Appendix A: Details of Instructions

■ Instruction List

① Load

LD	PUSH	POP	LDA	LDAR

② Exchange

EX	MIRR

③ Load Increment/Decrement & Compare Increment/Decrement

LDI	LDIR	LDD	LDDR	CPI	CPIR	CPD	CPDR

④ Arithmetic operations

ADD	ADC	SUB	SBC	CP	INC	DEC	NEG
EXTZ	EXTS	DAA	PAA	MUL	MULS	DIV	DIVS
MULA	MINC	MDEC					

⑤ Logical operations

AND	OR	XOR	CPL

⑥ Bit operations

LDCF	STCF	ANDCF	ORCF	XORCF	RCF	SCF	CCF
ZCF	BIT	RES	SET	CHG	TSET	BS1	

⑦ Special operations and CPU control

NOP	EI	DI	PUSH_SR	POP_SR	SWI	HALT	LDC
LDX	LINK	UNLK	LDF	INCF	DECF	SCC	

⑧ Rotate and shift

RLC	RRC	RL	RR	SLA	SRA	SLL	SRL
RLD	RRD						

⑨ Jump, call, and return

JP	JR	JRL	CALL	CALR	DJNZ	RET	RETD
RETI							

■ Explanations of symbols used in this document

dst	Destination: destination of data transfer or operation result load.
src	Source: source of data transfer or operation data read.
num	Number: numerical value.
condition	Condition: based on flag status.
R	Eight general-purpose registers including 8/16/32-bit current bank registers. 8-bit registers : W, A, B, C, D, E, H, L (only eight registers) 16-bit registers : WA, BC, DE, HL, IX, IY, IZ, SP (only eight registers) 32-bit registers : XWA, XBC, XDE, XHL, XIX, XIY, XIZ, XSP (only eight registers)
r	8/16/32-bit general-purpose registers } (Please refer to " Register map" on page CPU900H-45, 46.)
r16	16-bit general-purpose registers
r32	32-bit general-purpose registers
cr	All 8/16/32-bit CPU control registers DMAS0 to 3, DMAD0 to 3, DMAC0 to 3, DMAM0 to 3, INTNEST
A	A register (8 bits)
F	Flag registers (8 bits)
F'	Inverse flag registers (8 bits)
SR	Status registers (16 bits)
PC	Program counter (in minimum mode, 16 bits; in maximum mode, 32 bits)
(mem)	8/16/32-bit memory data
mem	Effective address value
<W>	When the operand size is a word, W must be specified.
[ ]	Operands enclosed in square brackets can be omitted.
#	8/16/32-bit immediate data.
#3	3-bit immediate data : 0 to 7 or 1 to 8 ... for abbreviated codes.
#4	4-bit immediate data : 0 to 15 or 1 to 16
d8	8-bit displacement : − 80H to + 7FH
d16	16-bit displacement : − 8000H to + 7FFFH
cc	Condition code
CY	Carry flag
Z	Zero flag
(#8)	Direct addressing: (00H) to (0FFH) ... 256-byte area
(#16)	64K-byte area addressing: (0000H) to (0FFFFH)
(−r32)	Pre-decrement addressing
(r32+)	Post-increment addressing
$	Start address of instruction

■ Explanations of symbols in object codes

z
zz
zzz
s

Operand size specify code

	Byte	Word	Long word
z	0	1	—
zz	00	01	10
zzz	010	011	100
s	—	0	1

R
r

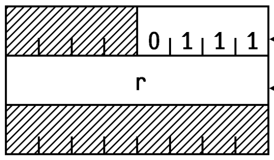
Register specify code

Code	Byte	Word	Long word
000	W	WA	XWA
001	A	BC	XBC
010	B	DE	XDE
011	C	HL	XHL
100	D	IX	XIX
101	E	IY	XIY
110	H	IZ	XIZ
111	L	SP	XSP

Note: In addition to the above, all registers can be specified by "r" using extension codes. In this case, the number of execution states increases by 1. The format is shown below.

First op code

Second op code



Sets the lower 4 bits to 0111.

Inserts the register code specified by 8 bits between the first and second op codes.

The code value in "r" must be:

Multiple of 2, if accessed as a word register.

Multiple of 4 ,if accessed as a long word.

For registers specified by 8 bits, see Register Maps.

mem

Memory addressing mode specify code

(XWA)	=	-0--0000				
(XBC)	=	-0--0001				
(XDE)	=	-0--0010				
(XHL)	=	-0--0011				
(XIX)	=	-0--0100				
(XIY)	=	-0--0101				
(XIZ)	=	-0--0110				
(XSP)	=	-0--0111				
(XWA+d8)	=	-0--1000	d<7:0>			
(XBC+d8)	=	-0--1001	d<7:0>			
(XDE+d8)	=	-0--1010	d<7:0>			
(XHL+d8)	=	-0--1011	d<7:0>			
(XIX+d8)	=	-0--1100	d<7:0>			
(XIY+d8)	=	-0--1101	d<7:0>			
(XIZ+d8)	=	-0--1110	d<7:0>			
(XSP+d8)	=	-0--1111	d<7:0>			
(#8)	=	-1--0000	#<7:0>			
(#16)	=	-1--0001	#<7:0>		#<15:8>	
(#24)	=	-1--0010	#<7:0>		#<15:8>	#<23:16>
(r32)	=	-1--0011	r32′	00		
(r32+d16)	=	-1--0011	r32′	01	d<7:0>	d<15:8>
(r32+r8)	=	-1--0011	000000	11	r32	r8
(r32+r16)	=	-1--0011	000001	11	r32	r16
(−r32)	=	-1--0100	r32′	zz		
(r32+)	=	-1--0101	r32′	zz		

<7:0>= Indicates the data bit range. This example means 8-bit data from bit 0 to bit 7.

r32: 32-bit register
r16: Signed 16-bit register
r8: Signed 8-bit register

zz= Code used to specify the value of increments or decrements.

