Zilog®

Z80®-Assembly Language Programming Manual

Z80 ASSEMBLY LANGUAGE PROGRAMMING MANUAL

Copyright© 1977 by Zilog, Inc. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Zilog.

Zilog assumes no responsibility for the use of any circuitry other than circuitry embodied in a Zilog product. No other circuit patent licenses are implied.

3.0 D.S.
REL. 2.1

TABLE OF CONTENTS

PAGE

I. INTRODUCTION..... 1

II. SPECIFICATION OF Z80 ASSEMBLY LANGUAGE
- A. THE ASSEMBLY LANGUAGE..... 2
- B. OPERANDS..... 4
- C. RULES FOR WRITING ASSEMBLY STATEMENTS (SYNTAX).. 6
- D. ASSEMBLY LANGUAGE CONVENTIONS..... 7
- E. ASSEMBLER COMMANDS..... 13

III. MACROS..... 15

IV. SUBROUTINES..... 18

V. Z80 CPU FLAGS..... 20

VI. Z80 INSTRUCTION SET..... 24

INSTRUCTION INDEX..... 275

APPENDIX:

- A. ERROR MESSAGES..... 280
- C. INSTRUCTION SORT LISTING (ALPHABETICAL)..... 284
- C. INSTRUCTION SORT LISTING (NUMERICAL)..... 290

Z80 ASSEMBLY LANGUAGE PROGRAMMING MANUAL

INTRODUCTION:

The assembly language provides a means for writing a program without having to be concerned with actual memory addresses or machine instruction formats. It allows the use of symbolic addresses to identify memory locations and mnemonic codes (opcodes and operands) to represent the instructions themselves. Labels (symbols) can be assigned to a particular instruction step in a source program to identify that step as an entry point for use in subsequent instructions. Operands following each instruction represent storage locations, registers, or constant values. The assembly language also includes assembler directives that supplement the machine instruction. A pseudo-op, for example, is a statement which is not translated into a machine instruction, but rather is interpreted as a directive that controls the assembly process.

A program written in assembly language is called a source program. It consists of symbolic commands called statements. Each statement is written on a single line and may consist of from one to four entries: A label field, an operation field, an operand field and a comment field. The source program is processed by the assembler to obtain a machine language program (object program) that can be executed directly by the Z80-CPU.

Zilog provides several different assemblers which differ in the features offered. Both absolute and relocatable assemblers are available with the Development and Microcomputer Systems. The absolute assembler is contained in base level software operating in a 16K memory space while the relocating assembler is part of the RIO environment operating in a 32K memory space.

II SPECIFICATION OF THE Z80 ASSEMBLY LANGUAGE

A. THE ASSEMBLY LANGUAGE

The assembly language of the Z80 is designed to minimize the number of different opcodes corresponding to the set of basic machine operations and to provide for a consistent description of instruction operands. The nomenclature has been defined with special emphasis on mnemonic value and readability.

The movement of data is indicated primarily by a single opcode, LD for example, regardless of whether the movement is between different registers or between registers and memory locations.

The first operand of an LD instruction is the destination of the operation, and the second operand is the source of the operation. For example:

```
LD A,B
```

indicates that the contents of the second operand, register B, are to be transferred to the first operand, register A. Similarly,

```
LD C,3FH
```

indicates that the constant 3FH is to be loaded into the register C. In addition, enclosing an operand wholly in parentheses indicates a memory location addressed by the contents of the parentheses. For example,

```
LD HL,(1200)
```

indicates the contents of memory locations 1200 and 1201 are to be loaded into the 16-bit register pair HL. Similarly,

```
LD (IX+6),C
```

indicates the contents of the register C are to be stored in the memory location addressed by the current value of the 16-bit index register IX plus 6.

The regular formation of assembly instructions minimizes the number of mnemonics and format rules that the user must learn and manipulate. Additionally, the resulting programs are easier to interpret which in turn reduces programming errors and improves the maintainability of the software.

B. OPERANDS

Operands modify the opcodes and provide the information needed by the assembler to perform the designated operation.

Certain symbolic names are reserved as key words in the assembly language operand fields. They are:

1) The contents of 8-bit registers are specified by the character corresponding to the register names. The register names are A,B,C,D,E,H,L,I,R.

2) The contents of 16-bit double registers and register pairs consisting of two 8-bit registers are specified by the two characters corresponding to the register name or register pair. The names of double registers are IX,IY and SP. The names of registers pairs are AF,BC,DE and HL.

3) The contents of the auxiliary register pairs consisting of two 8-bit registers are specified by the two characters corresponding to the register pair names followed by an apostrophe. The auxiliary register pair names are AF′,BC′,DE′ and HL′. Only the pair AF′ is actually allowed as an operand, and then only in the EX AF,AF′ instruction.

4) The state of the four testable flags is specified as follows:

FLAG CONDITION	ON CONDITION	OFF
Carry	C	NC
Zero	Z	NZ
Sign	M (minus)	P (plus)
Parity	PE (even)	PO (odd)

OPERAND NOTATION

The following notation is used in the description of the assembly language:

1) r specifies any one of the following registers: A,B,C,D,E,H,L.
2) (HL) specifies the contents of memory at the location addressed by the contents of the register pair HL.
3) n specifies a one-byte expression in the range (0 to 255) nn specifies a two-byte expression in the range (0 to 65535).
4) d specifies a one-byte expression in the range (-128,127).
5) (nn) specifies the contents of memory at the location addressed by the two-byte expression nn.
6) b specifies an expression in the range (0,7).
7) e specifies a one-byte expression in the range (-126,129).
8) cc specifies the state of the Flags for conditional JR, JP, CALL and RET instructions.
9) qq specifies any one of the register pairs BC, DE, HL or AF.
10) ss specifies any one of the following register pairs: BC,DE,HL,SP.
11) pp specifies any one of the following register pairs: BC,DE,IX,SP.
12) rr specifies any one of the following register pairs: BC,DE,IY,SP.
13) s specifies any of r,n,(HL),(IX+d),(IY+d).
14) dd specifies any one of the following register pairs: BC,DE,HL,SP.
15) m specifies any of r,(HL),(IX+d),(IY+d).

C. RULES FOR WRITING ASSEMBLY STATEMENTS (SYNTAX)

An assembly language program (source program) consists of labels, opcodes, operands, comments and pseudo-ops in a sequence which defines the user's program.

There are 74 generic opcodes (such as LD), 25 operand key words (such as A), and 694 legitimate combinations of opcodes and operands in the Z80 instruction set.

ASSEMBLER STATEMENT FORMAT:

Statements are always written in a particular format. A typical Assembler statement is shown below:

```
LABEL      OPCODE     OPERANDS   COMMENT
LOOP:      LD         HL,VALUE   ;GET VALUE
```

In this example, the label, LOOP, provides a means for assigning a specific name to the instruction LOAD (LD), and is used to address the statement in other statements. The operand field contains one or two entries separated by one or more commas, tabs or spaces. The comment field is used by the programmer to quickly identify the action defined by the statement. Comments must begin with a semicolon and labels must be terminated by a colon, unless the label starts in column No. 1.

D ASSEMBLY LANGUAGE CONVENTIONS

LABELS

A label is a symbol representing up to 16 bits of information and is used to specify an address or data. By using labels effectively, the user can write assembly language programs more rapidly and make fewer errors. If the programmer attempts to use a symbol that has been defined as greater than 8 bits for an 8-bit data constant, the assembler will generate an error message.

A label is composed of a string of one or more characters, of which the first six must be unique. For example, the labels `longname' and `longnamealso' will be considered to be the same label. The first character must be alphabetic and any following characters must be either alphanumeric, the question mark (?) or the under bar character (_). Any other characters within a label will cause an error. A label can start in any column if immediately followed by a colon. It does not require a colon if started in column one.

The assembler maintains a location counter to provide addresses for the symbols in the label field. When a symbol is found in the label field, the assembler places the symbol and the corresponding location counter value in a symbol table.

The symbol table normally resides in RAM, but it will automatically overflow to disk, so there is no limit to the number of labels that can be processed.

EXPRESSIONS

An expression is an operand entry consisting of either a single term (unary) or a combination of terms (binary). It contains a valid series of constants, variables and functions that can be connected by operation symbols. The Z80 Assembler will accept a wide range of expressions involving arithmetic and logical operations. The assembler will evaluate all expressions from left to right in the order indicated in the table below:

OPERATOR	FUNCTION	PRIORITY
+	UNARY PLUS	1
-	UNARY MINUS	1
.NOT. or \	LOGICAL NOT	1
.RES.	RESULT	1
**	EXPONENTIATION	2
*	MULTIPLICATION	3
/	DIVISION	3
.MOD.	MODULO	3
.SHR.	LOGICAL SHIFT RIGHT	3
.SHL.	LOGICAL SHIFT LEFT	3
+	ADDITION	4
-	SUBTRACTION	4
.AND. or &	LOGICAL AND	5
.OR. or ↑	LOGICAL OR	6
.XOR.	LOGICAL XOR	6
.EQ. or =	EQUALS	7
.GT. or >	GREATER THAN	7
.LT. or <	LESS THAN	7
.UGT.	UNSIGNED GREATER THAN	7
.ULT.	UNSIGNED LESS THAN	7

Parenthesis can be used to ensure correct expression evaluation. Note, however, that enclosing an expression wholly in parenthesis indicates a memory address.

Delimiters such as spaces or commas are not allowed within an expression since they serve to separate the expression from other portions of the statement.

16-bit integer arithmetic is used throughout.

Note that the negative of an expression can be formed by a preceding minus sign -. For example:

```
LD HL,-0EA9H.
```

The five comparison operators (.EQ., .GT., .LT., .UGT. and .ULT.) will evaluate to a logical True (all ones) if the comparison is true logical False (zero) otherwise. The operators .GT. and .LT. deal with signed numbers whereas .UGT. and .ULT. assume unsigned arguments.

The Result operator (.RES.) causes overflow to be

suppressed during evaluation of its argument, thus overflow is not flagged with an error message.

For example:

LD BC,7FFFH+1 would cause an error message, whereas LD BC,.RES.(7FFFH+1) would not.

The Modulo operator (.MOD.) is defined as:

X.MOD.Y. = X-Y*(X/Y) where the division (X/Y) is integer division.

The Shift operator (.SHR.,.SHL.) shifts the first argument right or left by the number of positions given in the second argument. Zeros are shifted into the high-order or low-order bits, respectively.

In specifying relative addressing with either the JR (Jump Relative) or DJNZ (Decrement and Jump if Not Zero) instructions, the Assembler automatically subtracts the value of the next instruction's reference counter from the value given in the operand field to form the relative address for the jump instruction. For example:

```
JR C,LOOP
```

will jump relative to the instruction labeled LOOP if the Carry flag is set. The limits on the range of a relative address is 128 bytes in either direction from the reference counter of the next instruction. An error message will be generated if this range is exceeded.

The symbol $ is used to represent the value of the reference counter of the current instruction, and can be used in general expressions. An expression which evaluates to a displacement in the range <-126,+129> can be added to the reference counter to form a relative address. For example:

```
JR  C,$+5
```

will jump relative to the instruction which is 5 bytes beyond the current instruction.

PSEUDO-OPS (ASSEMBLER DIRECTIVES)

There are several pseudo-ops which the various Zilog assemblers will recognize. These assembler directives, although written much like processor instructions, are commands to the assembler instead of to the processor. They direct the assembler to perform specific tasks during the assembly process but have no meaning to the Z80 processor. These assembler pseudo-ops are:

ORG	nn	Sets address reference counter to the value nn.
EQU	nn	Sets value of a label to nn in the program: can occur only once for any label.
DEFL	nn	Sets value of a label to nn and can be repeated in the program with different values for the same label.
END		Signifies the end of the source program so that any following statement will be ignored. If there is no end statement, then the end-of-file mark in the last source file will designate the end of the source program.
DEFB	n	Defines the contents of a byte at the current reference counter to be n.
DEFB	's'	Defines the content of one byte of memory to be the ASCII representation of character s.
DEFW	nn	Defines the contents of a two-byte word to be nn. The least significant byte is located at the current reference counter while the most significant byte is located at the reference counter plus one.
DEFS	nn	Reserves nn bytes of memory starting at the current value of the reference counter.

DEFM	's'	Defines the content of n bytes of memory to be the ASCII representation of string s, where n is the length of s and must be in the range 0<=n<=63.
MACRO		#Po #P1...#Pn Declares the label to be a macro name with formal parameters Po through Pn. Subsequent statements define the body of the macro.
ENDM		Marks the end of a macro definition.

Pseudo-ops are assembled exactly like executable instructions, and may be preceded by a label and followed by a comment. (The label is required for EQU, DEFL and MACR pseudo-ops.) In the above pseudo-op definitions, the reference counter corresponds to the program counter and is used to assign and calculate machine-language addresses for the object file.

CONDITIONAL PSEUDO-OPS

Conditional pseudo-ops provide the programmer with the capability to conditionally include or not include portions of his source code in the assembly process. Conditional pseudo-ops are:

COND	nn	Evaluates expression nn. If the expression is true (non-zero), the COND pseudo-op is ignored. If the expression is false (zero), the assembly of subsequent statements is disabled. COND pseudo-ops cannot be nested.
ENDC		Re-enables assembly of subsequent statements.

DELIMITERS

A delimiter is used to specify the bounds of a certain related group of characters in a source program. The delimiters recognized by the assembler are commas or spaces. A delimiter cannot

occur within an expression.

COMMENTS

Comments are not a functional part of an assembly program, but instead are used for program documentation to add clarity, and to facilitate software maintenance. A comment is defined as any string following a semicolon in a line, and is ignored by the assembler. Comments can begin in any column.

I/O BUFFERS

The Z80 Assembler uses a buffered I/O technique for handling the assembly language source file, listing file, object file and temporary files. The assembler automatically determines the available work space and allocates the buffer sizes accordingly. Hence there are no constraints on the size of the assembly language source file that can be assembled.

UPPER/LOWER CASE

The assembler processes source text which contains both upper and lower case alphabetic characters in the following manner. All opcodes and keywords, such as register names or condition codes, must be either all capitals or all lower case. Label names may consist of any permutation of upper and lower case, however, two names which differ in case will be treated as two different names. Thus, LABEL, label and LaBel will be considered as three different names. Notice that one could use a mixture of case to allow definition of labels or macros which look similar to opcodes, such as Push or LdiR, without redefining the meaning of the opcode. All assembler commands, such as *List or *Include (see below) can be in either upper or lower case, as can arithmetic operators such as NOT.,.AND. or .EQ., and numbers can be any mixture of case, such as 0ffffh, 0AbCdH or 011001b.

NUMBER BASES

The Assembler will accept numbers in several

different bases: binary, octal, decimal and hexadecimal. Numbers must always start with a digit (leading zeros are sufficient), and may be followed immediately by a single letter which signifies the base of the number ('B' for binary, 'O' or 'Q' for octal, 'D' for decimal and 'H' for hexadecimal). If no base is specified decimal is assumed. For example, the same number is represented in each of the four bases:

1011100B, 134Q, 134O, 92, 92D, 05CH

E. ASSEMBLER COMMANDS

The Z80 Assembler recognizes several commands to modify the listing format. An assembler command is a line of the source file beginning with an * in column one. The character in column two identifies the type of command. Arguments, if any, are separated from the command by any number of blanks or commas. The following commands are recognized by the assembler:

*Eject	Causes the listing to advance to a new page starting with this line.
*Heading s	Causes string s to be taken as a heading to be printed at the top of each new page. Strings s may be any string of zero to 28 characters, not containing leading blanks. This command does an automatic Eject.
*List OFF	Causes listing and printing to be suspended, starting with this line.
*List ON	Causes listing and printing to resume, starting with this line.
*Maclist OFF	Causes listing and printing of macro expansions to be suspended, starting with this line.
*Maclist ON	Causes listing and printing of macro expansions to resume, starting with this line.
*Include filename	Causes the source file filename to be included in the source stream following the command statement.

The expected use of *Include is for files of macro definitions, lists of EQUates, or commonly used subroutines, although it can be used anywhere in a program that the other commands would be legal. The filename must follow the normal convention for specifying filenames, and furthermore only file types 'F' through 'T' are allowed. The default type is 'S'. The included file may also contain a *Include command, up to a nested level of four.

*Include will always try to shoe-horn the file in inside a macro definition, and although the *Include statement will appear in a macro expansion, the file will not be included again at the point of expansion. *Include works in the expected manner in conjunction with conditional assembly.
For example:

```
     COND exp

*Include FILE1

     ENDC

;FILE1 is included only if the value of exp is
non-zero.
```

III. MACROS

Macros provide a means for the user to define his own opcodes, or to redefine existing opcodes. A macro defines a body of text which will be automatically inserted in the source stream at each occurrence of a macro call. In addition, parameters provide a capability for making limited changes in the macro at each call.

If a macro is used to redefine an existing opcode, a warning message is generated to indicate that future use of that opcode will always be processed as a macro call. If a program uses macros, then the asembly option M must be specified.

MACRO DEFINITION

The body of text to be used as a macro is given in the macro definition. Each definition begins with a MACRO statement and end with an ENDM statement. The general forms are:

```
<name> MACRO [#<P0>,#<P1>,...,#<Pn>]

[<label>] ENDM
```

The label <name> is required, and must obey all the usual rules for forming labels. The quantity in brackets is an optional set of parameters.

There can be any number of parameters, each starting with the symbol #. The rest of the parameter name can be any string not containing a delimiter (blank, comma, semicolon) or the symbol #. However, parameters will be scanned left to right for a match, so the user is cautioned not to use parameter names which are prefix substrings of later parameter names. Parameter names are not entered in the symbol table.

The label on an ENDM is optional, but if one is given it must obey all the usual rules for forming labels.

Each statement between the MACRO and ENDM statements is entered into a temporary macro file. The only restriction on these statements is that they do not include another macro definition. (Nested definitions are not allowed.) They may

include macro calls. (Recursion is allowed.)

The statements of the macro body are not assembled at definition time, so they will not define labels, generate code, or cause errors. Exceptions are the assembler commands such as *List, which are executed wherever they occur. Within the macro body text, the formal parameter names may occur anywhere that an expansion-time substitution is desired. This includes comments and quoted strings. The symbol # may not occur except as the first symbol of a parameter name.

Macros must be defined before they are called.

MACRO CALLS AND MACRO EXPANSION

A macro is called by using its name as an opcode at any point after the definition. The general form is:

[<label>] <name> ['<S0>','<S1>',...,'Sn>']

The <label> is optional, and <name> must be a previously defined macro. There may be any number of argument strings, <Sn>, separated by any number of blanks or commas. Commas do not serve as parameter place holders, only as string delimeters. If there are too few parameters, the missing ones are assumed to be null. If there are too many, the extras are ignored. The position of each string in the list corresponds with the position of the macro parameter name it is to replace. Thus, the third string in a macro call statement will be substituted for each occurrence of the third parameter name.

The strings may be of any length and may contain any characters. The outer level quotes around the string are generally optional, but are required if the string contains delimiters or the quote character itself. The quote character is represented by two successive quote marks at the inner level. The outer level quotes, if present, will not occur in the substitution. The null string, represented by two successive quote marks at the outer level, may be used in any parameter position.

After processing the macro call statement, the assembler switches its input from the source file

to the macro file. Each statement of the macro body is scanned for occurrences of parameter names, and for each occurrence found, the corresponding string from the macro call statement is substituted. After substitution, the statement is assembled normally.

SYMBOL GENERATOR

Every macro definition has an implicit parameter named #$YM. This may be referenced by the user in the macro body, but should not explicitly appear in the MACRO statement. At expansion time, each occurrence of #$YM in the definition is replaced by a string representing a 4-digit hexadecimal constant.

This string is constant over a given level of macro expansion, but increases by one for each new macro call. The most common use of #$YM is to provide unigue labels for different expansion of the same macro. Otherwise, a macro containing a label would cause multiple definition errors if it were called more than once.

LISTING FORMAT

By default, each expanded statement is listed with a blank STMT field. If the Maclist flag is turned off by the NOM option or *M OFF, then only the macro call is listed.

IV. SUBROUTINES

Subroutines are blocks of instructions that can be called during the execution of a sequence of instructions. Subroutines can be called from main programs or from other subroutines. A subroutine is entered by the CALL opcode as in:

```
CALL REWIND
```

Parameters such as those used by the macros are not used with subroutines. When a call instruction is encountered during execution of a program, the PC is changed to the first instruction of the subroutine. The subsequent address of the invoking program is pushed on the stack. Control will return to this point when the subroutine is finished. The processor continues to execute the subroutine until it encounters a RET (return) instruction. At this point the return address is popped off the stack into the PC, and the processor returns to the address of the instruction following the CALL, to continue execution from that point.

Subroutines of any size can be invoked from programs or other subroutines of any size, without restriction. Care must be taken when nesting subroutines (subroutines within subroutines) that pushes and pops remain balanced at each level. If the processor encounters a RET with an un-popped push on the stack, the PC will be set to a meaningless address rather than to the next instruction following the CALL.

Tradeoffs must be considered between:

a) using a block of code repetitively in line, and
b) calling the block repetitively as a subroutine.

Program size can usually be saved by using the subroutine. If the repetitive block contains N bytes and it is repeated on M occasions in the program,

a) MxN bytes would be used in direct programming, while
b) 3M (for CALLS)

+ N (for the block)
+ 1 (for the RET)
= 3M+N+1 bytes would be required if using a subroutine.

For example, for a block of 20 bytes used 5 times, in-line programming would require 100 bytes while a subroutine would require 36.

An added advantage of subroutines is that with careful naming, program structures become clearer, easier to read and easier to debug and maintain. Subroutines written for one purpose can be employed elsewhere in other programs requiring the same function.

Subroutines differ from Macros in several ways:

a) Subroutine code is assembled into an object program only once although it may be called many times. Macro code is assembled in line every place the macro is used.

b) Registers and pointers required by a subroutine must be set up before the CALL. No parameters are used and no argument string can be issued. Macros, through their use of parameters, can modify the settings of registers on each occurrence.

V. Z80 STATUS INDICATORS (FLAGS)

The flag register (F and F') supplies information to the user regarding the status of the Z80 at any given time. The bit positions for each flag is shown below:

7	6	5	4	3	2	1	0
S	Z	X	H	X	P/V	N	C

WHERE:

C = CARRY FLAG
N = ADD/SUBTRACT FLAG
P/V = PARITY/OVERFLOW FLAG
H = HALF-CARRY FLAG
Z = ZERO FLAG
S = SIGN FLAG
X = NOT USED

Each of the two Z-80 Flag Registers contains 6 bits of status information which are set or reset by CPU operations. (Bits 3 and 5 are not used.) Four of these bits are testable (C,P/V,Z and S) for use with conditional jump, call or return instructions. Two flags are not testable (H,N) and are used for BCD arithmetic.

CARRY FLAG (C)

The carry bit is set or reset depending on the operation being performed. For 'ADD' instructions that generate a carry and 'SUBTRACT' instructions that generate a borrow, the Carry Flag will be set. The Carry Flag is reset by an ADD that does not generate a carry and a 'SUBTRACT' that generates no borrow. This saved carry facilitates software routines for extended precision arithmetic. Also, the "DAA" instruction will set the Carry Flag if the conditions for making the decimal adjustment are met.

For instructions RLA, RRA, RLS and RRS, the carry bit is used as a link between the LSB and MSB for any register or memory location. During instructions RLCA, RLC s and SLA s, the carry contains the last value shifted out of bit 7 of any register or memory location. During

instructions RRCA, RRC s, SRA s and SRL s the carry contains the last value shifted out of bit 0 of any register or memory location.

For the logical instructions AND s, OR s and XOR s, the carry will be reset.

The Carry Flag can also be set (SCF) and complemented (CCF).

ADD/SUBTRACT FLAG (N)

This flag is used by the decimal adjust accumulator instruction (DAA) to distinguish between 'ADD' and 'SUBTRACT' instructions. For all 'ADD' instructions, N will be set to an '0'. For all 'SUBTRACT' instructions, N will be set to a '1'.

PARITY/OVERFLOW FLAG

This flag is set to a particular state depending on the operation being performed.

For arithmetic operations, this flag indicates an overflow condition when the result in the Accumulator is greater than the maximum possible number (+127) or is less than the minimum possible number (-128). This overflow condition can be determined by examining the sign bits of the operands.

For addition, operands with different signs will never cause overflow. When adding operands with like signs and the result has a different sign, the overflow flag is set. For example:

```
+120 = 0111 1000   ADDEND
+105 = 0110 1001   AUGEND
+225   1110 0001   (-95) SUM
```

The two numbers added together has resulted in a number that exceeds +127 and the two positive operands has resulted in a negative number (-95) which is incorrect. The overflow flag is therefore set.

For subtraction, overflow can occur for operands of unlike signs. Operands of like sign will never cause overflow. For example:

```
   +127   0111 1111   MINUEND
(-) -64   1100 0000   SUBTRAHEND
   +191   1011 1111   DIFFERENCE
```

The minuend sign has changed from a positive to a negative, giving an incorrect difference. Overflow is therefore set.

Another method for predicting an overflow is to observe the carry into and out of the sign bit. If there is a carry in and no carry out, or if there is no carry in and a carry out, then overflow has occurred.

This flag is also used with logical operations and rotate instructions to indicate the parity of the result. The number of '1' bits in a byte are counted. If the total is odd, 'ODD' parity (P=0) is flagged. If the total is even, 'EVEN' parity is flagged (P=1).

During search instructions (CPI,CPIR,CPD,CPDR) and block transfer instructions (LDI,LDIR, LDD,LDDR) the P/V flag monitors the state of the byte count register (BC). When decrementing, the byte counter results in a zero value, the flag is reset to 0, otherwise the flag is a Logic 1.

During LD A,I and LD A,R instructions, the P/V flag will be set with the contents of the interrupt enable flip-flop (IFF2) for storage or testing.

When inputting a byte from an I/O device, IN r,(C), the flag will be adjusted to indicate the parity of the data.

THE HALF CARRY FLAG (H)

The Half Carry Flag (H) will be set or reset depending on the carry and borrow status between bits 3 and 4 of an 8-bit arithmetic operation. This flag is used by the decimal adjust accumulator instruction (DAA) to correct the result of a packed BCD add or subtract operation. The H flag will be set (1) or reset (0) according to the following table:

H	ADD	SUBTRACT
1	There is a carry from Bit 3 to Bit 4	There is borrow from bit 4
0	There is no carry from Bit 3 to Bit 4	There is no borrow from Bit 4

THE ZERO FLAG (Z)

The Zero Flag (Z) is set or reset if the result generated by the execution of certain instructions is a zero.

For 8-bit arithmetic and logical operations, the Z flag will be set to a '1' if the resulting byte in the Accumulator is zero. If the byte is not zero, the Z flag is reset to '0'.

For compare (search) instructions, the Z flag will be set to a '1' if a comparison is found between the value in the Accumulator and the memory location pointed to by the contents of the register pair HL.

When testing a bit in a register or memory location, the Z flag will contain the complemented state of the indicated bit (see Bit b,s).

When inputting or outputting a byte between a memory location and an I/O device (INI;IND;0UTI and OUTD), if the result of B-1 is zero, the Z flag is set, otherwise it is reset. Also for byte inputs from I/O devices using IN r,(C), the Z Flag is set to indicate a zero byte input.

THE SIGN FLAG (S)

The Sign Flag (S) stores the state of the most significant bit of the Accumulator (Bit 7). When the Z80 performs arithmetic operations on signed numbers, binary two's complement notation is used to represent and process numeric information. A positive number is identified by a '0' in bit 7. A negative number is identified by a '1'. The binary equivalent of the magnitude of a positive number is stored in bits 0 to 6 for a total range of from 0 to 127. A negative number is represented by the two's complement of the equivalent positive number. The total range for negative numbers is from -1 to -128.

When inputting a byte from an I/O device to a register, IN r,(C), the S flag will indicate either positive (S=0) or negative (S=1) data.

VI. Z80 INSTRUCTION SET

NOTE: Execution time (E.T.) for each instruction is given in microseconds for an assumed 4 MHZ clock. Total machine cycles (M) are indicated with total clock periods (T States). Also indicated are the number of T States for each M cycle. For example:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

indicates that the instruction consists of 2 machine cycles. The first cycle contains 4 clock periods (T States). The second cycle contains 3 clock periods for a total of 7 clock periods or T States. The instruction will execute in 1.75 microseconds.

Register format is shown for each instruction with the most significant bit to the left and the least significant bit to the right.

Z80 INSTRUCTION SET
TABLE OF CONTENTS

PAGE

-8 BIT LOAD GROUP..... 26

-16 BIT LOAD GROUP..... 52

-EXCHANGE, BLOCK TRANSFER
AND SEARCH GROUP..... 76

-8 BIT ARITHMETIC AND LOGICAL GROUP..... 99

-GENERAL PURPOSE ARITHMETIC
AND CPU CONTROL GROUPS..... 131

-16 BIT ARITHMETIC GROUP..... 146

-ROTATE AND SHIFT GROUP..... 163

-BIT SET, RESET AND TEST GROUP..... 202

-JUMP GROUP..... 219

-CALL AND RETURN GROUP..... 237

-INPUT AND OUTPUT GROUP..... 252

-INSTRUCTION INDEX..... 275

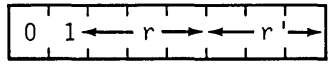
8 BIT LOAD GROUP

LD r, r′

Operation: r←r′

Format:

Opcode	Operands
LD	r,r′



Description:

The contents of any register r′ are loaded into any other register r. Note: r,r′ identifies any of the registers A, B, C, D, E, H, or L, assembled as follows in the object code:

Register	r,r′
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.0

Condition Bits Affected: None

Example:

If the H register contains the number 8AH, and the E register contains 10H, the instruction

```
LD H, E
```

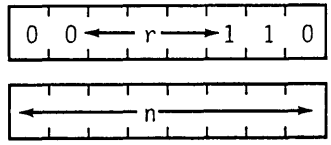
would result in both registers containing 10H.

LD r, n

Operation: r ← n

Format:

Opcode	Operands
LD	r, n



Description:

The eight-bit integer n is loaded into any register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

After the execution of

```
LD E, A5H
```

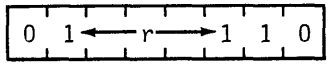
the contents of register E will be A5H.

LD r, (HL)

Operation: r ← (HL)

Format:

Opcode	Operands
LD	r, (HL)



Description:

The eight-bit contents of memory location (HL) are loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If register pair HL contains the number 75A1H, and memory address 75A1H contains the byte 58H, the execution of

```
LD C, (HL)
```

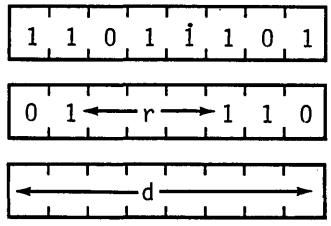
will result in 58H in register C.

LD r, (IX+d)

Operation: r ← (IX+d)

Format:

Opcode	Operands
LD	r, (IX+d)



Description:

The operand (IX+d) (the contents of the Index Register IX summed with a two's complement displacement integer d) is loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register r

A = 111
B = 000
C = 001
D = 010
E = 011
H = 100
L = 101

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the Index Register IX contains the number 25AFH, the instruction

```
LD B, (IX+19H)
```

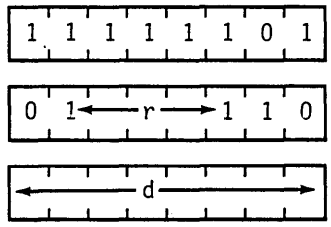
will cause the calculation of the sum 25AFH + 19H, which points to memory location 25C8H. If this address contains byte 39H, the instruction will result in register B also containing 39H.

