

Implied

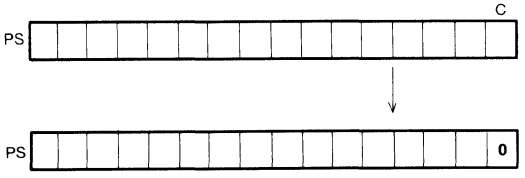
Mode : Implied addressing mode

Function : The single-instruction inherently address an internal register.

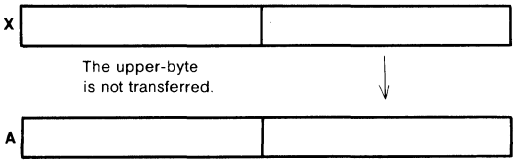
Instruction :

BRK,	CLC,	CLI,	CLM,	CLV,	DEX,	DEY,	INX,	INY,	NOP,
RTI,	RTL,	RTS,	SEC,	SEI,	SEM,	STP,	TAD,	TAS,	TAX,
TAY,	TBD,	TBS,	TBX,	TBY,	TDA,	TDB,	TSA,	TSB,	TSX,
TXA,	TXB,	TXS,	TXY,	TYA,	TYB,	TYX,	WIT,	XAB	

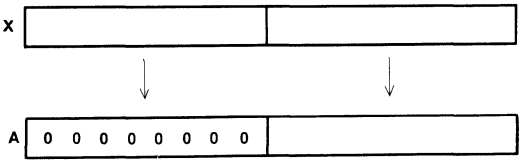
ex. : Mnemonic **CLC** Machine Code 18_{16}



ex. : Mnemonic **TXA** **(m = 1, x = 0)** Machine Code $8A_{16}$



ex. : Mnemonic **TXA** **(m = 0, x = 1)** Machine Code $8A_{16}$



(Note) When the data length differ between the transfer-from and transfer-to locations, data is transferred at the data length for the transfer-to location. If, however, the index register is specified as the transfer-to location and the x flag is set to 1, 0016 is sent as the upper byte value.

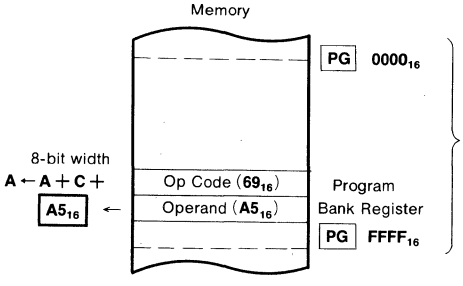
Immediate

Mode : Immediate addressing mode

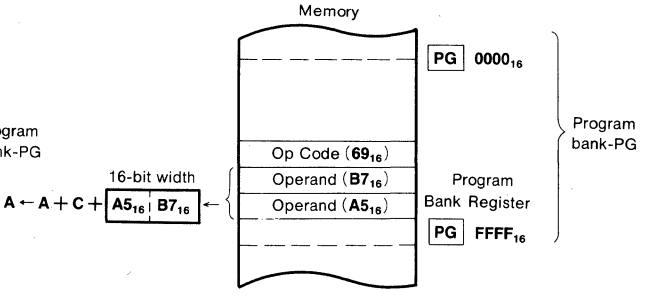
Function : A portion of the instruction is the actual data. Such instruction code may cross over the bank boundary.

Instruction : ADC, AND, CLP, CMP, CPX, CPY, DIV, EOR, LDA, LDT, LDX, LDY, MPY, ORA, RLA, SBC, SEP

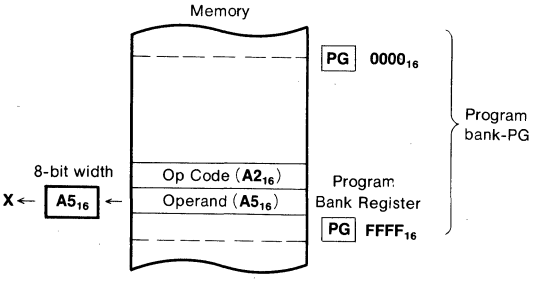
ex. : Mnemonic **ADC A,#0A5H** (**m = 1**) Machine Code **69_{16} $A5_{16}$**



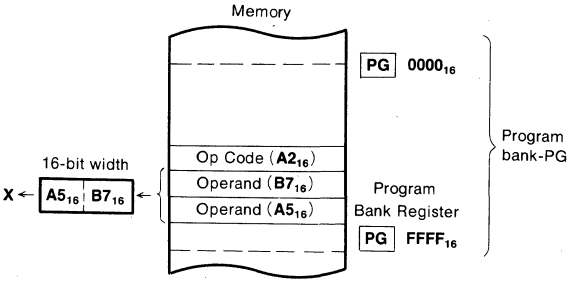
ex. : Mnemonic **ADC A,#0A5B7H** (**m = 0**) Machine Code **69_{16} $B7_{16}$ $A5_{16}$**



ex. : Mnemonic **LDX #0A5H** (**x = 1**) Machine Code **$A2_{16}$ $A5_{16}$**



ex. : Mnemonic **LDX #0A5B7H** (**x = 0**) Machine Code **$A2_{16}$ $B7_{16}$ $A5_{16}$**



Accumulator

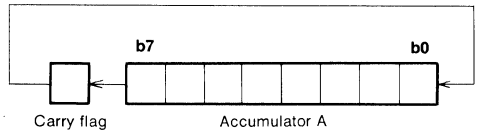
Mode : Accumulator addressing mode

Function : The contents of accumulator are the actual data.

Instruction : ASL, DEC, INC, LSR, ROL, ROR

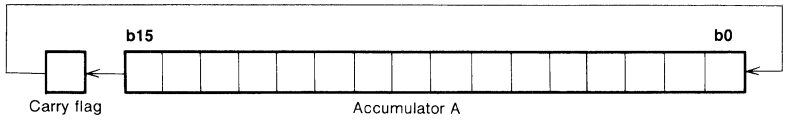
ex. : Mnemonic
ROL A
(m = 1)

Machine Code
$2A_{16}$



ex. : Mnemonic
ROL A
(m = 0)

Machine Code
$2A_{16}$



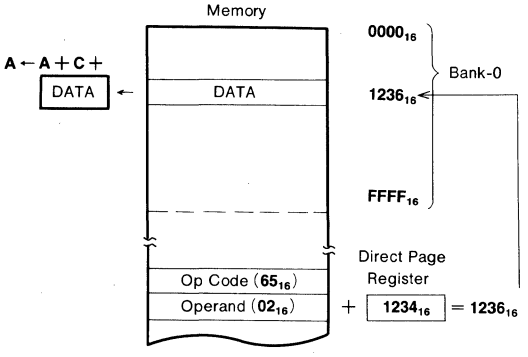
Direct

Mode : Direct addressing mode

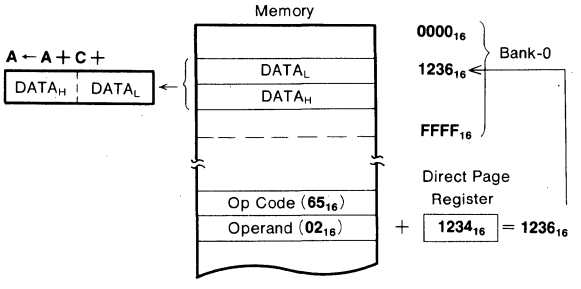
Function : The contents of the bank-0 memory location specified by the result of adding the second byte of the instruction to the contents of the direct page register become the actual data. If, however, addition of the instruction's second byte to the direct page register's contents result in a value that exceeds the bank-0 range, the specified location will be in bank-1.

Instruction : ADC, AND, ASL, CMP, CPX, CPY, DEC, DIV, EOR, INC, LDA, LDM, LDX, LDY, LSR, MPY, ORA, ROL, ROR, SBC, STA, STX, STY

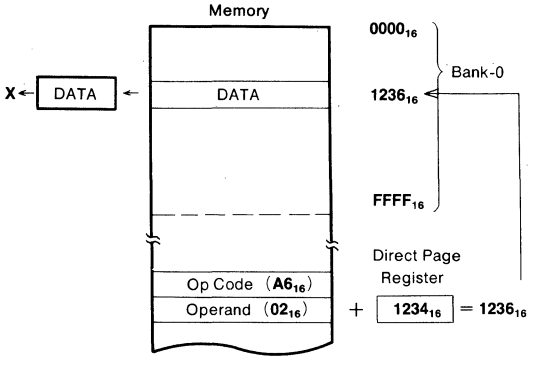
ex. : Mnemonic **ADC A,02H** (**m = 1**) Machine Code **65_{16} 02_{16}**



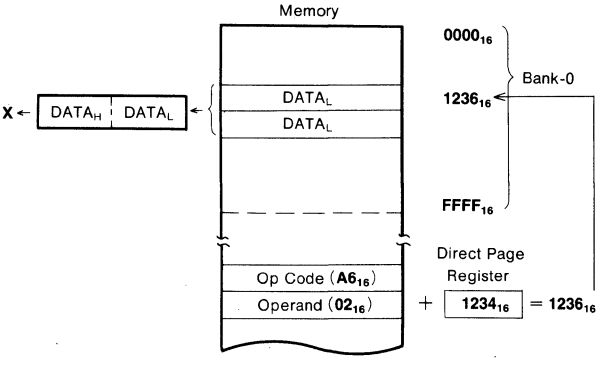
ex. : Mnemonic **ADC A,02H** (**m = 0**) Machine Code **65_{16} 02_{16}**



ex. : Mnemonic **LDX 02H** (**x = 1**) Machine Code **$A6_{16}$ 02_{16}**



ex. : Mnemonic **LDX 02H** (**x = 0**) Machine Code **$A6_{16}$ 02_{16}**



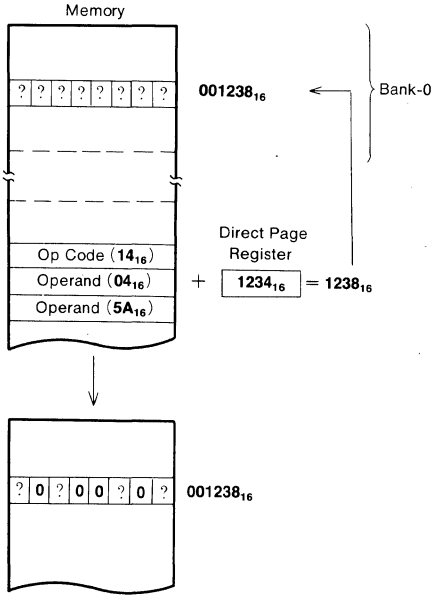
Direct Bit

Mode : Direct bit addressing mode

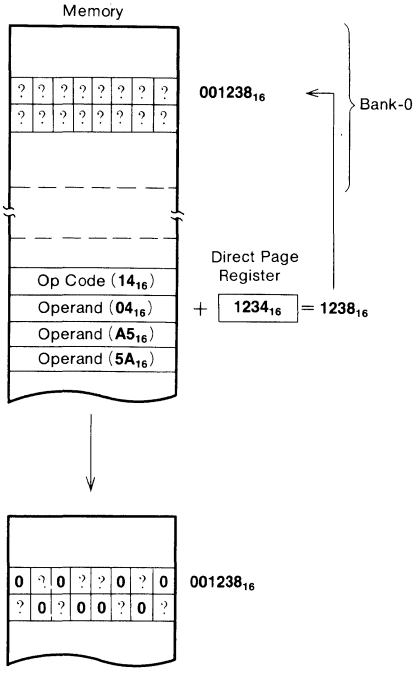
Function : Specifies the bank-0 memory location by the value obtained by adding the instruction's second byte to the direct page register's contents, and specifies the positions of multiple bits in the memory location by the bit pattern in the third and fourth bytes of the instruction (third byte only when the m flag is set to 1). If, however, addition of the instruction's second byte to the direct page register's contents result in a value that exceeds the bank-0 range, the specified location will be in bank-1.

Instruction : CLB, SEB

ex. : Mnemonic **CLB #5AH, 04H** (**m = 1**) Machine Code **14_{16} 04_{16} $5A_{16}$**



ex. : Mnemonic **CLB #5AA5H, 04H** (**m = 0**) Machine Code **14_{16} 04_{16} $A5_{16}$ $5A_{16}$**



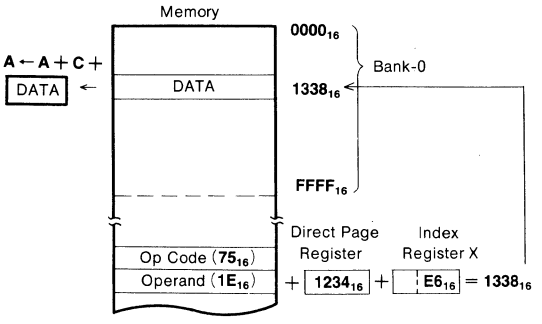
Direct Indexed X

Mode : Direct indexed X addressing mode

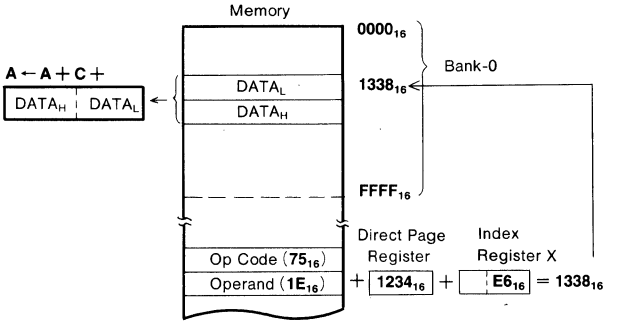
Function : The contents of the bank-0 memory location specified by the result of adding the second byte of the instruction, the contents of the direct page register and the contents of the index register X become the actual data. If, however, addition of the instruction's second byte, the direct page register's contents and the index register X's contents results in a value that exceeds the bank-0 or bank-1 range, the specified location will be in bank-1 or bank-2.

Instruction : ADC, AND, ASL, CMP, DEC, DIV, EOR, INC, LDA, LDM, LDY, LSR, MPY, ORA, ROL, ROR, SBC, STA, STY

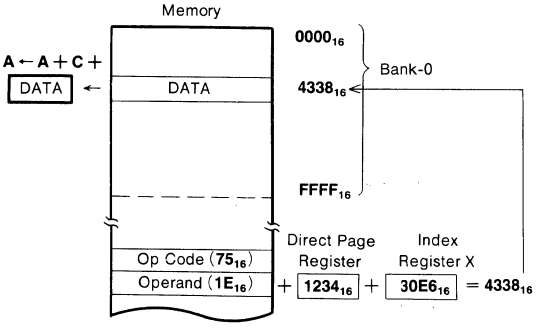
ex. : Mnemonic **ADC A,1EH,X** (**m = 1, x = 1**) Machine Code 75_{16} $1E_{16}$



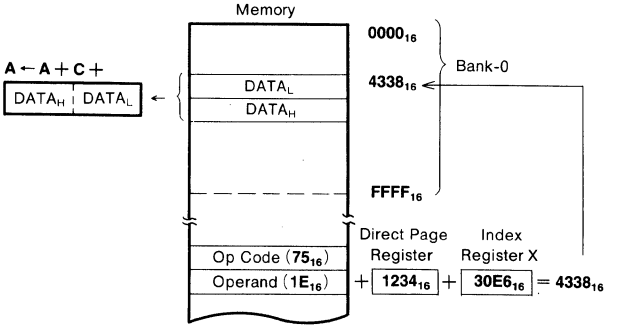
ex. : Mnemonic **ADC A,1EH,X** (**m = 0, x = 1**) Machine Code 75_{16} $1E_{16}$



ex. : Mnemonic **ADC A,1EH,X** (**m = 1, x = 0**) Machine Code 75_{16} $1E_{16}$



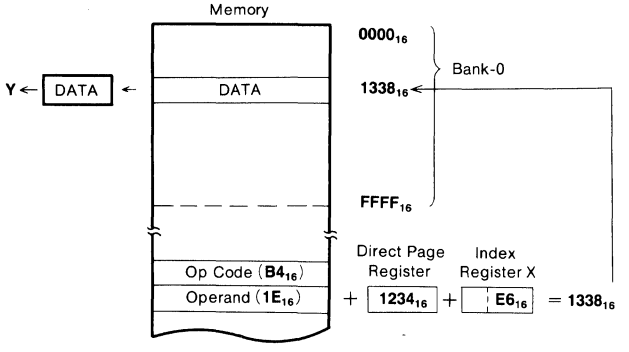
ex. : Mnemonic **ADC A,1EH,X** (**m = 0, x = 0**) Machine Code 75_{16} $1E_{16}$



Direct Indexed X

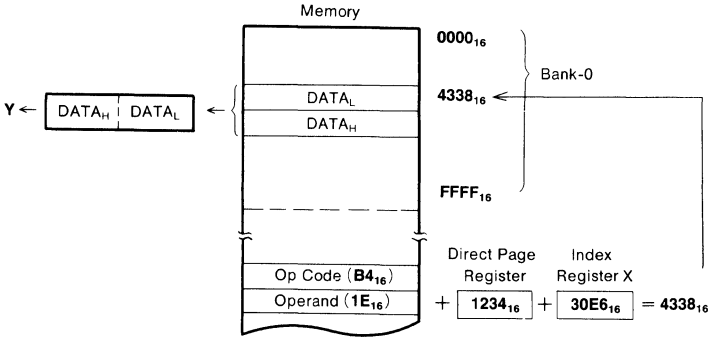
ex. : Mnemonic **LDY 1EH,X** (**x = 1**)

Machine Code **$B4_{16}$ $1E_{16}$**



ex. : Mnemonic **LDY 1EH,X** (**x = 0**)

Machine Code **$B4_{16}$ $1E_{16}$**



Direct Indexed Y

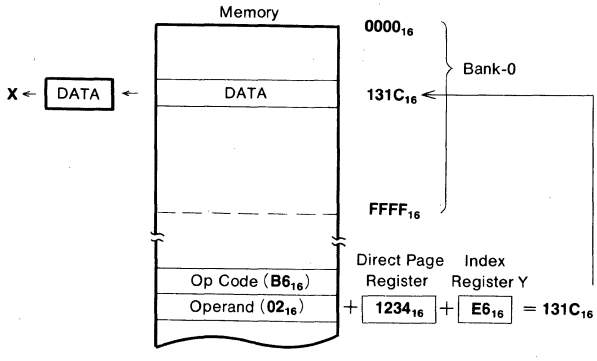
Mode : Direct indexed Y addressing mode

Function : The contents of the bank-0 memory location specified by the result of adding the second byte of the instruction, the contents of the direct page register and the contents of the index register Y become the actual data. If, however, addition of the instruction's second byte, the direct page register's contents and the index register Y's contents results in a value that exceeds the bank-0 or bank-1 range, the specified location will be in bank-1 or bank-2.

Instruction : LDX, STX

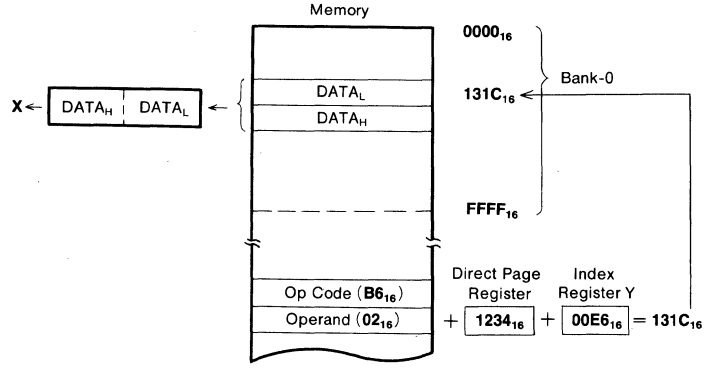
ex. : Mnemonic
LDX 02H,Y
(x = 1)

Machine Code
$B6_{16}$ 02_{16}



ex. : Mnemonic
LDX 02H,Y
(x = 0)

Machine Code
$B6_{16}$ 02_{16}



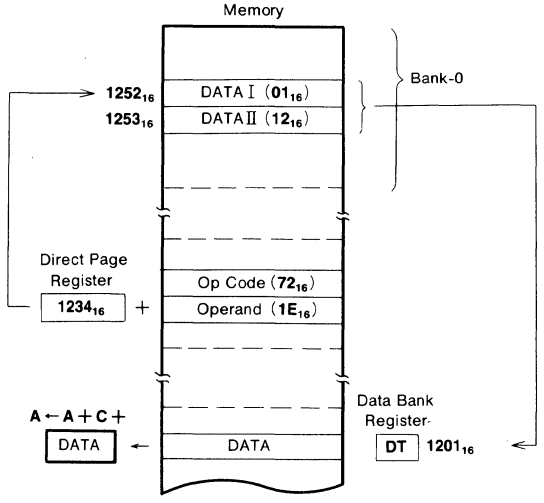
Direct Indirect

Mode : Direct indirect addressing mode

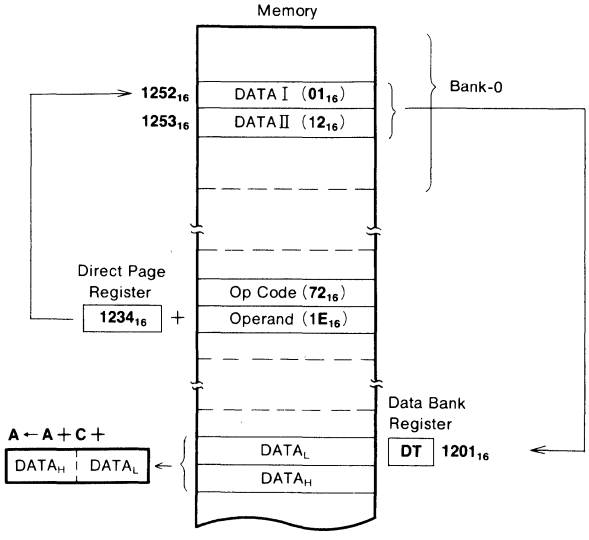
Function : The value obtained by adding the instruction's second byte to the contents of the direct page register specifies 2 adjacent bytes in memory bank-0, and the contents of these bytes in memory bank-DT (DT is contents of data bank register) become the actual data. If, however, the value obtained by adding the instruction's second byte and the direct page register's contents exceeds the bank-0 range, the specified location will be in bank-1.

Instruction : ADC, AND, CMP, DIV, EOR, LDA, MPY, ORA, SBC, STA

ex. : Mnemonic **ADC A,(1EH)** **(m = 1)** Machine Code **72_{16} $1E_{16}$**



ex. : Mnemonic **ADC A,(1EH)** **(m = 0)** Machine Code **72_{16} $1E_{16}$**



Direct Indexed X Indirect

Mode : Direct indexed X indirect addressing mode

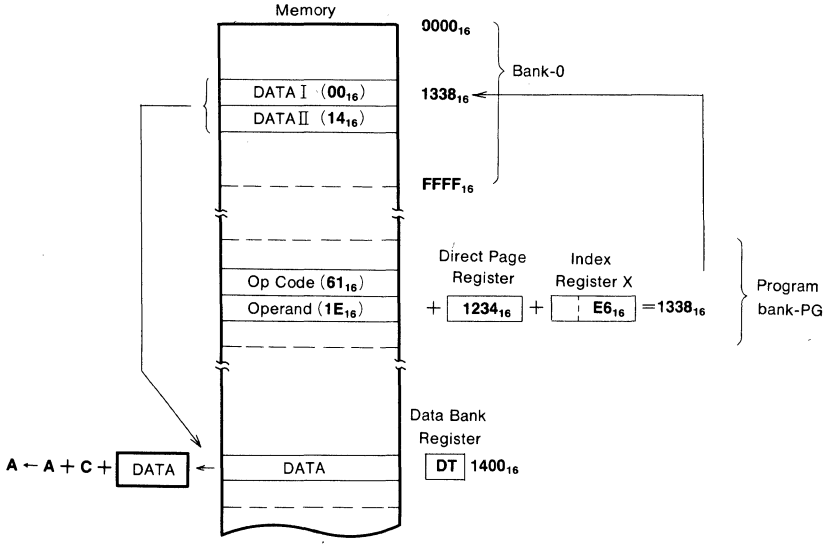
Function : The value obtained by adding the instruction's second byte, the contents of the direct page register and the contents of the index register X specifies 2 adjacent bytes in memory bank-0, and the contents of these bytes in memory bank-0, and the contents of these bytes in memory bank-DT (DT is contents of data bank register) become the actual data. If, however, the value obtained by adding the instruction's second byte, the direct page register's contents and the index register X's contents exceeds the bank-0 or bank-1 range, the specified location will be in bank-1 or bank-2.

Instruction : ADC, AND, CMP, DIV, EOR, LDA, MPY, ORA, SBC, STA

ex. : Mnemonic — Machine Code

ADC A, (1EH, X) — **61_{16} $1E_{16}$**

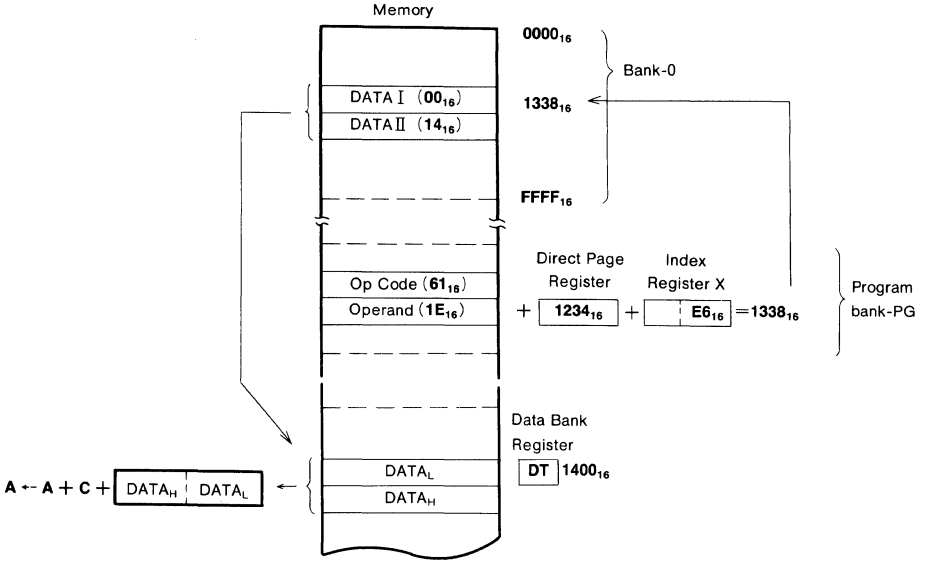
(m = 1, x = 1)



Direct Indexed X Indirect

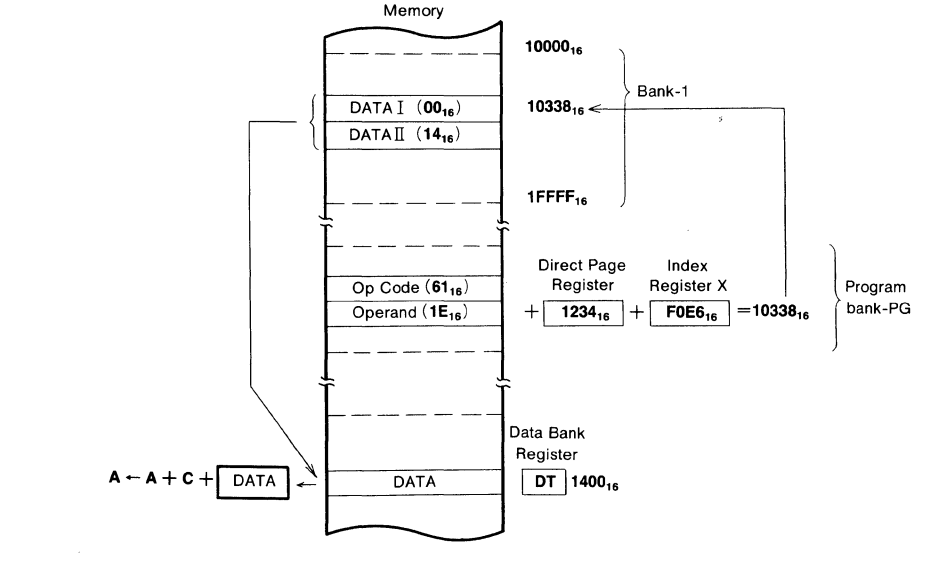
ex. : Mnemonic **ADC A, (1EH, X)** (**m = 0, x = 1**)

Machine Code **61_{16} $1E_{16}$**



ex. : Mnemonic **ADC A, (1EH, X)** (**m = 1, x = 0**)

Machine Code **61_{16} $1E_{16}$**

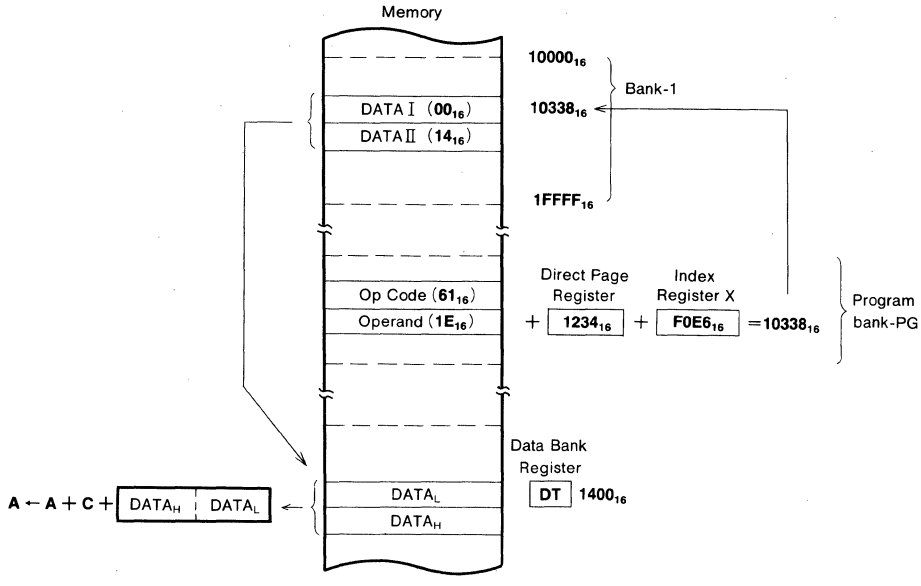


Direct Indexed X Indirect

ex. : Mnemonic — Machine Code

ADC A, (1EH, X) — 61_{16} $1E_{16}$

(m = 0, x = 0)



Direct Indirect Indexed Y

Mode : Direct indirect indexed Y addressing mode

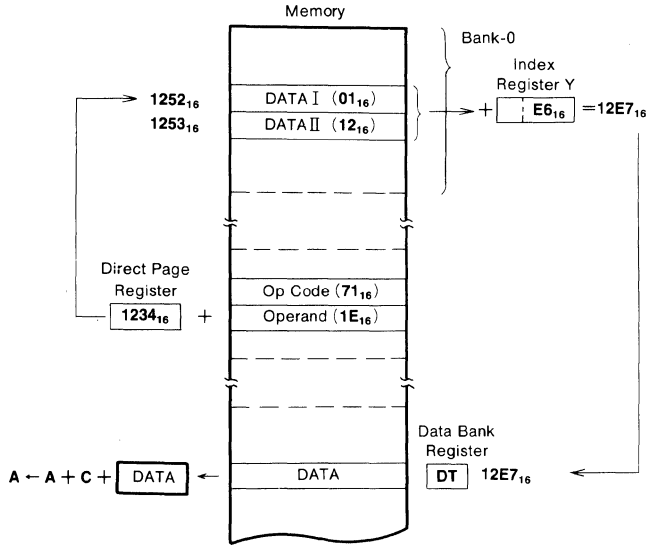
Function : The value obtained by adding the instruction's second byte and the contents of the direct page register specifies 2 adjacent bytes in memory bank-0.
The value obtained by adding the contents of these bytes and the contents of the index register Y specifies address of the actual data in memory bank-DT (DT is contents of data bank register). If, however, the value obtained by adding the contents of the instruction's second byte and the direct page register exceeds the bank-0 range, the specified location will be in bank-1. Also, if addition of the contents of memory and index register Y generate a carry, the bank number will be 1 larger than the contents of the data bank register.

Instruction : ADC, AND, CMP, DIV, EOR, LDA, MPY, ORA, SBC, STA

ex. : Mnemonic — Machine Code

ADC A, (1EH),Y — 71_{16} $1E_{16}$

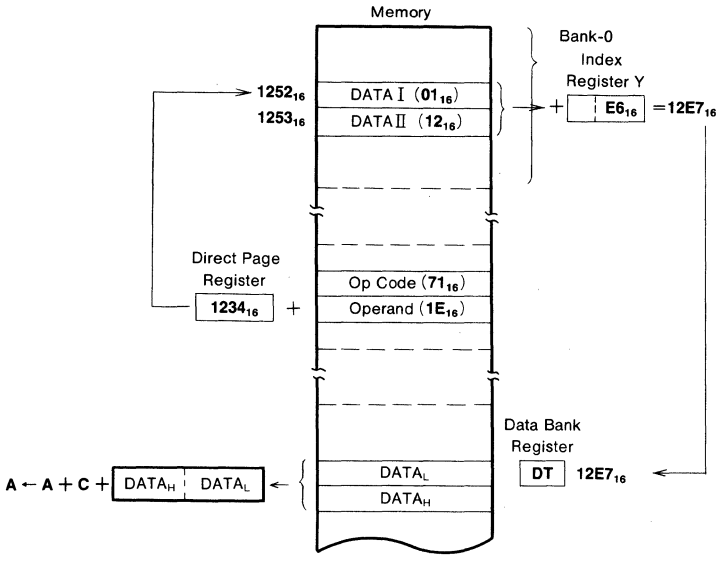
(m = 1, x = 1)



Direct Indirect Indexed Y

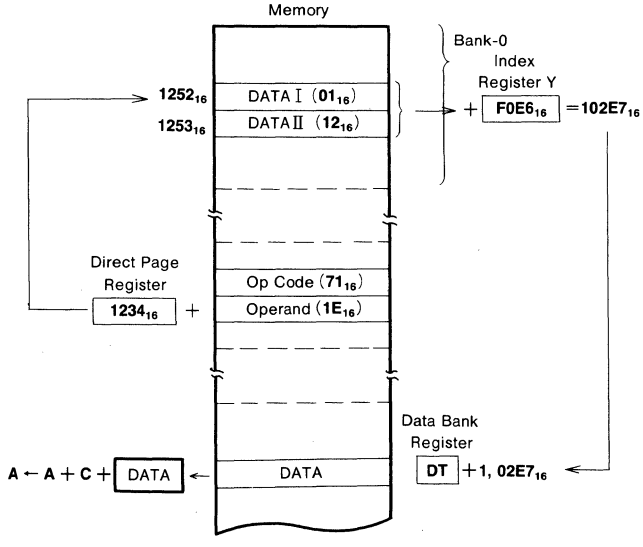
ex. : Mnemonic
ADC A, (1EH), Y
(m = 0, x = 1)

Machine Code
71_{16} $1E_{16}$



ex. : Mnemonic
ADC A, (1EH), Y
(m = 1, x = 0)

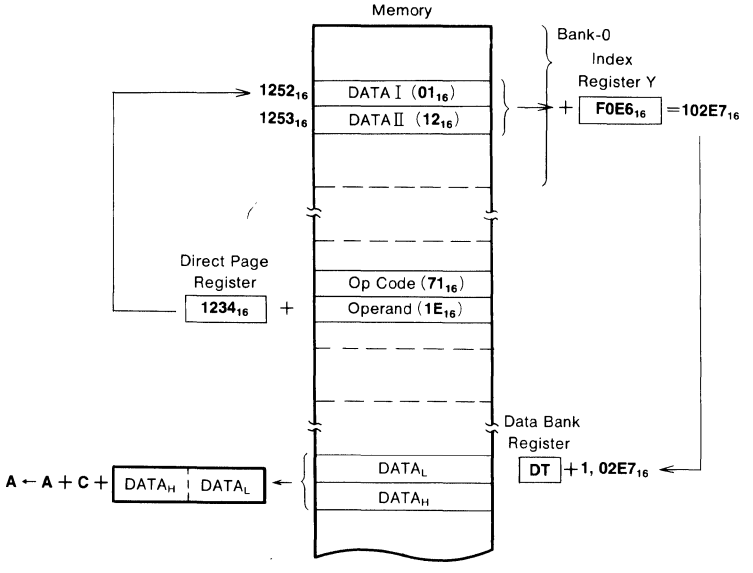
Machine Code
71_{16} $1E_{16}$



Direct Indirect Indexed Y

ex. : Mnemonic
ADC A, (1EH), Y
(m = 0, x = 0)

Machine Code
71_{16} $1E_{16}$



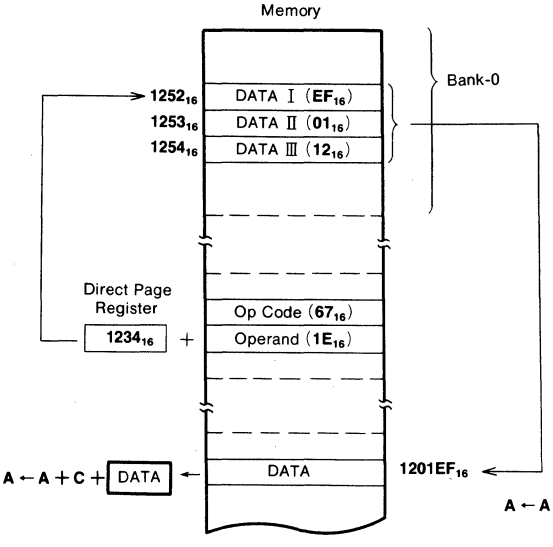
Direct Indirect Long

Mode : Direct indirect long addressing mode

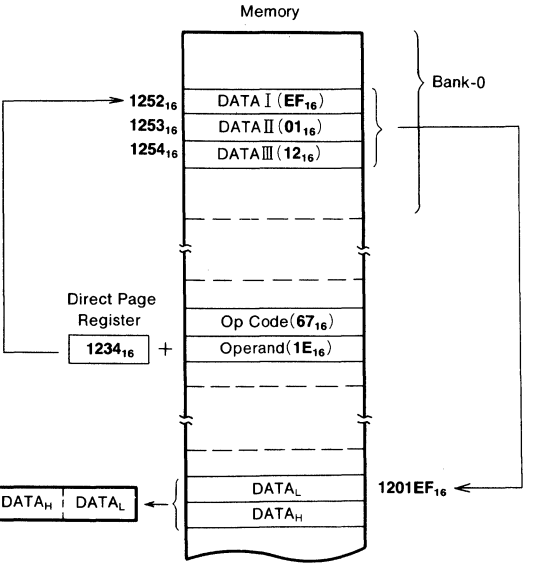
Function : The value obtained by adding the instruction's second byte and the contents of the direct page register specifies 3 adjacent bytes in memory bank-0, and the contents of these bytes specify the address of the memory location that contains the actual data. If, however, the value obtained by adding the contents of the instruction's second byte and the direct page register exceeds the bank-0 range, the specified location will be in bank-1. The 3 adjacent bytes memory location may be spread over two different banks.

Instruction : ADC, AND, CMP, DIV, EOR, LDA, MPY, ORA, SBC, STA

ex. : Mnemonic **ADCL A, (1EH) (m=1)** Machine Code 67_{16} $1E_{16}$



ex. : Mnemonic **ADCL A, (1EH) (m=0)** Machine Code 67_{16} $1E_{16}$



Direct Indirect Long Indexed Y

Mode : Direct indirect long indexed Y addressing mode

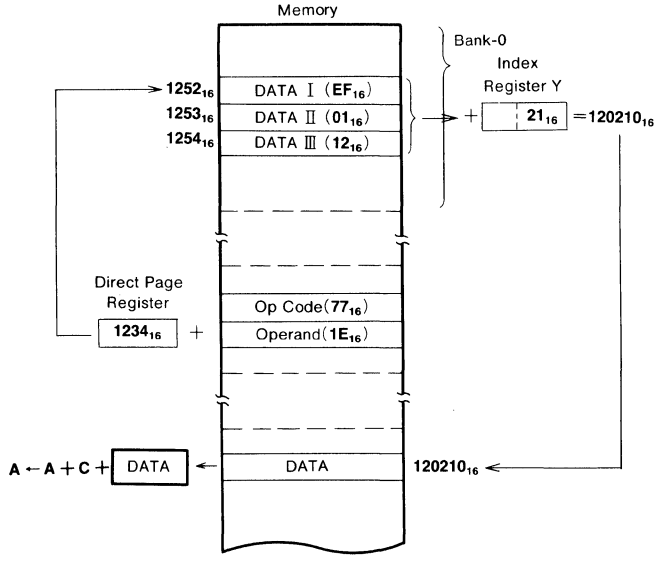
Function : The value obtained by adding the instruction's second byte and the contents of the direct page register specifies 3 adjacent bytes in memory bank-0, and the value obtained by adding the contents of these bytes and the contents of the index register Y specifies the address of the memory location where the actual data is stored. If, however, the value obtained by adding the contents of the instruction's second byte and the direct page register exceeds the bank-0 range, the specified location will be in bank-1. The 3 adjacent bytes memory location may be spread over two different banks.

Instruction : ADC, AND, CMP, DIV, EOR, LDA, MPY, ORA, SBC, STA

ex. : Mnemonic — Machine Code

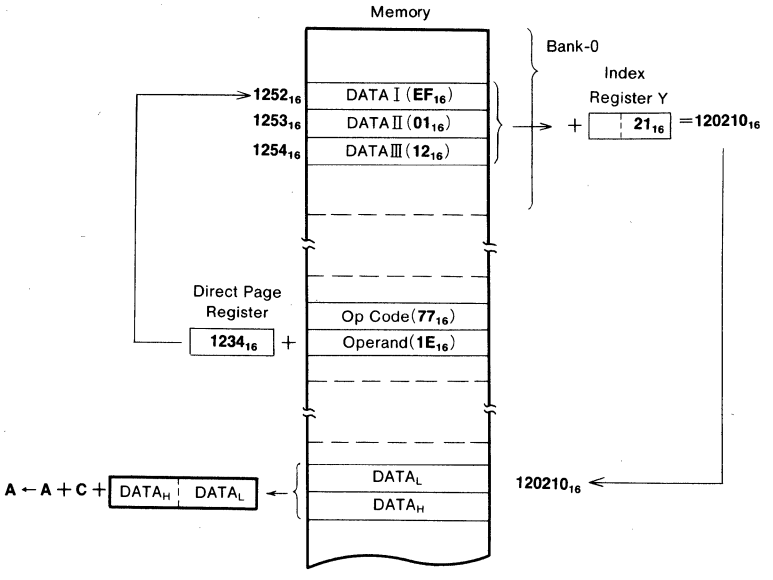
ADCL A,(1EH), Y — 77_{16} $1E_{16}$

(m=1, x=1)

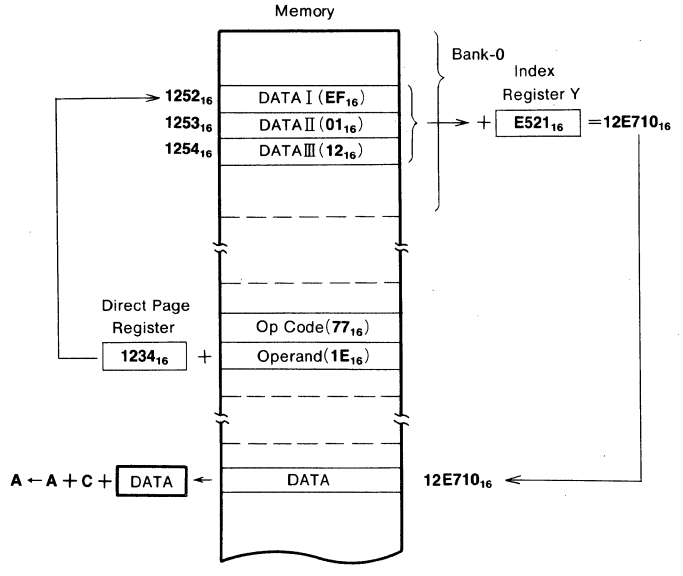


Direct Indirect Long Indexed Y

ex. : Mnemonic **ADCL A,(1EH), Y** **(m=0, x=1)** Machine Code **77_{16} $1E_{16}$**



ex. : Mnemonic **ADCL A,(1EH), Y** **(m=1, x=0)** Machine Code **77_{16} $1E_{16}$**



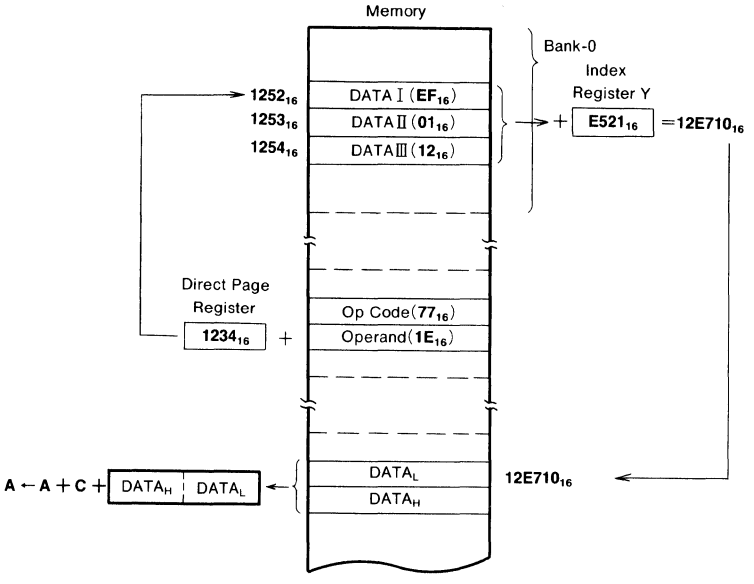
Direct Indirect Long Indexed Y

ex. : Mnemonic

ADCL A,(1EH), Y
(m=0, x=0)

Machine Code

77_{16} $1E_{16}$



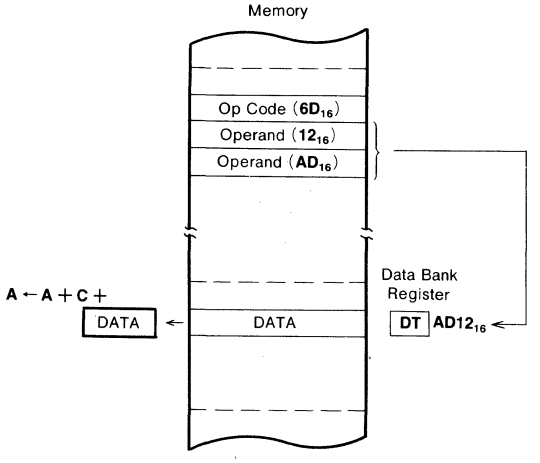
Absolute

Mode : Absolute addressing mode

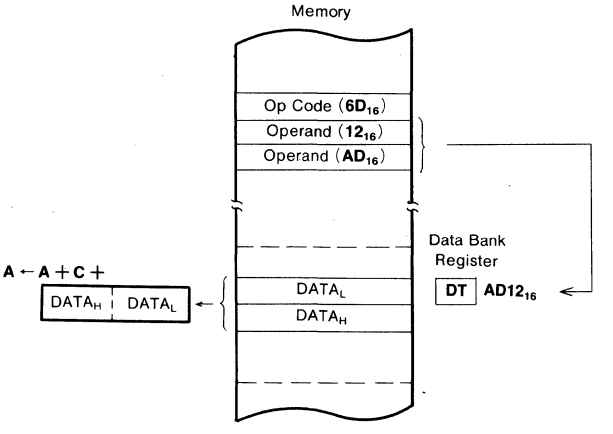
Function : The contents of the memory locations specified by the instruction's second and third bytes and the contents of the data bank register are the actual data. Note that, in the cases of the JMP and JSR instructions, the instructions' second and third byte contents are transferred to the program counter.

Instruction : ADC, AND, ASL, CMP, CPX, CPY, DEC, DIV, EOR, INC, JMP, JSR, LDA, LDM, LDX, LDY, LSR, MPY, ORA, ROL, ROR, SBC, STA, STX, STY

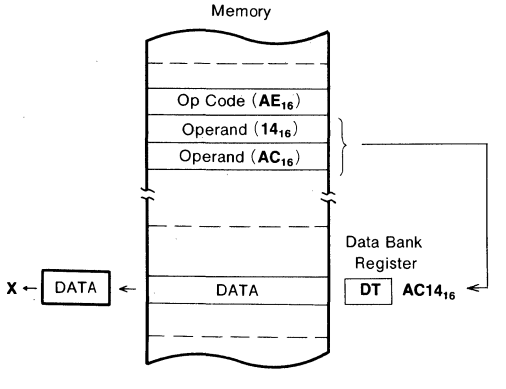
ex. : Mnemonic **ADC A, 0AD12H** (**m=1**) Machine Code $6D_{16}$ 12_{16} AD_{16}



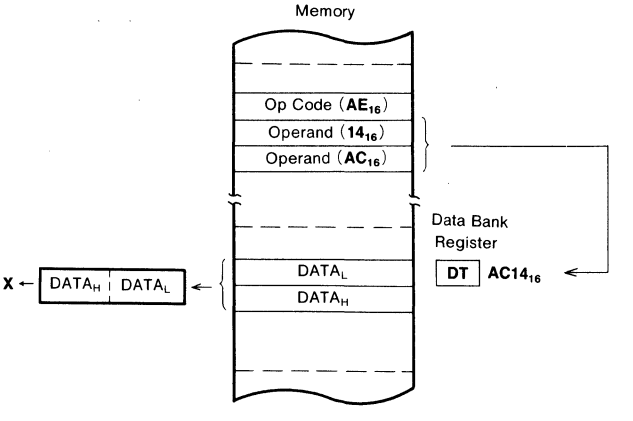
ex. : Mnemonic **ADC A, 0AD12H** (**m=0**) Machine Code $6D_{16}$ 12_{16} AD_{16}



ex. : Mnemonic **LDX 0AC14H** (**x=1**) Machine Code AE_{16} 14_{16} AC_{16}



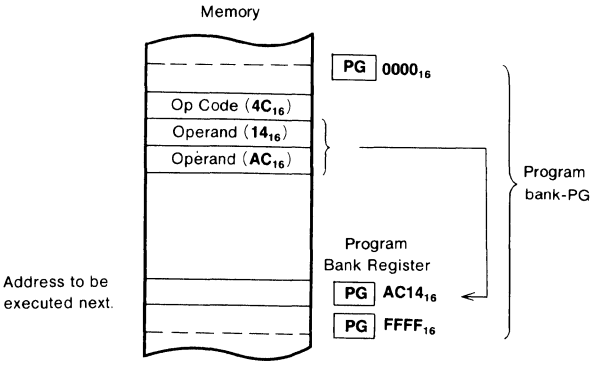
ex. : Mnemonic **LDX 0AC14H** (**x=0**) Machine Code AE_{16} 14_{16} AC_{16}



Absolute

ex. : Mnemonic — Machine Code

JMP 0AC14H — **$4C_{16}$ 14_{16} AC_{16}**



Program bank register contents are not affected.

