7. TLCS-47/470/470A Series Instruction Set

There are 90, 92, 105 basic machine instructions in the TLCS-47/470/470A series instruction set with software compatibility within the series. Table 1 (a) shows the instructions classified according to the number of bytes and the number of cycles. The TLCS-47/470/470A series instruction set mainly consists of 1-byte 1-cycle instructions, in consideration of program efficiency. In addition to basic machine instructions, expansion machine instructions are also available in order to improve coding efficiency. Refer to "TLCS-47/470/470A Development System Manual" for details concerning the expansion machine instructions.

Classification of instruction	Number of instruction		
	47	470	470A
1 - byte 1 - cycle instruction	40	37	37
1 - byte 2 - cycle instruction	11	11	11
2 - byte 2 - cycle instruction	39	43	45
3 - byte 3 - cycle instruction	0	1	12
Total	90	92	105

(a) Classification by number of byte / cycle

Classification of instruction	Number of instruction		
	47	470	470A
Data transfer			
• Load / Store / Move	14	14	14
• Table look-up	2	2	2
Exchange	5	5	5
Input / Output			
• Input	2	2	2
• Output	3	3	3
• 5-bit to 8-bit data conversion	1	1	1
Arithmetic & Logical operation			
• Compare	6	6	6
• Arithmetic (Addition and subtraction)	10	10	10
• Increment and decrement	6	6	6
• Logical	7	7	7
Rotation	2	2	2
Bit manipulation			
• Clear	4	4	4
• Set	4	4	4
• Test	7	7	7
Flag manipulation			
• Carry flag manipulation	2	2	2
• Zero flag manipulation	1	1	1
• General flag manipulation	3	0	0
Data Memory Bank control	0	4	13
Branch	2	3	3
Subroutine			
• Call	2	2	2
• Return	1	1	1
Interrupt control			
• EIR manipulation	1	1	1
• Interrupt latch, EIF manipulation	3	3	3
• Interrupt return	1	1	1
STK13 manipulation	0	0	4
CPU control			
• No operation	1	1	1
Total	90	92	105

(b) Classification by function

Table 1. Classification of instruction

7.1 Description of symbols

Symbol	Description
Acc	Accumulator
HR	H register
LR	L register
DMB	Data memory bank selector
DTB	Data table bank selector
DC	Data counter
PC	Program counter
SPW	Stack pointer word
STACK [SPW]	Stack (Stack location is indicated by the contents of stack pointer word)
SPW13	Stack pointer word for STK13
STK13 [SPW13]	Stack for PC_{13} (Stack location is indicated by the SPW13 or SPW)
FLAG	Flag
CF	Carry flag
$\overline{CF}$	Inversion of carry flag contents
ZF	Zero flag
SF	Status flag
EIR	Enable interrupt register
EIF	Enable interrupt master Flip-Flop
IL	Interrupt latch
RAM [x]	Data memory (Address x) (Note 1)
RAM [y]	Data memory (Address y in bank0)
RAM [HL]	Data memory (Address HL) (Note 1)
$RAM [HL]_b$	Contents of bit assigned by b of data memory (Address HL) (Note 1)
Acc_b	Contents of bit assigned by b of accumulator

Symbol	Description
$ROM [DTB \cdot DC]_L$	Lower 4bits of program memory (Note 2) (Address DTB and DC)
$ROM [DTB \cdot DC]_H$	Higher 4bits of program memory (Note 2) (Address DTB and DC)
PORT [p]	Port (Address p)
←	Transfer
↔	Exchange
+	Addition
−	Subtraction
∧	Logical AND of the corresponding bits
∨	Logical OR of the corresponding bits
∀	Exclusive OR of the corresponding bits
·	Concatenation
null	No operation
PC_{12-6}	Bit 12 and bit 6 of program counter
LR_{3-2}	Contents of bit assigned by bit 3 and bit 2 of L register
LR_{1-0}	Contents of bit assigned by bit 1 and bit 0 of L register

Note 1: TLCS-470, 470A need to set DMB otherwise.

Note 2: TLCS-47 regard DTB as "0".
TLCS-470 regard DTB as "1".

Data	Executed instruction	Description
—		The flag is not set and retains its value before instruction execution.
Carry or borrow information C	Addition	Sets "1" if an overflow occurs in the MSB. Otherwise, sets "0".
	Subtraction /comparison	Sets "0" if a borrow occurs from the MSB, that is, the LSB of the following address. Otherwise, sets "1".
	Rotation	Sets data shifted out from the accumulator.
$\overline{C}$		Inverted carry data value
Zero detect information	Data transfer or exchange Input or rotation	Sets "1" if the data moved to the accumulator is "0000_B". Otherwise, sets "0".
Z	Operation	Sets "1" if the operation result is "0000_B". Otherwise, sets "0".
$\overline{Z}$		Inverted zero detect data value
*		Sets the value specified by the operation.

7.2 Description of instructions

(1) Data transfer

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
LD A, @HL	0000 1100	0C	Acc←RAM[HL]	-	Z	1	1	○	○	○
	Loads the contents of data memory at the address specified by the HL register pair in the accumulator. With the TLCS-470 and 470A series, the bank need to be specified by DMB.									
LD A, x	0011 1100 xxxx xxxx	3C xx	Acc←RAM[x]	-	Z	1	2	○	○	○
	Loads the contents of data memory at the address specified by operand x in the accumulator. With the TLCS-470 and 470A series, the bank need to be specified by DMB.									
LD HL, x	0010 1000 xxxx xx00	28 xx	LR←RAM[x], HR←RAM[x+1]	-	-	1	2	○	○	○
	Loads consecutive two-words from the data memory address specified by operand x in the L and H register pair. However, the lower two bits at the address of x must be "0". That is, the lower four bits must be 0_H or 4_H or 8_H or C_H. With the TLCS-470 and 470A series, the bank need to be specified by DMB. Example : DMB = 0 and addresses 74 and 75_H contain the values 3 and 9 respectively. Executing LD HL, 74_H results in HR = 9 and LR = 3 (TLCS-470 series).									
LDL A, @DC	0011 0011	33	Acc←ROM[DC]L	-	Z	1	2	○	○	○
	Loads the lower four bits of the data read from the data table in program memory in the accumulator. The data table address is specified by the data counter. (The address range is 000_H to FFF_H for the TLCS-47 series and 1000_H to $1FFF_H$ for the TLCS-470 series. The program memory bank is specified by the data table bank selector (DTB) in the TLCS-470A series.) The zero flag is set by the data loaded to the accumulator.									
LDH A, @DC+	0011 0010	32	Acc←ROM[DC]H, DC←DC+1	-	Z	1	2	○	○	○
	Loads the higher four bits of the data read from the data table in program memory in the accumulator. The data table address is specified by the data counter. The data counter is incremented at the completion of the operation. (The address range is 000_H to FFF_H for the TLCS-47 series and 1000_H to $1FFF_H$ for the TLCS-470 series. The program memory bank is specified by the data table bank selector (DTB) in the TLCS-470A series.) The zero flag is set by the data loaded to the accumulator. Example 1 : $DC = 0C70_H$ and the content of program memory at address $1C70_H$ is $9E_H$. Executing LDH A, @DC + results in Acc = 9_H and DC = $0C71_H$. (TLCS-470 series). Example 2 : $DC = 0C70_H$, DTB = 3_H, and the contents of program memory address $3C70_H$ are $9E_H$. Executing LDH A, @DC + results in Acc = 9_H, DC = $0C71_H$, and DTB = 3. (TLCS-470A series).									
ST A, @HL	0000 1111	0F	RAM[HL]←Acc	-	-	1	1	○	○	○
	Stores the contents of the accumulator in data memory at the address specified by the HL register pair. With the TLCS-470 and 470A series, the bank need to be specified by DMB.									
ST A, @HL+	0001 1010	1A	RAM[HL]←Acc, LR←LR+1	-	Z	C	1	○	○	○
	Stores the contents of the accumulator in data memory at the address specified by the HL register pair. The L register is incremented at the completion of the operation. The status flag and zero flag are set by the increment result of the L register.									
ST A, @HL−	0001 1011	1B	RAM[HL]←Acc, LR←LR−1	-	Z	$\overline{C}$	1	○	○	○
	Stores the contents of the accumulator in data memory at the address specified by the HL register pair. The L register is decremented at the completion of the operation. The status flag and zero flag are set by the decrement result of the L register. Example : DMB = 1_H, HR = 7_H, LR = 0_H, and Acc = 3_H. Executing ST A, @HL- writes 3_H to address 70_H of bank 1, and results in HR = 7_H, LR = F_H, SF = 0, and ZF = 0 (TLCS-470 series).									
ST A, x	0011 1111 xxxx xxxx	3F xx	RAM[x]←Acc	-	-	1	2	○	○	○
	Stores the contents of the accumulator in data memory at the address specified by operand x. With the TLCS-470 and 470A series, the bank need to be specified by DMB.									