LD r, (IY+d)

Operation: r ← (IY+d)

Format:

Opcode	Operands
LD	r, (IY+d)



Description:

The operand (IY+d) (the contents of the Index Register IY summed with a two's complement displacement integer d) is loaded into register r, where r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the Index Register IY contains the number 25AFH, the instruction

```
LD B, (IY+19H)
```

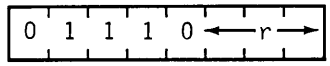
will cause the calculation of the sum 25AFH + 19H, which points to memory location 25C8H. If this address contains byte 39H, the instruction will result in register B also containing 39H.

LD (HL), r

Operation: (HL) ← r

Format:

Opcode	Operands
LD	(HL), r



Description:

The contents of register r are loaded into the memory location specified by the contents of the HL register pair. The symbol r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the contents of register pair HL specifies memory location 2146H, and the B register contains the byte 29H, after the execution of

```
LD (HL), B
```

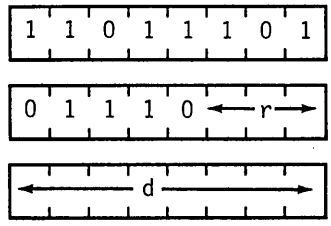
memory address 2146H will also contain 29H.

LD (IX+d), r

Operation: (IX+d) ← r

Format:

Opcode	Operands
LD	(IX+d), r



Description:

The contents of register r are loaded into the memory address specified by the contents of Index Register IX summed with d, a two's complement displacement integer. The symbol r identifies register A, B, C, D, E, H or L, assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the C register contains the byte 1CH, and the Index Register IX contains 3100H, then the instruction

```
LD (IX+6H), C
```

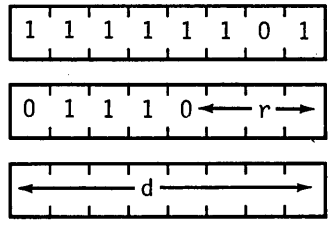
will perform the sum 3100H + 6H and will load 1CH into memory location 3106H.

LD (IY+d), r

Operation: (IY+d) ← r

Format:

Opcode	Operands
LD	(IY+d), r



Description:

The contents of register r are loaded into the memory address specified by the sum of the contents of the Index Register IY and d, a two's complement displacement integer. The symbol r is specified according to the following table.

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the C register contains the byte 48H, and the Index Register IY contains 2A11H, then the instruction

```
LD (IY+4H), C
```

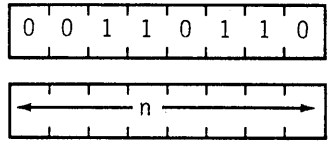
will perform the sum 2A11H + 4H, and will load 48H into memory location 2A15.

LD (HL), n

Operation: (HL) ← n

Format:

Opcode	Operand
LD	(HL),n



Description:

Integer n is loaded into the memory address specified by the contents of the HL register pair.

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the HL register pair contains 4444H, the instruction

```
LD (HL), 28H
```

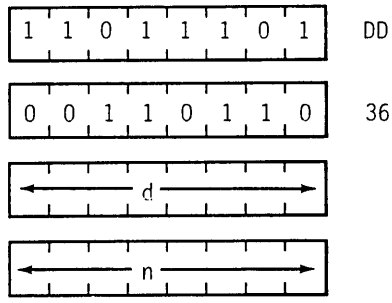
will result in the memory location 4444H containing the byte 28H.

LD (IX+d), n

Operation: (IX+d) ← n

Format:

Opcode	Operands
LD	(IX+d), n



Description:

The n operand is loaded into the memory address specified by the sum of the contents of the Index Register IX and the two's complement displacement operand d.

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the Index Register IX contains the number 219AH the instruction

LD (IX+5H), 5AH

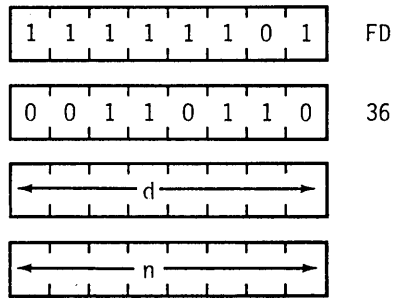
would result in the byte 5AH in the memory address 219FH.

LD (IY+d), n

Operation: (IY+d) ← n

Format:

Opcode	Operands
LD	(IY+d),n



Description:

Integer n is loaded into the memory location specified by the contents of the Index Register summed with the two's complement displacement integer d.

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected: NONE

Example:

If the Index Register IY contains the number A940H, the instruction

LD (IY+10H), 97H

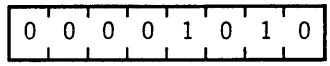
would result in byte 97 in memory location A950H.

LD A, (BC)

Operation: A ← (BC)

Format:

Opcode	Operands
LD	A, (BC)



0A

Description:

The contents of the memory location specified by the contents of the BC register pair are loaded into the Accumulator.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the BC register pair contains the number 4747H, and memory address 4747H contains the byte 12H, then the instruction

```
LD A, (BC)
```

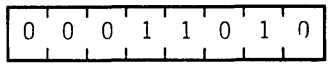
will result in byte 12H in register A.

LD A, (DE)

Operation: A ← (DE)

Format:

Opcode	Operands
LD	A, (DE)



1A

Description:

The contents of the memory location specified by the register pair DE are loaded into the Accumulator.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the DE register pair contains the number 30A2H and memory address 30A2H contains the byte 22H, then the instruction

```
LD A, (DE)
```

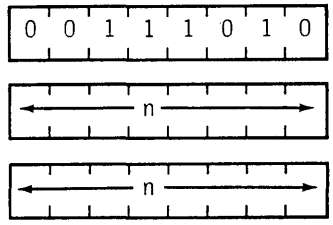
will result in byte 22H in register A.

LD A, (nn)

Operation: A ← (nn)

Format:

Opcode	Operands
LD	A, (nn)



Description:

The contents of the memory location specified by the operands nn are loaded into the Accumulator. The first n operand after the op code is the low ordder byte of a two-byte memory address.

M CYCLES: 4 T STATES: 13(4,3,3,3) 4 MHZ E.T.: 3.25

Condition Bits Affected: None

Example:

If the contents of nn is number 8832H, and the content of memory address 8832H is byte 04H, after the instruction

LD A, (nn)

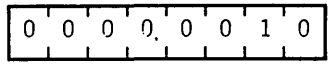
byte 04H will be in the Accumulator.

LD (BC), A

Operation: (BC) ← A

Format:

Opcode	Operands
LD	(BC),A



02

Description:

The contents of the Accumulator are loaded into the memory location specified by the contents of the register pair BC.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the Accumulator contains 7AH and the BC register pair contains 1212H the instruction

```
LD (BC),A
```

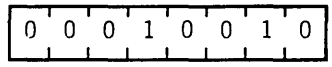
will result in 7AH being in memory location 1212H.

LD (DE), A

Operation: (DE) ← A

Format:

Opcode	Operands
LD	(DE),A



12

Description:

The contents of the Accumulator are loaded into the memory location specified by the contents of the DE register pair.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

If the contents of register pair DE are 1128H, and the Accumulator contains byte A0H, the instruction

```
LD (DE),A
```

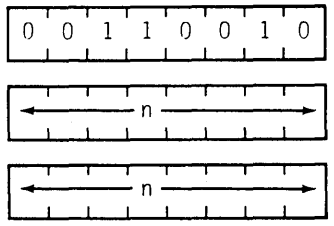
will result in A0H being in memory location 1128H.

LD (nn), A

Operation: (nn) ← A

Format:

Opcode	Operands
LD	(nn),A



Description:

The contents of the Accumulator are loaded into the memory address specified by the operand nn. The first n operand after the op code is the low order byte of nn.

M CYCLES: 4 T STATES: 13(4,3,3,3) 4 MHZ E.T.: 3.25

Condition Bits Affected: None

Example:

If the contents of the Accumulator are byte D7H, after the execution of

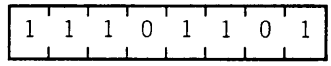
```
LD (3141H),A
```

D7H will be in memory location 3141H.

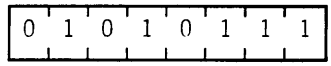
Operation: A ← I

Format:

Opcode	Operands
LD	A, I



ED



57

Description:

The contents of the Interrupt Vector Register I are loaded into the Accumulator.

M CYCLES: 2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

Condition Bits Affected:

S:	Set if I-Reg. is negative; reset otherwise
Z:	Set if I-Reg. is zero; reset otherwise
H:	Reset
P/V:	Contains contents of IFF2
N:	Reset
C:	Not affected

Note:

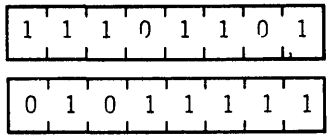
If an interrupt occurs during execution of this instruction, the Parity flag will contain a 0.

LD A, R

Operation: A ← R

Format:

Opcode	Operands
LD	A,R



Description:

The contents of Memory Refresh Register R are loaded into the Accumulator.

M CYCLES: 2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

Condition Bits Affected:

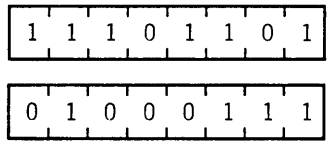
S:	Set if R-Reg. is negative; reset otherwise
Z:	Set if R-Reg. is zero; reset otherwise
H:	Reset
P/V:	Contains contents of IFF2
N:	Reset
C:	Not affected

LD I, A

Operation: I ← A

Format:

Opcode	Operands
LD	I,A



Description:

The contents of the Accumulator are loaded into the Interrupt Control Vector Register, I.

M CYCLES: 2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

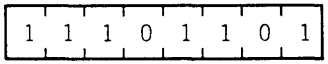
Condition Bits Affected: None

LD R, A

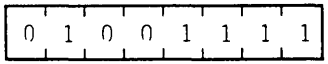
Operation: R ← A

Format:

Opcode	Operands
LD	R,A



ED



4F

Description:

The contents of the Accumulator are loaded into the Memory Refresh register R.

M CYCLES: 2 T STATES: 9(4,5) 4 MHZ E.T.: 2.25

Condition Bits Affected: None

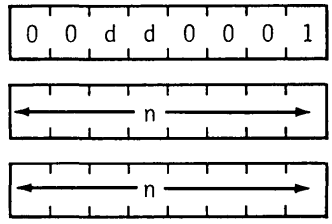
-16 BIT LOAD GROUP-

LD dd, nn

Operation: **dd ← nn**

Format:

Opcode	Operands
LD	dd, nn



Description:

The two-byte integer nn is loaded into the dd register pair, where dd defines the BC, DE, HL, or SP register pairs, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand after the op code is the low order byte.

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

After the execution of

```
LD HL, 5000H
```

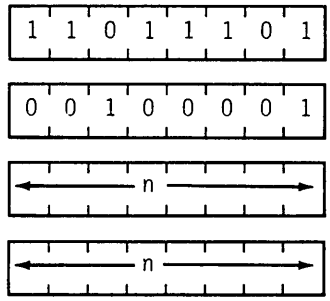
the contents of the HL register pair will be 5000H.

LD IX, nn

Operation: IX ← nn

Format:

Opcode	Operands
LD	IX,nn



Description:

Integer nn is loaded into the Index Register IX. The first n operand after the op code is the low order byte.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

After the instruction

```
LD IX,45A2H
```

the Index Register will contain integer 45A2H.

LD IY, nn

Operation: IY ← nn

Format:

Opcode	Operands	
LD	IY,nn	
1 1 1 1 1 1 0 1		FD
0 0 1 0 0 0 0 1		21
n		
n		

Description:

Integer nn is loaded into the Index Register IY. The first n operand after the op code is the low order byte.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

After the instruction:

LD IY,7733H

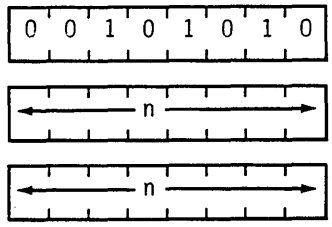
the Index Register IY will contain the integer 7733H.

LD HL, (nn)

Operation: H ← (nn+1), L ← (nn)

Format:

Opcode	Operands
LD	HL,(nn)



Description:

The contents of memory address (nn) are loaded into the low order portion of register pair HL (register L), and the contents of the next highest memory address (nn+1) are loaded into the high order portion of HL (register H). The first n operand after the op code is the low order byte of nn.

M CYCLES: 5 T STATES: 16(4,3,3,3,3) 4 MHZ E.T.: 4.00

Condition Bits Affected: None

Example:

If address 4545H contains 37H and address 4546H contains A1H after the instruction

LD HL,(4545H)

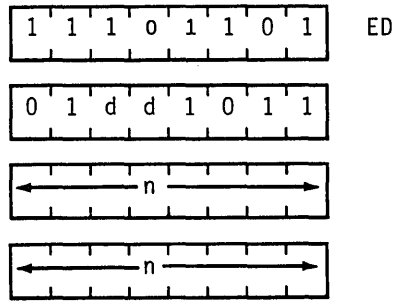
the HL register pair will contain A137H.

LD dd, (nn)

Operation: $dd_H \leftarrow (nn+1)$ $dd_L \leftarrow (nn)$

Format:

Opcode	Operands
LD	dd,(nn)



Description:

The contents of address (nn) are loaded into the low order portion of register pair dd, and the contents of the next highest memory address (nn+1) are loaded into the high order portion of dd. Register pair dd defines BC, DE, HL, or SP register pairs, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand after the op code is the low order byte of (nn).

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: None

Example:

If Address 2130H contains 65H and address 2131M contains 78H after the instruction

```
LD BC,(2130H)
```

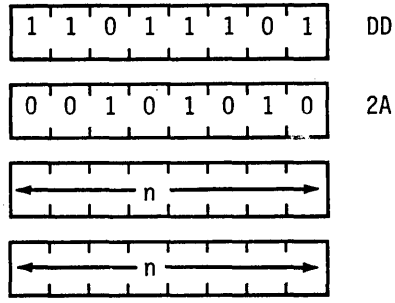
the BC register pair will contain 7865H.

LD IX, (nn)

Operation: $IX_H \leftarrow (nn+1)$, $IX_L \leftarrow (nn)$

Format:

Opcode	Operands
LD	IX,(nn)



Description:

The contents of the address (nn) are loaded into the low order portion of Index Register IX, and the contents of the next highest memory address (nn+1) are loaded into the high order portion of IX. The first n operand after the op code is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: None

Example:

If address 6666H contains 92H and address 6667H contains DAH, after the instruction

```
LD IX,(6666H)
```

the Index Register IX will contain DA92H.

LD IY, (nn)

Operation: $IY_H \leftarrow (nn+1)$, $IY_L \leftarrow (nn)$

Format:

Opcode	Operands	
LD	IY,(nn)	
1 1 1 1 1 1 0 1		FD
0 0 1 0 1 0 1 0		2A
n		
n		

Description:

The contents of address (nn) are loaded into the low order portion of Index Register IY, and the contents of the next highest memory address (nn+1) are loaded into the high order portion of IY. The first n operand after the op code is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: None

Example:

If address 6666H contains 92H and address 6667H contains DAH, after the instruction

```
LD IY,(6666H)
```

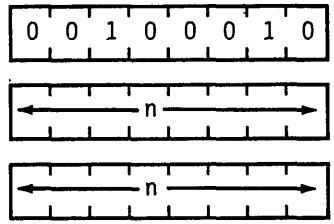
the Index Register IY will contain DA92H.

LD (nn), HL

Operation: (nn+1) ← H, (nn) ← L

Format:

Opcode	Operands
LD	(nn),HL



Description:

The contents of the low order portion of register pair HL (register L) are loaded into memory address (nn), and the contents of the high order portion of HL (register H) are loaded into the next highest memory address (nn+1). The first n operand after the op code is the low order byte of nn.

M CYCLES: 5 T STATES: 16(4,3,3,3,3) 4 MHZ E.T.: 4.00

Condition Bits Affected: None

Example:

If the content of register pair HL is 483AH, after the instruction

```
LD (B229H),HL
```

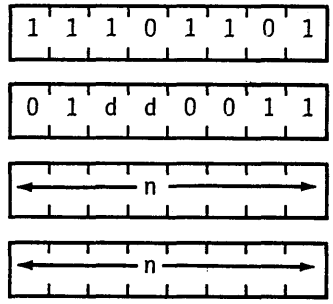
address B229H) will contain 3AH, and address B22AH will contain 48H.

LD (nn), dd

Operation: $(nn+1) \leftarrow dd_H$, $(nn) \leftarrow dd_L$

Format:

Opcode	Operands
LD	(nn),dd



Description:

The low order byte of register pair dd is loaded into memory address (nn); the upper byte is loaded into memory address (nn+1). Register pair dd defines either BC, DE, HL, or SP, assembled as follows in the object code:

Pair	dd
BC	00
DE	01
HL	10
SP	11

The first n operand after the op code is the low order byte of a two byte memory address.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: None

Example:

If register pair BC contains the number 4644H, the instruction

```
LD (1000H),BC
```

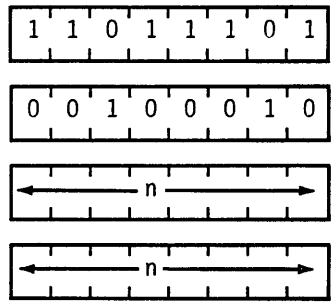
will result in 44H in memory location 1000H, and 46H in memory location 1001H.

LD (nn), IX

Operation: $(nn+1) \leftarrow IX_H$, $(nn) \leftarrow IX_L$

Format:

Opcode	Operands
LD	(nn),IX



Description:

The low order byte in Index Register IX is loaded into memory address (nn); the upper order byte is loaded into the next highest address (nn+1). The first n operand after the op code is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: None

Example:

If the Index Register IX contains 5A30H, after the instruction

```
LD (4392H),IX
```

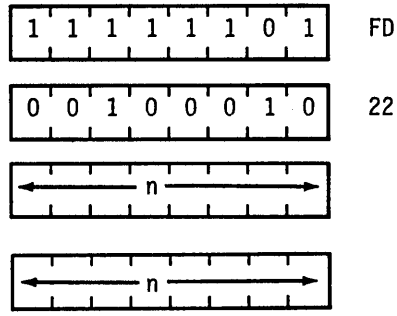
memory location 4392H will contain number 30H and location 4393H will contain 5AH.

LD (nn), IY

Operation: $(nn+1) \leftarrow IY_H$, $(nn) \leftarrow IY_L$

Format:

Opcode	Operands
LD	(nn),IY



Description:

The low order byte in Index Register IY is loaded into memory address (nn); the upper order byte is loaded into memory location (nn+1). The first n operand after the op code is the low order byte of nn.

M CYCLES: 6 T STATES: 20(4,4,3,3,3,3) 4 MHZ E.T.: 5.00

Condition Bits Affected: None

Example:

If the Index Register IY contains 4174H after the instruction

LD (8838H),IY

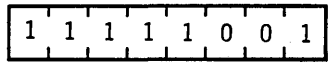
memory location 8838H will contain number 74H and memory location 8839H will contain 41H.

LD SP, HL

Operation: SP ← HL

Format:

Opcode	Operands
LD	SP,HL



F9

Description:

The contents of the register pair HL are loaded into the Stack Pointer SP.

M CYCLES: 1 T STATES: 6 4 MHZ E.T.: 1.50

Condition Bits Affected: None

Example:

If the register pair HL contains 442EH, after the instruction

```
LD SP,HL
```

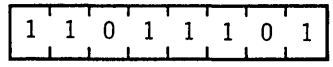
the Stack Pointer will also contain 442EH.

LD SP, IX

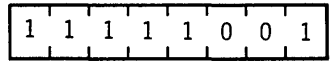
Operation: SP ← IX

Format:

Opcode	Operands
LD	SP,IX



DD



F9

Description:

The two byte contents of Index Register IX are loaded into the Stack Pointer SP.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Index Register IX are 98DAH, after the instruction

LD SP,IX

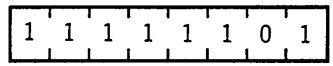
the contents of the Stack Pointer will also be 98DAH.

LD SP, IY

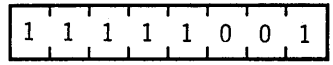
Operation: SP ← IY

Format:

Opcode	Operands
LD	SP,IY



FD



F9

Description:

The two byte contents of Index Register IY are loaded into the Stack Pointer SP.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If Index Register IY contains the integer A227H, after the instruction

```
LD SP,IY
```

the Stack Pointer will also contain A227H.

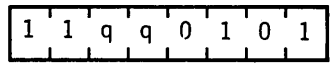
PUSH qq

PUSH qq

Operation: $(SP-2) \leftarrow qq_L$, $(SP-1) \leftarrow qq_H$

Format:

Opcode	Operands
PUSH	qq



Description:

The contents of the register pair qq are pushed into the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first decrements the SP and loads the high order byte of register pair qq into the memory address now specified by the SP; then decrements the SP again and loads the low order byte of qq into the memory location corresponding to this new address in the SP. The operand qq identifies register pair BC, DE, HL, or AF, assembled as follows in the object code:

Pair	qq
BC	00
DE	01
HL	10
AF	11

M CYCLES: 3 T STATES: 11(5,3,3) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

If the AF register pair contains 2233H and the Stack Pointer contains 1007H, after the instruction

```
PUSH AF
```

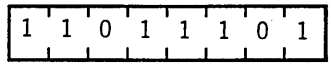
memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H.

PUSH IX

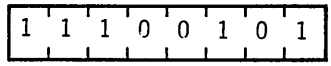
Operation: $(SP-2) \leftarrow IX_L$, $(SP-1) \leftarrow IX_H$

Format:

Opcode	Operands
PUSH	IX



DD



E5

Description:

The contents of the Index Register IX are pushed into the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first decrements the SP and loads the high order byte of IX into the memory address now specified by the SP; then decrements the SP again and loads the low order byte into the memory location corresponding to this new address in the SP.

M CYCLES: 3 T STATES: 15(4,5,3,3) 4 MHZ E.T.: 3.75

Condition Bits Affected: None

Example:

If the Index Register IX contains 2233H and the Stack Pointer contains 1007H, after the instruction

```
PUSH IX
```

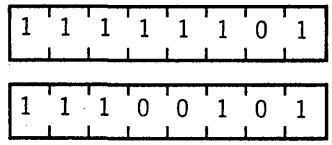
memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H.

PUSH IY

Operation: $(SP-2) \leftarrow IY_L$, $(SP-1) \leftarrow IY_H$

Format:

Opcode	Operands
PUSH	IY



Description:

The contents of the Index Register IY are pushed into the external memory LIFO (last-in, first-out) Stack. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first decrements the SP and loads the high order byte of IY into the memory address now specified by the SP; then decrements the SP again and loads the low order byte into the memory location corresponding to this new address in the SP.

M CYCLES: 4 T STATES: 15(4,5,3,3) 4 MHZ E.T.: 3.75

Condition Bits Affected: None

Example:

If the Index Register IY contains 2233H and the Stack Pointer contains 1007H, after the instruction

```
PUSH IY
```

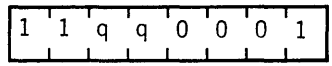
memory address 1006H will contain 22H, memory address 1005H will contain 33H, and the Stack Pointer will contain 1005H.

POP qq

Operation: $qq_H \leftarrow (SP+1)$, $qq_L \leftarrow (SP)$

Format:

Opcode	Operands
POP	qq



Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into register pair qq. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first loads into the low order portion of qq, the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded into the high order portion of qq and the SP is now incremented again. The operand qq identifies register pair BC, DE, HL, or AF, assembled as follows in the object code:

Pair	r
BC	00
DE	01
HL	10
AF	11

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction

```
POP HL
```

will result in register pair HL containing 3355H, and the Stack Pointer containing 1002H.

Operation: $IX_H \leftarrow (SP+1)$, $IX_L \leftarrow (SP)$

Format:

Opcode	Operands	
POP	IX	
1 1 0 1 1 1 0 1		DD
1 1 1 0 0 0 0 1		E1

Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into Index Register IX. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first loads into the low order portion of IX the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded into the high order portion of IX. The SP is now incremented again.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction

```
POP IX
```

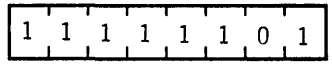
will result in Index Register IX containing 3355H, and the Stack Pointer containing 1002H.

POP IY

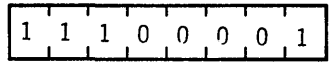
Operation: $IY_H \leftarrow (SP+1)$, $IY_L \leftarrow (SP)$

Format:

Opcode	Operands
POP	IY



FD



E1

Description:

The top two bytes of the external memory LIFO (last-in, first-out) Stack are popped into Index Register IY. The Stack Pointer (SP) register pair holds the 16-bit address of the current "top" of the Stack. This instruction first loads into the low order portion of IY the byte at the memory location corresponding to the contents of SP; then SP is incremented and the contents of the corresponding adjacent memory location are loaded into the high order portion of IY. The SP is now incremented again.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

If the Stack Pointer contains 1000H, memory location 1000H contains 55H, and location 1001H contains 33H, the instruction

```
POP IY
```

will result in Index Register IY containing 3355H, and the Stack Pointer containing 1002H.

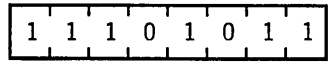
-EXCHANGE, BLOCK TRANSFER AND SEARCH GROUP-

EX DE, HL

Operation: DE ↔ HL

Format:

Opcode	Operands
EX	DE,HL



EB

Description:

The two-byte contents of register pairs DE and HL are exchanged.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

If the content of register pair DE is the number 2822H, and the content of the register pair HL is number 499AH, after the instruction

```
EX DE,HL
```

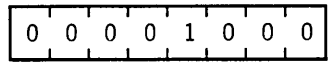
the content of register pair DE will be 499AH and the content of register pair HL will be 2822H.

EX AF, AF'

Operation: AF ↔ AF'

Format:

Opcode	Operands
EX	AF,AF'



08

Description:

The two-byte contents of the register pairs AF and AF' are exchanged. (Note: register pair AF' consists of registers A' and F'.)

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

If the content of register pair AF is number 9900H, and the content of register pair AF' is number 5944H, after the instruction

```
EX AF,AF'
```

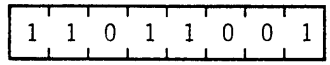
the contents of AF will be 5944H, and the contents of AF' will be 9900H.

EXX

Operation: (BC) ↔ (BC'), (DE) ↔ (DE'), (HL) ↔ (HL')

Format:

Opcode	Operands
EXX	



D9

Description:

Each two-byte value in register pairs BC, DE, and HL is exchanged with the two-byte value in BC′, DE′, and HL′, respectively.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

If the contents of register pairs BC, DE, and HL are the numbers 445AH, 3DA2H, and 8859H, respectively, and the contents of register pairs BC′, DE′, and HL′ are 0988H, 9300H, and 00E7H, respectively, after the instruction

```
EXX
```

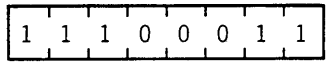
the contents of the register pairs will be as follows: BC: 0988H; DE: 9300H; HL: 00E7H; BC′: 445AH; DE′: 3DA2H; and HL′: 8859H.

EX (SP), HL

Operation: H ↔ (SP+1), L ↔ (SP)

Format:

Opcode	Operands
EX	(SP),HL



E3

Description:

The low order byte contained in register pair HL is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of HL is exchanged with the next highest memory address (SP+1).

M CYCLES: 5 T STATES: 19(4,3,4,3,5) 4 MHZ E.T.: 4.75

Condition Bits Affected: None

Example:

If the HL register pair contains 7012H, the SP register pair contains 8856H, the memory location 8856H contains the byte 11H, and the memory location 8857H contains the byte 22H, then the instruction

```
EX (SP),HL
```

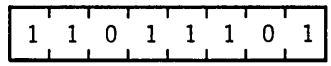
will result in the HL register pair containing number 2211H, memory location 8856H containing the byte 12H, the memory location 8857H containing the byte 70H and the Stack Pointer containing 8856H.

EX (SP), IX

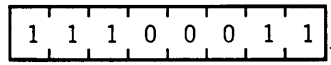
Operation: $IX_H \leftrightarrow (SP+1), IX_L \leftrightarrow (SP)$

Format:

Opcode	Operands
EX	(SP),IX



DD



E3

Description:

The low order byte in Index Register IX is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of IX is exchanged with the next highest memory address (SP+1).

M CYCLES: 6 T STATES: 23(4,4,3,4,3,5) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the Index Register IX contains 3988H, the SP register pair contains 0100H, the memory location 0100H contains the byte 90H, and memory location 0101H contains byte 48H, then the instruction

```
EX (SP),IX
```

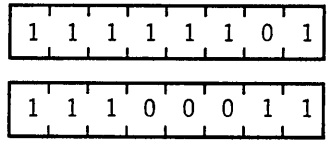
will result in the IX register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H and the Stack Pointer containing 0100H.

EX (SP), IY

Operation: $IY_H \leftrightarrow (SP+1)$, $IY_L \leftrightarrow (SP)$

Format:

Opcode	Operands
EX	(SP),IY



Description:

The low order byte in Index Register IY is exchanged with the contents of the memory address specified by the contents of register pair SP (Stack Pointer), and the high order byte of IY is exchanged with the next highest memory address (SP+1).

M CYCLES: 6 T STATES: 23(4,4,3,4,3,5) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the Index Register IY contains 3988H, the SP register pair contains 0100H, the memory location 0100H contains the byte 90H, and memory location 0101H contains byte 48H, then the instruction

```
EX (SP),IY
```

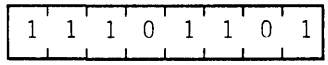
will result in the IY register pair containing number 4890H, memory location 0100H containing 88H, memory location 0101H containing 39H, and the Stack Pointer containing 0100H.

LDI

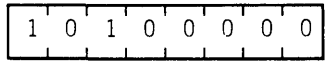
Operation: (DE) ← (HL), DE ← DE+1, HL ← HL+1, BC ← BC-1

Format:

Opcode	Operands
LDI	



ED



A0

Description:

A byte of data is transferred from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both these register pairs are incremented and the BC (Byte Counter) register pair is decremented.

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Set if BC-1≠0; reset otherwise
N:	Reset
C:	Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains contains the byte 88H, the DE register pair contains 2222H, the memory location 2222H contains byte 66H, and the BC register pair contains 7H, then the instruction

```
LDI
```

will result in the following contents in register pairs and memory addresses:

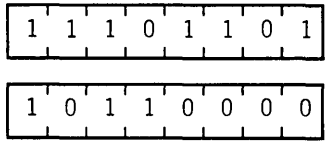
```
    HL    :   1112H
(1111H)   :   88H
    DE    : 2223H
(2222H)   :   88H
    BC    :    6H
```

LDIR

Operation: **(DE) ← (HL), DE ← DE+1, HL ← HL+1, BC ← BC-1**

Format:

Opcode	Operands
LDIR	



Description:

This two byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the DE register pair. Then both these register pairs are incremented and the BC (Byte Counter) register pair is decremented. If decrementing causes the BC to go to zero, the instruction is terminated. If BC is not zero the program counter is decremented by 2 and the instruction is repeated. Interrupts will be recognized and two refresh cycless will be executed after each data transfer. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes.

For BC≠0:

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC=0:

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Reset
N:	Reset
C:	Not affected

Example:

If the HL register pair contains 1111H, the DE register pair contains 2222H, the BC register pair contains 0003H, and memory locations have these contents:

```
(1111H) : 88H       (2222H) : 66H
(1112H) : 36H       (2223H) : 59H
(1113H) : A5H       (2224H) : C5H
```

then after the execution of

```
LDIR
```

the contents of register pairs and memory locations will be:

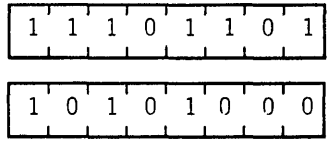
```
HL : 1114H
DE : 2225H
BC : 0000H

(1111H) : 88H       (2222H) : 88H
(1112H) : 36H       (2223H) : 36H
(1113H) : A5H       (2224H) : A5H
```

LDD

Operation: (DE) ← (HL), DE ← DE-1, HL ← HL-1, BC ← BC-1

Format:

Opcode	Operands
LDD	



Descripttion:

This two byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both of these register pairs including the BC (Byte Counter) register pair are decremented.