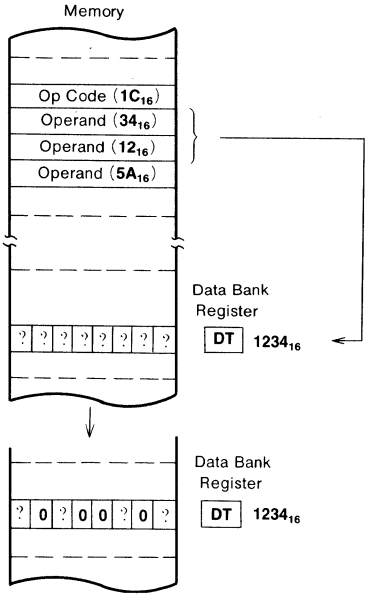
Absolute Bit

Mode : Absolute bit addressing mode

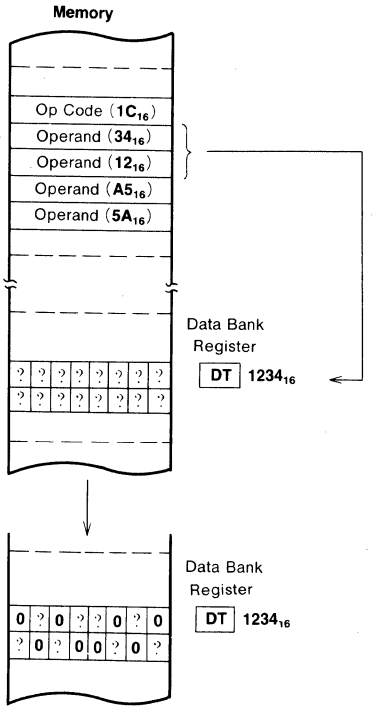
Function : The contents of the instruction's second and third bytes and the contents of the data bank register specify the memory locations, and data for multiple bit positions in the memory locations are specified by a bit pattern specified in the instruction's fourth and fifth bytes (the fourth byte only if the m flag is set to 1).

Instruction : CLB, SEB

ex. : Mnemonic **CLB #5AH, 1234H (m=1)** Machine Code **$1C_{16}$ 34_{16} 12_{16} $5A_{16}$**



ex. : Mnemonic **CLB #5AA5H, 1234H (m=0)** Machine Code **$1C_{16}$ 34_{16} 12_{16} $A5_{16}$ $5A_{16}$**



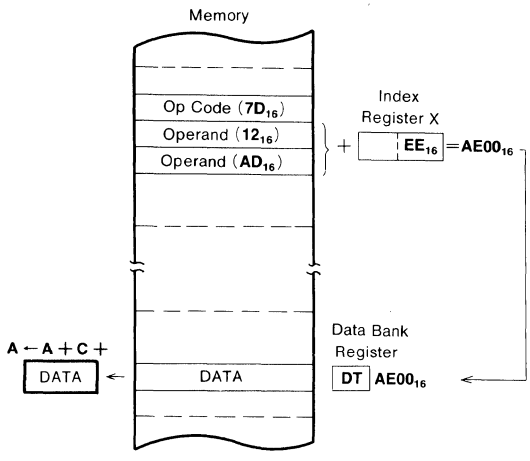
Absolute Indexed X

Mode : Absolute indexed X addressing mode

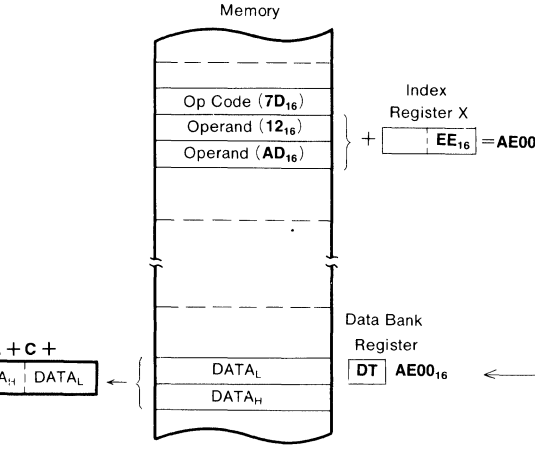
Function : The contents of the memory locations specified by a value resulting from addition of a 16-bit numeric value expressed by the instruction's second and third bytes with the contents of the index register X and the contents of the data bank register are the actual data. If, however, addition of the numeric value expressed by the instruction's second and third bytes with the contents of the index register X generates a carry, the bank number will be 1 larger than the contents of the data bank register.

Instruction : ADC, AND, ASL, CMP, DEC, DIV, EOR, INC, LDA, LDM, LDY, LSR, MPY, ORA, ROL, ROR, SBC, STA

ex. : Mnemonic **ADC A, 0AD12H, X** (**m=1, x=1**) Machine Code **$7D_{16}$ 12_{16} AD_{16}**

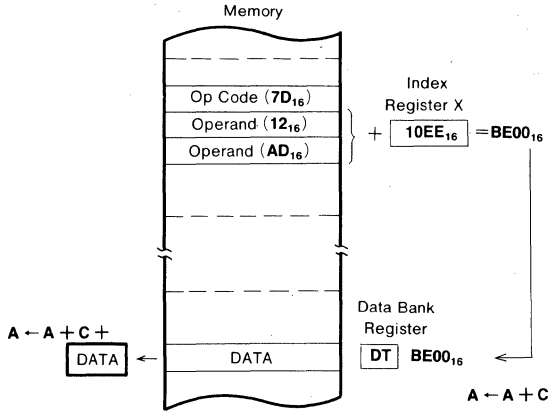


ex. : Mnemonic **ADC A, 0AD12H, X** (**m=0, x=1**) Machine Code **$7D_{16}$ 12_{16} AD_{16}**

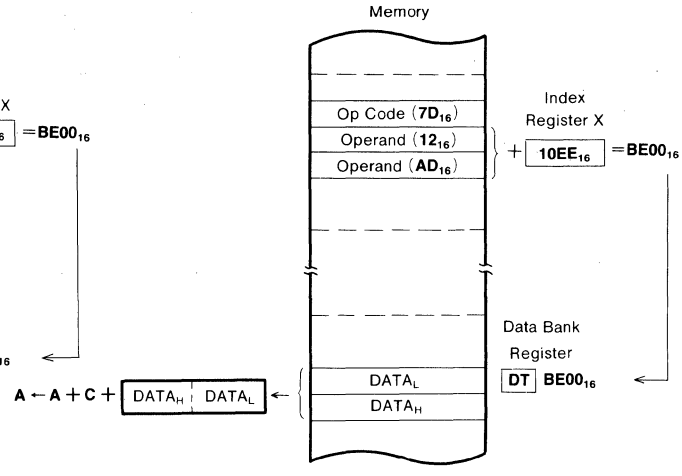


Absolute Indexed X

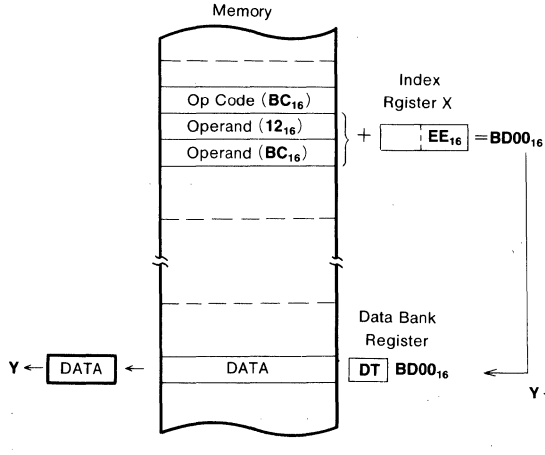
ex. : Mnemonic **ADC A, 0AD12H, X** (**m=1, x=0**) Machine Code $7D_{16}$ 12_{16} AD_{16}



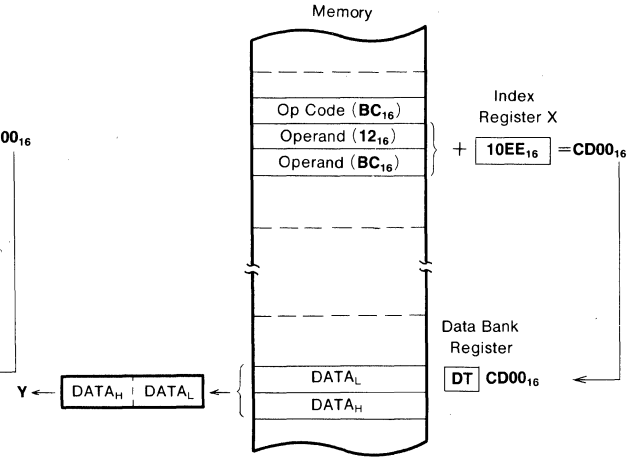
ex. : Mnemonic **ADC A, 0AD12H, X** (**m=0, x=0**) Machine Code $7D_{16}$ 12_{16} AD_{16}



ex. : Mnemonic **LDY 0BC12H, X** (**x=1**) Machine Code BC_{16} 12_{16} BC_{16}



ex. : Mnemonic **LDY 0BC12H, X** (**x=0**) Machine Code BC_{16} 12_{16} BC_{16}



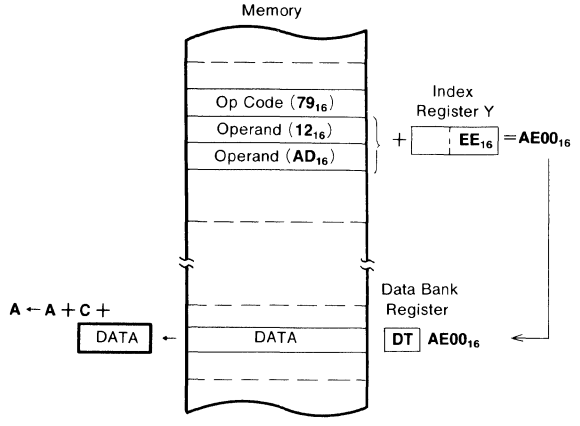
Absolute Indexed Y

Mode : Absolute indexed Y addressing mode

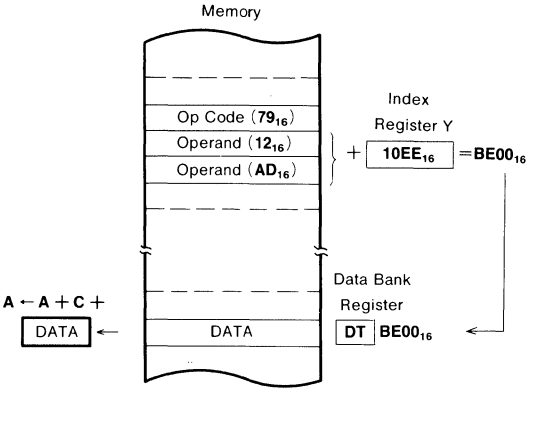
Function : The contents of the memory locations specified by a value resulting from addition of a 16-bit numeric value expressed by the instruction's second and third bytes with the contents of the index register Y and the contents of the data bank register are the actual data. If, however, addition of the numeric value expressed by the instruction's second and third bytes with the contents of the index register Y generates a carry, the bank number will be 1 larger than the contents of the data bank register.

Instruction : ADC, AND, CMP, DIV, EOR, LDA, LDX, MPY, ORA, SBC, STA

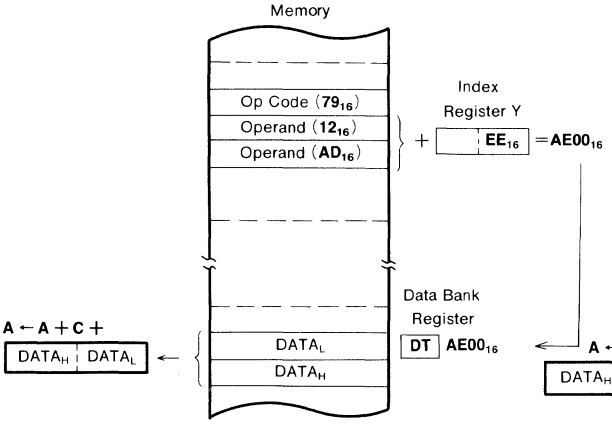
ex. : Mnemonic **ADC A, 0AD12H, Y** (**m=1, x=1**) Machine Code **79_{16} 12_{16} AD_{16}**



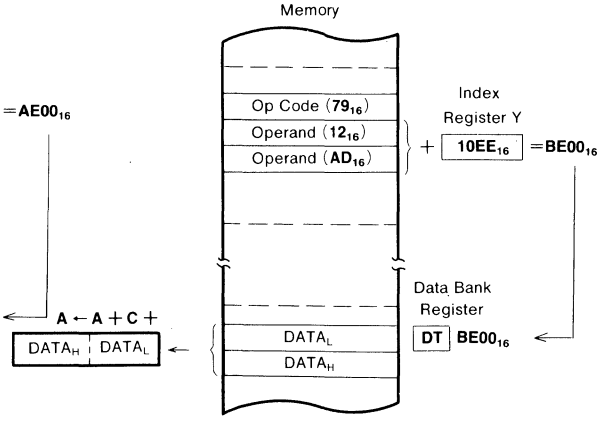
ex. : Mnemonic **ADC A, 0AD12H, Y** (**m=1, x=0**) Machine Code **79_{16} 12_{16} AD_{16}**



ex. : Mnemonic **ADC A, 0AD12H, Y** (**m=0, x=1**) Machine Code **79_{16} 12_{16} AD_{16}**

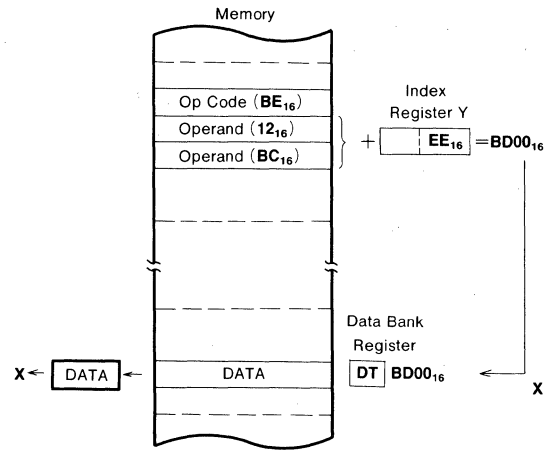


ex. : Mnemonic **ADC A, 0AD12H, Y** (**m=0, x=0**) Machine Code **79_{16} 12_{16} AD_{16}**

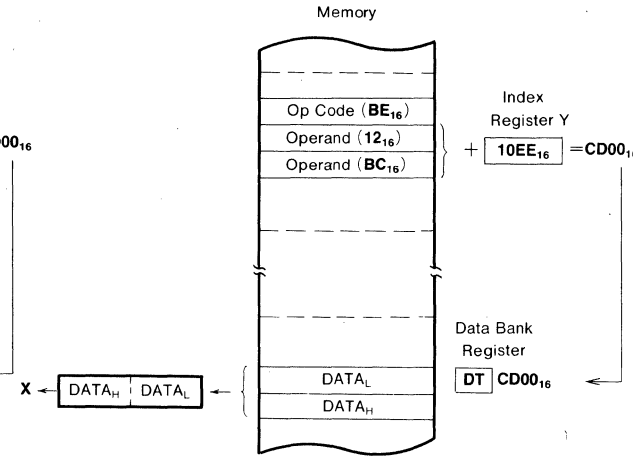


Absolute Indexed Y

ex. : Mnemonic **LDX 0BC12H, Y** (**x=1**) Machine Code **BE_{16} 12_{16} BC_{16}**



ex. : Mnemonic **LDX 0BC12H, Y** (**x=0**) Machine Code **BE_{16} 12_{16} BC_{16}**



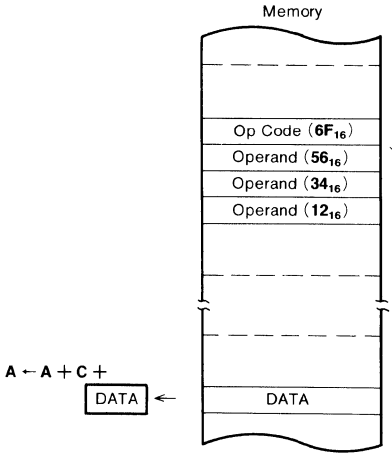
Absolute Long

Mode : Absolute long addressing mode

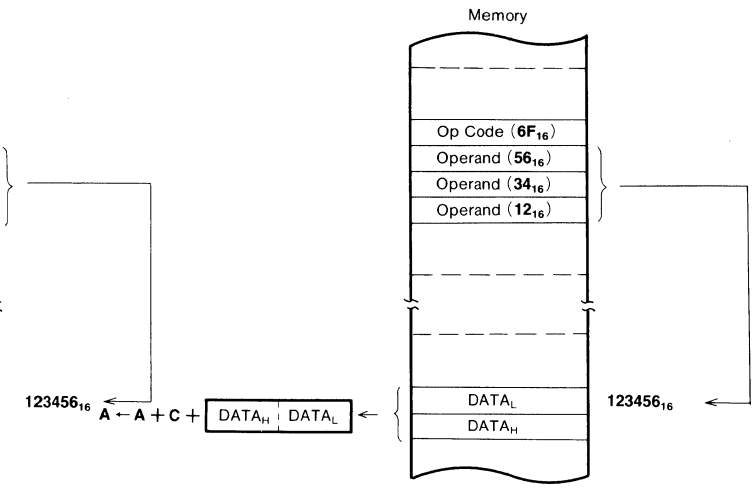
Function : The contents of the memory locations specified by the instruction's second, third and fourth bytes become the actual data. Note that, in the cases of the JMP and JSR instructions, the instructions' second and third byte contents are transferred to the program counter and the fourth byte contents are transferred to the program bank register.

Instruction : ADC, AND, CMP, DIV, EOR, JMP, JSR, LDA, MPY, ORA, SBC, STA

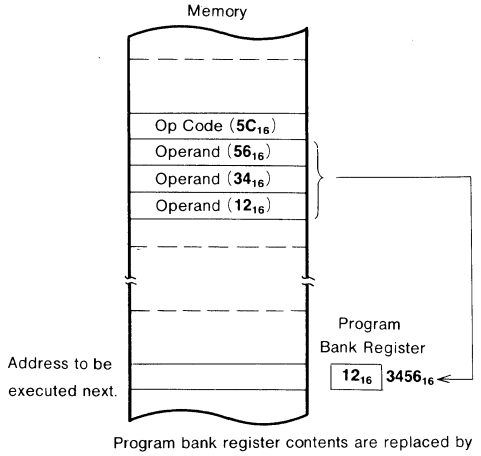
ex. : Mnemonic **ADC A, 123456H (m=1)** Machine Code **$6F_{16}$ 56_{16} 34_{16} 12_{16}**



ex. : Mnemonic **ADC A, 123456H (m=0)** Machine Code **$6F_{16}$ 56_{16} 34_{16} 12_{16}**



ex. : Mnemonic **JMP 123456H** Machine Code **$5C_{16}$ 56_{16} 34_{16} 12_{16}**



Program bank register contents are replaced by the third operand.

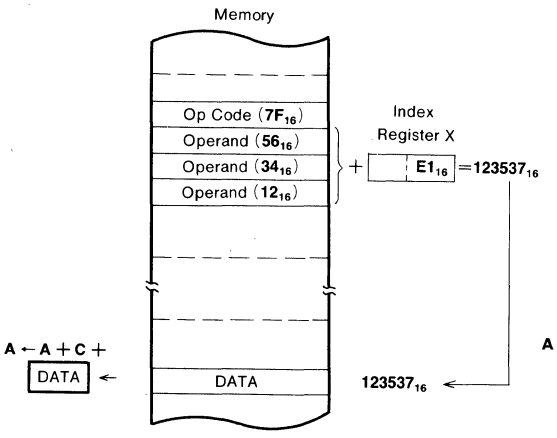
Absolute Long Indexed X

Mode : Absolute long indexed X addressing mode

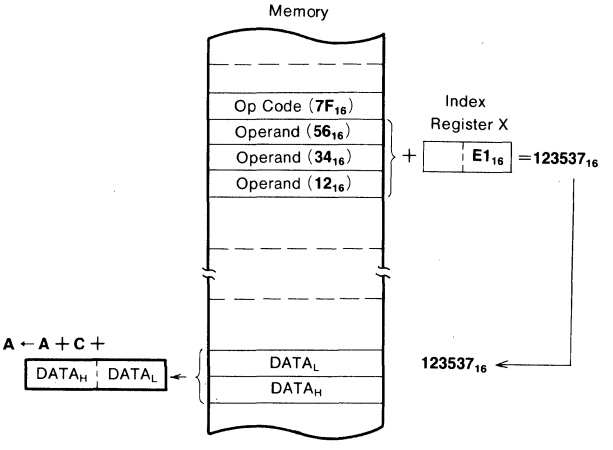
Function : The contents of the memory location specified by adding the numeric value expressed by the instruction's second, third and fourth bytes with the contents of the index register X are the actual data.

Instruction : ADC, AND, CMP, DIV, EOR, LDA, MPY, ORA, SBC, STA

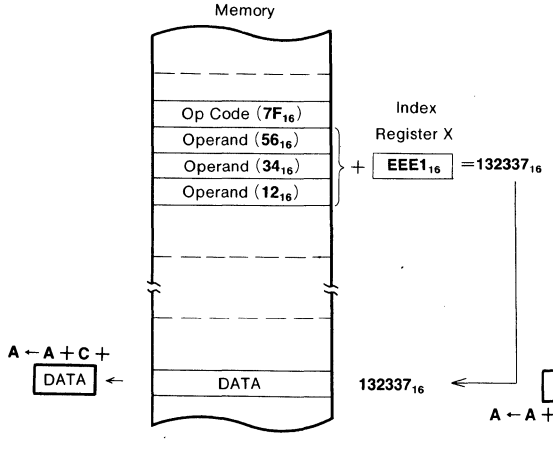
ex. : Mnemonic **ADC A, 123456H, X** (**m=1, x=1**) Machine Code $7F_{16}$ 56_{16} 34_{16} 12_{16}



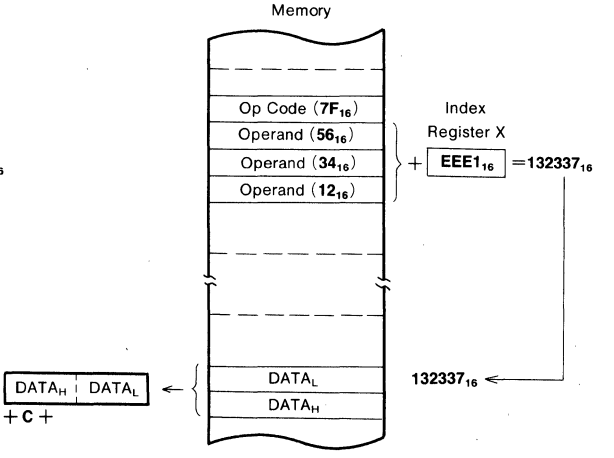
ex. : Mnemonic **ADC A, 123456H, X** (**m=0, x=1**) Machine Code $7F_{16}$ 56_{16} 34_{16} 12_{16}



ex. : Mnemonic **ADC A, 123456H, X** (**m=0, x=1**) Machine Code $7F_{16}$ 56_{16} 34_{16} 12_{16}



ex. : Mnemonic **ADC A, 123456H, X** (**m=0, x=0**) Machine Code $7F_{16}$ 56_{16} 34_{16} 12_{16}



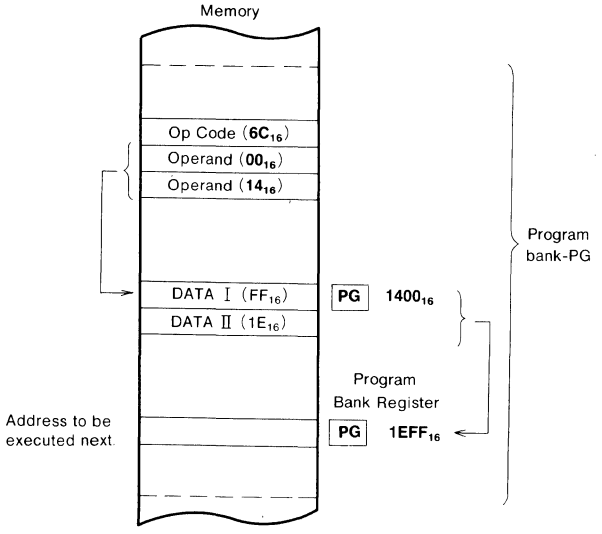
Absolute Indirect

Mode : Absolute indirect addressing mode

Function : The instruction's second and third bytes specify 2 adjacent bytes in memory, and the contents of these bytes specify the address within the same program bank to which a jump is to be made.

Instruction : JMP

ex. : Mnemonic **JMP(1400H)** Machine Code **$6C_{16}$ 00_{16} 14_{16}**



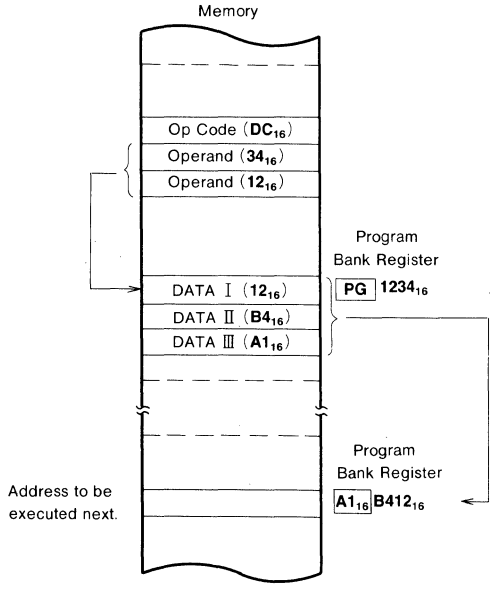
Absolute Indirect Long

Mode : Absolute indirect long addressing mode

Function : The instruction's second and third bytes specify 3 adjacent bytes in memory, and the contents of these bytes specify the address to which a jump is to be made.

Instruction : JMP

ex. : Mnemonic **JMPL(1234H)** Machine Code **DC_{16} 34_{16} 12_{16}**



DATA Ⅲ is loaded in the program bank register.

Absolute Indexed X Indirect

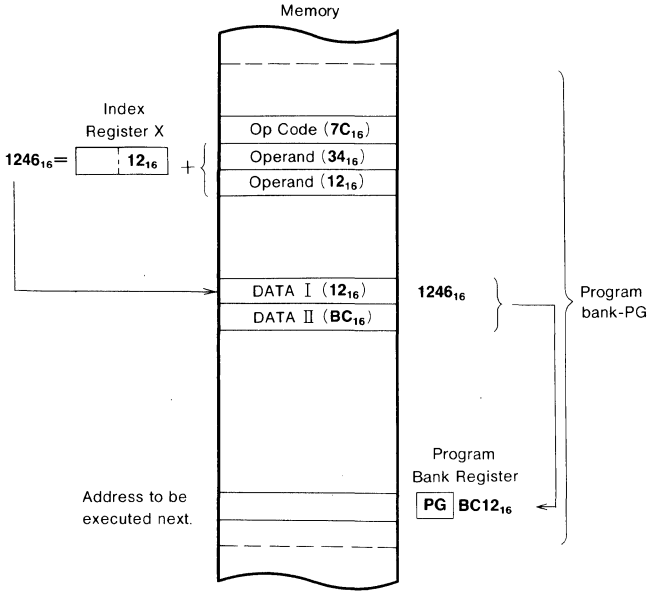
Mode : Absolute indexed X indirect addressing mode

Function : The value obtained by adding the instruction's second and third bytes and the contents of the index register X specifies 2 adjacent bytes in memory, and the contents of these bytes specify the address to which a jump is to be made.

Instruction : JMP, JSR

ex. : Mnemonic
JMP(1234H, X)
(x=1)

Machine Code
$7C_{16}$ 34_{16} 12_{16}



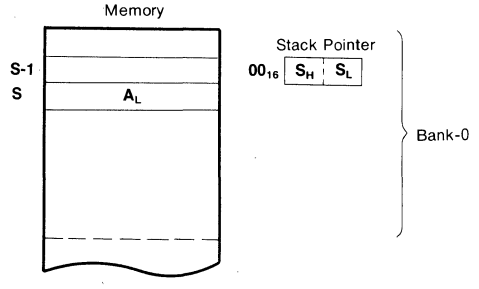
Stack

Mode : Stack addressing mode

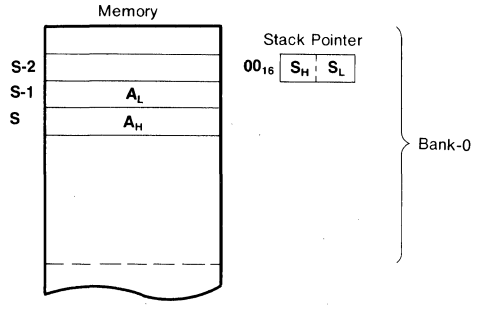
Function : Register contents are saved to or restored from the memory location specified by the stack pointer. The stack pointer is set in bank-0.

Instruction : PEA, PEI, PER, PHA, PHB, PHD, PHG, PHP, PHT, PHX, PHY, PLA, PLB, PLD, PLP, PLT, PLX, PLY, PSH, PUL

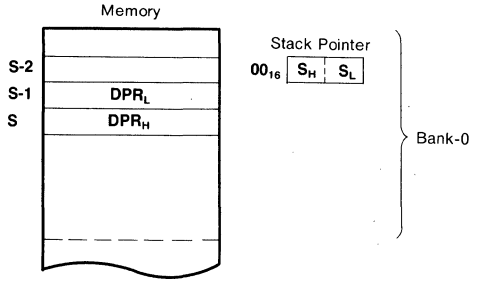
ex. : Mnemonic **PHA** (**m=1**) Machine Code **48_{16}**



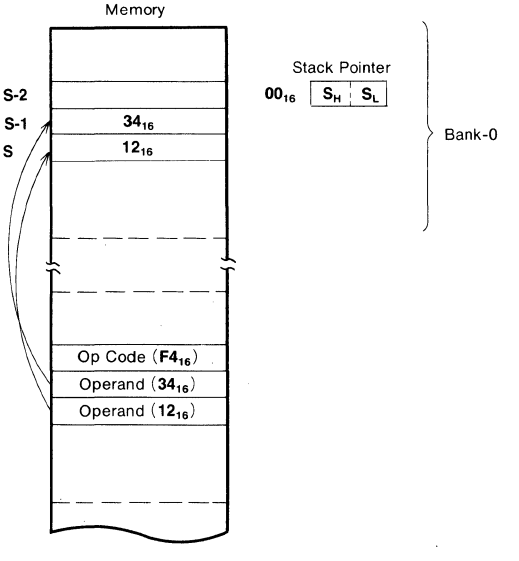
ex. : Mnemonic **PHA** (**m=0**) Machine Code **48_{16}**



ex. : Mnemonic **PHD** Machine Code **$0B_{16}$**

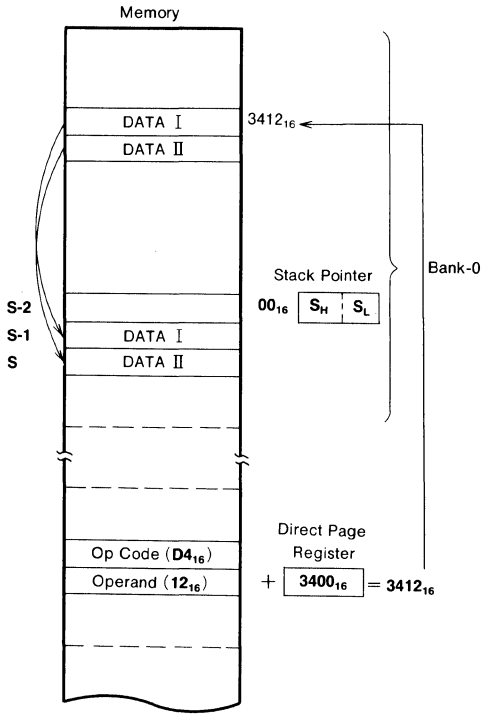


ex. : Mnemonic **PEA # 1234H** Machine Code **$F4_{16}$ 34_{16} 12_{16}**

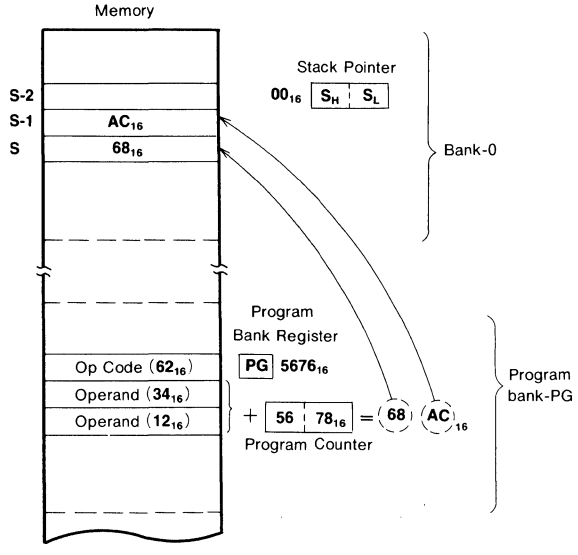


Stack

ex. : Mnemonic **PEI # 12H** Machine Code **$D4_{16}$ 12_{16}**



ex. : Mnemonic **PER # 1234H** Machine Code **62_{16} 34_{16} 12_{16}**



Relative

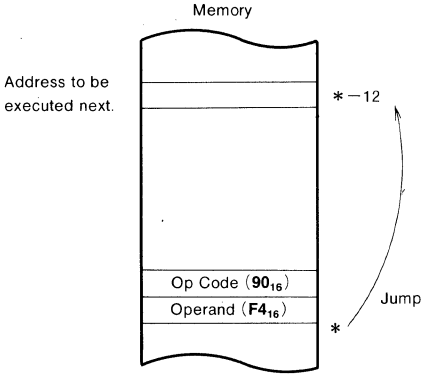
Mode : Relative addressing mode

Function : Branching occurs to the address specified by the value resulting from addition of the contents of the program counter and the instruction's second byte. In the case of a long branch by the BRA instruction, a 15-bit signed numeric value formed by the contents of the instruction's second and third bytes is added to the program counter contents. If the addition generates a carry or borrow, 1 is added to or subtracted from the program bank register.

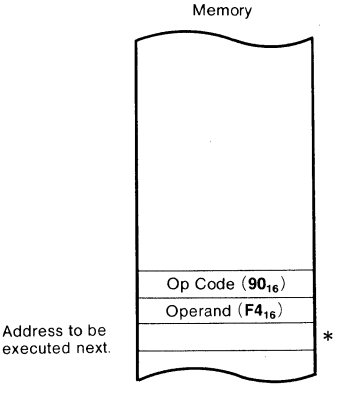
Instruction : BCC, BCS, BEQ, BMI, BNE, BPL, BRA, BVC, BVS

ex. : Mnemonic **BCC ＊−12** Machine Code **90_{16} $F4_{16}$**

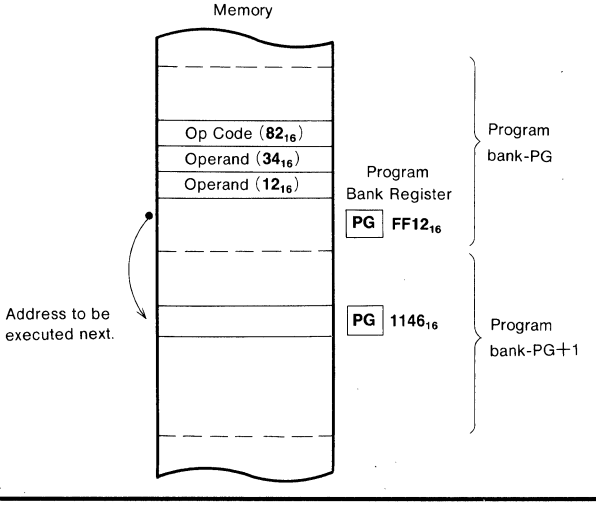
Branches to the address ＊−12 if the carry flag (C) has been cleared.



Advances to the address ＊ if the carry flag (C) has been set.



ex. : Mnemonic **BRA 1234H** Machine Code **82_{16} 34_{16} 12_{16}**



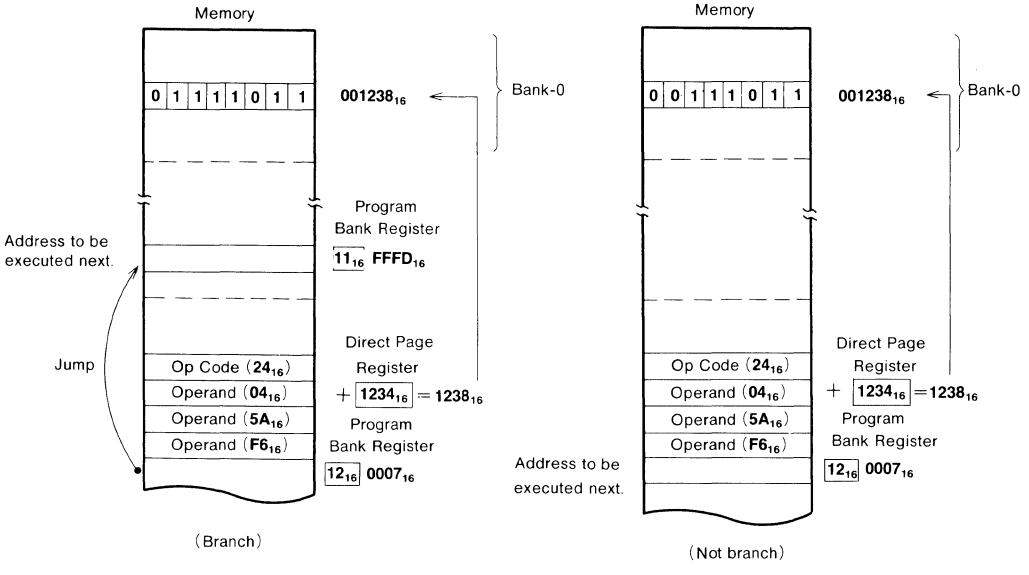
Direct Bit Relative

Mode : Direct bit relative addressing mode

Function : Specifies the bank-0 memory location by the value obtained by adding the instruction's second byte to the direct page register's contents, and specifies the positions of multiple bits in the memory location by the bit pattern in the third and fourth bytes (the third byte only if the m flag is set to 1). Then, if the specified bits all satisfy the branching conditions, the instruction's fifth byte (or the fourth byte if the m flag is set to 1) is added to the program counter as a signed value, generating the branching destination address. If, however, addition of the instruction's second byte to the direct page register's contents result in a value that exceeds the bank-0 range, the specified location will be in bank-1.

Instruction : BBC, BBS

ex. : Mnemonic Machine Code
BBS #5AH, 04H, 0F6H 24_{16} 04_{16} $5A_{16}$ $F6_{16}$
(m=1)

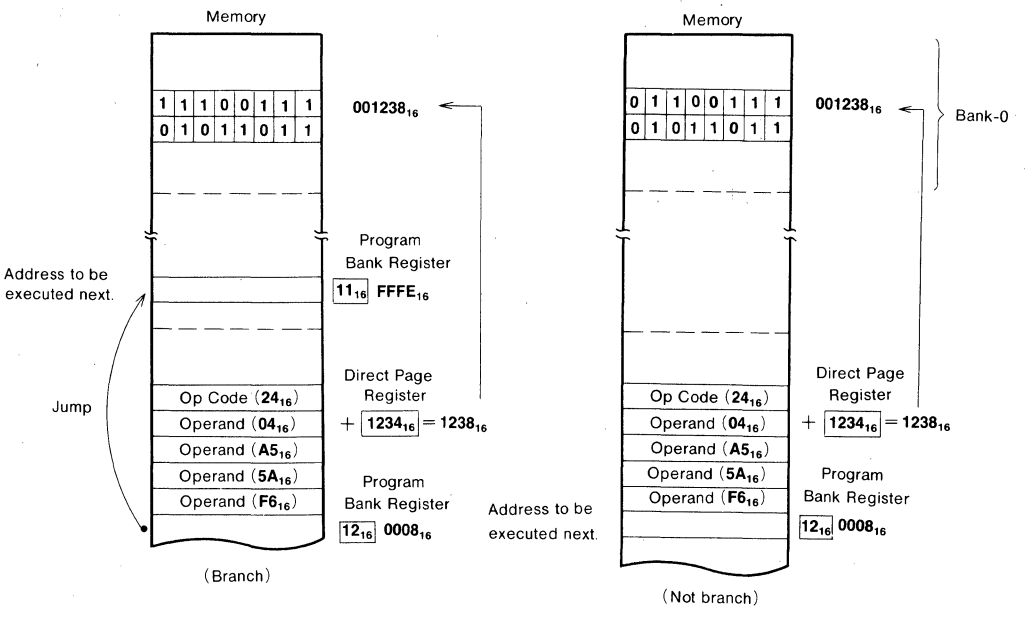


Direct Bit Relative

ex. : Mnemonic — Machine Code

BBS #5AA5H, 04H, 0F6H — **24_{16} 04_{16} $A5_{16}$ $5A_{16}$ $F6_{16}$**

(m=0)



Absolute Bit Relative

Mode : Absolute bit relative addressing mode

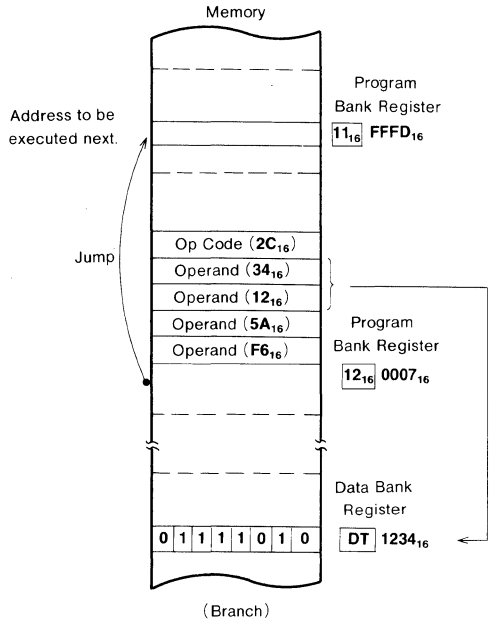
Function : The instruction's second and third bytes and the contents of the data bank register specify the memory location, and data for the memory location's multiple bits is specified by a bit pattern in the instruction's fourth and fifth bytes (the fourth byte only if the m flag is set to 1). Then, if the specified bits all satisfy the branching conditions, the instruction's sixth byte (or the fifth byte if the m flag is set to 1) is added to the program counter as a signed value, generating the branching destination address.

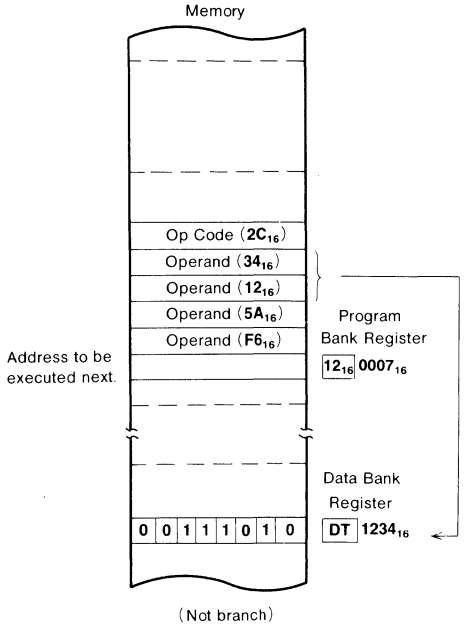
Instruction : BBC, BBS

ex. : Mnemonic — Machine Code

BBS #5AH, 1234H, 0F6H — $2C_{16}\ 34_{16}\ 12_{16}\ 5A_{16}\ F6_{16}$

(m=1)

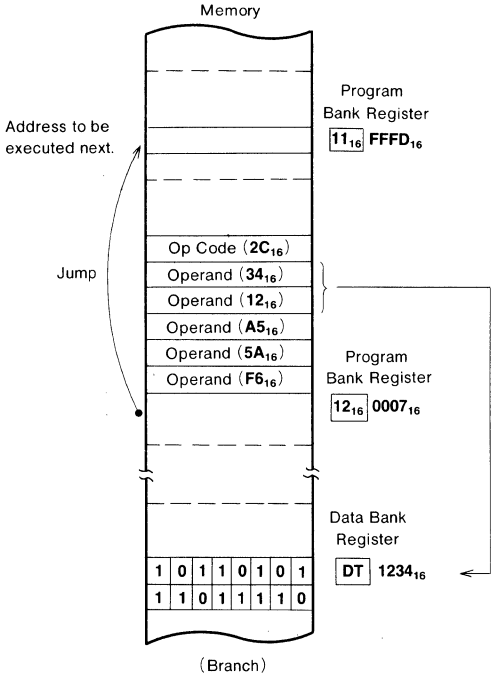


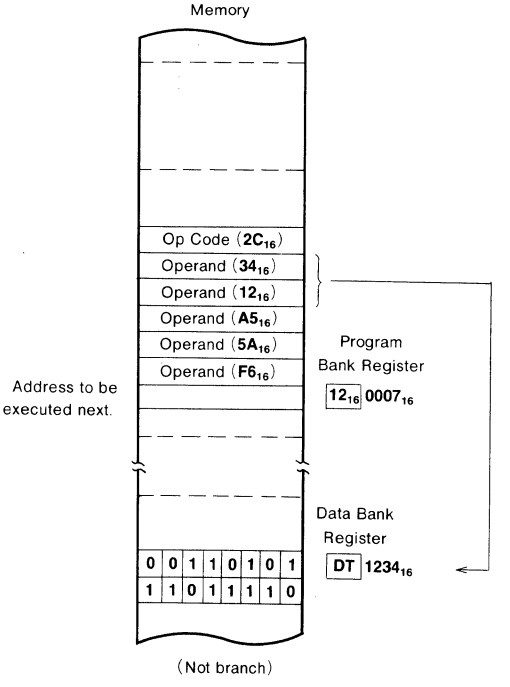


Absolute Bit Relative

ex. : Mnemonic
BBS #5AA5H, 1234H, 0F6H
(m=0)

Machine Code
$2C_{16}$ 34_{16} 12_{16} $A5_{16}$ $5A_{16}$ $F6_{16}$





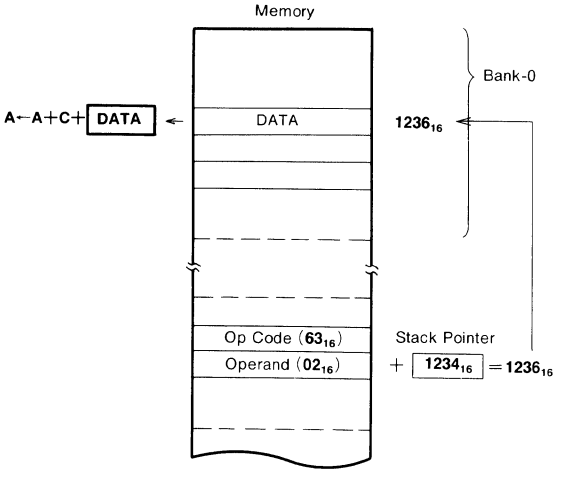
Stack Pointer Relative

Mode : Stack pointer relative addressing mode

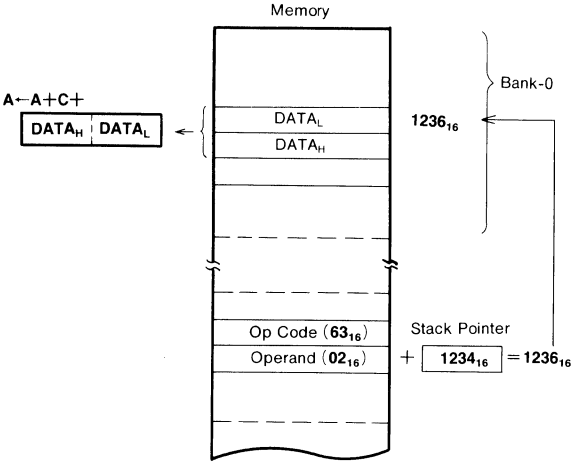
Function : The contents of a bank-0 memory location specified by the value resulting from addition of the instruction's second byte and the contents of the stack pointer become the actual data. If, however, the value obtained by adding the contents of the instruction's second byte and the stack pointer's contents exceeds the bank-0 range, the specified location will be in bank-1.

Instruction : ADC, AND, CMP, DIV, EOR, LDA, MPY, ORA, SBC, STA

ex. : Mnemonic **ADC A, 02H, S** (**m=1**) Machine Code 63_{16} 02_{16}



ex. : Mnemonic **ADC A, 02H, S** (**m=0**) Machine Code 63_{16} 02_{16}



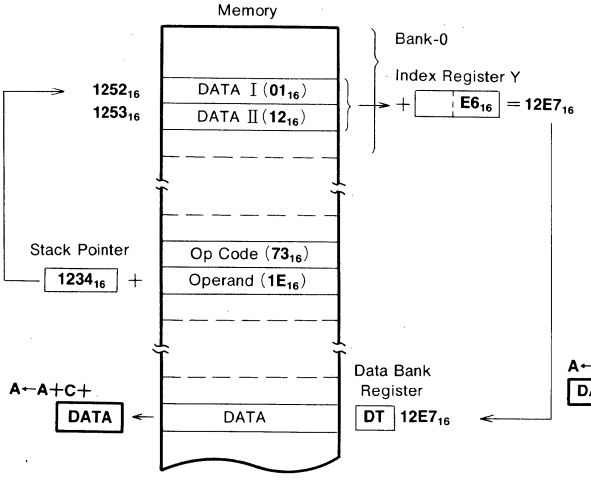
Stack Pointer Relative Indirect Indexed Y

Mode : Stack pointer relative indirect indexed Y addressing mode

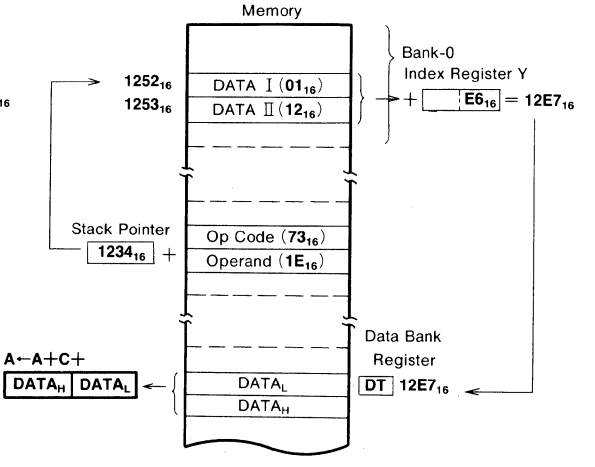
Function : The value obtained by adding the instruction's second byte and the contents of the stack pointer specifies 2 adjacent bytes in memory. The value obtained by adding the contents of these bytes and the contents of the index register Y specifies address of the actual data in memory bank-DT (DT is contents of data bank register). If addition of the 2 bytes in memory with the contents of the index register Y generate a carry, the bank number will be 1 larger than the contents of the data bank register.