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
ST #k, @HL+	1111 kkkk	Fk	RAM[HL]←k, LR←LR+1	-	Z	C	1	○	○	○
	Stores immediate value k in data memory at the address specified by the HL register pair. The L register is incremented at the completion of the operation. Specifying k = 0_H is equivalent to the clear instruction. The status flag and zero flag are set by the increment result of the L register. For example, the status flag is cleared to "0" if the content of the L register before execution is $0F_H$.									
ST #k, y	0010 1101 kkkk yyyy	2D ky	RAM[y]←k	-	-	1	2	○	○	○
	Stores immediate value k in the address specified by y in page 0 of bank 0 of data memory (address range 00_H to 0FH). Specifying k = 0_H is equivalent to the clear instruction.									
LD A, #k	0100 kkkk	Ck	Acc←k	-	Z	1	1	○	○	○
	Loads immediate value k in the accumulator. Specifying k = 0_H is equivalent to the clear instruction. Example : Executing LD A, #5 results in Acc = 5_H and ZF = 0.									
LD H, #k	1100 kkkk	4k	HR←k	-	-	1	1	○	○	○
	Loads immediate value k in the H register. Specifying k = 0_H is equivalent to the clear instruction. Example : Executing LD H, #$0E_H$ results in HR = $0E_H$.									
LD L, #k	1110 kkkk	Ek	LR←k	-	-	1	1	○	○	○
	Loads immediate value k in the L register. Specifying k = 0_H is equivalent to the clear instruction. Example : Executing LD L, #3_H results in LR = 3_H.									
LD HL, #jk	1100 jjjj 1110 kkkk	cj Ek	HR←j LR←k	-	-	1	2	○	○	○
	Loads the immediate data jk of the instruction field in the pair of the H and L registers.									
MOV H, A	0001 0000	10	Acc←HR	-	Z	1	1	○	○	○
	Loads the contents of the H register in the accumulator. Sets the zero flag to 1 when HR = 0_H; clears the zero flag to "0" when HR ≠ 0_H.									
MOV L, A	0001 0001	11	Acc←LR	-	Z	1	1	○	○	○
	Loads the contents of the L register in the accumulator. Sets the zero flag to "1" when LR = 0_H; clears the zero flag to "0" when LR ≠ 0_H.									

(2) Exchange

Mnemonic	Object code		Operation	Flags			Cycle(s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
XCH A, @HL	0000 1101	0D	Acc⇔RAM[HL]	-	Z	1	1	○	○	○
	Exchanges the contents of the accumulator with the contents of data memory at the address specified by the HL register pair. Sets the zero flag to "1" when the content of RAM[HL] is 0000_B before execution.									
XCH A, x	0011 1101 xxxx xxxx	3D xx	Acc⇔RAM[x]	-	Z	1	2	○	○	○
	Exchanges the contents of the accumulator with the contents of data memory at the address specified by operand x. Sets the zero flag to "1" when the content of RAM[x] is 0000_B before execution.									
XCH HL, x	0010 1001 xxxx xx00	29 xx	LR⇔RAM[x], HR⇔RAM[x+1]	-	-	1	2	○	○	○
	Exchanges the contents of the HL registers with the two consecutive memory words at the address specified by operand x (the lower two bits of x must be "0").									
XCH A, H	0011 0000	30	Acc⇔HR	-	Z	1	2	○	○	○
	Exchanges the contents of the H register with the contents of the accumulator. Sets the zero flag to "1" when HR is 0000_B before the exchange. Example : Acc = 4_H and HR = 8_H. Executing XCH A, H results in Acc = 8_H, HR = 4_H, and ZF = 0.									
XCH A, L	0011 0001	31	Acc⇔LR		Z	1	2	○	○	○
	Exchanges the contents of the L register with the contents of the accumulator. Sets the zero flag to "1" when LR is 0000_B before the exchange. Example: Acc = B_H and LR = 0_H. Executing XCH A, L results in Acc = 0_H, LR = B_H, and ZF = 1.									

(3) Input / Output

Mnemonic	Object code		Operation	Flags			Cycle(s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
IN %p, A	0011 1010 0010 pppp	3A 2p	Acc←PORT[p]	-	Z	$\overline{Z}$	2	○	○	○
	Inputs data from the port specified by operand p (port number : 00 to 0F) to the accumulator. Clears the status flag to "0" if the input data are 0000_B. Otherwise, sets the status flag to "1".									
IN %p, A	0011 1010 0000 pppp	3A 0p	Acc←PORT[p]	-	Z	1	2	-	○	○
	Inputs data from the port specified by operand p (port number : 10_H to $1F_H$) to the accumulator.									
IN %p, @HL	0011 1010 0110 pppp	3A 6p	RAM[HL]←PORT[p]	-	-	$\overline{Z}$	2	○	○	○
	Inputs data from the port specified by operand p (port number : 00_H to $0F_H$) to the address in data memory specified by the HL register pair.									
IN %p, @HL	0011 1010 0100 pppp	3A 4p	RAM[HL]←PORT[p]	-	-	1	2	-	○	○
	Inputs data from the port specified by operand p (port number : 10_H to $1F_H$) to the address in data memory specified by the HL register pair.									
OUT A, %p	0011 1010 $10\overline{p}0$ pppp	3A $8+2\overline{p}\cdot p$	PORT[p]←Acc	-	-	1	2	○	○	○
	Outputs the contents of the accumulator to the port specified by operand p (port number : 00_H to $1F_H$). Example : When Acc = 6_H, executing OUT A, %04_H outputs value 6 to port R4.									
OUT @HL, %p	0011 1010 $11\overline{p}0$ pppp	3A $0C_H+2\overline{p}\cdot p$	PORT[p]←RAM[HL]	-	-	1	2	○	○	○
	Outputs the contents of data memory at the address specified by the HL register pair to the port specified by operand p (port number : 00_H to $1F_H$).									