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Set if BC-1≠0; reset otherwise
N:	Reset
C:	Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains the byte 88H, the DE register pair contains 2222H, memory location 2222H contains byte 66H, and the BC register pair contains 7H, then the instruction

```
LDD
```

will result in the following contents in register pairs and memory addresses:

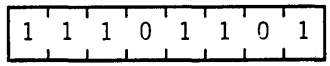
```
    HL   :  1110H
(1111H)  :    88H
    DE   :  2221H
(2222H)  :    88H
    BC   :     6H
```

LDDR

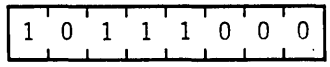
Operation: (DE) ← (HL), DE ← DE-1, HL ← HL-1, BC ← BC-1

Format:

Opcode	Operands
LDDR	



ED



B8

Description:

This two byte instruction transfers a byte of data from the memory location addressed by the contents of the HL register pair to the memory location addressed by the contents of the DE register pair. Then both of these registers as well as the BC (Byte Counter) are decremented. If decrementing causes the BC to go to zero, the instruction is terminated. If BC is not zero, the program counter is decremented by 2 and the instruction is repeated. Interrupts will be recognized and two refresh cycless will be executed after each data transfer. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes.

For BC=0:

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC=0:

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Reset
N:	Reset

Example:

If the HL register pair contains 1114H, the DE register pair contains 2225H, the BC register pair contains 0003H, and memory locations have these contents:

```
(1114H) : A5H     (2225H) : C5H
(1113H) : 36H     (2224H) : 59H
(1112H) : 88H     (2223H) : 66H
```

then after the execution of

```
LDDR
```

the contents of register pairs and memory locations will be:

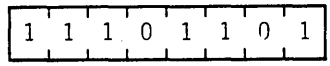
```
HL : 1111H
DE : 2222H
BC : 0000H

(1114H) : A5H     (2225H) : A5H
(1113H) : 36H     (2224H) : 36H
(1112H) : 88H     (2223H) : 88H
```

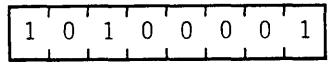
CPI

Operation: A - (HL), HL ← HL+1, BC ← BC-1

Format:

Opcode	Operands
CPI	





Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. Then HL is incremented and the Byte Counter (register pair BC) is decremented.

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A=(HL); reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Set if BC-1=0; reset otherwise
N:	Set
C:	Not affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains 0001H, then after the execution of

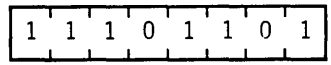
the Byte Counter will contain 0000H, the HL register pair will contain 1112H, the Z flag in the F register will be set, and the P/V flag in the F register will be reset. There will be no effect on the contents of the Accumulator or address 1111H.

CPIR

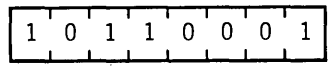
Operation: A - (HL), HL ← HL+1, BC ← BC-1

Format:

Opcode	Operands
CPIR	



ED



B1

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL is incremented and the Byte Counter (register pair BC) is decremented. If decrementing causes the BC to go to zero or if A=(HL), the instruction is terminated. If BC is not zero and A=(HL), the program counter is decremented by 2 and the instruction is repeated. Interrupts will be recognized and two refresh cycles will be executed after each data transfer. Note that if BC is set to zero before instruction execution, the instruction will loop through 64K bytes, if no match is found.

For BC≠0 and A≠(HL):

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC=0 or A=(HL):

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A=(HL); reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Set if BC-1=0; reset otherwise
N:	Set
C:	Not affected

Example:

If the HL register pair contains 1111H, the Accumulator contains F3H, the Byte Counter contains 0007H, and memory locations have these contents:

```
(1111H) : 52H
(1112H) : 00H
(1113H) : F3H
```

then after the execution of

```
CPIR
```

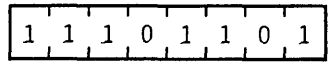
the contents of register pair HL will be 1114H, the contents of the Byte Counter will be 0004H, the P/V flag in the F register will be set and the Z flag in the F register will be set.

CPD

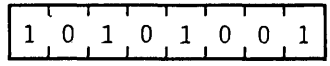
Operation: A - (HL), HL ← HL-1, BC ← BC-1

Format:

Opcode	Operands
CPD	



ED



A9

Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL and the Byte Counter (register pair BC) are decremented.

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A=(HL); reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Set if BC-1=0; reset otherwise
N:	Set
C:	Not Affected

Example:

If the HL register pair contains 1111H, memory location 1111H contains 3BH, the Accumulator contains 3BH, and the Byte Counter contains 0001H, then after the execution of

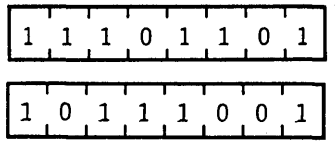
the Byte Counter will contain 0000H, the HL register pair will contain 1110H, the Z flag in the F register will be set, and the P/V flag in the F register will be reset. There will be no effect on the contents of the Accumulator or address 1111H.

CPDR

Operation: A - (HL), HL ← HL-1, BC ← BC-1

Format:

Opcode	Operands
CPDR	



Description:

The contents of the memory location addressed by the HL register pair is compared with the contents of the Accumulator. In case of a true compare, a condition bit is set. The HL and BC (Byte Counter) register pairs are decremented. If decrementing causes the BC to go to zero or if A=(HL), the instruction is terminated. If BC is not zero and A=(HL), the program counter is decremented by 2 and the instruction is repeated. Interrupts will be recognized and two refresh cycless will be executed after each data transfer. Note that if BC is set to zero prior to instruction execution, the instruction will loop through 64K bytes, if no match is found.

For BC≠0 and A≠(HL):

M CYCLES: 5 T STATES: 21(4,4,3,5,5) 4 MHZ E.T.: 5.25

For BC=0 or A=(HL):

M CYCLES: 4 T STATES: 16(4,4,3,5) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if A=(HL); reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Set if BC-1=0; reset otherwise
N:	Set
C:	Not affected

Example:

If the HL register pair contains 1118H, the Accumulator contains F3H, the Byte Counter contains 0007H, and memory locations have these contents:

(1118H) : 52H
(1117H) : 00H
(1116H) : F3H

then after the execution of

CPDR

the contents of register pair HL will be 1115H, the contents of the Byte Counter will be 0004H, the P/V flag in the F register will be set, and the Z flag in the F register will be set.

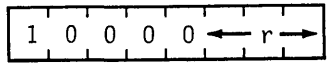
-8 BIT ARITHMETIC AND LOGICAL GROUP-

ADD A, r

Operation: A ← A + r

Format:

Opcode	Operands
ADD	A,r



Description:

The contents of register r are added to the contents of the Accumulator, and the result is stored in the Accumulator. The symbol r identifies the registers A,B,C,D,E,H or L assembled as follows in the object code:

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 7; reset otherwise

Example:

If the contents of the Accumulator are 44H, and the contents of register C are 11H, after the execution of

```
ADD A,C
```

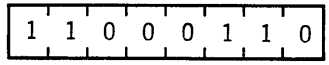
the contents of the Accumulator will be 55H.

ADD A, n

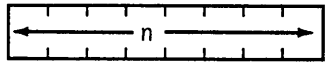
Operation: A ← A + n

Format:

Opcode	Operands
ADD	A,n



C6



Description:

The integer n is added to the contents of the Accumulator and the results are stored in the Accumulator.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 7; reset otherwise

Example:

If the contents of the Accumulator are 23H, after the execution of

```
ADD A,33H
```

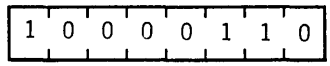
the contents of the Accumulator will be 56H.

ADD A, (HL)

Operation: A ← A + (HL)

Format:

Opcode	Operands
ADD	A,(HL)



86

Description:

The byte at the memory address specified by the contents of the HL register pair is added to the contents of the Accumulator and the result is stored in the Accumulator.

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 7; reset otherwise

Example:

If the contents of the Accumulator are A0H, and the content of the register pair HL is 2323H, and memory location 2323H contains byte 08H, after the execution of

```
ADD A,(HL)
```

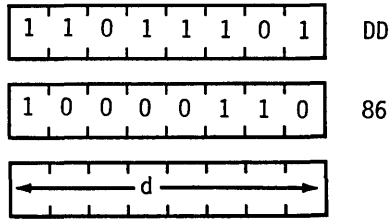
the Accumulator will contain A8H.

ADD A, (IX+d)

Operation: $A \leftarrow A + (IX+d)$

Format:

Opcode	Operands
ADD	A,(IX+d)



Description:

The contents of the Index Register (register pair IX) is added to a two's complement displacement d to point to an address in memory. The contents of this address is then added to the contents of the Accumulator and the result is stored in the Accumulator.

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 7; reset otherwise

Example:

If the Accumulator contents are 11H, the Index Register IX contains 1000H, and if the content of memory location

1005H is 22H, after the execution of

```
ADD A,(IX+5H)
```

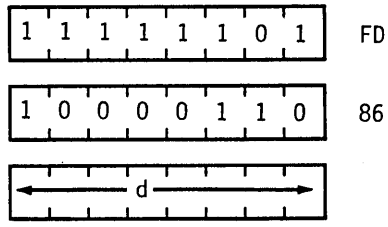
the contents of the Accumulator will be 33H.

ADD A, (IY+d)

Operation: A ← A+(IY+d)

Format:

Opcode	Operands
ADD	A,(IY+d)



Description:

The contents of the Index Register (register pair IY) is added to a two's complement displacement d to point to an address in memory. The contents of this address is then added to the contents of the Accumulator and the result is stored in the Accumulator.

M CYCLES: 5 T STATES: 19(4,4,3,5,3) 4 MHZ E.T.: 4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from bit 7; reset otherwise

Example:

If the Accumulator contents are 11H, the Index Register pair IY contains 1000H, and if the content of memory

location 1005H is 22H, after the execution of

```
ADD A,(IY+5H)
```

the contents of the Accumulator will be 33H.

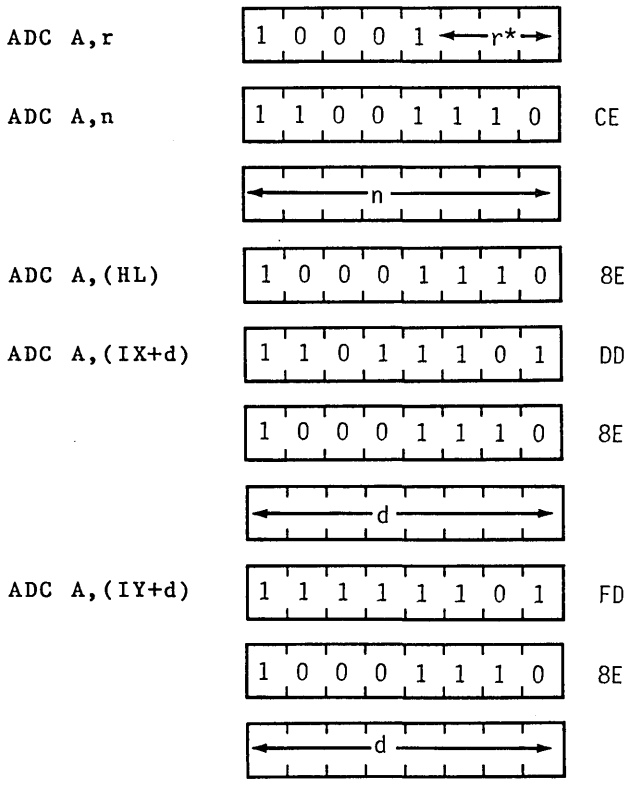
ADC A, s

Operation: A ← A + s + CY

Format:

Opcode	Operands
ADC	A,s

The s operand is any of r,n,(HL),(IX+d) or (IY+d) as defined for the analogous ADD instruction. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The s operand, along with the Carry Flag ("C" in the F register) is added to the contents of the Accumulator, and the result is stored in the Accumulator.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
ADC A,r	1	4	1.00
ADC A,n	2	7(4,3)	1.75
ADC A,(HL)	2	7(4,3)	1.75
ADC A,(IX+d)	5	19(4,4,3,5,3)	4.75
ADC A,(IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 7; reset otherwise

Example:

If the Accumulator contains 16H, the Carry Flag is set, the HL register pair contains 6666H, and address 6666H contains 10H, after the execution of

```
ADC A,(HL)
```

the Accumulator will contain 27H.

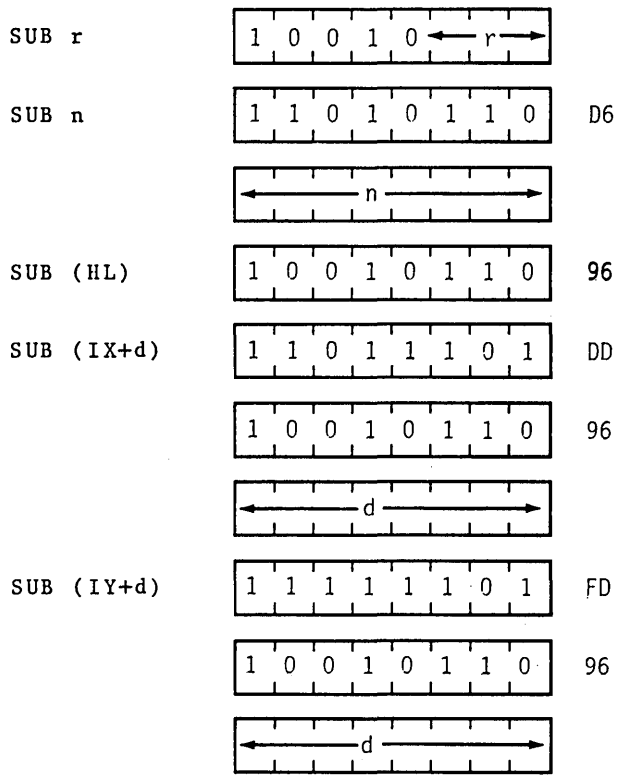
SUB s

Operation: A ← A - s

Format:

Opcode	Operands
SUB	s

The s operand is any of r,n,(HL),(IX+d) or (IY+d) as defined for the analogous ADD instruction. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The s operand is subtracted from the contents of the Accumulator, and the result is stored in the Accumulator.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SUB r	1	4	1.00
SUB n	2	7(4,3)	1.75
SUB (HL)	2	7(4,3)	1.75
SUB (IX+d)	5	19(4,4,3,5,3)	4.75
SUB (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Set
C:	Set if borrow; reset otherwise

Example:

If the Accumulator contains 29H and register D contains 11H, after the execution of

```
SUB D
```

the Accumulator will contain 18H.

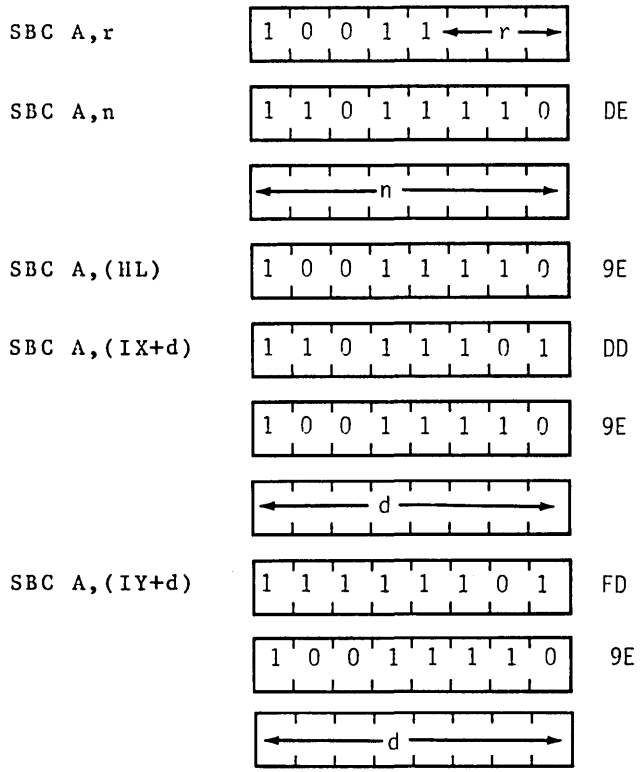
SBC A, s

Operation: A ← A - s - CY

Format:

Opcode	Operands
SBC	A,s

The s operand is any of r,n,(HL),(IX+d) or (IY+d) as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The s operand, along with the Carry Flag ("C" in the F register) is subtracted from the contents of the Accumulator, and the result is stored in the Accumulator.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SBC A,r	1	4	1.00
SBC A,n	2	7(4,3)	1.75
SBC A,(HL)	2	7(4,3)	1.75
SBC A,(IX+d)	5	19(4,4,3,5,3)	4.75
SBC A,(IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set if borrow from Bit 4; reset otherwise
P/V: Set if overflow; reset otherwise
N: Set
C: Set if borrow; reset otherwise

Example:

If the Accumulator contains 16H, the carry flag is set, the HL register pair contains 3433H, and address 3433H contains 05H, after the execution of

```
SBC A,(HL)
```

the Accumulator will contain 10H.

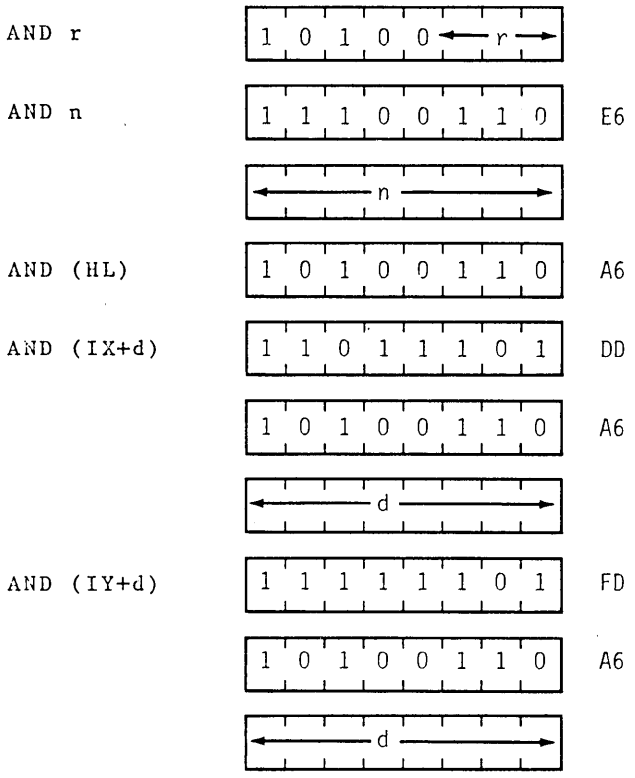
AND s

Operation: $A \leftarrow A \wedge s$

Format:

Opcode	Operands
AND	s

The s operand is any of r,n,(HL),(IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

A logical AND operation is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
AND r	1	4	1.00
AND n	2	7(4,3)	1.75
AND (HL)	2	7(4,3)	1.75
AND (IX+d)	5	19(4,4,3,5,3)	4.75
AND (IX+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S: Set if result is negative; reset otherwise
Z: Set if result is zero; reset otherwise
H: Set
P/V: Set if parity even; reset otherwise
N: Reset
C: Reset

Example:

If the B register contains 7BH (0111 1011) and the Accumulator contains C3H (1100 0011) after the execution of

```
AND B
```

the Accumulator will contain 43H (01000011).

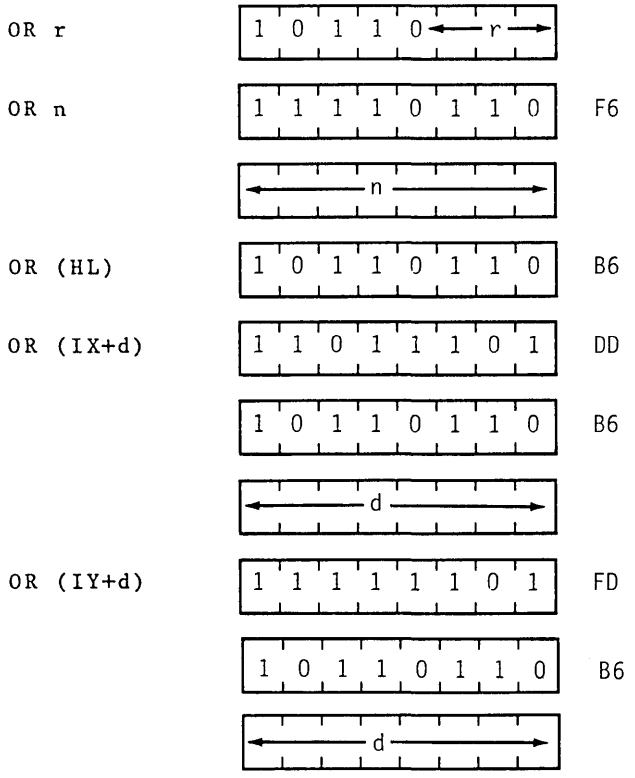
OR s

Operation: $A \leftarrow A \vee s$

Format:

Opcode	Operands
OR	s

The s operand is any of r,n,(HL),(IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

A logical OR operation is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
OR r	1	4	1.00
OR n	2	7(4,3)	1.75
OR (HL)	2	7(4,3)	1.75
OR (IX+d)	5	19(4,4,3,5,3)	4.75
OR (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Reset

Example:

If the H register contains 48H (010001000) and the Accumulator contains 12H (00010010) after the execution of

OR H

the Accumulator will contain 5AH (01011010).

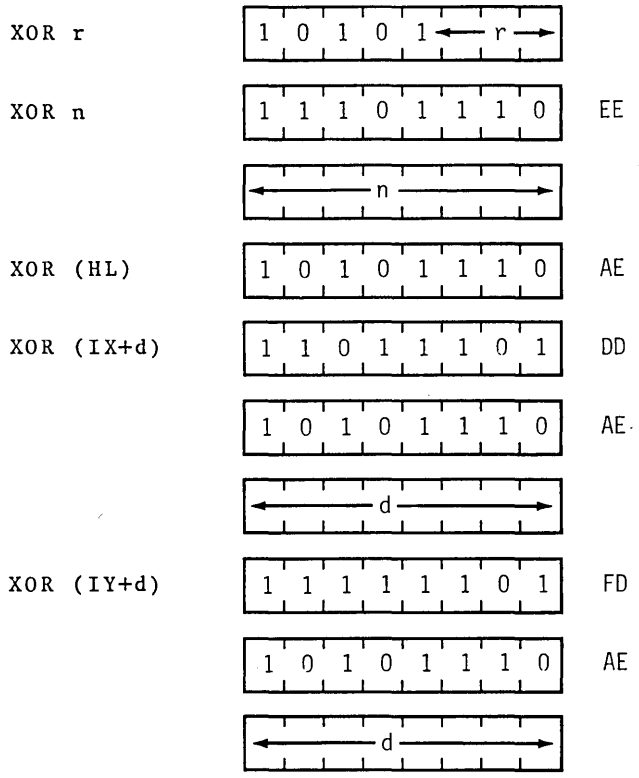
XOR s

Operation: A←A⊕s

Format:

Opcode	Operands
XOR	s

The s operand is any of r,n, (HL),(IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

A logical exclusive-OR operation is performed between the byte specified by the s operand and the byte contained in the Accumulator; the result is stored in the Accumulator.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
XOR r	1	4	1.00
XOR n	2	7(4,3)	1.75
XOR (HL)	2	7(4,3)	1.75
XOR (IX+d)	5	19(4,4,3,5,3)	4.75
XOR (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Reset

Example:

If the Accumulator contains 96H (10010110), after the execution of

XOR 5DH (Note: 5DH = 01011101)

the Accumulator will contain CBH (11001011).

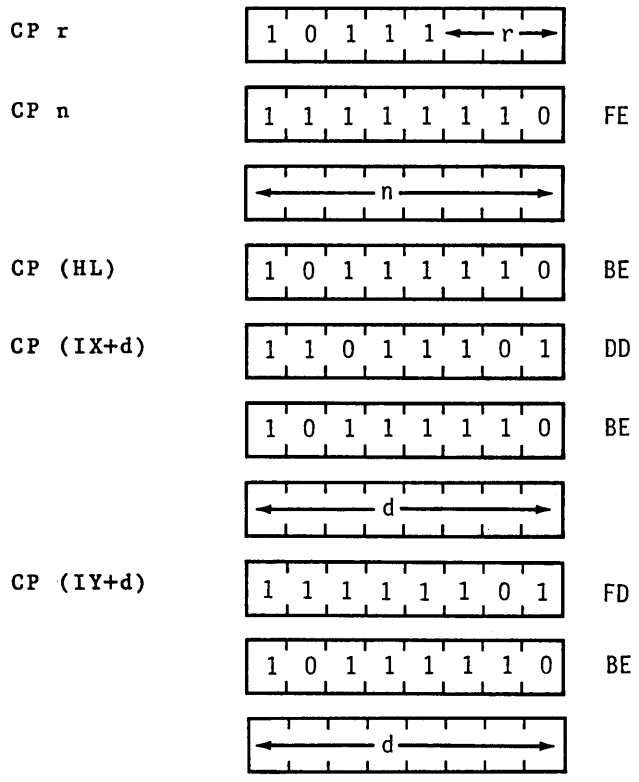
CP s

Operation: A - s

Format:

Opcode	Operands
CP	s

The s operand is any of r,n,(HL),(IX+d) or (IY+d), as defined for the analogous ADD instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of the s operand are compared with the contents of the Accumulator. If there is a true compare, the Z flag is set. The execution of this instruction does not affect the contents of the Accumulator.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
CP r	1	4	1.00
CP n	2	7(4,3)	1.75
CP (HL)	2	7(4,3)	1.75
CP (IX+d)	5	19(4,4,3,5,3)	4.75
CP (IY+d)	5	19(4,4,3,5,3)	4.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Set
C:	Set if borrow; reset otherwise

Example:

If the Accumulator contains 63H, the HL register pair contains 6000H and memory location 6000H contains 60H, the instruction

```
CP (HL)
```

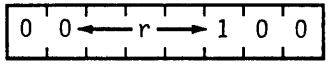
will result in the P/V flag in the F register being reset.

INC r

Operation: r←r+1

Format:

Opcode	Operands
INC	r



Description:

Register r is incremented. r identifies any of the registers A,B, C,D,E,H or L, assembled as follows in the object code.

Register	r
A	111
B	000
C	001
D	010
E	011
H	100
L	101

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if r was 7FH before operation; reset otherwise
N:	Reset
C:	Not affected

Example:

If the contents of register D are 28H, after the execution of

```
INC D
```

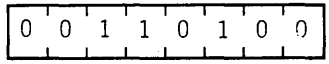
the contents of register D will be 29H.

INC (HL)

Operation: (HL) ← (HL)+1

Format:

Opcode	Operands
INC	(HL)



34

Description:

The byte contained in the address specified by the contents of the HL register pair is incremented.

M CYCLES: 3 T STATES: 11(4,4,3) 4 MHZ E.T.: 2.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if (HL) was 7FH before operation; reset otherwise
N:	Reset
C:	Not Affected

Example:

If the contents of the HL register pair are 3434H, and the contents of address 3434H are 82H, after the execution of

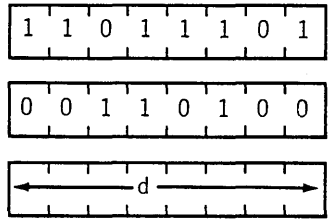
```
INC (HL)
```

memory location 3434H will contain 83H.

Operation: (IX+d) ← (IX+d)+1

Format:

Opcode	Operands
INC	(IX+d)



Description:

The contents of the Index Register IX (register pair IX) are added to a two's complement displacement integer d to point to an address in memory. The contents of this address are then incremented.

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if (IX+d) was 7FH before operation; reset otherwise
N:	Reset
C:	Not affected

Example:

If the contents of the Index Register pair IX are 2020H, and the memory location 2030H contains byte 34H, after the execution of

```
INC (IX+10H)
```

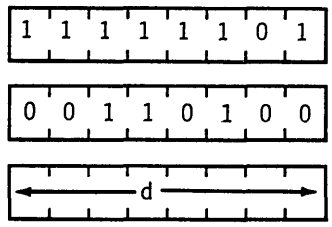
the contents of memory location 2030H will be 35H.

INC (IY+d)

Operation: (IY+d)← (IY+d)+1

Format:

Opcode	Operands
INC	(IY+d)



Description:

The contents of the Index Register IY (register pair IY) are added to a two's complement displacement integer d to point to an address in memory. The contents of this address are then incremented.

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry from Bit 3; reset otherwise
P/V:	Set if (IY+d) was 7FH before operation; reset otherwise
N:	Reset
C:	Not Affected

Example:

If the contents of the Index Register pair IY are 2020H, and the memory location 2030H contain byte 34H, after the execution of

```
INC (IY+10H)
```

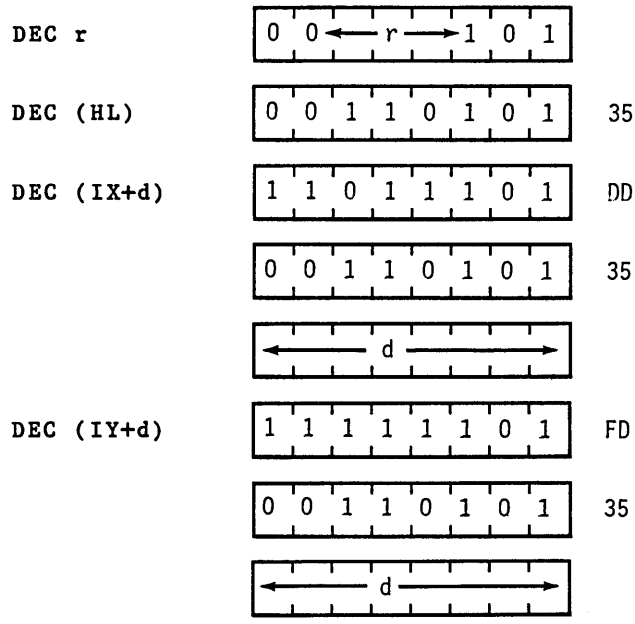
the contents of memory location 2030H will be 35H.

Operation: m ← m-1

Format:

Opcode	Operands
DEC	m

The m operand is any of r, (HL),(IX+d) or (IY+d), as defined for the analogous INC instructions. These various possible opcode-operand combinations are assembled as follows in the object code:



*r identifies registers B,C,D,E,H,L or A assembled as follows in the object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The byte specified by the m operand is decremented.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
DEC r	1	4	1.00
DEC (HL)	3	11(4,4,3)	2.75
DEC (IX+d)	6	23(4,4,3,5,4,3)	5.75
DEC (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if borrow from Bit 4, reset otherwise
P/V:	Set if m was 80H before operation; reset otherwise
N:	Set
C:	Not affected

Example:

If the D register contains byte 2AH, after the execution of

```
DEC D
```

register D will contain 29H.