Instruction : ADC, AND, CMP, DIV, EOR, LDA, MPY, ORA, SBC, STA

ex. : Mnemonic **ADC A,(1EH, S), Y** **(m=1, x=1)** Machine Code 73_{16} $1E_{16}$



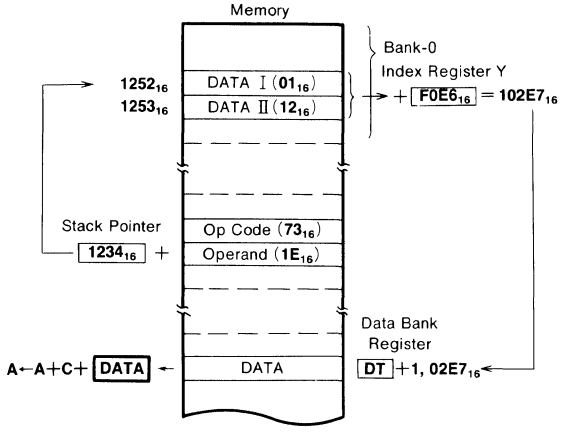
ex. : Mnemonic **ADC A,(1EH, S), Y** **(m=0, x=1)** Machine Code 73_{16} $1E_{16}$



Stack Pointer Relative Indirect Indexed Y

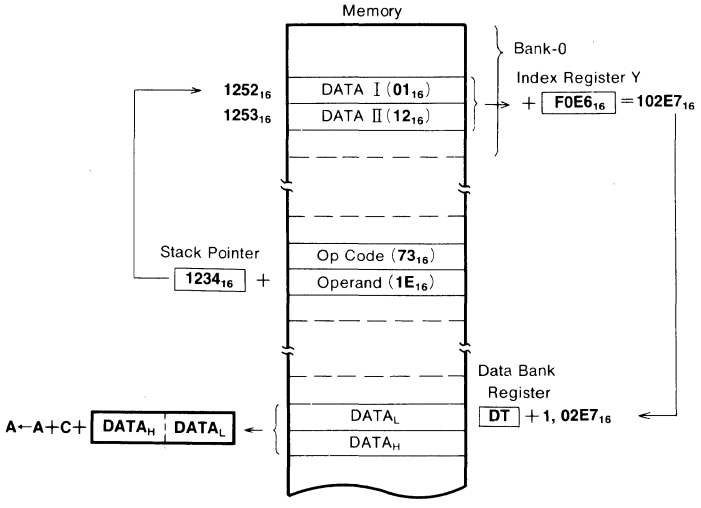
ex. : Mnemonic **ADC A, (1EH, S), Y** (**m=1,x=0**)

Machine Code **73_{16} $1E_{16}$**



ex. : Mnemonic **ADC A, (1EH, S), Y** (**m=0, x=0**)

Machine Code **73_{16} $1E_{16}$**



Block Transfer

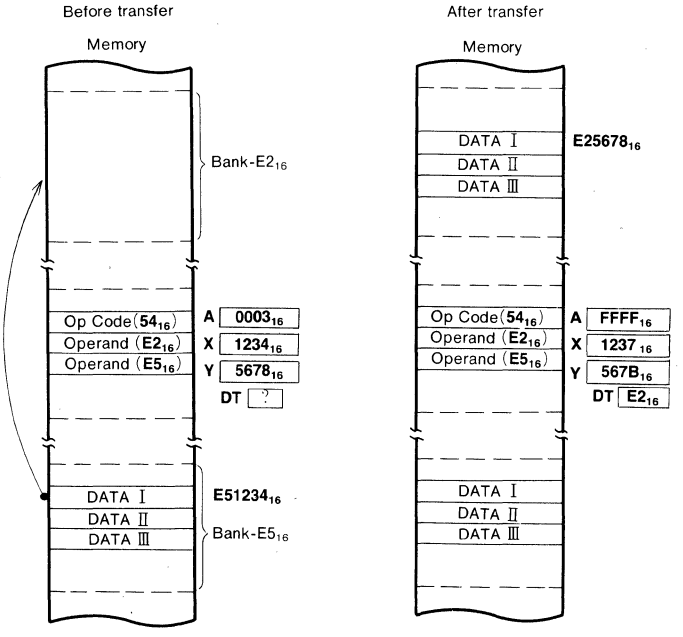
Mode : Block transfer addressing mode

Function : The instruction's second byte specifies the transfer-to data bank, and the contents of the index register Y specify the transfer-to address within the data bank. The instruction's third byte specifies the transfer-from data bank, and the contents of the index register X specify the address in the data bank where the data to be transferred is stored. The contents of the accumulator A constitute the number of bytes to be transferred. Upon termination of transfer, the contents of the data bank register will specify the transfer-to data bank. The MVN instruction is used for transfer to lower address location. In this case, the contents of the index registers X and Y are incremented each time data is transferred. The MVP instruction is used for transfer to higher address location. In this case, the contents of the index registers X and Y are decremented each time data is transferred. The block of data to be transferred may cross over the bank boundary.

Instruction : MVN, MVP

ex. : Mnemonic — Machine Code

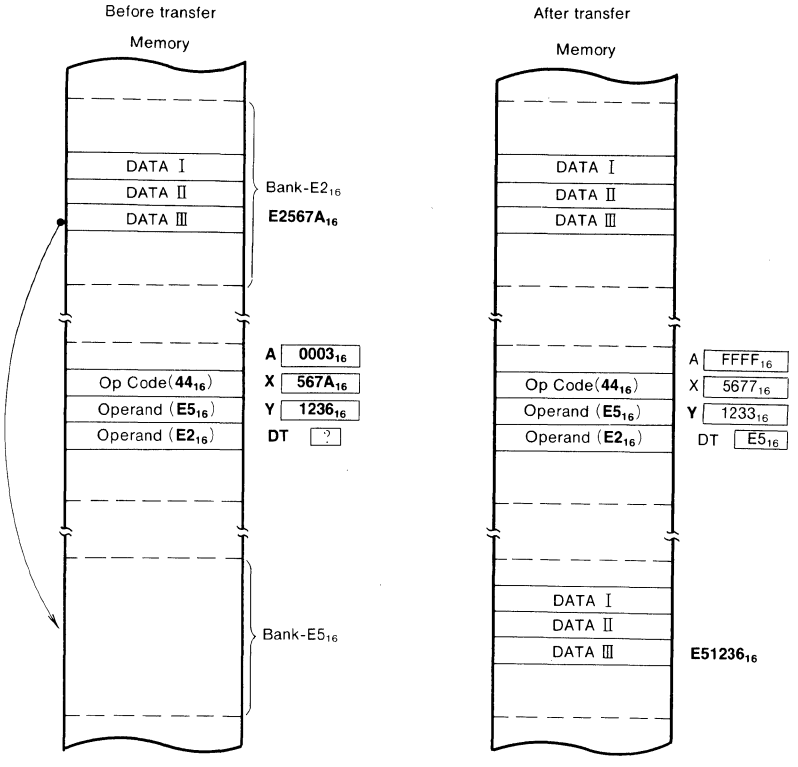
MVN 0E2H, 0E5H — **54_{16} $E2_{16}$ $E5_{16}$**



Block Transfer

ex. : Mnemonic — Machine Code

MVP 0E5H, 0E2H — **44_{16} $E5_{16}$ $E2_{16}$**



4. Instructions

4.1 Instruction Set

The **Series MELPS 7700** microcomputers support a set of 103 instructions which are described in this chapter. This section presents overviews of these instructions, and Sec. 4.2 presents the detailed description for each instruction.

4.1.1 Data Transfer Instructions

The data transfer instructions move data between data and registers, between a register and the memory, between registers or between memory devices.

The following table summarizes the various data transfer instructions supported by the **Series MELPS 7700** :

Category	Instruction	Description
Load	LDA	Loads the contents of memory into the accumulator.
	LDM	Loads an immediate value into the memory.
	LDT	Loads an immediate value into the data bank register.
	LDX	Loads the contents of memory into the index register X.
	LDY	Loads the contents of memory into the index register Y.
Store	STA	Stores the contents of the accumulator in the memory.
	STX	Stores the contents of the index register X in the memory.
	STY	Stores the contents of the index register Y in the memory.
Transfer	TAX	Transfers the contents of the accumulator A to the index register X.
	TXA	Transfers the contents of the index register X to the accumulator A.
	TAY	Transfers the contents of the accumulator A to the index register Y.
	TYA	Transfers the contents of the index register Y to the accumulator A.
	TSX	Transfers the contents of the stack pointer to the index register X.
	TXS	Transfers the contents of the index register X to the stack pointer.
	TAD	Transfers the contents of the accumulator A to the direct page register.
	TDA	Transfers the contents of the direct page register to the accumulator A.
	TAS	Transfers the contents of the accumulator A to the stack pointer.
	TSA	Transfers the contents of the stack pointer to the accumulator A.
	TBD	Transfers the contents of the accumulator B to the direct page register.
	TDB	Transfers the contents of the direct page register to the accumulator B.
	TBS	Transfers the contents of the accumulator B to the stack pointer.

Category	Instruction	Description
Transfer	TSB	Transfers the contents of the stack pointer to the accumulator B.
	TBX	Transfers the contents of the accumulator B to the index register X.
	TXB	Transfers the contents of the index register X to the accumulator B.
	TBY	Transfers the contents of the accumulator B to the index register Y.
	TYB	Transfers the contents of the index register Y to the accumulator B.
	TXY	Transfers the contents of the index register X to the index register Y.
	TYX	Transfers the contents of the index register Y to the index register X.
	MVN	Transfers a block of data from the lower addresses.
	MVP	Transfers a block of data from the higher addresses.
Stack operation	PSH	Saves the contents of the specified register to the stack.
	PUL	Restores the contents of stack to the specified register.
	PHA	Saves the contents of the accumulator A to the stack.
	PLA	Restores the contents of stack to the accumulator A.
	PHP	Saves the contents of the program status register to the stack.
	PLP	Restores the contents of stack to the program status register.
	PHB	Saves the contents of the accumulator B to the stack.
	PLB	Restores the contents of stack to the accumulator B.
	PHD	Saves the contents of the direct page register to the stack.
	PLD	Restores the contents of stack to the direct page register.
	PHT	Saves the contents of the data bank register to the stack.
	PLT	Restores the contents of stack to the data bank register.
	PHX	Saves the contents of the index register X to the stack.
	PLX	Restores the contents of stack to the index register X.
	PHY	Saves the contents of the index register Y to the stack.
	PLY	Restores the contents of stack to the index register Y.
Stack	PHG	Saves the contents of the program bank register to the stack.
	PEA	Saves a the numeric of 2 bytes to the stack.
	PEI	Saves the contents of 2 consecutive bytes in the direct page area to the stack.
	PER	Saves the result of adding a 16-bit numeric value to the program counter contents to the stack.
Exchange	XAB	Swaps the contents of the accumulator A with the contents of the accumulator B.

4.1.2 Arithmetic Instructions

The arithmetic instructions perform addition, subtraction, multiplication, division, logical operation, comparison, rotation and shifting of register and memory contents.

The following table summarizes the arithmetic instructions supported:

Category	Instruction	Description
Addition Subtraction Multiplication Division	ADC	Adds the contents of the accumulator,the contents of memory and the contents of the carry flag.
	SBC	Subtracts the complements of the contents of memory and carry flag from the contents of the accumulator.
	INC	Increments the accumulator or memory contents by 1.
	DEC	Decrements the accumulator or memory contents by 1.
	INX	Increments the contents of the index register X by 1.
	DEX	Decrements the contents of the index register X by 1.
	INY	Increments the contents of the index register Y by 1.
	DEY	Decrements the contents of the index register Y by 1.
	MPY	Multiples the contents of the accumulator A and the contents of memory.
	DIV	Divides the numeric value whose lower byte is the contents of the accumulator A and upper byte is the contents of the accumulator B by the contents of memory.
Logical operation	AND	Performs logical AND between the contents of the accumulator and the contents of memory.
	ORA	Performs logical OR between the contents of the accumulator and the contents of memory.
	EOR	Performs logical exclusive-OR between the contents of the accumulator and the contents of memory.
Comparison	CMP	Compares the contents of the accumulator with the contents of memory.
	CPX	Compares the contents of the index register X and the contents of memory.
	CPY	Compares the contents of the index register Y and the contents of memory.
Shifting, Lotation	ASL	Shifts the contents of the accumulator or memory to the left by 1 bit.
	LSR	Shifts the contents of the accumulator or memory to the right by 1 bit.
	ROL	Links the contents of accumulator or memory with the carry flag, and rotates the result to the left by 1 bit.
	ROR	Links the contents of accumulator or memory with the carry flag, and rotates the result to the right by 1 bit.
	RLA	Rotates the contents of the accumulator A to the left by the specified number of bits.

4.1.3 Bit Manipulation Instructions

The bit manipulation instructions set the specified bits of the processor status register or memory to "1" or "0".

The following table summarizes the bit manipulation instructions supported:

Category	Instruction	Description
Bit manipulation	CLB	Clears the specified memory bit to "0".
	SEB	Sets the specified memory bit to "1".
	CLP	Clears the specified bit of the processor status register's lower byte (PS_L) to "0".
	SEP	Sets the specified bit of the processor status register's lower byte (PS_L) to "1".

4.1.4 Flag Manipulation Instructions

The flag manipulation instructions set to "1" or clear to "0" the C, I, m and V flags.

The following table summarizes the flag manipulation instructions supported:

Category	Instruction	Description
Flag setting	CLC	Clears the contents of carry flag to "0".
	SEC	Sets the contents of carry flag to "1".
	CLM	Clears the contents of data length selection flag to "0".
	SEM	Sets the contents of data length selection flag to "1".
	CLI	Clears the contents of interrupt disable flag to "0".
	SEI	Sets the contents of interrupt disable flag to "1".
	CLV	Clears the contents of overflow flag to "0".

4.1.5 Branching and Return Instructions

The branching and return instructions enable changing the program execution sequence.

The following table summarizes the branching and return instructions:

Category	Instruction	Description
Jump	JMP	Sets a new address in the program counter and jumps to the new address.
	BRA	Jumps to the address obtained by adding an offset value to the contents of the program counter.
	JSR	Saves the contents of the program counter to the stack and then jumps to the new address.

Category	Instruction	Description
Branch	BBC	Causes a branch if the specified memory bits are all "0".
	BBS	Causes a branch if the specified memory bits are all "1".
	BCC	Causes a branch if the carry flag is set to "0".
	BCS	Causes a branch if the carry flag is set to "1".
	BNE	Causes a branch if the zero flag is set to "0".
	BEQ	Causes a branch if the zero flag is set to "1".
	BPL	Causes a branch if the negative flag is set to "0".
	BMI	Causes a branch if the negative flag is set to "1".
	BVC	Causes a branch if the overflow flag is set to "0".
	BVS	Causes a branch if the overflow flag is set to "1".
Return	RTI	Returns from the interrupt routine to the original routine.
	RTS	Returns from a subroutine to the original routine. The program bank register contents are not restored.
	RTL	Returns from a subroutine to the original routine. The program bank register contents are restored.

4.1.6 Interrupt Instruction (Break Instruction)

The interrupt instruction executes software interrupt.

Category	Instruction	Description
Break	BRK	Executes a software interrupt.

4.1.7 Special Instructions

The special instructions listed below control the clock generator circuit.

Category	Instruction	Description
Special	WIT	Stops the internal clock.
	STP	Stops the oscillator.

4.1.8 Other Instruction

Category	Instruction	Description
Other	NOP	Only advances the program counter.

4.2 Description of Instructions

This section describes the **Series MELPS 7700** instructions individually. To the extent possible, each instruction is described using one page per instruction. Each instruction description page is headed by the instruction mnemonic, and the pages are arranged in alphabetical order of the mnemonics. For each instruction, operation and description of the instruction, status flag changes and a listing sorted by addressing modes of the assembler coding format (Note 1), machine code, bytes-count and cycles-count (Note 2) are presented.

Note1. The assembler coding formats shown are general examples, and they may differ from the actual formats for the assembler used. Please be sure to refer to the mnemonic coding description in the manual for the assembler actually used for programming.

Note2. The cycles-counts shown are the minimum possible, and they vary depending on the following conditions:

- Value of direct page register's lower byte

  The cycles-count shown are for when the direct page register's lower byte (DPR_L) is 00_{16}. When using an addressing mode that uses the direct page register with $DPR_L \neq$"00_{16}", the cycles-count will be 1 more than the value shown.

- Number of bytes that have been loaded in the instruction queue buffer

- Whether the first address of the memory read/write is even- or odd-numbered in accessing the 16-bit data length.

- Accessing of an external memory are with BYTE=1(using 8-bit external bus)

The table below lists the symbols that are used in this section:

Symbol	Description	Symbol	Description
C	Carry flag	DPR	Direct page register
Z	Zero flag	DPR_H	Direct page register's upper 8 bits
I	Interrupt disable flag	DPR_L	Direct page register's lower 8 bits
D	Decimal operation mode flag	PS	Processor status register
x	Index register length selection flag	PS_H	Processor status register's upper 8 bits
m	Data length selection flag	PS_L	Processor status register's lower 8 bits
V	Overflow flag	PS_n	Processor status register's n-th bit
N	Negative flag	M	Memory contents
IPL	Processor interrupt priority level	M(n)	Contents of memory location specified by operand
+	Addition		
−	Subtraction	M(S)	Contents of memory at address indicated by stack pointer
×	Multiplication		
/	Division	M_n	n-th memory location
∧	Logical AND	AD_G	Value of 24-bit address' upper 8-bit (A_{23}~A_{16})
∨	Logical OR		
∨	Exclusive OR	AD_H	Value of 24-bit address' middle 8-bit (A_{15}~A_8)
¯	Negation		
←	Movement to the arrow direction	AD_L	Value of 24-bit address' lower 8-bit (A_7~A_0)
→	Movement to the arrow direction	b_n	n-th bit of data
↔	Movement to the arrow direction	dd	8-bit offset value
Acc	Accumulator	i	Number of transfer bytes or rotation
Acc_H	Accumulator's upper 8 bits	i_1, i_2	Number of registers pushed or pulled
Acc_L	Accumulator's lower 8 bits	imm	8-bit immediate value
A	Accumulator A	imm_1, imm_2	16-bit immediate value (imm_1 specifies the upper 8-bit,and imm_2 specifies the lower 8-bit)
A_H	Accumulator A's upper 8 bits		
A_L	Accumulator A's lower 8 bits	ll	8-bit address value
B	Accumulator B	mmll	16-bit address value (mm specifies the upper 8-bit and ll specifies the lower 8-bit)
B_H	Accumulator B's upper 8 bits		
B_L	Accumulator B's lower 8 bits	hhmmll	24-bit address value (hh specifies the upper 8-bit, mm specifies the middle 8-bit and ll specifies the lower 8-bit)
X	Index register X		
X_H	Index register X's upper 8 bits	nn	8-bit data value
X_L	Index register X's lower 8 bits	n_1, n_2	8-bit data value (Used when coding two 8-bit data side by side)
Y	Index register Y		
Y_H	Index register Y's upper 8 bits	rr	Signed 8-bit data value
Y_L	Index register Y's lower 8 bits	rr_1rr_2	Signed 16-bit data value (rr_1 is the upper 8-bit value, and rr_2 is the lower 8-bit value)
S	Stack pointer		
PC	Program counter		
PC_H	Program counter's upper 8 bits		
PC_L	Program counter's lower 8 bits		
REL	Relative address		
PG	Program bank register		
DT	Data bank register		

Operation : Acc, C ← Acc + M + C

Description : Adds the contents of the accumulator, memory and carry flag, and places the result in the accumulator.

Executed as binary addition if the decimal operation mode flag D is set to 0.

Executed as decimal addition if the decimal operation mode flag D is set to 1.

Status flags

IPL: Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0. Meaningless for decimal addition.

V : Set to 1 when binary addition of signed data result in a value outside the range of -32768 to +32767 (-128 to +127 if the data length selection flag m is set to 1). Otherwise, cleared to 0. Meaningless for decimal addition.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0. Meaningless for decimal addition.

C : When the data length selection flag m is set to 0, set to 1 if binary addition exceeds +65535 or if decimal addition exceeds +9999. Otherwise, cleared to 0. When the data length selection flag m is set to 1, set to 1 if binary addition exceeds +255 or if decimal addition exceeds +99. Otherwise, cleared to 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	ADC A, #imm	69_{16}, imm	2	2
Direct	ADC A, dd	65_{16}, dd	2	4
Direct indexed X	ADC A, dd, X	75_{16}, dd	2	5
Direct indirect	ADC A, (dd)	72_{16}, dd	2	6
Direct indexed X indirect	ADC A, (dd, X)	61_{16}, dd	2	7
Direct indirect indexed Y	ADC A, (dd), Y	71_{16}, dd	2	8
Direct indirect long	ADCL A, (dd)	67_{16}, dd	2	10
Direct indirect long indexed Y	ADCL A, (dd), Y	77_{16}, dd	2	11
Absolute	ADC A, mmll	$6D_{16}$, ll, mm	3	4
Absolute indexed X	ADC A, mmll, X	$7D_{16}$, ll, mm	3	6
Absolute indexed Y	ADC A, mmll, Y	79_{16}, ll, mm	3	6
Absolute long	ADC A, hhmmll	$6F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	ADC A, hhmmll, X	$7F_{16}$, ll, mm, hh	4	7
Stack pointer relative	ADC A, nn,S	63_{16}, nn	2	5
Stack pointer relative indirect indexed Y	ADC A, (nn, S), Y	73_{16}, nn	2	8

(Note1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

(Note2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Operation : Acc ← Acc ∧ M

Description : Performs logical AND between the contents of the accumulator and the contents of memory, and places the result in the accumulator.

Status flags

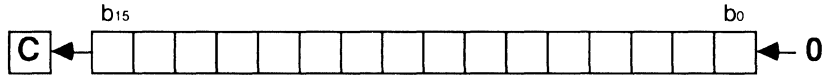
IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	AND A, #imm	29_{16}, imm	2	2
Direct	AND A, dd	25_{16}, dd	2	4
Direct indexed X	AND A, dd, X	35_{16}, dd	2	5
Direct indirect	AND A, (dd)	32_{16}, dd	2	6
Direct indexed X indirect	AND A, (dd, X)	21_{16}, dd	2	7
Direct indirect indexed Y	AND A, (dd), Y	31_{16}, dd	2	8
Direct indirect long	ANDL A, (dd)	27_{16}, dd	2	10
Direct indirect long indexed Y	ANDL A, (dd), Y	37_{16}, dd	2	11
Absolute	AND A, mmll	$2D_{16}$, ll, mm	3	4
Absolute indexed X	AND A, mmll, X	$3D_{16}$, ll, mm	3	6
Absolute indexed Y	AND A, mmll, Y	39_{16}, ll, mm	3	6
Absolute long	AND A, hhmmll	$2F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	AND A, hhmmll, X	$3F_{16}$, ll, mm, hh	4	7
Stack pointer relative	AND A, nn, S	23_{16}, nn	2	5
Stack pointer relative indirect indexed Y	AND A, (nn, S), Y	33_{16}, nn	2	8

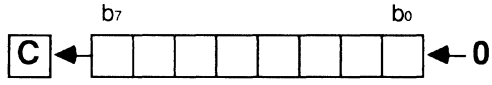
(Note1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

(Note2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Operation : When m=0



When m=1



Description : Shifts all bits of the accumulator or memory one place to the left. Bit 0 is loaded with 0. The carry flag C is loaded from bit 15 (or bit 7 when the data length selection flag m is set to 1) of the data before the shift.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 when bit 15 (or bit 7 when the data length selection flag m is set to1) before the operation is 1. Otherwise, cleared to 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	ASL A	$0A_{16}$	1	2
Direct	ASL dd	06_{16}, dd	2	7
Direct indexed X	ASL dd, X	16_{16}, dd	2	7
Absolute	ASL mmll	$0E_{16}$, ll, mm	3	7
Absolute indexed X	ASL mmll, X	$1E_{16}$, ll, mm	3	8

(Note1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Operation : When M ∧ IMM=0

PC ← PC + n ± REL (REL is instruction's second byte)

PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When M ∧ IMM≠0

PC ← PC + n

PG ← PG + 1 (if carry on PC)

IMM is the bit pattern that specifies the bit positions to be tested.

The value of n is determined as follows:

If the data length selection flag m is set to 1, n=4 if direct bit relative addressing mode, and n=5 if absolute bit relative addressing mode.

If the data length selection flag m is set to 0, n=5 if direct bit relative addressing mode, and n=6 if absolute bit relative addressing mode.

Description : The BBC instruction tests the specified bits (which may be specified simultaneously) of memory. The instruction causes a branch to the specified address when the specified bits are all 0. The branch address is specified by a relative address.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct bit relative	BBC #imm, dd, rr	34_{16}, dd, imm, rr	4	7
Absolute bit relative	BBC #imm, mmll, rr	$3C_{16}$, ll, mm, imm, rr	5	8

(Note1) The bytes-count increases by 1 when operating on 16-bit data with the data length selection flag m set to 0.

(Note2) The cycles-count increases by 2 when a branch occurs.

Operation : When $\overline{M} \wedge IMM=0$

PC ← PC + n ± REL (REL is instruction's second byte)
PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When $\overline{M} \wedge IMM \neq 0$

PC ← PC + n
PG ← PG + 1 (if carry on PC)

IMM is the bit pattern that specifies the bit positions to be tested. The value of n is determined as follows:

If the data length selection flag m is set to 1, n=4 if direct bit relative addressing mode, and n=5 if absolute bit relative addressing mode.

If the data length selection flag m is set to 0, n=5 if direct bit relative addressing mode, and n=6 if absolute bit relative addressing mode.

Description : The BBS instruction tests the specified bits (which may be specified simultaneously) of memory. The instruction causes a branch to the specified address when the specified bits are all 1. The branch address is specified by a relative address.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct bit relative	BBS #imm, dd, rr	24_{16}, dd, imm, rr	4	7
Absolute bit relative	BBS #imm, mmll, rr	$2C_{16}$, ll, mm, imm, rr	5	8

(Note1) The bytes-count increases by 1 when operating on 16-bit data with the data length selection flag m set to 0.

(Note2) The cycles-count increases by 2 when a branch occurs.

Operation : When C=0,

PC ← PC + 2 ± REL (REL is instruction's second byte)
PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When C=1,

PC ← PC + 2
PG ← PG + 1 (if carry on PC)

Description : When the carry flag C is clear (0), the BCC instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the carry flag C is set (1), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BCC rr	90_{16}, rr	2	4

(Note1) The cycles-count increases by 2 when a branch occurs.

Operation : When C=1,

PC ← PC + 2 ± REL (REL is instruction's second byte)
PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When C=0,

PC ← PC + 2
PG ← PG + 1 (if carry on PC)

Description : When the carry flag C is set (1), the BCS instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the carry flag C is clear (0), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BCS rr	$B0_{16}$, rr	2	4

(Note1) The cycles-count increases by 2 when a branch occurs.

Operation : When Z=1,

PC ← PC + 2 ± REL (REL is instruction's second byte)
PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When Z=0,

PC ← PC + 2
PG ← PG + 1 (if carry on PC)

Description : When the zero flag Z is set (1), the BEQ instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the zero flag Z is clear (0), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BEQ rr	$F0_{16}$, rr	2	4

(Note1) The cycles-count increases by 2 when a branch occurs.

Operation : When N=1,

PC ← PC + 2 ± REL (REL is instruction's second byte)
PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When N=0,

PC ← PC + 2
PG ← PG + 1 (if carry on PC)

Description : When the negative flag N is set (1), the BMI instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the negative flag N is clear (0), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BMI rr	30_{16}, rr	2	4

(Note1) The cycles-count increases by 2 when a branch occurs.

Operation : When Z=0,

PC ← PC + 2 ± REL (REL is instruction's second byte)
PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When Z=1,

PC ← PC + 2
PG ← PG + 1 (if carry on PC)

Description : When the zero flag Z is clear (0), the BNE instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the zero flag Z is set (1), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BNE rr	$D0_{16}$, rr	2	4

(Note1) The cycles-count increases by 2 when a branch occurs.

Operation : When N=0,

PC ← PC + 2 ± REL (REL is instruction's second byte)
PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When N=1,

PC ← PC + 2
PG ← PG + 1 (if carry on PC)

Description : When the negative flag N is clear (0), the BPL instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the negative flag N is set (1), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BPL rr	10_{16}, rr	2	4

(Note1) The cycles-count increases by 2 when a branch occurs.

Operation :

For short relative branch,

PC ← PC + 2 ± REL (REL is instruction's second byte)

PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

For long relative branch,

PC ← PC + 3 ± REL (REL is a numeric value represented by the instruction's second and third bytes)

Description : The BRA instruction causes a branch to the specified address. The branch address is specified by a relative address.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BRA rr	80_{16}, rr	2	4
	BRAL rr_1rr_2	82_{16}, rr_2, rr_1	3	4

Operation : PC ← PC + 2
M(S) ← PG
S ← S - 1
M(S) ← PC_H
S ← S - 1
M(S) ← PC_L
S ← S - 1
M(S) ← PS_H
S ← S - 1
M(S) ← PS_L
S ← S - 1
I ← 1
PC_L ← M($FFFA_{16}$)
PC_H ← M($FFFB_{16}$)
PG ← 00_{16}

Description : When the BRK instruction is executed, the CPU first saves the address where the next instruction is stored, and then saves the contents of the processor status register on the stack. Then, the CPU executes a branch to the address in bank-0 the lower portion of which is specified by the contents of $FFFA_{16}$ in bank-0 and the upper portion specified by the contents of $FFFB_{16}$ in bank-0.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Set to 1.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	BRK #nn	00_{16},EA_{16}	2	15

(Note1) The instruction's second byte is ignored, so any value impossible.

Operation : When V=0,

PC ← PC + 2 ± REL (REL is instruction's second byte)

PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When V=1,

PC ← PC + 2

PG ← PG + 1 (if carry on PC)

Description : When the overflow flag V is clear (0), the BVC instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the overflow flag V is set (1), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BVC rr	50_{16}, rr	2	4

(Note 1) The cycles-count increases by 2 when a branch occurs.

Operation : When V=1,

PC ← PC + 2 ± REL (REL is instruction's second byte)
PG ← PG + 1 (if carry on PC), PG ← PG - 1 (if borrow on PC)

When V=0,

PC ← PC + 2
PG ← PG + 1 (if carry on PC)

Description : When the overflow flag V is set (1), the BVS instruction causes a branch to the specified address. The branch address is specified by a relative address.

When the overflow flag V is clear (0), the program advances to next step without any action.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Relative	BVS rr	70_{16}, rr	2	4

(Note1)The cycles-count increases by 2 when a branch occurs.

Operation : $M \leftarrow M \wedge \overline{IMM}$

IMM is the bit pattern that specifies the bit positions that are to be cleared to 0. The bit positions that are to be cleared are indicated by 1 in IMM, and the bit positions that are not to be cleared are indicated by 0 in IMM.

When the data length selection flag m is set to 1, IMM is placed in the third byte (direct bit addressing mode) or the fourth byte (absolute bit addressing mode) of the instruction.

When the data length selection flag m is set to 0, IMM is placed in the third and fourth bytes (direct bit addressing mode) or the fourth and fifth bytes (absolute bit addressing mode) of the instruction.

Description : The CLB instruction clears the specified memory bits to 0. Multiple bits to be cleared can be specified at one time.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct bit	CLB #imm, dd	14_{16}, dd, imm	3	8
Absolute bit	CLB #imm, mmll	$1C_{16}$, ll, mm, imm	4	9

(Note1) The bytes-count increases by 1 when operating on 16-bit data with the data length selection flag m set to 0.

Operation : C ← 0

Description : Clears the contents of carry flag C to 0.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Cleared to 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	CLC	18_{16}	1	2

Operation : I ← 0

Description : Clears the interrupt disable flag I to 0.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Cleared to 0.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	CLI	58_{16}	1	2

Operation : m ← 0

Description : Clears the data length selection flag m to 0.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Cleared to 0.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	CLM	$D8_{16}$	1	2

Operation : $PS_L \leftarrow PS_L \wedge \overline{IMM}$
(IMM is the immediate value. Its specified in the second byte of the instruction.)

Description : Clears the processor status flags specified by the bit pattern in the second byte of the instruction to 0.

Status flags : The specifed flags are cleared. IPL is not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	CLP #imm	$C2_{16}$, imm	2	4

Operation : V ← 0

Description : Clears the overflow flag V to 0.

Status flags

IPL : Not affected.
N : Not affected.
V : Cleared to 0.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	CLV	$B8_{16}$	1	2

Operation : Acc - M

Description : Subtracts the contents of memory from the contents of the accumulator. The accumulator and memory contents are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Set to 1 if the result of operation is 0 or larger. Otherwise, cleared to 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	CMP A, #imm	$C9_{16}$, imm	2	2
Direct	CMP A, dd	$C5_{16}$, dd	2	4
Direct indexed X	CMP A, dd, X	$D5_{16}$, dd	2	5
Direct indirect	CMP A, (dd)	$D2_{16}$, dd	2	6
Direct indexed X indirect	CMP A, (dd, X)	$C1_{16}$, dd	2	7
Direct indirect indexed Y	CMP A, (dd), Y	$D1_{16}$, dd	2	8
Direct indirect long	CMPL A, (dd)	$C7_{16}$, dd	2	10
Direct indirect long indexed Y	CMPL A, (dd), Y	$D7_{16}$, dd	2	11
Absolute	CMP A, mmll	CD_{16}, ll, mm	3	4
Absolute indexed X	CMP A, mmll, X	DD_{16}, ll, mm	3	6
Absolute indexed Y	CMP A, mmll, Y	$D9_{16}$, ll, mm	3	6
Absolute long	CMP A, hhmmll	CF_{16}, ll, mm, hh	4	6
Absolute long indexed X	CMP A, hhmmll, X	DF_{16}, ll, mm, hh	4	7
Stack pointer relative	CMP A, nn, S	$C3_{16}$, nn	2	5
Stack pointer relative indirect indexed Y	CMP A, (nn, S), Y	$D3_{16}$, nn	2	8

(Note1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.
(Note2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Operation : X - M

Description : Subtracts the contents of memory from the contents of the index register X. The index register X and memory contents are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 if the result of operation is 0 or larger. Otherwise, cleared to 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	CPX #imm	$E0_{16}$, imm	2	2
Direct	CPX dd	$E4_{16}$, dd	2	4
Absolute	CPX mmll	EC_{16}, ll, mm	3	4

(Note1) When operating on 16-bit data in the immediate addressing mode with the index register length selection flag x set to 0, the bytes-count increases by 1.

Operation : Y - M

Description : Subtracts the contents of memory from the contents of the index register Y. The index register Y and memory contents are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Set to 1 if the result of operation is 0 or larger. Otherwise, cleared to 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	CPY #imm	$C0_{16}$, imm	2	2
Direct	CPY dd	$C4_{16}$,dd	2	4
Absolute	CPY mmll	CC_{16}, ll, mm	3	4

(Note1) When operating on 16-bit data in the immediate addressing mode with the index register length selection flag x set to 0, the bytes-count increases by 1.

Operation : Acc ← Acc - 1 or M ← M -1

Description : Subtracts 1 from the contents of the accumulator or memory.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	DEC A	$1A_{16}$	1	2
Direct	DEC dd	$C6_{16}$, dd	2	7
Direct indexed X	DEC dd, X	$D6_{16}$, dd	2	7
Absolute	DEC mmll	CE_{16}, ll, mm	3	7
Absolute indexed X	DEC mmll, X	DE_{16}, ll, mm	3	8

(Note1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Operation : X ← X - 1

Description : Subtracts 1 from the contents of the index register X.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	DEX	CA_{16}	1	2

Operation : Y ← Y - 1

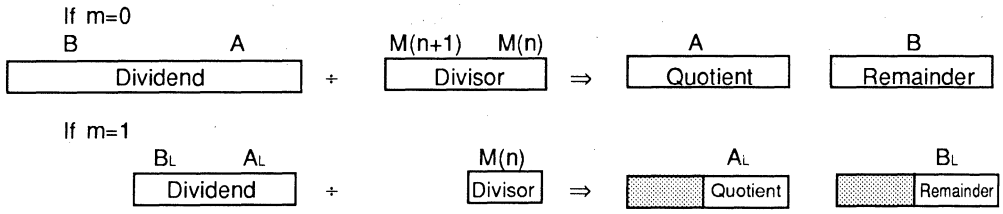
Description : Subtracts 1 from the contents of the index register Y.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	DEY	88_{16}	1	2

Operation : B(remainder), A(quotient) ← (B, A) / M



Description : When the data length selection flag m is set to 0, a 32-bit data stored in the accumulators B (upper 16 bits) and A (lower 16 bits) are divided by a 16-bit data in memory. The quotient is placed in the accumulator A, and the remainder is placed in the accumulator B.

When the data length selection flag m is set to 1, a 16-bit data stored in the lower 8 bits of the accumulators B (upper 8 bits) and A (lower 8 bits) are divided by an 8 bit data in memory. The quotient is placed in the lower 8 bits of the accumulator A, and the remainder is placed in the lower 8 bits of the accumulator B.

When an overflow results from this operation negrect removed out, the V flag is set.

When divisor is 0, the zero division interrupt is generated, in which case the contents of the processor status register are saved on the stack and a branch occurs to the address in bank-0 as specified by the zero division interrupt vector. Accumulator contents are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of quotient from the operation is 1. Otherwise, cleared to 0.

V : Set to 1 when the quotient from the operation exceeds 16 bits (or 8 bits if the data length selection flag m is set to 1) (i.e., an overflow has occurred). Otherwise, cleared to 0. No changes occur when divisor is 0.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the quotient from the operation is 0. Otherwise, cleared to 0. No changes occur when divisor is 0.

C : Set to 1 when the quotient from the operation exceeds 16 bits (or 8 bits if the data length selection flag m is set to 1) (i.e., an overflow has occurred). Otherwise, cleared to 0. No changes occur when divisor is 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	DIV #imm	89_{16}, 29_{16}, imm	3	27
Direct	DIV dd	89_{16}, 25_{16}, dd	3	29
Direct indexed X	DIV dd, X	89_{16}, 35_{16}, dd	3	30
Direct indirect	DIV (dd)	89_{16}, 32_{16}, dd	3	31
Direct indexed X indirect	DIV (dd, X)	89_{16}, 21_{16}, dd	3	32
Direct indirect indexed Y	DIV (dd), Y	89_{16}, 31_{16}, dd	3	33
Direct indirect long	DIVL (dd)	89_{16}, 27_{16}, dd	3	35
Direct indirect long indexed Y	DIVL (dd), Y	89_{16}, 37_{16}, dd	3	36
Absolute	DIV mmll	89_{16}, $2D_{16}$, ll, mm	4	29
Absolute indexed X	DIV mmll, X	89_{16}, $3D_{16}$, ll ,mm	4	31
Absolute indexed Y	DIV mmll, Y	89_{16}, 39_{16}, ll ,mm	4	31
Absolute long	DIV hhmmll	89_{16}, $2F_{16}$, ll, mm, hh	5	31
Absolute long indexed X	DIV hhmmll, X	89_{16}, $3F_{16}$, ll, mm, hh	5	32
Stack pointer relative	DIV nn, S	89_{16}, 23_{16}, nn	3	30
Stack pointer relative indirect indexed Y	DIV (nn, S), Y	89_{16}, 33_{16}, nn	3	33

(Note1) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

(Note2) The cycles-count in this table are for 16-bit ÷ 8-bit operations. For 32-bit ÷ 16-bit operations, the cycles-count increases by 16.

Operation : Acc ← Acc ∀ M

Description : Performs the logical EXCLUSIVE OR between the contents of the accumulator and the contents of memory, and places the result in the accumulator.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	EOR A, #imm	49_{16}, imm	2	2
Direct	EOR A, dd	45_{16}, dd	2	4
Direct indexed X	EOR A, dd, X	55_{16}, dd	2	5
Direct indirect	EOR A, (dd)	52_{16}, dd	2	6
Direct indexed X indirect	EOR A, (dd, X)	41_{16}, dd	2	7
Direct indirect indexed Y	EOR A, (dd), Y	51_{16}, dd	2	8
Direct indirect long	EORL A, (dd)	47_{16}, dd	2	10
Direct indirect long indexed Y	EORL A, (dd), Y	57_{16}, dd	2	11
Absolute	EOR A, mmll	$4D_{16}$, ll, mm	3	4
Absolute indexed X	EOR A, mmll, X	$5D_{16}$, ll, mm	3	6
Absolute indexed Y	EOR A, mmll, Y	59_{16}, ll, mm	3	6
Absolute long	EOR A, hhmmll	$4F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	EOR A, hhmmll, X	$5F_{16}$, ll, mm, hh	4	7
Stack pointer relative	EOR A, nn, S	43_{16}, nn	2	5
Stack pointer relative indirect indexed Y	EOR A, (nn, S), Y	53_{16}, nn	2	8

(Note1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.
(Note2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Operation : Acc ← Acc + 1 or M ← M + 1

Description : Adds 1 to the contents of the accumulator or memory.

Status flags

IPL : Not affcted.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	INC A	$3A_{16}$	1	2
Direct	INC dd	$E6_{16}$, dd	2	7
Direct indexed X	INC dd, X	$F6_{16}$, dd	2	7
Absolute	INC mmll	EE_{16}, ll, mm	3	7
Absolute indexed X	INC mmll, X	FE_{16}, ll, mm	3	8

(Note1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Operation : $X \leftarrow X + 1$

Description : Adds 1 to the contents of the index register X.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	INX	$E8_{16}$	1	2

Operation : Y ← Y + 1

Description : Adds 1 to the contents of the index register Y.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	INY	$C8_{16}$	1	2

Operation : If absolute addressing mode,

$PC_L \leftarrow AD_L$

$PC_H \leftarrow AD_H$

If absolute long addressing mode,

$PC_L \leftarrow AD_L$

$PC_H \leftarrow AD_H$

$PG \leftarrow AD_G$

If absolute indirect addressing mode,

$PC_L \leftarrow (AD_H, AD_L)$

$PC_H \leftarrow (AD_H, AD_L + 1)$

If absolute indirect long addressing mode,

$PC_L \leftarrow (AD_H, AD_L)$

$PC_H \leftarrow (AD_H, AD_L + 1)$

$PG \leftarrow (AD_H, AD_L + 2)$

If absolute indexed X indirect addressing mode,

$PC_L \leftarrow (AD_H, AD_L + X)$

$PC_H \leftarrow (AD_H, AD_L + X + 1)$

(AD_L, AD_H and AD_G specify the instruction's second, third and fourth bytes, respectively.)

Description : The JMP instruction causes a jump to the address specified for the addressing mode in use.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Absolute	JMP mmll	$4C_{16}$, ll, mm	3	2
Absolute long	JMPL hhmmll	$5C_{16}$, ll, mm, hh	4	4
Absolute indirect	JMP (mmll)	$6C_{16}$, ll, mm	3	4
Absolute indirect long	JMPL (mmll)	DC_{16}, ll, mm	3	8
Absolute indexed X indirect	JMP (mmll, X)	$7C_{16}$, ll, mm	3	6

Operation : If absolute addressing mode,

$M(S) \leftarrow PC_H$
$S \leftarrow S - 1$
$M(S) \leftarrow PC_L$
$S \leftarrow S - 1$
$PC_L \leftarrow AD_L$
$PC_H \leftarrow AD_H$

If absolute long addressing mode,

$M(S) \leftarrow PG$
$S \leftarrow S - 1$
$M(S) \leftarrow PC_H$
$S \leftarrow S - 1$
$M(S) \leftarrow PC_L$
$S \leftarrow S - 1$
$PC_L \leftarrow AD_L$
$PC_H \leftarrow AD_H$
$PG \leftarrow AD_G$

If absolute indexed X indirect addressing mode,

$M(S) \leftarrow PC_H$
$S \leftarrow S - 1$
$M(S) \leftarrow PC_L$
$S \leftarrow S - 1$
$PC_L \leftarrow (AD_H, AD_L + X)$
$PC_H \leftarrow (AD_H, AD_L + X + 1)$

(AD_L, AD_H and AD_G specify the instruction's second, third and fourth bytes, respectively.)

Description : The contents of the program counter PC (or the program bank register PG and the program counter PC if absolute long addressing mode) are first saved on the stack, then a jump occurs to the address shown for each addressing mode.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Absolute	JSR mmll	20_{16}, ll, mm	3	6
Absolute long	JSRL hhmmll	22_{16}, ll, mm, hh	4	8
Absolute indexed X indirect	JSR (mmll, X)	FC_{16}, ll, mm	3	8

Operation : Acc ← M

Description : Loads the contents of memory into the accumulator.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	LDA A, #imm	$A9_{16}$, imm	2	2
Direct	LDA A, dd	$A5_{16}$, dd	2	4
Direct indexed X	LDA A, dd, X	$B5_{16}$, dd	2	5
Direct indirect	LDA A, (dd)	$B2_{16}$, dd	2	6
Direct indexed X indirect	LDA A, (dd, X)	$A1_{16}$, dd	2	7
Direct indirect indexed Y	LDA A, (dd), Y	$B1_{16}$, dd	2	8
Direct indirect long	LDAL A, (dd)	$A7_{16}$, dd	2	10
Direct indirect long indexed Y	LDAL A, (dd), Y	$B7_{16}$, dd	2	11
Absolute	LDA A, mmll	AD_{16}, ll, mm	3	4
Absolute indexed X	LDA A, mmll, X	BD_{16}, ll, mm	3	6
Absolute indexed Y	LDA A, mmll, Y	$B9_{16}$, ll, mm	3	6
Absolute long	LDA A, hhmmll	AF_{16}, ll, mm, hh	4	6
Absolute long indexed X	LDA A, hhmmll, X	BF_{16}, ll, mm, hh	4	7
Stack pointer relative	LDA A, nn, S	$A3_{16}$, nn	2	5
Stack pointer relative indirect indexed Y	LDA A, (nn, S), Y	$B3_{16}$, nn	2	8

(Note1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

(Note2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Operation : M ← IMM (IMM is an immediate value)

Description : Loads an immediate value into memory.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct	LDM #imm, dd	64_{16}, dd, imm	3	4
Direct indexed X	LDM #imm, dd, X	74_{16}, dd, imm	3	5
Absolute	LDM #imm, mmll	$9C_{16}$, ll, mm, imm	4	5
Absolute indexed X	LDM #imm, mmll, X	$9E_{16}$, ll, mm, imm	4	6

(Note1) When operating on 16-bit data with the data length selection flag m set to 0, the bytes-count increases by 1.

Operation : DT ← IMM (IMM is an immediate value)

Description : Loads an immediate value into the data bank register DT.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	LDT #imm	89_{16}, $C2_{16}$, imm	3	5

Operation : X ← M

Description : Loads the contents of memory into the index register X.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	LDX #imm	$A2_{16}$, imm	2	2
Direct	LDX dd	$A6_{16}$, dd	2	4
Direct indexed Y	LDX dd, Y	$B6_{16}$, dd	2	5
Absolute	LDX mmll	AE_{16}, ll, mm	3	4
Absolute indexed Y	LDX mmll, Y	BE_{16}, ll, mm	3	6

(Note1) When operating on 16-bit data in the immediate addressing mode with the index register length selection flag x set to 0, the bytes-count increases by 1.

Operation : Y ← M

Description : Loads the contents of memory into the index register Y.

Status flags

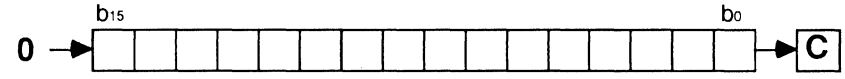
IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	LDY #imm	$A0_{16}$, imm	2	2
Direct	LDY dd	$A4_{16}$, dd	2	4
Direct indexed X	LDY dd, X	$B4_{16}$, dd	2	5
Absolute	LDY mmll	AC_{16}, ll, mm	3	4
Absolute indexed X	LDY mmll, X	BC_{16}, ll, mm	3	6

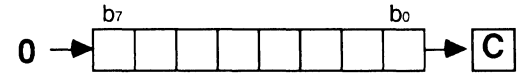
(Note1) When operating on 16-bit data in the immediate addressing mode with the index register length selection flag x set to 0, the bytes-count increases by 1.

Operation :

When m=0



When m=1



Description : Shifts all bits of the accumulator or memory one place to the right. Bit 15 (or bit 7 if the data length selection flag m is set to 1) of the accumulator or memory is loaded with 0.

The carry flag C is loaded from bit 0 of the data before the shift.

Status flags

IPL : Not affected.
N : Cleared to "0".
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Set to 1 when bit 0 before the operation is 1. Otherwise, cleared to 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	LSR A	$4A_{16}$	1	2
Direct	LSR dd	46_{16}, dd	2	7
Direct indexed X	LSR dd, X	56_{16}, dd	2	7
Absolute	LSR mmll	$4E_{16}$, ll, mm	3	7
Absolute indexed X	LSR mmll, X	$5E_{16}$, ll, mm	3	8

(Note1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Operation : B, A ← A × M

Description : When the data length selection flag m is set to 0, The contents of the accumulator A and the contents of memory are multiplied. Multiplication is performed as 16-bit × 16-bit, and the result is a 32-bit data which is placed in the accumulators B (upper 16 bits of the result) and A (lower 16 bits of the result).