Mnemonic	Object code		Operation	Flags			Cycle(s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
OUT #k, %p	0010 1100 kkkk pppp	2C kp	PORT[p]←k	-	-	1	2	○	○	○
	Outputs immediate value k to the port specified by operand p (port number : 00 to 0F).									
OUTB @HL	0001 0010	12	PORT[2] · PORT[1]← ROM[0FE0$_H$+CF · RAM[HL]]	-	-	1	2	○	○	○
	Reads the 8-bit data from the data table in program memory and outputs the lower four bits to port P1 and the higher four bits to port P2. The address of the contents to be read, is the sum of the offset value, and the 5-bit value, determined by linking the carry flag and the contents at the data memory address specified by the HL register pair. Example : (If program memory size is 4K bytes) CF = 1, HR = 8_H, LR = 2_H and the content of data memory address 82_H is 5_H. The 5-bit value determined by linking the carry flag and the data at the address specified by the HL register pair is 15_H (10101_B). Adding the offset value (= $0FE0_H$) (see note) to 5-bit value 15_H results in a data table address of $0FF5_H$ in program memory. If the contents at this address ($0FF5_H$) is $6D_H$, 6_H is output to port P2 and $0D_H$ is output to port P1. Note : The data table is allocated to the final 32 bytes of program memory. Consequently, the offset value varies according to the size of program memory. For example, if the program memory size is 4K bytes, the offset value is $0FE0_H$. If the program memory is 16K bytes, the offset value is $3FE0_H$.									

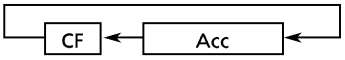
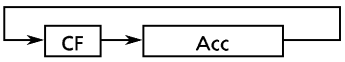
(4) Arithmetic and Logical Operation

Mnemonic	Object code		Operation	Flags			Cycle(s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
CMPR A, @HL	0001 0110	16	RAM[HL]−Acc	C	Z	$\overline{Z}$	1	○	○	○
	Compares the contents of the accumulator with the contents of data memory at the address specified by the HL register pair. Sets the carry flag to "1" when RAM[HL]≧Acc; clears the carry flag to "0" when RAM[HL]<Acc.									
CMPR A, x	0011 1110 xxxx xxxx	3E xx	RAM[x]−Acc	C	Z	$\overline{Z}$	2	○	○	○
	Compares the contents of the accumulator with the contents of data memory at the address specified by operand x. Sets the carry flag to "1" when RAM[x]≧Acc; clears the carry flag to "0" when RAM[x]<Acc.									
CMPR y, #k	0010 1110 kkkk yyyy	2E ky	k−RAM[y]	C	Z	$\overline{Z}$	2	○	○	○
	Compares the contents of the address specified by operand y in page 0 of bank 0 of data memory with immediate value k. Specifying k = 0_H is equivalent to the data memory test instruction . Sets the carry flag to "1" when RAM[y]≦k; clears the carry flag to "0" when RAM[y]>k.									
CMPR A, #k	1101 kkkk	Dk	k−Acc	C	Z	$\overline{Z}$	1	○	○	○
	Compares the accumulator contents with immediate value k. Specifying k = 0_H is equivalent to the accumulator test instruction. Sets the carry flag to "1" when Acc≦k; clears the carry flag to "0" when Acc>k. Example : When Acc = 2_H, executing CMPR A, #1 results in SF = 1, ZF = 0, and CF = 0.									
CMPR H, #k	0011 1000 1101 kkkk	38 Dk	k−HR	-	Z	C	2	○	○	○
	Compares the contents of the H register with immediate value k. Specifying k = 0_H is equivalent to the H register test instruction. Sets the status flag to "1" when HR≦k; clears the status flag to "0" when HR>k. Example : When HR = 3_H, executing CMPR H, #5 results in SF = 1 and ZF = 0.									
CMPR L, #k	0011 1000 1001 kkkk	38 9k	k−LR	-	Z	C	2	○	○	○
	Compares the contents of the L register with immediate value k. Specifying k = 0_H is equivalent to the L register test instruction. Sets the status flag to "1" when LR≦k; clears the status flag to "0" when LR>k.									

Mnemonic	Object code		Operation	Flags			Cycle(s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
ADD A, @HL	0001 0111	17	Acc←Acc+RAM[HL]	-	Z	$\overline{C}$	1	○	○	○
	Adds the contents of data memory at the address specified by the HL register pair, to the contents of the accumulator. Loads the result to the accumulator.									
ADDC A, @HL	0001 0101	15	Acc←Acc+RAM[HL]+CF	C	Z	$\overline{C}$	1	○	○	○
	Adds the carry flag and contents of data memory at the address specified by the HL register pair to the contents of the accumulator. Loads the result to the accumulator.									
ADD @HL, #k	0011 1000 0100 kkkk	38 4k	RAM[HL]←RAM[HL]+k	-	Z	$\overline{C}$	2	○	○	○
	Adds immediate value k to the contents of data memory at the address specified by the HL register pair. Loads the result to data memory. Specifying $k = 6_H$ or $k = -10_H$ is equivalent to the hexadecimal to decimal conversion instruction.									
ADD y, #k	0010 1111 kkkk yyyy	2F ky	RAM[y]←RAM[y]+k	-	Z	$\overline{C}$	2	○	○	○
	Adds immediate value k to the contents at the address specified by operand y in page 0 of bank 0 of data memory. Loads the result to data memory.									
ADD A, #k	0011 1000 0000 kkkk	38 0k	Acc←Acc+k	-	Z	$\overline{C}$	2	○	○	○
	Adds immediate value k to the contents of the accumulator. Loads the result to the accumulator. Specifying $k = 6_H$ or $k = -10_H$ is equivalent to the hexadecimal to decimal conversion instruction. Example : When Acc = 9_H, executing ADD A, #$0E_H$ results in Acc = 7_H, SF = 0, and ZF = 0.									
ADD H, #k	0011 1000 1100 kkkk	38 Ck	HR←HR+k	-	Z	$\overline{C}$	2	○	○	○
	Adds immediate value k to the contents of the H register. Loads the result to the H register. Specifying $k = 1_H$ is equivalent to the H register increment instruction, $k = 0F_H$, to the H register decrement instruction.									
ADD L, #k	0011 1000 1000 kkkk	38 8k	LR←LR+k	-	Z	$\overline{C}$	2	○	○	○
	Adds immediate value k to the contents of the L register. Loads the result to the L register. Specifying $k = 1_H$ is equivalent to the L register increment instruction, $k = 0F_H$, the L register decrement instruction (see the INC L and DEC L instructions).									
SUBRC A, @HL	0001 0100	14	Acc←RAM[HL]−Acc−$\overline{CF}$	C	Z	C	1	○	○	○
	Subtracts the contents of the accumulator and the inverted contents of the carry flag from the contents of data memory at the address specified by the HL register pair. Loads the result to the accumulator.									
SUBR @HL, #k	0011 1000 0101 kkkk	38 5k	RAM[HL]←k−RAM[HL]	-	Z	C	2	○	○	○
	Subtracts the contents of data memory at the address specified by the HL register pair from immediate value k. Loads the result to the data memory address. Specifying $k = 0_H$ is equivalent to the data memory twos complement instruction; $K = 0F_H$, to the ones complement (data inversion) instruction.									
SUBR A, #k	0011 1000 0001 kkkk	38 1k	Acc←k−Acc	-	Z	C	2	○	○	○
	Subtracts the contents of the accumulator from immediate value k and places the result in the accumulator. Equivalent to the accumulator twos complement instruction if $k = 0_H$ or the accumulator ones complement (data inversion) instruction if $k = 0F_H$. Example : When Acc = 4_H, executing SUBR A,#$0B_H$ results in Acc = 7_H, SF = 1, and ZF = 0.									
INC @HL	0000 1010	0A	RAM[HL]←RAM[HL]+1	-	Z	$\overline{C}$	1	○	○	○
	Increments the contents of data memory at the address specified by the HL register pair.									
DEC @HL	0000 1011	0B	RAM[HL]←RAM[HL]−1	-	Z	C	1	○	○	○
	Decrements the contents of data memory at the address specified by the HL register pair.									
INC A	0000 1000	08	Acc←Acc+1	-	Z	$\overline{C}$	1	○	○	○
	Increments the contents of the accumulator. Example 1 : When Acc = 7_H, executing INC A results in Acc = 8_H, SF = 1, and ZF = 0. Example 2 : When Acc = $0F_H$, executing INC A results in Acc = 0_H, SF = 0, and ZF = 1.									