00: ±1
01: ±2
10: ±4
11: (Not defined)

r32′ = Upper 6 bits of register code

cc	Condition codes

Code	Symbol	Description	Conditional expression
0000	F	always False	-
1000	(none)	always True	-
0110	Z	Zero	Z=1
1110	NZ	Not Zero	Z=0
0111	C	Carry	C=1
1111	NC	Not Carry	C=0
1101	PL or P	PLus	S=0
0101	MI or M	MInus	S=1
1110	NE	Not Equal	Z=0
0110	EQ	EQual	Z=1
0100	OV	OVerflow	P/V=1
1100	NOV	No OVerflow	P/V=0
0100	PE	Parity is Even	P/V=1
1100	PO	Parity is Odd	P/V=0
1001	GE	Greater than or Equal (signed)	(S xor P/V) =0
0001	LT	Less Than (signed)	(S xor P/V) =1
1010	GT	Greater Than (signed)	[Z or (S xor P/V)]=0
0010	LE	Less than or Equal (signed)	[Z or (S xor P/V)]=1
1111	UGE	Unsigned Greater than or Equal	C=0
0111	ULT	Unsigned Less Than	C=1
1011	UGT	Unsigned Greater Than	(C or Z) =0
0011	ULE	Unsigned Less than or Equal	(C or Z) =1

■ Register map "r"

	+3	+2	+1	+0	
00H	QW0 (QWA	0) QA0 <XWA	0> RW0 (RWA	0) RA0	Bank 0
04H	QB0 (QBC	0) QC0 <XBC	0> RB0 (RBC	0) RC0	
08H	QD0 (QDE	0) QE0 <XDE	0> RD0 (RDE	0) RE0	
0CH	QH0 (QHL	0) QL0 <XHL	0> RH0 (RHL	0) RL0	
10H	QW1 (QWA	1) QA1 <XWA	1> RW1 (RWA	1) RA1	Bank 1
14H	QB1 (QBC	1) QC1 <XBC	1> RB1 (RBC	1) RC1	
18H	QD1 (QDE	1) QE1 <XDE	1> RD1 (RDE	1) RE1	
1CH	QH1 (QHL	1) QL1 <XHL	1> RH1 (RHL	1) RL1	
20H	QW2 (QWA	2) QA2 <XWA	2> RW2 (RWA	2) RA2	Bank 2
24H	QB2 (QBC	2) QC2 <XBC	2> RB2 (RBC	2) RC2	
28H	QD2 (QDE	2) QE2 <XDE	2> RD2 (RDE	2) RE2	
2CH	QH2 (QHL	2) QL2 <XHL	2> RH2 (RHL	2) RL2	
30H	QW3 (QWA	3) QA3 <XWA	3> RW3 (RWA	3) RA3	Bank 3
34H	QB3 (QBC	3) QC3 <XBC	3> RB3 (RBC	3) RC3	
38H	QD3 (QDE	3) QE3 <XDE	3> RD3 (RDE	3) RE3	
3CH	QH3 (QHL	3) QL3 <XHL	3> RH3 (RHL	3) RL3	
40H					
44H					
48H					
4CH					
50H					
54H					
58H					
5CH					
60H					
64H					
68H					
6CH					
70H					
74H					
78H					
7CH					
D0H	QW' (Q	WA') QA' <X	WA'> W' (W	A') A'	Previous bank
D4H	QB' (Q	BC') QC' <X	BC'> B' (B	C') C'	
D8H	QD' (Q	DE') QE' <X	DE'> D' (D	E') E'	
DCH	QH' (Q	HL') QL' <X	HL'> H' (H	L') L'	
E0H	QW (Q	WA) QA <X	WA > W (W	A) A	Current bank
E4H	QB (Q	BC) QC <X	BC > B (B	C) C	
E8H	QD (Q	DE) QE <X	DE > D (D	E) E	
ECH	QH (Q	HL) QL <X	HL > H (H	L) L	
F0H	QIXH (Q	IX) QIXL <X	IX> IXH (I	X) IXL	
F4H	QIYH (Q	IY) QIYL <X	IY> IYH (I	Y) IYL	
F8H	QIZH (Q	IZ) QIZL <X	IZ> IZH (I	Z) IZL	
FCH	QSPH (Q	SP) QSPL <X	SP> SPH (S	P) SPL	

( ) : Word register name (16 bits)
< > : Long word register name (32 bits)

■ Control register map "cr"

	+3	+2	+1	+0	
00H		<DMA	S0>		DMA source register
04H		<DMA	S1>		
08H		<DMA	S2>		
0CH		<DMA	S3>		
10H		<DMA	D0>		DMA destination register
14H		<DMA	D1>		
18H		<DMA	D2>		
1CH		<DMA	D3>		
20H		DMAM0	(DMA	C0)	DMA mode/counter register
24H		DMAM1	(DMA	C1)	
28H		DMAM2	(DMA	C2)	
2CH		DMAM3	(DMA	C3)	
3CH			(INT	NEST)	→ Interrupt Nesting Counter

( ) : Word register name (16 bits)
< > : Long word register name (32 bits)

ADC dst, src

< Add with Carry >

Operation : dst ← dst + src + CY

Description : Adds the contents of dst, src, and carry flag, and transfers the result to dst.