When the data length selection flag m is set to 1, the lower 8-bit contents of the accumulator A and the contents of memory are multiplied. Multiplication is performed as 8-bit × 8-bit, and the result is a 16-bit data which is placed in the lower 8 bits of the accumulators B (upper 8 bits of the result) and A (lower 8 bits of the result).

Status flags

IPL : Not affected.

N : Set to 1 when bit 31 (or bit 15 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	MPY #imm	89_{16}, 09_{16}, imm	3	16
Direct	MPY dd	89_{16}, 05_{16}, dd	3	18
Direct indexed X	MPY dd, X	89_{16}, 15_{16}, dd	3	19
Direct indirect	MPY (dd)	89_{16}, 12_{16}, dd	3	20
Direct indexed X indirect	MPY (dd, X)	89_{16}, 01_{16}, dd	3	21
Direct indirect indexed Y	MPY (dd), Y	89_{16}, 11_{16}, dd	3	22
Direct indirect long	MPYL (dd)	89_{16}, 07_{16}, dd	3	24
Direct indirect long indexed Y	MPYL (dd), Y	89_{16}, 17_{16}, dd	3	25
Absolute	MPY mmll	89_{16}, $0D_{16}$, ll, mm	4	18
Absolute indexed X	MPY mmll, X	89_{16}, $1D_{16}$, ll, mm	4	20
Absolute indexed Y	MPY mmll, Y	89_{16}, 19_{16}, ll, mm	4	20
Absolute long	MPY hhmmll	89_{16}, $0F_{16}$, ll, mm, hh	5	20
Absolute long indexed X	MPY hhmmll, X	89_{16}, $1F_{16}$, ll, mm, hh	5	21
Stack pointer relative	MPY nn, S	89_{16}, 03_{16}, nn	3	19
Stack pointer relative indirect indexed Y	MPY (nn, S), Y	89_{16}, 13_{16}, nn	3	22

(Note1) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

(Note2) The cycles-count in this table are for 8-bit × 8-bit multiplications. For 16-bit × 16-bit multiplications, the cycles-count increases by 8.

Operation : $M_n \sim M_{n+k} \leftarrow M_m \sim M_{m+k}$

Description : Normally, a block of data is transferred from upper addresses to lower addresses. The transfer is performed in the ascending address order of the block being transferred. The target bank is specified by the instruction's second byte, and the address within the target bank is specified by the contents of the index register Y. The source bank is specified by the instruction's third byte, and the address within the source bank is specified by the contents of the index register X. The accumulator A is loaded with the bytes-count of the data to be transferred. As each byte of data is transferred, the index registers X and Y are incremented by 1, so that the index register X will become a value equal to 1 larger than the source address of the last byte transferred and the index register Y will become a value equal to 1 larger than the target address of the last byte received. The data bank register DT will become the terget bank number, and the accumulator A will become $FFFF_{16}$.

The accumulator A is affected by flag m. The index register X and Y are affected by flag x.

When the contents of the accumulator A is "00_{16}", the data are not transferred.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Block transfer	MVN n_1, n_2	54_{16}, n_1, n_2	3	7+(i/2)×7

(Note1) The cycles-count shown above is for when the number of bytes transferred, i, is an even number. If i is an odd number, the cycles-count is obtained as follows:

$7 + (i \div 2) \times 7 + 4$.

Note that $(i \div 2)$ denotes the integer part of the result of dividing i by 2.

Operation : $M_{n-k} \sim M_n \leftarrow M_{m-k} \sim M_m$

Description : Normally, a block of data is transferred from lower addresses to upper addresses. The transfer is performed in the descending address order of the block being transferred. The target bank is specified by the instruction's second byte, and the address within the target bank is specified by the contents of the index register Y. The source bank is specified by the instruction's third byte, and the address within the source bank is specified by the contents of the index register X. The accumulator A is loaded with the bytes-count of the data to be transferred. As each byte of data is transferred, the index registers X and Y are decremented by 1, so that the index register X will become a value equal to 1 less than the source address of the last byte transferred and the index register Y will become a value equal to 1 smaller than the target address of the last byte received. The data bank register DT will become the target bank number, and the accumulator A will become $FFFF_{16}$.

The accumulator A is affected by flag m. The index register X and Y are affected by flag x.

When the contents of the accumulator A is "00_{16}", the data are not transferred.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Block transfer	MVP n_1, n_2	44_{16}, n_1, n_2	3	9+(i/2)×7

(Note1) The cycles-count shown above is for when the number of bytes transferred, i, is an even number. If i is an odd number, the cycles-count is obtained as follows:
$9 + (i \div 2) \times 7 + 5$.
Note that $(i \div 2)$ denotes the integer part of the result of dividing i by 2.

Operation : PC ← PC + 1
PG ← PG + 1 (if carry on PC)

Description : This instruction only causes the program counter to be incremented by 1 and nothing else.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	NOP	EA_{16}	1	2

Operation : Acc ← Acc V M

Description : Performs the logical OR between the contents of the accumulator and the contents of memory, and places the result in the accumulator.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	ORA A, #imm	09_{16}, imm	2	2
Direct	ORA A, dd	05_{16}, dd	2	4
Direct indexed X	ORA A, dd, X	15_{16}, dd	2	5
Direct indirect	ORA A, (dd)	12_{16}, dd	2	6
Direct indexed X indirect	ORA A, (dd, X)	01_{16}, dd	2	7
Direct indirect indexed Y	ORA A, (dd), Y	11_{16}, dd	2	8
Direct indirect long	ORAL A, (dd)	07_{16}, dd	2	10
Direct indirect long indexed Y	ORAL A, (dd), Y	17_{16}, dd	2	11
Absolute	ORA A, mmll	$0D_{16}$, ll, mm	3	4
Absolute indexed X	ORA A, mmll, X	$1D_{16}$, ll, mm	3	6
Absolute indexed Y	ORA A, mmll, Y	19_{16}, ll, mm	3	6
Absolute long	ORA A, hhmmll	$0F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	ORA A, hhmmll, X	$1F_{16}$, ll, mm, hh	4	7
Stack pointer relative	ORA A, nn, S	03_{16}, nn	2	5
Stack pointer relative indirect indexed Y	ORA A, (nn, S), Y	13_{16}, nn	2	8

(Note1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

(Note2) When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Operation : $M(S) \leftarrow IMM_2$ (IMM_2 is the immediate value specified by the instruction's third byte)
$S \leftarrow S - 1$
$M(S) \leftarrow IMM_1$ (IMM_1 is the immediate value specified by the instruction's second byte)
$S \leftarrow S - 1$

Description : The instruction's third and second bytes are saved on the stack in this order.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PEA #imm1imm2	$F4_{16}$, imm2, imm1	3	5

Operation : M(S) ← M (DPR + IMM + 1)
S ← S - 1
M(S) ← M (DPR + IMM)
S ← S - 1

DPR represents the contents of the direct page register, and IMM represents the offset address within the direct page as specified by the instruction's second byte.

Description : Saves the contents of the consecutive 2 bytes in the direct page as specified by the sum of the contents of the direct page register DPR and the instruction's second byte on the stack in the order of upper address first and lower address second.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PEI #imm	$D4_{16}$, imm	2	5

Operation : EAR ← PC + IMM_2, IMM_1
M(S) ← Upper byte of EAR
S ← S - 1
M(S) ← Lower byte of EAR
S ← S - 1

EAR represents the value obtained by adding the 16-bit data represented by "IMM_2, IMM_1" and the contents of the program counter. IMM_2 and IMM_1 represent the instruction's third and second bytes, respectively, and "IMM_2, IMM_1" represents a 16-bit data with IMM_2 being the upper byte and IMM_1 being the lower byte.

Description : Saves the result of adding a 16-bit data consisting of an upper byte specified by the instruction's third byte and a lower byte specified by the instruction's second byte with the contents of the program counter on the stack in the order of the result's upper byte first and lower byte second.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PER #imm_1imm_2	62_{16}, imm_2, imm_1	3	5

Operation : If m=0,

$M(S) \leftarrow A_H$

$S \leftarrow S - 1$

$M(S) \leftarrow A_L$

$S \leftarrow S - 1$

If m=1,

$M(S) \leftarrow A_L$

$S \leftarrow S - 1$

Description : Saves the contents of the accumulator A to the address specified by the stack pointer S. When the data length selection flag m is set to 0, the accumulator A's upper byte is saved on the stack first and then the lower byte. When the data length selection flag m is set to 1, only the accumulator A's lower byte is saved on the stack.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHA	48_{16}	1	4

Operation : If m=0,

$M(S) \leftarrow B_H$

$S \leftarrow S - 1$

$M(S) \leftarrow B_L$

$S \leftarrow S - 1$

If m=1,

$M(S) \leftarrow B_L$

$S \leftarrow S - 1$

Description : Saves the contents of the accumulator B to the address indicated by the stack pointer S. When the data length selection flag m is set to 0, the accumulator B's upper byte is saved on the stack first and then the lower byte. When the data length selection flag m is set to 1, only the accumulator B's lower byte is saved on the stack.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHB	42_{16}, 48_{16}	2	6

Operation : $M(S) \leftarrow DPR_H$
$S \leftarrow S - 1$
$M(S) \leftarrow DPR_L$
$S \leftarrow S - 1$

Description : Saves the contents of the direct page register DPR to the address indicated by the stack pointer S in the order of upper byte first and then lower byte.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHD	$0B_{16}$	1	4

Operation : M(S) ← PG
S ← S - 1

Description : Saves the contents of the program bank register to the address indicated by the stack pointer S.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHG	$4B_{16}$	1	3

Operation : $M(S) \leftarrow PS_H$
$S \leftarrow S - 1$
$M(S) \leftarrow PS_L$
$S \leftarrow S - 1$

Description : Saves the contents of the processor status register PS to the address indicated by the stack pointer S in the order of upper byte and then lower byte.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHP	08_{16}	1	4

Operation : M(S) ← DT
S ← S - 1

Description : Saves the contents of the data bank register DT to the address indicated by the stack pointer S.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHT	$8B_{16}$	1	3

Operation : If x=0,
$M(S) \leftarrow X_H$
$S \leftarrow S - 1$
$M(S) \leftarrow X_L$
$S \leftarrow S - 1$

If x=1,
$M(S) \leftarrow X_L$
$S \leftarrow S - 1$

Description : Saves the contents of the index register X to the address indicated by the stack pointer S. When the index register length selection flag x is set to 0, the contents are saved in the order of upper byte and then lower byte. When the index register length selection flag x is set to 1, only the lower byte is saved on the stack.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHX	DA_{16}	1	4

Operation : If x=0,

$M(S) \leftarrow Y_H$
$S \leftarrow S - 1$
$M(S) \leftarrow Y_L$
$S \leftarrow S - 1$

If x=1,

$M(S) \leftarrow Y_L$
$S \leftarrow S - 1$

Description : Saves the contents of the index register Y to the address indicated by the stack pointer S. When the index register length selection flag x is set to 0, the contents are saved in the order of upper byte and then lower byte. When the index register length selection flag x is set to 1, only the lower byte is saved on the stack.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PHY	$5A_{16}$	1	4

Operation : If m=0,
$S \leftarrow S + 1$
$A_L \leftarrow M(S)$
$S \leftarrow S + 1$
$A_H \leftarrow M(S)$

If m=1,
$S \leftarrow S + 1$
$A_L \leftarrow M(S)$

Description : The stack pointer S is incremented, and then restores the lower byte of the accumulator A with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the accumulator A with the data at the address indicated by the stack pointer S. When the data length selection flag m is set to 0, 2 bytes data are restored. When the data length selection flag m is set to 1, only 1 byte data is restored (to the lower byte of the accumulator A).

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLA	68_{16}	1	5

Operation : If m=0,
$S \leftarrow S + 1$
$B_L \leftarrow M(S)$
$S \leftarrow S + 1$
$B_H \leftarrow M(S)$

If m=1,
$S \leftarrow S + 1$
$B_L \leftarrow M(S)$

Description : The stack pointer S is incremented, and then restores the lower byte of the accumulator B with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the accumulator B with the data at the address indicated by the stack pointer S. When the data length selection flag m is set to 0, 2 bytes data are restored. When the data length selection flag m is set to 1, only 1 byte data is restored (to the lower byte of the accumulator B).

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1)of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLB	42_{16}, 68_{16}	2	7

Operation : $S \leftarrow S + 1$
$DPR_L \leftarrow M(S)$
$S \leftarrow S + 1$
$DPR_H \leftarrow M(S)$

Description : The stack pointer S is incremented, and then restores the lower byte of the direct page register DPR with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the direct page register DPR with the data at the address indicated by the stack pointer S.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLD	$2B_{16}$	1	5

Operation : $S \leftarrow S + 1$
$PS_L \leftarrow M(S)$
$S \leftarrow S + 1$
$PS_H \leftarrow M(S)$

Description : The stack pointer S is incremented and then restores the lower byte of the processor status register PS with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the processor status register PS with the data at the address indicated by the stack pointer S.

Status flags : Changes to the values restored from the stack.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLP	28_{16}	1	6

Operation : S ← S + 1
DT ← M(S)

Description : The stack pointer S is incremented, and then the data bank register DT is restored with the data at the address indicated by the stack pointer S.

Status flags

IPL : Not affected.
N : Set to 1 when bit 7 of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLT	AB_{16}	1	6

Operation : If x=0,
$S \leftarrow S + 1$
$X_L \leftarrow M(S)$
$S \leftarrow S + 1$
$X_H \leftarrow M(S)$

If x=1,
$S \leftarrow S + 1$
$X_L \leftarrow M(S)$

Description : The stack pointer S is incremented, and then restores the lower byte of the index register X with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the index register X with the data at the address indicated by the stack pointer S. When the index register length selection flag x is set to 0, 2 bytes are restored. When the index register length selection flag x is set to 1, only 1 byte is restored (to the lower byte of the index register X).

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLX	FA_{16}	1	5

Operation : If x=0,
$S \leftarrow S + 1$
$Y_L \leftarrow M(S)$
$S \leftarrow S + 1$
$Y_H \leftarrow M(S)$

If x=1,
$S \leftarrow S + 1$
$Y_L \leftarrow M(S)$

Description : The stack pointer S is incremented, and then restores the lower byte of the index register Y with the data at the address indicated by the stack pointer S. Again, increments the stack pointer S and then restores the upper byte of the index register Y with the data at the address indicated by the stack pointer S. When the index register length selection flag x is set to 0, 2 bytes are restored. When the index register length selection flag x is set to 1, only 1 byte is restored (to the lower byte of the index register Y).

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PLY	$7A_{16}$	1	5

Operation : M(S) ← A, B, X, Y, DPR, DT, PG or PS

Description : This instruction's second byte specifies the registers to be saved. The registers corresponding to the bits in the second byte that are 1 are saved on the stack. The bit and register correspondence is as shown below:

b_7							b_0
PS	PG	DT	DPR	Y	X	B	A

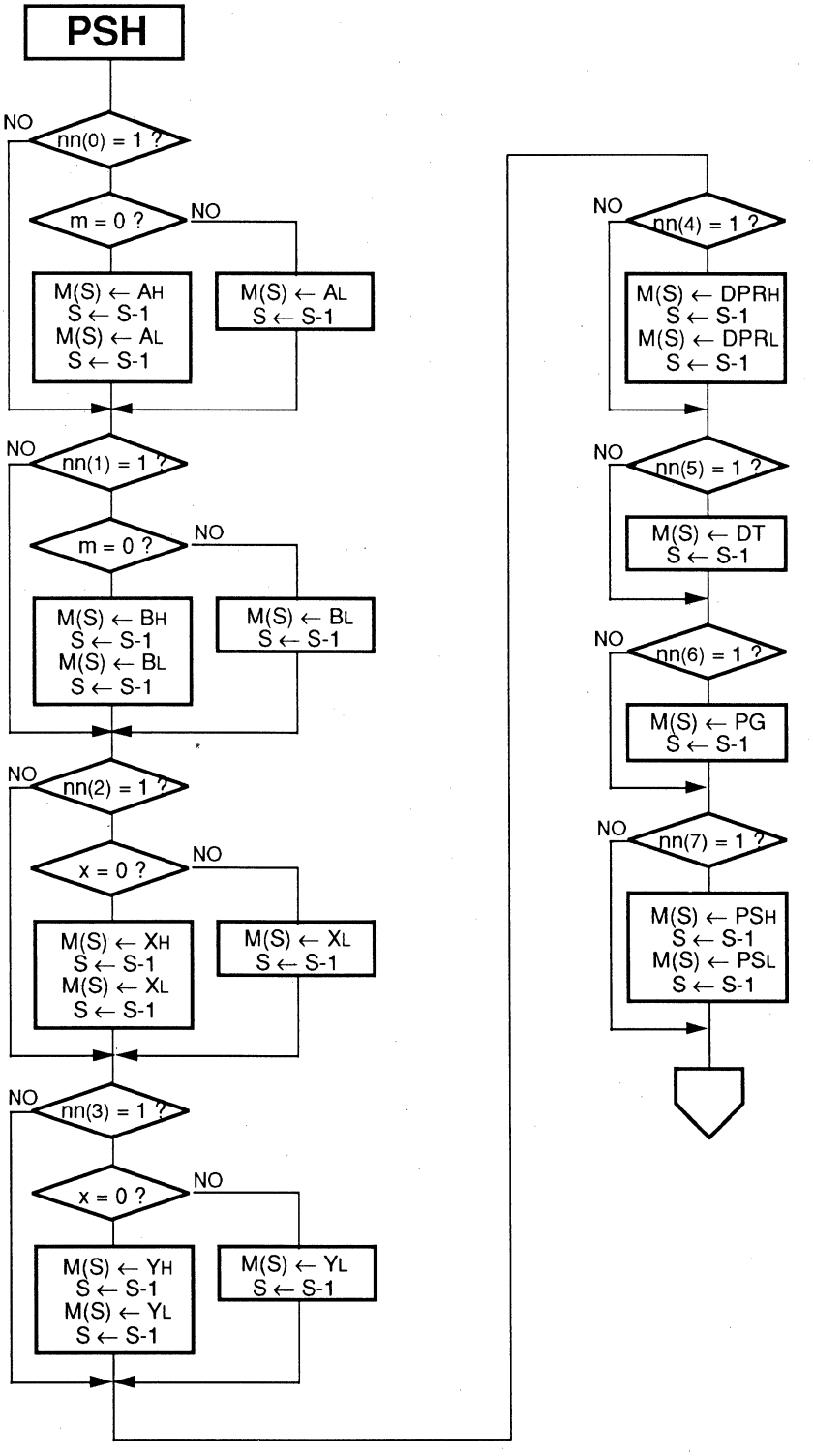
← Saved on the stack in this order.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PSH #nn	EB_{16}, nn	2	$12+2xi_1+i_2$

(Note1) To the cycles-count shown above, the values shown below are added depending on the registers being saved. The count is 12 cycles when no registers are saved. i_1 in above table represents the number of registers (chosen from A, B, X, Y, DPR and PS) to be saved, and i_2 represents the number of registers (chosen from DT and PG) to be saved.

Register type	PS	PG	DT	DPR	Y	X	B	A
Cycles-count	2	1	1	2	2	2	2	2


PSH
NO
nn(0) = 1 ?
m = 0 ?
NO
M(S) ← AH
S ← S-1
M(S) ← AL
S ← S-1
M(S) ← AL
S ← S-1
NO
nn(1) = 1 ?
m = 0 ?
NO
M(S) ← BH
S ← S-1
M(S) ← BL
S ← S-1
M(S) ← BL
S ← S-1
NO
nn(2) = 1 ?
x = 0 ?
NO
M(S) ← XH
S ← S-1
M(S) ← XL
S ← S-1
M(S) ← XL
S ← S-1
NO
nn(3) = 1 ?
x = 0 ?
NO
M(S) ← YH
S ← S-1
M(S) ← YL
S ← S-1
M(S) ← YL
S ← S-1
NO
nn(4) = 1 ?
M(S) ← DPRH
S ← S-1
M(S) ← DPRL
S ← S-1
NO
nn(5) = 1 ?
M(S) ← DT
S ← S-1
NO
nn(6) = 1 ?
M(S) ← PG
S ← S-1
NO
nn(7) = 1 ?
M(S) ← PSH
S ← S-1
M(S) ← PSL
S ← S-1

Operation : M(S) → A, B, X, Y, DPR, DT or PS

Description : This instruction's second byte specifies the registers to be restored. The registers corresponding to the bits in the second byte that are 1 are restored from the stack. The bit and register correspondence is as shown below:

b_7							b_0
PS		DT	DPR	Y	X	B	A

Restored from the stack in this order. →

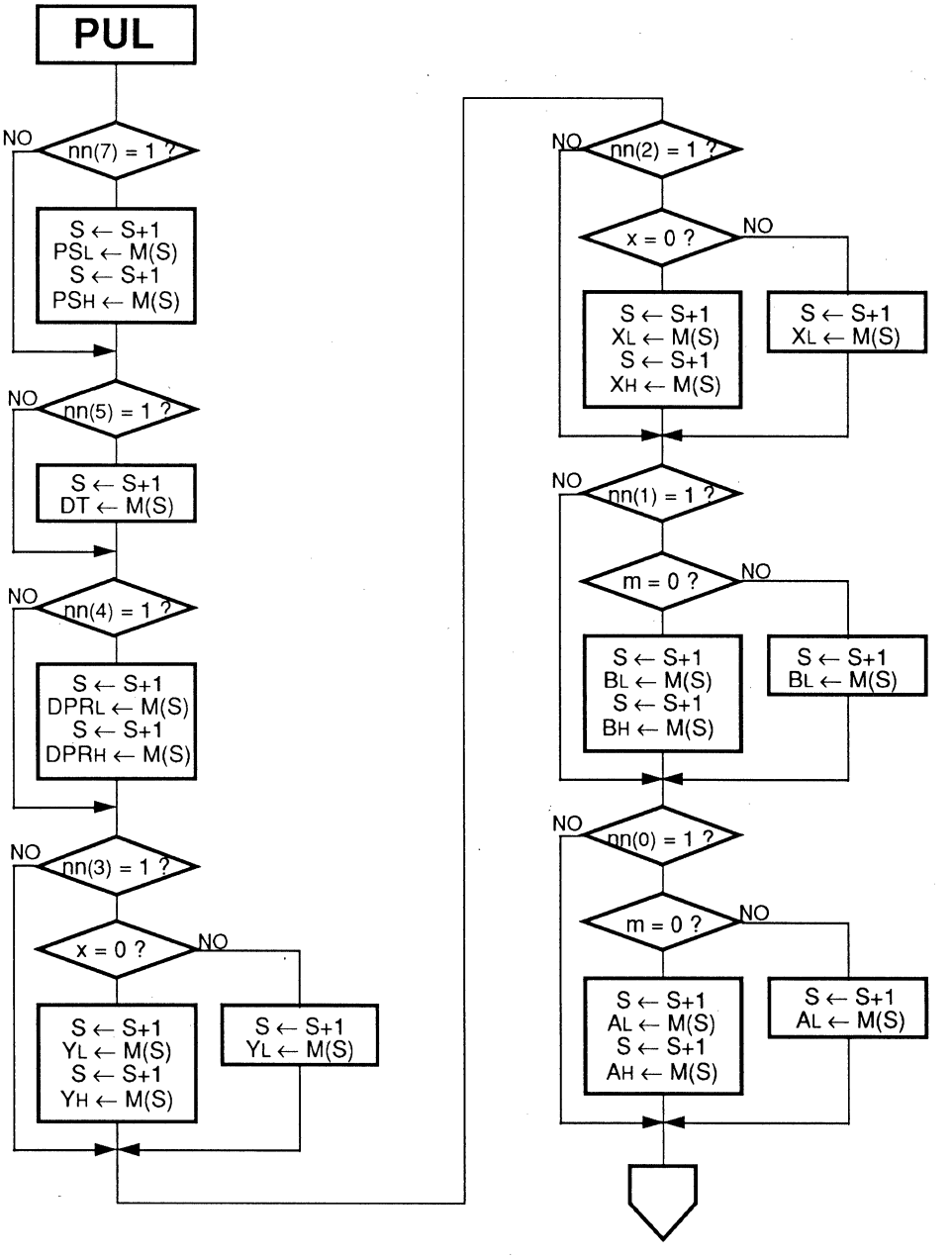
(Note) The contents of accumulator B's higher 8-bit will be changed, when PUL instruction is executed with m=0 and the restored registor including PS whose m=1.

Status flags : When bit 7 of the instruction's second byte is 1, specifying that the program status register PS is to be restored, the status flags are restored to the values that had been restored from the stack. Otherwise, the status flags are not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Stack	PUL #nn	FB_{16}, nn	2	$14+3xi_1+4xi_2$

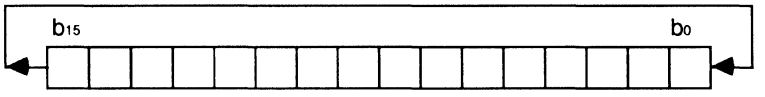
(Note1) To the cycles-count shown above, the values shown below are added depending on the registers being restored. The count is 14 cycles when no registers are restored. i_1 in above table represents the number of registers (chosen from A, B, X, Y, PS and DT) to be saved. i_2=1 if DPR is to be restored, and i_2=0 if DPR is not to be restored.

Register type	PS	DT	DPR	Y	X	B	A
Cycles-count	3	3	4	3	3	3	3

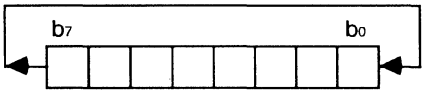

PUL
NO
nn(7) = 1 ?
S ← S+1
PSL ← M(S)
S ← S+1
PSH ← M(S)
NO
nn(5) = 1 ?
S ← S+1
DT ← M(S)
NO
nn(4) = 1 ?
S ← S+1
DPRL ← M(S)
S ← S+1
DPRH ← M(S)
NO
nn(3) = 1 ?
x = 0 ?
NO
S ← S+1
YL ← M(S)
S ← S+1
YH ← M(S)
S ← S+1
YL ← M(S)
NO
nn(2) = 1 ?
x = 0 ?
NO
S ← S+1
XL ← M(S)
S ← S+1
XH ← M(S)
S ← S+1
XL ← M(S)
NO
nn(1) = 1 ?
m = 0 ?
NO
S ← S+1
BL ← M(S)
S ← S+1
BH ← M(S)
S ← S+1
BL ← M(S)
NO
nn(0) = 1 ?
m = 0 ?
NO
S ← S+1
AL ← M(S)
S ← S+1
AH ← M(S)
S ← S+1
AL ← M(S)

Operation :

If m=0, rotate n bits to left (n=0-65535)



If m=1, rotate n bits to left (n=0-255)



Description : The contents of the accumulator A are rotated to the left by n bits. The value of n is specified by the instruction's third byte (or third and fourth bytes when m=0).

Status flags : Not affected.

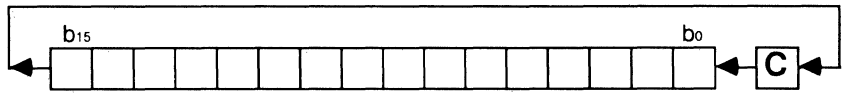
Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	RLA #imm	89_{16}, 49_{16}, imm	3	6+i

i: Number of rotation

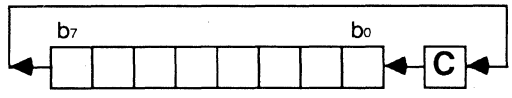
(Note1) When the data length selection flag m is 0, the bytes-count increases by 1.

Operation :

If m=0,



If m=1,



Description : The carry flag C is linked to the accumulator or memory, and the combined contents are rotated by 1 bit to the left.

Bit 0 of the accumulator or memory is loaded with the content of the carry flag C before execution of this instruction, and the carry flag C is loaded with the content of bit 15 (or bit 7 if the data length selection flag m is set to 1) of the accumulator or memory before execution of this instruction.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

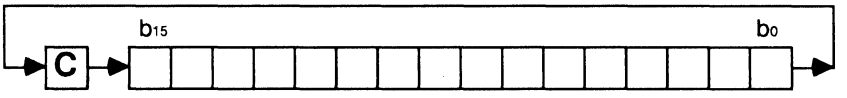
C : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) before execution of the instruction is 1. Otherwise, cleared to 0

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	ROL A	$2A_{16}$	1	2
Direct	ROL dd	26_{16}, dd	2	7
Direct indexed X	ROL dd, X	36_{16}, dd	2	7
Absolute	ROL mmll	$2E_{16}$, ll, mm	3	7
Absolute indexed x	ROL mmll, X	$3E_{16}$, ll, mm	3	8

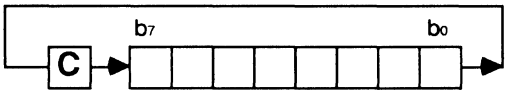
(Note1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Operation :

If m=0,



If m=1,



Description : The carry flag C is linked to the accumulator or memory, and the combined contents are shifted by 1 bit to the right.

Bit 15 (or bit 7 if the data length selection flag m is set to 1) of the accumulator or memory is loaded with the content of the carry flag C, and the carry flag C is loaded with the content of bit 0 of the accumulator or memory before execution of this instruction.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 when bit 0 before execution of the instruction is 1. Otherwise, cleared to 0.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Accumulator	ROR A	$6A_{16}$	1	2
Direct	ROR dd	66_{16}, dd	2	7
Direct indexed X	ROR dd, X	76_{16}, dd	2	7
Absolute	ROR mmll	$6E_{16}$, ll, mm	3	7
Absolute indexed X	ROR mmll, X	$7E_{16}$, ll, mm	3	8

(Note1) The accumulator addressing mode's specification in this table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Operation :

$S \leftarrow S + 1$
$PS_L \leftarrow M(S)$
$S \leftarrow S + 1$
$PS_H \leftarrow M(S)$
$S \leftarrow S + 1$
$PC_L \leftarrow M(S)$
$S \leftarrow S + 1$
$PC_H \leftarrow M(S)$
$S \leftarrow S + 1$
$PG \leftarrow M(S)$

Description : The contents of the processor status register PS, program counter PC, and program bank register PG, which are saved on the stack when the last interrupt was accepted, are restored these registers.

Status flags : Restored according to the values that had been on the stack.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	RTI	40_{16}	1	11

Operation : $S \leftarrow S + 1$
$PC_L \leftarrow M(S)$
$S \leftarrow S + 1$
$PC_H \leftarrow M(S)$
$S \leftarrow S + 1$
$PG \leftarrow M(S)$

Description : The program counter PC and program bank register PG are restored according to the state previously saved on the stack.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	RTL	$6B_{16}$	1	8

Operation : $S \leftarrow S + 1$
$PC_L \leftarrow M(S)$
$S \leftarrow S + 1$
$PC_H \leftarrow M(S)$

Description : The program counter PC is restored according to the state previously saved on the stack.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	RTS	60_{16}	1	5

Operation : Acc, C ← Acc - M - $\overline{C}$

Description : Subtracts the contents of memory and the 1's complements of carry flag from the contents of the accumulator , and places the result in the accumulator. Executed as a binary subtraction if the decimal operation mode flag D is set to 0. Executed as a decimal subtraction if the decimal operation mode flag D is set to 1.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0. Meaningless for decimal subtraction.

V : Set to 1 when binary subtraction of signed data results in a value outside the range of -32768 to +32767 (-128 to +127 if the data length selection flag m is set to 1). Otherwise, cleared to 0. Meaningless for decimal subtraction.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Set to 1 when the result of operation is equal to or larger than 0. Otherwise, cleared to 0, and a borrow is indicated.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	SBC A, #imm	$E9_{16}$, imm	2	2
Direct	SBC A, dd	$E5_{16}$, dd	2	4
Direct indexed X	SBC A,dd, X	$F5_{16}$, dd	2	5
Direct indirect	SBC A, (dd)	$F2_{16}$, dd	2	6
Direct indexed X indirect	SBC A,(dd, X)	$E1_{16}$, dd	2	7
Direct indirect indexed Y	SBC A,(dd), Y	$F1_{16}$, dd	2	8
Direct indirect long	SBCL A, (dd)	$E7_{16}$, dd	2	10
Direct indirect long indexed Y	SBCL A, (dd), Y	$F7_{16}$, dd	2	11
Absolute	SBC A,mmll	ED_{16},ll,mm	3	4
Absolute indexed X	SBC A, mmll, X	FD_{16}, ll, mm	3	6
Absolute indexed Y	SBC A, mmll, Y	$F9_{16}$, ll, mm	3	6
Absolute long	SBC A, hhmmll	EF_{16}, ll, mm, hh	4	6
Absolute long indexed X	SBC A, hhmmll, X	FF_{16}, ll, mm, hh	4	7
Stack pointer relative	SBC A, nn, S	$E3_{16}$, nn	2	5
Stack pointer relative indirect indexed Y	SBC A, (nn, S), Y	$F3_{16}$, nn	2	8

(Note1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

(Note 2)When operating on 16-bit data in the immediate addressing mode with the data length selection flag m set to 0, the bytes-count increases by 1.

Operation : M ← M V IMM

IMM is the bit pattern that specifies the bit positions that are to be set to 1.

When the data length selection flag m is set to 1, IMM is placed in the third byte (direct bit addressing mode) or the fourth byte (absolute bit addressing mode) of the instruction.

When the data length selection flag m is set to 0, IMM is placed in the third and fourth bytes (direct bit addressing mode) or the fourth and fifth bytes (absolute bit addressing mode) of the instruction.

Description : The SEB instruction sets the specified memory bits to 1. Multiple bits to be set can be specified at one time.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct bit	SEB #imm, dd	04_{16}, dd, imm	3	8
Absolute bit	SEB #imm, mmll	$0C_{16}$, ll, mm, imm	4	9

(Note1) When operating on 16-bit data with the data length selection flag m set to 0, the bytes-count increases by 1.

Operation : C ← 1

Description : Sets the carry flag C to 1.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Set to 1.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	SEC	38_{16}	1	2

Operation : I ← 1

Description : Sets the interrupt disable flag I to 1.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Set to 1.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	SEI	78_{16}	1	2

Operation : m ← 1

Description : Sets the data length selection flag m to 1.

Status flags

IPL : Not affected.
N : Not affected.
V : Not affected.
m : Set to 1.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Not affected.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	SEM	$F8_{16}$	1	2

Operation : $PS_L \leftarrow PS_L \vee IMM$
(IMM is the immediate value specified in the second byte of the instruction.)

Description : Sets the processor status flags specified by the bit pattern in the second byte of the instruction to 1.

Status flags : The specified flags are set. IPL is not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Immediate	SEP #imm	$E2_{16}$, imm	2	3

Operation : When m=0, When m=1
$M(n) \leftarrow Acc_L$ $M(n) \leftarrow Acc_L$
$M(n+1) \leftarrow Acc_H$

Description : Stores the contents of the accumulator in memory.
The contents of the accumulator are not changed.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct	STA A, dd	85_{16}, dd	2	4
Direct indexed X	STA A, dd, X	95_{16}, dd	2	5
Direct indirect	STA A, (dd)	92_{16} dd	2	7
Direct indexed X indirect	STA A, (dd, X)	81_{16}, dd	2	7
Direct indirect indexed Y	STA A, (dd), Y	91_{16}, dd	2	7
Direct indirect long	STAL A, (dd)	87_{16}, dd	2	10
Direct indirect long indexed Y	STAL A, (dd), Y	97_{16}, dd	2	11
Absolute	STA A, mmll	$8D_{16}$, ll, mm	3	5
Absolute indexed X	STA A, mmll, X	$9D_{16}$, ll, mm	3	5
Absolute indexed Y	STA A, mmll, Y	99_{16}, ll, mm	3	5
Absolute long	STA A, hhmmll	$8F_{16}$, ll, mm, hh	4	6
Absolute long indexed X	STA A, hhmmll, X	$9F_{16}$, ll, mm, hh	4	7
Stack pointer relative	STA A, nn, S	83_{16}, nn	2	5
Stack pointer relative indirect indexed Y	STA A, (nn, S), Y	93_{16}, nn	2	8

(Note1) This table applies when using the accumulator A. If using the accumulator B, replace "A" with "B". In this case, "42_{16}" is added at the beginning of the machine code, the bytes-count increases by 1 and the cycles-count increases by 2.

Operation : Stop the oscillator.

Description : Resets the oscillator controlling flip-flop circuit to inhibit the oscillator. To restart the oscillator, either an interrupt or reset must be executed.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	STP	DB_{16}	1	3

Operation : When x=0,
$M(n) \leftarrow X_L$
$M(n+1) \leftarrow X_H$

When x=1
$M(n) \leftarrow X_L$

Description : Stores the contents of the index register X in memory. The contents of the index register X remain the same.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct	STX dd	86_{16}, dd	2	4
Direct indexed Y	STX dd, Y	96_{16}, dd	2	5
Absolute	STX mmll	$8E_{16}$, ll, mm	3	5

Operation : When x=0,

$M(n) \leftarrow Y_L$

$M(n+1) \leftarrow Y_H$

When x=1

$M(n) \leftarrow Y_L$

Description : Stores the contents of the index register Y in memory. The contents of the index register Y remain the same.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Direct	STY dd	84_{16}, dd	2	4
Direct indexed X	STY dd, X	94_{16}, dd	2	5
Absolute	STY mmll	$8C_{16}$, ll, mm	3	5

Operation : DPR ← A

Description : Loads the direct page register DPR with the contents of the accumulator A. Data is transferred as 16-bit data regardless of the status of the data length selection flag m. The contents of the accumulator A are not changed.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TAD	$5B_{16}$	1	2

Operation : S ← A

Description : Loads the stack pointer S with the contents of the accumulator A. Data is transferred as 16-bit data regardless of the status of the data length selection flag m. The contents of the accumulator A are not changed.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TAS	$1B_{16}$	1	2

Operation : If x=0,

$X_L \leftarrow A_L$

$X_H \leftarrow A_H$

If x=1,

$X_L \leftarrow A_L$

Description : Loads the index register X with the contents of the accumulator A. The contents of the accumulator A are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TAX	AA_{16}	1	2

Operation : If x=0,

$Y_L \leftarrow A_L$

$Y_H \leftarrow A_H$

If x=1,

$Y_L \leftarrow A_L$

Description : Loads the index register Y with the contents of the accumulator A. The contents of the accumulator A are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TAY	$A8_{16}$	1	2

Operation : DPR ← B

Description : Loads the direct page register DPR with the contents of the accumulator B. Data is transferred as 16-bit data regardless of the status of the data length selection flag m. The contents of the accumulator B are not changed.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TBD	42_{16}, $5B_{16}$	2	4

Operation : S ← B

Description : Loads the stack pointer S with the contents of the accumulator B. Data is transferred as 16-bit data regardless of the status of the data length selection flag m. The contents of the accumulator B are not changed.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TBS	42_{16}, $1B_{16}$	2	4

Operation : If x=0,

$X_L \leftarrow B_L$

$X_H \leftarrow B_H$

If x=1,

$X_L \leftarrow B_L$

Description : Loads the index register X with the contents of the accumulator B. The contents of the accumulator B are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TBX	42_{16}, AA_{16}	2	4

Operation : If x=0,
$Y_L \leftarrow B_L$
$Y_H \leftarrow B_H$

If x=1,
$Y_L \leftarrow B_L$

Description : Loads the index register Y with the contents of the accumulator B. The contents of the accumulator B are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TBY	42_{16}, $A8_{16}$	2	4

Operation : If m=0,

$A_L \leftarrow DPR_L$

$A_H \leftarrow DPR_H$

If m=1,

$A_L \leftarrow DPR_L$

Description : Loads the accumulator A with the contents of the direct page register DPR. The contents of the direct page register DPR are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TDA	$7B_{16}$	1	2

Operation : If m=0,

$B_L \leftarrow DPR_L$

$B_H \leftarrow DPR_H$

If m=1,

$B_L \leftarrow DPR_L$

Description : Loads the accumulator B with the contents of the direct page register DPR. The contents of the direct page register DPR are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TDB	42_{16}, $7B_{16}$	2	4

Operation : If m=0,
$A_L \leftarrow S_L$
$A_H \leftarrow S_H$

If m=1,
$A_L \leftarrow S_L$

Description : Loads the accumulator A with the contents of the stack pointer S. The contents of the stack pointer S are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TSA	$3B_{16}$	1	2

Operation : If m=0,

$B_L \leftarrow S_L$

$B_H \leftarrow S_H$

If m=1,

$B_L \leftarrow S_L$

Description : Loads the accumulator B with the contents of the stack pointer S. The contents of the stack pointer S are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TSB	42_{16}, $3B_{16}$	2	4

Operation : If x=0,
$X_L \leftarrow S_L$
$X_H \leftarrow S_H$

If x=1,
$X_L \leftarrow S_L$

Description : Loads the index register X with the contents of the stack pointer S. The contents of the stack pointer S are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TSX	BA_{16}	1	2

Operation : If m=0 and x=0,
$A_L \leftarrow X_L$
$A_H \leftarrow X_H$

If m=0 and x=1,
$A_L \leftarrow X_L$
$A_H \leftarrow 00_{16}$

If m=1,
$A_L \leftarrow X_L$

Description : Loads the accumulator A with the contents of the index register X. The contents of the index register X are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TXA	$8A_{16}$	1	2

Operation : If m=0 and x=0, $B_L \leftarrow X_L$, $B_H \leftarrow X_H$ / If m=0 and x=1, $B_L \leftarrow X_L$, $B_H \leftarrow 00_{16}$ / If m=1, $B_L \leftarrow X_L$

Description : Loads the accumulator B with the contents of the index register X. The contents of the index register X are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TXB	42_{16}, $8A_{16}$	2	4

Operation : If x=0,
$S_L \leftarrow X_L$
$S_H \leftarrow X_H$

If x=1,
$S_L \leftarrow X_L$
$S_H \leftarrow 00_{16}$

Description : Loads the stack pointers with the contents of the index register X. The contents of the index register X are not changed.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TXS	$9A_{16}$	1	2

Operation : If x=0,
$Y_L \leftarrow X_L$
$Y_H \leftarrow X_H$

If x=1,
$Y_L \leftarrow X_L$

Description : Loads the index register Y with the contents of the index register X. The contents of the index register X are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) cf the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TXY	$9B_{16}$	1	2

Operation : If m=0 and x=0, If m=0 and x=1, If m=1,

$A_L \leftarrow Y_L$ $A_L \leftarrow Y_L$ $A_L \leftarrow Y_L$

$A_H \leftarrow Y_H$ $A_H \leftarrow 00_{16}$

Description : Loads the accumulator A with the contents of the index register Y. The contents of the index register Y are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TYA	98_{16}	1	2

Operation : If m=0 and x=0,
$B_L \leftarrow Y_L$
$B_H \leftarrow Y_H$

If m=0 and x=1,
$B_L \leftarrow Y_L$
$B_H \leftarrow 00_{16}$

If m=1,
$B_L \leftarrow Y_L$

Description : Loads the accumulator B with the contents of the index register Y. The contents of the index register Y are not changed.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the operation result is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TYB	42_{16}, 98_{16}	2	4

Operation : If x=0,
$X_L \leftarrow Y_L$
$X_H \leftarrow Y_H$

If x=1,
$X_L \leftarrow Y_L$

Description : Loads the index register X with the contents of the index register Y. The contents of the index register Y are not changed.

Status flags

IPL : Not affected.
N : Set to 1 when bit 15 (or bit 7 if the index register length selection flag x is set to 1) of the operation result is 1. Otherwise, cleared to 0.
V : Not affected.
m : Not affected.
x : Not affected.
D : Not affected.
I : Not affected.
Z : Set to 1 when the result of operation is 0. Otherwise, cleared to 0.
C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	TYX	BB_{16}	1	2

Operation : Stop the internal clock.

Description : The WIT instruction stops the internal clock but not the external clock is not stopped. To restart the internal clock, either an interrupt or reset must be executed.

Status flags : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	WIT	CB_{16}	1	3

Operation : If m=0,
$A_L \leftrightarrow B_L$
$A_H \leftrightarrow B_H$

If m=1,
$A_L \leftrightarrow B_L$

Description : Swaps the contents of the accumulators A and B.

Status flags

IPL : Not affected.

N : Set to 1 when bit 15 (or bit 7 if the data length selection flag m is set to 1) of the accumulator A after the operation is 1. Otherwise, cleared to 0.

V : Not affected.

m : Not affected.

x : Not affected.

D : Not affected.

I : Not affected.

Z : Set to 1 when the contents of the accumulator A is cleared to 0 by the operation. Otherwise, cleared to 0.

C : Not affected.

Addressing mode	Syntax	Machine code	Bytes	Cycles
Implied	XAB	89_{16}, 28_{16}	2	6

5. Notes for Programming

Take care of the following when programming with the MELPS 7700 series.

(1) The stack pointer S is undefined immediately after the reset is commanded. Always set the initial value.

Example)
```
LDX  #27FH
TXS
```

(2) The program bank register PG and the data bank register DT are disabled under the single chip mode. Do not set value other than "00_{16}" here.

(3) When "1" is set in the D-flag for decimal operation:

The C-flag alone is effective in the ADC instruction, while the Z, N, and V flags are disabled. The C and Z flags alone are effective in the SBC instruction, while the N and V flags are disabled. (Decimal operation can be done in the ADC and the SBC instructions alone.)

(4) Using the 16-bit immediate data with "1" (data length : 8 bits) in the data length selection flag m, or using the 8- bit immediate data with "0" (data length : 16 bits) in flag m, will cause the program run-away. The same rule is applied to the index register length selection flag x. Take care of the condition of these flags when coding the program.

(5) The MELPS 7700 can prefetch the instructions using the 3-byte instruction queue buffer. Keep in mind when creating the timer with the software, that the number of cycles shown in the list of machine language instructions is the minimum value. (Also see Chapter 6.)

(6) When value other than "00_{16}" is set in the lower order 8 bits of the direct page register DPR (DPR_L), the processing time will become 1 machine cycle longer than when "00_{16}" is set.

(7) The processing speed will deteriorate if a 16- bit data will be accessed from an odd address. Place the 16-bit data from an even address if the processing speed is important.

(8) The N and Z flags will change by execution of the PLA instruction, but the contents of the processor status register will not change if the accumulator A alone is recovered by the PUL instruction.

(9) The program bank register PG can be saved into the stack by setting "1" in bit 6 of the operation by the PSH instruction. However, the PG cannot be recovered by the PUL instruction.

(10) When the PUL or the PSH instruction is executed, the flag m and the flag x are affected in addition.

(11) The code in the second byte of the BRK instruction will not affect the CPU.

(12) When the block transfer instruction (MVN or MVP) is executed with x=1, the contents of middle order 8-bit in source and destination address (There are consists of 24-bit.) will be fixed "00$_{16}$".

6. Instruction Execution Sequence

The basic clock of the MELPS 7700 central processing unit (CPU) is clock ϕ (1/2 the oscillation frequency f(X_{IN})). The basic clock of the bus is an $\overline{E}$ derived from clock ϕ, so data exchange between the CPU and the internal bus is done via the bus interface unit. The frequency of $\overline{E}$ is normally 1/2 that of clock ϕ, but it becomes 1/4 that of ϕ, when accessing external memory while the wait is enabled by the wait bit.

6.1 Bus Interface Unit

The bus interface unit is a unit that helps data exchange between the CPU and the internal bus. The unit is structured by registers and buffers as shown in Figure 6.1.1. The functions of these registers and buffers are shown in Table 6.1.1. The CPU reads the instruction code from the instruction queue buffer, and the data from the data buffer of the bus interface unit. Then, data is written in the data buffer of the bus interface unit. The bus interface unit reads or writes data from the memory or I/O via the bus, instead of the CPU.

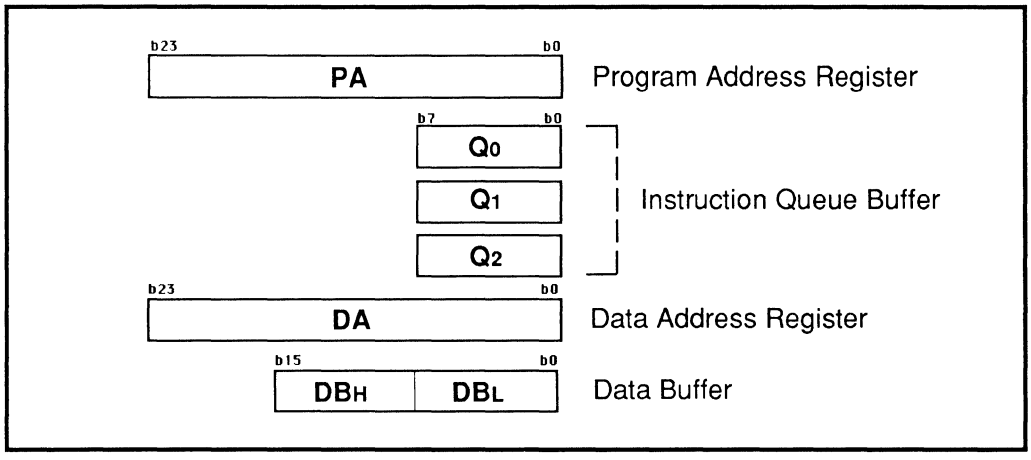


Fig. 6.1.1 Bus Interface Unit Register Model

Table 6.1.1 Functions of the Registers and Buffers

Name	Function
Program address register	This register indicates the address where the program is stored.
Instruction queue buffer	The 3-byte buffer for temporal storage of the instruction prefetched from the memory.
Data address register	The register that indicates the address for data read or data write.
Data buffer	The buffer where the bus interface unit temporarily stores data read from the memory or I/O or where the CPU temporarily stores data to be written into the memory or I/O.

6.2 Change of the CPU Basic Clock ϕ_{CPU}

When the bus interface unit is not ready, the CPU extends the basic clock to synchronize with the bus, and waits till it is ready. As the CPU basic clock waits owing to some conditions, this clock will be called ϕ_{CPU} to be distinguished from the clock . The following are the cases in which the ϕ_{CPU} waits.

Causes for the ϕ_{CPU} to wait

<Cause 1>

When the CPU requests operation codes and operands, but the operation codes and operands in the instruction queue buffer did not reach the necessary number.

<Cause 2>

When the CPU tried to access data, but the bus interface unit was using the bus for fetching some data into the instruction queue buffer or writing data.

<Cause 3>

When the bus interface unit was reading data from the internal/external memory or I/O, according to the request of the CPU.