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
DEC A	0000 1001	09	Acc←Acc−1	-	Z	C	1	○	○	○
	Decrements the contents of the accumulator. Example 1 : when Acc = 5_H, executing DEC A results in Acc = 4, SF = 1, and ZF = 0.									
INC L	0001 1000	18	LR←LR+1	-	Z	$\overline{C}$	1	○	○	○
	Increments the contents of the L register.									
DEC L	0001 1001	19	LR←LR−1	-	Z	C	1	○	○	○
	Decrements the contents of the L register.									
AND A, @HL	0001 1110	1E	Acc←Acc∧RAM[HL]	-	Z	$\overline{Z}$	1	○	○	○
	Performs a bitwise logical AND on the contents of data memory at the address specified by the HL register pair and the contents of the accumulator. Loads the result to the accumulator.									
AND @HL, #k	0011 1000 0111 kkkk	38 7k	RAM[HL]←RAM[HL]∧k	-	Z	$\overline{Z}$	2	○	○	○
	Performs a bitwise logical AND on the contents of data memory at the address specified by the HL register pair and immediate value k. Loads the result to data memory.									
AND A, #k	0011 1000 0011 kkkk	38 3k	Acc←Acc∧k	-	Z	$\overline{Z}$	2	○	○	○
	Performs a bitwise logical AND on immediate value k and the contents of the accumulator. Loads the result to the accumulator. Example : When Acc = 0011_B, executing AND A, #0101_B results in Acc = 0001_B, SF = 1, and ZF = 0.									
OR A, @HL	0001 1101	1D	Acc←Acc∨RAM[HL]	-	Z	$\overline{Z}$	1	○	○	○
	Performs a bitwise logical OR on the contents of data memory at the address specified by the HL register pair and the contents of the accumulator. Loads the result to the accumulator.									
OR @HL, #k	0011 1000 0110 kkkk	38 6k	RAM[HL]←RAM[HL]∨k	-	Z	$\overline{Z}$	2	○	○	○
	Performs a bitwise logical OR on the contents of data memory at the address specified by the HL register pair and immediate value k. Loads the result to data memory.									
OR A, #k	0011 1000 0010 kkkk	38 2k	Acc←Acc∨k	-	Z	$\overline{Z}$	2	○	○	○
	Performs a bitwise logical OR on immediate value k and the contents of the accumulator. Loads the result to the accumulator. Example : When Acc = 1100_B, executing OR A, #0101_B results in Acc = 1101_B, SF = 1, and ZF = 0.									
XOR A, @HL	0001 1111	1F	Acc←Acc∨RAM[HL]	-	Z	$\overline{Z}$	1	○	○	○
	Performs a bitwise exclusive OR on the contents of data memory at the address specified by the HL register pair and the contents of the accumulator. Loads the result to the accumulator. Example : HR = 2_H, LR = 5_H, Acc = 0110_B, and the content of data memory address 25_H is 1010_B. Executing XOR A, @HL results in Acc = 1100_B, SF = 1, and ZF = 0. (TLCS-47 series).									

(5) Rotation

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
ROLC A	0000 0101	05	 (rotate left by 1bit)	*	Z	$\overline{C}$	1	○	○	○
	Cyclically rotates the contents of the accumulator and the carry flag by one bit to the left. Loads the inverted value of the data shifted out of the accumulator into the carry flag, to the status flag. Equivalent to the left shift instruction (twice contents of the accumulator) if the carry flag is "0" before execution. Example : When Acc = 1011_B and CF = 0, executing ROLC A results in Acc = 0110_B, CF = 1, SF = 0, and ZF = 0.									
RORC A	0000 0111	07	 (rotate right by 1bit)	*	Z	$\overline{C}$	1	○	○	○
	Cyclically rotates the contents of the accumulator and carry flag by one bit to the right. Loads the inverted value of the data shifted out of the accumulator into the carry flag, to the status flag. Equivalent to the right shift instruction (half contents of the accumulator) if the carry flag is "0" before execution. Example : When Acc = 0100_B and CF = 1, executing RORC A results in Acc = 1010_B, CF = 0, SF = 1, and ZF = 0.									

(6) Bit Manipulation

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
CLR @HL, b	0101 01bb	5•4+b	RAM[HL]b←0	-	-	1	1	○	○	○
	Clears a bit in data memory. Operand b specifies the bit and the HL register pair specifies the data memory address.									
CLR y, b	0011 1001 01bb yyyy	39 4+b•y	RAM[y]b←0	-	-	1	2	○	○	○
	Clears a bit in page 0 of bank 0 of data memory. Operand b specifies the bit and operand y specifies the address in page 0 of bank 0 of data memory. Example : When the content of address $0C_H$ of bank 0 is 1101_B, executing CLR $0C_H$, 2 results in the content of address $0C_H$ changing to 1001_B.									
CLR %p, b	0011 1011 01bb pppp	3B 4+b•p	PORT[p]b←0	-	-	1	2	○	○	○
	Clears a bit in a port. Operand b specifies the bit and operand p specifies the port (port number : 00_H to $0F_H$).									
CLR @L	0011 0101	35	PORT[LR_{3-2}+4]LR_{1-0}←0	-	-	1	2	○	○	○
	Clears the bit specified by the lower two bits of the L register in the port specified by the higher two bits of the L register (port 4 (00) to 7 (11)). Example : When LR = 0110_B and the output latch of port R5 is 1101_B, executing CLR @L results in the output latch of port R5 changing to 1001_B.									
SET @HL, b	0101 00bb	5•0+b	RAM[HL]b←1	-	-	1	1	○	○	○
	Sets a bit in data memory. Operand b specifies the bit and the data memory is specified by the HL register pair.									
SET y, b	0011 1001 00bb yyyy	39 by	RAM[y]b←1	-	-	1	2	○	○	○
	Sets a bit in page 0 of bank 0 of data memory. Operand b specifies the bit and operand y specifies the address in page 0 of bank 0 of data memory.									