Details :

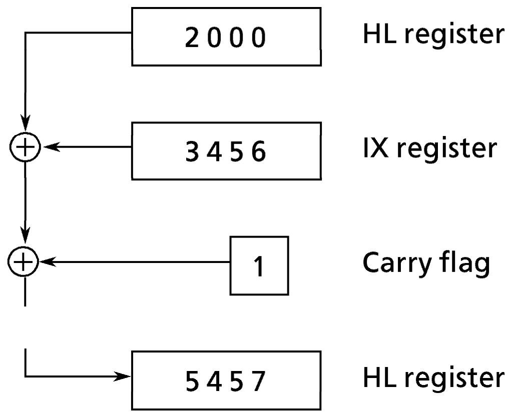
Size			Mnemonic		Code
Byte	Word	Long word			
○	○	○	ADC	R, r	1 1 z z 1 r
					1 0 0 1 0 R
○	○	○	ADC	r, #	1 1 z z 1 r
					1 1 0 0 1 0 0 1
					#<7:0>
					#<15:8>
					#<23:16>
					#<31:24>
○	○	○	ADC	R, (mem)	1 m z z m m m m
					1 0 0 1 0 R
○	○	○	ADC	(mem), R	1 m z z m m m m
					1 0 0 1 1 R
○	○	×	ADC<W>	(mem), #	1 m 0 z m m m m
					0 0 1 1 1 0 0 1
					#<7:0>
					#<15:8>

Flags:

S	Z	H	V	N	C
*	*	*	*	0	*

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a carry from bit 3 to bit 4 occurs as a result of the operation; otherwise, 0. If the operand is 32-bit, an undefined value is set.
V = 1 is set if an overflow occurs as a result of the operation; otherwise, 0.
N = Cleared to zero.
C = 1 is set if a carry occurs from the MSB, otherwise 0.

Execution example: ADC HL,IX
When the HL register = 2000H, the IX register = 3456H, and the carry flag = 1, execution sets the HL register to 5457H.



ADD dst, src

< Add >

Operation : dst ← dst + src

Description : Adds the contents of dst to those of src and transfers the result to dst.

Details :

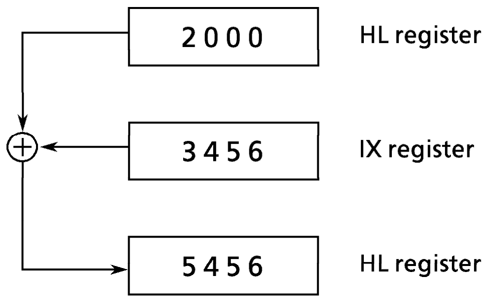
Size			Mnemonic		Code
Byte	Word	Long word			
○	○	○	ADD	R, r	1 1 z z 1 r 1 0 0 0 0 R
○	○	○	ADD	r, #	1 1 z z 1 r 1 1 0 0 1 0 0 0 #<7:0> #<15:8> #<23:16> #<31:24>
○	○	○	ADD	R, (mem)	1 m z z m m m m 1 0 0 0 0 R
○	○	○	ADD	(mem), R	1 m z z m m m m 1 0 0 0 1 R
○	○	×	ADD<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 0 0 0 #<7:0> #<15:8>

Flags :

S	Z	H	V	N	C
*	*	*	*	0	*

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a carry from bit 3 to bit 4 occurs as a result of the operation, otherwise 0. If the operand is 32-bit, an undefined value is set.
V = 1 is set if an overflow occurs as a result of the operation, otherwise 0.
N = Cleared to zero.
C = 1 is set if a carry occurs from the MSB, otherwise 0.

Execution example: ADD HL,IX
When the HL register = 2000H and the IX register = 3456H, execution sets the HL register to 5456H.



AND dst, src

< And >

Operation : dst ← dst AND src

Description : Ands the contents of dst and src, then transfers the result to dst.

(Truth table)

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
○	○	○	AND	R, r	1 1 z z 1 r 1 1 0 0 0 R
○	○	○	AND	r, #	1 1 z z 1 r 1 1 0 0 1 1 0 0 #<7:0> #<15:8> #<23:16> #<31:24>
○	○	○	AND	R, (mem)	1 m z z m m m m 1 1 0 0 0 R
○	○	○	AND	(mem), R	1 m z z m m m m 1 1 0 0 1 R
○	○	×	AND<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 1 0 0 #<7:0> #<15:8>

Flags :

S	Z	H	V	N	C
*	*	1	*	0	0

S	=	MSB value of the result is set.
Z	=	1 is set if the result is 0, otherwise 0.
H	=	1 is set.
V	=	1 is set if a parity of the result is even, 0 if odd. If the operand is 32 bits, an undefined value is set.
N	=	Cleared to zero.
C	=	Cleared to zero.

Execution example: AND HL,IX

When the HL register = 7350H and the IX register = 3456H, execution sets the HL register to 3050H.

```
      0111  0011  0101  0000  ←  HL register (before execution)
AND)  0011  0100  0101  0110  ←  IX register (before execution)
      ----------------------
      0011  0000  0101  0000  ←  HL register (after execution)
```

ANDCF num, src

< And Carry Flag >

Operation : CY ← CY and src<num>

Description : Ands the contents of the carry flag and bit num of src, and transfers the result to the carry flag.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
○	○	×	ANDCF	#4, r	1 1 0 z 1 r 0 0 1 0 0 0 0 0 0 0 0 0 # 4
○	○	×	ANDCF	A, r	1 1 0 z 1 r 0 0 1 0 1 0 0 0
○	×	×	ANDCF	#3, (mem)	1 m 1 1 m m m m 1 0 0 0 0 #3
○	×	×	ANDCF	A, (mem)	1 m 1 1 m m m m 0 0 1 0 1 0 0 0

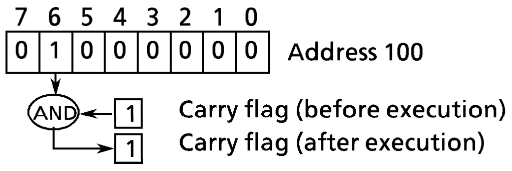
Notes : When bit num is specified by the A register, the value of the lower 4 bits of the A register is used as bit num. When the operand is a byte and the value of the lower 4 bits of bit num is from 8 to 15, the result is undefined.