In addition to the above, the following are also causes for the ϕ_{CPU} to be extended.

- When 16-bit data is accessed from odd address.
- When external memory 16-bit data is accessed while the BYTE terminal level is "H".
- When external memory is accessed with wait commanded by the wait bit.

The above conditions causes the execution time to differ each time, even with the same instruction and same addressing mode. Two example instructions are given in the next section to see the variation of the number of cycles according to the above conditions.

The " CPU execution sequence per addressing mode " of Appendix-A is the CPU instruction execution sequences based on the ϕ_{CPU} . The number of cycles shown in " 4.2 Instructions " and " Appendix-B List of machine language instructions " are the count for the shortest case, and cannot always be applied when calculating the actual cycles or the execution time of instructions.

6.3 Instruction Execution Sequence

The instruction execution sequence of the CPU based on the ϕ_{CPU}, and the variation of the actual instruction execution cycle when various conditions are applied are shown here.

- Example 1. ASL instruction Direct addressing mode
- Example 2. LDA instruction Direct indirect long addressing mode

Before observing the ϕ_{CPU} based CPU instruction execution sequence

The following table describes the ϕ_{CPU} based CPU instruction execution sequence symbols. The signals indicated in this execution sequence are all CPU internal signals, that show data exchange between the bus interface unit and the CPU. Accordingly, these signals cannot be observed from outside.

ϕ_{CPU} Based CPU Instruction Execution Sequence Symbols

Symbol	Description
ϕ_{CPU}	CPU basic clock
$A_{P(CPU)}$	Higher order 8 bits of the address (24 bits) of the program that the CPU is actually execution
$A_HA_{L(CPU)}$	Lower order 16 bits of the address (24 bits) of the program that the CPU is actually execution
$DATA_{(CPU)}$	Data information the CPU is processing
$R/\overline{W}_{(CPU)}$	Data read/write request to the data buffer in the bus interface
PG,PC	Contents of the program bank register (PG) and the program counter (PC)
AD_P	Data indicating the address (higher order 8 bits)
AD_H,AD_L	Data indicating the address (middle order 8 bits, lower order 8 bits)
DPR_H	Contents of the higher order 8 bits of the direct page register
DPR_L	Contents of the lower order 8 bits of the direct page register (DPRL = 0 in the examples)
D_H	Data to be fetched or written from the data buffer by the CPU (higher order 8 bits)
D_L	Data to be fetched or written from the data buffer by the CPU (lower order 8 bits)
dd	Contents of the operand (DPR_L = 0 in examples 1 and 2, so dd represents the lower order 8 bits of the address)

Before observing the φ based instruction execution sequence

The φ based execution sequence symbols are shown in the following table. The signals in this execution sequence indicates data exchange of the bus interface unit with the memory and I/O. The internal instruction execution sequence of the CPU can be guessed from these signals. However, the ϕ_{CPU} and the number of data in the instruction queue buffer shown here cannot be observed from the outside.

φ Based Execution Sequence Symbols

Symbol	Description
φ	Basic operation clock of the microcomputer **f(XIN) / 2**
$\overline{E}$	Basic operation clock of the bus **φ / 2**
hh	Higher order 8 bits of the address where the bus interface unit is to access to (bank)
mm	Middle order 8 bits of the address where the bus interface unit is to access to
ll	Lower order 8 bits of the address where the bus interface unit is to access to
DPR DPR_H DPR_L	Contents of the direct page Contents of the higher order 8 bits of the direct page register Contents of the lower order 8 bits of the direct page register
OP_1 OP_2 OP_3 :	Data to be fetched into the instruction queue buffer by the bus interface (Operation code or operand) The subscript represents the fetch sequence.
D_L D_H	Data to be fetched into the data buffer or data to be written into the memory by the bus interface unit
dd	Data obtained as the operand (The lower order 8 bits of the address are given in examples 1 and 2, because DPR_L = 0.)
AD_P	Higher order 8 bits of data that indicates the address (contents of the data address register)
AD_H	Middle order 8 bits of data that indicates the address (contents of the data address register)
AD_L	Lower order 8 bits of data that indicates the address (contents of the data address register)

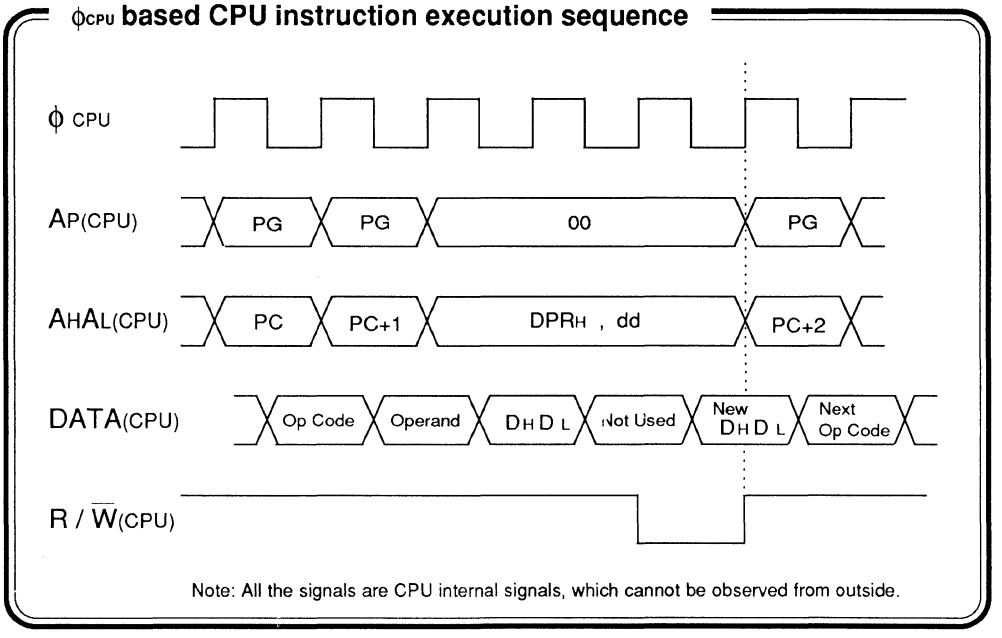
The following are the cause of the "ϕ_{CPU} to queue" in the φ based execution sequence.

Cause 1
When the CPU required operation codes and operands, but the number of operation codes and operands did not reach the requested number.

Cause 2
When the CPU tried to access data, but the bus interface was using the bus for fetching data into the instruction queue buffer or for writing data.

Cause 3
When the bus interface unit is reading data from the internal/external memory or I/O, etc., according to the request of the CPU.

Example 1. ASL instruction / direct addressing mode (DPR_L = 00_{16})



The following examples 1-1 to 1-6 are examples of the ϕ_{CPU} based instruction execution sequences under various conditions.

Example 1-1 **When the instruction queue buffer is vacant**

Example 1-2 **When two data are in the instruction queue buffer**

Example 1-3 **When three data are in the instruction queue buffer**

Example 1-4 **When 16-bit data is accessed from odd address**

Example 1-5 **When external memory is accessed from the BYTE terminal using 8-bit external bus width**

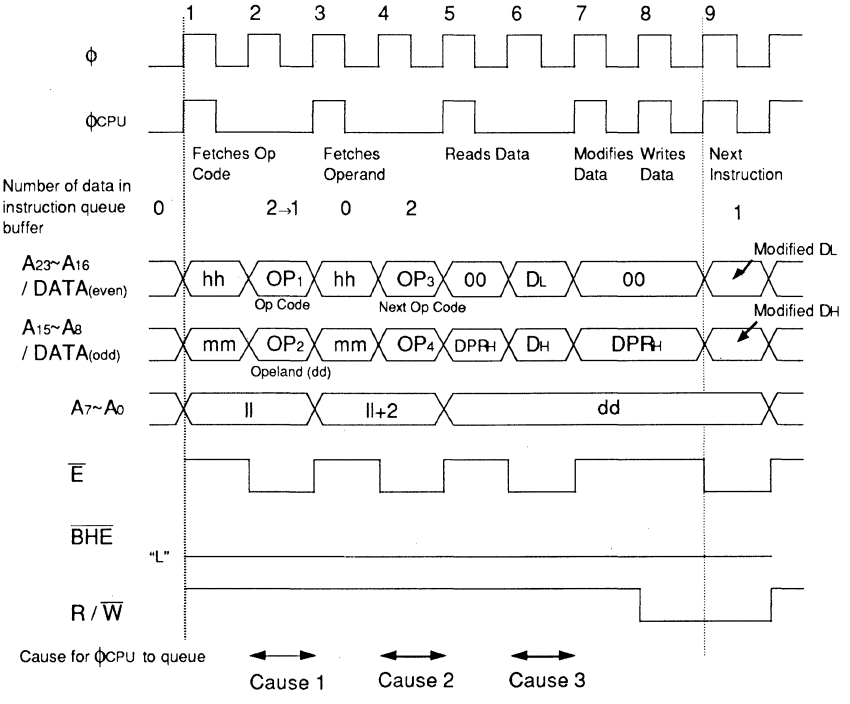
Example 1-6 **When external memory is accessed with wait by the wait bit**

(Example 1-1) When the instruction queue buffer is vacant

Conditions

- Number of data in the instruction queue buffer — 0
- ROM, RAM — External memory is used (Note)
- Data length selection flag m — "0" (16-bit length)
- BYTE terminal level — "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter — Even
- Contents of the operand (dd) — Even

ϕ based execution sequence



Note. The operation when internal ROM and internal RAM are used, will be as shown above, regardless of the level of the BYTE terminal. However, the address/data bus, $\overline{BHE}$, R/$\overline{W}$ signal cannot be observed from outside, when the mode is single-chip mode.

Operation of the CPU and bus interface unit under various cycles

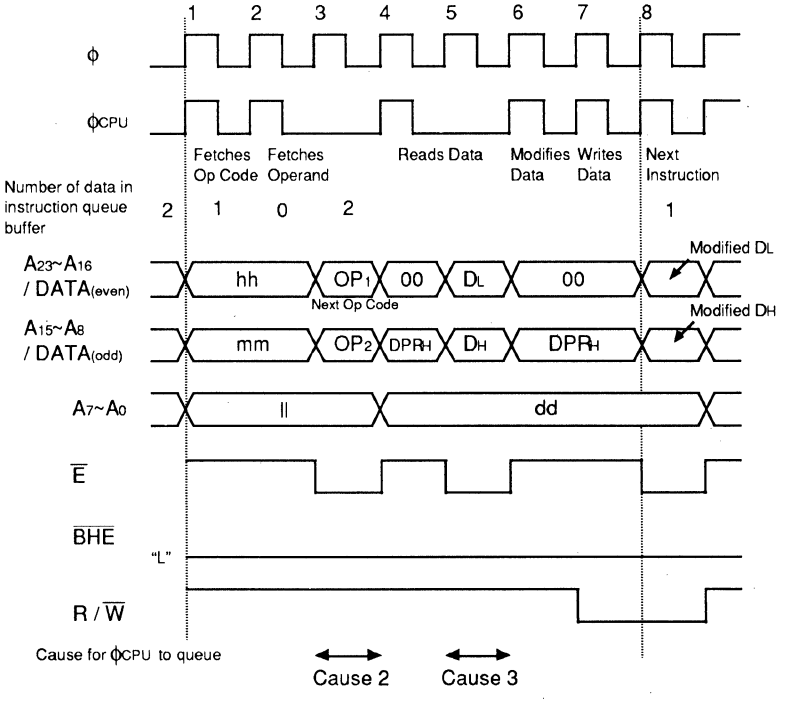
φ No.	CPU	Bus interface unit
1	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
2	Fetches the operation code.	Fetches 2-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
3	Fetches the operand.	Prefetches the instruction, because the instruction queue buffer is vacant and the CPU is not using the bus.
4	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2 bytes worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
5	Waits for $\overline{E}$ to become "L", to read data.	
6	Reads data when $\overline{E}$ becomes "L".	
7	Modifies data.	
8	Writes data into the data buffer.	
9	Fetches the next operation code.	Writes the contents of the data buffer into the original address, when E becomes "L".

(Example 1-2) When two data are in the instruction queue buffer

Conditions

- Number of data in the instruction queue buffer — 2
- ROM, RAM — External memory is used (Note)
- Data length selection flag m — "0" (16-bit length)
- BYTE terminal level — "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter — Even
- Contents of the operand (dd) — Even

φ based execution sequence



Note. The operation when internal ROM and internal RAM are used, will be as shown above, regardless of the level of the BYTE terminal. However, the address/data bus, $\overline{BHE}$, R/$\overline{W}$ signal cannot be observed from outside, when the mode is single chip mode.

Operation of the CPU and bus interface unit under various cycles

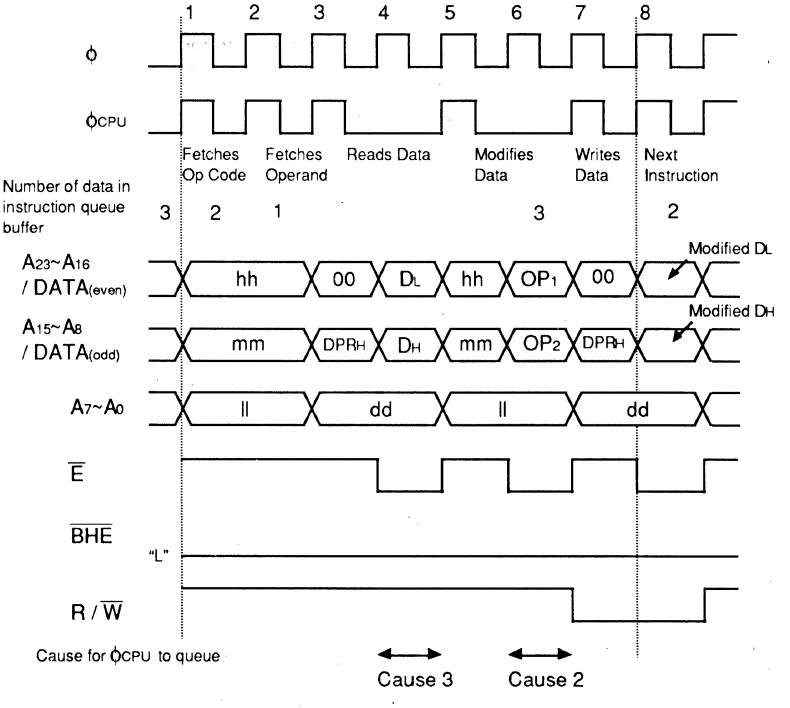
ϕ No.	CPU	Bus interface unit
1	Fetches operation code.	
2	Fetches operand (dd).	Prefetches the instruction, because the instruct queue buffer is vacant and the CPU is not using the bus.
3	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
4	Waits for $\overline{E}$ to become "L", to read data.	
5	Reads data when $\overline{E}$ becomes "L".	
6	Modifies data.	
7	Writes data into the data buffer.	
8	Fetches the next operation code.	Writes the contents of the data buffer into the original address, when $\overline{E}$ becomes "L".

(Example 1-3) When three data are in the instruction queue buffer

Conditions

- Number of data in the instruction queue buffer — 3
- ROM, RAM — External memory is used (Note)
- Data length selection flag m — "0" (16-bit length)
- BYTE terminal level — "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter — Even
- Contents of the operand (dd) — Even

φ based execution sequence



Note. The operation when internal ROM and internal RAM are used, will be as shown above, regardless of the level of the BYTE terminal. However, the address/data bus, $\overline{BHE}$, R/$\overline{W}$ signal cannot be observed from outside, when the mode is single chip mode.

Operation of the CPU and bus interface unit under various cycles

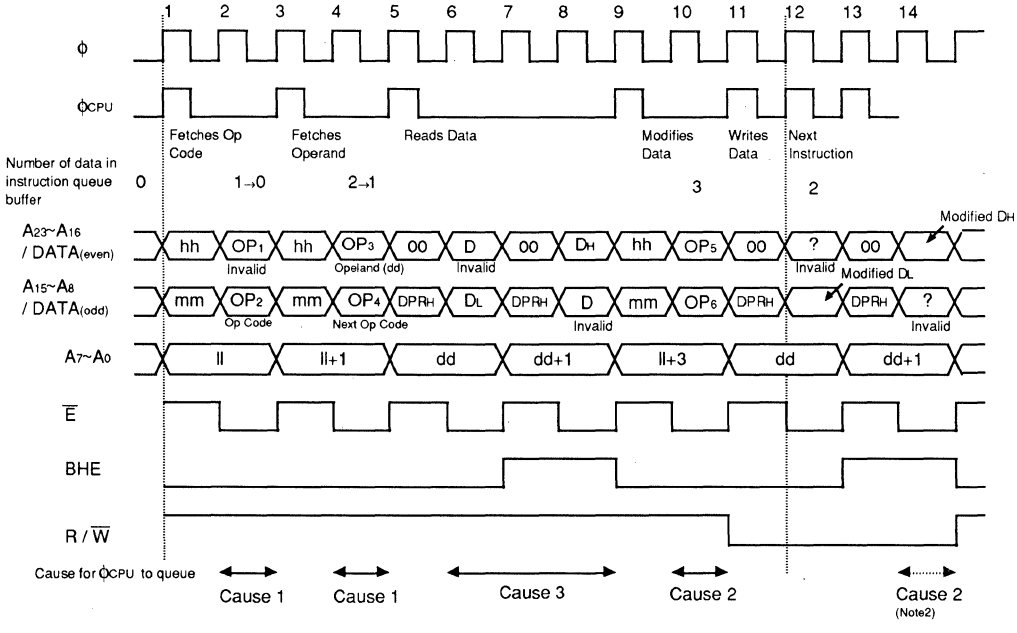
φ No.	CPU	Bus interface unit
1	Fetches operation code .	
2	Fetches operand (dd).	
3	Waits for $\overline{E}$ to become "L", to read data.	
4	Reads data when $\overline{E}$ becomes "L".	
5	Modifies data.	Prefetches the instruction, because there are two vacant instruction queue buffers and the CPU is not using the bus.
6	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
7	Writes data into the data buffer.	
8	Fetches the next operation code.	Writes the contents of the data buffer into the original address, as $\overline{E}$ becomes "L".

(Example 1-4) When 16-bit data is accessed from odd address

Conditions

- Number of data in the instruction queue buffer — 0
- ROM, RAM — External memory is used (Note1)
- Data length selection flag m — "0" (16-bit length)
- BYTE terminal level — "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter — Odd
- Contents of the operand (dd) — Odd

φ based execution sequence



Note1. The operation when internal ROM and internal RAM are used, will be as shown above, regardless of the level of the BYTE terminal. However, the address/data bus, $\overline{BHE}$, R/$\overline{W}$ signal cannot be observed from outside, when the mode is single chip mode.

Note 2. At the <- - -> part

* When the CPU does not use the bus, φCPU corresponds with φ.
* When the CPU uses the bus, the φCPU queues till the writing in the bus interface unit completes. (the ϕ_{14} cycle)

Operation of the CPU and bus interface unit under various cycles

φ No.	CPU	Bus interface unit
1	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
2	Fetches operation code.	Fetches 1 odd address byte worth of data into the instruction queue buffer, when $\overline{E}$ becomes "L".
3	(No fetching can be done, because there are no operands in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
4	Fetches operand (dd).	Fetches 2-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
5	Waits for $\overline{E}$ to become "L", to read data.	
6	Reads data in the odd addresses (D_L) alone into the data buffer when $\overline{E}$ becomes "L".	
7	Waits for $\overline{E}$ to become "L", to read data.	
8	Reads data in the even addresses (D_H) alone into the data buffer when $\overline{E}$ becomes "L".	
9	Modifies data.	Prefetches the instruction, because there are two vacant positions in the instruction queue buffer, and the CPU is not using the bus.
10	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2 bytes worth of data into the instruction queue buffer, when $\overline{E}$ becomes "L".
11	Writes data into the data buffer.	Waits till $\overline{E}$ becomes "L" to write data.
12	Fetches the next operation code.	Writes the contents of the data buffer (D_L) into the original address (odd address), when $\overline{E}$ becomes "L".
13	?	Waits till $\overline{E}$ becomes "L" to write data.
14	?	Writes the contents of the data buffer (D_H) into the original address (even address), when $\overline{E}$ becomes "L".

When internal ROM or BYTE terminal level "L" external memory is used as the program memory, the instruction is fetched into the instruction queue buffer normally in 2-byte (word) unit of sequential even and odd addresses in this order. However, when the instruction must be fetched from odd address like after execution of the JMP instruction, the 1-byte of the first odd address alone is fetched into the instruction queue buffer (ϕ_2 cycle), and the later instructions are fetched into the instruction queue buffer in 2-byte units (ϕ_4, ϕ_{10} cycle).

The bus interface unit automatically selects whether to fetch one word or to fetch the 1 byte of odd address alone. The operation status can be observed from outside, according to the output of the $\overline{BHE}$ terminal and the address bus signal A_0, as long as the mode is not single chip mode.

- When one word is fetched
  The output from both the $\overline{BHE}$ terminal and the address bus A_0 are at the "L" level.
- When 1 byte of odd address alone is fetched
  The output from the $\overline{BHE}$ terminal is "L", while the output from address bus A_0 is "H".

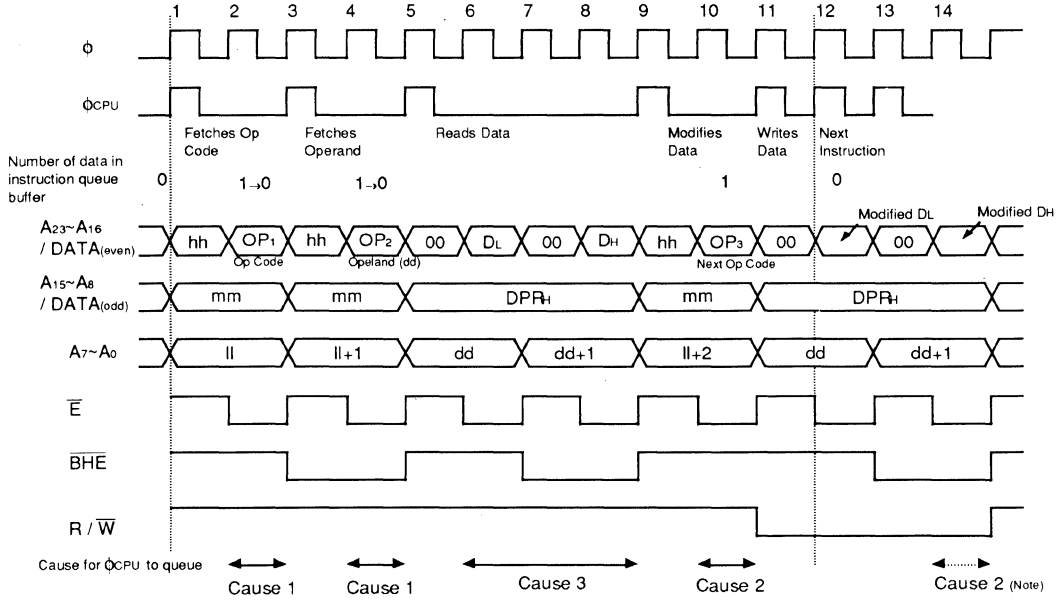
When internal RAM and external memory at BYTE terminal level "L" are used as the data memory, with data length selection flag m = 0, both data read and write are normally done in 2-byte units of even and odd addresses, in this sequence. However, access can also be done when the word data is defined from an odd address. In other words, "H" is output first from address bus A_0 and then "L" from the $\overline{BHE}$ terminal to access to odd address alone. Next, "L" is output from A_0, and "H" from the $\overline{BHE}$ terminal to access to the even address. (ϕ_5 to ϕ_8, ϕ_{11} to ϕ_{14} cycle)

(Example 1-5) When external memory is accessed from the BYTE terminal using 8-bit external bus width

Conditions

- Number of data in the instruction queue buffer — 0
- ROM, RAM — External memory is used
- Data length selection flag m — "0" (16-bit length)
- BYTE terminal level — "H" (External bus width is 8 bits)
- Contents of lower order bytes (PC_L) of the program counter — Even
- Contents of the operand (dd) — Even

φ based execution sequence



Note. At the <- - -> part

* When the CPU does not use the bus, ϕ_{CPU} corresponds with φ.
* When the CPU uses the bus, the ϕ_{CPU} queues till the writing in the bus interface unit completes. (the ϕ_{13} to ϕ_{14} cycle)

Operation of the CPU and bus interface unit under various cycles

φ No.	CPU	Bus interface unit
1	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because the instruction queue buffer is vacant and the CPU is not using the bus.
2	Fetches operation code.	Fetches 1 odd address byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
3	(No fetching can be done, because there are no operands in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
4	Fetches operand (dd).	Fetches 1-byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
5	Waits for $\overline{E}$ to become "L", to read data.	
6	Reads data (D_L) into the data buffer when $\overline{E}$ becomes "L".	
7	Waits for $\overline{E}$ to become "L", to read data.	
8	Reads data (D_H) alone into the data buffer when $\overline{E}$ becomes "L".	
9	Modifies data.	Prefetches the instruction, because there are two vacant positions in the instruction queue buffer, and the CPU is not using the bus.
10	(Waits till the bus used by the bus interface unit is vacant.)	Fetches 1 byte worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
11	Writes data into the data buffer.	Waits till $\overline{E}$ becomes "L" to write data.
12	Fetches the next operation code.	Writes the contents of the data buffer (D_L) into the original address (odd address), when $\overline{E}$ becomes "L".
13	?	Waits till $\overline{E}$ becomes "L" to write data.
14	?	Writes the contents of the data buffer (D_H) into the original address (even address), when $\overline{E}$ becomes "L".

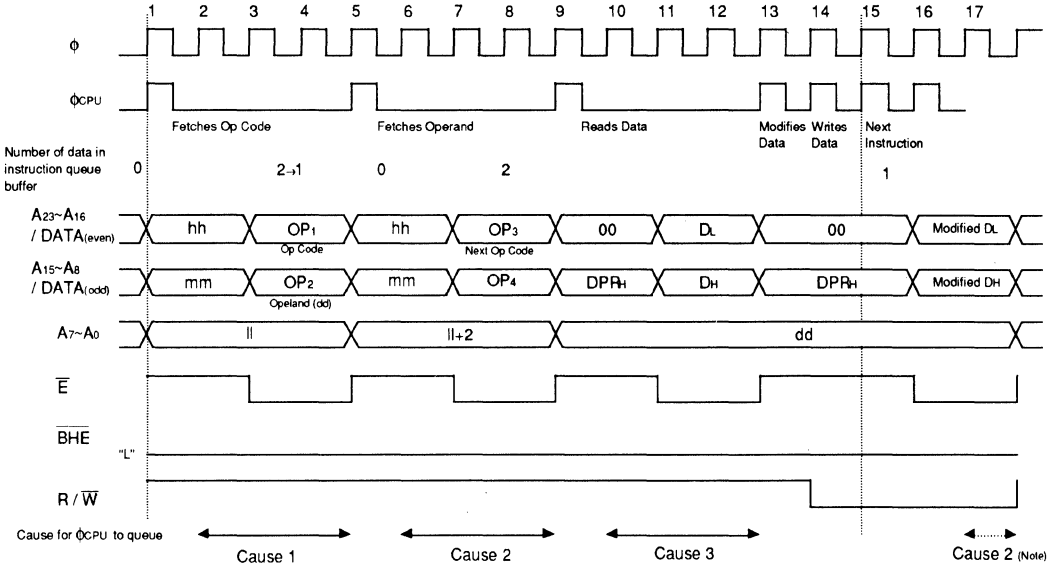
The external bus width becomes 8 bits when the "H" level is applied to the BYTE terminal. (The width of the internal bus is 16 bits, regardless of the level of the BYTE terminal.) When external ROM is used under this mode, the instruction can only be fetched byte by byte. (ϕ_2, ϕ_4, ϕ_{10} cycle) When external RAM is used, the data can likewise only be handled byte by byte. Accordingly, when data length selection flag m = 0 is selected, it takes time worth 2 cycles of the enable output $\overline{E}$ for data read and write. (ϕ_5 to ϕ_8, ϕ_{11} to ϕ_{14} cycle)

(Example 1-6) When external memory is accessed with wait by the wait bit

Conditions

- Number of data in the instruction queue buffer — 0
- ROM, RAM — External memory is used
- Data length selection flag m — "0" (16-bit length)
- BYTE terminal level — "L" (External bus width is 16 bits)
- Contents of lower order bytes (PC_L) of the program counter — Even
- Contents of the operand (dd) — Even

ϕ based execution sequence



Operation of the CPU and bus interface unit under various cycles

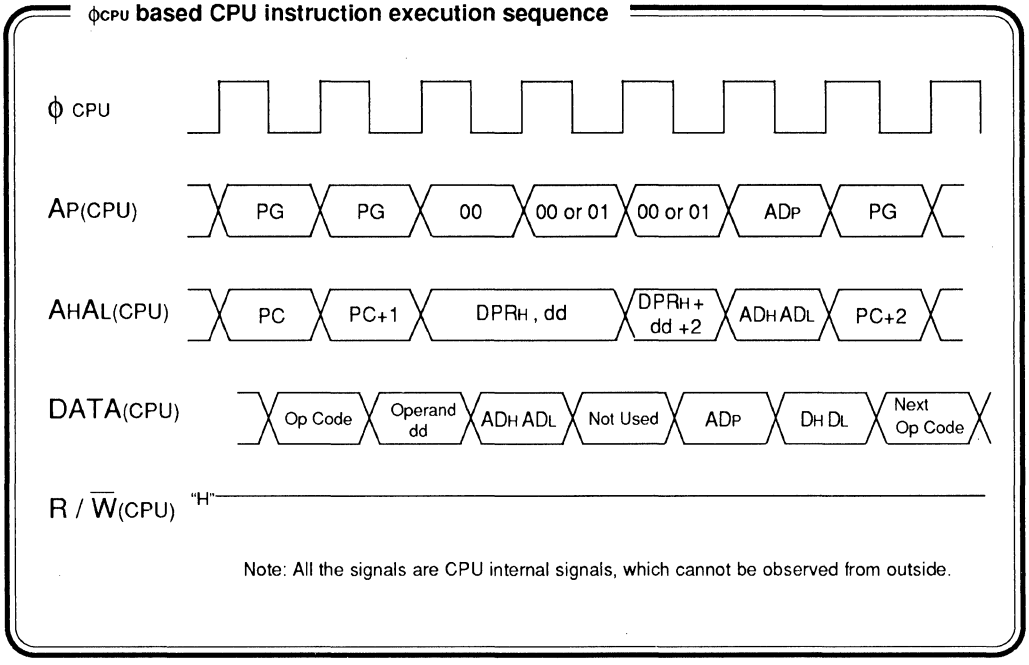
ϕ No.	CPU	Bus interface unit
1 2	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
3 4	Fetches the operation code .	Fetches 2 bytes worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
5 6	Fetches operand (dd).	Prefetches the instruction because the instruction queue buffer is vacant and the CPU is not using the bus.
7 8	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2 bytes worth of data into the instruction queue buffer when becomes "L".
9 10	Waits till $\overline{E}$ becomes "L" to write data.	
11 12	Reads data when $\overline{E}$ becomes "L".	
13	Modifies data.	
14	Writes data into the data buffer.	
15	Fetches the next operation code.	
16	?	Writes the contents of the data buffer into the original address (odd address), when $\overline{E}$ becomes "L".

Note. At the <- - -> part

* When the CPU does not use the bus, ϕ_{CPU} corresponds with ϕ.
* When the CPU uses the bus, the ϕ_{CPU} extends till the writing in the bus interface unit completes. (the ϕ_{16} to ϕ_{17} cycle)

The conditions are the same, except when wait is commanded by the wait bit (example 1-1). When accessing to the external memory, the cycle of enable output $\overline{E}$ becomes twice that for no-wait, and thus the ϕ_{CPU} wait time also becomes twice the cycle. (ϕ_3 to ϕ_4, ϕ_7 to ϕ_8, ϕ_{11} to ϕ_{12}, ϕ_{16} to ϕ_{17} cycle)

Example 2. LDA instruction / Direct indirect long addressing mode (DPR_L = 00_{16})

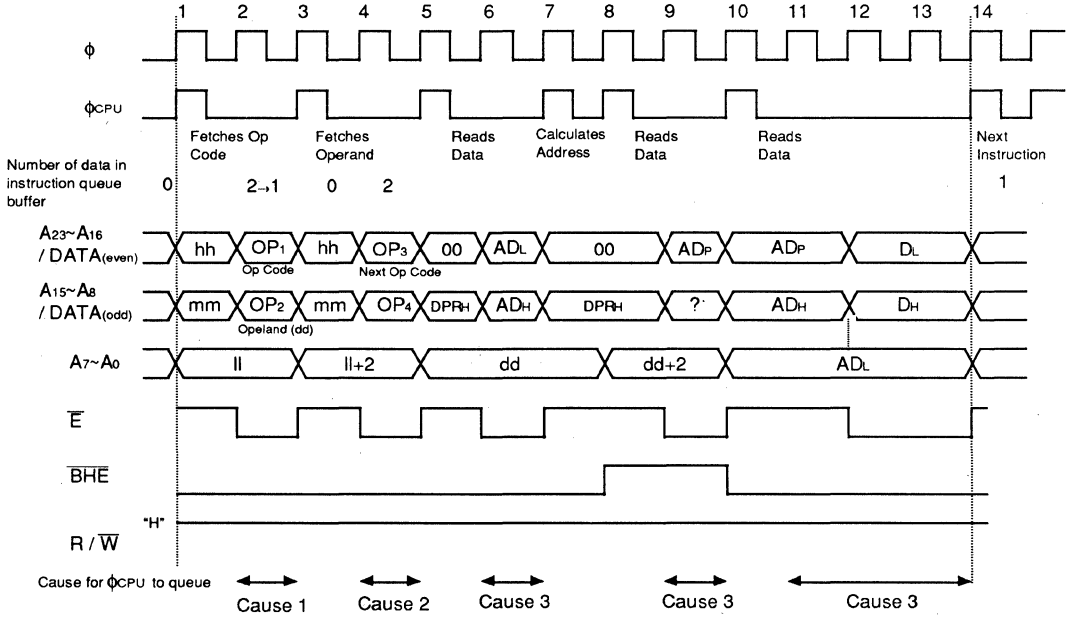


(Example 2-1) When the internal as well as the external memories are used together while wait is commanded by the wait bit.

Conditions

• Number of data in the instruction queue buffer	0
• Bank 0	Internal ROM, RAM are used
Bank 1 and after	External memory is used
• Data length selection flag m	"0" (16-bit length)
• BYTE terminal level	"L" (External bus width is 16 bits)
• Contents of lower order bytes (PC_L) of the program counter	Even
• Contents of the operand (dd)	Even
• Data indicated by the address AD_L	Even
AD_P	1 or more (bank 1 and after)

ϕ based execution sequence



Operation of the CPU and bus interface unit under various cycles

φ No.	CPU	Bus interface unit
1	(No fetching can be done, because there are no operation codes in the instruction queue buffer.)	Fetches the instruction, because instruction queue buffer is vacant and the CPU is not using the bus.
2	Fetches the operation code .	Fetches 2 bytes worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
3	Fetches operand (dd).	Prefetches the instruction because the instruction queue buffer is vacant and the CPU is not using the bus.
4	(Waits till the bus used by the bus interface unit becomes vacant.)	Fetches 2 bytes worth of data into the instruction queue buffer when $\overline{E}$ becomes "L".
5	Waits for $\overline{E}$ to becomes "L", to read data (AD_H AD_L) indicated by the address obtained by adding the contents of the operand (dd) and the DPR_L.	
6	Reads data when $\overline{E}$ becomes "L".	
7		Calculated address.
8	Waits for $\overline{E}$ to become "L", to read data (AD_P).	
9	Reads data when $\overline{E}$ becomes "L".	
10 11	Waits for $\overline{E}$ to become "L", to read the data (D_H D_L) at the address specified by AD_P AD_H AD_L.	
12 13	Reads data when $\overline{E}$ becomes "L".	

The above is the case when bank 1 and after are used by the external memory under the memory expansion mode. The currently executed program is in bank 0. The contents of the lower order bytes of the direct page register DPR_L is "00_{16}", so the direct pages are all in bank 0. The access to the outside (ϕ_{10} to ϕ_{13} cycle) alone is affected by the wait bit, and access to the internal memory is not affected by the bit.

APPENDIX A. CPU Instruction Execution Sequence for each Addressing Mode

The following are the CPU instruction execution sequences for each addressing mode. The execution sequences shown here describe the internal operation of the CPU. Therefore, the signals are all CPU internal signals, and cannot be observed from outside. The CPU internal operation, the actual execution time, and the relation between signals that can be externally checked are described in Chapter 6 "Instruction Execution Sequence".

The following are the signals and the symbols indicating the contents.

Symbol	Description
ϕ_{CPU}	CPU basic cycle
$AP_{(CPU)}$ $AHAL_{(CPU)}$ PG PC	Higher order 8 bits of the CPU internal address bus. Lower order 16 bits of the CPU internal address bus. Contents of the program bank register. Contents of the program counter. Others are data that indicates the address obtained as result of address calculation.
$DATA_{(CPU)}$	The CPU internal data bus. The signal is output with a half-cycle delay from the CPU internal address bus. The operation codes and the operands are fetched from the instruction buffer. They are not directly fetched from the memory indicated by the PG and PC of this cycle.
$R/\overline{W}_{(CPU)}$	Becomes "L" when the CPU writes data into the data buffer of the bus interface unit.

The accumulator used in the above instructions in the CPU instruction execution sequence is accumulator A. When accumulator B is used, the execution cycle will have the two cycles of a "42_{16}" that indicates accumulator B, and an internal processing cycle added at the front. (See the figure in the next page.)

The number of ϕ_{CPU} cycles differs in the addressing mode that uses the direct page register, according to whether the lower order 8 bits (DPR_L) are "00_{16}". The number of cycles when $DPR_L = 00_{16}$ is 1 cycle (address calculation cycle) less than when $DPR_L \neq 00_{16}$.

The number of cycles differs in the PSH and PUL instructions according to the number and type of registers placed in (taken out of) the stack.

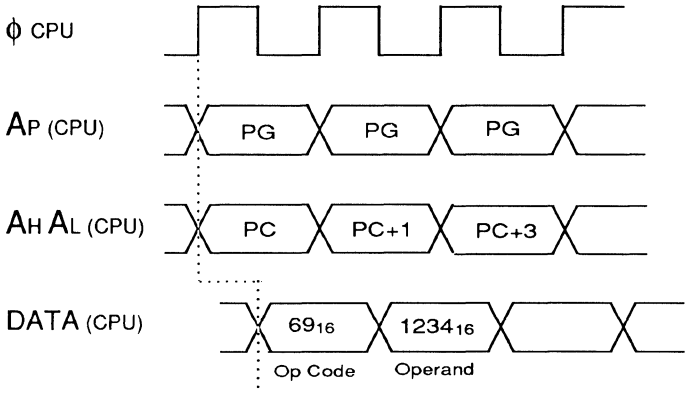
The number of cycles differs in the block transmission instruction (MVN, MVP), according to the number of the data transmitted.

Variation of the execution cycles according to the accumulator used

ADC Instruction / Immediate addressing mode

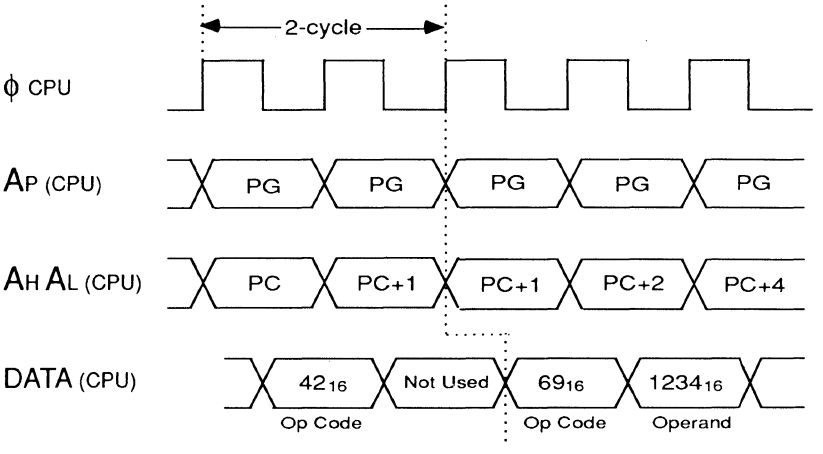
<<When accumulator A is used>>

Mnemonic : ADC A,#1234H Machine code : 69_{16} 34_{16} 12_{16}



<<When accumulator B is used>>

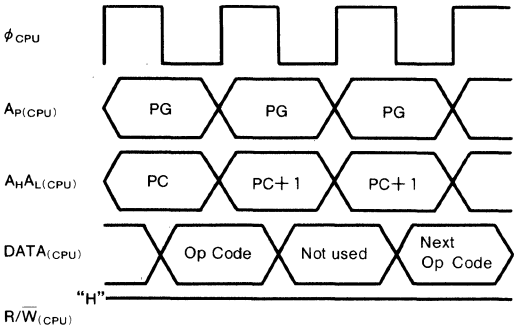
Mnemonic : ADC B,#1234H Machine code : 42_{16} 69_{16} 34_{16} 12_{16}



Implied

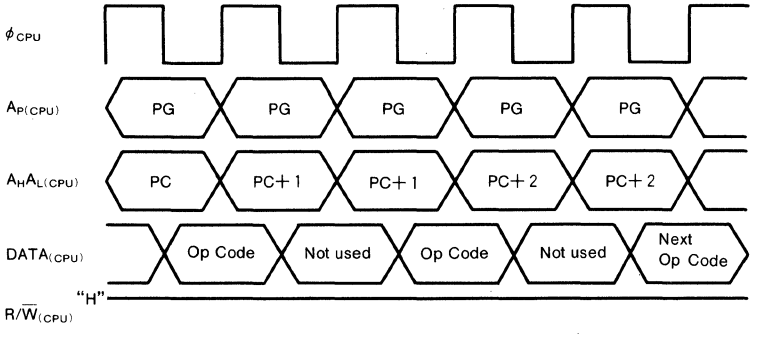
Instruction : CLC, CLI, CLM, CLV, DEX, DEY, INX, INY, NOP, SEC, SEI, SEM, TAD, TAS, TAX, TAY, TDA, TSA, TSX, TXA, TXS, TXY, TYA, TYX

Timing :



Instruction : TBD, TBS, TBX, TBY, TDB, TSD, TXB, TYB

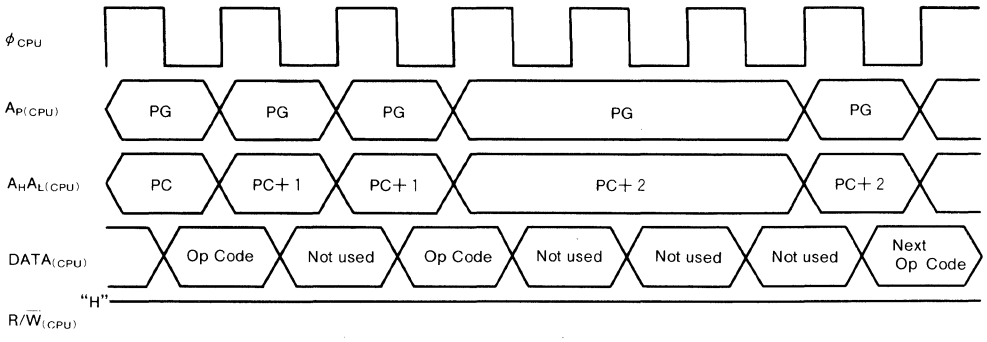
Timing :



Implied

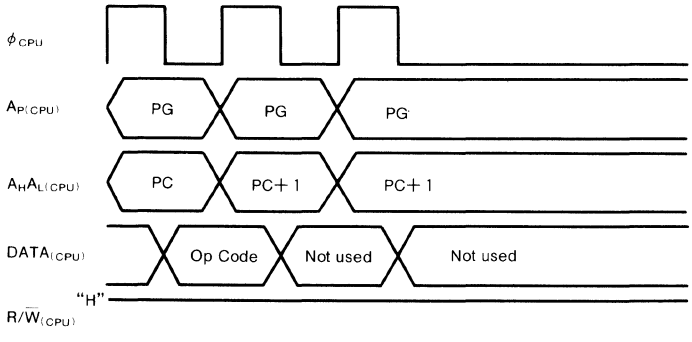
Instruction X A B

Timing :



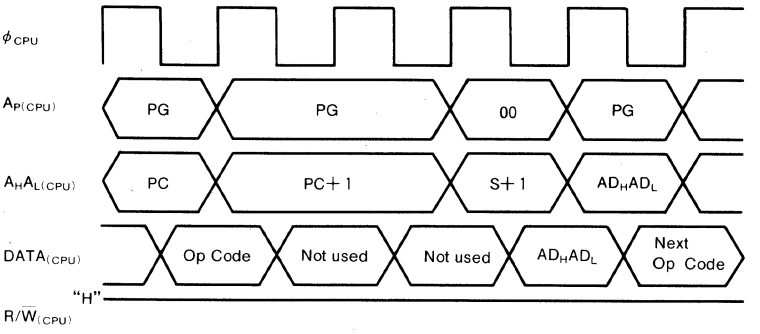
Instruction : S T P , W I T

Timing :



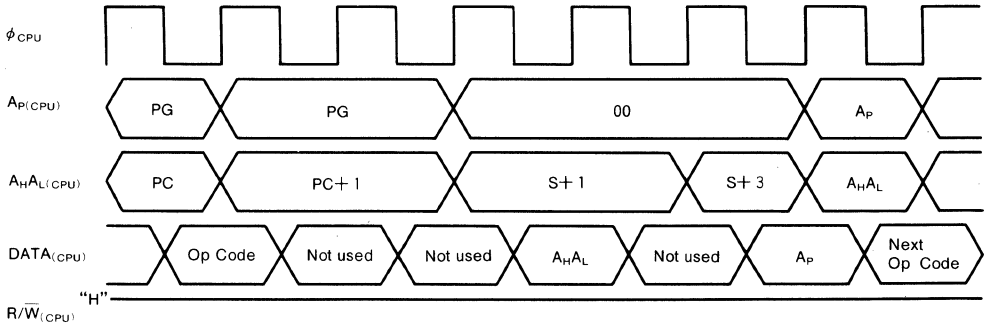
Instruction : RTS

Timing :



Instruction : RTL

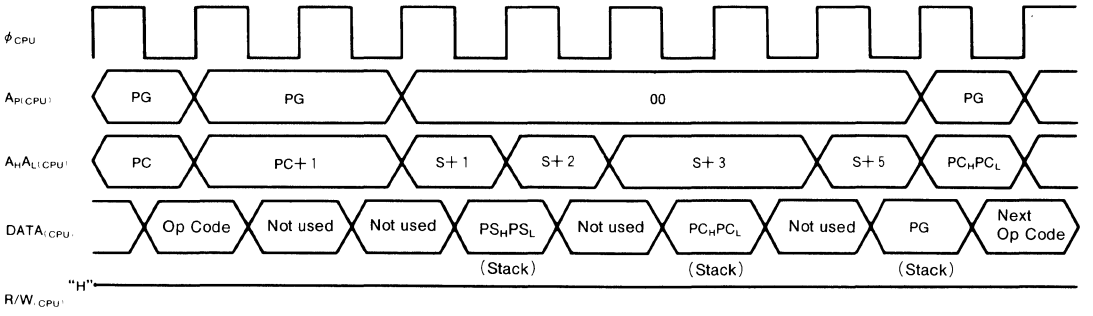
Timing :



Implied

Instruction : R T I

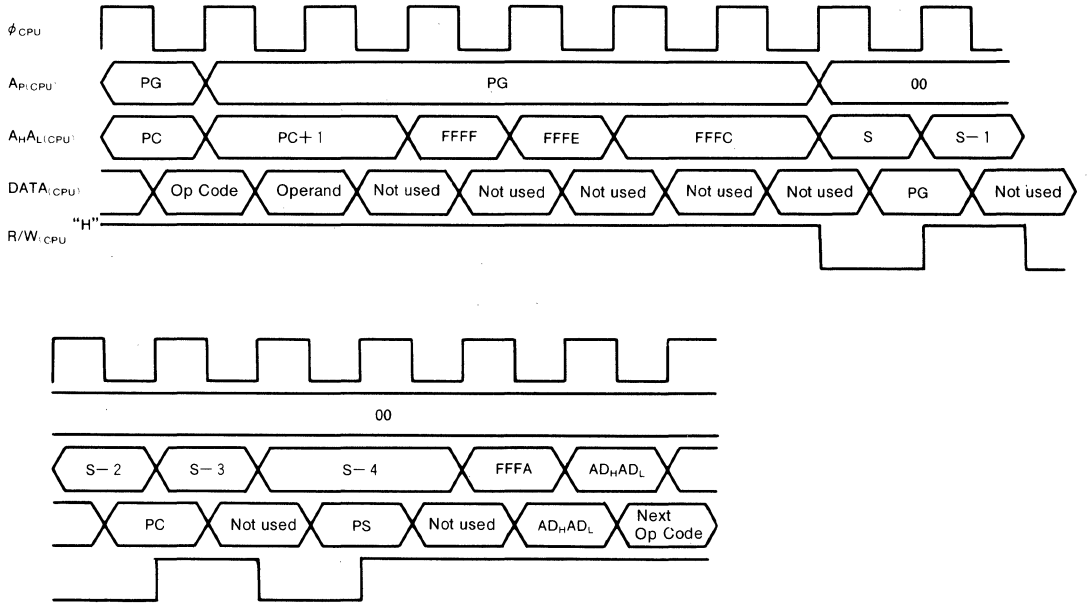
Timing :



Implied

Instruction : BRK

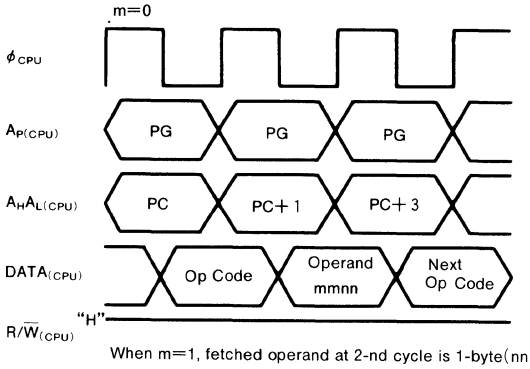
Timing :



Immediate

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

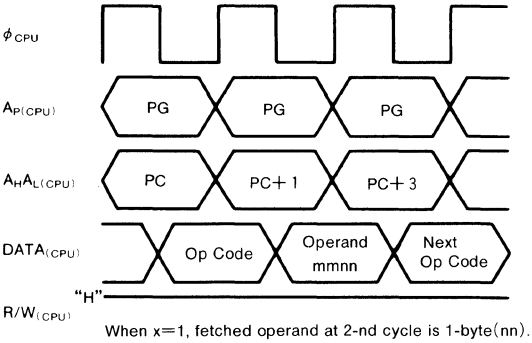
Timing :



When m=1, fetched operand at 2-nd cycle is 1-byte(nn).