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
SET %p, b	0011 1011 00bb pppp	3B bp	PORT[p]$_b$←1	-	-	1	2	○	○	○
	Sets a bit in a port. Operand b specifies the bit and operand p specifies the port (port number : 00_H to $0F_H$).									
SET @L	0011 0100	34	PORT[LR_{3-2}+4]$_{LR1-0}$←1	-	-	1	2	○	○	○
	Sets the bit specified by the lower two bits of the L register in the port specified by the higher two bits of the L register (port 4 (00) to 7 (11)).									
TEST @HL, b	0101 10bb	5•8+b	SF←$\overline{RAM[HL]_b}$	-	-	*	1	○	○	○
	Loads the inverted bit of data memory in the status flag. Operand b specifies the bit and the data memory is specified by the HL register pair.									
TEST y, b	0011 1001 10bb yyyy	39 8+b•y	SF←$\overline{RAM[y]_b}$	-	-	*	2	○	○	○
	Loads the inverted bit in page 0 of bank 0 of data memory in the status flag. Operand b specifies the bit and the data memory in page 0 of bank 0 is specified by operand y.									
TESTP y, b	0011 1001 11bb yyyy	39 C+b•y	SF←RAM[y]$_b$	-	-	*	2	○	○	○
	Loads a bit in page 0 of bank 0 of data memory in the status flag. Operand b specifies the bit and the data memory in page 0 of bank 0 is specified by operand y.									
TEST %p, b	0011 1011 10bb pppp	3B 8+b•p	SF←$\overline{PORT[p]_b}$	-	-	*	2	○	○	○
	Loads the inverted bit from a port in the status flag. Operand b specifies the bit and operand p (port number : 00_H to $0F_H$) specifies the port (output latch if output port; input pin if I/O or input port).									
TESTP %p, b	0011 1011 11bb pppp	3B C+b•p	SF←PORT[p]$_b$	-	-	*	2	○	○	○
	Loads a bit from a port in the status flag. Operand b specifies the bit and operand p (port number : 00_H to $0F_H$) specifies the port (output latch if output port; input pin if I/O or input port).									
TEST @L	0011 0111	37	SF←$\overline{PORT[LR_{3-2}+4]_{LR1-0}}$	-	-	*	2	○	○	○
	Loads the inverted bit from a port in the status flag. The lower two bits of the L register specify the bit and the higher two bits of the L register specify the port (port 4 (00) to 7 (11)).									
TEST A, b	0101 11bb	5•C+b	SF←$\overline{Acc_b}$	-	-	*	1	○	○	○
	Loads the inverted bit of the accumulator in the status flag. Operand b specifies the bit. Example : When Acc = 1001_B, executing TEST A, 3 results in SF = 0.									

(7) Flag Manipulation

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
TEST CF	0000 0110	06	SF←$\overline{CF}$, CF←0	0	-	*	1	○	○	○
	Loads the inverted contents of the carry flag in the status flag, then clears the carry flag to "0". Example : If CF = 1, executing TEST CF results in SF = 0 and CF = 0.									
TESTP CF	0000 0100	04	SF←CF, CF←1	1	-	*	1	○	○	○
	Loads the contents of the carry flag in the status flag, then sets the carry flag to "1". Example : If CF = 0, executing TEST PCF results in SF = 0 and CF = 1.									
TESTP ZF	0000 1110	0E	SF←ZF	-	-	*	1	○	○	○
	Loads the contents of the zero flag in the status flag.									
CLR GF	0000 0010	02	GF←0	-	-	1	1	○	-	-
	Clears the contents of the general flag to "0".									
SET GF	0000 0011	03	GF←1	-	-	1	1	○	-	-
	Sets the contents of the general flag to "1".									
TESTP GF	0000 0001	01	SF←GF	-	-	*	1	○	-	-
	Loads the contents of the general flag in the status flag.									

(8) Data Memory Bank Control

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
CLR DMB	0011 1011 0111 1001	3B 79	DMB←0	-	-	1	2	-	○	-
	Clears the data memory bank selector to "0".									
SET DMB	0011 1011 0011 1001	3B 39	DMB←1	-	-	1	2	-	○	-
	Sets the data memory bank selector to "1".									
TEST DMB	0011 1011 1011 1001	3B B9	SF←$\overline{DMB}$	-	-	*	2	-	○	-
	Loads the inverted contents of the data memory bank selector in the status flag.									
TESTP DMB	0011 1011 1111 1001	3B F9	SF←DMB	-	-	*	2	-	○	-
	Loads the contents of the data memory bank selector in the status flag.									
CLR DMB0	0011 1011 0111 1001	3B 79	DMB_0←0	-	-	1	2	-	-	○
	Clears data memory bank selector 0 to "0".									
SET DMB0	0011 1011 0011 1001	3B 39	DMB_0←1	-	-	1	2	-	-	○
	Sets data memory bank selector 0 to "1".									
TEST DMB0	0011 1011 1011 1001	3B B9	SF←$\overline{DMB_0}$	-	-	1	2	-	-	○
	Loads the inverted contents of the data memory bank selector 0 in the status flag.									
TESTP DMB0	0011 1011 1111 1001	3B F9	SF←DMB_0	-	-	1	2	-	-	○
	Loads the contents of the data memory bank selector 0 in the status flag.									

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
CLR DMB1	0000 0011 0011 1011 0101 1001	03 3B 59	$DMB_1 \leftarrow 0$	-	-	1	3	-	-	○
	Clears data memory bank selector 1 to "0".									
SET DMB1	0000 0011 0011 1011 0001 1001	03 3B 19	$DMB_1 \leftarrow 1$	-	-	1	3	-	-	○
	Sets data memory bank selector 1 to "1".									
TEST DMB1	0000 0011 0011 1011 1001 1001	03 3B 99	$SF \leftarrow \overline{DMB_1}$	-	-	*	3	-	-	○
	Loads the inverted contents of the data memory bank selector 1 in the status flag.									
TESTP DMB1	0000 0011 0011 1011 1101 1001	03 3B D9	$SF \leftarrow DMB_1$	-	-	*	3	-	-	○
	Loads the contents of the data memory bank selector 1 in the status flag.									
LD DMB, #k	0000 0011 0010 1100 00kk 1001	03 2C k9	$DMB \leftarrow k$	-	-	1	3	-	-	○
	Loads immediate value k (2 bits) in the data memory bank selector.									
LD DMB, @HL	0000 0011 0011 1010 1110 1001	03 3A E9	$DMB \leftarrow RAM[HL]_{1-0}$	-	-	1	3	-	-	○
	Loads the lower two bits of the contents of a data memory in the data memory bank selector. The data memory is specified by the HL register pair.									
MOV A, DMB	0000 0011 0011 1010 1010 1001	03 3A A9	$DMB \leftarrow Acc_{1-0}$	-	-	1	3	-	-	○
	Loads the lower two bits of the accumulator in the data memory bank selector. Example : When Acc = 1101_B, executing MOV A, DMB results in DMB = 01_B.									
MOV DMB, A	0000 0011 0011 1010 0010 1001	03 3A 29	$Acc_{1-0} \leftarrow DMB$ $Acc_{3-2} \leftarrow 0$	-	Z	$\overline{Z}$	3	-	-	○
	Loads the contents of the data memory bank selector to the lower two bits of the accumulator. Loads "0" in the higher two bits of the accumulator. Example : When DMB = 10_B, executing MOV DMB, A results in Acc = 0010_B, ZF = 0, and SF = 1.									
ST DMB, @HL	0000 0011 0011 1010 0110 1001	03 3A 69	$RAM[HL]_{1-0} \leftarrow DMB$ $RAM[HL]_{3-2} \leftarrow 0$	-	-	$\overline{Z}$	3	-	-	○
	Stores the contents of the data memory bank selector in the lower two bits of the data memory at the address specified by the HL register pair. Stores "0" in the higher two bits of data memory.									