Flags :

S	Z	H	V	N	C
–	–	–	–	–	*

S = No change
Z = No change
H = No change
V = No change
N = No change
C = The value obtained by anding the contents of the carry flag and the bit num of src is set.

Execution example: ANDCF 6,(100H)
When the contents of memory address 100 = 01000000B (binary) and the carry flag = 1, execution sets the carry flag to 1.



BIT num, src

< Bit test >

Operation : Z flag ← inverted value of src<num>

Description : Transfers the inverted value of the bit num of src to the Z flag.

Details :

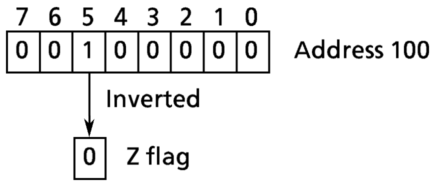
Size			Mnemonic		Code
Byte	Word	Long word			
○	○	×	BIT	#4, r	1 1 0 z 1 r
					0 0 1 1 0 0 1 1
					0 0 0 0 # 4
○	×	×	BIT	#3, (mem)	1 m 1 1 m m m m
					1 1 0 0 1 #3

Flags :

S	Z	H	V	N	C
×	*	1	×	0	–

S = An undefined value is set.
Z = The inverted value of src<num> is set.
H = 1 is set.
V = An undefined value is set.
N = Reset to 0.
C = No change

Execution example: BIT 5,(100H)
When the contents of memory address 100 = 00100000B (binary), execution sets the Z flag to 0.



BS1B dst, src

< Bit Search 1 Backward >

Operation : dst ← src backward searched value

Description : Searches the src bit pattern backward (from MSB to LSB) for the first bit set to 1 and transfers the bit number to dst.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
×	○	×	BS1B	A, r	1 1 0 1 1 r
					0 0 0 0 1 1 1 1

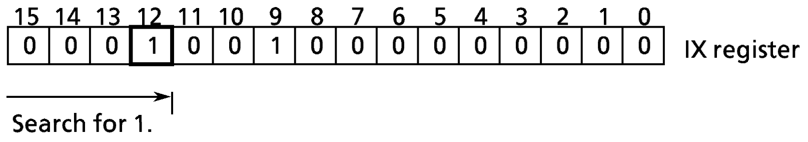
Note : dst in the operand must be the A register; src must be the register in words. If no bit set to 1 is found in the searched bit pattern, sets the A register to an undefined value and the V flag to 1.

Flags :

S	Z	H	V	N	C
–	–	–	*	–	–

S = No change
Z = No change
H = No change
V = 1 is set if the contents of src are all 0s (no bit is set to 1), otherwise 0.
N = No change
C = No change

Execution example: BS1B A,IX
When the IX register = 1200H, execution sets the A register to 0CH.



BS1F dst, src

< Bit Search 1 Forward >

Operation : dst ← src forward searched result

Description : Searches the src bit pattern forward (from LSB to MSB) for the first bit set to 1 and transfers the bit number to dst.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
×	○	×	BS1F	A, r	1 1 0 1 1 r 0 0 0 0 1 1 1 0

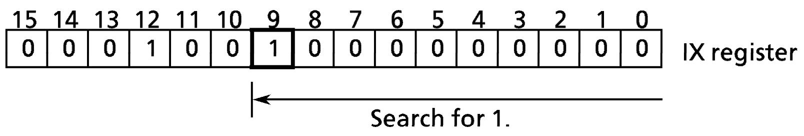
Note : dst in the operand must be the A register; src must be a register in words.
If no bit set to 1 is found in the searched bit pattern, sets the A register to an undefined value and the V flag to 1.

Flags :

S	Z	H	V	N	C
–	–	–	*	–	–

S = No change
Z = No change
H = No change
V = 1 is set if the contents of src are all 0s (no bit is set to 1), otherwise 0.
N = No change
C = No change

Execution example: BS1F A,IX
When the IX register = 1200H, execution sets the A register to 09H.



15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
0	0	0	1	0	0	$\boxed{1}$	0	0	0	0	0	0	0	0	0	IX register

CALL condition, dst

< Call subroutine >

Operation : If cc is true, then XSP ← XSP − 4,(XSP) ← 32-bit PC,PC ← dst.

Description : If the operand condition is true, saves the contents of the program counter to the stack area and jumps to the program address specified by dst.

Details :

Mnemonic	Code
CALL #16	0 0 0 1 1 1 0 0
	#<7:0>
	#<15:8>
CALL #24	0 0 0 1 1 1 0 1
	#<7:0>
	#<15:8>
	#<23:16>
CALL [cc,] mem	1 m 1 1 m m m m
	1 1 1 0 c c

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: CALL 9000H
When the stack pointer XSP is 100H, executing this instruction at memory address 8000H writes the return address 8003H (long word data) to memory address 0FCH, sets the stack pointer XSP to 0FCH, and jumps to address 9000H.

CALR dst

< Call Relative >

Operation : XSP ← XSP − 4,(XSP) ← 32-bit PC,PC ← dst.

Description : Saves the contents of the program counter to the stack area and makes a relative jump to the program address specified by dst.

Details :

Mnemonic	Code
CALR $+3+d16	0 0 0 1 1 1 1 0
	d<7:0>
	d<15:8>

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S	=	No change
Z	=	No change
H	=	No change
V	=	No change
N	=	No change
C	=	No change

CCF

< Complement Carry Flag >

Operation : CY ← inverted value of CY

Description : Inverts the contents of the carry flag.