-GENERAL PURPOSE ARITHMETIC AND CPU CONTROL GROUPS-

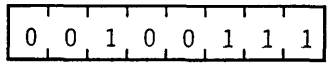
DAA

Operation: —

Format:

Opcode

DAA



27

Description:

This instruction conditionally adjusts the Accumulator for BCD addition and subtraction operations. For addition (ADD, ADC, INC) or subtraction (SUB, SBC,DEC,NEG), the following table indicates the operation performed:

OPERATION	C BEFORE DAA	HEX VALUE IN UPPER DIGIT (bit 7-4)	H BEFORE DAA	HEX VALUE IN LOWER DIGIT (bit 3-0)	NUMBER ADDED TO BYTE	C AFTER DAA
ADD, ADC, INC	0	0-9	0	0-9	00	0
	0	0-8	0	A-F	06	0
	0	0-9	1	0-3	06	0
	0	A-F	0	0-9	60	1
	0	9-F	0	A-F	66	1
	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
SUB, SBC, DEC, NEG	0	0-9	0	0-9	00	0
	0	0-8	1	6-F	FA	0
	1	7-F	0	0-9	A0	1
	1	6-F	1	6-F	9A	1

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Set if most significant bit of Acc. is 1 after operation; reset otherwise
Z:	Set if Acc. is zero after operation; reset otherwise
H:	See instruction
P/V:	Set if Acc. is even parity after operation; reset otherwise
N:	Not affected
C:	See instruction

Example:

If an addition operation is performed between 15 (BCD) and 27 (BCD), simple decimal arithmetic gives this result:

```
 15
+27
 42
```

But when the binary representations are added in the Accumulator according to standard binary arithmetic,

```
 0001  0101
+0010  0111
 0011  1100   3C
```

the sum is ambiguous. The DAA instruction adjusts this result so that the correct BCD representation is obtained:

```
 0011  1100
+0000  0110
 0100  0010 = 42
```

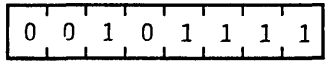
CPL

Operation: $A \leftarrow \overline{A}$.

Format:

Opcode

CPL



2F

Description:

The contents of the Accumulator (register A) are inverted (1's complement).

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Set
P/V: Not affected
N: Set
C: Not affected

Example:

If the contents of the Accumulator are 1011 0100, after the execution of

CPL

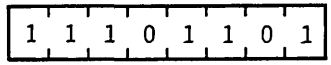
the Accumulator contents will be 0100 1011.

Operation: A ← 0-A

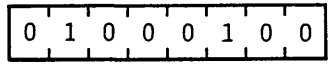
Format:

Opcode

NEG



ED



44

Description:

The contents of the Accumulator are negated (two's complement). This is the same as subtracting the contents of the Accumulator from zero. Note that 80H is left unchanged.

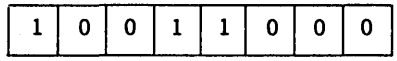
M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if borrow from Bit 4; reset otherwise
P/V:	Set if Acc. was 80H before operation; reset otherwise
N:	Set
C:	Set if Acc. was not 00H before operation; reset otherwise

Example:

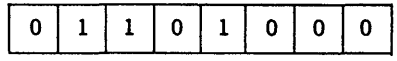
If the contents of the Accumulator are



after the execution of

NEG

the Accumulator contents will be

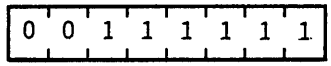


Operation: $CY \leftarrow \overline{CY}$

Format:

Opcode

CCF



3F

Description:

The Carry flag in the F register is inverted.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Previous carry will be copied
P/V:	Not affected
N:	Reset
C:	Set if CY was 0 before operation; reset otherwise

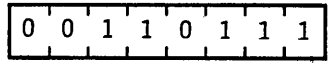
SCF

Operation: CY ← 1

Format:

Opcode

SCF



37

Description:

The Carry flag in the F register is set.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

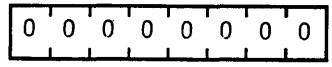
S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Not affected
N:	Reset
C:	Set

Operation: —

Format:

Opcode

NOP



00

Description:

The CPU performs no operation during this machine cycle.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

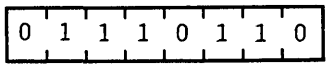
HALT

Operation: —

Format:

Opcode

HALT



76

Description:

The HALT instruction suspends CPU operation until a subsequent interrupt or reset is received. While in the halt state, the processor will execute NOP's to maintain memory refresh logic.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

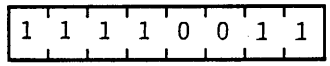
Condition Bits Affected: None

Operation: IFF ← 0

Format:

Opcode

DI



F3

Description:

DI disables the maskable interrupt by resetting the interrupt enable flip-flops(IFF1 and IFF2). Note that this instruction disables the maskable interrupt during its execution.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

When the CPU executes the instruction

DI

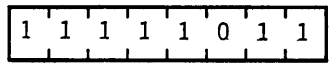
the maskable interrupt is disabled until it is subsequently re-enabled by an EI instruction. The CPU will not respond to an Interrupt Request (INT) signal.

EI

Operation: IFF ← 1

Format:

Opcode
EI



FB

Description:

The enable interrupt instruction will set both interrupt enable flip flops (IFF1 and IFF2) to a logic '1' allowing recognition of any maskable interrupt. Note that during the execution of this instruction and the following instruction, maskable interrupts will be disabled.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

When the CPU executes instruction

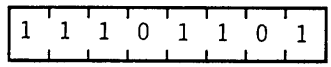
```
EI
RETI
```

the maskable interrupt will be enabled after the execution of the RETI instruction.

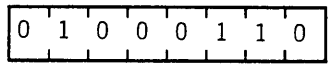
Operation: —

Format:

Opcode	Operands
IM	0



ED



46

Description:

The IM 0 instruction sets interrupt mode 0. In this mode the interrupting device can insert any instruction on the data bus for execution by the CPU. The first byte of a multi-byte instruction is read during the interrupt acknowledge cycle. Subsequent bytes are read in by a normal memory read sequence.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

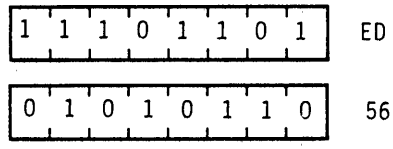
Condition Bits Affected: None

IM 1

Operation: —

Format:

Opcode	Operands
IM	1



Description:

The IM instruction sets interrupt mode 1. In this mode the processor will respond to an interrupt by executing a restart to location 0038H.

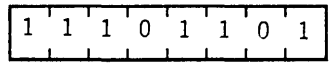
M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

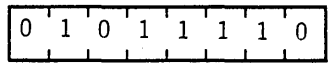
Operation: —

Format:

Opcode	Operands
IM	2



ED



5E

Description:

The IM 2 instruction sets the vectoreed interrupt mode 2. This mode allows an indirect call to any memory location by an 8 bit vector supplied from the peripheral device. This vector then becomes the least significant 8 bits of the indirect pointer while the I register in the CPU provides the most significant 8 bits. This address points to an addreess in a vector table which is the starting address for the interrupt service routine.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

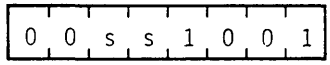
-16 BIT ARITHMETIC GROUP-

ADD HL, ss

Operation: HL ← HL+ss

Format:

Opcode	Operands
ADD	HL,ss



Description:

The contents of register pair ss (any of register pairs BC,DE,HL or SP) are added to the contents of register pair HL and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M CYCLES: 3 T STATES: 11(4,4,3) 4 MHZ E.T.: 2.75

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Set if carry out of Bit 11; reset otherwise
P/V:	Not affected
N:	Reset
C:	Set if carry from Bit 15; reset otherwise

Example:

If register pair HL contains the integer 4242H and register pair DE contains 1111H, after the execution of

```
ADD HL,DE
```

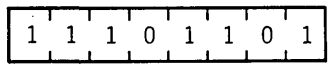
the HL register pair will contain 5353H.

ADC HL, ss

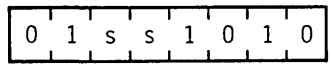
Operation: HL←HL+ss+CY

Format:

Opcode	Operands
ADC	HL,ss



ED



Description:

The contents of register pair ss (any of register pairs BC,DE,HL or SP) are added with the Carry Flag (C flag in the F register) to the contents of register pair HL, and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if carry out of Bit 11; reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Reset
C:	Set if carry from Bit 15; reset otherwise

Example:

If the register pair BC contains 2222H, register pair HL contains 5437H and the Carry Flag is set, after the execution of

```
ADC HL,BC
```

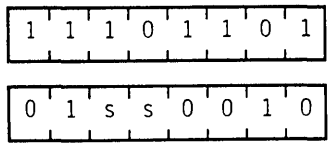
the contents of HL will be 765AH.

SBC HL, ss

Operation: **HL←HL-ss-CY**

Format:

Opcode	Operands
SBC	HL,ss



Description:

The contents of the register pair ss (any of register pairs BC,DE,HL or SP) and the Carry Flag (C flag in the F register) are subtracted from the contents of register pair HL and the result is stored in HL. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Set if a borrow from Bit 12;reset otherwise
P/V:	Set if overflow; reset otherwise
N:	Set
C:	Set if borrow; reset otherwise

Example:

If the contents of the HL register pair are 9999H, the contents of register pair DE are 1111H, and the Carry Flag is set, after the execution of

```
SBC HL,DE
```

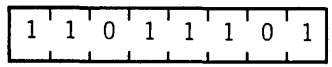
the contents of HL will be 8887H.

ADD IX, pp

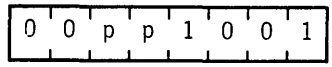
Operation: IX ← IX + pp

Format:

Opcode	Operands
ADD	IX,pp



DD



Description:

The contents of register pair pp (any of register pairs BC,DE,IX or SP) are added to the contents of the Index Register IX, and the results are stored in IX. Operand pp is specified as follows in the assembled object code.

Register Pair	pp
BC	00
DE	01
IX	10
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Set if carry out of Bit 11; reset otherwise
P/V:	Not affected
N:	Reset
C:	Set if carry from Bit 15; reset otherwise

Example:

If the contents of Index Register IX are 333H and the contents of register pair BC are 5555H, after the execution of

```
ADD IX,BC
```

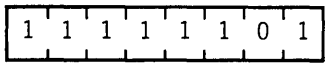
the contents of IX will be 8888H.

ADD IY, rr

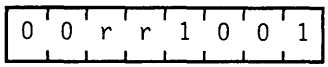
Operation: IY←IY+rr

Format:

Opcode	Operands
ADD	IY,rr



FD



Description:

The contents of register pair rr (any of register pairs BC,DE,IY or SP) are added to the contents of Index Register IY, and the result is stored in IY. Operand rr is specified as follows in the assembled object code.

Register Pair	rr
BC	00
DE	01
IY	10
SP	11

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Set if carry out of Bit 11; reset otherwise
P/V:	Not affected
N:	Reset
C:	Set if carry from Bit 15; reset otherwise

Example:

If the contents of Index Register IY are 333H and the contents of register pair BC are 555H, after the execution of

```
ADD IY,BC
```

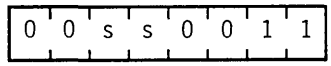
the contents of IY will be 8888H.

INC ss

Operation: ss ← ss + 1

Format:

Opcodes	Operands
INC	ss



Description:

The contents of register pair ss (any of register pairs BC, DE,HL or SP) are incremented. Operand ss is specified as follows in the assembled object code.

Register Pair	ss
BC	00
DE	01
HL	10
SP	11

M CYCLES: 1 T STATES: 6 4 MHZ E.T. 1.50

Condition Bits Affected: None

Example:

If the register pair contains 1000H, after the execution of

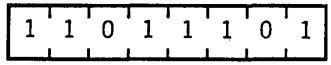
```
INC HL
```

HL will contain 1001H.

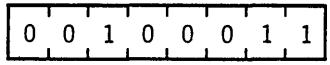
Operation: IX ← IX + 1

Format:

Opcode	Operands
INC	IX



DD



23

Description:

The contents of the Index Register IX are incremented.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the Index Register IX contains the integer 3300H after the execution of

```
INC IX
```

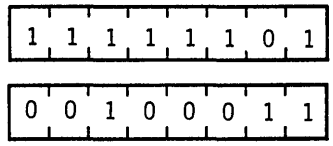
the contents of Index Register IX will be 3301H.

INC IY

Operation: IY ← IY + 1

Format:

Opcode	Operands
INC	IY



Description:

The contents of the Index Register IY are incremented.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Index Register are 2977H, after the execution of

```
INC IY
```

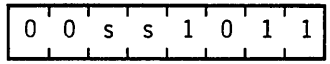
the contents of Index Register IY will be 2978H.

DEC ss

Operation: ss ← ss -1

Format:

Opcode	Operands
DEC	ss



Description:

The contents of register pair ss (any of the register pairs BC,DE,HL or SP) are decremented. Operand ss is specified as follows in the assembled object code.

Pair	ss
BC	00
DE	01
HL	10
SP	11

M CYCLES: 1 T STATES: 6 4 MHZ E.T.: 1.50

Condition Bits Affected: None

Example:

If register pair HL contains 1001H, after the execution of

```
DEC HL
```

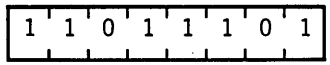
the contents of HL will be 1000H.

DEC IX

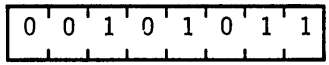
Operation: IX ← IX -1

Format:

Opcode	Operands
DEC	IX



DD



2B

Description:

The contents of Index Register IX are decremented.

M CYCLES: 2 T STATES: 10(4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of Index Register IX are 2006H, after the execution of

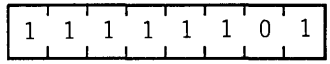
```
DEC IX
```

the contents of Index Register IX will be 2005H.

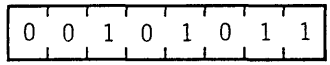
Operation: IY ← IY -1

Format:

Opcode	Operands
DEC	IY



FD



2B

Description:

The contents of the Index Register IY are decremented.

M CYCLES: 2 T STATES: 10 (4,6) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Index Register IY are 7649H, after the execution of

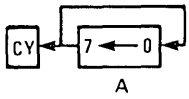
```
DEC IY
```

the contents of Index Register IY will be 7648H.

-ROTATE AND SHIFT GROUP-

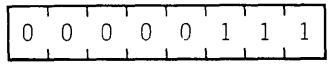
RLCA

Operation:



Format:

Opcode	Operands
RLCA	



07

Description:

The contents of the Accumulator (register A) are rotated left one bit position. The sign bit (bit 7) is copied into the Carry Flag and also into bit 0. Bit 0 is the least significant bit.

M CYCLES: 1 T STATES 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Not affected
N:	Reset
C:	Data from Bit 7 of Acc.

Example:

If the contents of the Accumulator are

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

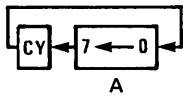
after the execution of

RLCA

the contents of the Accumulator and Carry Flag will be

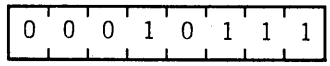
C	7	6	5	4	3	2	1	0
1	0	0	0	1	0	0	0	1

RLA

Operation: 

Format:

Opcode	Operands
RLA	



17

Description:

The contents of the Accumulator (register A) are rotated left one bit position through the Carry Flag. The previous content of the Carry Flag is copied into bit 0. Bit 0 is the least significant bit.

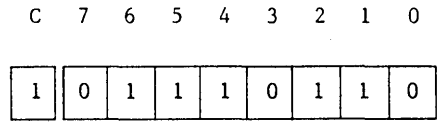
M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Not affected
N:	Reset
C:	Data from Bit 7 of Acc.

Example:

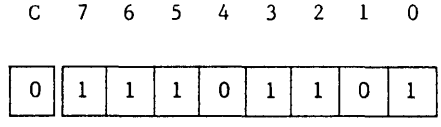
If the contents of the Accumulator and the Carry Flag are



after the execution of

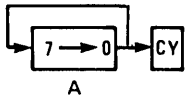
```
RLA
```

the contents of the Accumulator and the Carry Flag will be



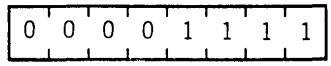
RRCA

Operation:



Format:

Opcode	Operands
RRCA	



0F

Description:

The contents of the Accumulator (register A) are rotated right one bit position. Bit 0 is copied into the Carry Flag and also into bit 7. Bit 0 is the least significant bit.

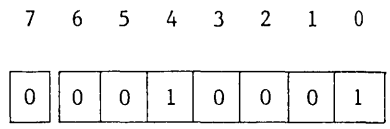
M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S: Not affected
Z: Not affected
H: Reset
P/V: Not affected
N: Reset
C: Data from Bit 0 of Acc.

Example:

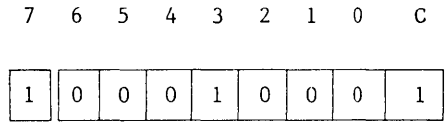
If the contents of the Accumulator are



After the execution of

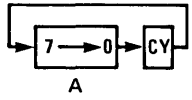
RRCA

the contents of the Accumulator and the Carry Flag will be



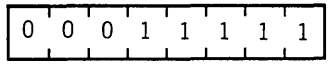
RRA

Operation:



Format:

Opcode	Operands
RRA	



1F

Description:

The contents of the Accumulator (register A) are rotated right one bit position through the Carry Flag. The previous content of the Carry Flag is copied into bit 7. Bit 0 is the least significant bit.

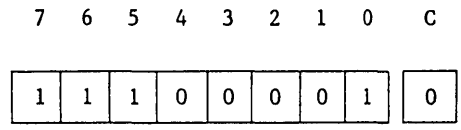
M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected:

S:	Not affected
Z:	Not affected
H:	Reset
P/V:	Not affected
N:	Reset
C:	Data from Bit 0 of Acc.

Example:

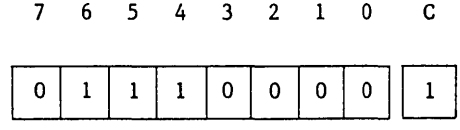
If the contents of the Accumulator and the Carry Flag are



after the execution of

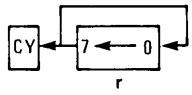
RRA

the contents of the Accumulator and the Carry Flag will be



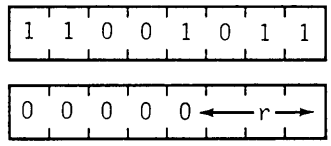
RLC r

Operation:



Format:

Opcode	Operands
RLC	r



Description:

The contents of register r are rotated left one bit position. The content of bit 7 is copied into the Carry Flag and also into bit 0. Operand r is specified as follows in the assembled object code:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

If the contents of register r are

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

after the execution of

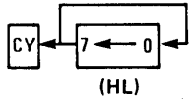
```
RLC r
```

the contents of register r and the Carry Flag will be

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

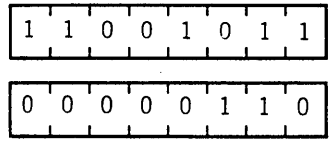
RLC (HL)

Operation:



Format:

Opcode	Operands
RLC	(HL)



Description:

The contents of the memory address specified by the contents of register pair HL are rotated left one bit position. The content of bit 7 is copied into the Carry Flag and also into bit 0. Bit 0 is the least significant bit.

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

If the contents of the HL register pair are 2828H, and the contents of memory location 2828H are

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

after the execution of

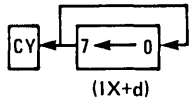
RLC (HL)

the contents of memory location 2828H and the Carry Flag will be

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

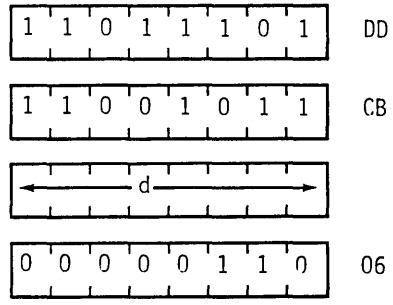
RLC (IX+d)

Operation:



Format:

Opcode	Operands
RLC	(IX+d)



Description:

The contents of the memory address specified by the sum of the contents of the Index Register IX and a two's complement displacement integer d, are rotated left one bit position. The content of bit 7 is copied into the Carry Flag and also into bit 0. Bit 0 is the least significant bit.

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

If the contents of the Index Register IX are 1000H, and the contents of memory location 1022H are

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

after the execution of

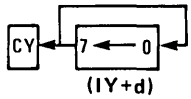
RLC (IX+2H)

the contents of memory location 1002H and the Carry Flag will be

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

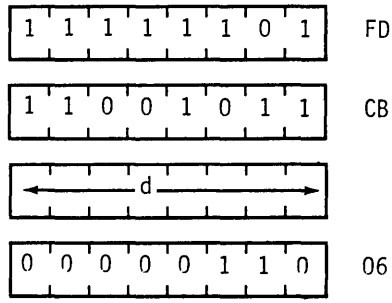
RLC (IY+d)

Operation:



Format:

Opcode	Operands
RLC	(IY+d)



Description:

The contents of the memory address specified by the sum of the contents of the Index Register IY and a two's complement displacement integer d are rotated left one bit position. The content of bit 7 is copied into the Carry Flag and also into bit 0. Bit 0 is the least significant bit.

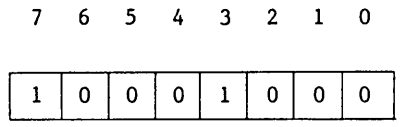
M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

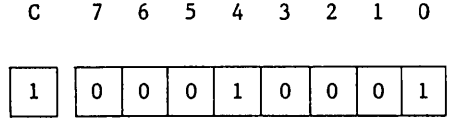
If the contents of the Index Register IY are 1000H, and the contents of memory location 1002H are



after the execution of

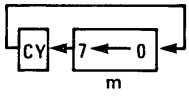
RLC (IY+2H)

the contents of memory location 1002H and the Carry Flag will be



RL m

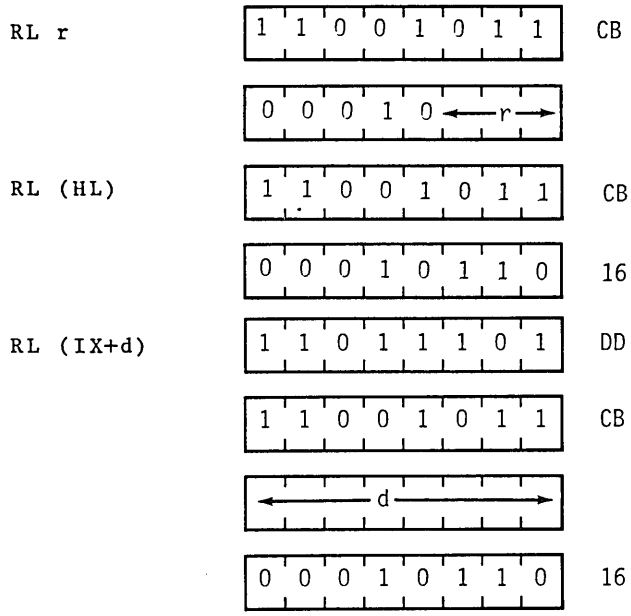
Operation:



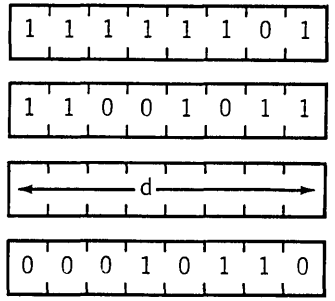
Format:

Opcode	Operands
RL	m

The m operand is any of r,(HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:



RL (IY+d)



*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code above:

Register	r
B	000
C	001
D	010
E	011
H	011
L	101
A	111

Description:

The contents of the m operand are rotated left one bit position. The content of bit 7 is copied into the Carry Flag and the previous content of the Carry Flag is copied into bit 0.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RL r	2	8(4,4)	2.00
RL (HL)	4	15(4,4,4,3)	3.75
RL (IX+d)	6	23(4,4,3,5,4,3)	5.75
RL (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 7 of source register

Example:

If the contents of register D and the Carry Flag are

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

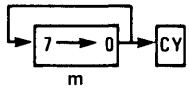
after the execution of

```
RL D
```

the contents of register D and the Carry Flag will be

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

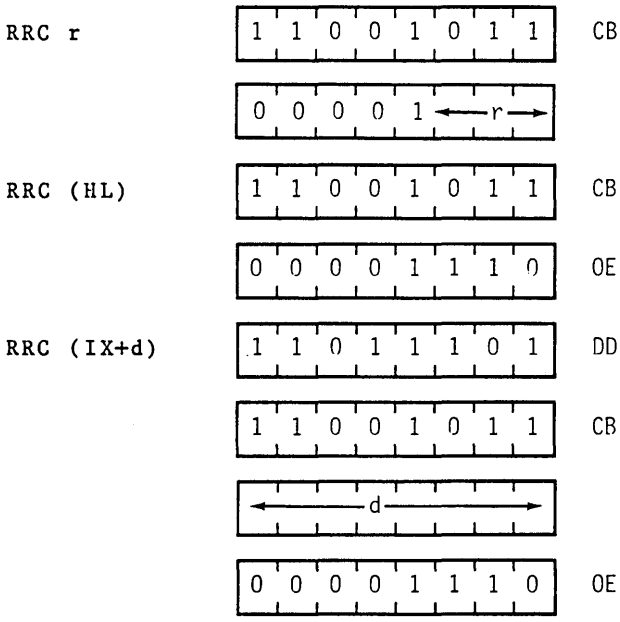
Operation:



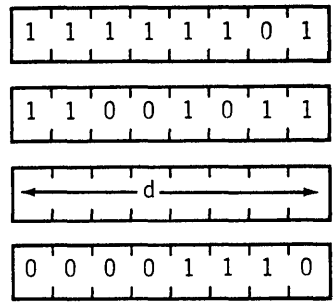
Format:

Opcode	Operands
RRC	m

The m operand is any of r,(HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:



RRC (IY+d)



*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of operand m are rotated right one bit position. The content of bit 0 is copied into the Carry Flag and also into bit 7. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RRC r	2	8(4,4)	2.00
RRC (HL)	4	15(4,4,4,3)	3.75
RRC (IX+d)	6	23(4,4,3,5,4,3)	5.75
RRC (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity even; reset otherwise
N:	Reset
C:	Data from Bit 0 of source register

Example:

If the contents of register A are

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

after the execution of

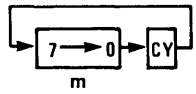
```
RRC A
```

the contents of register A and the Carry Flag will be

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

RR m

Operation:

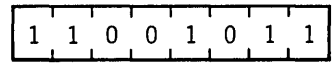


Format:

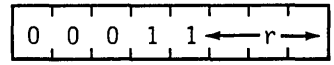
Opcode	Operand
RR	m

The m operand is any of r, (HL), (IX+d), or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

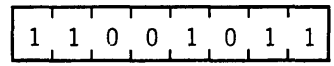
RR r



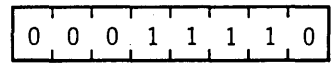
CB



RR (HL)

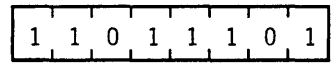


CB

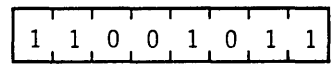


1E

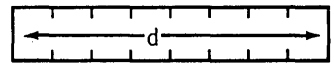
RR (IX+d)

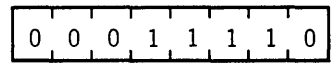


DD



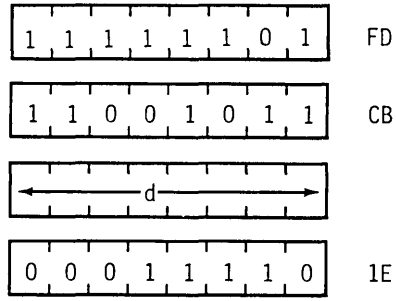
CB





1E

RR (IY+d)



*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of operand m are rotated right one bit position through the Carry flag. The content of bit 0 is copied into the Carry Flag and the previous content of the Carry Flag is copied into bit 7. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RR r	2	8(4,4)	2.00
RR (HL)	4	15(4,4,4,3)	3.75
RR (IX+d)	6	23(4,4,3,5,4,3)	5.75
RR (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Data from Bit 0 of source register

Example:

If the contents of the HL register pair are 4343H, and the contents of memory location 4343H and the Carry Flag are

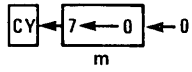
7	6	5	4	3	2	1	0	C
1	1	0	1	1	1	0	1	0

after the execution of

RR (HL)

the contents of location 4343H and the Carry Flag will be

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

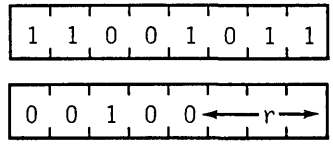
Operation: 

Format:

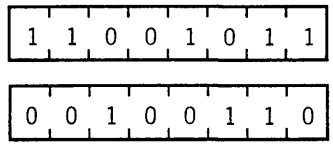
Opcode	Operands
SLA	m

The m operand is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

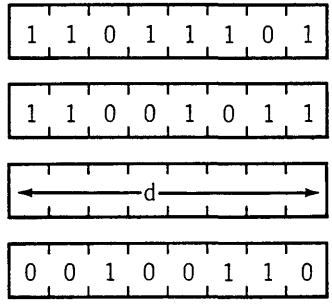
SLA r



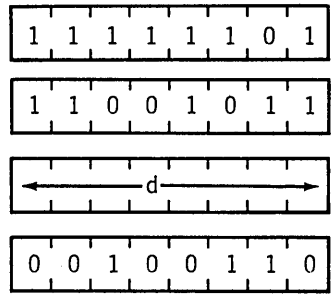
SLA (HL)



SLA (IX+d)



SLA (IY+d)



*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

An arithmetic shift left one bit position is performed on the contents of operand m. The content of bit 7 is copied into the Carry Flag. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SLA r	2	8(4,4)	2.00
SLA (HL)	4	15(4,4,4,3)	3.75
SLA (IX+d)	6	23(4,4,3,5,4,3)	5.75
SLA (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Data from Bit 7

Example:

If the contents of register L are

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

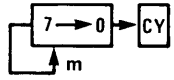
after the execution of

```
SLA L
```

the contents of register L and the Carry Flag will be

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

SRA m

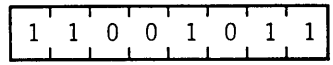
Operation: 

Format:

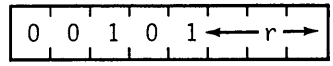
Opcode	Operands
SRA	m

The m operand is any of r, (HL), (IX+d) or (IY+d),as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

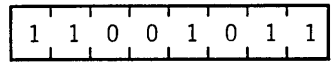
SRA r



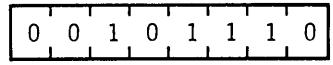
CB



SRA(HL)

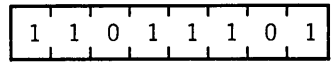


CB

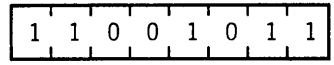


2E

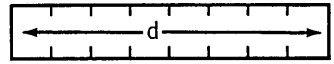
SRA(IX+d)

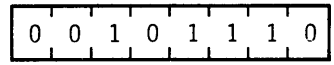


DD



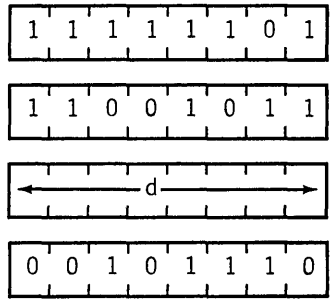
CB





2E

SRA(IY+d)



*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code field above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

An arithmetic shift right one bit position is performed on the contents of operand m. The content of bit 0 is copied into the Carry Flag and the previous content of bit 7 is unchanged. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SRA r	2	8(4,4)	2.00
SRA (HL)	4	15(4,4,4,3)	3.75
SRA (IX+d)	6	23(4,4,3,5,4,3)	5.75
SRA (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Set if result is negative; reset otherwise
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Data from Bit 0 of source register

Example:

If the contents of the Index Register IX are 1000H, and the contents of memory location 1003H are

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

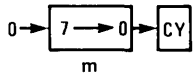
after the execution of

SRA (IX+3H)

the contents of memory location 1003H and the Carry Flag will be

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

Operation:

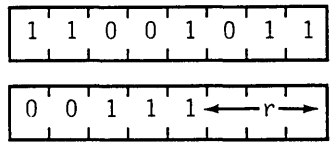


Format:

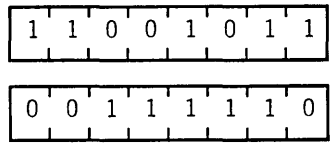
Opcode	Operands
SRL	m

The operand m is any of r, (HL), (IX+d) or (IY+d), as defined for the analogous RLC instructions. These various possible opcode-operand combinations are specified as follows in the assembled object code:

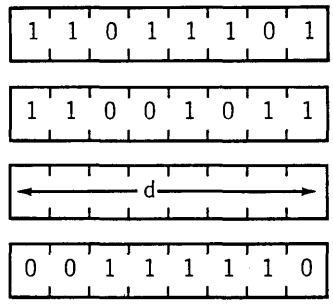
SRL r



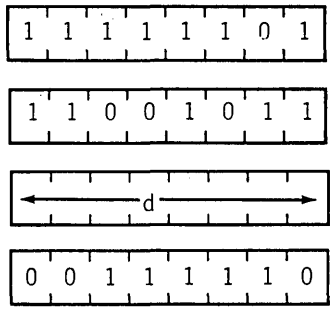
SRL (HL)



SRL (IX+d)



SRL (IY+d)



*r identifies registers B,C,D,E,H,L or A specified as follows in the assembled object code fields above:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

Description:

The contents of operand m are shifted right one bit position. The content of bit 0 is copied into the Carry Flag, and bit 7 is reset. Bit 0 is the least significant bit.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
SRL r	2	8(4,4)	2.00
SRL (HL)	4	15(4,4,4,3)	3.75
SRL (IX+d)	6	23(4,4,3,5,4,3)	5.75
SRL (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected:

S:	Reset
Z:	Set if result is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Data from Bit 0 of source register

Example:

If the contents of register B are

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

after the execution of

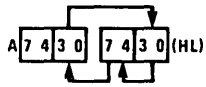
```
SRL B
```

the contents of register B and the Carry Flag will be

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

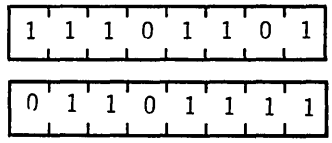
RLD

Operation:



Format:

Opcode	Operands
RLD	



Description:

The contents of the low order four bits (bits 3,2,1 and 0) of the memory location (HL) are copied into the high order four bits (7,6,5 and 4) of that same memory location; the previous contents of those high order four bits are copied into the low order four bits of the Accumulator (register A); and the previous contents of the low order four bits of the Accumulator are copied into the low order four bits of memory location (HL). The contents of the high order bits of the Accumulator are unaffected. Note: (HL) means the memory location specified by the contents of the HL register pair.