(9) Branch

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
BSS a	10dd dddd	8•d+d	if SF=1 then PC←a(a=PC_{13-6}•d)else null	-	-	1	1	○	○	○
	<branch within a page> Performs a branch within a page if the status flag is "1". The effect is to load immediate value d to the lower six bits (00_H to $03F_H$) of the 14-bit program counter (TLCS-470A series) after the program counter is incremented at the completion of the instruction. The higher eight bits retain their values. Therefore, if this instruction appears at the final address in a page, the instruction branches to an address in the next page. When the status flag is "0", the instruction sets the status flag to "1" and executes the instruction at the next address.									
BS a	0110 eeee eeee eeee	6e ee	if SF=1 then PC←a(a=PC_{13-12}•e)else null	-	-	1	2	○	○	○
	<branch within a bank> • For TLCS-47 series If the status flag is "1", loads immediate value a to the program counter. If the status flag is "0", sets the status flag to "1" and executes the instruction at the next address. • For TLCS-470 and 470A series Performs a branch within a bank if the status flag is "1". The effect is to load immediate value e to the lower twelve bits (00_H to $03F_H$) of the program counter after the program counter is incremented at the completion of the instruction. The higher two bits retain their values. Therefore, if this instruction appears at the final address in a bank, the instruction branches to an address in the next bank. When the status flag is "0", the instruction sets the status flag to "1" and executes the instruction at the next address.									
BSL a	0000 0010 0110 eeee eeee eeee	02 6e ee	if SF=1 then PC←a(a=0•0•e)else null	-	-	1	3	-	○	○
	0000 0011 0110 eeee eeee eeee	03 6e ee	if SF=1 then PC←a(a=0•1•e)else null	-	-	1	3			
	0001 1100 0110 eeee eeee eeee	1C 6e ee	if SF=1 then PC←a(a=1•0•e)else null	-	-	1	3			
	0000 0001 0110 eeee eeee eeee	01 6e ee	if SF=1 then PC←a(a=1•1•e)else null	-	-	1	3			
	If the status flag is "1", loads immediate value a to the program counter. When the status flag is "0", the instruction sets the status flag to "1" and executes the instruction at the next address.									

Note: TLCS-47 Assembler optimizes B instruction (extended instruction) to BSS or BS BSL instruction automatically. So users have only to describe B instruction.

(10) Subroutines

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
CALLS a	0111 nnnn	7n	STACK[SPW]←PC_{12-0}, STK13[SPW]←PC_{13}, SPW←SPW-1, PC←a a=8n+6(n≠0), 0086_H(n=0)	-	-	-	2	○	○	○
	Performs a short-form subroutine call. The instruction saves the contents of the program counter to the stack (location : SPW) and decrements the stack pointer word. The instruction then loads the value determined by operand n (for example, $000E_H$ when n = 1) to the program counter.									
CALLSS n	0111 nnnn	7n	STACK[SPW]←PC, SPW←SPW-1, PC←8n+6 (n≠0), 0086H (n=0)	-	-	-	2	○	○	○
	Carries out the short form subroutine call specified by the entry number n of instruction field. Refer to "CALL instruction" for operation of explanation.									
CALL a	0010 0aaa aaaa aaaa	2a aa	STACK[SPW]←PC_{12-0}, STK13[SPW]←PC_{13}, SPW←SPW-1, PC←a	-	-	-	2	○	○	○
	Calls a subroutine. The instruction saves the contents of the program counter to the stack (location: SPW) and decrements the stack pointer word. The instruction then loads immediate value a to the program counter. Note that the entry address for the subroutine must be in the range 0000_H to $07FF_H$.									
RET	0010 1010	2A	SPW←SPW+1, PC_{12-0}←STACK[SPW], PC_{13}←STK13[SPW]	-	-	-	2	○	○	○
	Returns from a subroutine. The instruction increments the stack pointer word and restores the return address from the stack (location: SPW) to the program counter.									

(11) Interrupt Control

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
XCH A, EIR	0001 0011	13	EIR↔Acc	-	-	1	1	○	○	○
	Exchanges the contents of the interrupt enable register and the accumulator. Example : When Acc = 1101B and EIR = 0001_B, executing XCH A, EIR results in Acc = 0001_B and EIR = 1101_B.									
EICLR IL, r	0011 0110 01rr rrrr	36 4+r•r	EIF←1, IL←IL∧r	-	-	1	2	○	○	○
	Sets the interrupt enable master F/F to "1" (enable interrupts). The interrupt latch ILi (i = 0 to 5) is cleared if operand r_i is "0".									
EI	0011 0110 0111 1111	36 7F	EIF←1	-	-	1	2	○	○	○
	Sets the interrupt enable master F / F to "1" (Interrupt enable).									
DICLR IL, r	0011 0110 10rr rrrr	36 8+r•r	EIF←0, IL←IL∧r	-	-	1	2	○	○	○
	Clears the interrupt enable master F/F to "0" (disable interrupts). The interrupt latch ILi (i = 0 to 5) is cleared if operand r_i is "0".									
DI	0011 0110 1011 1111	36 BF	EIF←0	-	-	1	2	○	○	○
	Clears the interrupt enable master F / F to "0" (Interrupt disable).									
CLR IL, r	0011 0110 11rr rrrr	36 C+r•r	IL←IL∧r	-	-	1	2	○	○	○
	The interrupt latch clear instruction. ILi (i = 0 to 5) is cleared if ri is "0". Example : Executing CLR IL, 101100_B instruction clears IL_4, IL_1, and IL_0. The remaining ILs do not change.									
RETI	0010 1011	2B	SPW←SPW+1, FLAG•PC_{12-0}←STACK[SPW], PC_{13}←STK13[SPW], EIF←1	*	*	*	2	○	○	○
	Returns from an interrupt service routine. The instruction increments the stack pointer word and restores the return address and flag data from the stack (location: SPW) to the program counter and flags. The instruction then sets the interrupt enable master F/F to "1" (enable interrupts).									

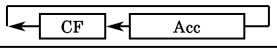
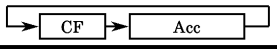
(12) STK13 Manipulation

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
CLR STK13	0000 0010 0011 1010 1000 0100	02 3A 84	SPW13←Acc, STK13[SPW13]←0	-	-	1	3	-	-	○
	Stores the contents of the accumulator in SPW13 and clears the STK13 whose location is specified by the new SPW13 contents.									
SET STK13	0000 0011 0011 1010 1000 0100	03 3A 84	SPW13←Acc, STK13[SPW13]←1	-	-	1	3	-	-	○
	Stores the contents of the accumulator in SPW13 and sets 1 in the STK13 whose location is specified by the new SPW13 contents.									
MOV A, SPW13	0011 1010 1000 0100	3A 84	SPW13←Acc	-	-	1	2	-	-	○
	Stores the contents of the accumulator in SPW13.									
MOV STK13, A	0011 1010 0000 0100	3A 04	Acc_0←STK13[SPW13] Acc_{3-1}←0	-	Z	1	2	-	-	○
	Loads the contents of the STK13 whose location is specified by SPW13 in the lower bit of the accumulator. Zero-clears the higher three bits of the accumulator.									