Details :

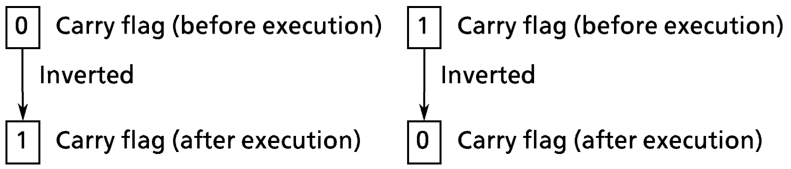
Mnemonic	Code
CCF	0 0 0 1 0 0 1 0

Flags :

S	Z	H	V	N	C
–	–	×	–	0	*

S = No change
Z = No change
H = An undefined value is set.
V = No change
N = Reset to 0.
C = Inverted value of itself is set.

Execution example: When the carry flag = 0, executing CCF sets the carry flag to 1; executing CCF again sets the carry flag to 0.


0 Carry flag (before execution)
Inverted
1 Carry flag (after execution)
1 Carry flag (before execution)
Inverted
0 Carry flag (after execution)

CHG num, dst

< Change >

Operation : dst<num> ← Inverted value of dst<num>

Description : Inverts the value of bit num of dst.

Details :

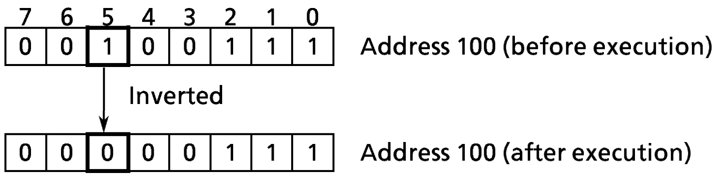
Size			Mnemonic		Code
Byte	Word	Long word			
○	○	×	CHG	#4, r	1 1 0 z 1 r
					0 0 1 1 0 0 1 0
					0 0 0 0 #4
○	×	×	CHG	#3, (mem)	1 m 1 1 m m m m
					1 1 0 0 0 #3

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: CHG 5,(100H)
When the contents of memory address 100 = 00100111B (binary), execution sets the contents to 00000111B (binary).



CP src1, src2

< Compare >

Operation : src1 − src2

Description : Compares the contents of src1 with those of src2 and indicates the results in flag register F.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
○	○	○	CP	R, r	1 1 z z 1 r 1 1 1 1 0 R
○	○	×	CP	r, #3	1 1 0 z 1 r 1 1 0 1 1 #3
○	○	○	CP	r, #	1 1 z z 1 r 1 1 0 0 1 1 1 1 #<7:0> #<15:8> #<23:16> #<31:24>
○	○	○	CP	R, (mem)	1 m z z m m m m 1 1 1 1 0 R
○	○	○	CP	(mem), R	1 m z z m m m m 1 1 1 1 1 R
○	○	×	CP<W>	(mem), #	1 m 0 z m m m m 0 0 1 1 1 1 1 1 #<7:0> #<15:8>

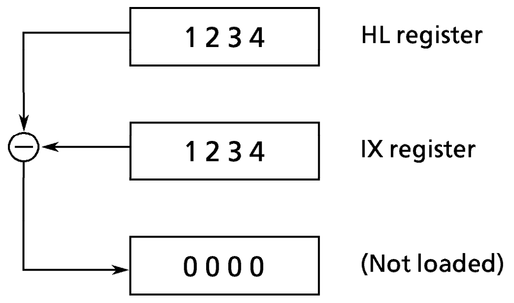
Note : #3 in operands indicates from 0 to 7.

Flags :

S	Z	H	V	N	C
*	*	*	*	1	*

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of the operation, otherwise 0. If the operand is 32 bits, an undefined value is set.
V = 1 is set if an overflow occurs as a result of the operation, otherwise 0.
N = 1 is set.
C = 1 is set if a borrow occurs from the MSB bit as a result of the operation, otherwise 0.

Execution example: CP HL,IX
When the HL register = 1234H and the IX register = 1234H, execution sets the Z and N flags to 1 and clears the S, H, V, and C flags to zero.



CPD src1, src2

< Compare Decrement >

Operation : src1 − src2, BC ← BC − 1

Description : Compares the contents of src1 with those of src2, then decrements the contents of the BC register by 1. src1 must be the A or WA register. src2 must be in post-decrement register indirect addressing mode.

Details :

Size: Byte	Size: Word	Size: Long word	Mnemonic		Code
○	○	×	CPD	[A/WA, (R−)]	1 0 0 z 0 R
					0 0 0 1 0 1 1 0

Note : Omitting operands in square brackets [] specifies A,(XHL−).

Flags :

S	Z	H	V	N	C
*	*	*	*	1	–

S = MSB value of the result of src1-src2 is set.
Z = 1 is set if the result of src1-src2 is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of src1-src2, otherwise 0.
V = 0 is set if the BC register value is 0 after execution, otherwise 1.
N = 1 is set.
C = No change

Execution example: CPD A,(XIX−)
When the XIX register = 00123456H and the BC register = 0200H, execution compares the contents of the A register with those of memory address 123456H, then sets the XIX register to 00123455H, the BC register to 01FFH.

CPDR src1, src2

< Compare Decrement Repeat >

Operation : src1 − src2, BC ← BC − 1, Repeat until src1 = src2 or BC = 0

Description : Compares the contents of src1 with those of src2. Then decrements the contents of the BC register by 1. Repeats until src1 = src2 or BC = 0. src1 must be the A or WA register. src2 must be in post-decrement register indirect addressing mode.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
○	○	×	CPDR	[A/WA, (R−)]	1 0 0 z 0 R 0 0 0 1 0 1 1 1

Note : Omitting operands in square brackets [] specifies A,(XHL−).

Flags :

S	Z	H	V	N	C
*	*	*	*	1	–

S = MSB value of the result of src1 - src2 is set.
Z = 1 is set if the result of src1 - src2 is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of src1 - src2, otherwise 0.
V = 0 is set if the BC register value is 0 after execution, otherwise 1.
N = 1 is set.
C = No change

Execution example: CPDR A,(XIX−)

Under the following conditions, execution reads the contents of memory addresses 123456H, 123455H, and 123454H. The instruction ends with condition BC = 0 and sets the XIX register to 00123453H and the BC register to 0000H.