Instruction : LDX, LDY, CPX, CPY

Timing :

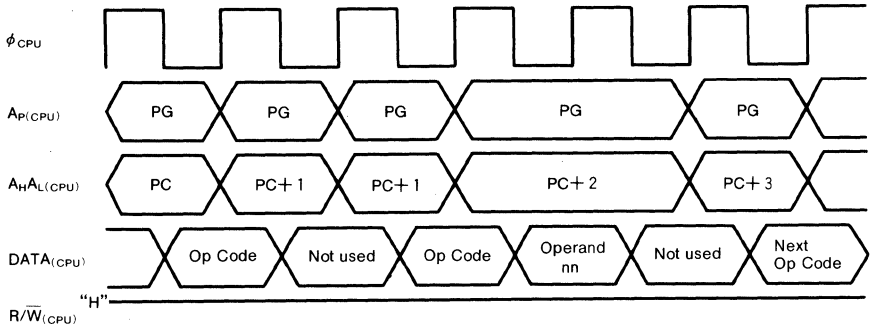


When x=1, fetched operand at 2-nd cycle is 1-byte(nn).

Immediate

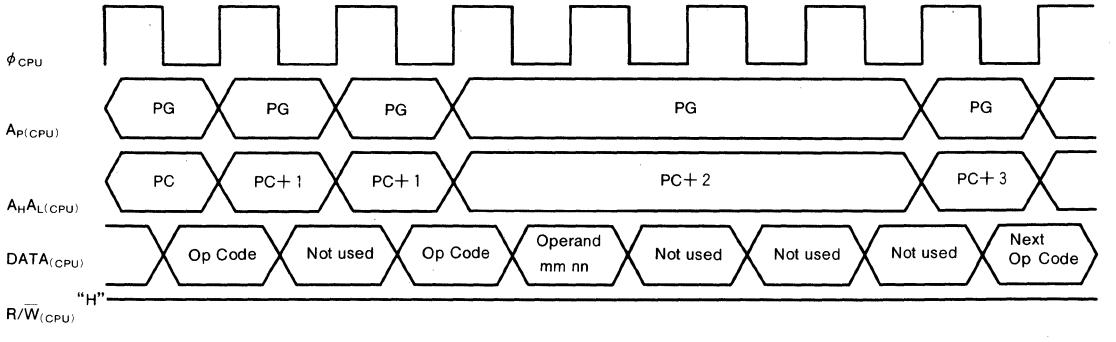
Instruction : LDT

Timing :



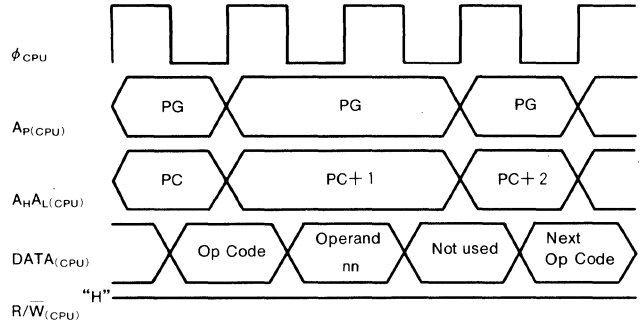
Instruction : RLA

Timing :



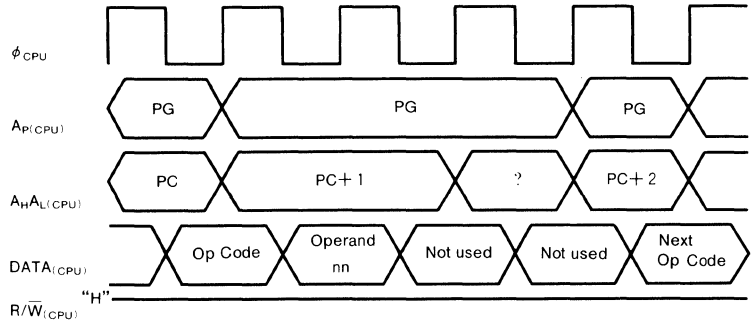
Instruction : S E P

Timing :



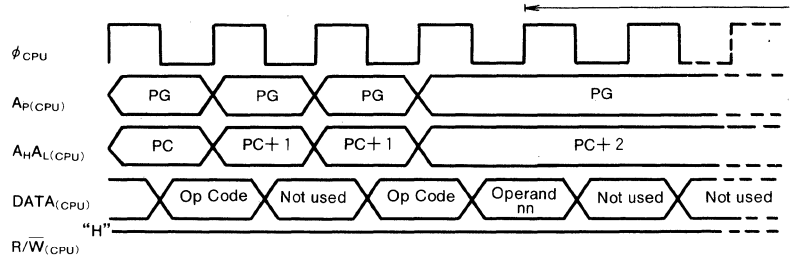
Instruction : C L P

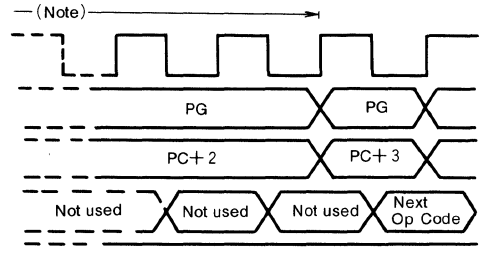
Timing :



Instruction : D I V, MPY

Timing :



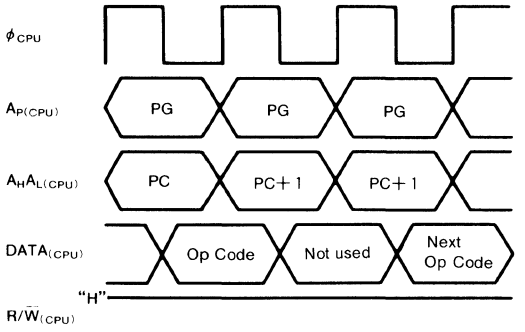


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Accumulator

Instruction : ASL, DEC, INC, LSR, ROL, ROR

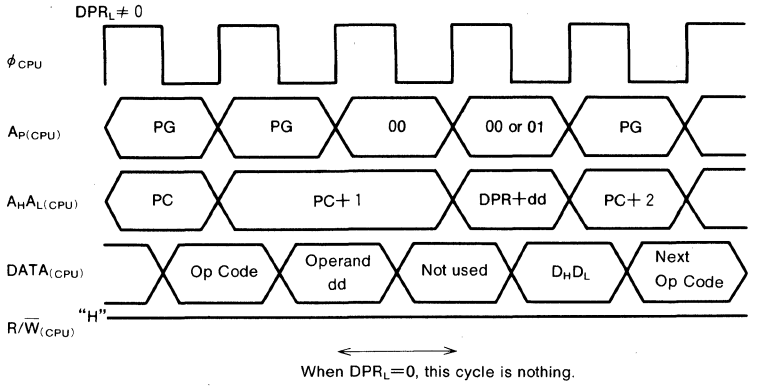
Timing :



Direct

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

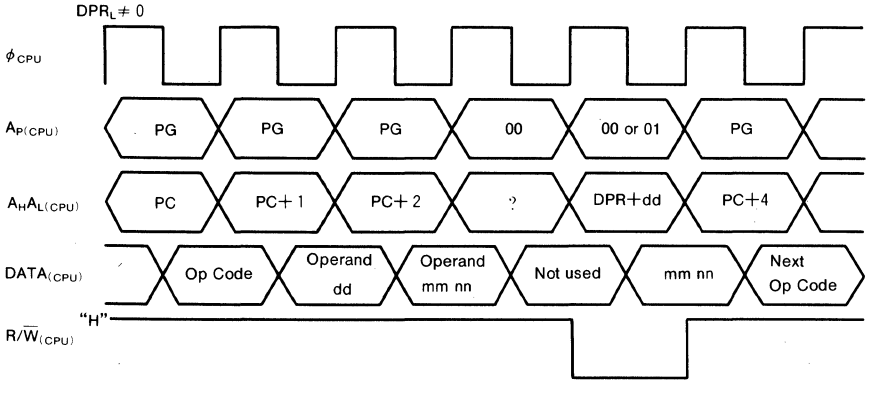
Timing :



When DPR_L=0, this cycle is nothing.

Instruction : LDM

Timing :

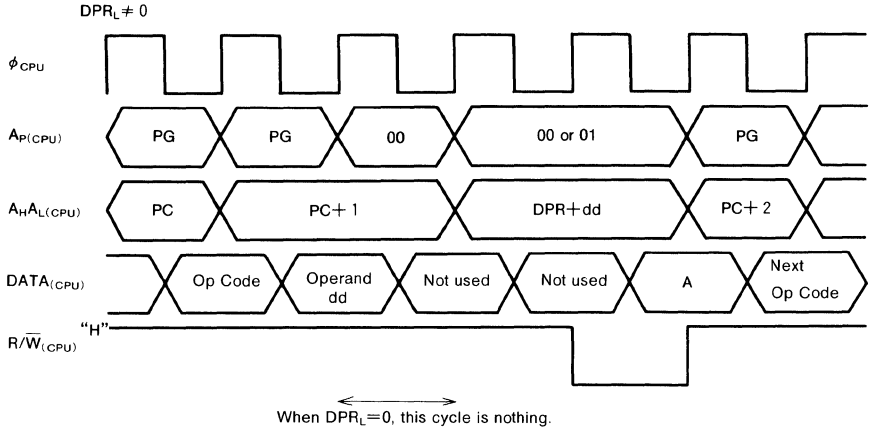


When DPR_L=0, this cycle is nothing.

Direct

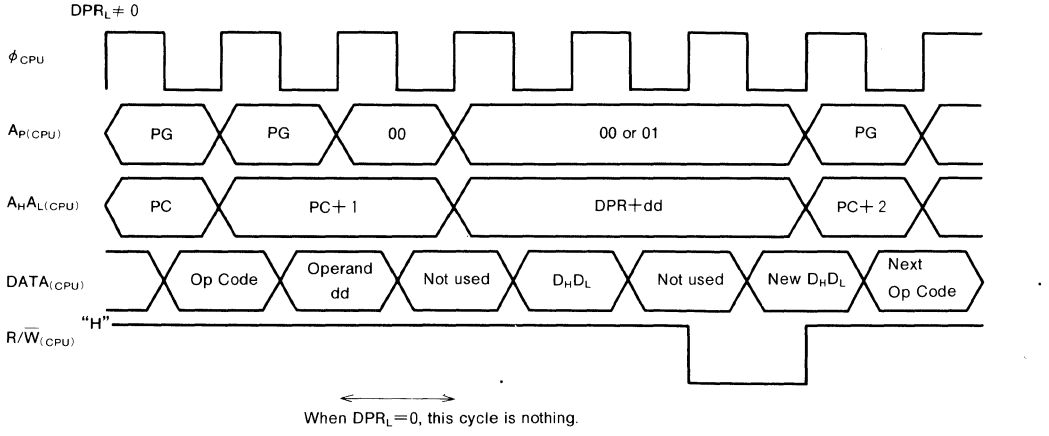
Instruction : S T A, S T X, S T Y

Timing :



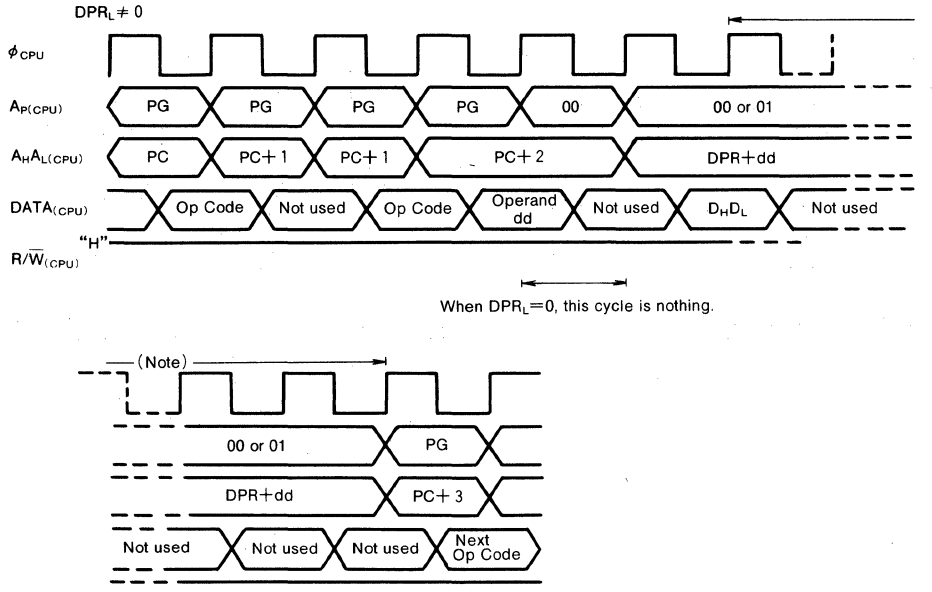
Instruction : A S L, D E C, I N C, L S R, R O L, R O R

Timing :



Instruction : D I V, M P Y

Timing :

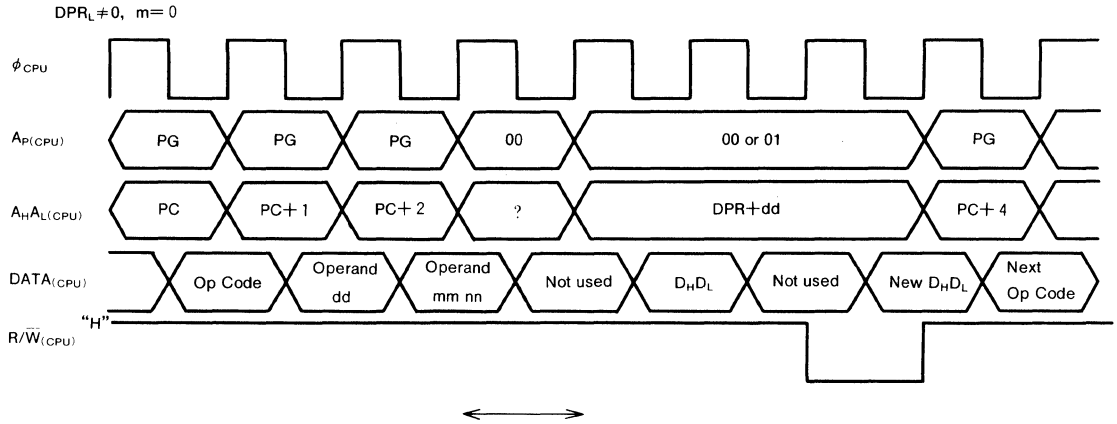


MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Direct Bit

Instruction : CLB, SEB

Timing :

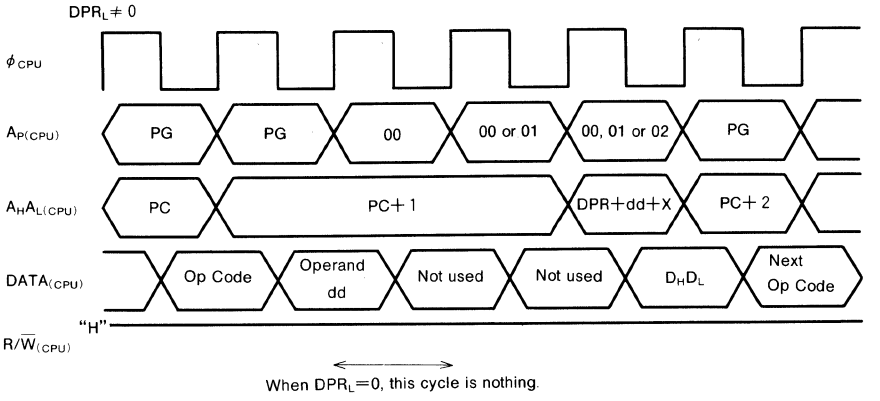


When DPR$_L$=0, this cycle is nothing.
When m=1, fetched operand at 3-rd cycle is 1-byte(nn).

Direct Indexed X

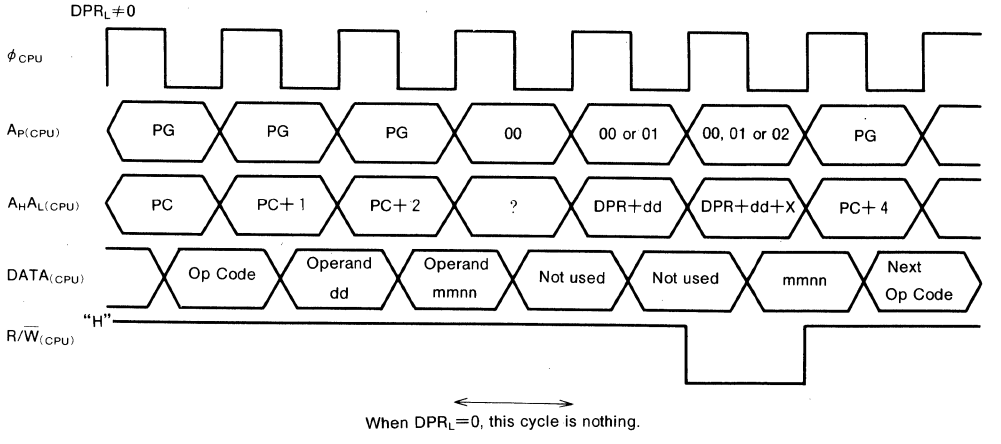
Instruction : ADC, AND, CMP, EOR, LDA, LDY, ORA, SBC

Timing :



Instruction : LDM

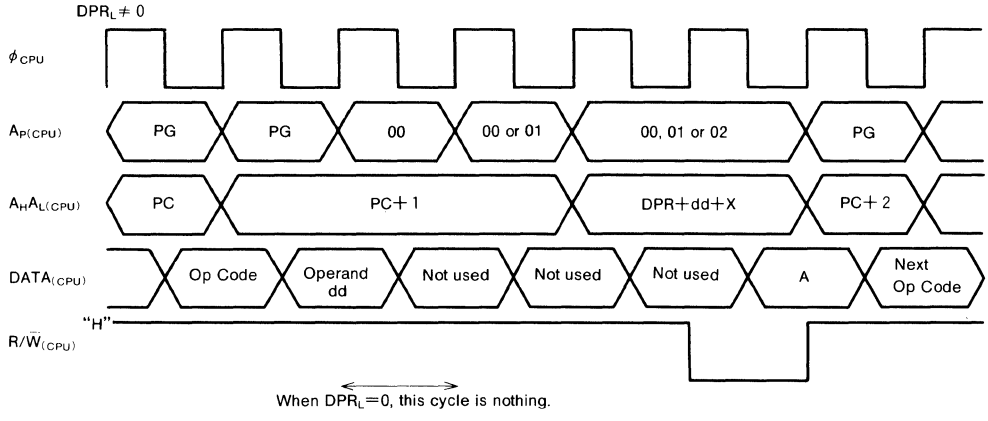
Timing :



Direct Indexed X

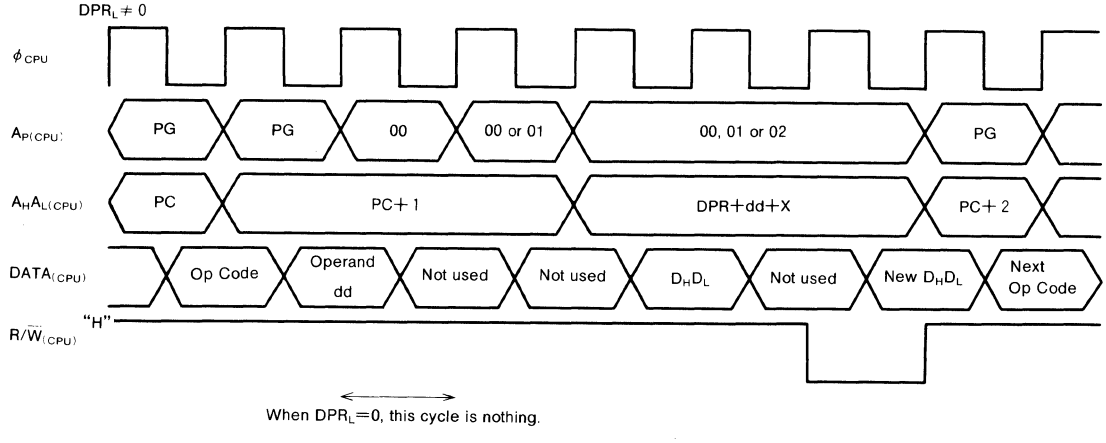
Instruction : STA, STY

Timing :



Instruction : ASL, DEC, INC, LSR, ROL, ROR

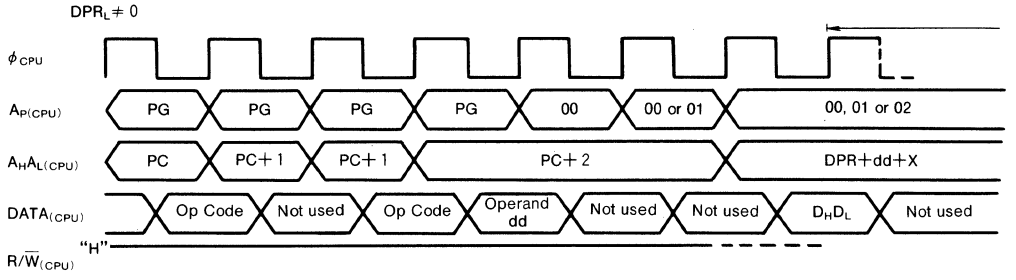
Timing :

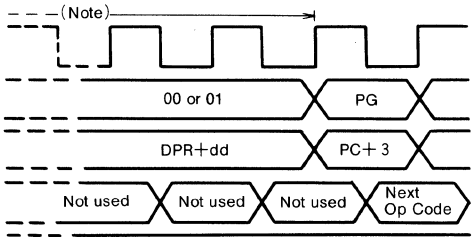


Direct Indexed X

Instruction : D I V, M P Y

Timing :



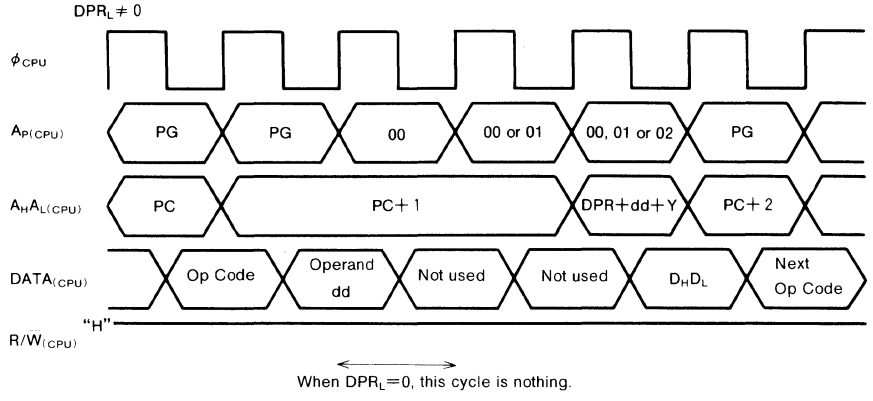


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Direct Indexed Y

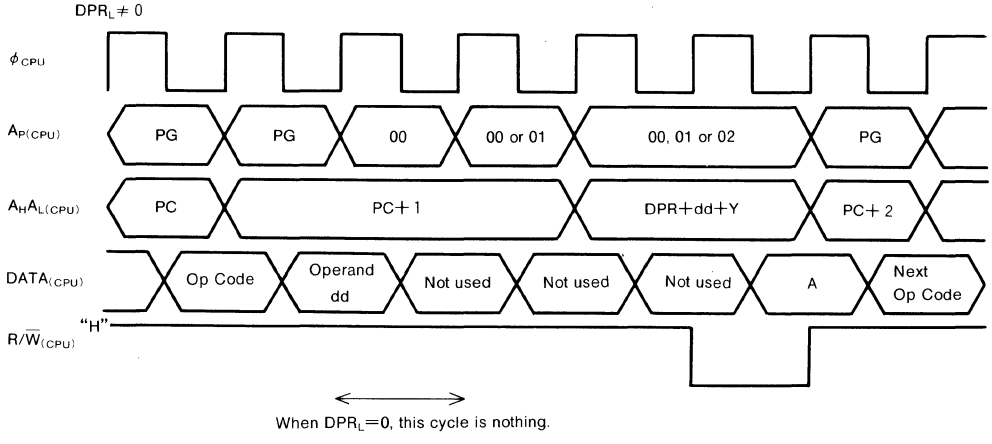
Instruction : LDX

Timing :



Instruction : STX

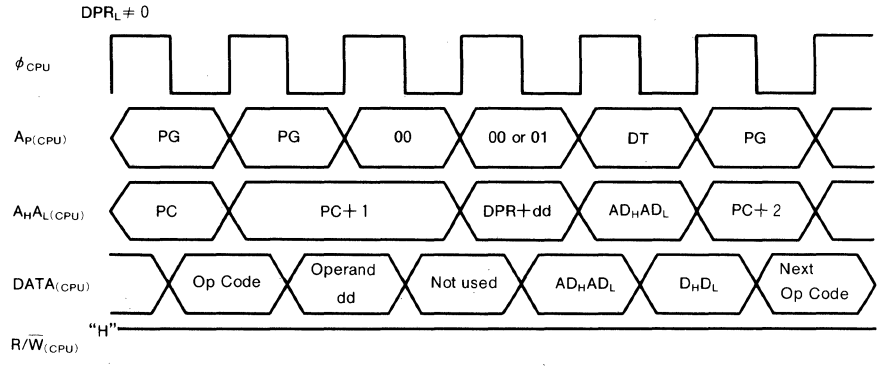
Timing :



Direct Indirect

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

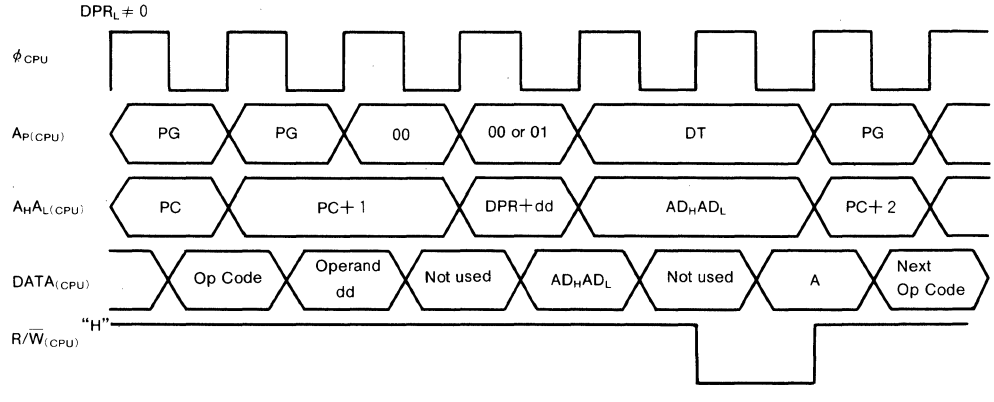
Timing :



When DPR$_L$=0, this cycle is nothing.

Instruction : STA

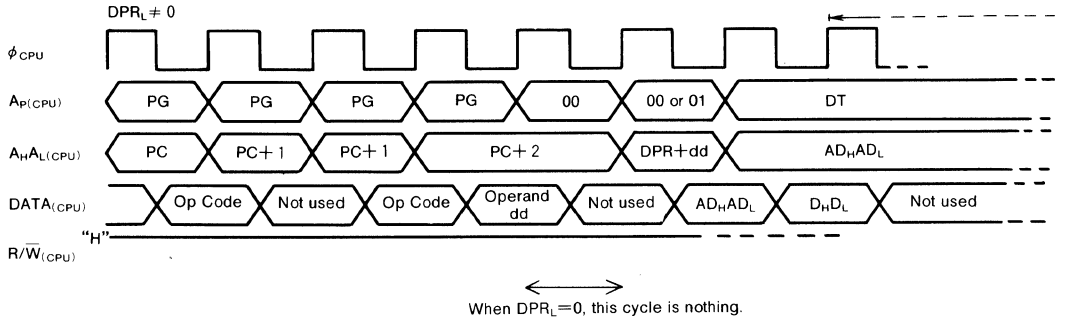
Timing :

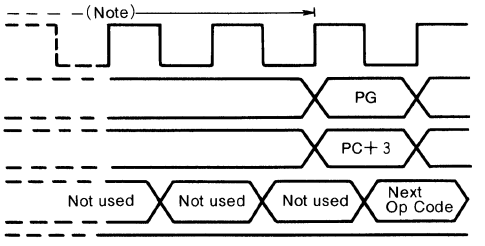


When DPR$_L$=0, this cycle is nothing.

Instruction : D I V, MPY

Timing :



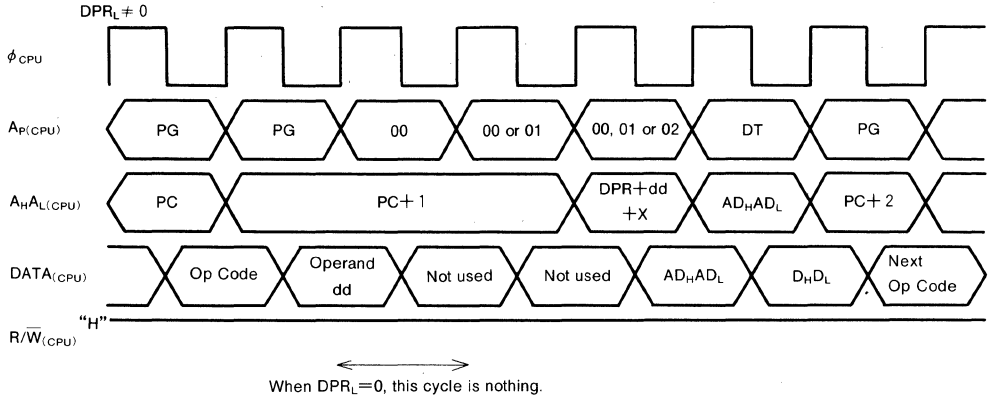


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Direct Indexed X Indirect

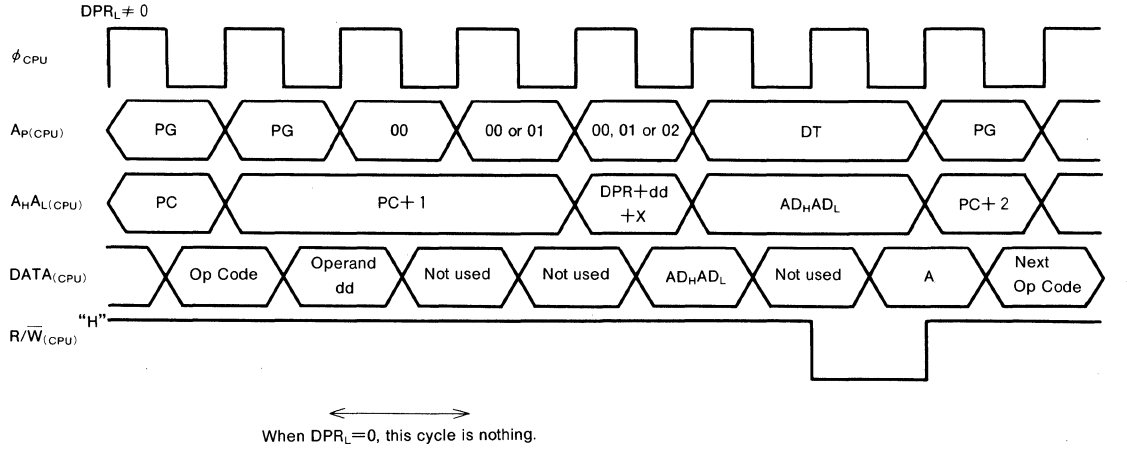
Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

Timing :



Instruction : STA

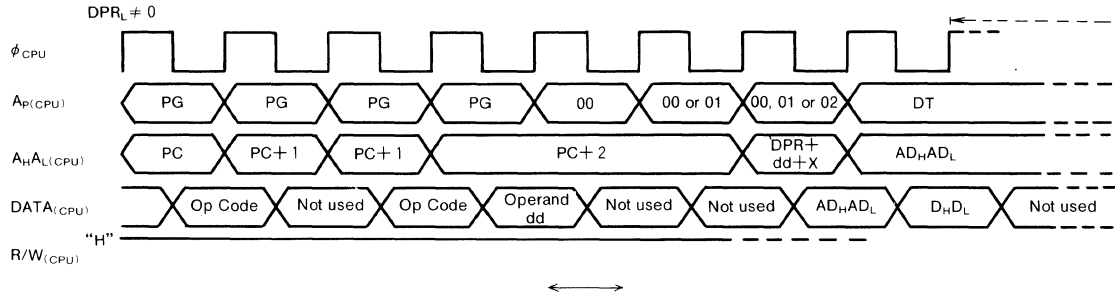
Timing :

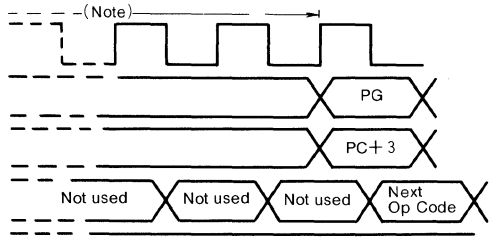


Direct Indexed X Indirect

Instruction : D I V, MPY

Timing :



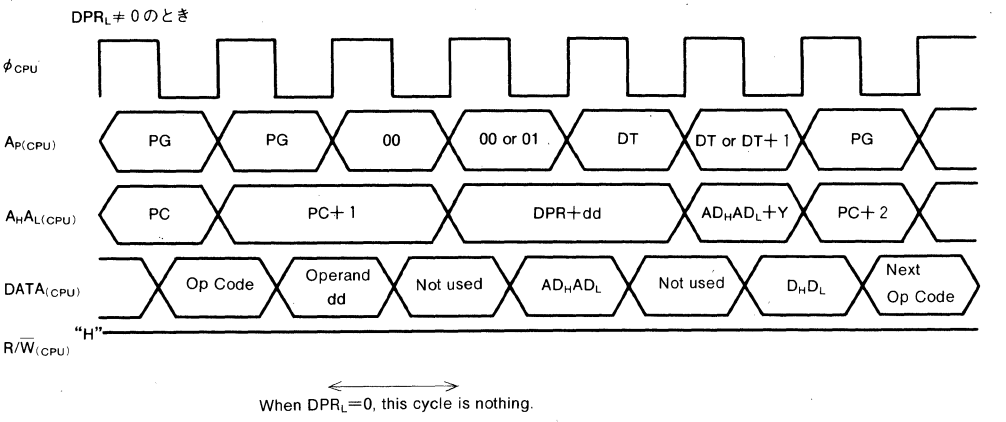


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Direct Indirect Indexed Y

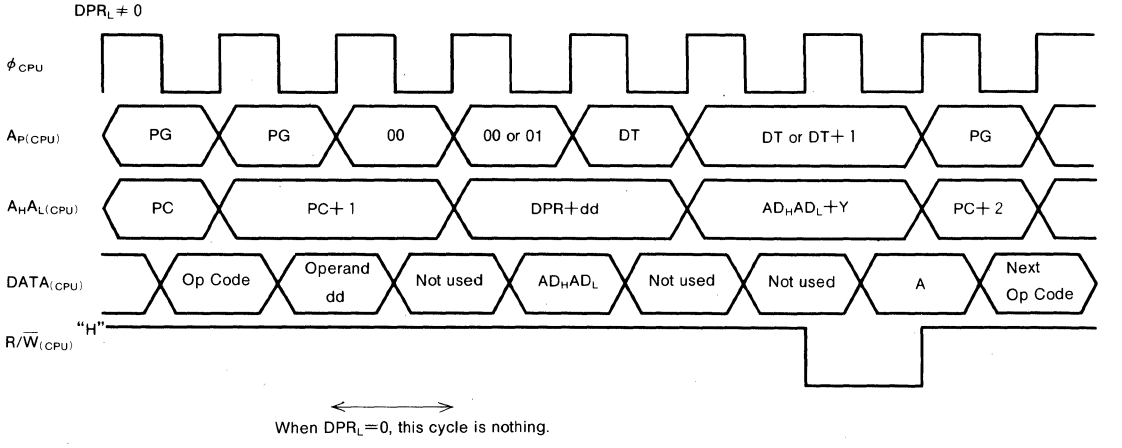
Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

Timing :



Instruction : STA

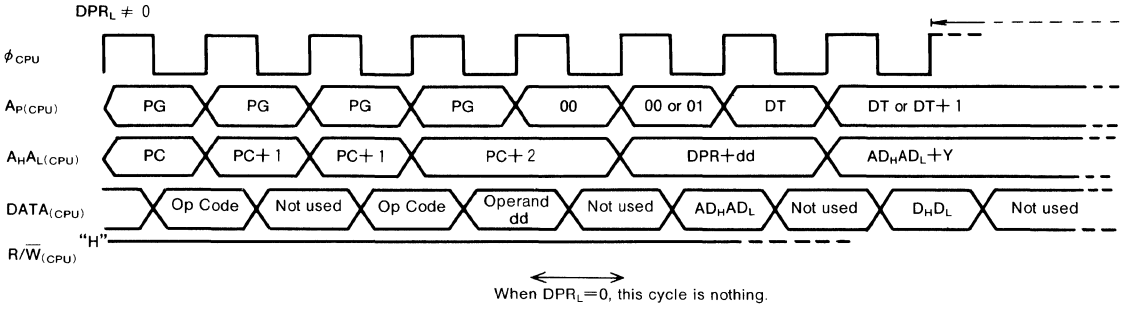
Timing :

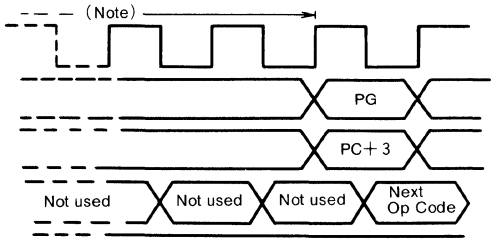


Direct Indirect Indexed Y

Instruction : D I V, M P Y

Timing :



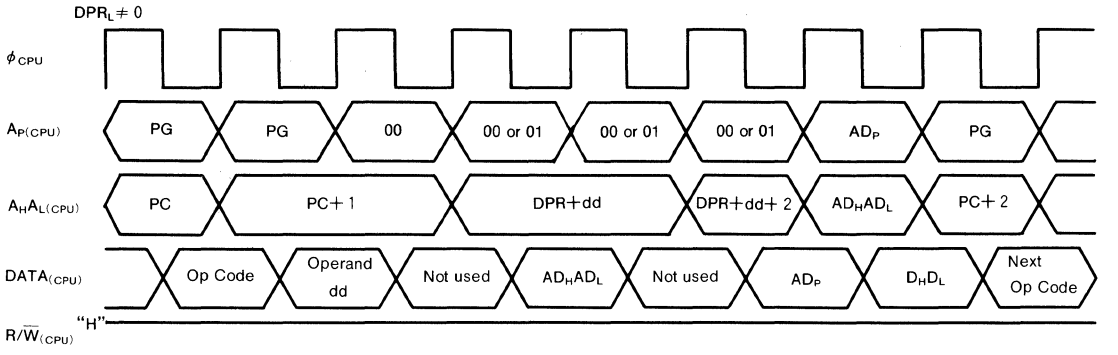


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Direct Indirect Long

Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

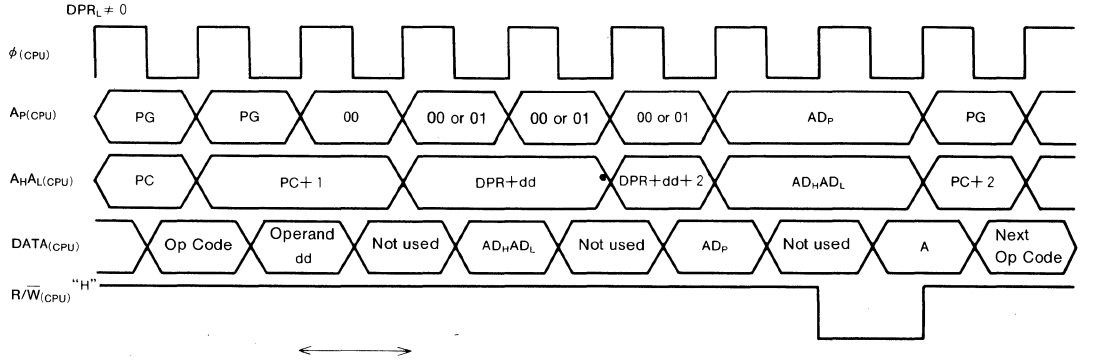
Timing :



When DPR$_L$=0, this cycle is nothing.

Instruction : STA

Timing :

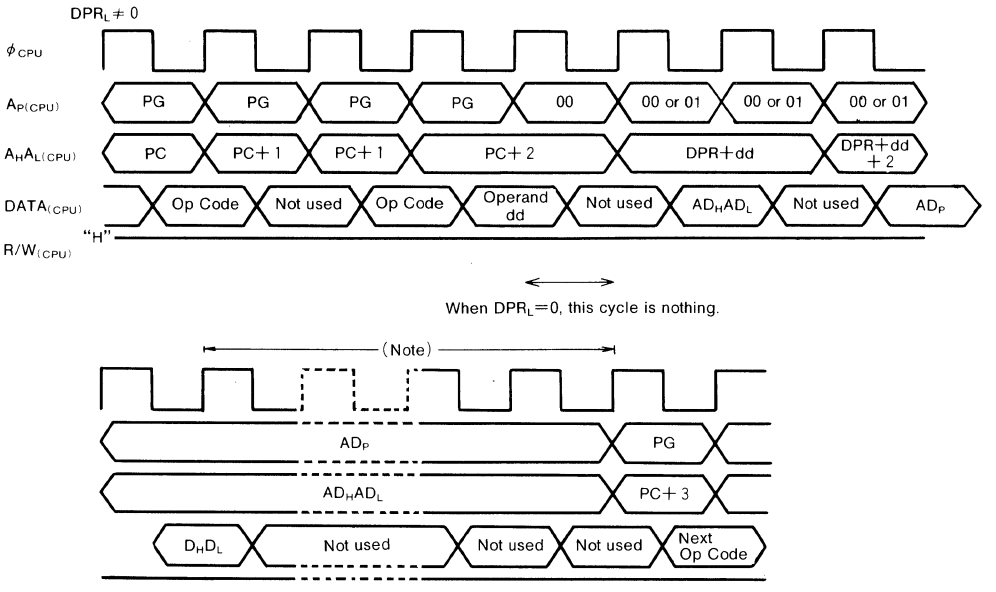


When DPR$_L$=0, this cycle is nothing.