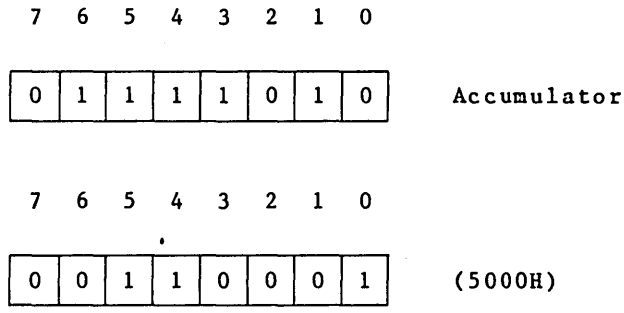
M CYCLES: 5 T STATES: 18(4,4,3,4,3) 4 MHZ E.T.: 4.50

Condition Bits Affected:

S:	Set if Acc. is negative after operation; reset otherwise
Z:	Set if Acc. is zero after operation; reset otherwise
H:	Reset
P/V:	Set if parity of Acc. is even after operation; reset otherwise
N:	Reset
C:	Not affected

Example:

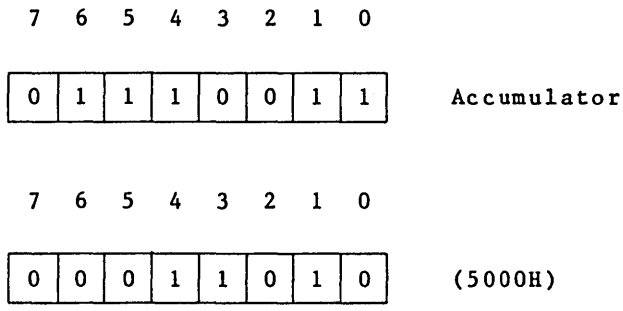
If the contents of the HL register pair are 5000H, and the contents of the Accumulator and memory location 5000H are



after the execution of

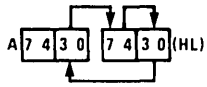
RLD

the contents of the Accumulator and memory location 5000H will be



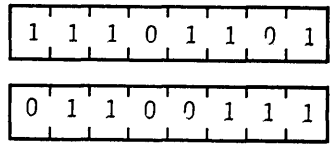
RRD

Operation:



Format:

Opcode	Operands
RRD	



Description:

The contents of the low order four bits (bits 3,2,1 and 0) of memory location (HL) are copied into the low order four bits of the Accumulator (register A); the previous contents of the low order four bits of the Accumulator are copied into the high order four bits (7,6,5 and 4) of location (HL); and the previous contents of the high order four bits of (HL) are copied into the low order four bits of (HL). The contents of the high order bits of the Accumulator are unaffected. Note: (HL) means the memory location specified by the contents of the HL register pair.

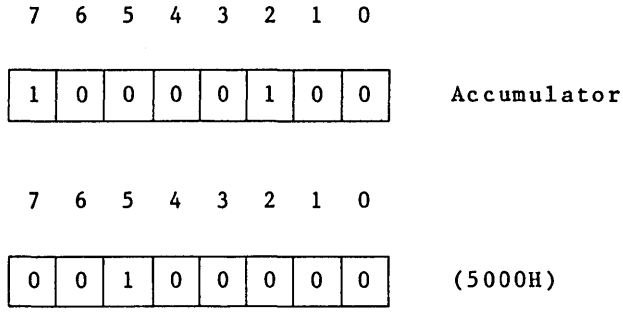
M CYCLES: 5 T STATES: 18(4,4,3,4,3) 4 MHZ E.T.: 4.50

Condition Bits Affected:

S:	Set if Acc. is negative after operation; reset otherwise
Z:	Set if Acc. is zero after operation; reset otherwise
H:	Reset
P/V:	Set if parity of Acc. is even after operation; reset otherwise
N:	Reset
C:	Not affected

Example:

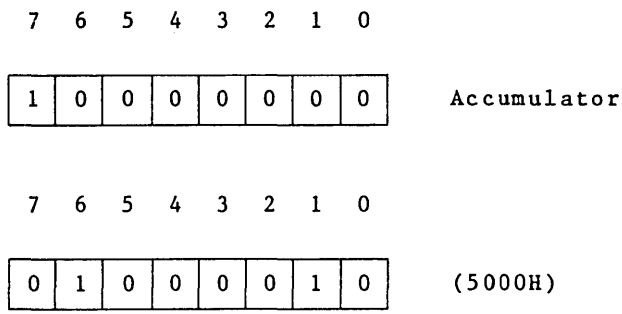
If the contents of the HL register pair are 5000H, and the contents of the Accumulator and memory location 5000H are



after the execution of

RRD

the contents of the Accumulator and memory location 5000H will be



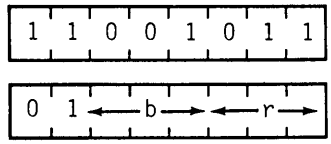
-BIT SET, RESET AND TEST GROUP-

BIT b, r

Operation: $Z \leftarrow \overline{r}_b$

Format:

Opcode	Operands
BIT	b,r



Description:

This instruction tests Bit b in register r and sets the Z flag accordingly. Operands b and r are specified as follows in the assembled object code:

Bit Tested	b	Register	r
0	000	B	000
1	001	C	001
2	010	D	010
3	011	E	011
4	100	H	100
5	101	L	101
6	110	A	111
7	111		

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected:

S:	Unknown
Z:	Set if specified Bit is 0; reset otherwise
H:	Set
P/V:	Unknown
N:	Reset
C:	Not affected

Example:

If bit 2 in register B contains 0, after the execution of

```
BIT 2,B
```

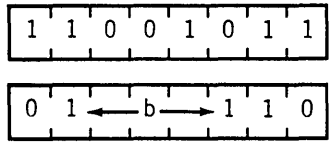
the Z flag in the F register will contain 1, and bit 2 in register B will remain 0. Bit 0 in register B is the least significant bit.

BIT b, (HL)

Operation: $Z \leftarrow \overline{(HL)}_b$

Format:

Opcode	Operands
BIT	b,(HL)



Description:

This instruction tests bit b in the memory location specified by the contents of the HL register pair and sets the Z flag accordingly. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 3 T STATES: 12(4,4,4) 4 MHZ E.T.: 3.00

Condition Bits Affected:

S:	Unknown
Z:	Set if specified Bit is 0; reset otherwise
H:	Set
P/V:	Unknown
H:	Reset
C:	Not affected

Example:

If the HL register pair contains 4444H, and bit 4 in the memory location 444H contains 1, after the execution of

```
BIT 4, (HL)
```

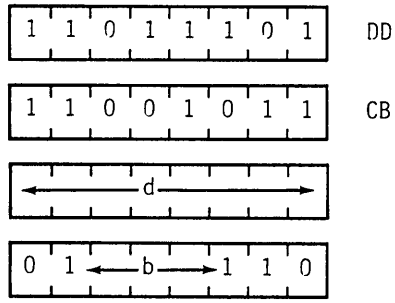
the Z flag in the F register will contain 0, and bit 4 in memory location 4444H will still contain 1. (Bit 0 in memory location 4444H is the least significant bit.)

BIT b, (IX+d)

Operation: $Z \leftarrow \overline{(IX+d)}_b$

Format:

Opcode	Operands
BIT	b,(IX+d)



Description:

This instruction tests bit b in the memory location specified by the contents of register pair IX combined with the two's complement displacement d and sets the Z flag accordingly. Operand b is specified as follows in the assembled object code.

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 5 T STATES: 20(4,4,3,5,4) 4 MHZ E.T.: 5.00

Condition Bits Affected:

S: Unknown
Z: Set if specified Bit is 0; reset otherwise

Condition Bits Affected:

S:	Unknown
Z:	Set if specified Bit is 0; reset otherwise
H:	Set
P/V:	Unknown
N:	Reset
C:	Not affected

Example:

If the contents of Index Register IX are 2000H, and bit 6 in memory location 2004H contains 1, after the execution of

```
BIT 6, (IX+4H)
```

the Z flag in the F register will contain 0, and bit 6 in memory location 2004H will still contain 1. (Bit 0 in memory location 2004H is the least significant bit.)

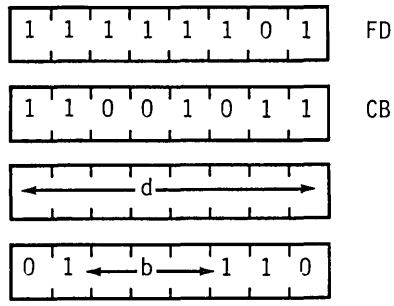
BIT b, (IY+d)

BIT b,(IY+d)

Operation: $Z \leftarrow \overline{(IY+d)_b}$

Format:

Opcode	Operands
BIT	b,(IY+d)



Description:

This instruction tests bit b in the memory location specified by the contents of register pair IY combined with the two's complement displacement d and sets the Z flag accordingly. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 5 T STATES: 20(4,4,3,5,4) 4 MHZ E.T.: 5.00

Condition Bits Affected:

S:	Unknown
Z:	Set if specified Bit is 0; reset otherwise
H:	Set
P/V:	Unknown
N:	Reset
C:	Not affected

Example:

If the contents of Index Register are 2000H, and bit 6 in memory location 2004H contains 1, after the execution of

```
BIT 6, (IY+4H)
```

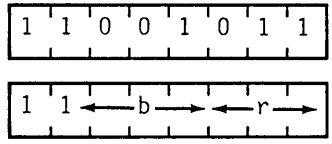
the Z flag in the F register sill contain 0, and bit 6 in memory location 2004H will still contain 1. (Bit 0 in memory location 2004H is the least significant bit.)

SET b, r

Operation: $r_b \leftarrow 1$

Format:

Opcode	Operands
SET	b,r



Description:

Bit b in register r (any of registers B,C,D,E,H,L or A) is set. Operands b and r are specified as follows in the assembled object code:

Bit	b	Register	r
0	000	B	000
1	001	C	001
2	010	D	010
3	011	E	011
4	100	H	100
5	101	L	101
6	110	A	111
7	111		

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

Example:

After the execution of

```
SET 4,A
```

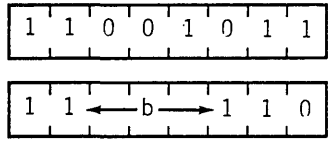
bit 4 in register A will be set. (Bit 0 is the least significant bit.)

SET b, (HL)

Operation: $(HL)_b \leftarrow 1$

Format:

Opcode	Operands
SET	b,(HL)



Description:

Bit b in the memory location addressed by the contents of register pair HL is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 4 T STATES: 15(4,4,4,3) 4 MHZ E.T.: 3.75

Condition Bits Affected: None

Example:

If the contents of the HL register pair are 3000H, after the execution of

```
SET 4,(HL)
```

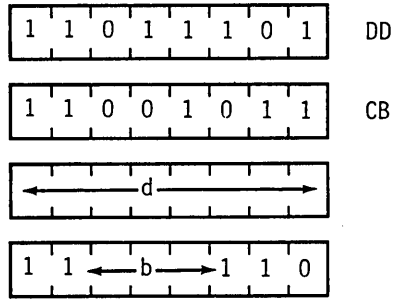
bit 4 in memory location 3000H will be 1. (Bit 0 in memory location 3000H is the least significant bit.)

SET b, (IX+d)

Operation: $(IX+d)_b \leftarrow 1$

Format:

Opcode	Operands
SET	b,(IX+d)



Description:

Bit b in the memory location addressed by the sum of the contents of the IX register pair and the two's complement integer d is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the contents of Index Register are 2000H, after the execution of

```
SET 0,(IX+3H)
```

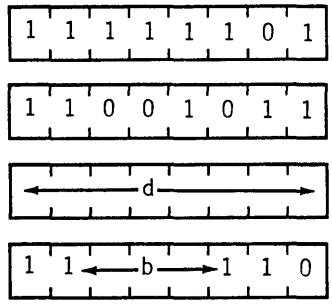
bit 0 in memory location 2003H will be 1. (Bit 0 in memory location 2003H is the least significant bit.)

SET b, (IY+d)

Operation: $(IY+d)_b \leftarrow 1$

Format:

Opcode	Operands
SET	b,(IY+d)



Description:

Bit b in the memory location addressed by the sum of the contents of the IY register pair and the two's complement displacement d is set. Operand b is specified as follows in the assembled object code:

Bit Tested	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

M CYCLES: 6 T STATES: 23(4,4,3,5,4,3) 4 MHZ E.T.: 5.75

Condition Bits Affected: None

Example:

If the contents of Index Register IY are 2000H, after

the execution of

```
SET 0,(IY+3H)
```

bit 0 in memory location 2003H will be 1. (Bit 0 in memory location 2003H is the least significant bit.)

RES b, m

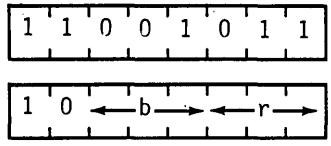
Operation: $s_b \leftarrow 0$

Format:

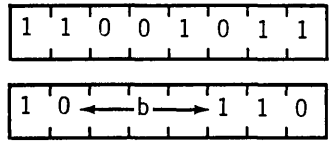
Opcode	Operands
RES	b,m

Operand b is any bit (7 through 0) of the contents of the m operand, (any of r, (HL), (IX+d) or (IY+d)) as defined for the analogous SET instructions. These various possible opcode-operand combinations are assembled as follows in the object code:

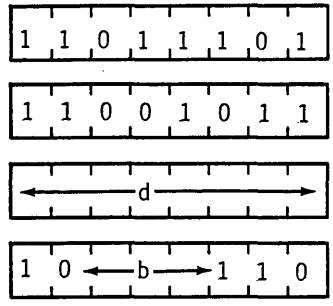
RES b,r



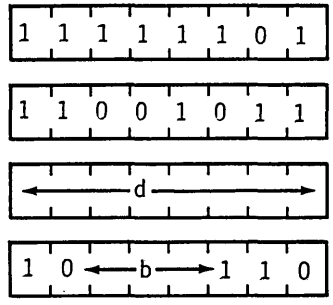
RES b,(HL)



RES b,(IX+d)



RES b,(IY+d)



Bit Reset	b	Register	r
0	000	B	000
1	001	C	001
2	010	D	010
3	011	E	011
4	100	H	100
5	101	L	101
6	110	A	111
7	111		

Description:

Bit b in operand m is reset.

INSTRUCTION	M CYCLES	T STATES	4 MHZ E.T.
RES r	4	8(4,4)	2.00
RES (HL)	4	15(4,4,4,3)	3.75
RES (IX+d)	6	23(4,4,3,5,4,3)	5.75
RES (IY+d)	6	23(4,4,3,5,4,3)	5.75

Condition Bits Affected: None

Example:

After the execution of

```
RES 6,D
```

bit 6 in register D will be reset. (Bit 0 in register D is the least significant bit.)

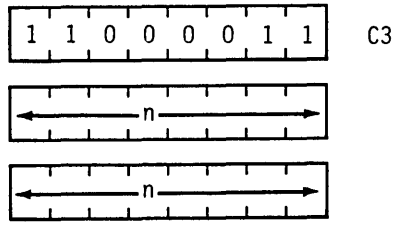
-JUMP GROUP-

JP nn

Operation: PC ← nn

Format:

Opcode	Operands
JP	nn



Note: The first operand in this assembled object code is the low order byte of a 2-byte address.

Description:

Operand nn is loaded into register pair PC (Program Counter). The next instruction is fetched from the location designated by the new contents of the PC.

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

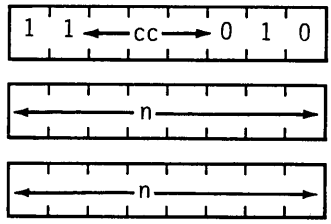
Condition Bits Affected: None

JP cc, nn

Operation: IF cc TRUE, PC ← nn

Format:

Opcode	Operands
JP	cc,nn



Note: The first n operand in this assembled object code is the low order byte of a 2-byte memory address.

Description:

If condition cc is true, the instruction loads operand nn into register pair PC (Program Counter), and the program continues with the instruction beginning at address nn. If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status which corresponds to condition bits in the Flag Register (register F). These eight status are defined in the table below which also specifies the corresponding cc bit fields in the assembled object code.

cc	CONDITION	RELEVANT FLAG
000	NZ non zero	Z
001	Z zero	Z
010	NC no carry	C
011	C carry	C
100	PO parity odd	P/V
101	PE parity even	P/V
110	P sign positive	S
111	M sign negative	S

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the Carry Flag (C flag in the F register) is set and the contents of address 1520 are 03H, after the execution of

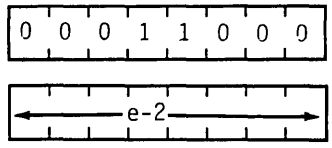
```
JP C,1520H
```

the Program Counter will contain 1520H, and on the next machine cycle the CPU will fetch from address 1520H the byte 03H.

Operation: PC ← PC + e

Format:

Opcode	Operand
JR	e



Description:

This instruction provides for unconditional branching to other segments of a program. The value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. This jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.00

Condition Bits Affected: None

Example:

To jump forward 5 locations from address 480, the following assembly language statement is used:

```
JR $+5
```

The resulting object code and final PC value is shown below:

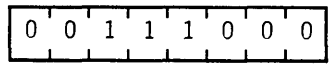
Location	Instruction	
480	18	
481	03	
482	—	
483	—	
484	—	
485	←	PC after jump

JR C, e

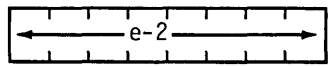
Operation: If C = 0, continue
If C = 1, PC ← PC + e

Format:

Opcode	Operands
JR	C,e



38



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to a '1', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the flag is equal to a '0', the next instruction to be executed is taken from the location following this instruction.

If condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.:3.00

If condition is not met:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Carry Flag is set and it is required to jump back 4 locations from 480. The assembly language statement is:

```
JR C,$-4
```

The resulting object code and final PC value is shown below:

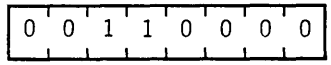
Location	Instruction
47C	← PC after jump
47D	—
47E	—
47F	—
480	38
481	FA (2's complement-6)

JR NC, e

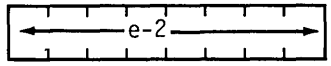
Operation: If C = 1, continue
If C = 0, PC ← PC + e

Format:

Opcode	Operands
JR	NC,e



30



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Carry Flag. If the flag is equal to '0', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the flag is equal to a '1', the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.00

If the condition is not met:

M CYCLES: 7 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Carry Flag is reset and it is required to repeat the jump instruction. The assembly language statement is:

```
JR NC,$
```

The resulting object code and PC after the jump are shown below:

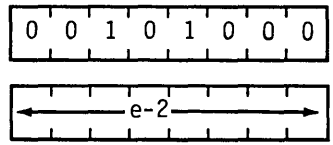
Location	Instruction
480	30 ← PC after jump
481	00

JR Z, e

Operation: If Z = 0, continue
If Z = 1, PC ← PC + e

Format:

Opcode	Operands
JR	Z,e



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Zero Flag. If the flag is equal to a '1', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the Zero Flag is equal to a '0', the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.00

If the condition is not net:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Zero Flag is set and it is required to jump forward 5 locations from address 300. The following assembly language statement is used:

```
JR Z,$ +5
```

The resulting object code and final PC value is shown below:

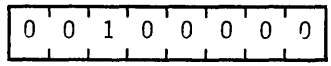
Location	Instruction	
300	28	
301	03	
302	—	
303	—	
304	—	
305	←	PC after jump

JR NZ, e

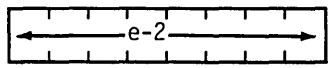
Operation: If Z = 1, continue
If Z = 0, PC ← PC + e

Format:

Opcode	Operands
JR	NZ,e



20



Description:

This instruction provides for conditional branching to other segments of a program depending on the results of a test on the Zero Flag. If the flag is equal to a '0', the value of the displacement e is added to the Program Counter (PC) and the next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the Zero Flag is equal to a '1', the next instruction to be executed is taken from the location following this instruction.

If the condition is met:

M CYCLES: 3 T STATES: 12(4,3,5) 4 MHZ E.T.: 3.00

If the condition is not met:

M CYCLES: 2 T STATES: 7(4,3) 4 MHZ E.T.: 1.75

Condition Bits Affected: None

Example:

The Zero Flag is reset and it is required to jump back 4 locations from 480. The assembly language statement is:

```
JR NZ,$-4
```

The resulting object code and final PC value is shown below:

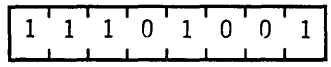
Location	Instruction	
47C	←	PC after jump
47D	—	
47E	—	
47F	—	
480	20	
481	FA	(2' complement-6)

JP (HL)

Operation: PC ← HL

Format:

Opcode	Operands
JP	(HL)



E9

Description:

The Program Counter (register pair PC) is loaded with the contents of the HL register pair. The next instruction is fetched from the location designated by the new contents of the PC.

M CYCLES: 1 T STATES: 4 4 MHZ E.T.: 1.00

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H and the contents of the HL register pair are 4800H, after the execution of

JP (HL)

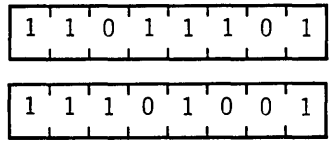
the contents of the Program Counter will be 4800H.

JP (IX)

Operation: PC ← IX

Format:

Opcode	Operands
JP	(IX)



Description:

The Program Counter (register pair PC) is loaded with the contents of the IX Register Pair. The next instruction is fetched from the location designated by the new contents of the PC.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H, and the contents of the IX Register Pair are 4800H, after the execution of

```
JP (IX)
```

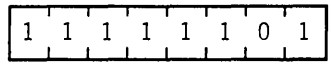
the contents of the Program Counter will be 4800H.

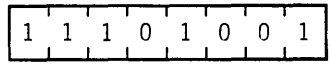
JP (IY)

Operation: PC ← IY

Format:

Opcode	Operands
JP	(IY)





Description:

The Program Counter (register pair PC) is loaded with the contents of the IY Register Pair. The next instruction is fetched from the location designated by the new contents of the PC.

M CYCLES: 2 T STATES: 8(4,4) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1000H and the contents of the IY Register Pair are 4800H, after the execution of

```
JP (IY)
```

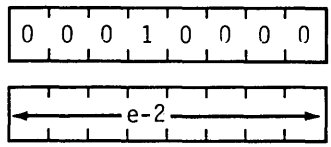
the contents of the Program Counter will be 4800H.

DJNZ, e

Operation: —

Format:

Opcode	Operand
DJNZ	e



Description:

This instruction is similar to the conditional jump instructions except that a register value is used to determine branching. The B register is decremented and if a non zero value remains, the value of the displacement e is added to the Program Counter (PC). The next instruction is fetched from the location designated by the new contents of the PC. The jump is measured from the address of the instruction opcode and has a range of -126 to +129 bytes. The assembler automatically adjusts for the twice incremented PC.

If the result of decrementing leaves B with a zero value, the next instruction to be executed is taken from the location following this instruction.

If B≠0:

M CYCLES: 3 T STATES: 13(5,3,5) 4 MHZ E.T.: 3.25

If B=0:

M CYCLES: 2 T STATES: 8(5,3) 4 MHZ E.T.: 2.00

Condition Bits Affected: None

Example:

A typical software routine is used to demonstrate the use of the DJNZ instruction. This routine moves a line from an input buffer (INBUF) to an output buffer

(OUTBUF). It moves the bytes until it finds a CR, or until it has moved 80 bytes, whichever occurs first.

```
        LD          B,80          ;Set up counter
        LD          HL,Inbuf      ;Set up pointers
        LD          DE,Outbuf

LOOP:   LD          A,(HL)        ;Get next byte from
                                  ;input buffer
        LD          (DE),A        ;Store in output buffer
        CP          0DH           ;Is it a CR?
        JR          Z,DONE        ;Yes finished
        INC         HL            ;Increment pointers
        INC         DE
        DJNZ        LOOP          ;Loop back if 80
                                  ;bytes have not
                                  ;been moved
DONE:
```

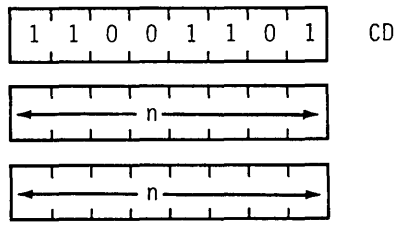
-CALL AND RETURN GROUP-

CALL nn

Operation: (SP-1) ← PC_H, (SP-2) ← PC_L, PC ← nn

Format:

Opcode	Operands
CALL	nn



Note: The first of the two n operands in the assembled object code above is the least significant byte of a two-byte memory address.

Description:

The current contents of the Program Counter (PC) are pushed onto the top of the external memory stack. The operands nn are then loaded into the PC to point to the address in memory where the first opcode of a subroutine is to be fetched. (At the end of the subroutine,a RETurn instruction can be used to return to the original program flow by popping the top of the stack back into the PC.) The push is accomplished by first decrementing the current contents of the Stack Pointer (register pair SP), loading the high-order byte of the PC contents into the memory address now pointed to by the SP; then decrementing SP again, and loading the low-order byte of the PC contents into the top of stack. Note: Because this is a 3-byte instruction, the Program Counter will have been incremented by 3 before the push is executed.

M CYCLES: 5 T STATES: 17(4,3,4,3,3) 4 MHZ E.T.: 4.25

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 1A47H, the contents of the Stack Pointer are 3002H, and memory locations have the contents:

Location	Contents
1A47H	CDH
1A48H	35H
1A49H	21H

then if an instruction fetch sequence begins, the three-byte instruction CD3521H will be fetched to the CPU for execution. The mnemonic equivalent of this is

```
CALL 2135H
```

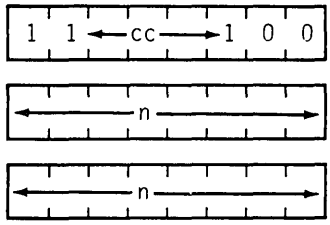
After the execution of this instruction, the contents of memory address 3001H will be 1AH, the contents of address 3000H will be 4AH, the contents of the Stack Pointer will be 3000H, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

CALL cc, nn

Operation: IF cc TRUE: $(SP\text{-}1) \leftarrow PC_H$
$(SP\text{-}2) \leftarrow PC_L, PC \leftarrow nn$

Format:

Opcode	Operands
CALL	cc,nn



Note: The first of the two n operands in the assembled object code above is the least significant byte of the two-byte memory address.

Description:

If condition cc is true, this instruction pushes the current contents of the Program Counter (PC) onto the top of the external memory stack, then loads the operands nn into PC to point to the address in memory where the first opcode of a subroutine is to be fetched. (At the end of the subroutine, a RETurn instruction can be used to return to the original program flow by popping the top of the stack back into PC.) If condition cc is false, the Program Counter is incremented as usual, and the program continues with the next sequential instruction. The stack push is accomplished by first decrementing the current contents of the Stack Pointer (SP), loading the high-order byte of the PC contents into the memory address now pointed to by SP; then decrementing SP again, and loading the low-order byte of the PC contents into the top of the stack. Note: Because this is a 3-byte instruction, the Program Counter will have been incremented by 3 before

the push is executed. Condition cc is programmed as one of eight status which corresponds to condition bits in the Flag Register (register F). These eight status are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code:

cc	Condition	Relevant Flag
000	NZ non zero	Z
001	Z zero	Z
010	NC non carry	C
011	C carry	C
100	PO parity odd	P/V
101	PE parity even	P/V
110	P sign positive	S
111	M sign negative	S

If cc is true:

M CYCLES: 5 T STATES: 17(4,3,4,3,3) 4 MHZ E.T.: 4.25

If cc is false:

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the C Flag in the F register is reset, the contents of the Program Counter are 1A47H, the contents of the Stack Pointer are 3002H, and memory locations have the contents:

Location	Contents
1A47H	D4H
1A48H	35H
1A49H	21H

then if an instruction fetch sequence begins, the three-byte instruction D43521H will be fetched to the CPU for execution. The mnemonic equivalent of this is

```
CALL NC,2135H
```

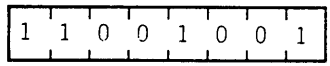
After the execution of this instruction, the contents of memory address 3001H will be 1AH, the contents of address 3000H will be 4AH, the contents of the Stack Pointer will be 3000H, and the contents of the Program Counter will be 2135H, pointing to the address of the first opcode of the subroutine now to be executed.

Operation: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$

Format:

Opcode

RET



C9

Description:

The byte at the memory location specified by the contents of the Stack Pointer (SP) register pair are moved to the low order eight bits of the Program Counter (PC). The SP is now incremented and the byte at the memory location specified by the new contents of the SP are moved to the high order eight bits of the PC. The SP is now incremented again. The next op code following this instruction will be fetched from the memory location specified by the PC. This instruction is normally used to return to the main line program at the completion of a routine entered by a CALL instruction.