(13) CPU Control

Mnemonic	Object code		Operation	Flags			Cycle (s)	47	470	470A
	Binary	Hexadecimal		CF	ZF	SF				
NOP	0000 0000	00	no operation	-	-	-	1	○	○	○
	No operation (nothing is executed, execution proceeds to the next instruction).									

7.3 Instruction table

Function	Mnemonic		Object code		Operation	Flags CF	Flags ZF	Flags SF	Cycle (s)	47	470	470A
Data transfer	LD	A, @HL	0C		Acc←RAM[HL]	–	Z	1	1	○	○	○
	LD	A, x	3C	xx	Acc←RAM[x]	–	Z	1	2	○	○	○
	LD	HL, x	28	xx	LR←RAM[x], HR←RAM[x+1]	–	–	1	2	○	○	○
	LDL	A, @DC	33		Acc←ROM[DTB·DC]$_L$	–	Z	1	2	○	○	○
	LDH	A, @DC+	32		Acc←ROM[DTB·DC]$_H$, DC←DC+1	–	Z	1	2	○	○	○
	ST	A, @HL	0F		RAM[HL]←Acc	–	–	1	1	○	○	○
	ST	A, @HL+	1A		RAM[HL]←Acc, LR←LR+1	–	Z	$\overline{C}$	1	○	○	○
	ST	A, @HL–	1B		RAM[HL]←Acc, LR←LR–1	–	Z	C	1	○	○	○
	ST	A, x	3F	xx	RAM[x]←Acc	–	–	1	2	○	○	○
	ST	#k,@HL+	Fk		RAM[HL]←k, LR←LR+1	–	Z	$\overline{C}$	1	○	○	○
	ST	#k, y	2D	ky	RAM[y]←k	–	–	1	2	○	○	○
	LD	A, #k	4k		Acc←k	–	Z	1	1	○	○	○
	LD	H, #k	Ck		HR←k	–	–	1	1	○	○	○
	LD	L, #k	Ek		LR←k	–	–	1	1	○	○	○
	MOV	H, A	10		Acc←HR	–	Z	1	1	○	○	○
	MOV	L, A	11		Acc←LR	–	Z	1	1	○	○	○
Exchange	XCH	A, @HL	0D		Acc↔RAM[HL]	–	Z	1	1	○	○	○
	XCH	A, x	3D	xx	Acc↔RAM[x]	–	Z	1	2	○	○	○
	XCH	HL, x	29	xx	LR↔RAM[x], HR↔RAM[x+1]	–	–	1	2	○	○	○
	XCH	A, H	30		Acc↔HR	–	Z	1	2	○	○	○
	XCH	A, L	31		Acc↔LR	–	Z	1	2	○	○	○
Input/ Output	IN	%p, A	3A	2p	Acc←PORT[p] ($00 \leqq p \leqq 0F_H$)	–	Z	$\overline{Z}$	2	○	○	○
	IN	%p, A	3A	0p	Acc←PORT[p] ($10_H \leqq p \leqq 1F_H$)	–	Z	1	2	–	○	○
	IN	%p, @HL	3A	6p	RAM[HL]←PORT[p]($00 \leqq p \leqq 0F_H$)	–	–	$\overline{Z}$	2	○	○	○
	IN	%p, @HL	3A	4p	RAM[HL]←PORT[p]($10_H \leqq p \leqq 1F_H$)	–	–	1	2	–	○	○
	OUT	A, %p	3A	$8+2\overline{p_4}\cdot p$	PORT[p]←Acc	–	–	1	2	○	○	○
	OUT	@HL, %p	3A	$C+2\overline{p_4}\cdot p$	PORT[p]←RAM[HL]	–	–	1	2	○	○	○
	OUT	#k, %p	2C	kp	PORT[p]←k ($00 \leqq p \leqq 0F_H$)	–	–	1	2	○	○	○
	OUTB	@HL	12		PORT[2]·PORT[1]← ROM[3FE0$_H$+CF·RAM[HL]]	–	–	1	2	○	○	○
Compare	CMPR	A, @HL	16		RAM[HL] – Acc	C	Z	$\overline{Z}$	1	○	○	○
	CMPR	A, x	3E	xx	RAM[x] – Acc	C	Z	$\overline{Z}$	2	○	○	○
	CMPR	y, #k	2E	ky	k – RAM[y]	C	Z	$\overline{Z}$	2	○	○	○
	CMPR	A, #k	Dk		k – Acc	C	Z	$\overline{Z}$	1	○	○	○
	CMPR	H, #k	38	Dk	k – HR	–	Z	C	2	○	○	○
	CMPR	L, #k	38	9k	k – LR	–	Z	C	2	○	○	○
Logical operation	ADD	A, @HL	17		Acc←Acc+RAM[HL]	–	Z	$\overline{C}$	1	○	○	○
	ADDC	A, @HL	15		Acc←Acc+RAM[HL]+CF	C	Z	$\overline{C}$	1	○	○	○
	ADD	@HL, #k	38	4k	RAM[HL]←RAM[HL]+k	–	Z	$\overline{C}$	2	○	○	○
	ADD	y, #k	2F	ky	RAM[y]←RAM[y]+k	–	Z	$\overline{C}$	2	○	○	○
	ADD	A, #k	38	0k	Acc←Acc+k	–	Z	$\overline{C}$	2	○	○	○
	ADD	H, #k	38	Ck	HR←HR+k	–	Z	$\overline{C}$	2	○	○	○
	ADD	L, #k	38	8k	LR←LR+k	–	Z	$\overline{C}$	2	○	○	○
	SUBRC	A, @HL	14		Acc←RAM[HL] – Acc – $\overline{CF}$	C	Z	C	1	○	○	○
	SUBR	@HL, #k	38	5k	RAM[HL]←k – RAM[HL]	–	Z	$\overline{C}$	2	○	○	○
	SUBR	A, #k	38	1k	Acc←k – Acc	–	Z	$\overline{C}$	2	○	○	○
	INC	@HL	0A		RAM[HL]←RAM[HL]+1	–	Z	$\overline{C}$	1	○	○	○
	DEC	@HL	0B		RAM[HL]←RAM[HL]–1	–	Z	$\overline{C}$	1	○	○	○
	INC	A	08		Acc←Acc+1	–	Z	$\overline{C}$	1	○	○	○
	DEC	A	09		Acc←Acc–1	–	Z	$\overline{C}$	1	○	○	○
	INC	L	18		LR←LR+1	–	Z	$\overline{C}$	1	○	○	○
	DEC	L	19		LR←LR–1	–	Z	$\overline{C}$	1	○	○	○
	AND	A, @HL	1E		Acc←Acc∧RAM[HL]	–	Z	$\overline{Z}$	1	○	○	○
	AND	@HL, #k	38	7k	RAM[HL]←RAM[HL]∧k	–	Z	$\overline{Z}$	2	○	○	○
	AND	A, #k	38	3k	Acc←Acc∧k	–	Z	$\overline{Z}$	2	○	○	○
	OR	A, @HL	1D		Acc←Acc∨RAM[HL]	–	Z	$\overline{Z}$	1	○	○	○
	OR	@HL, #k	38	6k	RAM[HL]←RAM[HL]∨k	–	Z	$\overline{Z}$	2	○	○	○
	OR	A, #k	38	2k	Acc←Acc∨k	–	Z	$\overline{Z}$	2	○	○	○
	XOR	A, @HL	1F		Acc←Acc▽RAM[HL]	–	Z	$\overline{Z}$	1	○	○	○
Rotate	ROLC	A	05			*	Z	$\overline{C}$	1	○	○	○
	RORC	A	07			*	Z	$\overline{C}$	1	○	○	○