Conditions : A register = 55H
XIX register = 00123456H
BC register = 0003H
Memory address 123456H = 11H
Memory address 123455H = 22H
Memory address 123454H = 33H

CPI src1, src2

< Compare Increment >

Operation : src1-src2, BC ← BC − 1

Description : Compares the contents of src1 with those of src2, then decrements the contents of the BC register by 1. src1 must be the A or WA register. src2 must be in post-increment register indirect addressing mode.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
○	○	×	CPI	[A/WA, (R+)]	1 0 0 z 0 R 0 0 0 1 0 1 0 0

Note : Omitting operands enclosed in square brackets [] specifies A,(XHL+).

Flags :

S	Z	H	V	N	C
*	*	*	*	1	–

S = MSB value of the result of src1-src2 is set.
Z = 1 is set if the result of src1-src2 is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of src1-src2, otherwise 0.
V = 0 is set if the BC register value is 0 after execution, otherwise 1.
N = 1 is set.
C = No change

Execution example: CPI A, (XIX+)
When the XIX register = 00123456H and the BC register = 0200H, execution compares the contents of the A register with those of memory address 123456H, and sets the XIX register to 00123457H and the BC register to 01FFH.

CPIR src1, src2

< Compare Increment Repeat >

Operation : src1-src2, BC ← BC − 1, repeat until src1 = src2 or BC = 0

Description : Compares the contents of src1 with those of src2. Then decrements the contents of the BC register by 1. Repeats until src1 = src2 or BC = 0. src1 must be the A or WA register. src2 must be in post-increment register indirect addressing mode.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
○	○	×	CPIR	[A/WA, (R+)]	1 0 0 z 0 R
					0 0 0 1 0 1 0 1

Note : Omitting operands in square brackets [ ] specifies A,(XHL+).

Flags :

S	Z	H	V	N	C
*	*	*	*	1	–

S = MSB value of the result of src1-src2 is set.
Z = 1 is set if the result of src1-src2 is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of src1-src2, otherwise 0.
V = 0 is set if the BC register value is 0 after execution, otherwise 1.
N = 1 is set.
C = No change

Execution example: CPIR A,(XIX+)

Under the following conditions, execution reads memory addresses 123456H, 123457H, and 123458H. The instruction ends with condition src1 = src2, sets the XIX register to 00123459H and the BC register to 01FDH.

Conditions : A register = 33H
XIX register = 00123456H
BC register = 0200H
Memory address 123456H = 11H
Memory address 123457H = 22H
Memory address 123458H = 33H

CPL dst

< Complement >

Operation : dst ← Ones complement of dst

Description : Transfers the value of ones complement (inverted bit of 0/1) of dst to dst.

Details :

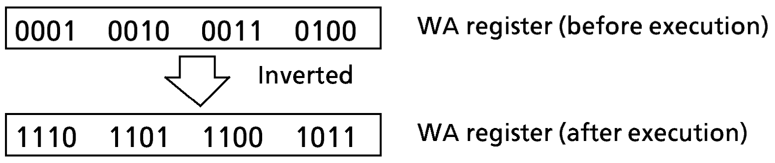
Size			Mnemonic		Code
Byte	Word	Long word			
○	○	×	CPL	r	1 1 0 z 1 r 0 0 0 0 0 1 1 0

Flags :

S	Z	H	V	N	C
–	–	1	–	1	–

S = No change
Z = No change
H = 1 is set.
V = No change
N = 1 is set.
C = No change

Execution example: CPL WA
When the WA register = 1234H, execution sets the WA register to EDCBH.



DAA dst

< Decimal Adjust Accumulator >

Operation : dst ← decimal adjustment of dst

Description : Decimal adjusts the contents of dst depending on the states of the C, H, and N flags. Used to adjust the execution result of the add or subtract instruction as binary-coded decimal (BCD).

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
○	×	×	DAA	r	1 1 0 0 1 r 0 0 0 1 0 0 0 0

Opera - tion	N flag before DAA instruction execution	C flag before DAA instruction execution	Upper 4 bits of dst	H flag before DAA instruction execution	Lower 4 bits of dst	Added value	C flag after DAA instruction execution
	0	0	0 to 9	0	0 to 9	00	0
	0	0	0 to 8	0	A to F	06	0
ADD	0	0	0 to 9	1	0 to 3	06	0
	0	0	A to F	0	0 to 9	60	1
ADC	0	0	9 to F	0	A to F	66	1
	0	0	A to F	1	0 to 3	66	1
	0	1	0 to 2	0	0 to 9	60	1
	0	1	0 to 2	0	A to F	66	1
	0	1	0 to 3	1	0 to 3	66	1
SUB	1	0	0 to 9	0	0 to 9	00	0
SBC	1	0	0 to 8	1	6 to F	FA	0
NEG	1	1	7 to F	0	0 to 9	A0	1
	1	1	6 to F	1	6 to F	9A	1

Note : Decimal adjustment cannot be performed for the INC or DEC instruction. This is because the C flag does not change.

Flags :

S	Z	H	V	N	C
*	*	*	*	–	*

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a carry from bit 3 to bit 4 occurs as a result of the operation, otherwise 0.
V = 1 is set if the parity (number of 1s) of the result is even, otherwise 0.
N = No change
C = 1 is set if a carry occurs from the MSB as a result of the operation or a carry was 1 before operation, otherwise 0.

Execution example:
```
ADD   A,B
DAA   A
```
When the A register = 59H and the B register = 13 H,
execution sets the A register to 72H.