M CYCLES: 3 T STATES: 10(4,3,3) 4 MHZ E.T.: 2.50

Condition Bits Affected: None

Example:

If the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2000H, the contents of memory location 2000H are B5H, and the contents of memory location 2001H are 18H, then after the execution of

RET

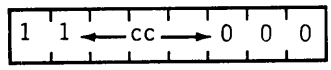
the contetns of the Stack Pointer will be 2002H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

RET cc

Operation: IF cc TRUE: $PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$

Format:

Opcode	Operand
RET	cc



Description:

If condition cc is true, the byte at the memory location specified by the contents of the Stack Pointer (SP) register pair are moved to the low order eight bits of the Program Counter (PC). The SP is now incremented and the byte at the memory location specified by the new contents of the SP are moved to the high order eight bits of the PC. The SP is now incremented again. The next op code following this instruction will be fetched from the memory location specified by the PC. This instruction is normally used to return to the main line program at the completion of a routine entered by a CALL instruction. If condition cc is false, the PC is simply incremented as usual, and the program continues with the next sequential instruction. Condition cc is programmed as one of eight status which correspond to condition bits in the Flag Register (register F). These eight status are defined in the table below, which also specifies the corresponding cc bit fields in the assembled object code.

cc	Condition	Relevant Flag
000	NZ non zero	Z
001	Z zero	Z
010	NC non carry	C
011	C carry	C
100	PO parity odd	P/V
101	PE parity even	P/V
110	P sign positive	S
111	M sign negative	S

If cc is true:

M CYCLES: 3 T STATES: 11(5,3,3) 4 MHZ E.T.: 2.75

If cc is false:

M CYCLES: 1 T STATES: 5 4 MHZ E.T.: 1.25

Condition Bits Affected: None

Example:

If the S flag in the F register is set, the contents of the Program Counter are 3535H, the contents of the Stack Pointer are 2000H, the contents of memory location 2000H are B5H, and the contents of memory location 2001H are 18H, then after the execution of

```
RET M
```

the contents of the Stack Pointer will be 2002H and the contents of the Program Counter will be 18B5H, pointing to the address of the next program opcode to be fetched.

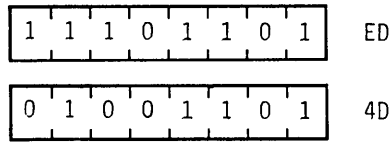
RETI

Operation: **Return from interrupt**

Format:

Opcode

RETI



Description:

This instruction is used at the end of a maskable interrupt service routine to:

1. Restore the contents of the Program Counter (PC) (analogous to the RET instruction)

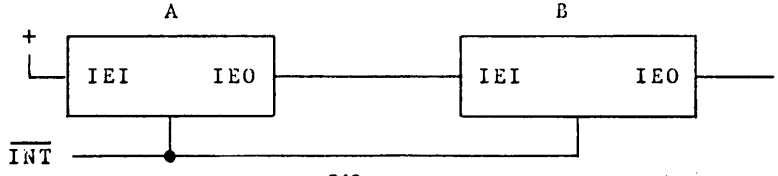
2. To signal an I/O device that the interrupt routine has been completed. The RETI instruction also facilitates the nesting of interrupts allowing higher priority devicess to temporarily suspend service of lower priority service routines. Note: This instruction does not enable interrupts which were disabled when the interrupt routine was entered. Before doing the RETI instruction, the enable interrupt instruction (EI) should be executed to allow recognition of interrupts after completion of the current service routine.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

Given: Two interrupting devices, A and B connected in a daisy chain configuration with A having a higher priority than B.



B generates an interrupt and is acknowledged. (The interrupt enable out, IEO, of B goes low, blocking any lower priority devices from interrupting while B is being serviced). Then A generates an interrupt, suspending service of B. (The IEO of A goes 'low' indicating that a higher priority device is being serviced.) The A routine is completed and a RETI is issued resetting the IEO of A, allowing the B routine to continue. A second RETI is issued on completion of the B routine and the IEO of B is reset (high) allowing lower priority devices interrupt access.

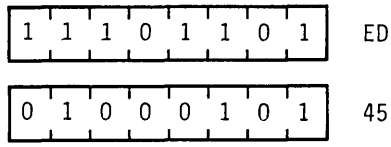
RETN

Operation: **Return from non maskable interrupt**

Format:

Opcode

RETN



Description:

This instruction is used at the end of a non-maskable interrupt service routine to restore the contents of the Program Counter (PC) (analogous to the RET instruction). The state of IFF2 is copied back into IFF1 so that maskable interrupts are enabled immediately following the RETN if they were enabled before the non-maskable interrupt.

M CYCLES: 4 T STATES: 14(4,4,3,3) 4 MHZ E.T.: 3.50

Condition Bits Affected: None

Example:

If the contents of the Stack Pointer are 1000H and the contents of the Program Counter are 1A45H when a non maskable interrupt (NMI) signal is received, the CPU will ignore the next instruction and will instead restart to memory address 0066H. That is, the current Program Counter contents of 1A45H will be pushed onto the external stack address of 0FFFH and 0FFEH, high order-byte first, and 0066H will be loaded onto the Program Counter. That address begins an interrupt service routine which ends with RETN instruction. Upon the execution of RETN, the former Program Counter contents are popped off the external memory stack, low-order first, resulting in a Stack Pointer contents again of 1000H. The program flow continues where it left off with an opcode fetch to address 1A45H.

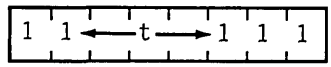
order-byte first, and 0066H will be loaded onto the Program Counter. That address begins an interrupt service routine which ends with RETN instruction. Upon the execution of RETN, the former Program Counter contents are popped off the external memory stack, low-order first, resulting in a Stack Pointer contents again of 1000H. The program flow continues where it left off with an opcode fetch to address 1A45H.

RST p

Operation: $(SP-1) \leftarrow PC_H$, $(SP-2) \leftarrow PC_L$, $PC_H \leftarrow 0$, $PC_L \leftarrow P$

Format:

Opcode	Operand
RST	p



Description:

The current Program Counter (PC) contents are pushed onto the external memory stack, and the page zero memory location given by operand p is loaded into the PC. Program execution then begins with the opcode in the address now pointed to by PC. The push is performed by first decrementing the contents of the Stack Pointer (SP), loading the high-order byte of PC into the memory address now pointed to by SP, decrementing SP again, and loading the low-order byte of PC into the address now pointed to by SP. The ReSTart instruction allows for a jump to one of eight addresses as shown in the table below. The operand p is assembled into the object code using the corresponding T state. Note: Since all addresses are in page zero of memory, the high order byte of PC is loaded with 00H. The number selected from the "p" column of the table is loaded into the low-order byte of PC.

p	t
00H	000
08H	001
10H	010
18H	011
20H	100
28H	101
30H	110
38H	111

M CYCLES: 3 T STATES: 11(5,3,3) 4 MHZ E.T.: 2.75

Example:

If the contents of the Program Counter are 15B3H, after the execution of

RST 18H (Object code 1101111)

the PC will contain 0018H, as the address of the next opcode to be fetched.

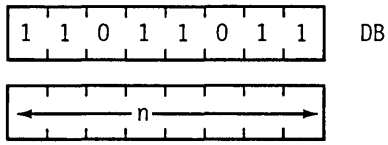
-INPUT AND OUTPUT GROUP-

IN A, (n)

Operation: A ← (n)

Format:

Opcode	Operands
IN	A,(n)



Description:

The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator also appear on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written into the Accumulator (register A) in the CPU.

M CYCLES: 3 T STATES: 11(4,3,4) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H and the byte 7BH is available at the peripheral device mapped to I/O port address 01H, then after the execution of

```
IN A,(01H)
```

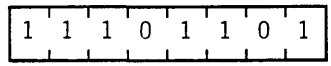
the Accumulator will contain 7BH.

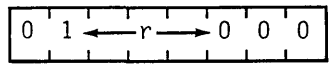
IN r, (C)

Operation: r ← (C)

Format:

Opcode	Operands
IN	r,(C)





Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written into register r in the CPU. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding 3-bit "r" field for each. The flags will be affected, checking the input data.

Reg.	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

M CYCLES: 3 T STATES: 12(4,4,4) 4 MHZ E.T.: 3.00

Condition Bits Affected:

S:	Set if input data is negative; reset otherwise
Z:	Set if input data is zero; reset otherwise
H:	Reset
P/V:	Set if parity is even; reset otherwise
N:	Reset
C:	Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, and the byte 7BH is available at the peripheral device mapped to I/O port address 07H, then after the execution of

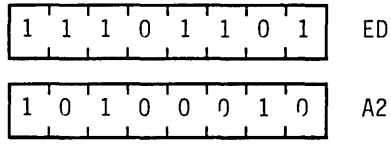
```
IN D,(C)
```

Operation: (HL) ← (C) , B ← B-1 , HL ← HL + 1

Format:

Opcode

INI



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are then placed on the address bus and the input byte is written into the corresponding location of memory. Finally the byte counter is decremented and register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Unknown
Z:	Set if B-1=0; reset otherwise
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the byte 7BH is available at the peripheral device mapped to I/O port address 07H, then

after the execution of

INI

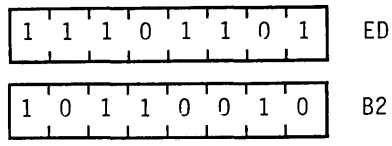
memory location 1000H will contain 7BH, the HL register pair will contain 1001H, and register B will contain 0FH.

Operation: (HL) ← (C) , B ← B-1 , HL ← HL + 1

Format:

Opcode

INIR



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Then register pair HL is incremented, the byte counter is decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Interrupts will be recognized and two refresh cycles will be executed after each data transfer. Note that if B is set to zero prior to instruction execution, 256 bytes of data will be input.

If B≠0:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If B=0:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Unknown
Z:	Set
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and the following sequence of bytes are available at the peripheral device mapped to I/O port of address 07H:

```
51H
A9H
03H
```

then after the execution of

```
INIR
```

the HL register pair will contain 1003H, register B will contain zero, and memory locations will have contents as follows:

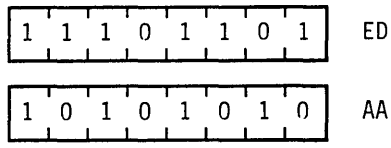
Location	Contents
1000H	51H
1001H	A9H
1002H	03H

Operation: (HL) ← (C), B ← B-1, HL ← HL-1

Format:

Opcode

IND



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Finally the byte counter and register pair HL are decremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set if B-1=0; reset otherwise
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the byte 7BH is available at the

peripheral device mapped to I/O port address 07H, then after the execution of

```
IND
```

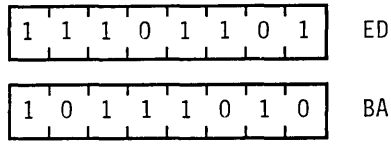
memory location 1000H will contain 7BH, the HL register pair will contain 0FFFH, and register B will contain 0FH.

Operation: (HL) ← (C), B ← B-1, HL ← HL-1

Format:

Opcode

INDR



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B is used as a byte counter, and its contents are placed on the top half (A8 through A15) of the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are placed on the address bus and the input byte is written into the corresponding location of memory. Then HL and the byte counter are decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the PC is decremented by two and the instruction repeated. Interrupts will be recognized and two refresh cycles will be executed after each data transfer. Note that if B is set to zero prior to instruction execution, 256 bytes of data will be input.

If B≠0:

M CYCLES: 5 T STATES:21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If B=0:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Unknown
Z:	Set
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and the following sequence of bytes are available at the peripheral device mapped to I/O port address 07H:

```
51H
A9H
03H
```

then after the execution of

```
INDR
```

the HL register pair will contain 0FFDH, register B will contain zero, and memory locations will have contents as follows:

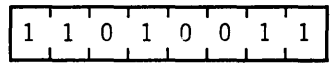
Location	Contents
0FFEH	03H
0FFFH	A9H
1000H	51H

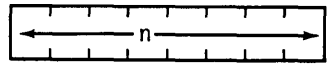
OUT (n), A

Operation: (n) ← A

Format:

Opcode	Operands
OUT	(n),A





Description:

The operand n is placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of the Accumulator (register A) also appear on the top half (A8 through A15) of the address bus at this time. Then the byte contained in the Accumulator is placed on the data bus and written into the selected peripheral device.

M CYCLES: 3 T STATES: 11(4,3,4) 4 MHZ E.T.: 2.75

Condition Bits Affected: None

Example:

If the contents of the Accumulator are 23H, then after the execution of

```
OUT (01H),A
```

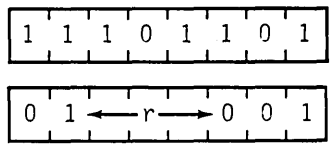
the byte 23H will have been written to the peripheral device mapped to I/O port address 01H.

OUT (C), r

Operation: (C) ← r

Format:

Opcode	Operands
OUT	(C),r



Description:

The contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. The contents of Register B are placed on the top half (A8 through A15) of the address bus at this time. Then the byte contained in register r is placed on the data bus and written into the selected peripheral device. Register r identifies any of the CPU registers shown in the following table, which also shows the corresponding 3-bit "r" field for each which appears in the assembled object code:

Register	r
B	000
C	001
D	010
E	011
H	100
L	101
A	111

M CYCLES: 3 T STATES: 12(4,4,4) 4 MHZ E.T.: 3.00

Condition Bits Affected: None

Example:

If the contents of register C are 01H and the contents of register D are 5AH, after the execution of

```
OUT (C),D
```

the byte 5AH will have been written to the peripheral device mapped to I/O port address 01H.

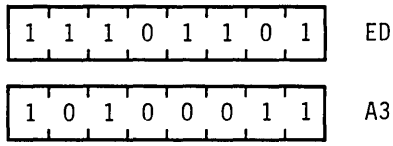
OUTI

Operation: (C) ← (HL), B ← B-1, HL ← HL + 1

Format:

Opcode

OUTI



Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus. The byte to be output is placed on the data bus and written into selected peripheral device. Finally the register pair HL is incremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S: Unknown
Z: Set if B-1=0; reset otherwise
H: Unknown
P/V: Unknown
N: Set
C: Not affected

Example:

If the contents of register C are 07H, the contents of register B are 10H, the contents of the HL register pair are 1000H, and the contents of memory address 1000H are

59H, then after the execution of

```
OUTI
```

register B will contain 0FH, the HL register pair will contain 1001H, and the byte 59H will have been written to the peripheral device mapped to I/O port address 07H.

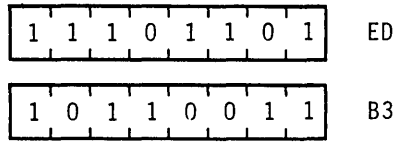
OTIR

Operation: (C) ← (HL), B ← B-1, HL ← HL + 1

Format:

Opcode

OTIR



Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Then register pair HL is incremented. If the decremented B register is not zero, the Program Counter (PC) is decremented by 2 and the instruction is repeated. If B has gone to zero, the instruction is terminated. Interrupts will be recognized and two refresh cycles will be executed after each data transfer. Note that if B is set to zero prior to instruction execution, the instruction will output 256 bytes of data.

If B≠0:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If B=0:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Unknown
Z:	Set
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and memory locations have the following contents:

Location	Contents
1000H	51H
1001H	A9H
1002H	03H

then after the execution of

```
OTIR
```

the HL register pair will contain 1003H, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port address 07H in the following sequence:

```
51H
A9H
03H
```

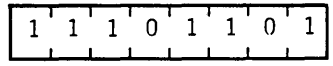
OUTD

Operation: (C) ← (HL), B ← B-1, HL ← HL-1

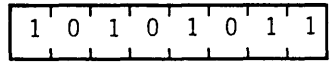
Format:

Opcode

OUTD



ED



AB

Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Finally the register pair HL is decremented.

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Unknown
Z:	Set if B-1=0; reset otherwise
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are 07H, the contents of

register B are 10H, the contents of the HL register pair are 1000H, and the contents of memory location 1000H are 59H, after the execution of

```
OUTD
```

register B will contain 0FH, the HL register pair will contain 0FFFH, and the byte 59H will have been written to the peripheral device mapped to I/O port address 07H.

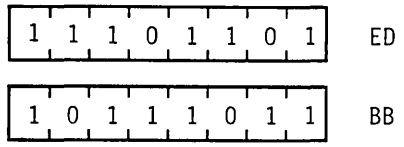
OTDR

Operation: (C) ← (HL), B ← B-1, HL ← HL-1

Format:

Opcode

OTDR



Description:

The contents of the HL register pair are placed on the address bus to select a location in memory. The byte contained in this memory location is temporarily stored in the CPU. Then, after the byte counter (B) is decremented, the contents of register C are placed on the bottom half (A0 through A7) of the address bus to select the I/O device at one of 256 possible ports. Register B may be used as a byte counter, and its decremented value is placed on the top half (A8 through A15) of the address bus at this time. Next the byte to be output is placed on the data bus and written into the selected peripheral device. Then register pair HL is decremented and if the decremented B register is not zero, the Program Counter (PC) is decremented by 2 and the instruction is repeated. If B has gone to zero, the instruction is terminated. Interrupts will be recognized and two refresh cycles will be executed after each data transfer. Note that if B is set to zero prior to instruction execution, the instruction will output 256 bytes of data.

If B≠0:

M CYCLES: 5 T STATES: 21(4,5,3,4,5) 4 MHZ E.T.: 5.25

If B=0:

M CYCLES: 4 T STATES: 16(4,5,3,4) 4 MHZ E.T.: 4.00

Condition Bits Affected:

S:	Unknown
Z:	Set
H:	Unknown
P/V:	Unknown
N:	Set
C:	Not affected

Example:

If the contents of register C are 07H, the contents of register B are 03H, the contents of the HL register pair are 1000H, and memory locations have the following contents:

Location	Contents
0FFEH	51H
0FFFH	A9H
1000H	03H

then after the execution of

```
OTDR
```

the HL register pair will contain 0FFDH, register B will contain zero, and a group of bytes will have been written to the peripheral device mapped to I/O port addres 07H in the following sequence:

```
03H
A9H
51H
```

ALPHABETICAL

ASSEMBLY MNEMONIC	OPERATION	PAGE
ADC HL,ss	Add with Carry Reg. pair ss to HL	149
ADC A,s	Add with carry operand s to Acc.	108
ADD A,n	Add value n to Acc.	102
ADD A,r	Add Reg. r to Acc.	100
ADD A,(HL)	Add location (HL) to Acc.	103
ADD A,(IX+d)	Add location (IX+d) to Acc.	104
ADD A,(IY+d)	Add location (IY+d) to Acc.	106
ADD HL,ss	Add Reg. pair ss to HL	147
ADD IX,pp	Add Reg. pair pp to IX	153
ADD IY,rr	Add Reg. pair rr to IY	155
AND s	Logical 'AND' of operand s and Acc.	114
BIT b,(HL)	Test BIT b of location (HL)	205
BIT b,(IX+d)	Test BIT b of location (IX+d)	207
BIT b,(IY+d)	Test BIT b of location (IY+d)	209
BIT b,r	Test BIT b of Reg. r	203
CALL cc,nn	Call subroutine at location nn if condition cc is true	240
CALL nn	Unconditional call subroutine at location nn	238
CCF	Complement carry flag	137
CP s	Compare operand s with Acc.	120
CPD	Compare location (HL) and Acc. decrement HL and BC	95
CPDR	Compare location (HL) and Acc. decrement HL and BC, repeat until BC=0	97
CPI	Compare location (HL) and Acc. increment HL and decrement BC	91
CPIR	Compare location (HL) and Acc. increment HL, decrement BC repeat until BC=0	93
CPL	Complement Acc. (1's comp)	134
DAA	Decimal adjust Acc.	132
DEC m	Decrement operand m	129
DEC IX	Decrement IX	161
DEC IY	Decrement IY	162
DEC ss	Decrement Reg. pair ss	160
DI	Disable interrupts	141
DJNZ e	Decrement B and Jump relative if B=0	235
EI	Enable interrupts	142
EX (SP),HL	Exchange the location (SP) and HL	80

EX (SP),IX	Exchange the location (SP) and IX	81
EX (SP),IY	Exchange the location (SP) and IY	82
EX AF,AF′	Exchange the contents of AF and AF′	78
EX DE,HL	Exchange the contents of DE and HL	77
EXX	Exchange the contents of BC,DE,HL with contents of BC′,DE′,HL′ respectively	79
HALT	HALT (wait for interrupt or reset)	140
IM 0	Set interrupt mode 0	143
IM 1	Set interrupt mode 1	144
IM 2	Set interrupt mode 2	145
IN A,(n)	Load the Acc. with input from device n	253
IN r,(C)	Load the Reg. r with input from device (C)	254
INC (HL)	Increment location (HL)	124
INC IX	Increment IX	158
INC (IX+d)	Increment location (IX+d)	125
INC IY	Increment IY	159
INC (IY+d)	Increment location (IY+d)	127
INC r	Increment Reg. r	122
INC ss	Increment Reg. pair ss	157
IND	Load location (HL) with input from port (C), decrement HL and B	260
INDR	Load location (HL) with input from port (C), decrement HL and decrement B, repeat until B=0	262
INI	Load location (HL) with input from port (C); and increment HL and decrement B	256
INIR	Load location (HL) with input from port (C), increment HL and decrement B, repeat until B=0	258
JP (HL)	Unconditional Jump to (HL)	232
JP (IX)	Unconditional Jump to (IX)	233
JP (IY)	Unconditional Jump to (IY)	234
JP cc,nn	Jump to location nn if condition cc is true	221
JP nn	Unconditional jump to location nn	220
JR C,e	Jump relative to PC+e if carry=1	224
JR e	Unconditional Jump relative to PC+e	223
JR NC,e	Jump relative to PC+e if carry=0	226

JR NZ,e	Jump relative to PC+e if non zero (Z=0)	230
JR Z,e	Jump relative to PC+e if zero (Z=1)	228
LD A,(BC)	Load Acc. with location (BC)	42
LD A,(DE)	Load Acc. with location (DE)	43
LD A,I	Load Acc. with I	48
LD A,(nn)	Load Acc. with location nn	44
LD A,R	Load Acc. with Reg. R	49
LD (BC),A	Load location (BC) with Acc.	45
LD (DE),A	Load location (DE) with Acc.	46
LD (HL),n	Load location (HL) with value n	39
LD dd,nn	Load Reg. pair dd with value nn	53
LD dd,(nn)	Load Reg. pair dd with location (nn)	57
LD HL,(nn)	Load HL with location (nn)	56
LD (HL),r	Load location (HL) with Reg. r	34
LD I,A	Load I with Acc.	50
LF IX,nn	Load IX with value nn	54
LD IX,(nn)	Load IX with location (nn)	59
LD (IX+d),n	Load location (IX+d) with value n	40
LD (IX+d),r	Load location (IX+d) with Reg. r	35
LD IY,nn	Load IY with value nn	55
LD IY,(nn)	Load IY with location (nn)	60
LD (IY+d),n	Load location (IY+d) with value n	41
LD (IY+d),r	Load location (IY+d) with Reg. r	37
LD (nn),A	Load location (nn) with Acc.	47
LD (nn),dd	Load location (nn) with Reg. pair dd	62
LD (nn),HL	Load location (nn) with HL	61
LD (nn),IX	Load location (nn) with IX	64
LD (nn),IY	Load location (nn) with IY	65
LD R,A	Load R with Acc.	51
LD r,(HL)	Load Reg. r with location (HL)	29
LD r,(IX+d)	Load Reg. r with location (IX+d)	30
LD r,(IY+d)	Load Reg. r with location (IY+d)	32
LD r,n	Load Reg. r with value n	28
LD r,r′	Load Reg. r with Reg. r′	27
LD SP,HL	Load SP with HL	66
LD SP,IX	Load SP with IX	67
LD SP,IY	Load SP with IY	68
LDD	Load location (DE) with location (HL), decrement DE,HL and BC	87
LDDR	Load location (DE) with location (HL), decrement DE,HL and BC; repeat until BC=0	89

LDI	Load location (DE) with location (HL), increment DE,HL, decrement BC	83
LDIR	Load location (DE) with location (HL), increment DE,HL, decrement BC and repeat until BC=0	85
NEG	Negate Acc. (2's complement)	135
NOP	No operation	139
OR s	Logical 'OR' of operand s and Acc.	116
OTDR	Load output port (C) with location (HL) decrement HL and B, repeat until B=0	273
OTIR	Load output port (C) with location (HL), increment HL, decrement B, repeat until B=0	269
OUT (C),r	Load output port (C) with Reg. r	265
OUT (n),A	Load output port (n) with Acc.	264
OUTD	Load output port (C) with location (HL), decrement HL and B	271
OUTI	Load output port (C) with location (HL), increment HL and decrement B	267
POP IX	Load IX with top of stack	74
POP IY	Load IY with top of stack	75
POP qq	Load Reg. pair qq with top of stack	72
PUSH IX	Load IX onto stack	70
PUSH IY	Load IY onto stack	71
PUSH qq	Load Reg. pair qq onto stack	69
RES b,m	Reset Bit b of operand m	217
RET	Return from subroutine	243
RET cc	Return from subroutine if condition cc is true	244
RETI	Return from interrupt	246
RETN	Return from non maskable interrupt	248
RL m	Rotate left through carry operand m	180
RLA	Rotate left Acc. through carry	166
RLC (HL)	Rotate location (HL) left circular	174
RLC (IX+d)	Rotate location (IX+d) left circclar	176
RLC (IY+d)	Rotate location (IY+d) left circular	178
RLC r	Rotate Reg. r left circular	172
RLCA	Rotate left circular Acc.	164
RLD	Rotate digit left and right between Acc. and location (HL)	198
RR m	Rotate right through carry operand m	186
RRA	Rotate right Acc. through carry	170
RRC m	Rotate operand m right circular	183

RRCA	Rotate right circular Acc.	168
RRD	Rotate digit right and left between Acc. and location (HL)	200
RST p	Restart to location p	250
SBC A,s	Subtract operand s from Acc. with carry	112
SBC HL,ss	Subtract Reg. pair ss from HL with carry	151
SCF	Set carry flag (C=1)	138
SET b,(HL)	Set Bit b of location (HL)	212
SET b,(IX+d)	Set Bit b of location (IX+d)	213
SET b,(IY+d)	Set Bit b of location (IY+d)	215
SET b,r	Set Bit b of Reg. r	211
SLA m	Shift operand m left arithmetic	189
SRA m	Shift operand m right arithmetic	192
SRL m	Shift operand m right logical	195
SUB s	Subtract operand s from Acc.	110
XOR s	Exclusive 'OR' operand s and Acc.	118

APPENDIX A
ERROR MESSAGES AND EXPLANATIONS

1) WARNING - OPCODE REDEFINED
   Indicates that an opcode has been redefined by a macro so that future uses of the opcode will result in the appropriate macro call. This message may be suppressed by the NOW option.
2) NAME CONTAINS INVALID CHARACTERS
   Indicates that a name (either a label or an operand) contains illegal characters. Names must start with an alphabetic character and any following characters must be either alphanumeric (A...Z or 0...9), a question mark (?) or an underbar ( ).
3) INVALID OPCODE
   Indicates that the opcode was not recognized. Occurs when the opcode contains an illegal character (including non-printing control characters), when the opcode is not either all upper case or all lower case, or when macros are used and the M option is not specified.
4) INVALID NUMBER
   Indicates an invalid character in a number. Occurs when a number contains an illegal character (including non-printing control characters) or a number contains a digit not allowed in the specified base (e.g., 8 or 9 in an octal number or a letter in a hexadecimal number where the trailing H was omitted.)
5) INVALID OPERATOR
   Indicates use of an invalid operator in an expression. Occurs when an operator such as AND or XOR is misspelled or contains illegal characters.
6) SYNTAX ERROR
   Indicates the syntax of the statement is invalid. Occurs when an expression is incorrectly formed, unmatched parenthesis are found in an operand field, or a DEFM string is either too long (greater than 63 characters) or contains unbalanced quotes.
7) ASSEMBLER ERROR
   Indicates that the assembler has failed to process this instruction. Usually occurs when an expression is incorrectly formed.
8) UNDEFINED SYMBOL
   Indicates that a symbol in an operand field

was never defined. Occurs when a name is misspelled or not declared as a label for an instruction or pseudo-op.

9) INVALID OPERAND COMBINATION
Indicates that the operand combination for this opcode is invalid. Occurs when a register name or condition code is missspelled or incorrectly used with the particular opcode.

10) EXPRESSION OUT OF RANGE
Indicates that the value of an expression is either too large or too small for the appropriate quantity. Occurs on 16-bit arithmetic overflow or division by zero in an expression, incrementing the reference counter beyond a 16-bit value, or trying to use a value which will not fit into a particular bit-field - typically a byte.

11) MULTIPLE DECLARATION
Indicates that an attempt was made to redefine a label. Occurs when a label is misspelled, or mistakenly used several times. The pseudo-op DEFL can be used to assign a value to a label which can then be redefined by another DEFL.

12) MACRO DEFINITION ERROR
Indicates that a macro is incorrectly defined. Occurs when the M option is not specified but macros are used, when a macro is defined within another macro definition, when the parameters are not correctly specified, or an unrecognized parameter is found in the macro body.

13) UNBALANCED QUOTES
Indicates that a string is not properly bounded by single quote marks or quote marks inside a string are not properly matched in pairs.

14) ASSEMBLER COMMAND ERROR
Indicates that an assembler command is not recognized or is incorrectly formed. The command must begin with an asterisk (*) in column one, the first letter identifies the command, and any parameters such as 'ON','OFF' or a filename must be properly delimited. The command will be ignored.

15) MACRO EXPANSION ERROR
Indicates that the expansion of a single line in a macro has overflowed the expansion buffer. Occurs when substitution of parameter causes the line to increase in length beyond the capacity of the buffer (currently 128

bytes). The line will be truncated.

16) MACRO STACK OVERFLOW
Indicates that the depth of nesting of macro calls has exceeded the macro parameter stack buffer capacity. Occurs when the sum of the parameter string lengths (plus some additional information for each macro call) is longer than the buffer (currently 256 bytes), which often happens if infinitely recursive macro calls are used. The macro call which caused the error will be ignored.

17) INCLUDE NESTED TOO DEEP
Indicates that a *Include command was found which would have caused a nesting of included source files to a depth greater than four, where the original source file is considered to be level one. The command will be ignored.

18) GLOBAL DEFINITION ERROR
Indicates that either a label was present on a GLOBAL pseudo-op statement, or there was an attempt to give an absolute value to a GLOBAL symbol in a relocatable module. The latter case is not allowed since all GLOBALs in a relocatable module will be relocated by the Linker. May occur either after a GLOBAL pseudo-op or after an EQU or DEFL statement which is attempting to absolutize a relocatable GLOBAL symbol.

19) EXTERNAL DEFINITION ERROR
Indicates that either a label was present on an EXTERNAL pseudo-op statement, or there was an attempt to declare a symbol to be EXTERNAL which had previously been defined within the module to have an absolute value. May occur due to a misspelling or other oversight.

20) NAME DECLARED GLOBAL AND EXTERNAL
Indicates that the name was found in both a GLOBAL pseudo-op and an EXTERNAL pseudo-op which is contradictory. May occur due to a misspelling or other oversight.

21) LABEL DECLARED AS EXTERNAL
Indicates that a name has been declared in both an EXTERNAL pseudo-op and as a label in this module. May occur due to a misspelling or other oversight.

22) INVALID EXTERNAL EXPRESSION
Indicates that a symbol name which has been declared in an EXTERNAL pseudo-op is improperly used in an expression. May occur when invalid arithmetic operators are applied to an external expression or when the mode of

an operand must be either absolute or relocatable.

23) INVALID RELOCATABLE EXPRESSION
Indicates than an expression which contains a relocatable value (either a label or the reference counter sumbol $ in a relocatable module) is improperly formed or used. May occur when invalid arithmetic operators are applied to a relocatable expression or when the mode of an operand must be absolute. Remember that all relocatable values (addresses) must be represented in 16 bits.

24) EXPRESSION MUST BE ABSOLUTE
Indicates that the mode of an expression is not absolute when it should be. May occur when a relocatable or external expression is used to specify a quantity that must be either constant or representable in less than 16 bits.

25) UNDEFINED GLOBAL(S)
Indicates that one or more sumbols which were declared in a GLOBAL pseudo-op were never actually defined as a label in this module. May occur due to a misspelling or other oversight.