Bit manipulation	CLR	@HL, b	5·4+b			RAM[HL]$_b$←0	−	−	1	1	○	○	○
	CLR	y, b	39	4+b·y		RAM[y]$_b$←0	−	−	1	2	○	○	○
	CLR	%p, b	3B	4+b·p		PORT[p]$_b$←0	−	−	1	2	○	○	○
	CLR	@L	35			PORT[LR$_{3-2}$+4]$_{LR1-0}$←0	−	−	1	2	○	○	○
	SET	@HL, b	5b			RAM[HL]$_b$←1	−	−	1	1	○	○	○
	SET	y, b	39	by		RAM[y]$_b$←1	−	−	1	2	○	○	○
	SET	%p, b	3B	bp		PORT[p]$_b$←1	−	−	1	2	○	○	○
	SET	@L	34			PORT[LR$_{3-2}$+4]$_{LR1-0}$←1	−	−	1	2	○	○	○
	TEST	@HL, b	5·8+b			SF←$\overline{\text{RAM[HL]}_b}$	−	−	*	1	○	○	○
	TEST	y, b	39	8+b·y		SF←$\overline{\text{RAM[y]}_b}$	−	−	*	2	○	○	○
	TESTP	y, b	39	C+b·y		SF←RAM[y]$_b$	−	−	*	2	○	○	○
	TEST	%p, b	3B	8+b·p		SF←$\overline{\text{PORT[p]}_b}$	−	−	*	2	○	○	○
	TESTP	%p, b	3B	C+b·p		SF←PORT[p]$_b$	−	−	*	2	○	○	○
	TEST	@L	37			SF←$\overline{\text{PORT[LR}_{3-2}+4]_{LR1-0}}$	−	−	*	2	○	○	○
	TEST	A, b	5·C+b			SF←$\overline{\text{Acc}}_b$	−	−	*	1	○	○	○
Flag manipulation	TEST	CF	06			SF←$\overline{\text{CF}}$, CF←0	0	−	*	1	○	○	○
	TESTP	CF	04			SF←CF, CF←1	1	−	*	1	○	○	○
	TESTP	ZF	0E			SF←ZF	−	−	*	1	○	○	○
	CLR	GF	02			GF←0	−	−	1	1	○	−	−
	SET	GF	03			GF←1	−	−	1	1	○	−	−
	TESTP	GF	01			SF←GF	−	−	*	1	○	−	−
Data Memory Bank control	CLR	DMB	3B	79		DMB←0	−	−	1	2	−	○	−
	SET	DMB	3B	39		DMB←1	−	−	1	2	−	○	−
	TEST	DMB	3B	B9		SF←$\overline{\text{DMB}}$	−	−	*	2	−	○	−
	TESTP	DMB	3B	F9		SF←DMB	−	−	*	2	−	○	−
	CLR	DMB0	3B	79		DMB$_0$←0	−	−	1	2	−	−	○
	SET	DMB0	3B	39		DMB$_0$←1	−	−	1	2	−	−	○
	TEST	DMB0	3B	B9		SF←$\overline{\text{DMB}_0}$	−	−	*	2	−	−	○
	TESTP	DMB0	3B	F9		SF←DMB$_0$	−	−	*	2	−	−	○
	CLR	DMB1	03	3B	59	DMB$_1$←0	−	−	1	3	−	−	○
	SET	DMB1	03	3B	19	DMB$_1$←1	−	−	1	3	−	−	○
	TEST	DMB1	03	3B	99	SF←$\overline{\text{DMB}_1}$	−	−	*	3	−	−	○
	TESTP	DMB1	03	3B	D9	SF←DMB$_1$	−	−	*	3	−	−	○
	LD	DMB, #k	03	2C	k9	DMB←k	−	−	1	3	−	−	○
	LD	DMB, @HL	03	3A	E9	DMB←RAM[HL]$_{1-0}$	−	−	1	3	−	−	○
	MOV	A, DMB	03	3A	A9	DMB←Acc$_{1-0}$	−	−	1	3	−	−	○
	MOV	DMB, A	03	3A	29	Acc$_{1-0}$←DMB, Acc$_{3-2}$←0	−	Z	$\overline{\text{Z}}$	3	−	−	○
	ST	DMB, @HL	03	3A	69	RAM[HL]$_{1-0}$←DMB, RAM[HL]$_{3-2}$←0	−	−	$\overline{\text{Z}}$	3	−	−	○
Branch	BSS	a	8+d·d			if SF=1 then PC←a else null, a=PC$_{13-6}$.d	−	−	1	1	○	○	○
	BS	a	6e	ee		if SF=1 then PC←a else null, a=PC$_{13-12}$.e	−	−	1	2	○	○	○
	BSL	a (0000$_H$≦a≦0FFF$_H$)	02	6e	ee	if SF=1 then PC←a else null, a=00.e	−	−	1	3	−	○	○
		a (1000$_H$≦a≦1FFF$_H$)	03	6e	ee	if SF=1 then PC←a else null, a=01.e	−	−	1	3	−	○	○
		a (2000$_H$≦a≦2FFF$_H$)	0C	6e	ee	if SF=1 then PC←a else null, a=10.e	−	−	1	3	−	−	○
		a (3000$_H$≦a≦3FFF$_H$)	01	6e	ee	if SF=1 then PC←a else null, a=11.e	−	−	1	3	−	−	○
Sub−routine	CALLS	a	7n			STACK[SPW]←PC, SPW←SPW−1, PC←a, a=8n+6 (n≠0), 0086$_H$ (n=0)	−	−	−	2	○	○	○
	CALL	a	2a	aa		STACK[SPW]←PC, SPW←SPW−1, PC←a	−	−	−	2	○	○	○
	RET		2A			SPW←SPW+1, PC←STACK[SPW]	−	−	−	2	○	○	○
Interrupt Control	XCH	A, EIR	13			EIR↔Acc	−	−	1	1	○	○	○
	EICLR	IL, r	36	4+r·r		EIF←1, IL←IL∧r	−	−	1	2	○	○	○
	DICLR	IL, r	36	8+r·r		EIF←0, IL←IL∧r	−	−	1	2	○	○	○
	CLR	IL, r	36	C+r·r		IL←IL∧r	−	−	1	2	○	○	○
	RETI		2B			SPW←SPW+1, FLAG·PC←STACK[SPW], EIF←1	*	*	*	2	○	○	○
STK13 manipulation	CLR	STK13	02	3A	84	SPW13←Acc, STK13[SPW13]←0	−	−	1	3	−	−	○
	SET	STK13	03	3A	84	SPW13←Acc, STK13[SPW13]←1	−	−	1	3	−	−	○
	MOV	A, SPW13	3A	84		SPW13←Acc	−	−	1	2	−	−	○
	MOV	STK13, A	3A	04		Acc$_0$←STK13[SPW13], Acc$_{3-1}$←0	−	Z	1	2	−	−	○
CPU control	NOP		00			no operation	−	−	−	1	○	○	○