DEC num, dst

< Decrement >

Operation : dst ← dst − num

Description : Decrements dst by the contents of num and transfers the result to dst.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
○	○	○	DEC	#3, r	1 1 z z 1 r / 0 1 1 0 1 #3
○	○	×	DEC<W>	#3, (mem)	1 m 0 z m m m m / 0 1 1 0 1 #3

Note : #3 in operands indicates from 1 to 8; object codes correspond from 1 to 7,0.

Flags :

S	Z	H	V	N	C
*	*	*	*	1	–

S = MSB value of the result is set.
Z = 1 is set if the result is 0, otherwise 0.
H = 1 is set if a borrow from bit 3 to bit 4 occurs as a result of the operation, otherwise 0.
V = 1 is set if an overflow occurs as a result of the operation, otherwise 0.
N = 1 is set.
C = No change

Note : With the DEC #3, r instruction, if the operand is a word or a long word, no flags change.

Execution example: DEC 4, HL
When the HL register = 5678H, execution sets the HL register to 5674H.

DECF

< Decrement Register File Pointer >

Operation : RFP<2:0> ← RFP<2:0> − 1

Description : Decrements the contents of register file pointer RFP <2:0> in the status register by 1. RFP2 is fixed to 0.

Details :

Mnemonic	Code
DECF	0 0 0 0 1 1 0 1

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S	=	No change
Z	=	No change
H	=	No change
V	=	No change
N	=	No change
C	=	No change

Execution example: DECF

When the contents of RFP<2:0> = 2, execution sets the contents of RFP<2:0> to 1.

DI

<Disable Interrupt>

Operation : IFF<2:0> ← 7

Description : Sets the contents of the interrupt enable flag (IFF) <2:0> in status register to 7. After execution, only non-maskable interrupts (interrupt level 7) can be received.

Details :

Mnemonic	Code
DI	0 0 0 0 0 1 1 0
	0 0 0 0 0 1 1 1

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

DIV dst, src

< Divide >

Operation : dst<lower half> ← dst ÷ src,dst<upper half> ← remainder (unsigned)

Description : Divides unsigned the contents of dst by those of src and transfers the quotient to the lower half of dst, the remainder to the upper half of dst.

Details :

Size: Byte	Size: Word	Size: Long word	Mnemonic		Code
○	○	×	DIV	RR, r	1 1 0 z 1 r
					0 1 0 1 0 R
○	○	×	DIV	rr, #	1 1 0 z 1 r
					0 0 0 0 1 0 1 0
					#<7:0>
					#<15:8>
○	○	×	DIV	RR, (mem)	1 m 0 z m m m m
					0 1 0 1 0 R

*For RR, see the following page.

Notes : When the operation is in bytes, dst (lower byte) ← dst (word) ÷ src (byte), dst (upper byte) ← remainder.
When the operation is in words, dst (lower word) ← dst (long word) ÷ src (word), dst (upper word) ← remainder. Match coding of the operand dst with the size of the <u>dividend</u>.

Flags :

S	Z	H	V	N	C
–	–	–	*	–	–

S = No change
Z = No change
H = No change
V = 1 is set when divided by 0 or the quotient exceeds the numerals which can be expressed in bits of dst for load; otherwise, 0 is set.
N = No change
C = No change

Execution example: DIV XIX,IY
When the XIX register = 12345678H and the IY register = 89ABH, execution results in a quotient of 21DAH and a remainder of 0FDAH, and sets the XIX register to 0FDA21DAH.

Note : "RR" of the DIV RR,r and DIV RR,(mem) instruction is as listed below.

Operation size in bytes
(8 bits ← 16 bits ÷ 8 bits)

RR	Code "R"
WA	001
BC	011
DE	101
HL	111
IX	Specification not possible!
IY	Specification not possible!
IZ	Specification not possible!
SP	Specification not possible!

Operation size in words
(16 bits ← 32 bits ÷ 16 bits)

RR	Code "R"
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

"rr" of the DIV rr,# instruction is as listed below.

Operation size in bytes
(8 bits ← 16 bits ÷ 8 bits)

rr	Code "r"
WA	001
BC	011
DE	101
HL	111
IX	C7H : F0H
IY	C7H : F4H
IZ	C7H : F8H
SP	C7H : FCH
	1st byte 2nd byte

Note: Any other word registers can be specified in the same extension coding as IX to SP.

Operation size in words
(16 bits ← 32 bits ÷ 16 bits)

rr	Code "r"
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

Note: Any other long word registers can be specified in the extension coding.

DIVS dst, src

< Divide Signed >

Operation : dst<lower half> ← dst ÷ src,dst<upper half> ← remainder (signed)

Description : Divides signed the contents of dst by those of src and transfers the quotient to the lower half of dst, the remainder to the upper half of dst.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
○	○	×	DIVS	RR, r	1 1 0 z 1 r / 0 1 0 1 1 R
○	○	×	DIVS	rr, #	1 1 0 z 1 r / 0 0 0 0 1 0 1 1 / #<7:0> / #<15:8>
○	○	×	DIVS	RR, (mem)	1 m 0 z m m m m / 0 1 0 1 1 R

* For RR, see the following page.

Notes : When the operation is in bytes, dst (lower byte) ← dst (word) ÷ src (byte), dst (upper byte) ← remainder.
When the operation is in words,dst (lower word) ← dst (long word) ÷ src (word), dst (upper word) ← remainder.
Match coding of the operand dst with the size of the dividend. The sign of the remainder is the same as that of the dividend.

Flags :

S	Z	H	V	N	C
–	–	–	*	–	–

S = No change
Z = No change
H = No change
V = 1 is set when divided by 0, or the quotient exceeds the value which can be expressed in bits of the dst used for loading, otherwise 0.
N = No change
C = No change

Execution example: DIVS XIX,IY
When the XIX register = 12345678H and the IY register = 89ABH, execution results in the quotient as 16EEH and the remainder as D89EH, and sets the XIX register to 16EED89EH.

Note : "RR" of the DIVS RR,r and DIVS RR,(mem) instruction is as listed below.