26) WARNING - ORG IS RELOCATABLE
Indicates that an ORG statement was encountered in a relocatable module. This warning is issued to remind the user that the reference counter is set to a relocatable value, not an absolute one. May occur when the Absolute option is not specified for an absolute module. This warning may be suppressed by the NOW option.

APPENDIX B
INSTRUCTION SET ALPHABETICAL ORDER

```
                Z-80 CROSS ASSEMBLER    VERSION  1.06  OF 06/18/76
07/09/76  10:22:47              OPCODE   LISTING
 LOC     OBJ CODE   STMT SOURCE STATEMENT

0000    8E            1        ADC     A,(HL)
0001    DD8E05        2        ADC     A,(IX+IND)
0004    FD8E05        3        ADC     A,(IY+IND)
0007    8F            4        ADC     A,A
0008    88            5        ADC     A,B
0009    89            6        ADC     A,C
000A    8A            7        ADC     A,D
000B    8B            8        ADC     A,E
000C    8C            9        ADC     A,H
000D    8D           10        ADC     A,L
000E    CE20         11        ADC     A,N
0010    ED4A         12        ADC     HL,BC
0012    ED5A         13        ADC     HL,DE
0014    ED6A         14        ADC     HL,HL
0016    ED7A         15        ADC     HL,SP
0018    86           16        ADD     A,(HL)
0019    DD8605       17        ADD     A,(IX+IND)
001C    FD8605       18        ADD     A,(IY+IND)
001F    87           19        ADD     A,A
0020    80           20        ADD     A,B
0021    81           21        ADD     A,C
0022    82           22        ADD     A,D
0023    83           23        ADD     A,E
0024    84           24        ADD     A,H
0025    85           25        ADD     A,L
0026    C620         26        ADD     A,N
0028    09           27        ADD     HL,BC
0029    19           28        ADD     HL,DE
002A    29           29        ADD     HL,HL
002B    39           30        ADD     HL,SP
002C    DD09         31        ADD     IX,BC
002E    DD19         32        ADD     IX,DE
0030    DD29         33        ADD     IX,IX
0032    DD39         34        ADD     IX,SP
0034    FD09         35        ADD     IY,BC
0036    FD19         36        ADD     IY,DE
0038    FD29         37        ADD     IY,IY
003A    FD39         38        ADD     IY,SP
003C    A6           39        AND     (HL)
003D    DDA605       40        AND     (IX+IND)
0040    FDA605       41        AND     (IY+IND)
0043    A7           42        AND     A
0044    A0           43        AND     B
0045    A1           44        AND     C
0046    A2           45        AND     D
0047    A3           46        AND     E
0048    A4           47        AND     H
0049    A5           48        AND     L
004A    E620         49        AND     N
004C    CB46         50        BIT     0,(HL)
004E    DDCB0546     51        BIT     0,(IX+IND)
0052    FDCB0546     52        BIT     0,(IY+IND)
0056    CB47         53        BIT     0,A
0058    CB40         54        BIT     0,B
005A    CB41         55        BIT     0,C
005C    CB42         56        BIT     0,D
005E    CB43         57        BIT     0,E
0060    CB44         58        BIT     0,H
0062    CB45         59        BIT     0,L
0064    CB4E         60        BIT     1,(HL)
0066    DDCB054E     61        BIT     1,(IX+IND)
006A    FDCB054E     62        BIT     1,(IY+IND)
006E    CB4F         63        BIT     1,A
0070    CB48         64        BIT     1,B
0072    CB49         65        BIT     1,C
0074    CB4A         66        BIT     1,D
0076    CB4B         67        BIT     1,E
0078    CB4C         68        BIT     1,H
007A    CB4D         69        BIT     1,L

007C    CB56         70        BIT     2,(HL)
007E    DDCB0556     71        BIT     2,(IX+IND)
0082    FDCB0556     72        BIT     2,(IY+IND)
0086    CB57         73        BIT     2,A
0088    CB50         74        BIT     2,B
008A    CB51         75        BIT     2,C
008C    CB52         76        BIT     2,D
008E    CB53         77        BIT     2,E
0090    CB54         78        BIT     2,H
0092    CB55         79        BIT     2,L
0094    CB5E         80        BIT     3,(HL)
0096    DDCB055E     81        BIT     3,(IX+IND)
009A    FDCB055E     82        BIT     3,(IY+IND)
009E    CB5F         83        BIT     3,A
00A0    CB58         84        BIT     3,B
00A2    CB59         85        BIT     3,C
00A4    CB5A         86        BIT     3,D
00A6    CB5B         87        BIT     3,E
00A8    CB5C         88        BIT     3,H
00AA    CB5D         89        BIT     3,L
00AC    CB66         90        BIT     4,(HL)
00AE    DDCB0566     91        BIT     4,(IX+IND)
00B2    FDCB0566     92        BIT     4,(IY+IND)
00B6    CB67         93        BIT     4,A
00B8    CB60         94        BIT     4,B
00BA    CB61         95        BIT     4,C
00BC    CB62         96        BIT     4,D
00BE    CB63         97        BIT     4,E
00C0    CB64         98        BIT     4,H
00C2    CB65         99        BIT     4,L
00C4    CB6E        100        BIT     5,(HL)
00C6    DDCB056E    101        BIT     5,(IX+IND)
00CA    FDCB056E    102        BIT     5,(IY+IND)
00CE    CB6F        103        BIT     5,A
00D0    CB68        104        BIT     5,B
00D2    CB69        105        BIT     5,C
00D4    CB6A        106        BIT     5,D
00D6    CB6B        107        BIT     5,E
00D8    CB6C        108        BIT     5,H
00DA    CB6D        109        BIT     5,L
00DC    CB76        110        BIT     6,(HL)
00DE    DDCB0576    111        BIT     6,(IX+IND)
00E2    FDCB0576    112        BIT     6,(IY+IND)
00E6    CB77        113        BIT     6,A
00E8    CB70        114        BIT     6,B
00EA    CB71        115        BIT     6,C
00EC    CB72        116        BIT     6,D
00EE    CB73        117        BIT     6,E
00F0    CB74        118        BIT     6,H
00F2    CB75        119        BIT     6,L
00F4    CB7E        120        BIT     7,(HL)
00F6    DDCB057E    121        BIT     7,(IX+IND)
00FA    FDCB057E    122        BIT     7,(IY+IND)
00FE    CB7F        123        BIT     7,A
0100    CB78        124        BIT     7,B
0102    CB79        125        BIT     7,C
0104    CB7A        126        BIT     7,D
0106    CB7B        127        BIT     7,E
0108    CB7C        128        BIT     7,H
010A    CB7D        129        BIT     7,L
010C    DC8405      130        CALL    C,NN
010F    FC8405      131        CALL    M,NN
0112    D48405      132        CALL    NC,NN
0115    CD8405      133        CALL    NN
0118    C48405      134        CALL    NZ,NN
011B    F48405      135        CALL    P,NN
011E    EC8405      136        CALL    PE,NN
0121    E48405      137        CALL    PO,NN
0124    CC8405      138        CALL    Z,NN
```

```
LOC   OBJ CODE   STMT SOURCE STATEMENT
0127  3F         139      CCF
0128  BE         140      CP   (HL)
0129  DDBE05     141      CP   (IX+IND)
012C  FDBE05     142      CP   (IY+IND)
012F  BF         143      CP   A
0130  B8         144      CP   B
0131  B9         145      CP   C
0132  BA         146      CP   D
0133  BB         147      CP   E
0134  BC         148      CP   H
0135  BD         149      CP   L
0136  FE20       150      CP   N
0138  EDA9       151      CPD
013A  EDB9       152      CPDR
013C  EDA1       153      CPI
013E  EDB1       154      CPIR
0140  2F         155      CPL
0141  27         156      DAA
0142  35         157      DEC  (HL)
0143  DD3505     158      DEC  (IX+IND)
0146  FD3505     159      DEC  (IY+IND)
0149  3D         160      DEC  A
014A  05         161      DEC  B
014B  0B         162      DEC  BC
014C  0D         163      DEC  C
014D  15         164      DEC  D
014E  1B         165      DEC  DE
014F  1D         166      DEC  E
0150  25         167      DEC  H
0151  2B         168      DEC  HL
0152  DD2B       169      DEC  IX
0154  FD2B       170      DEC  IY
0156  2D         171      DEC  L
0157  3B         172      DEC  SP
0158  F3         173      DI
0159  102E       174      DJNZ   DIS
015B  FB         175      EI
015C  E3         176      EX   (SP),HL
015D  DDE3       177      EX   (SP),IX
015F  FDE3       178      EX   (SP),IY
0161  08         179      EX   AF,AF'
0162  EB         180      EX   DE,HL
0163  D9         181      EXX
0164  76         182      HALT
0165  ED46       183      IM   0
0167  ED56       184      IM   1
0169  ED5E       185      IM   2
016B  ED78       186      IN   A,(C)
016D  DB20       187      IN   A,(N)
016F  ED40       188      IN   B,(C)
0171  ED48       189      IN   C,(C)
0173  ED50       190      IN   D,(C)
0175  ED58       191      IN   E,(C)
0177  ED60       192      IN   H,(C)
0179  ED68       193      IN   L,(C)
017B  34         194      INC  (HL)
017C  DD3405     195      INC  (IX+IND)
017F  FD3405     196      INC  (IY+IND)
0182  3C         197      INC  A
0183  04         198      INC  B
0184  03         199      INC  BC
0185  0C         200      INC  C
0186  14         201      INC  D
0187  13         202      INC  DE
0188  1C         203      INC  E
0189  24         204      INC  H
018A  23         205      INC  HL
018B  DD23       206      INC  IX
018D  FD23       207      INC  IY
018F  2C         208      INC  L
0190  33         209      INC  SP
0191  EDAA       210      IND
0193  EDBA       211      INDR
0195  EDA2       212      INI
0197  EDB2       213      INIR
0199  E9         214      JP   (HL)
019A  DDE9       215      JP   (IX)
019C  FDE9       216      JP   (IY)
019E  DA8405     217      JP   C,NN
01A1  FA8405     218      JP   M,NN
01A4  D28405     219      JP   NC,NN
01A7  C38405     220      JP   NN
01AA  C28405     221      JP   NZ,NN
01AD  F28405     222      JP   P,NN
01B0  EA8405     223      JP   PE,NN
01B3  E28405     224      JP   PO,NN
01B6  CA8405     225      JP   Z,NN
01B9  382E       226      JR   C,DIS
01BB  182E       227      JR   DIS
01BD  302E       228      JR   NC,DIS
01BF  202E       229      JR   NZ,DIS
01C1  282E       230      JR   Z,DIS
01C3  02         231      LD   (BC),A
01C4  12         232      LD   (DE),A
01C5  77         233      LD   (HL),A
01C6  70         234      LD   (HL),B
01C7  71         235      LD   (HL),C
01C8  72         236      LD   (HL),D
01C9  73         237      LD   (HL),E
01CA  74         238      LD   (HL),H
01CB  75         239      LD   (HL),L
01CC  3620       240      LD   (HL),N
01CE  DD7705     241      LD   (IX+IND),A
01D1  DD7005     242      LD   (IX+IND),B
01D4  DD7105     243      LD   (IX+IND),C
01D7  DD7205     244      LD   (IX+IND),D
01DA  DD7305     245      LD   (IX+IND),E
01DD  DD7405     246      LD   (IX+IND),H
01E0  DD7505     247      LD   (IX+IND),L
01E3  DD360520   248      LD   (IX+IND),N
01E7  FD7705     249      LD   (IY+IND),A
01EA  FD7005     250      LD   (IY+IND),B
01ED  FD7105     251      LD   (IY+IND),C
01F0  FD7205     252      LD   (IY+IND),D
01F3  FD7305     253      LD   (IY+IND),E
01F6  FD7405     254      LD   (IY+IND),H
01F9  FD7505     255      LD   (IY+IND),L
01FC  FD360520   256      LD   (IY+IND),N
0200  328405     257      LD   (NN),A
0203  ED438405   258      LD   (NN),BC
0207  ED538405   259      LD   (NN),DE
020B  228405     260      LD   (NN),HL
020E  DD228405   261      LD   (NN),IX
0212  FD228405   262      LD   (NN),IY
0216  ED738405   263      LD   (NN),SP
021A  0A         264      LD   A,(BC)
021B  1A         265      LD   A,(DE)
021C  7E         266      LD   A,(HL)
021D  DD7E05     267      LD   A,(IX+IND)
0220  FD7E05     268      LD   A,(IY+IND)
0223  3A8405     269      LD   A,(NN)
0226  7F         270      LD   A,A
0227  78         271      LD   A,B
0228  79         272      LD   A,C
0229  7A         273      LD   A,D
022A  7B         274      LD   A,E
022B  7C         275      LD   A,H
022C  ED57       276      LD   A,I
```

```
LOC   OBJ CODE   STMT SOURCE STATEMENT
022E  70         277      LD   A,L
022F  3E20       278      LD   A,N
0231  46         279      LD   B,(HL)
0232  DD4605     280      LD   B,(IX+IND)
0235  FD4605     281      LD   B,(IY+IND)
0238  47         282      LD   B,A
0239  40         283      LD   B,B
023A  41         284      LD   B,C
023B  42         285      LD   B,D
023C  43         286      LD   B,E
023D  44         287      LD   B,H,NN
023E  45         288      LD   B,L
023F  0620       289      LD   B,N
0241  ED4B8405   290      LD   BC,(NN)
0245  018405     291      LD   BC,NN
0248  4E         292      LD   C,(HL)
0249  DD4E05     293      LD   C,(IX+IND)
024C  FD4E05     294      LD   C,(IY+IND)
024F  4F         295      LD   C,A
0250  48         296      LD   C,B
0251  49         297      LD   C,C
0252  4A         298      LD   C,D
0253  4B         299      LD   C,E
0254  4C         300      LD   C,H
0255  4D         301      LD   C,L
0256  0E20       302      LD   C,N
0258  56         303      LD   D,(HL)
0259  DD5605     304      LD   D,(IX+IND)
025C  FD5605     305      LD   D,(IY+IND)
025F  57         306      LD   D,A
0260  50         307      LD   D,B
0261  51         308      LD   D,C
0262  52         309      LD   D,D
0263  53         310      LD   D,E
0264  54         311      LD   D,H
0265  55         312      LD   D,L
0266  1620       313      LD   D,N
0268  ED5B8405   314      LD   DE,(NN)
026C  118405     315      LD   DE,NN
026F  5E         316      LD   E,(HL)
0270  DD5E05     317      LD   E,(IX+IND)
0273  FD5E05     318      LD   E,(IY+IND)
0276  5F         319      LD   E,A
0277  58         320      LD   E,B
0278  59         321      LD   E,C
0279  5A         322      LD   E,D
027A  5B         323      LD   E,E
027B  5C         324      LD   E,H
027C  5D         325      LD   E,L
027D  1E20       326      LD   E,N
027F  66         327      LD   H,(HL)
0280  DD6605     328      LD   H,(IX+IND)
0283  FD6605     329      LD   H,(IY+IND)
0286  67         330      LD   H,A
0287  60         331      LD   H,B
0288  61         332      LD   H,C
0289  62         333      LD   H,D
028A  63         334      LD   H,E
028B  64         335      LD   H,H
028C  65         336      LD   H,L
028D  2620       337      LD   H,N
028F  2A8405     338      LD   HL,(NN)
0292  218405     339      LD   HL,NN
0295  ED47       340      LD   I,A
0297  DD2A8405   341      LD   IX,(NN)
029B  DD218405   342      LD   IX,NN
029F  FD2A8405   343      LD   IY,(NN)
02A3  FD218405   344      LD   IY,NN
02A7  6E         345      LD   L,(HL)
02A8  DD6E05     346      LD   L,(IX+IND)
02AB  FD6E05     347      LD   L,(IY+IND)
02AE  6F         348      LD   L,A
02AF  68         349      LD   L,B
02B0  69         350      LD   L,C
02B1  6A         351      LD   L,D
02B2  6B         352      LD   L,E
02B3  6C         353      LD   L,H
02B4  6D         354      LD   L,L
02B5  2E20       355      LD   L,N
02B7  ED7B8405   356      LD   SP,(NN)
02BB  F9         357      LD   SP,HL
02BC  DDF9       358      LD   SP,IX
02BE  FDF9       359      LD   SP,IY
02C0  318405     360      LD   SP,NN
02C3  EDA8       361      LDD
02C5  EDB8       362      LDDR
02C7  EDA0       363      LDI
02C9  EDB0       364      LDIR
02CB  ED44       365      NEG
02CD  00         366      NOP
02CE  B6         367      OR   (HL)
02CF  DDB605     368      OR   (IX+IND)
02D2  FDB605     369      OR   (IY+IND)
02D5  B7         370      OR   A
02D6  B0         371      OR   B
02D7  B1         372      OR   C
02D8  B2         373      OR   D
02D9  B3         374      OR   E
02DA  B4         375      OR   H
02DB  B5         376      OR   L
02DC  F620       377      OR   N
02DE  EDBB       378      OTDR
02E0  EDB3       379      OTIR
02E2  ED79       380      OUT  (C),A
02E4  ED41       381      OUT  (C),B
02E6  ED49       382      OUT  (C),C
02E8  ED51       383      OUT  (C),D
02EA  ED59       384      OUT  (C),E
02EC  ED61       385      OUT  (C),H
02EE  ED69       386      OUT  (C),L
02F0  D320       387      OUT  N,A
02F2  EDAB       388      OUTD
02F4  EDA3       389      OUTI
02F6  F1         390      POP  AF
02F7  C1         391      POP  BC
02F8  D1         392      POP  DE
02F9  E1         393      POP  HL
02FA  DDE1       394      POP  IX
02FC  FDE1       395      POP  IY
02FE  F5         396      PUSH AF
02FF  C5         397      PUSH BC
0300  D5         398      PUSH DE
0301  E5         399      PUSH HL
0302  DDE5       400      PUSH IX
0304  FDE5       401      PUSH IY
0306  CB86       402      RES  0,(HL)
0308  DDCB0586   403      RES  0,(IX+IND)
030C  FDCB0586   404      RES  0,(IY+IND)
0310  CB87       405      RES  0,A
0312  CB80       406      RES  0,B
0314  CB81       407      RES  0,C
0316  CB82       408      RES  0,D
0318  CB83       409      RES  0,E
031A  CB84       410      RES  0,H
031C  CB85       411      RES  0,L
031E  CB8E       412      RES  1,(HL)
0320  DDCB058E   413      RES  1,(IX+IND)
0324  FDCB058E   414      RES  1,(IY+IND)
```

```
LOC   OBJ CODE   STMT SOURCE STATEMENT
0328  CB8F       415      RES   1,A
032A  CB88       416      RES   1,B
032C  CB89       417      RES   1,C
032E  CB8A       418      RES   1,D
0330  CB8B       419      RES   1,E
0332  CB8C       420      RES   1,H
0334  CB8D       421      RES   1,L
0336  CB96       422      RES   2,(HL)
0338  DDCB0596   423      RES   2,(IX+IND)
033C  FDCB0596   424      RES   2,(IY+IND)
0340  CB97       425      RES   2,A
0342  CB90       426      RES   2,B
0344  CB91       427      RES   2,C
0346  CB92       428      RES   2,D
0348  CB93       429      RES   2,E
034A  CB94       430      RES   2,H
034C  CB95       431      RES   2,L
034E  CB9E       432      RES   3,(HL)
0350  DDCB059E   433      RES   3,(IX+IND)
0354  FDCB059E   434      RES   3,(IY+IND)
0358  CB9F       435      RES   3,A
035A  CB98       436      RES   3,B
035C  CB99       437      RES   3,C
035E  CB9A       438      RES   3,D
0360  CB9B       439      RES   3,E
0362  CB9C       440      RES   3,H
0364  CB9D       441      RES   3,L
0366  CBA6       442      RES   4,(HL)
0368  DDCB05A6   443      RES   4,(IX+IND)
036C  FDCB05A6   444      RES   4,(IY+IND)
0370  CBA7       445      RES   4,A
0372  CBA0       446      RES   4,B
0374  CBA1       447      RES   4,C
0376  CBA2       448      RES   4,D
0378  CBA3       449      RES   4,E
037A  CBA4       450      RES   4,H
037C  CBA5       451      RES   4,L
037E  CBAE       452      RES   5,(HL)
0380  DDCB05AE   453      RES   5,(IX+IND)
0384  FDCB05AE   454      RES   5,(IY+IND)
0388  CBAF       455      RES   5,A
038A  CBA8       456      RES   5,B
038C  CBA9       457      RES   5,C
038E  CBAA       458      RES   5,D
0390  CBAB       459      RES   5,E
0392  CBAC       460      RES   5,H
0394  CBAD       461      RES   5,L
0396  CBB6       462      RES   6,(HL)
0398  DDCB05B6   463      RES   6,(IX+IND)
039C  FDCB05B6   464      RES   6,(IY+IND)
03A0  CBB7       465      RES   6,A
03A2  CBB0       466      RES   6,B
03A4  CBB1       467      RES   6,C
03A6  CBB2       468      RES   6,D
03A8  CBB3       469      RES   6,E
03AA  CBB4       470      RES   6,H
03AC  CBB5       471      RES   6,L
03AE  CBBE       472      RES   7,(HL)
03B0  DDCB05BE   473      RES   7,(IX+IND)
03B4  FDCB05BE   474      RES   7,(IY+IND)
03B8  CBBF       475      RES   7,A
03BA  CBB8       476      RES   7,B
03BC  CBB9       477      RES   7,C
03BE  CBBA       478      RES   7,D
03C0  CBBB       479      RES   7,E
03C2  CBBC       480      RES   7,H
03C4  CBBD       481      RES   7,L
03C6  C9         482      RET
03C7  D8         483      RET   C
03C8  F8         484      RET   M
03C9  D0         485      RET   NC
03CA  C0         486      RET   NZ
03CB  F0         487      RET   P
03CC  E8         488      RET   PE
03CD  E0         489      RET   PO
03CE  C8         490      RET   Z
03CF  ED4D       491      RETI
03D1  ED45       492      RETN
03D3  CB16       493      RL    (HL)
03D5  DDCB0516   494      RL    (IX+IND)
03D9  FDCB0516   495      RL    (IY+IND)
03DD  CB17       496      RL    A
03DF  CB10       497      RL    B
03E1  CB11       498      RL    C
03E3  CB12       499      RL    D
03E5  CB13       500      RL    E
03E7  CB14       501      RL    H
03E9  CB15       502      RL    L
03EB  17         503      RLA
03EC  CB06       504      RLC   (HL)
03EE  DDCB0506   505      RLC   (IX+IND)
03F2  FDCB0506   506      RLC   (IY+IND)
03F6  CB07       507      RLC   A
03F8  CB00       508      RLC   B
03FA  CB01       509      RLC   C
03FC  CB02       510      RLC   D
03FE  CB03       511      RLC   E
0400  CB04       512      RLC   H
0402  CB05       513      RLC   L
0404  07         514      RLCA
0405  ED6F       515      RLD
0407  CB1E       516      RR    (HL)
0409  DDCB051E   517      RR    (IX+IND)
040D  FDCB051E   518      RR    (IY+IND)
0411  CB1F       519      RR    A
0413  CB18       520      RR    B
0415  CB19       521      RR    C
0417  CB1A       522      RR    D
0419  CB1B       523      RR    E
041B  CB1C       524      RR    H
041D  CB1D       525      RR    L
041F  1F         526      RRA
0420  CB0E       527      RRC   (HL)
0422  DDCB050E   528      RRC   (IX+IND)
0426  FDCB050E   529      RRC   (IY+IND)
042A  CB0F       530      RRC   A
042C  CB08       531      RRC   B
042E  CB09       532      RRC   C
0430  CB0A       533      RRC   D
0432  CB0B       534      RRC   E
0434  CB0C       535      RRC   H
0436  CB0D       536      RRC   L
0438  0F         537      RRCA
0439  ED67       538      RRD
043B  C7         539      RST   0
043C  D7         540      RST   10H
043D  DF         541      RST   18H
043E  E7         542      RST   20H
043F  EF         543      RST   28H
0440  F7         544      RST   30H
0441  FF         545      RST   38H
0442  CF         546      RST   8
0443  9E         547      SBC   A,(HL)
0444  DD9E05     548      SBC   A,(IX+IND)
0447  FD9E05     549      SBC   A,(IY+IND)
044A  9F         550      SBC   A,A
044B  98         551      SBC   A,B
044C  99         552      SBC   A,C
```

```
LOC   OBJ CODE   STMT SOURCE STATEMENT
044D  9A          553     SBC   A,D
044E  9B          554     SBC   A,E
044F  9C          555     SBC   A,H
0450  9D          556     SBC   A,L
0451  DE20        557     SBC   A,N
0453  ED42        558     SBC   HL,BC
0455  ED52        559     SBC   HL,DE
0457  ED62        560     SBC   HL,HL
0459  ED72        561     SBC   HL,SP
045B  37          562     SCF
045C  CBC6        563     SET   0,(HL)
045E  DDCB05C6    564     SET   0,(IX+IND)
0462  FDCB05C6    565     SET   0,(IY+IND)
0466  CBC7        566     SET   0,A
0468  CBC0        567     SET   0,B
046A  CBC1        568     SET   0,C
046C  CBC2        569     SET   0,D
046E  CBC3        570     SET   0,E
0470  CBC4        571     SET   0,H
0472  CBC5        572     SET   0,L
0474  CBCE        573     SET   1,(HL)
0476  DDCB05CE    574     SET   1,(IX+IND)
047A  FDCB05CE    575     SET   1,(IY+IND)
047E  CBCF        576     SET   1,A
0480  CBC8        577     SET   1,B
0482  CBC9        578     SET   1,C
0484  CBCA        579     SET   1,D
0486  CBCB        580     SET   1,E
0488  CBCC        581     SET   1,H
048A  CBCD        582     SET   1,L
048C  CBD6        583     SET   2,(HL)
048E  DDCB05D6    584     SET   2,(IX+IND)
0492  FDCB05D6    585     SET   2,(IY+IND)
0496  CBD7        586     SET   2,A
0498  CBD0        587     SET   2,B
049A  CBD1        588     SET   2,C
049C  CBD2        589     SET   2,D
049E  CBD3        590     SET   2,E
04A0  CBD4        591     SET   2,H
04A2  CBD5        592     SET   2,L
04A4  CBD8        593     SET   3 B
04A6  CBDE        594     SET   3,(HL)
04A8  DDCB05DE    595     SET   3,(IX+IND)
04AC  FDCB05DE    596     SET   3,(IY+IND)
04B0  CBDF        597     SET   3,A
04B2  CBD9        598     SET   3,C
04B4  CBDA        599     SET   3,D
04B6  CBDB        600     SET   3,E
04B8  CBDC        601     SET   3,H
04BA  CBDD        602     SET   3,L
04BC  CBE6        603     SET   4,(HL)
04BE  DDCB05E6    604     SET   4,(IX+IND)
04C2  FDCB05E6    605     SET   4,(IY+IND)
04C6  CBE7        606     SET   4,A
04C8  CBE0        607     SET   4,B
04CA  CBE1        608     SET   4,C
04CC  CBE2        609     SET   4,D
04CE  CBE3        610     SET   4,E
04D0  CBE4        611     SET   4,H
04D2  CBE5        612     SET   4,L
04D4  CBEE        613     SET   5,(HL)
04D6  DDCB05EE    614     SET   5,(IX+IND)
04DA  FDCB05EE    615     SET   5,(IY+IND)
04DE  CBEF        616     SET   5,A
04E0  CBE8        617     SET   5,B
04E2  CBE9        618     SET   5,C
04E4  CBEA        619     SET   5,D
04E6  CBEB        620     SET   5,E
04E8  CBEC        621     SET   5,H
04EA  CBED        622     SET   5,L
04EC  CBF6        623     SET   6,(HL)
04EE  DDCB05F6    624     SET   6,(IX+IND)
04F2  FDCB05F6    625     SET   6,(IY+IND)
04F6  CBF7        626     SET   6,A
04F8  CBF0        627     SET   6,B
04FA  CBF1        628     SET   6,C
04FC  CBF2        629     SET   6,D
04FE  CBF3        630     SET   6,E
0500  CBF4        631     SET   6,H
0502  CBF5        632     SET   6,L
0504  CBFE        633     SET   7,(HL)
0506  DDCB05FE    634     SET   7,(IX+IND)
050A  FDCB05FE    635     SET   7,(IY+IND)
050E  CBFF        636     SET   7,A
0510  CBF8        637     SET   7,B
0512  CBF9        638     SET   7,C
0514  CBFA        639     SET   7,D
0516  CBFB        640     SET   7,E
0518  CBFC        641     SET   7,H
051A  CBFD        642     SET   7,L
051C  CB26        643     SLA   (HL)
051E  DDCB0526    644     SLA   (IX+IND)
0522  FDCB0526    645     SLA   (IY+IND)
0526  CB27        646     SLA   A
0528  CB20        647     SLA   B
052A  CB21        648     SLA   C
052C  CB22        649     SLA   D
052E  CB23        650     SLA   E
0530  CB24        651     SLA   H
0532  CB25        652     SLA   L
0534  CB2E        653     SRA   (HL)
0536  DDCB052E    654     SRA   (IX+IND)
053A  FDCB052E    655     SRA   (IY+IND)
053E  CB2F        656     SRA   A
0540  CB28        657     SRA   B
0542  CB29        658     SRA   C
0544  CB2A        659     SRA   D
0546  CB2B        660     SRA   E
0548  CB2C        661     SRA   H
054A  CB2D        662     SRA   L
054C  CB3E        663     SRL   (HL)
054E  DDCB053E    664     SRL   (IX+IND)
0552  FDCB053E    665     SRL   (IY+IND)
0556  CB3F        666     SRL   A
0558  CB38        667     SRL   B
055A  CB39        668     SRL   C
055C  CB3A        669     SRL   D
055E  CB3B        670     SRL   E
0560  CB3C        671     SRL   H
0562  CB3D        672     SRL   L
0564  96          673     SUB   (HL)
0565  DD9605      674     SUB   (IX+IND)
0568  FD9605      675     SUB   (IY+IND)
056B  97          676     SUB   A
056C  90          677     SUB   B
056D  91          678     SUB   C
056E  92          679     SUB   D
056F  93          680     SUB   E
0570  94          681     SUB   H
0571  95          682     SUB   L
0572  D620        683     SUB   N
0574  AE          684     XOR   (HL)
0575  DDAE05      685     XOR   (IX+IND)
0578  FDAE05      686     XOR   (IY+IND)
057B  AF          687     XOR   A
057C  A8          688     XOR   B
057D  A9          689     XOR   C
057E  AA          690     XOR   D
```

```
LOC   OBJ CODE  STMT SOURCE STATEMENT

057F  AB          691       XOR   E
0580  AC          692       XOR   H
0581  AD          693       XOR   L
0582  EE20        694       XOR   N
0584              695 NN    DEFS   2
                  696 IND   EQU    5
                  697 M     EQU   10H
                  698 N     EQU   20H
                  699 DIS   EQU   30H
                  700       END
```