Direct Indirect Long

Instruction : D I V, M P Y

Timing :

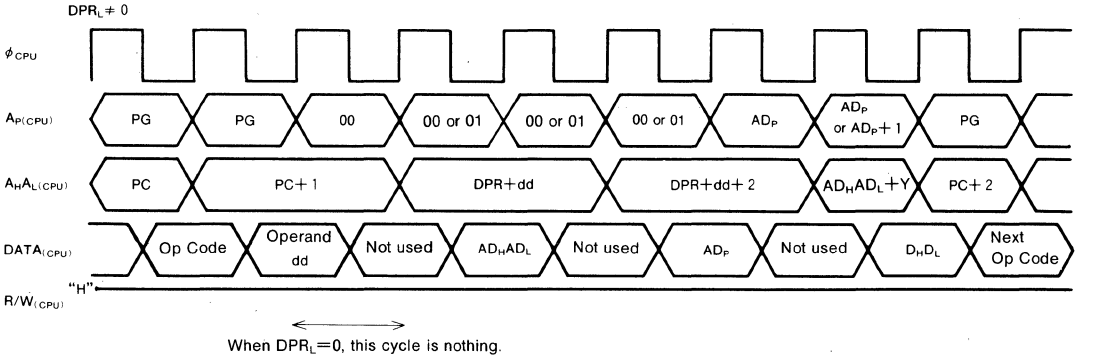


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Direct Indirect Long Indexed Y

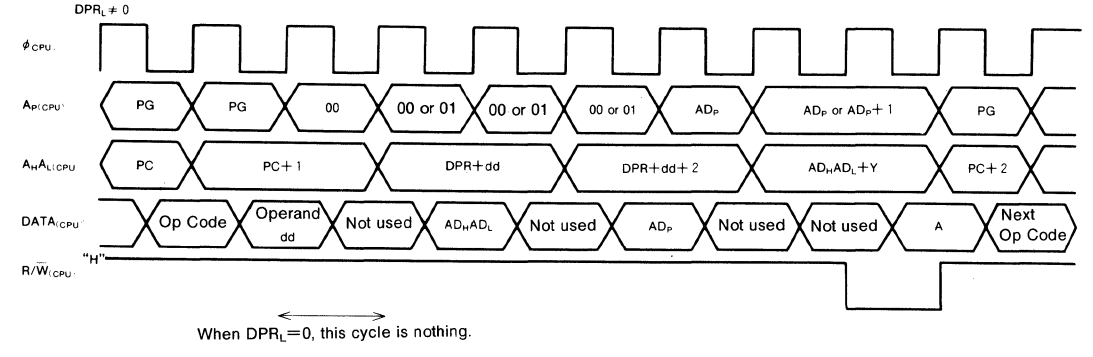
Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

Timing :



Instruction : STA

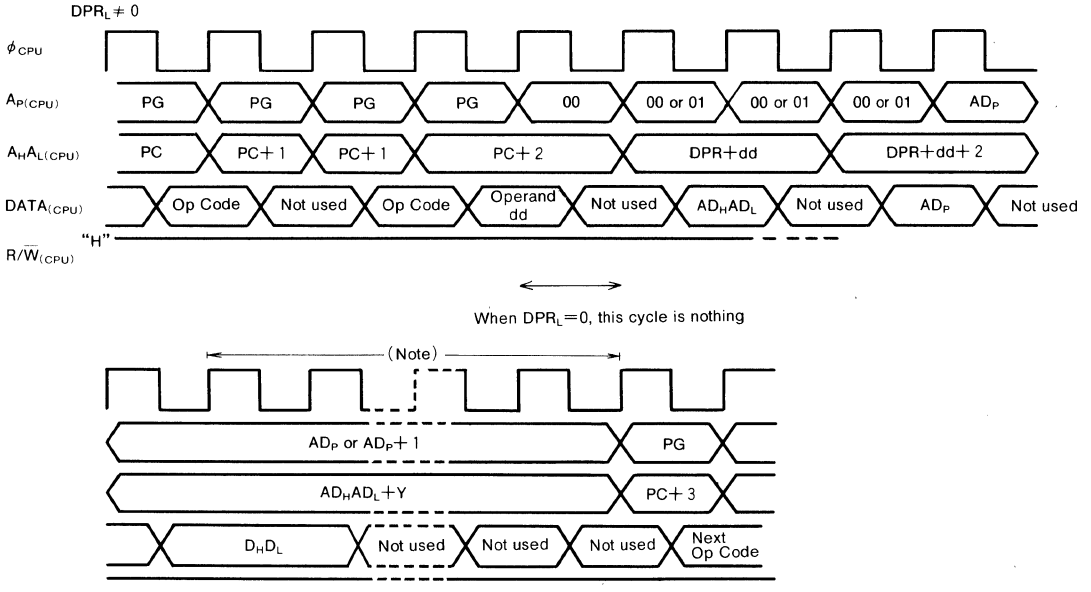
Timing :



Direct Indirect Long Indexed Y

Instruction : D I V, M P Y

Timing :

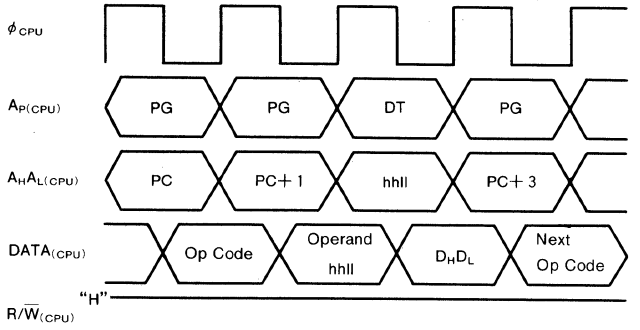


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Absolute

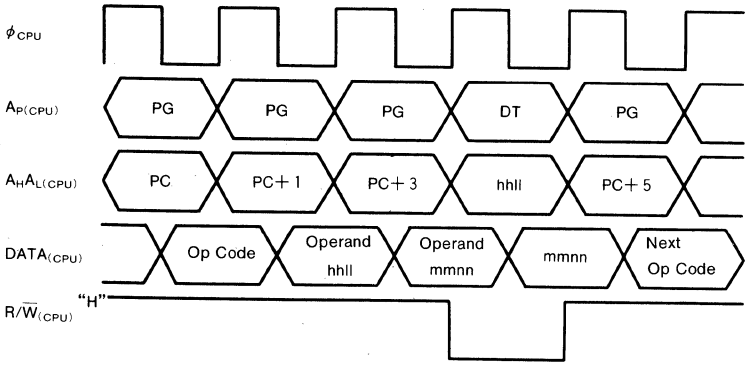
Instruction : ADC, AND, CMP, CPX, CPY, EOR, LDA, LDX, LDY, ORA, SBC

Timing :



Instruction : LDM

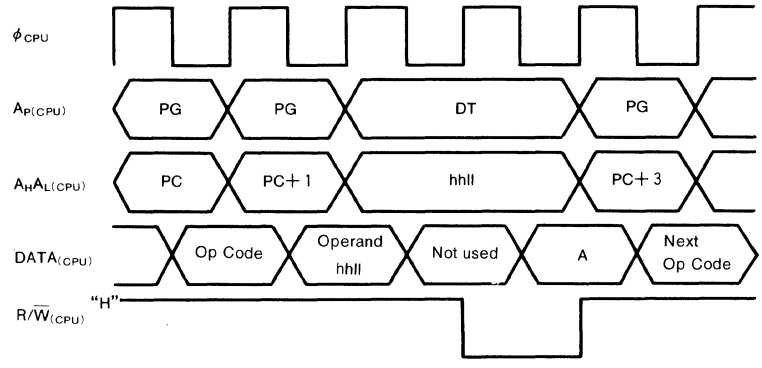
Timing :



Absolute

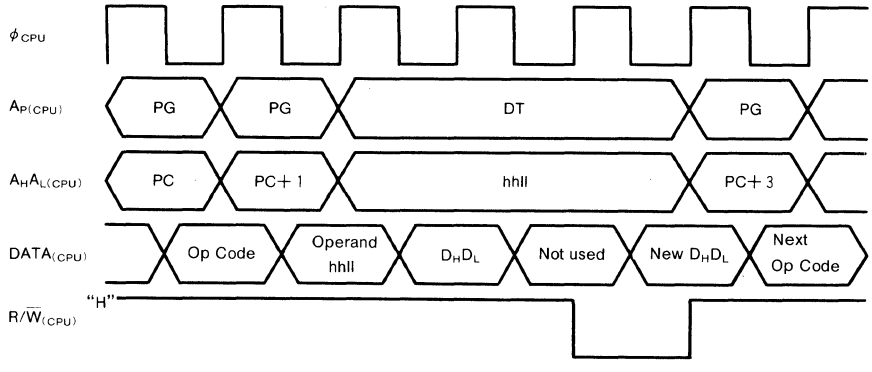
Instruction : S T A, S T X, S T Y

Timing :



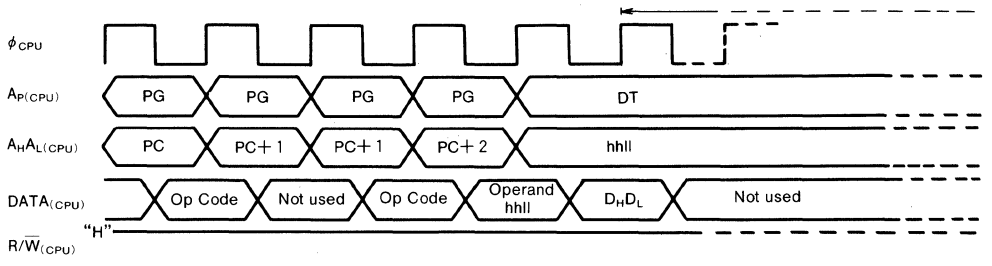
Instruction : A S L, D E C, I N C, L S R, R O L, R O R

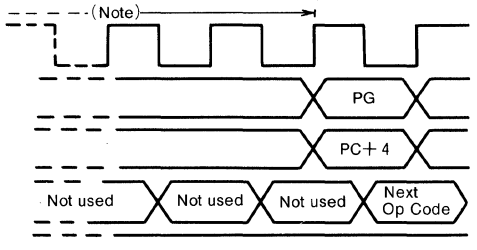
Timing :



Instruction : D I V, MPY

Timing :



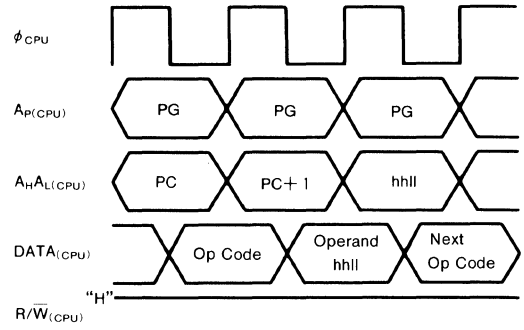


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Absolute

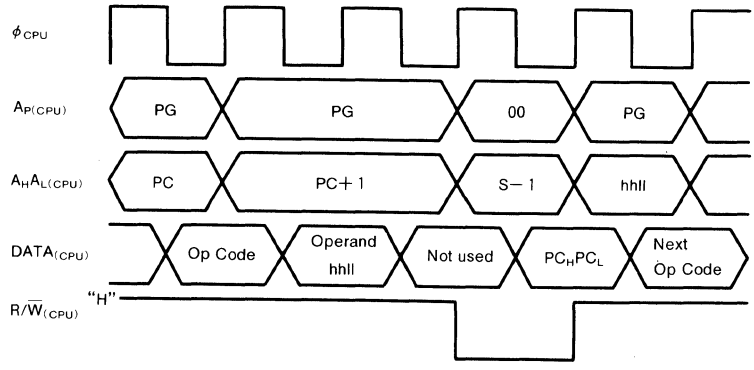
Instruction : JMP

Timing :



Instruction : JSR

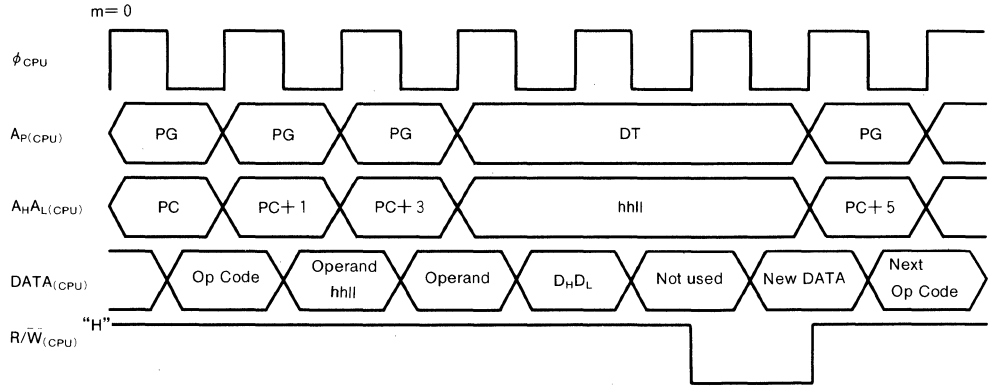
Timing :



Absolute Bit

Instruction : C L B, S E B

Timing :

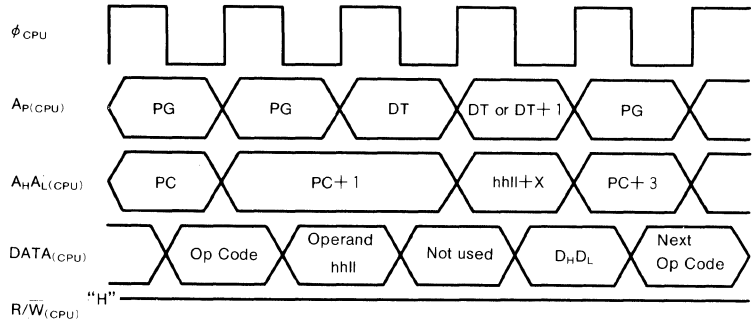


When m=1, fetched operand at 3-rd cycle is 1-byte(nn)

Absolute Indexed X

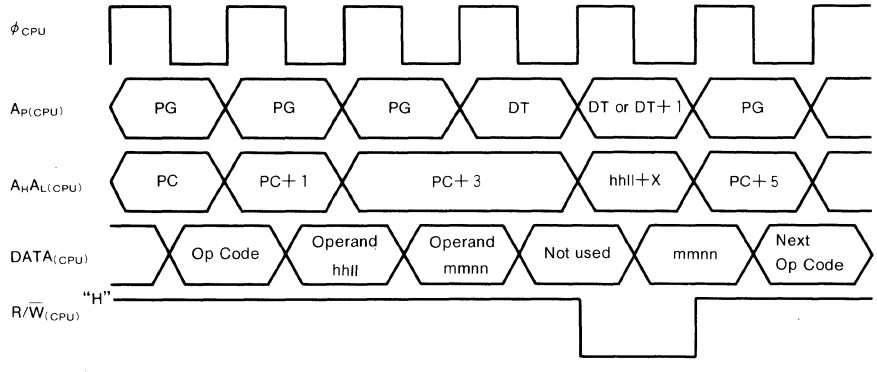
Instruction : ADC, AND, CMP, EOR, LDA, LDY, ORA, SBC

Timing :



Instruction : LDM

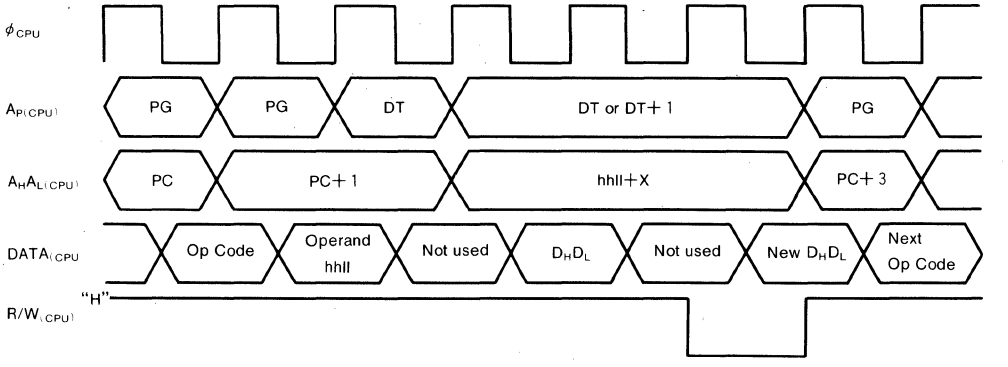
Timing :



Absolute Indexed X

Instruction : A S L, D E C, I N C, L S R, R O L, R O R

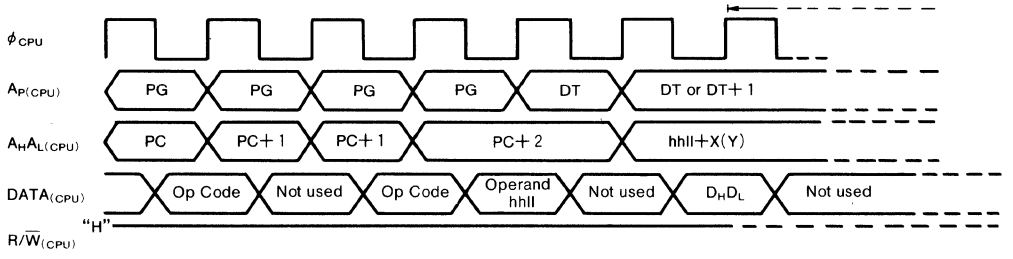
Timing :

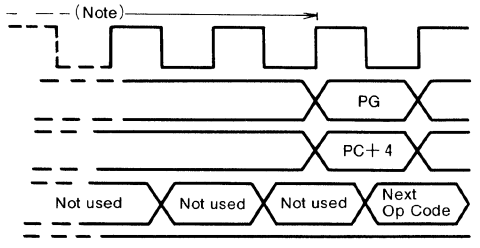


Absolute Indexed X
Absolute Indexed Y

Instruction : D I V, M P Y

Timing :



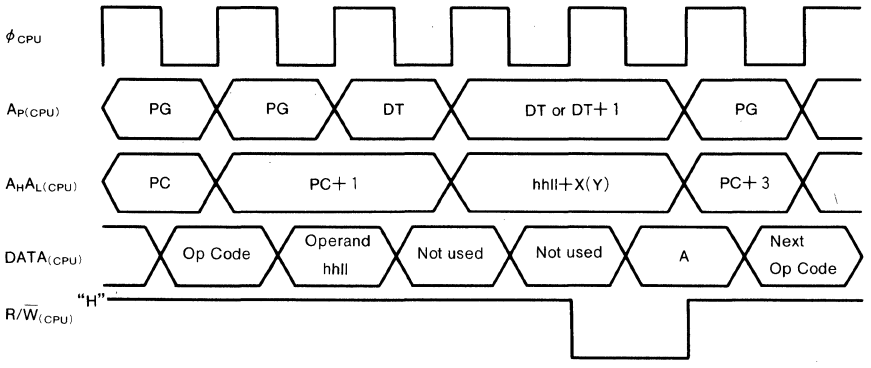


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Absolute Indexed X
Absolute Indexed Y

Instruction : STA

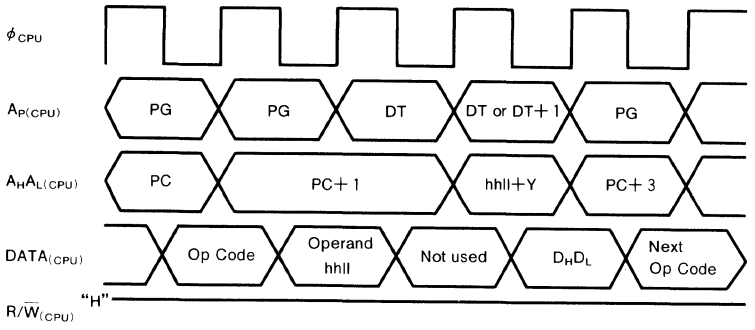
Timing :



Absolute Indexed Y

Instruction : ADC, AND, CMP, EOR, LDA, LDX, ORA, SBC

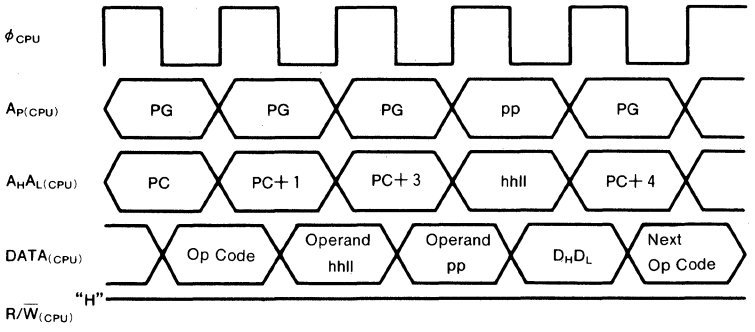
Timing :



Absolute Long

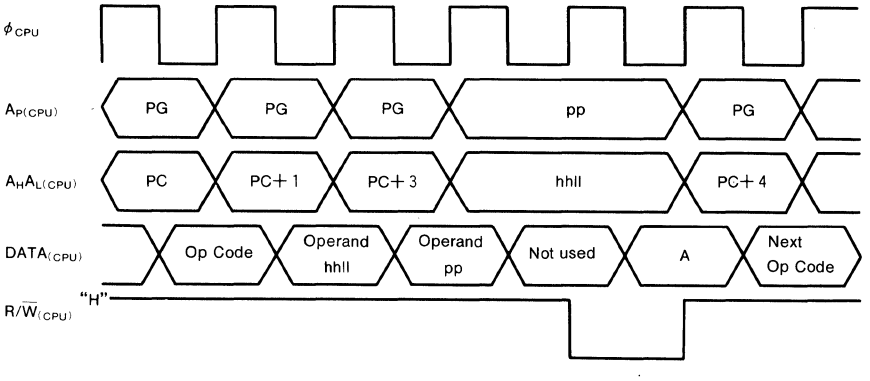
Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

Timing :



Instruction : STA

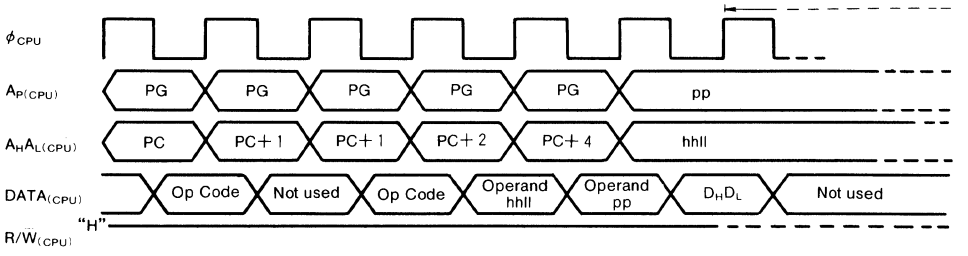
Timing :

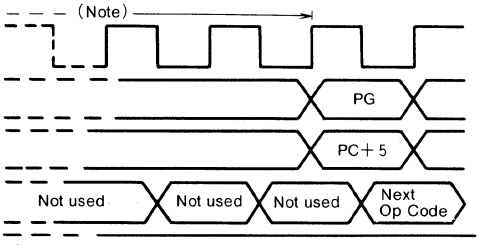


Absolute Long

Instruction : D I V, M P Y

Timing :



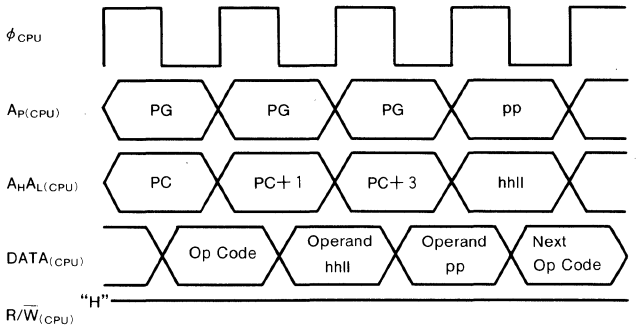


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Absolute Long

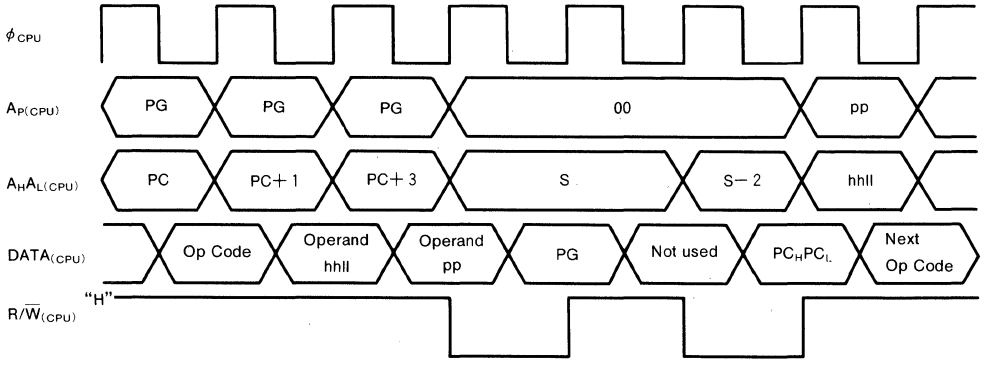
Instruction : JMP

Timing :



Instruction : JSR

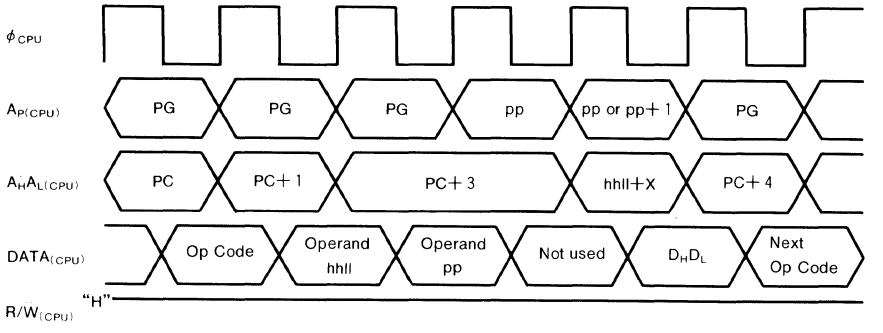
Timing :



Absolute Long Indexed X

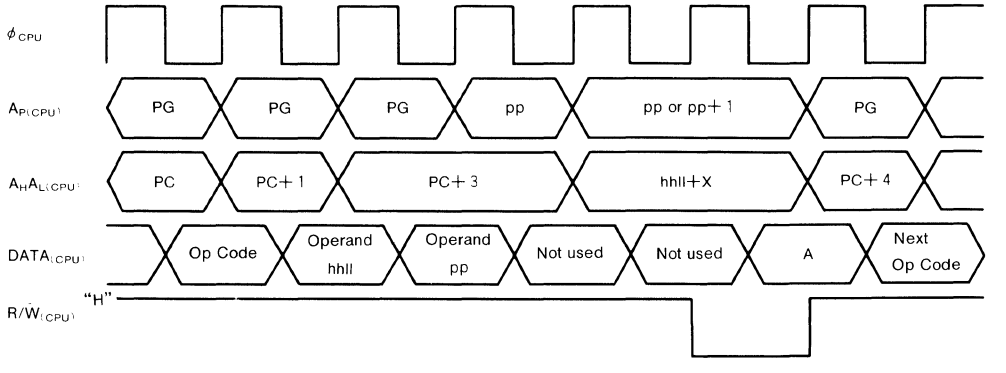
Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

Timing :



Instruction : STA

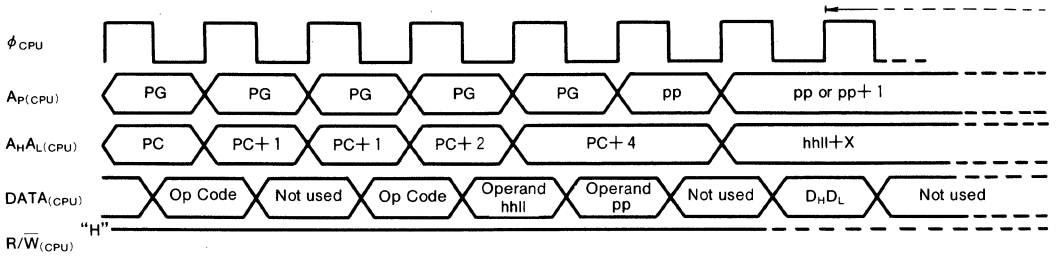
Timing :

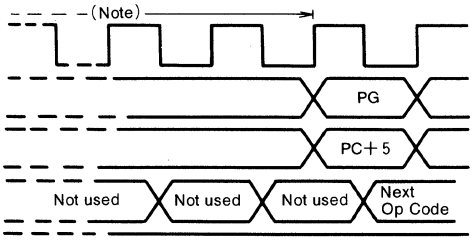


Absolute Long Indexed X

Instruction : D I V, MPY

Timing :



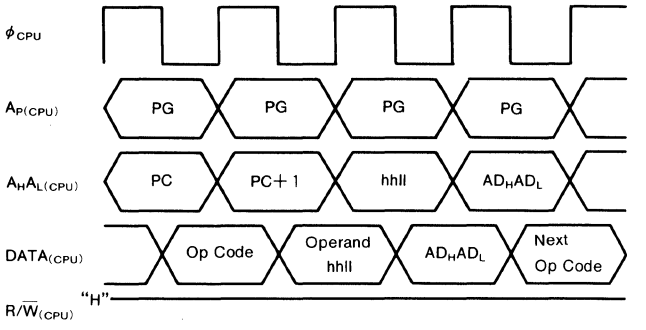


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Absolute Indirect

Instruction : JMP

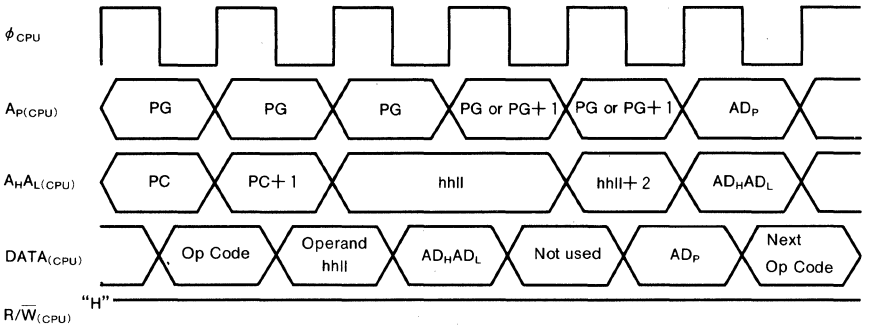
Timing :



Absolute Indirect Long

Instruction : JMP

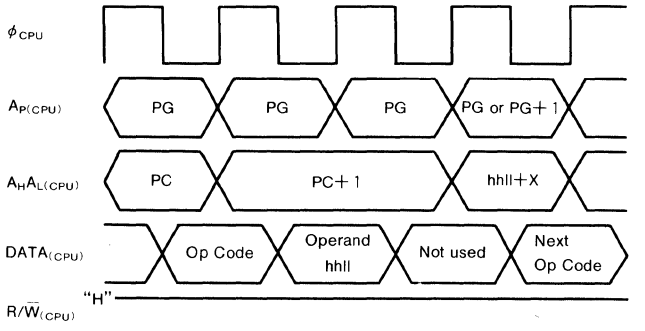
Timing :



Absolute Indexed X Indirect

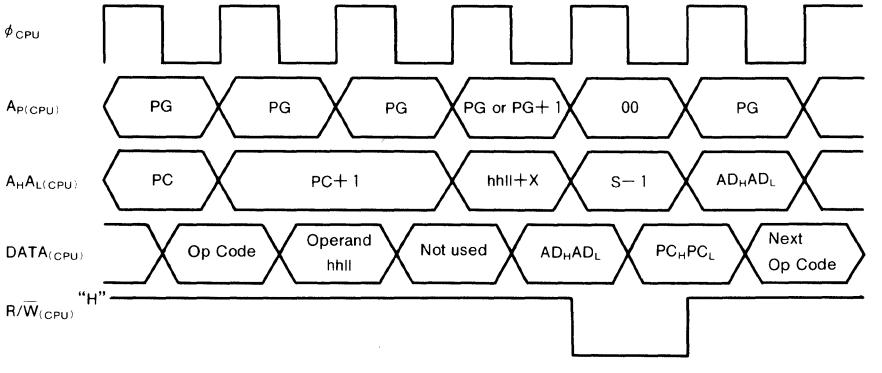
Instruction : JMP

Timing :



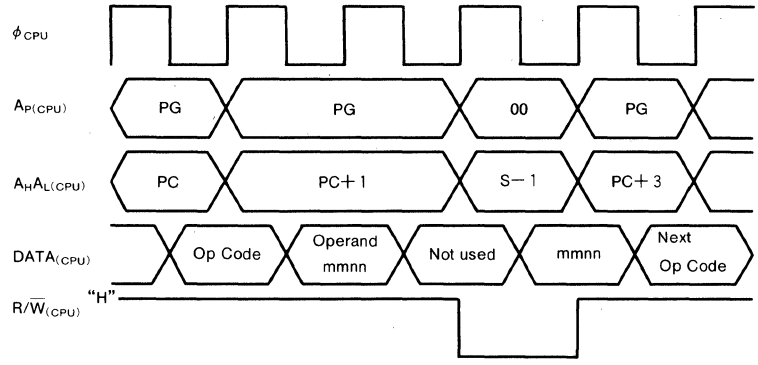
Instruction : JSR

Timing :



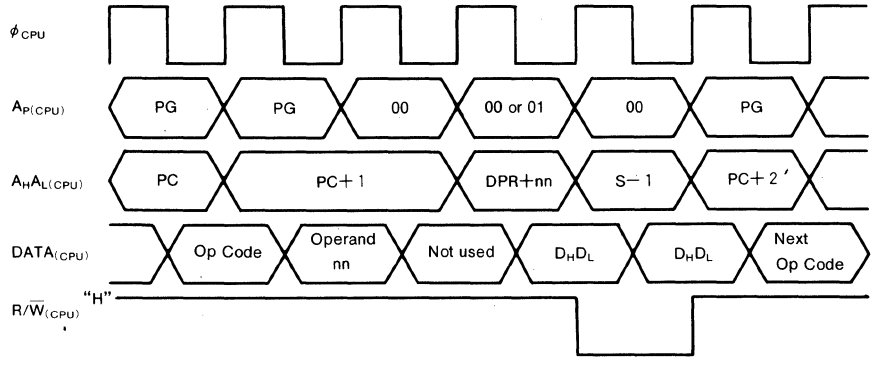
Instruction : P E A

Timing :



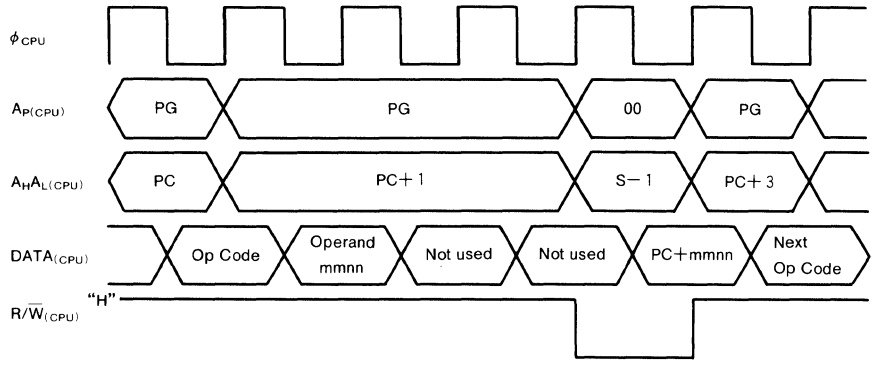
Instruction : P E I

Timing :



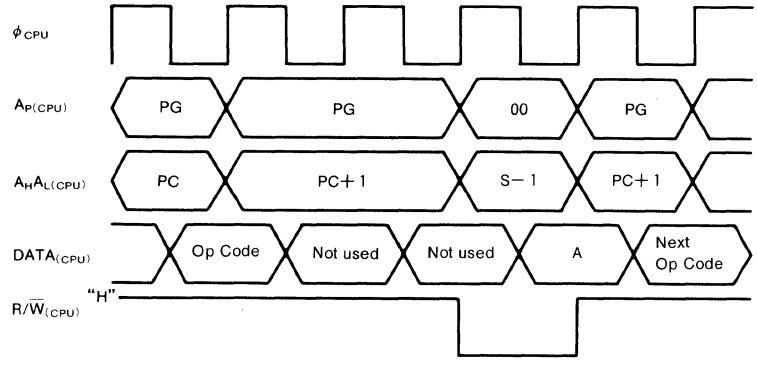
Instruction : PER

Timing :



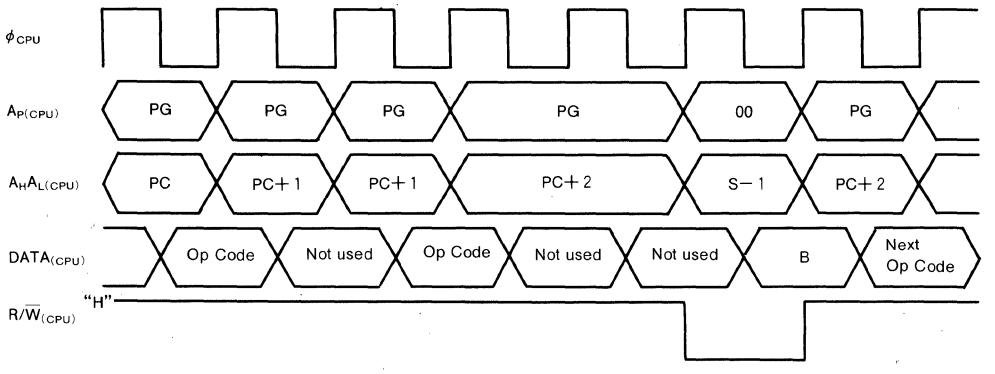
Instruction : PHA, PHD, PHP, PHX, PHY

Timing :



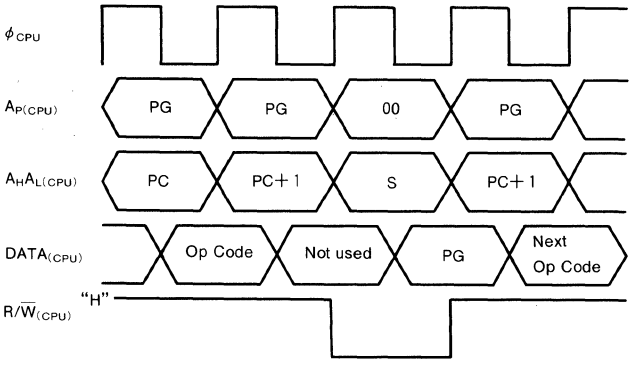
Instruction : PHB

Timing :



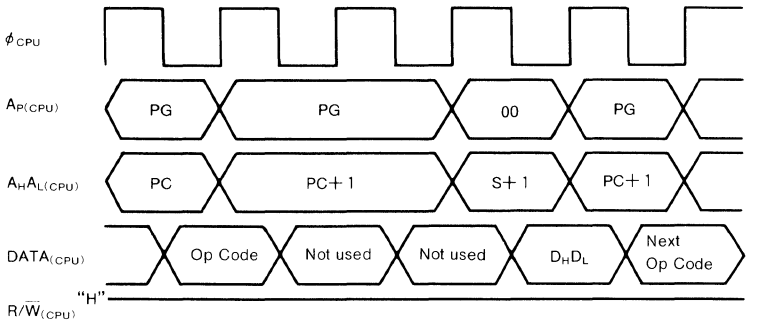
Instruction : PHG, PHT

Timing :



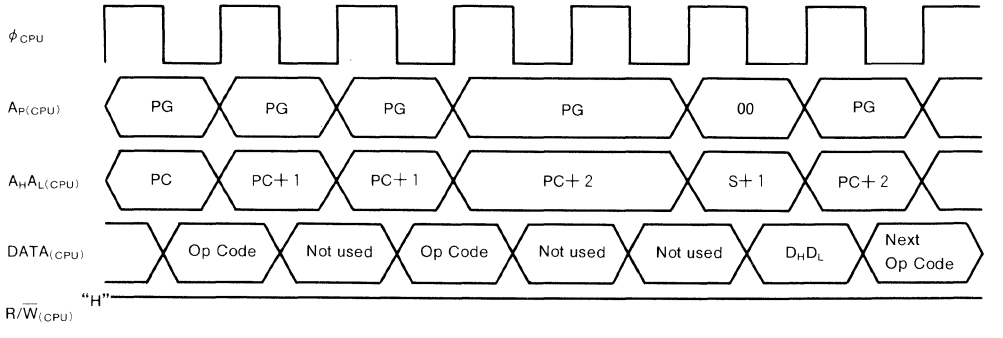
Instruction : PLA, PLD, PLX, PLY

Timing :



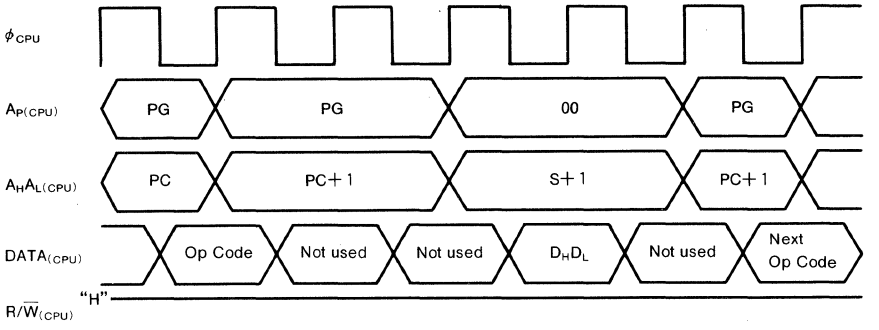
Instruction : PLB

Timing :



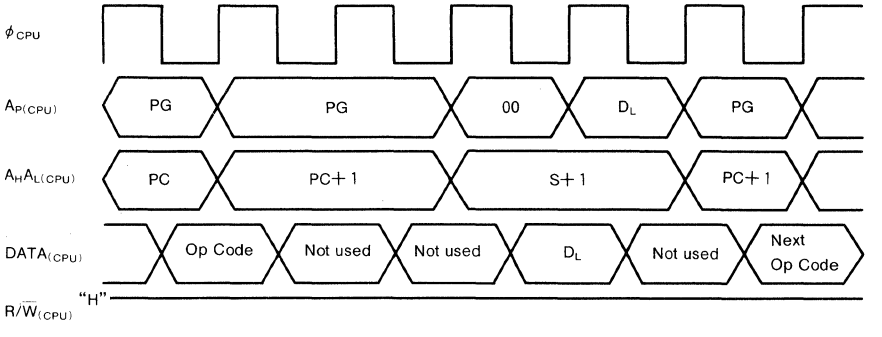
Instruction : P L P

Timing :



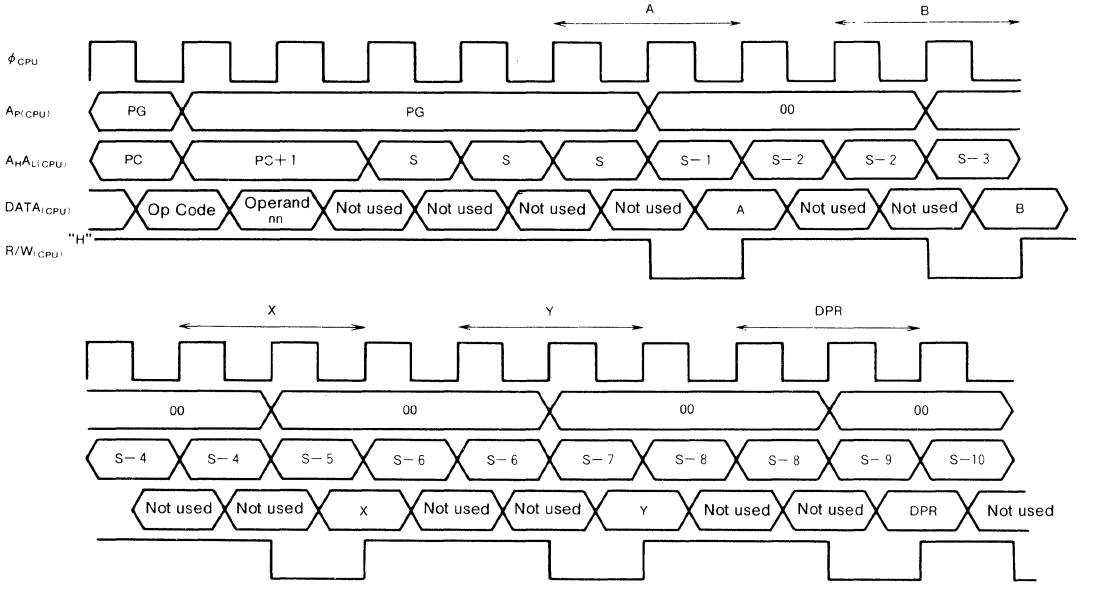
Instruction : P L T

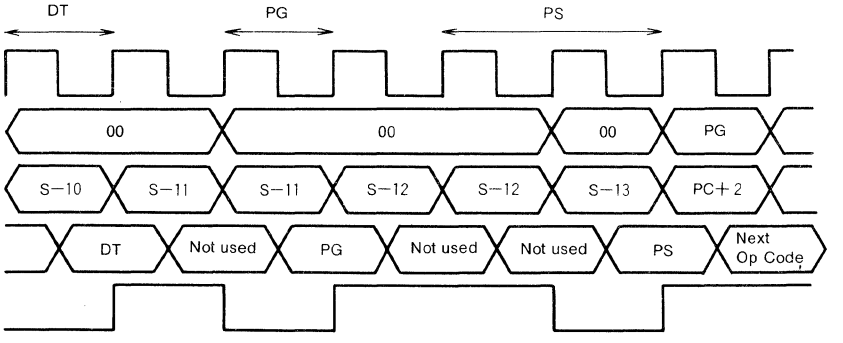
Timing :



Instruction : P S H

Timing :

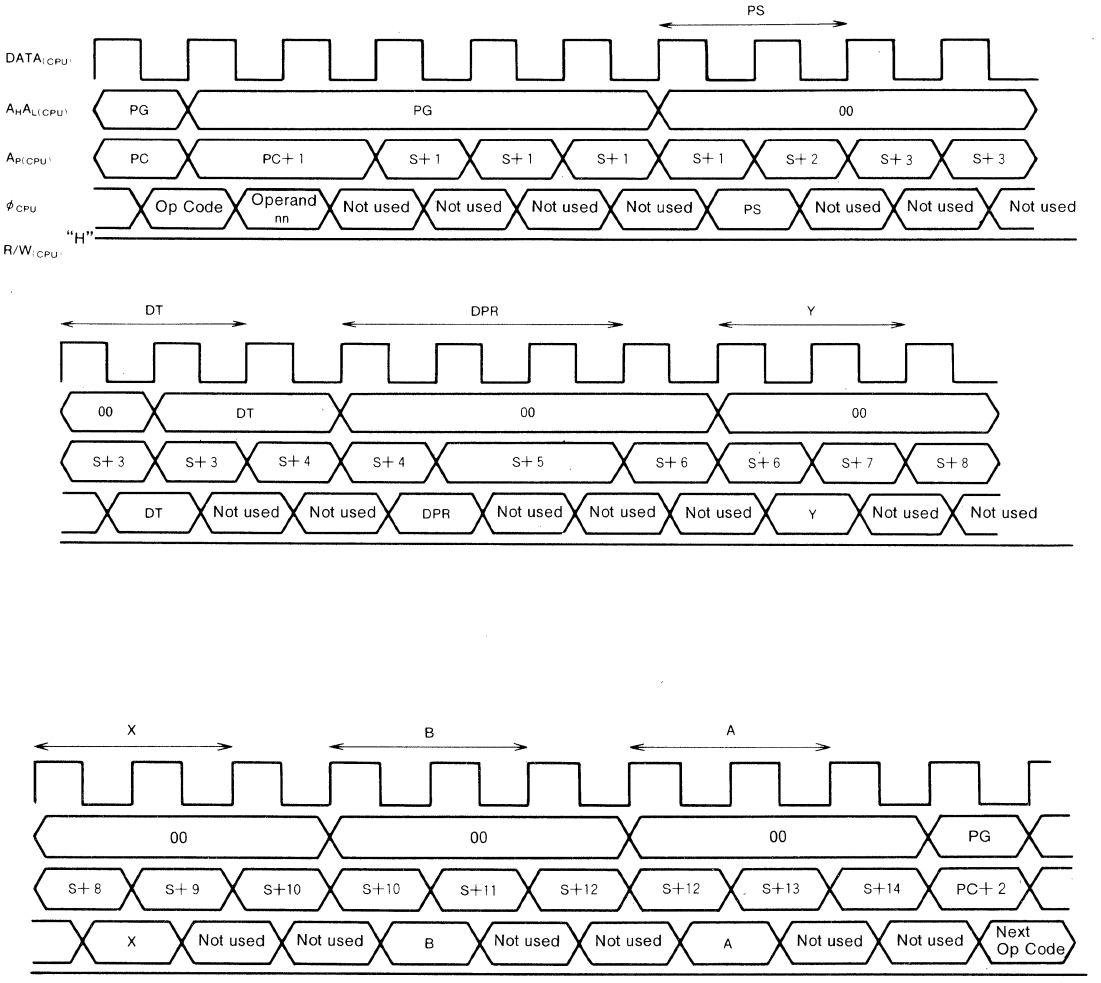




(Note) This figure is an example pushed all the registers by PSH instruction. If any register is not pushed, its cycle (↔) is nothing.

Instruction : PUL

Timing :

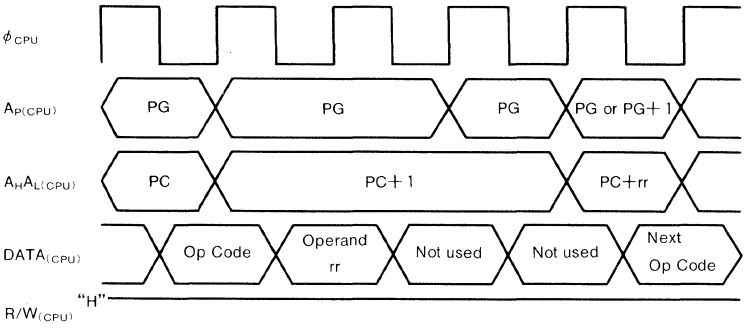


(Note) This figure is an example pulled all the registers by PUL instruction. If some register is not pulled, its cycle (↔) is nothing.

Relative

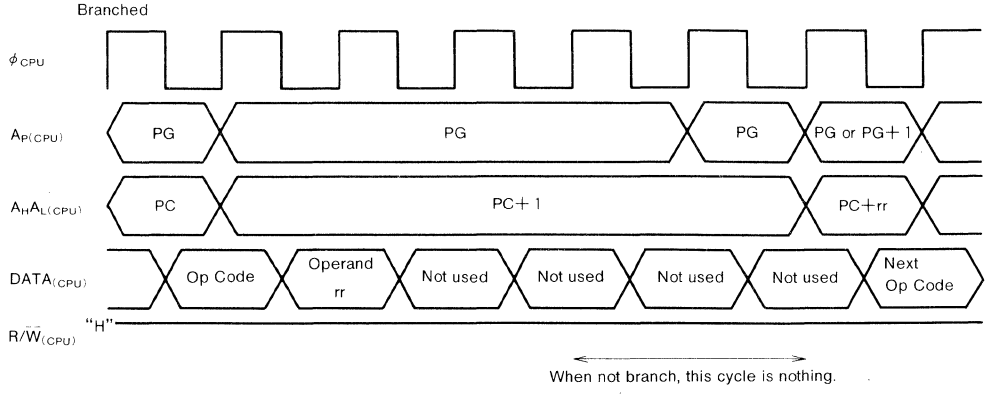
Instruction : B R A

Timing :



Instruction : B C C, B C S, B E Q, B M I, B N E, B P L, B V C, B V S

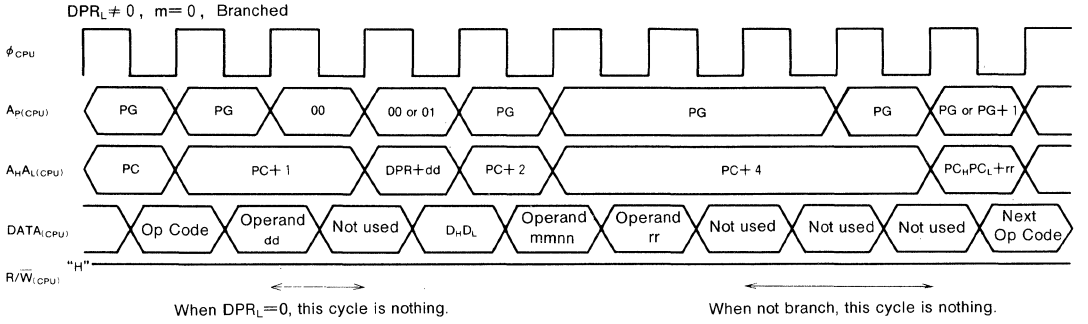
Timing :



Direct Bit Relative

Instruction : BBC, BBS

Timing :

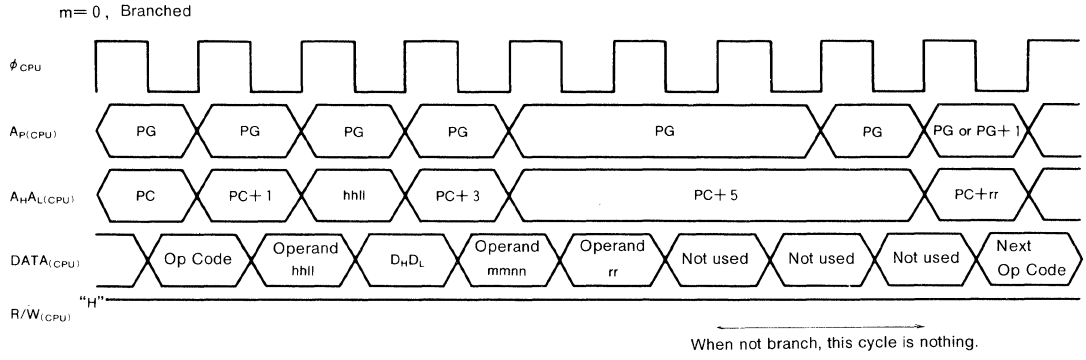


When m=1, fetched operand at 5-th cycle is 1-byte(nn).

Absolute Bit Relative

Instruction : BBC, BBS

Timing :

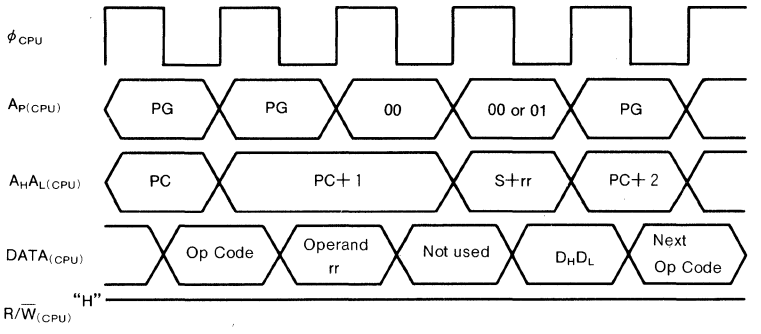


When m=1, fetched operand at 4-th cycle is 1-byte(nn).