Operation size in bytes
(8 bits ← 16 bits ÷ 8 bits)

RR	Code "R"
WA	001
BC	011
DE	101
HL	111
IX	Specification not possible!
IY	
IZ	
SP	

Operation size in words
(16 bits ← 32 bits ÷ 16 bits)

RR	Code "R"
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

"rr" of the DIVS rr,# instruction is as listed below.

Operation size in bytes
(8 bits ← 16 bits ÷ 8 bits)

rr	Code "r"
WA	001
BC	011
DE	101
HL	111
IX	C7H : F0H
IY	C7H : F4H
IZ	C7H : F8H
SP	C7H : FCH
	1st byte 2nd byte

Note: Any other word registers can be specified in the same extension coding as those for IX to SP.

Operation size in words
(16 bits ← 32 bits ÷ 16 bits)

rr	Code "r"
XWA	000
XBC	001
XDE	010
XHL	011
XIX	100
XIY	101
XIZ	110
XSP	111

Note: Any other long word registers can be specified in the extension coding.

DJNZ dst1, dst2

< Decrement and Jump if Non Zero >

Operation : dst1 ← dst1 − 1. if dst1 ≠ 0, then PC ← dst2.

Description : Decrements the contents of dst1 by 1. Makes a relative jump to the program address specified by dst2 if the result is other than 0.

Details: :

Size			Mnemonic		Code
Byte	Word	Long word			
○	○	×	DJNZ	[r,] $+3/4+d8	1 1 0 z 1 r
					0 0 0 1 1 1 0 0
					d<7:0>

(Note) $+4+d8 ("r" is specified using extension codes.)
$+3+d8 (otherwise)

Note : Omitting "r" of the operand in square brackets [ ] is regarded as specifying the B register.

Flags :

S	Z	H	V	N	C
−	−	−	−	−	−

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example:

```
LOOP: ADD    A, A
      DJNZ   W, LOOP
```

When the A register = 12H and the W register = 03H, execution loops three times and sets the A register to 24H→48→90H and the W register to 02H → 01H → 00H.

EI num

<Enable Interrupt>

Operation : IFF <2:0> ← num

Description : Sets the contents of the IFF<2:0> in the status register to num. After execution, the CPU interrupt receive level becomes num.

Details :

Mnemonic	Code
EI [#3]	0 0 0 0 0 1 1 0
	0 0 0 0 0 #3

Note : A value from 0 to 7 can be specified as the operand value. If the operand is omitted, the default value is "0" (EI 0).

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

EX dst, src

< Exchange >

Operation : dst ↔ src

Description : Exchanges the contents of dst and src.

Details :

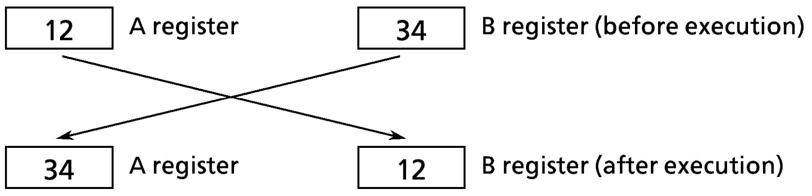
Size			Mnemonic		Code
Byte	Word	Long word			
○	×	×	EX	F, F’	0 0 0 1 0 1 1 0
○	○	×	EX	R, r	1 1 z z 1 r 1 0 1 1 1 R
○	○	×	EX	(mem), r	1 m z z m m m m 0 0 1 1 0 R

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change
* Executing EX F,F’ changes all flags.

Execution example: EX A,B
When the A register = 12H and the B register = 34H, execution sets the A register to 34H and the B register to 12H.



EXTS dst

< Extend Sign >

Operation : dst <upper half> ← signed bit of dst <lower half>

Description : Transfers (copies) the signed bit (bit 7 when the operand size is a word, bit 15 when a long word) of the lower half of dst to all bits of the upper half of dst.

Details :

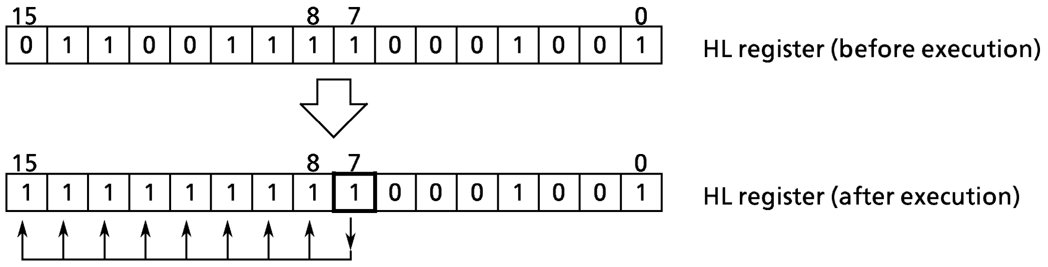
Size: Byte	Word	Long word	Mnemonic		Code
×	○	○	EXTS	r	1 1 z z 1 r / 0 0 0 1 0 0 1 1

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: EXTS HL
When the HL register = 6789H, execution sets the HL register to FF89H.



EXTZ dst

< Extend Zero >

Operation : dst<upper half> ← 0

Description : Clears the upper half of dst to zero. Used for making the operand sizes the same when they are different.

Details :

Size			Mnemonic		Code
Byte	Word	Long word			
×	○	○	EXTZ	r	1 1 z z 1 r / 0 0 0 1 0 0 1 0

Flags :

S	Z	H	V	N	C
–	–	–	–	–	–

S = No change
Z = No change
H = No change
V = No change
N = No change
C = No change

Execution example: EXTZ HL
When the HL register = 6789H, execution sets the HL register to 0089H.

EXTZ XIX
When the XIX register = 12345678H, execution sets the XIX register to 00005678H.