Stack Pointer Relative

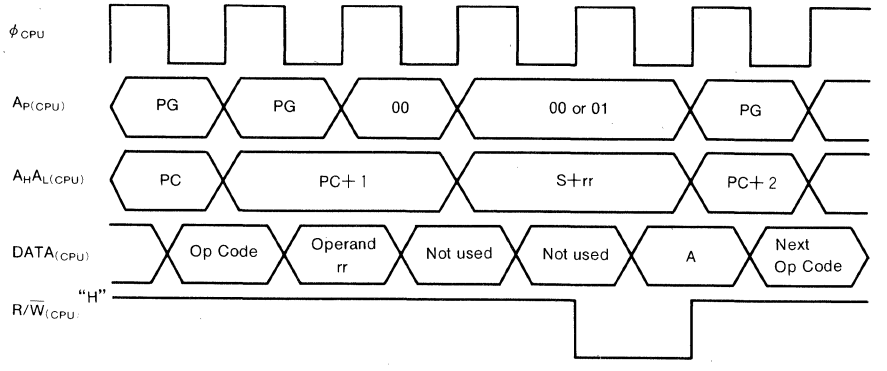
Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

Timing :



Instruction : STA

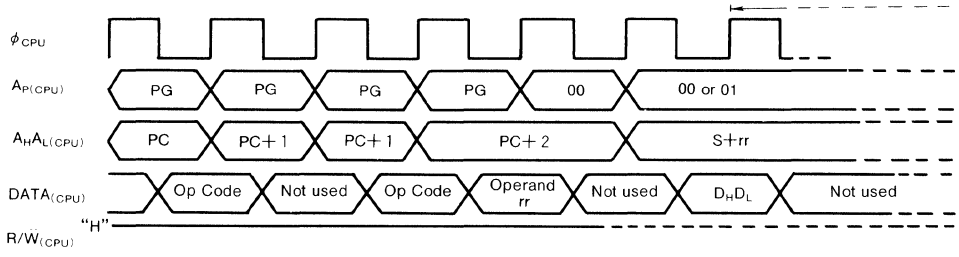
Timing :

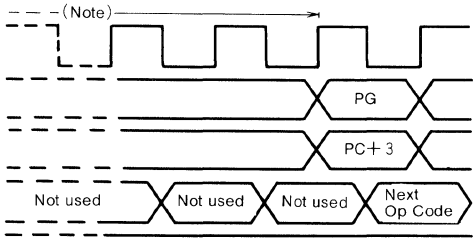


Stack Pointer Relative

Instruction : D I V, MPY

Timing :



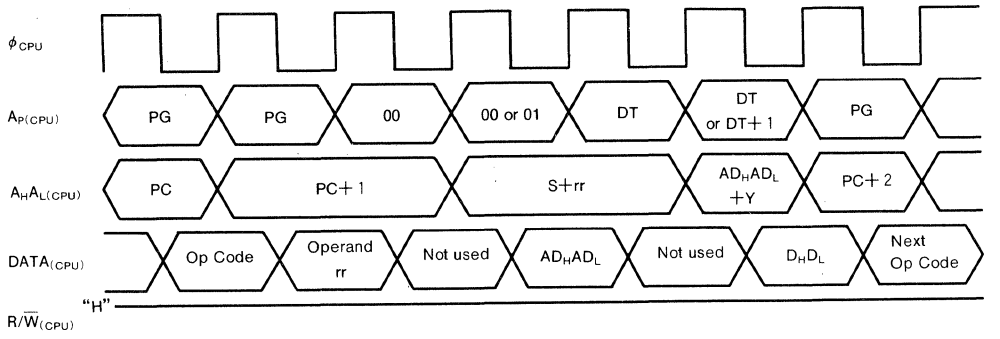


(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Stack Pointer Relative Indirect Indexed Y

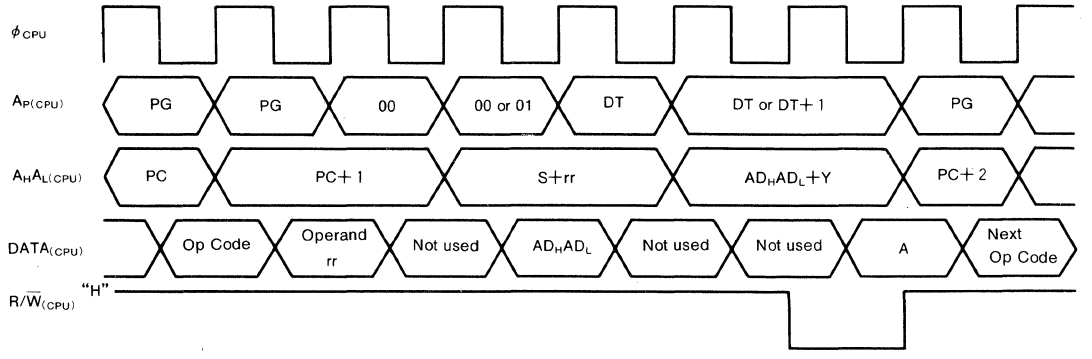
Instruction : ADC, AND, CMP, EOR, LDA, ORA, SBC

Timing :



Instruction : STA

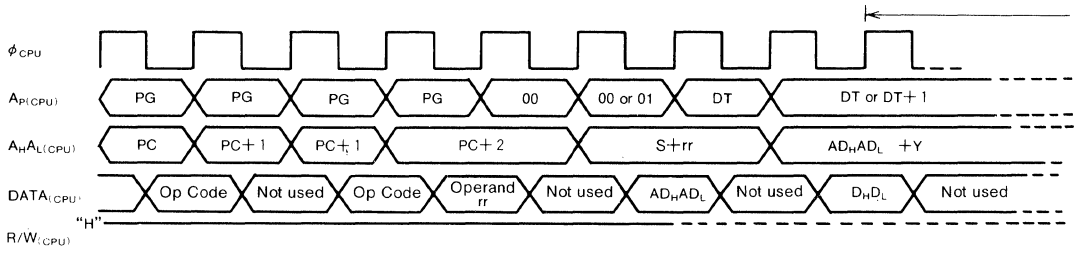
Timing :

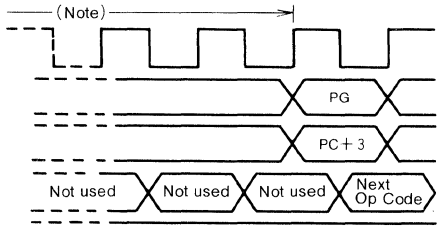


Stack Pointer Relative Indirect Indexed Y

Instruction : D I V, MPY

Timing :



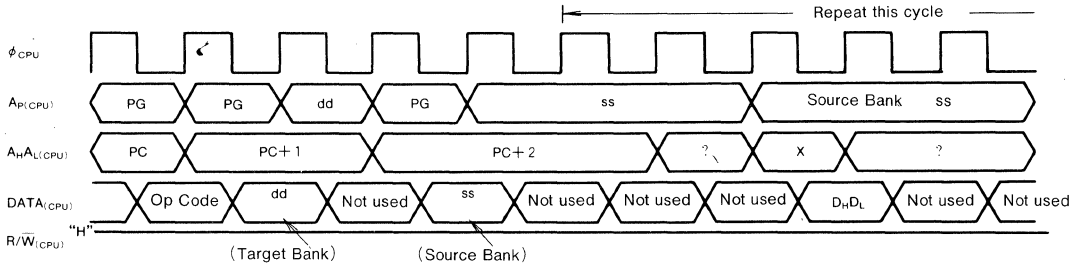


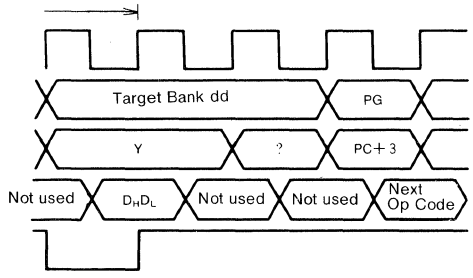
(Note) MPY instruction is 12-cycle, and DIV instruction is 23-cycle.

Block Transfer

Instruction : MVN

Timing :



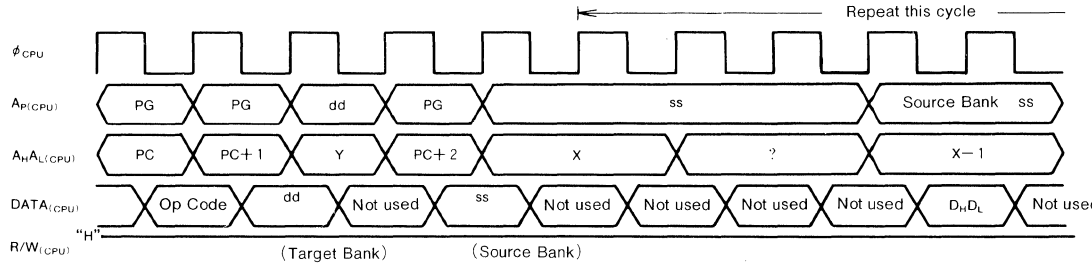


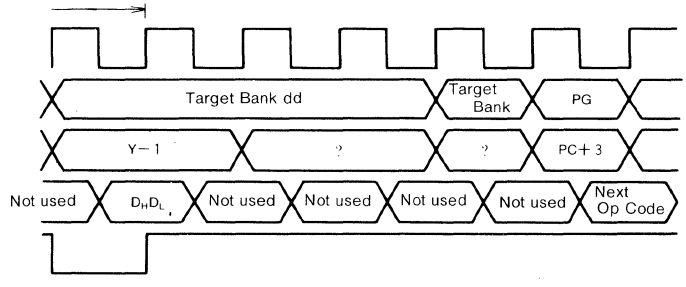
(Note) This figure is shown that transfered the 2-bytes data started from even address. If transfered more than 3-bytes data, the cycle(↔) is repeated each 2-bytes.

Block Transfer

Instruction : MVP

Timing :





(Note) This figure is shown that transfered the 2-bytes data started from even address. If transfered more than 3-bytes data, the cycle (↔) is repeated each 2-bytes.

MACHINE INSTRUCTIONS

Symbol	Function	Details	Addressing mode																																
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
ADC (Note 1,2)	A_{CC},C ← A_{CC}+M+C	Adds the carry, the accumulator and the memory contents. The result is entered into the accumulator. When the D flag is "0", binary additions is done, and when the D flag is "1", decimal addition is done.				69	2	2				65	4	2				75	5	2				72	6	2	61	7	2	71	8	2
						42 69	4	3				42 65	6	3				42 75	7	3				42 72	8	3	42 61	9	3	42 71	10	3
AND (Note 1,2)	A_{CC} ← A_{CC}∧M	Obtains the logical product of the contents of the accumulator and the contents of the memory. The result is entered into the accumulator.				29	2	2				25	4	2				35	5	2				32	6	2	21	7	2	31	8	2
						42 29	4	3				42 25	6	3				42 35	7	3				42 32	8	3	42 21	9	3	42 31	10	3
ASL (Note 1)	m=0 C ← b_{15} ··· b_0 ←0 m=1 C ← b_7 ··· b_0 ←0	Shifts the accumulator or the memory contents one bit to the left. "0" is entered into bit 0 of the accumulator or the memory. The contents of bit 15 (bit 7 when the m flag is "1") of the accumulator or memory before shift is entered into the C flag.							0A	2	1	06	7	2				16	7	2												
									42 0A	4	2																					
BBC (Note 3,5)	Mb=0 ?	Tests the specified bit of the memory. Branches when all the contents of the specified bit is "0".																																	
BBS (Note 3,5)	Mb=1 ?	Tests the specified bit of the memory. Branches when all the contents of the specified bit is "1".																																	
BCC (Note 3)	C=0 ?	Branches when the contents of the C flag is "0".																																	
BCS (Note 3)	C=1 ?	Branches when the contents of the C flag is "1".																																	
BEQ (Note 3)	Z=1 ?	Branches when the contents of the Z flag is "1".																																	
BMI (Note 3)	N=1 ?	Branches when the contents of the N flag is "1".																																	
BNE (Note 3)	Z=0 ?	Branches when the contents of the Z flag is "0".																																	
BPL (Note 3)	N=0 ?	Branches when the contents of the N flag is "0".																																	
BRA (Note 4)	PC←PC±offset PG←PG+1 (carry occured) PG←PG−1 (borrow occured)	Jumps to the address indicated by the program counter plus the offset value.																																	
BRK	PC←PC+2 M(S)←PG S←S−1 M(S)←PC_H S←S−1 M(S)←PC_L S←S−1 M(S)←PS_H S←S−1 M(S)←PS_L S←S−1 I←1 PC_L←AD_L PC_H←AD_H PG←00_{16}	Executes software interruption.	00	15	2																														
BVC (Note 3)	V=0 ?	Branches when the contents of the V flag is "0".																																	
BVS (Note 3)	V=1 ?	Branches when the contents of the V flag is "1".																																	
CLB (Note 5)	Mb←0	Makes the contents of the specified bit in the memory "0".													14	8	3																		
CLC	C←0	Makes the contents of the C flag "0".	18	2	1																														
CLI	I←0	Makes the contents of the I flag "0".	58	2	1																														
CLM	m←0	Makes the contents of the m flag "0".	D8	2	1																														
CLP	PSb←0	Specifies the bit position in the processor status register by the bit pattern of the second byte in the instruction, and sets "0" in that bit.				C2	4	2																											
CLV	V←0	Makes the contents of the V flag "0".	B8	2	1																														
CMP (Note 1,2)	A_{CC}−M	Compares the contents of the accumulator with the contents of the memory.				C9	2	2				C5	4	2				D5	5	2				D2	6	2	C1	7	2	D1	8	2
						42 C9	4	3				42 C5	6	3				42 D5	7	3				42 D2	8	3	42 C1	9	3	42 D1	10	3

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
67	10	2	77	11	2	6D	4	3				7D	6	3	79	6	3	6F	6	4	7F	7	4																						63	5	2	73	8	2				•	•	•	N	V	•	•	•	•	Z	C
42 67	12	3	42 77	13	3	42 6D	6	4				42 7D	8	4	42 79	8	4	42 6F	8	5	42 7F	9	5																						42 63	7	3	42 73	10	3														
27	10	2	37	11	2	2D	4	3				3D	6	3	39	6	3	2F	6	4	3F	7	4																						23	5	2	33	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 27	12	3	42 37	13	3	42 2D	6	4				42 3D	8	4	42 39	8	4	42 2F	8	5	42 3F	9	5																						42 23	7	3	42 33	10	3														
						0E	7	3				1E	8	3																																								•	•	•	N	•	•	•	•	•	Z	C
																																							34	7	4	3C	8	5										•	•	•	•	•	•	•	•	•	•	•
																																							24	7	4	2C	8	5										•	•	•	•	•	•	•	•	•	•	•
																																				90	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				B0	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				F0	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				30	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				D0	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				10	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				80	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				82	4	3																										
																																																						•	•	•	•	•	•	•	•	•	•	•
																																				50	4	2																•	•	•	•	•	•	•	•	•	•	•
																																				70	4	2																•	•	•	•	•	•	•	•	•	•	•
									1C	9	4																																													•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	0
																																																						•	•	•	•	•	•	•	•	0	•	•
																																																						•	•	•	•	•	0	•	•	•	•	•
																																																						•	•	•	Specified flag becomes "0".							
																																																						•	•	•	•	0	•	•	•	•	•	•
C7	10	2	D7	11	2	CD	4	3				DD	6	3	D9	6	3	CF	6	4	DF	7	4																						C3	5	2	D3	8	2				•	•	•	N	•	•	•	•	•	Z	C
42 C7	12	3	42 D7	13	3	42 CD	6	4				42 DD	8	4	42 D9	8	4	42 CF	8	5	42 DF	9	5																						42 C3	7	3	42 D3	10	3														

Symbol	Function	Details	Addressing mode																																
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y					
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#			
CPX (Note 1, 2)	X−M	Compares the contents of the index register X with the contents of the memory.				E0	2	2				E4	4	2																					
CPY (Note 1, 2)	Y−M	Compares the contents of the index register Y with the contents of the memory.				C0	2	2				C4	4	2																					
DEC (Note 1)	A_{CC}←A_{CC}−1 or M←M−1	Decrements the contents of the accumulator or memory by 1.							1A	2	1	C6	7	2				D6	7	2															
									42 1A	4	2																								
DEX	X←X−1	Decrements the contents of the index register X by 1.	CA	2	1																														
DEY	Y←Y−1	Decrements the contents of the index register Y by 1.	88	2	1																														
DIV (Note 2,10)	A(quotient)←B,A/M B(remainder)	The numeral that places the contents of accumulator B to the higher order and the contents of accumulator A to the lower order is divided by the contents of the memory. The quotient is entered into accumulator A and the remainder into accumulator B.				89 29	27	3				89 25	29	3				89 35	30	3				89 32	31	3	89 21	32	3	89 31	33	3			
EOR (Note 1,2)	A_{CC}←A_{CC}¥M	Logical exclusive sum is obtained of the contents of the accumulator and the contents of the memory. The result is placed into the accumulator.				49	2	2				45	4	2				55	5	2				52	6	2	41	7	2	51	8	2			
						42 49	4	3				42 45	6	3				42 55	7	3				42 52	8	3	42 41	9	3	42 51	10	3			
INC (Note 1)	A_{CC}←A_{CC}+1 or M ←M+1	Increments the contents of the accumulator or memory by 1.							3A	2	1	E6	7	2				F6	7	2															
									42 3A	4	2																								
INX	X←X+1	Increments the contents of the index register X by 1.	E8	2	1																														
INY	Y←Y+1	Increments the contents of the index register Y by 1.	C8	2	1																														
JMP	ABS PC_L ← AD_L PC_H ← AD_H ABL PC_L ← AD_L PC_H ← AD_H PG←AD_G (ABS) PC_L ←(AD_H, AD_L) PC_H ←(AD_H, AD_L+1) L(ABS) PC_L ←(AD_H, AD_L) PC_H ←(AD_H, AD_L+1) PG ←(AD_H, AD_L+2) (ABS, X) PC_L ←(AD_H, AD_L+X) PC_H ←(AD_H, AD_L+X +1)	Places a new address into the program counter and jumps to that new address.																																		
JSR	ABS M(S)← PC_H S←S−1 M(S)← PC_L S←S−1 PC_L ← AD_L PC_H ← AD_H ABL M(S)← PG S←S−1 M(S)← PC_H S←S−1 M(S)←PC_L S ← S−1 PC_L ← AD_L PC_H ← AD_H PG ← AD_G (ABS, X) M(S)← PC_H S←S−1 M(S)← PC_L S←S−1 PC_L ←(AD_H, AD_L+X) PC_H ←(AD_H, AD_L+X +1)	Saves the contents of the program counter (also the contents of the program bank register for ABL) into the stack, and jumps to the new address.																																		

APPENDIX B

Series MELPS 7700 Machine Instructions

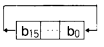
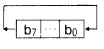
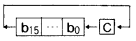
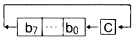
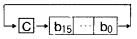
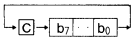
Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
						EC	4	3																																														•	•	•	N	•	•	•	•	•	Z	C
						CC	4	3																																														•	•	•	N	•	•	•	•	•	Z	C
						CE	7	3				DE	8	3																																								•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
89 27	35	3	89 37	36	3	89 2D	29	4				89 3D	31	4	89 39	31	4	89 2F	31	5	89 3F	32	5																								89 23	30	3	89 33	33	3				•	•	•	N	V	•	•	•	•	Z	C
47	10	2	57	11	2	4D	4	3				5D	6	3	59	6	3	4F	6	4	5F	7	4																								43	5	2	53	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 47	12	3	42 57	13	3	42 4D	6	4				42 5D	8	4	42 59	8	4	42 4F	8	5	42 5F	9	5																								42 43	7	3	42 53	10	3														
						EE	7	3				FE	8	3																																								•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
						4C	2	3										5C	4	4				6C	4	3	DC	8	3	7C	6	3																						•	•	•	•	•	•	•	•	•	•	•
						20	6	3										22	8	4										FC	8	3																						•	•	•	•	•	•	•	•	•	•	•

Symbol	Function	Details	Addressing mode																															
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
LDA (Note 1,2)	$A_{CC} \leftarrow M$	Enters the contents of the memory into the accumulator.				A9	2	2				A5	4	2				B5	5	2				B2	6	2	A1	7	2	B1	8	2
						42 A9	4	3				42 A5	6	3				42 B5	7	3				42 B2	8	3	42 A1	9	3	42 B1	10	3
LDM (Note 5)	$M \leftarrow IMM$	Enters the immediate value into the memory.										64	4	3				74	5	3												
LDT	$DT \leftarrow IMM$	Enters the immediate value into the data bank register.				89 C2	5	3																								
LDX (Note 1,2)	$X \leftarrow M$	Enters the contents of the memory into index register X.				A2	2	2				A6	4	2							B6	5	2									
LDY (Note 1,2)	$Y \leftarrow M$	Enters the contents of the memory into index register Y.				A0	2	2				A4	4	2				B4	5	2												
LSR (Note 1)	m=0 $0 \rightarrow [b_{15} \cdots b_0] \rightarrow C$ m=1 $0 \rightarrow [b_7 \cdots b_0] \rightarrow C$	Shifts the contents of the accumulator or the contents of the memory one bit to the right. The bit 0 of the accumulator or the memory is entered into the C flag. "0" is entered into bit 15 (bit 7 when the m flag is "1".)							4A	2	1	46	7	2				56	7	2												
									42 4A	4	2																					
MPY (Note 2,11)	B, A←A*M	Multiplies the contents of accumulator A and the contents of the memory. The higher order of the result of operation are entered into accumulator B, and the lower order into accumulator A.				89 09	16	3				89 05	18	3				89 15	19	3				89 12	20	3	89 01	21	3	89 11	22	3
MVN (Note 8)	Mn+i←Mm+i	Transmits the data block. The transmission is done from the lower order address of the block.																														
MVP (Note 9)	Mn−i←Mm−i	Transmits the data block. Transmission is done form the higher order address of the data block.																														
NOP	PC←PC+1	Advances the program counter, but performs nothing else.	EA	2	1																											
ORA (Note 1,2)	$A_{CC} \leftarrow A_{CC} \vee M$	Logical sum per bit of the contents of the accumulator and the contents of the memory is obtained. The result is entered into the accumulator.				09	2	2				05	4	2				15	5	2				12	6	2	01	7	2	11	8	2
						42 09	4	3				42 05	6	3				42 15	7	3				42 12	8	3	42 01	9	3	42 11	10	3
PEA	$M(S) \leftarrow IMM_2$ $S \leftarrow S-1$ $M(S) \leftarrow IMM_1$ $S \leftarrow S-1$	The 3rd and the 2nd bytes of the instruction are saved into the stack, in this order.																														
PEI	$M(S) \leftarrow M((DPR)+IMM+1)$ $S \leftarrow S-1$ $M(S) \leftarrow M((DPR)+IMM)$ $S \leftarrow S-1$	Specifies 2 sequential bytes in the direct page in the 2nd byte of the instruction, and saves the contents into the stack.																														
PER	$EAR \leftarrow PC+IMM_2,IMM_1$ $M(S) \leftarrow EAR_H$ $S \leftarrow S-1$ $M(S) \leftarrow EAR_L$ $S \leftarrow S-1$	Regards the 2nd and 3rd bytes of the instruction as 16-bit numerals, adds them to the program counter, and saves the result into the stack.																														
PHA	m=0 $M(S) \leftarrow A_H$ $S \leftarrow S-1$ $M(S) \leftarrow A_L$ $S \leftarrow S-1$ m=1 $M(S) \leftarrow A_L$ $S \leftarrow S-1$	Saves the contents of accumulator A into the stack.																														
PHB	m=0 $M(S) \leftarrow B_H$ $S \leftarrow S-1$ $M(S) \leftarrow B_L$ $S \leftarrow S-1$ m=1 $M(S) \leftarrow B_L$ $S \leftarrow S-1$	Saves the contents of accumulator B into the stack.																														

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
A7	10	2	B7	11	2	AD	4	3				BD	6	3	B9	6	3	AF	6	4	BF	7	4																							A3	5	2	B3	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 A7	12	3	42 B7	13	3	42 AD	6	4				42 BD	8	4	42 B9	8	4	42 AF	8	5	42 BF	9	5																							42 A3	7	3	42 B3	10	3														
						9C	5	4				9E	6	4																																								•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	•
						AE	4	3							BE	6	3																																					•	•	•	N	•	•	•	•	•	Z	•
						AC	4	3				BC	6	3																																								•	•	•	N	•	•	•	•	•	Z	•
						4E	7	3				5E	8	3																																								•	•	•	0	•	•	•	•	•	Z	C
89 07	24	3	89 17	25	3	89 0D	18	4				89 1D	20	4	89 19	20	4	89 0F	20	5	89 1F	21	5																							89 03	19	3	89 13	22	3				•	•	•	N	•	•	•	•	•	Z	0
																																																			54	7 + $\frac{i}{2}$×7	3	•	•	•	•	•	•	•	•	•	•	•
																																																				44	9 + $\frac{i}{2}$×7	3	•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	•
07	10	2	17	11	2	0D	4	3				1D	6	3	19	6	3	0F	6	4	1F	7	4																							03	5	2	13	8	2				•	•	•	N	•	•	•	•	•	Z	•
42 07	12	3	42 17	13	3	42 0D	6	4				42 1D	8	4	42 19	8	4	42 0F	8	5	42 1F	9	5																							42 03	7	3	42 13	10	3														
																																		F4	5	3																			•	•	•	•	•	•	•	•	•	•	•
																																		D4	5	2																			•	•	•	•	•	•	•	•	•	•	•
																																		62	5	3																			•	•	•	•	•	•	•	•	•	•	•
																																		48	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																		42 48	6	2																			•	•	•	•	•	•	•	•	•	•	•

Symbol	Function	Details	Addressing mode																															
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y				
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#		
PHD	M(S)←DPR$_H$ S←S−1 M(S)←DPR$_L$ S←S−1	Saves the contents of the direct page register into the stack.																																
PHG	M(S)←PG S←S−1	Saves the contents of the program bank register into the stack.																																
PHP	M(S)←PS$_H$ S←S−1 M(S)←PS$_L$ S←S−1	Saves the contents of the program status register into the stack.																																
PHT	M(S)←DT S←S−1	Saves the contents of the data bank register into the stack.																																
PHX	x=0 M(S)←X$_H$ S←S−1 M(S)←X$_L$ S←S−1 x=1 M(S)←X$_L$ S←S−1	Saves the contents of the index register X into the stack.																																
PHY	x=0 M(S)←Y$_H$ S←S−1 M(S)←Y$_L$ S←S−1 x=1 M(S)←Y$_L$ S←S−1	Saves the contents of the index register Y into the stack.																																
PLA	m=0 S←S+1 A$_L$←M(S) S←S+1 A$_H$←M(S) m=1 S←S+1 A$_L$←M(S)	Restores the contents of the stack on the accumulator A.																																
PLB	m=0 S←S+1 B$_L$←M(S) S←S+1 B$_H$←M(S) m=1 S←S+1 B$_L$←M(S)	Restores the contents of the stack on the accumulator B.																																
PLD	S←S+1 DPR$_L$←M(S) S←S+1 DPR$_H$←M(S)	Restores the contents of the stack on the direct page register.																																
PLP	S←S+1 PS$_L$←M(S) S←S+1 PS$_H$←M(S)	Restores the contents of the stack on the processor status register.																																
PLT	S←S+1 DT←M(S)	Restores the contents of the stack on the data bank register.																																
PLX	x=0 S←S+1 X$_L$←M(S) S←S+1 X$_H$←M(S) x=1 S←S+1 X$_L$←M(S)	Restores the contents of the stack on the index register X.																																

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
																																	0B	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	4B	3	1																			•	•	•	•	•	•	•	•	•	•	•
																																	08	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	8B	3	1																			•	•	•	•	•	•	•	•	•	•	•
																																	DA	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	5A	4	1																			•	•	•	•	•	•	•	•	•	•	•
																																	68	5	1																			•	•	•	N	•	•	•	•	•	Z	•
																																	42 68	7	2																			•	•	•	N	•	•	•	•	•	Z	•
																																	2B	5	1																			•	•	•	•	•	•	•	•	•	•	•
																																	28	6	1																			Value saved in stack.										
																																	AB	6	1																			•	•	•	N	•	•	•	•	•	Z	•
																																	FA	5	1																			•	•	•	N	•	•	•	•	•	Z	•

Symbol	Function	Details	Addressing mode																																
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y					
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
PLY	x=0 S←S+1 Y_L←M(S) S←S+1 Y_H←M(S) x=1 S←S+1 Y_L←M(S)	Restores the contents of the stack on the index register Y.																																	
PSH (Note 6)	M(S)←A, B, X···	Saves the registers among accumulator, index register, direct page register, data bank register, program bank register, or processor status register, specified by the bit pattern of the second byte of the instruction into the stack.																																	
PUL (Note 7)	A, B, X···←M(S)	Restores the contents of the stack to the registers among accumulator, index register, direct page register, data bank register, or processor status register, specified by the bit pattern of the second byte of the instruction.																																	
RLA (Note 13)	m=0 n bit rotate left  m=1 n bit rotate left  	Rotates the contents of the accumulator A, n bits to the left.				89 49	6 + i	3																												
ROL (Note 1)	m=0  m=1  	Links the accumulator or the memory to C flag, and rotates result to the left by 1 bit.							2A 42 2A	2 4	1 2	26 2	7	2				36	7	2																
ROR (Note 1)	m=0  m=1  	Links the accumulator or the memory to C flag, and rotates result to the right by 1 bit.							6A 42 6A	2 4	1 2	66 2	7	2				76	7	2																
RTI	S←S+1 PS_L←M(S) S←S+1 PS_H←M(S) S←S+1 PC_L←M(S) S←S+1 PC_H←M(S) S←S+1 PG←M(S)	Returns from the interruption routine.	40	11	1																														
RTL	S←S+1 PC_L←M(S) S←S+1 PC_H←M(S) S←S+1 PG←M(S)	Returns from the subroutine. The contents of the program bank register are also restored.	6B	8	1																														
RTS	S←S+1 PC_L←M(S) S←S+1 PC_H←M(S)	Returns from the subroutine. The contents of the program bank register are not restored.	60	5	1																														
SBC (Note 1,2)	A_{CC}, C←A_{CC}−M−$\overline{C}$	Subtracts the contents of the memory and the borrow from the contents of the accumulator.				E9 42 E9	2 4	2 3				E5 42 E5	4 6	2 3				F5 42 F5	5 7	2 3				F2 42 F2	6 8	2 3	E1 42 E1	7 9	2 3	F1 42 F1	8 10	2 3

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
																																	7A	5	1																			•	•	•	N	•	•	•	•	•	Z	•
																																	EB	12+2i_1+i_2	2																			•	•	•	•	•	•	•	•	•	•	•
																																	FB	14+3i_1+4i_2	2																			If restored the contents of PS, it becomes its value. And the other case is no change.										
																																																					•	•	•	•	•	•	•	•	•	•	•
						2E	7	3				3E	8	3																																									•	•	•	N	•	•	•	•	•	Z	C
						6E	7	3				7E	8	3																																									•	•	•	N	•	•	•	•	•	Z	C
																																																								Value saved in stack.										
																																																							•	•	•	•	•	•	•	•	•	•	•
																																																							•	•	•	•	•	•	•	•	•	•	•
E7	10	2	F7	11	2	ED	4	3				FD	6	3	F9	6	3	EF	6	4	FF	7	4																						E3	5	2	F3	8	2				•	•	•	N	V	•	•	•	•	Z	C
42 E7	12	3	42 F7	13	3	42 ED	6	4				42 FD	8	4	42 F9	8	4	42 EF	8	5	42 FF	9	5																						42 E3	7	3	42 F3	10	3														

Symbol	Function	Details	Addressing mode																														
			IMP			IMM			A			DIR			DIR,b			DIR,X			DIR,Y			(DIR)			(DIR,X)			(DIR),Y		
			op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#
SEB (Note 5)	Mb←1	Makes the contents of the specified bit in the memory "1".													04	8	3																
SEC	C←1	Makes the contents of the C flag "1".	38	2	1																												
SEI	I←1	Makes the contents of the I flag "1".	78	2	1																												
SEM	m←1	Makes the contents of the m flag "1".	F8	2	1																												
SEP	PSb←1	Set the specified bit of the processor status register's lower byte (PS_L) to "1".				E2	3	2																									
STA (Note 1)	M←A_{CC}	Stores the contents of the accumulator into the memory.										85	4	2				95	5	2				92	7	2	81	7	2	91	7	2
												42 85	6	3				42 95	7	3				42 92	9	3	42 81	9	3	42 91	9	3
STP		Stops the oscillation of the oscillator.	DB	3	1																												
STX	M←X	Stores the contents of the index register X into the memory.										86	4	2							96	5	2										
STY	M←Y	Stores the contents of the index register Y into the memory.										84	4	2				94	5	2													
TAD	DPR←A	Transmits the contents of the accumulator A to the direct page register.	5B	2	1																												
TAS	S←A	Transmits the contents of the accumulator A to the stack pointer.	1B	2	1																												
TAX	X←A	Transmits the contents of the accumulator A to the index register X.	AA	2	1																												
TAY	Y←A	Transmits the contents of the accumulator A to the index register Y.	A8	2	1																												
TBD	DPR←B	Transmits the contents of the accumulator B to the direct page register.	42 5B	4	2																												
TBS	S←B	Transmits the contents of the accumulator B to the stack pointer.	42 1B	4	2																												
TBX	X←B	Transmits the contents of the accumulator B to the index register X.	42 AA	4	2																												
TBY	Y←B	Transmits the contents of the accumulator B to the index register Y.	42 A8	4	2																												
TDA	A←DPR	Transmits the contents of the direct page register to the accumulator A.	7B	2	1																												
TDB	B←DPR	Transmits the contents of the direct page register to the accumulator B.	42 7B	4	2																												
TSA	A←S	Transmits the contents of the stack pointer to the accumulator A.	3B	2	1																												
TSB	B←S	Transmits the contents of the stack pointer to the accumulator B.	42 3B	4	2																												
TSX	X←S	Transmits the contents of the stack pointer to the index register X.	BA	2	1																												
TXA	A←X	Transmits the contents of the index register X to the accumulator A.	8A	2	1																												
TXB	B←X	Transmits the contents of the index register X to the accumulator B.	42 8A	4	2																												
TXS	S←X	Transmits the contents of the index register X to the stack pointer.	9A	2	1																												
TXY	Y←X	Transmits the contents of the index register X to the index register Y.	9B	2	1																												
TYA	A←Y	Transmits the contents of the index register Y to the accumulator A.	98	2	1																												
TYB	B←Y	Transmits the contents of the index register Y to the accumulator B.	42 98	4	2																												
TYX	X←Y	Transmits the contents of the index register Y to the index register X.	BB	2	1																												
WIT		Stops the internal clock.	CB	3	1																												
XAB	A⇄B	Exchanges the contents of the accumulator A and the contents of the accumulator B.	89 28	6	2																												

Addressing mode																																																						Processor status register										
L(DIR)			L(DIR),Y			ABS			ABS,b			ABS,X			ABS,Y			ABL			ABL,X			(ABS)			L(ABS)			(ABS,X)			STK			REL			DIR,b,R			ABS,b,R			SR			(SR),Y			BLK			10	9	8	7	6	5	4	3	2	1	0
op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	op	n	#	IPL			N	V	m	x	D	I	Z	C
									0C	9	4																																												•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	1
																																																						•	•	•	•	•	•	•	•	1	•	•
																																																						•	•	•	•	•	1	•	•	•	•	•
																																																						•	•	•	Specified flag becomes "1".							
87	10	2	97	11	2	8D	5	3				9D	5	3	99	5	3	8F	6	4	9F	7	4																						83	5	2	93	8	2				•	•	•	•	•	•	•	•	•	•	•
42 87	12	3	42 97	13	3	42 8D	7	4				42 9D	7	4	42 99	7	4	42 8F	8	5	42 9F	9	5																						42 83	7	3	42 93	10	3														
																																																						•	•	•	•	•	•	•	•	•	•	•
						8E	5	3																																														•	•	•	•	•	•	•	•	•	•	•
						8C	5	3																																														•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	N	•	•	•	•	•	Z	•
																																																						•	•	•	•	•	•	•	•	•	•	•
																																																						•	•	•	N	•	•	•	•	•	Z	•

The number of cycles shown in the table is described in case of the fastest mode for each instruction. The number of cycles shown in the table is calculated for DPR_L=0. The number of cycles in the addressing mode concerning the DPR when $DPR_L \neq 0$ must be incremented by 1.
The number of cycles shown in the table differs according to the bytes fetched into the instruction queue buffer, or according to whether the memory read/write address is odd or even. It also differs when the external region memory is accessed by BYTE="H".

Note 1. The operation code at the upper row is used for accumulator A, and the operation at the lower row is used for accumulator B.

Note 2. When setting flag m=0 to handle the data as 16-bit data in the immediate addressing mode, the number of bytes increments by 1.

Note 3. The number of cycles increments by 2 when branching.

Note 4. The operation code on the upper row is used for branching in the range of −128~+127, and the operation code on the lower row is used for branching in the range of −32768~+32767.

Note 5. When handling 16-bit data with flag m=0, the byte in the table is incremented by 1.

Note 6.

Type of register	A	B	X	Y	DPR	DT	PG	PS
Number of cycles	2	2	2	2	2	1	1	2

The number of cycles corresponding to the register to be pushed are added. The number of cycles when no pushing is done is 12. i_1 indicates the number of registers among A, B, X, Y, DPR, and PS to be saved, while i_2 indicates the number of registers among DT and PG to be saved.

Note 7.

Type of register	A	B	X	Y	DPR	DT	PS
Number of cycles	3	3	3	3	4	3	3

The number of cycles corresponding to the register to be pulled are added. The number of cycles when no pulling is done is 14. i_1 indicates the number of registers among A, B, X, Y, DT, and PS to be restored, while i_2=1 when DPR is to be restored.

Note 8. The number of cycles is the case when the number of bytes to be transfered is even.
When the number of bytes to be transfered is odd, the number is calculated as;

$$7 + (i/2) \times 7 + 4$$

Note that, (i/2) shows the integer part when i is divided by 2.

Note 9. The number of cycles is the case when the number of bytes to be transfered is even.
When the number of bytes to be transfered is odd, the number is calculated as;

$$9 + (i/2) \times 7 + 5$$

Note that, (i/2) shows the integer part when i is divided by 2.

Note 10. The number of cycles is the case in the 16-bit÷8-bit operation. The number of cycles is incremented by 16 for 32-bit÷16-bit operation.

Note 11. The number of cycles is the case in the 8-bit×8-bit operation. The number of cycles is incremented by 8 for 16-bit ×16-bit operation.

Note 12. When setting flag x=0 to handle the data as 16-bit data in the immediate addressing mode, the number of bytes increments by 1.

Note 13. When flag m is 0, the byte in the table is incremented by 1.

APPENDIX B

Series MELPS 7700 Machine Instructions

Symbol	Description	Symbol	Description
IMP	Implied addressing mode	$\forall$	Exclusive OR
IMM	Immediate addressing mode	—	Negation
A	Accumulator addressing mode	←	Movement to the arrow direction
DIR	Direct addressing mode	A_{CC}	Accumulator
DIR, b	Direct bit addressing mode	A_{CCH}	Accumulator's upper 8 bits
DIR, X	Direct indexed X addressing mode	A_{CCL}	Accumulator's lower 8 bits
DIR, Y	Direct indexed Y addressing mode	A	Accumulator A
(DIR)	Direct indirect addressing mode	A_H	Accumulator A's upper 8 bits
(DIR, X)	Direct indexed X indirect addressing mode	A_L	Accumulator A's lower 8 bits
(DIR), Y	Direct indirect indexed Y addressing mode	B	Accumulator B
L (DIR)	Direct indirect long addressing mode	B_H	Accumulator B's upper 8 bits
L (DIR), Y	Direct indirect long indexed Y addressing mode	B_L	Accumulator B's lower 8 bits
ABS	Absolute addressing mode	X	Index register X
ABS, b	Absolute bit addressing mode	X_H	Index register X's upper 8 bits
ABS, X	Absolute indexed X addressing mode	X_L	Index register X's lower 8 bits
ABS, Y	Absolute indexed Y addressing mode	Y	Index register Y
ABL	Absolute long addressing mode	Y_H	Index register Y's upper 8 bits
ABL, X	Absolute long indexed X addressing mode	Y_L	Index register Y's lower 8 bits
(ABS)	Absolute indirect addressing mode	S	Stack pointer
L (ABS)	Absolute indirect long addressing mode	PC	Program counter
(ABS, X)	Absolute indexed X indirect addressing mode	PC_H	Program counter's upper 8 bits
STK	Stack addressing mode	PC_L	Program counter's lower 8 bits
REL	Relative addressing mode	PG	Program bank register
DIR, b, REL	Direct bit relative addressing mode	DT	Data bank register
ABS, b, REL	Absolute bit relative addressing mode	DPR	Direct page register
SR	Stack pointer relative addressing mode	DPR_H	Direct page register's upper 8 bits
(SR), Y	Stack pointer relative indirect indexed Y addressing mode	DPR_L	Direct page register's lower 8 bits
		PS	Processor status register
BLK	Block transfer addressing mode	PS_H	Processor status register's upper 8 bits
C	Carry flag	PS_L	Processor status register's lower 8 bits
Z	Zero flag	PS_b	Processor status register's b-th bit
I	Interrupt disable flag	M(S)	Contents of memory at address indicated by stack pointer
D	Decimal operation mode flag		
x	Index register length selection flag	M_b	b-th memory location
m	Data length selection flag	AD_G	Value of 24-bit address's upper 8-bit (A_{23}~A_{16})
V	Overflow flag	AD_H	Value of 24-bit address's middle 8-bit (A_{15}~A_8)
N	Negative flag	AD_L	Value of 24-bit address's lower 8-bit (A_7~A_0)
IPL	Processor interrupt priority level	op	Operation code
+	Addition	n	Number of cycle
−	Subtraction	#	Number of byte
*	Multiplication	i	Number of transfer byte or rotation
/	Division	i_1, i_2	Number of registers pushed or pulled
$\wedge$	Logical AND		
$\vee$	Logical OR		

INSTRUCTION CODE TABLE-1

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 A,(DIR,X)		ORA A,SR	SEB DIR,b	ORA A,DIR	ASL DIR	ORA A,L(DIR)	PHP	ORA A,IMM	ASL A	PHD	SEB ABS,b	ORA A,ABS	ASL ABS	ORA A,ABL
0001	1	BPL	ORA A,(DIR),Y	ORA A,(DIR)	ORA A,(SR),Y	CLB DIR,b	ORA A,DIR,X	ASL DIR,X	ORA A,L(DIR),Y	CLC	ORA A,ABS,Y	DEC A	TAS	CLB ABS,b	ORA A,ABS,X	ASL ABS,X	ORA A,ABL,X
0010	2	JSR ABS	AND A,(DIR,X)	JSR ABL	AND A,SR	BBS DIR,b,R	AND A,DIR	ROL DIR	AND A,L(DIR)	PLP	AND A,IMM	ROL A	PLD	BBS ABS,b,R	AND A,ABS	ROL ABS	AND A,ABL
0011	3	BMI	AND A,(DIR),Y	AND A,(DIR)	AND A,(SR),Y	BBC DIR,b,R	AND A,DIR,X	ROL DIR,X	AND A,L(DIR),Y	SEC	AND A,ABS,Y	INC A	TSA	BBC ABS,b,R	AND A,ABS,X	ROL ABS,X	AND A,ABL,X
0100	4	RTI	EOR A,(DIR,X)	Note 1	EOR A,SR	MVP	EOR A,DIR	LSR DIR	EOR A,L(DIR)	PHA	EOR A,IMM	LSR A	PHG	JMP ABS	EOR A,ABS	LSR ABS	EOR A,ABL
0101	5	BVC	EOR A,(DIR),Y	EOR A,(DIR)	EOR A,(SR),Y	MVN	EOR A,DIR,X	LSR DIR,X	EOR A,L(DIR),Y	CLI	EOR A,ABS,Y	PHY	TAD	JMP ABL	EOR A,ABS,X	LSR ABS,X	EOR A,ABL,X
0110	6	RTS	ADC A,(DIR,X)	PER	ADC A,SR	LDM DIR	ADC A,DIR	ROR DIR	ADC A,L(DIR)	PLA	ADC A,IMM	ROR A	RTL	JMP (ABS)	ADC A,ABS	ROR ABS	ADC A,ABL
0111	7	BVS	ADC A,(DIR),Y	ADC A,(DIR)	ADC A,(SR),Y	LDM DIR,X	ADC A,DIR,X	ROR DIR,X	ADC A,L(DIR),Y	SEI	ADC A,ABS,Y	PLY	TDA	JMP (ABS,X)	ADC A,ABS,X	ROR ABS,X	ADC A,ABL,X
1000	8	BRA REL	STA A,(DIR,X)	BRA REL	STA A,SR	STY DIR	STA A,DIR	STX DIR	STA A,L(DIR)	DEY	Note 2	TXA	PHT	STY ABS	STA A,ABS	STX ABS	STA A,ABL
1001	9	BCC	STA A,(DIR),Y	STA A,(DIR)	STA A,(SR),Y	STY DIR,X	STA A,DIR,X	STX DIR,Y	STA A,L(DIR),Y	TYA	STA A,ABS,Y	TXS	TXY	LDM ABS	STA A,ABS,X	LDM ABS,X	STA A,ABL,X
1010	A	LDY IMM	LDA A,(DIR,X)	LDX IMM	LDA A,SR	LDY DIR	LDA A,DIR	LDX DIR	LDA A,L(DIR)	TAY	LDA A,IMM	TAX	PLT	LDY ABS	LDA A,ABS	LDX ABS	LDA A,ABL
1011	B	BCS	LDA A,(DIR),Y	LDA A,(DIR)	LDA A,(SR),Y	LDY DIR,X	LDA A,DIR,X	LDX DIR,Y	LDA A,L(DIR),Y	CLV	LDA A,ABS,Y	TSX	TYX	LDY ABS,X	LDA A,ABS,X	LDX ABS,Y	LDA A,ABL,X
1100	C	CPY IMM	CMP A,(DIR,X)	CLP IMM	CMP A,SR	CPY DIR	CMP A,DIR	DEC DIR	CMP A,L(DIR)	INY	CMP A,IMM	DEX	WIT	CPY ABS	CMP A,ABS	DEC ABS	CMP A,ABL
1101	D	BNE	CMP A,(DIR),Y	CMP A,(DIR)	CMP A,(SR),Y	PEI	CMP A,DIR,X	DEC DIR,X	CMP A,L(DIR),Y	CLM	CMP A,ABS,Y	PHX	STP	JMP L(ABS)	CMP A,ABS,X	DEC ABS,X	CMP A,ABL,X
1110	E	CPX IMM	SBC A,(DIR,X)	SEP IMM	SBC A,SR	CPX DIR	SBC A,DIR	INC DIR	SBC A,L(DIR)	INX	SBC A,IMM	NOP	PSH	CPX ABS	SBC A,ABS	INC ABS	SBC A,ABL
1111	F	BEQ	SBC A,(DIR),Y	SBC A,(DIR)	SBC A,(SR),Y	PEA	SBC A,DIR,X	INC DIR,X	SBC A,L(DIR),Y	SEM	SBC A,ABS,Y	PLX	PUL	JSR (ABS,X)	SBC A,ABS,X	INC ABS,X	SBC A,ABL,X

Note 1 : 42_{16} specifies the contents of the INSTRUCTION CODE TABLE-2.
About the second word's codes, refer to the INSTRUCTION CODE TABLE-2.

Note 2 : 89_{16} specifies the contents of the INSTRUCTION CODE TABLE-3.
About the third word's codes, refer to the INSTRUCTION CODE TABLE-2.

INSTRUCTION CODE TABLE-2 (The first word's code of each instruction is 42_{16})

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		ORA B,(DIR,X)		ORA B,SR		ORA B,DIR		ORA B,L(DIR)		ORA B,IMM	ASL B			ORA B,ABS		ORA B,ABL
0001	1		ORA B,(DIR),Y	ORA B,(DIR)	ORA B,(SR),Y		ORA B,DIR,X		ORA B,L(DIR),Y		ORA B,ABS,Y	DEC B	TBS		ORA B,ABS,X		ORA B,ABL,X
0010	2		AND B,(DIR,X)		AND B,SR		AND B,DIR		AND B,L(DIR)		AND B,IMM	ROL B			AND B,ABS		AND B,ABL
0011	3		AND B,(DIR),Y	AND B,(DIR)	AND B,(SR),Y		AND B,DIR,X		AND B,L(DIR),Y		AND B,ABS,Y	INC B	TSB		AND B,ABS,X		AND B,ABL,X
0100	4		EOR B,(DIR,X)		EOR B,SR		EOR B,DIR		EOR B,L(DIR)	PHB	EOR B,IMM	LSR B			EOR B,ABS		EOR B,ABL
0101	5		EOR B,(DIR),Y	EOR B,(DIR)	EOR B,(SR),Y		EOR B,DIR,X		EOR B,L(DIR),Y		EOR B,ABS,Y		TBD		EOR B,ABS,X		EOR B,ABL,X
0110	6		ADC B,(DIR,X)		ADC B,SR		ADC B,DIR		ADC B,L(DIR)	PLB	ADC B,IMM	ROR B			ADC B,ABS		ADC B,ABL
0111	7		ADC B,(DIR),Y	ADC B,(DIR)	ADC B,(SR),Y		ADC B,DIR,X		ADC B,L(DIR),Y		ADC B,ABS,Y		TDB		ADC B,ABS,X		ADC B,ABL,X
1000	8		STA B,(DIR,X)		STA B,SR		STA B,DIR		STA B,L(DIR)			TXB			STA B,ABS		STA B,ABL
1001	9		STA B,(DIR),Y	STA B,(DIR)	STA B,(SR),Y		STA B,DIR,X		STA B,L(DIR),Y	TYB	STA B,ABS,Y				STA B,ABS,X		STA B,ABL,X
1010	A		LDA B,(DIR,X)		LDA B,SR		LDA B,DIR		LDA B,L(DIR)	TBY	LDA B,IMM	TBX			LDA B,ABS		LDA B,ABL
1011	B		LDA B,(DIR),Y	LDA B,(DIR)	LDA B,(SR),Y		LDA B,DIR,X		LDA B,L(DIR),Y		LDA B,ABS,Y				LDA B,ABS,X		LDA B,ABL,X
1100	C		CMP B,(DIR,X)		CMP B,SR		CMP B,DIR		CMP B,L(DIR)		CMP B,IMM				CMP B,ABS		CMP B,ABL
1101	D		CMP B,(DIR),Y	CMP B,(DIR)	CMP B,(SR),Y		CMP B,DIR,X		CMP B,L(DIR),Y		CMP B,ABS,Y				CMP B,ABS,X		CMP B,ABL,X
1110	E		SBC B,(DIR,X)		SBC B,SR		SBC B,DIR		SBC B,L(DIR)		SBC B,IMM				SBC B,ABS		SBC B,ABL
1111	F		SBC B,(DIR),Y	SBC B,(DIR)	SBC B,(SR),Y		SBC B,DIR,X		SBC B,L(DIR),Y		SBC B,ABS,Y				SBC B,ABS,X		SBC B,ABL,X

INSTRUCTION CODE TABLE-3 (The first word's code of each instruction is 89_{16})

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		MPY (DIR,X)		MPY SR		MPY DIR		MPY L(DIR)		MPY IMM				MPY ABS		MPY ABL
0001	1		MPY (DIR),Y	MPY (DIR)	MPY (SR),Y		MPY DIR,X		MPY L(DIR),Y		MPY ABS,Y				MPY ABS,X		MPY ABL,X
0010	2		DIV (DIR,X)		DIV SR		DIV DIR		DIV L(DIR)	XAB	DIV IMM				DIV ABS		DIV ABL
0011	3		DIV (DIR),Y	DIV (DIR)	DIV (SR),Y		DIV DIR,X		DIV L(DIR),Y		DIV ABS,Y				DIV ABS,X		DIV ABL,X
0100	4										RLA IMM						
0101	5																
0110	6																
0111	7																
1000	8																
1001	9																
1010	A																
1011	B																
1100	C			LDT IMM													
1101	D																
1110	E																
1111	F																

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 i
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 7700 〈SOFT WARE〉 USER'S MANUAL

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

Published by
Mitsubishi Electric Corp., Semiconductor Marketing Division

MITSUBISHI SEMICONDUCTORS

MELPS 7700〈SOFT WARE〉

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

H-E0112-A KI-8907 Printed in Japan (ROD)

New publication, effective Jul. 1989.
Specifications subject to change without notice.