APPENDIX C
INSTRUCTION SET NUMERICAL ORDER

```
                Z-80 CROSS ASSEMBLER    VERSION  1.06  OF 06/18/76
 07/09/76  10:20:50              .OPCODE  LISTING
 LOC     OBJ CODE    STMT SOURCE STATEMENT

0000    00            1       NOP
0001    018405        2       LD BC,NN
0004    02            3       LD (BC),A
0005    03            4       INC BC
0006    04            5       INC B
0007    05            6       DEC B
0008    0620          7       LD B,N
000A    07            8       RLCA
0008    08            9       EX AF,AF'
000C    09           10       ADD HL,BC
000D    0A           11       LD A,(BC)
000E    0B           12       DEC BC
000F    0C           13       INC C
0010    0D           14       DEC C
0011    0E20         15       LD C,N
0013    0F           16       RRCA
0014    102E         17       DJNZ DIS
0016    118405       18       LD DE,NN
0019    12           19       LD (DE),A
001A    13           20       INC DE
001B    14           21       INC D
001C    15           22       DEC D
001D    1620         23       LD D,N
001F    17           24       RLA
0020    182E         25       JR DIS
0022    19           26       ADD HL,DE
0023    1A           27       LD A,(DE)
0024    1B           28       DEC DE
0025    1C           29       INC E
0026    1D           30       DEC E
0027    1E20         31       LD E,N
0029    1F           32       RRA
002A    202E         33       JR NZ,DIS
002C    218405       34       LD HL,NN
002F    228405       35       LD (NN),HL
0032    23           36       INC HL
0033    24           37       INC H
0034    25           38       DEC H
0035    2620         39       LD H,N
0037    27           40       DAA
0038    282E         41       JR Z,DIS
003A    29           42       ADD HL,HL
003B    2A8405       43       LD HL,(NN)
003E    2B           44       DEC HL
003F    2C           45       INC L
0040    2D           46       DEC L
0041    2E20         47       LD L,N
0043    2F           48       CPL
0044    302E         49       JR NC,DIS
0046    318405       50       LD SP,NN
0049    328405       51       LD (NN),A
004C    33           52       INC SP
004D    34           53       INC (HL)
004E    35           54       DEC (HL)
004F    3620         55       LD (HL),N
0051    37           56       SCF
0052    382E         57       JR C,DIS
0054    39           58       ADD HL,SP
0055    3A8405       59       LD A,(NN)
0058    3B           60       DEC SP
0059    3C           61       INC A
005A    3D           62       DEC A
005B    3E20         63       LD A,N
005D    3F           64       CCF
005E    40           65       LD B,B
005F    41           66       LD B,C
0060    42           67       LD B,D
0061    43           68       LD B,E
0062    44           69       LD B,H,NN

0063    45           70       LD B,L
0064    46           71       LD B,(HL)
0065    47           72       LD B,A
0066    48           73       LD C,B
0067    49           74       LD C,C
0068    4A           75       LD C,D
0069    4B           76       LD C,E
006A    4C           77       LD C,H
006B    4D           78       LD C,L
006C    4E           79       LD C,(HL)
006D    4F           80       LD C,A
006E    50           81       LD D,B
006F    51           82       LD D,C
0070    52           83       LD D,D
0071    53           84       LD D,E
0072    54           85       LD D,H
0073    55           86       LD D,L
0074    56           87       LD D,(HL)
0075    57           88       LD D,A
0076    58           89       LD E,B
0077    59           90       LD E,C
0078    5A           91       LD E,D
0079    5B           92       LD E,E
007A    5C           93       LD E,H
007B    5D           94       LD E,L
007C    5E           95       LD E,(HL)
007D    5F           96       LD E,A
007E    60           97       LD H,B
007F    61           98       LD H,C
0080    62           99       LD H,D
0081    63          100       LD H,E
0082    64          101       LD H,H
0083    65          102       LD H,L
0084    66          103       LD H,(HL)
0085    67          104       LD H,A
0086    68          105       LD L,B
0087    69          106       LD L,C
0088    6A          107       LD L,D
0089    6B          108       LD L,E
008A    6C          109       LD L,H
008B    6D          110       LD L,L
008C    6E          111       LD L,(HL)
008D    6F          112       LD L,A
008E    70          113       LD (HL),B
008F    71          114       LD (HL),C
0090    72          115       LD (HL),D
0091    73          116       LD (HL),E
0092    74          117       LD (HL),H
0093    75          118       LD (HL),L
0094    76          119       HALT
0095    77          120       LD (HL),A
0096    78          121       LD A,B
0097    79          122       LD A,C
0098    7A          123       LD A,D
0099    7B          124       LD A,E
009A    7C          125       LD A,H
009B    7D          126       LD A,L
009C    7E          127       LD A,(HL)
009D    7F          128       LD A,A
009E    80          129       ADD A,B
009F    81          130       ADD A,C
00A0    82          131       ADD A,D
00A1    83          132       ADD A,E
00A2    84          133       ADD A,H
00A3    85          134       ADD A,L
00A4    86          135       ADD A,(HL)
00A5    87          136       ADD A,A
00A6    88          137       ADC A,B
00A7    89          138       ADC A,C
```

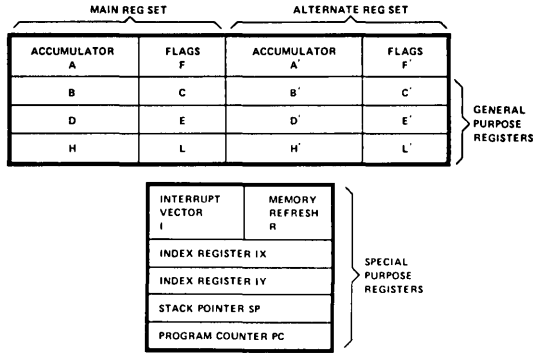
LOC	OBJ CODE	STMT	SOURCE STATEMENT
00A8	8A	139	ADC A,D
00A9	8B	140	ADC A,E
00AA	8C	141	ADC A,H
00AB	8D	142	ADC A,L
00AC	8E	143	ADC A,(HL)
00AD	8F	144	ADC A,A
00AE	90	145	SUB B
00AF	91	146	SUB C
00B0	92	147	SUB D
00B1	93	148	SUB E
00B2	94	149	SUB H
00B3	95	150	SUB L
00B4	96	151	SUB (HL)
00B5	97	152	SUB A
00B6	98	153	SBC A,B
00B7	99	154	SBC A,C
00B8	9A	155	SBC A,D
00B9	9B	156	SBC A,E
00BA	9C	157	SBC A,H
00BB	9D	158	SBC A,L
00BC	9E	159	SBC A,(HL)
00BD	9F	160	SBC A,A
00BE	A0	161	AND B
00BF	A1	162	AND C
00C0	A2	163	AND D
00C1	A3	164	AND E
00C2	A4	165	AND H
00C3	A5	166	AND L
00C4	A6	167	AND (HL)
00C5	A7	168	AND A
00C6	A8	169	XOR B
00C7	A9	170	XOR C
00C8	AA	171	XOR D
00C9	AB	172	XOR E
00CA	AC	173	XOR H
00CB	AD	174	XOR L
00CC	AE	175	XOR (HL)
00CD	AF	176	XOR A
00CE	B0	177	OR B
00CF	B1	178	OR C
00D0	B2	179	OR D
00D1	B3	180	OR E
00D2	B4	181	OR H
00D3	B5	182	OR L
00D4	B6	183	OR (HL)
00D5	B7	184	OR A
00D6	B8	185	CP B
00D7	B9	186	CP C
00D8	BA	187	CP D
00D9	BB	188	CP E
00DA	BC	189	CP H
00DB	BD	190	CP L
00DC	BE	191	CP (HL)
00DD	BF	192	CP A
00DE	C0	193	RET NZ
00DF	C1	194	POP BC
00E0	C28405	195	JP NZ,NN
00E3	C38405	196	JP NN
00E6	C48405	197	CALL NZ,NN
00E9	C5	198	PUSH BC
00EA	C620	199	ADD A,N
00EC	C7	200	RST 0
00ED	C8	201	RET Z
00EE	C9	202	RET
00EF	CA8405	203	JP Z,NN
00F2	CC8405	204	CALL Z,NN
00F5	CD8405	205	CALL NN
00F8	CE20	206	ADC A,N
00FA	CF	207	RST 8
00FB	D0	208	RET NC
00FC	D1	209	POP DE
00FD	D28405	210	JP NC,NN
0100	D320	211	OUT N,A
0102	D48405	212	CALL NC,NN
0105	D5	213	PUSH DE
0106	D620	214	SUB N
0108	D7	215	RST 10H
0109	D8	216	RET C
010A	D9	217	EXX
010B	DA8405	218	JP C,NN
010E	DB20	219	IN A,N
0110	DC8405	220	CALL C,NN
0113	DE20	221	SBC A,N
0115	DF	222	RST 18H
0116	E0	223	RET PO
0117	E1	224	POP HL
0118	E28405	225	JP PO,NN
011B	E3	226	EX (SP),HL
011C	E48405	227	CALL PO,NN
011F	E5	228	PUSH HL
0120	E620	229	AND N
0122	E7	230	RST 20H
0123	E8	231	RET PE
0124	E9	232	JP (HL)
0125	EA8405	233	JP PE,NN
0128	EB	234	EX DE,HL
0129	EC8405	235	CALL PE,NN
012C	EE20	236	XOR N
012E	EF	237	RST 28H
012F	F0	238	RET P
0130	F1	239	POP AF
0131	F28405	240	JP P,NN
0134	F3	241	DI
0135	F48405	242	CALL P,NN
0138	F5	243	PUSH AF
0139	F620	244	OR N
013B	F7	245	RST 30H
013C	F8	246	RET M
013D	F9	247	LD SP,HL
013E	FA8405	248	JP M,NN
0141	FB	249	EI
0142	FC8405	250	CALL M,NN
0145	FE20	251	CP N
0147	FF	252	RST 38H
0148	CB00	253	RLC B
014A	CB01	254	RLC C
014C	CB02	255	RLC D
014E	CB03	256	RLC E
0150	CB04	257	RLC H
0152	CB05	258	RLC L
0154	CB06	259	RLC (HL)
0156	CB07	260	RLC A
0158	CB08	261	RRC B
015A	CB09	262	RRC C
015C	CB0A	263	RRC D
015E	CB0B	264	RRC E
0160	CB0C	265	RRC H
0162	CB0D	266	RRC L
0164	CB0E	267	RRC (HL)
0166	CB0F	268	RRC A
0168	CB10	269	RL B
016A	CB11	270	RL C
016C	CB12	271	RL D
016E	CB13	272	RL E
0170	CB14	273	RL H
0172	CB15	274	RL L
0174	CB16	275	RL (HL)
0176	CB17	276	RL A

```
 LOC    OBJ CODE  STMT SOURCE STATEMENT
0178    CB18       277      RR B
017A    CB19       278      RR C
017C    CB1A       279      RR D
017E    CB1B       280      RR E
0180    CB1C       281      RR H
0182    CB1D       282      RR L
0184    CB1E       283      RR (HL)
0186    CB1F       284      RR A
0188    CB20       285      SLA B
018A    CB21       286      SLA C
018C    CB22       287      SLA D
018E    CB23       288      SLA E
0190    CB24       289      SLA H
0192    CB25       290      SLA L
0194    CB26       291      SLA (HL)
0196    CB27       292      SLA A
0198    CB28       293      SRA B
019A    CB29       294      SRA C
019C    CB2A       295      SRA D
019E    CB2B       296      SRA E
01A0    CB2C       297      SRA H
01A2    CB2D       298      SRA L
01A4    CB2E       299      SRA (HL)
01A6    CB2F       300      SRA A
01A8    CB38       301      SRL B
01AA    CB39       302      SRL C
01AC    CB3A       303      SRL D
01AE    CB3B       304      SRL E
01B0    CB3C       305      SRL H
01B2    CB3D       306      SRL L
01B4    CB3E       307      SRL (HL)
01B6    CB3F       308      SRL A
01B8    CB40       309      BIT 0,B
01BA    CB41       310      BIT 0,C
01BC    CB42       311      BIT 0,D
01BE    CB43       312      BIT 0,E
01C0    CB44       313      BIT 0,H
01C2    CB45       314      BIT 0,L
01C4    CB46       315      BIT 0,(HL)
01C6    CB47       316      BIT 0,A
01C8    CB48       317      BIT 1,B
01CA    CB49       318      BIT 1,C
01CC    CB4A       319      BIT 1,D
01CE    CB4B       320      BIT 1,E
01D0    CB4C       321      BIT 1,H
01D2    CB4D       322      BIT 1,L
01D4    CB4E       323      BIT 1,(HL)
01D6    CB4F       324      BIT 1,A
01D8    CB50       325      BIT 2,B
01DA    CB51       326      BIT 2,C
01DC    CB52       327      BIT 2,D
01DE    CB53       328      BIT 2,E
01E0    CB54       329      BIT 2,H
01E2    CB55       330      BIT 2,L
01E4    CB56       331      BIT 2,(HL)
01E6    CB57       332      BIT 2,A
01E8    CB58       333      BIT 3,B
01EA    CB59       334      BIT 3,C
01EC    CB5A       335      BIT 3,D
01EE    CB5B       336      BIT 3,E
01F0    CB5C       337      BIT 3,H
01F2    CB5D       338      BIT 3,L
01F4    CB5E       339      BIT 3,(HL)
01F6    CB5F       340      BIT 3,A
01F8    CB60       341      BIT 4,B
01FA    CB61       342      BIT 4,C
01FC    CB62       343      BIT 4,D
01FE    CB63       344      BIT 4,E
0200    CB64       345      BIT 4,H
0202    CB65       346      BIT 4,L
0204    CB66       347      BIT 4,(HL)
0206    CB67       348      BIT 4,A
0208    CB68       349      BIT 5,B
020A    CB69       350      BIT 5,C
020C    CB6A       351      BIT 5,D
020E    CB6B       352      BIT 5,E
0210    CB6C       353      BIT 5,H
0212    CB6D       354      BIT 5,L
0214    CB6E       355      BIT 5,(HL)
0216    CB6F       356      BIT 5,A
0218    CB70       357      BIT 6,B
021A    CB71       358      BIT 6,C
021C    CB72       359      BIT 6,D
021E    CB73       360      BIT 6,E
0220    CB74       361      BIT 6,H
0222    CB75       362      BIT 6,L
0224    CB76       363      BIT 6,(HL)
0226    CB77       364      BIT 6,A
0228    CB78       365      BIT 7,B
022A    CB79       366      BIT 7,C
022C    CB7A       367      BIT 7,D
022E    CB7B       368      BIT 7,E
0230    CB7C       369      BIT 7,H
0232    CB7D       370      BIT 7,L
0234    CB7E       371      BIT 7,(HL)
0236    CB7F       372      BIT 7,A
0238    CB80       373      RES 0,B
023A    CB81       374      RES 0,C
023C    CB82       375      RES 0,D
023E    CB83       376      RES 0,E
0240    CB84       377      RES 0,H
0242    CB85       378      RES 0,L
0244    CB86       379      RES 0,(HL)
0246    CB87       380      RES 0,A
0248    CB88       381      RES 1,B
024A    CB89       382      RES 1,C
024C    CB8A       383      RES 1,D
024E    CB8B       384      RES 1,E
0250    CB8C       385      RES 1,H
0252    CB8D       386      RES 1,L
0254    CB8E       387      RES 1,(HL)
0256    CB8F       388      RES 1,A
0258    CB90       389      RES 2,B
025A    CB91       390      RES 2,C
025C    CB92       391      RES 2,D
025E    CB93       392      RES 2,E
0260    CB94       393      RES 2,H
0262    CB95       394      RES 2,L
0264    CB96       395      RES 2,(HL)
0266    CB97       396      RES 2,A
0268    CB98       397      RES 3,B
026A    CB99       398      RES 3,C
026C    CB9A       399      RES 3,D
026E    CB9B       400      RES 3,E
0270    CB9C       401      RES 3,H
0272    CB9D       402      RES 3,L
0274    CB9E       403      RES 3,(HL)
0276    CB9F       404      RES 3,A
0278    CBA0       405      RES 4,B
027A    CBA1       406      RES 4,C
027C    CBA2       407      RES 4,D
027E    CBA3       408      RES 4,E
0280    CBA4       409      RES 4,H
0282    CBA5       410      RES 4,L
0284    CBA6       411      RES 4,(HL)
0286    CBA7       412      RES 4,A
0288    CBA8       413      RES 5,B
028A    CBA9       414      RES 5,C
```

```
LOC  OBJ CODE  STMT SOURCE STATEMENT
028C  CBAA      415      RES 5,D
028E  CBAB      416      RES 5,E
0290  CBAC      417      RES 5,H
0292  CBAD      418      RES 5,L
0294  CBAE      419      RES 5,(HL)
0296  CBAF      420      RES 5,A
0298  CBB0      421      RES 6,B
029A  CBB1      422      RES 6,C
029C  CBB2      423      RES 6,D
029E  CBB3      424      RES 6,E
02A0  CBB4      425      RES 6,H
02A2  CBB5      426      RES 6,L
02A4  CBB6      427      RES 6,(HL)
02A6  CBB7      428      RES 6,A
02A8  CBB8      429      RES 7,B
02AA  CBB9      430      RES 7,C
02AC  CBBA      431      RES 7,D
02AE  CBBB      432      RES 7,E
02B0  CBBC      433      RES 7,H
02B2  CBBD      434      RES 7,L
02B4  CBBE      435      RES 7,(HL)
02B6  CBBF      436      RES 7,A
02B8  CBC0      437      SET 0,B
02BA  CBC1      438      SET 0,C
02BC  CBC2      439      SET 0,D
02BE  CBC3      440      SET 0,E
02C0  CBC4      441      SET 0,H
02C2  CBC5      442      SET 0,L
02C4  CBC6      443      SET 0,(HL)
02C6  CBC7      444      SET 0,A
02C8  CBC8      445      SET 1,B
02CA  CBC9      446      SET 1,C
02CC  CBCA      447      SET 1,D
02CE  CBCB      448      SET 1,E
02D0  CBCC      449      SET 1,H
02D2  CBCD      450      SET 1,L
02D4  CBCE      451      SET 1,(HL)
02D6  CBCF      452      SET 1,A
02D8  CBD0      453      SET 2,B
02DA  CBD1      454      SET 2,C
02DC  CBD2      455      SET 2,D
02DE  CBD3      456      SET 2,E
02E0  CBD4      457      SET 2,H
02E2  CBD5      458      SET 2,L
02E4  CBD6      459      SET 2,(HL)
02E6  CBD7      460      SET 2,A
02E8  CBD8      461      SET 3 B
02EA  CBD9      462      SET 3,C
02EC  CBDA      463      SET 3,D
02EE  CBDB      464      SET 3,E
02F0  CBDC      465      SET 3,H
02F2  CBDD      466      SET 3,L
02F4  CBDE      467      SET 3,(HL)
02F6  CBDF      468      SET 3,A
02F8  CBE0      469      SET 4,B
02FA  CBE1      470      SET 4,C
02FC  CBE2      471      SET 4,D
02FE  CBE3      472      SET 4,E
0300  CBE4      473      SET 4,H
0302  CBE5      474      SET 4,L
0304  CBE6      475      SET 4,(HL)
0306  CBE7      476      SET 4,A
0308  CBE8      477      SET 5,B
030A  CBE9      478      SET 5,C
030C  CBEA      479      SET 5,D
030E  CBEB      480      SET 5,E
0310  CBEC      481      SET 5,H
0312  CBED      482      SET 5,L
0314  CBEE      483      SET 5,(HL)
0316  CBEF      484      SET 5,A
0318  CBF0      485      SET 6,B
031A  CBF1      486      SET 6,C
031C  CBF2      487      SET 6,D
031E  CBF3      488      SET 6,E
0320  CBF4      489      SET 6,H
0322  CBF5      490      SET 6,L
0324  CBF6      491      SET 6,(HL)
0326  CBF7      492      SET 6,A
0328  CBF8      493      SET 7,B
032A  CBF9      494      SET 7,C
032C  CBFA      495      SET 7,D
032E  CBFB      496      SET 7,E
0330  CBFC      497      SET 7,H
0332  CBFD      498      SET 7,L
0334  CBFE      499      SET 7,(HL)
0336  CBFF      500      SET 7,A
0338  DD09      501      ADD IX,BC
033A  DD19      502      ADD IX,DE
033C  DD218405  503      LD IX,NN
0340  DD228405  504      LD (NN),IX
0344  DD23      505      INC IX
0346  DD29      506      ADD IX,IX
0348  DD2A8405  507      LD IX,(NN)
034C  DD2B      508      DEC IX
034E  DD3405    509      INC (IX+IND)
0351  DD3505    510      DEC (IX+IND)
0354  DD360520  511      LD (IX+IND),N
0358  DD39      512      ADD IX,SP
035A  DD4605    513      LD B,(IX+IND)
035D  DD4E05    514      LD C,(IX+IND)
0360  DD5605    515      LD D,(IX+IND)
0363  DD5E05    516      LD E,(IX+IND)
0366  DD6605    517      LD H,(IX+IND)
0369  DD6E05    518      LD L,(IX+IND)
036C  DD7005    519      LD (IX+IND),B
036F  DD7105    520      LD (IX+IND),C
0372  DD7205    521      LD (IX+IND),D
0375  DD7305    522      LD (IX+IND),E
0378  DD7405    523      LD (IX+IND),H
037B  DD7505    524      LD (IX+IND),L
037E  DD7705    525      LD (IX+IND),A
0381  DD7E05    526      LD A,(IX+IND)
0384  DD8605    527      ADD A,(IX+IND)
0387  DD8E05    528      ADC A,(IX+IND)
038A  DD9605    529      SUB (IX+IND)
038D  DD9E05    530      SBC A,(IX+IND)
0390  DDA605    531      AND (IX+IND)
0393  DDAE05    532      XOR (IX+IND)
0396  DDB605    533      OR (IX+IND)
0399  DDBE05    534      CP (IX+IND)
039C  DDE1      535      POP IX
039E  DDE3      536      EX (SP),IX
03A0  DDE5      537      PUSH IX
03A2  DDE9      538      JP (IX)
03A4  DDF9      539      LD SP,IX
03A6  DDCB0506  540      RLC (IX+IND)
03AA  DDCB050E  541      RRC (IX+IND)
03AE  DDCB0516  542      RL (IX+IND)
03B2  DDCB051E  543      RR (IX+IND)
03B6  DDCB0526  544      SLA (IX+IND)
03BA  DDCB052E  545      SRA (IX+IND)
03BE  DDCB053E  546      SRL (IX+IND)
03C2  DDCB0546  547      BIT 0,(IX+IND)
03C6  DDCB054E  548      BIT 1,(IX+IND)
03CA  DDCB0556  549      BIT 2,(IX+IND)
03CE  DDCB055E  550      BIT 3,(IX+IND)
03D2  DDCB0566  551      BIT 4,(IX+IND)
03D6  DDCB056E  552      BIT 5,(IX+IND)
```

```
LOC   OBJ CODE    STMT SOURCE STATEMENT
03DA  DDCB0576     553      BIT 6,(IX+IND)
03DE  DDCB057E     554      BIT 7,(IX+IND)
03E2  DDCB0586     555      RES 0,(IX+IND)
03E6  DDCB058E     556      RES 1,(IX+IND)
03EA  DDCB0596     557      RES 2,(IX+IND)
03EE  DDCB059E     558      RES 3,(IX+IND)
03F2  DDCB05A6     559      RES 4,(IX+IND)
03F6  DDCB05AE     560      RES 5,(IX+IND)
03FA  DDCB05B6     561      RES 6,(IX+IND)
03FE  DDCB05BE     562      RES 7,(IX+IND)
0402  DDCB05C6     563      SET 0,(IX+IND)
0406  DDCB05CE     564      SET 1,(IX+IND)
040A  DDCB05D6     565      SET 2,(IX+IND)
040E  DDCB05DE     566      SET 3,(IX+IND)
0412  DDCB05E6     567      SET 4,(IX+IND)
0416  DDCB05EE     568      SET 5,(IX+IND)
041A  DDCB05F6     569      SET 6,(IX+IND)
041E  DDCB05FE     570      SET 7,(IX+IND)
0422  ED40         571      IN B,(C)
0424  ED41         572      OUT (C),B
0426  ED42         573      SBC HL,BC
0428  ED438405     574      LD (NN),BC
042C  ED44         575      NEG
042E  ED45         576      RETN
0430  ED46         577      IM 0
0432  ED47         578      LD I,A
0434  ED48         579      IN C,(C)
0436  ED49         580      OUT (C),C
0438  ED4A         581      ADC HL,BC
043A  ED4B8405     582      LD BC,(NN)
043E  ED4D         583      RETI
0440  ED50         584      IN D,(C)
0442  ED51         585      OUT (C),D
0444  ED52         586      SBC HL,DE
0446  ED538405     587      LD (NN),DE
044A  ED56         588      IM 1
044C  ED57         589      LD A,I
044E  ED58         590      IN E,(C)
0450  ED59         591      OUT (C),E
0452  ED5A         592      ADC HL,DE
0454  ED5B8405     593      LD DE,(NN)
0458  ED5E         594      IM 2
045A  ED60         595      IN H,(C)
045C  ED61         596      OUT (C),H
045E  ED62         597      SBC HL,HL
0460  ED67         598      RRD
0462  ED68         599      IN L,(C)
0464  ED69         600      OUT (C),L
0466  ED6A         601      ADC HL,HL
0468  ED6F         602      RLD
046A  ED72         603      SBC HL,SP
046C  ED738405     604      LD (NN),SP
0470  ED78         605      IN A,(C)
0472  ED79         606      OUT (C),A
0474  ED7A         607      ADC HL,SP
0476  ED7B8405     608      LD SP,(NN)
047A  EDA0         609      LDI
047C  EDA1         610      CPI
047E  EDA2         611      INI
0480  EDA3         612      OUTI
0482  EDA8         613      LDD
0484  EDA9         614      CPD
0486  EDAA         615      IND
0488  EDAB         616      OUTD
048A  EDB0         617      LDIR
048C  EDB1         618      CPIR
048E  EDB2         619      INIR
0490  EDB3         620      OTIR
0492  EDB8         621      LDDR
0494  EDB9         622      CPDR
0496  EDBA         623      INDR
0498  EDBB         624      OTDR
049A  FD09         625      ADD IY,BC
049C  FD19         626      ADD IY,DE
049E  FD218405     627      LD IY,NN
04A2  FD228405     628      LD (NN),IY
04A6  FD23         629      INC IY
04A8  FD29         630      ADD IY,IY
04AA  FD2A8405     631      LD IY,(NN)
04AE  FD2B         632      DEC IY
04B0  FD3405       633      INC (IY+IND)
04B3  FD3505       634      DEC (IY+IND)
04B6  FD360520     635      LD (IY+IND),N
04BA  FD39         636      ADD IY,SP
04BC  FD4605       637      LD B,(IY+IND)
04BF  FD4E05       638      LD C,(IY+IND)
04C2  FD5605       639      LD D,(IY+IND)
04C5  FD5E05       640      LD E,(IY+IND)
04C8  FD6605       641      LD H,(IY+IND)
04CB  FD6E05       642      LD L,(IY+IND)
04CE  FD7005       643      LD (IY+IND),B
04D1  FD7105       644      LD (IY+IND),C
04D4  FD7205       645      LD (IY+IND),D
04D7  FD7305       646      LD (IY+IND),E
04DA  FD7405       647      LD (IY+IND),H
04DD  FD7505       648      LD (IY+IND),L
04E0  FD7705       649      LD (IY+IND),A
04E3  FD7E05       650      LD A,(IY+IND)
04E6  FD8605       651      ADD A,(IY+IND)
04E9  FD8E05       652      ADC A,(IY+IND)
04EC  FD9605       653      SUB (IY+IND)
04EF  FD9E05       654      SBC A,(IY+IND)
04F2  FDA605       655      AND (IY+IND)
04F5  FDAE05       656      XOR (IY+IND)
04F8  FDB605       657      OR (IY+IND)
04FB  FDBE05       658      CP (IY+IND)
04FE  FDE1         659      POP IY
0500  FDE3         660      EX (SP),IY
0502  FDE5         661      PUSH IY
0504  FDE9         662      JP (IY)
0506  FDF9         663      LD SP,IY
0508  FDCB0506     664      RLC (IY+IND)
050C  FDCB050E     665      RRC (IY+IND)
0510  FDCB0516     666      RL (IY+IND)
0514  FDCB051E     667      RR (IY+IND)
0518  FDCB0526     668      SLA (IY+IND)
051C  FDCB052E     669      SRA (IY+IND)
0520  FDCB053E     670      SRL (IY+IND)
0524  FDCB0546     671      BIT 0,(IY+IND)
0528  FDCB054E     672      BIT 1,(IY+IND)
052C  FDCB0556     673      BIT 2,(IY+IND)
0530  FDCB055E     674      BIT 3,(IY+IND)
0534  FDCB0566     675      BIT 4,(IY+IND)
0538  FDCB056E     676      BIT 5,(IY+IND)
053C  FDCB0576     677      BIT 6,(IY+IND)
0540  FDCB057E     678      BIT 7,(IY+IND)
0544  FDCB0586     679      RES 0,(IY+IND)
0548  FDCB058E     680      RES 1,(IY+IND)
054C  FDCB0596     681      RES 2,(IY+IND)
0550  FDCB059E     682      RES 3,(IY+IND)
0554  FDCB05A6     683      RES 4,(IY+IND)
0558  FDCB05AE     684      RES 5,(IY+IND)
055C  FDCB05B6     685      RES 6,(IY+IND)
0560  FDCB05BE     686      RES 7,(IY+IND)
0564  FDCB05C6     687      SET 0,(IY+IND)
0568  FDCB05CE     688      SET 1,(IY+IND)
056C  FDCB05D6     689      SET 2,(IY+IND)
0570  FDCB05DE     690      SET 3,(IY+IND)
```

```
               Z-80 CROSS ASSEMBLER     VERSION  1.06   OF 06/18/76
 07/09/76  10:20:50              .OPCODE  LISTING
 LOC     OBJ CODE   STMT SOURCE STATEMENT

0574    FDCB05E6     691        SET 4,(IY+IND)
0578    FDCB05EE     692        SET 5,(IY+IND)
057C    FDCB05F6     693        SET 6,(IY+IND)
0580    FDCB05FE     694        SET 7,(IY+IND)
0584                 695 NN     DEFS 2
                     696 IND    EQU  5
                     697 M      EQU 10H
                     698 N      EQU 20H
                     699 DIS    EQU 30H
                     700        END
```



Z80-CPU REGISTER CONFIGURATION

HEXADECIMAL COLUMNS					
6	5	4	3	2	1
HEX = DEC	HEX = DEC	HEX = DEC	HEX = DEC	HEX = DEC	HEX = DEC
0 0	0 0	0 0	0 0	0 0	0 0
1 1,048,576	1 65,536	1 4,096	1 256	1 16	1 1
2 2,097,152	2 131,072	2 8,192	2 512	2 32	2 2
3 3,145,728	3 196,608	3 12,288	3 768	3 48	3 3
4 4,194,304	4 262,144	4 16,384	4 1,024	4 64	4 4
5 5,242,880	5 327,680	5 20,480	5 1,280	5 80	5 5
6 6,291,456	6 393,216	6 24,576	6 1,536	6 96	6 6
7 7,340,032	7 458,752	7 28,672	7 1,792	7 112	7 7
8 8,388,608	8 524,288	8 32,768	8 2,048	8 128	8 8
9 9,437,184	9 589,824	9 36,864	9 2,304	9 144	9 9
A 10,485,760	A 655,360	A 40,960	A 2,560	A 160	A 10
B 11,534,336	B 720,896	B 45,056	B 2,816	B 176	B 11
C 12,582,912	C 786,432	C 49,152	C 3,072	C 192	C 12
D 13,631,488	D 851,968	D 53,248	D 3,328	D 208	D 13
E 14,680,064	E 917,504	E 57,344	E 3,584	E 224	E 14
F 15,728,640	F 983,040	F 61,440	F 3,840	F 240	F 15
0123	4567	0123	4567	0123	4567
BYTE		BYTE		BYTE	

ASCII CHARACTER SET (7-BIT CODE)

LSD \ MSD	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL	DLE	SP	0	@	P		p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	$	4	D	T	d	t
5 0101	ENG	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	
E 1110	SO	RS	.	>	N	↑	n	~
F 1111	SI	VS	/	?	O	←	o	DEL

POWERS OF 2

2^n	n
256	8
512	9
1 024	10
2 048	11
4 096	12
8 192	13
16 384	14
32 768	15
65 536	16
131 072	17
262 144	18
524 288	19
1 048 576	20
2 097 152	21
4 194 304	22
8 388 608	23
16 777 216	24

$2^0 = 16^0$
$2^4 = 16^1$
$2^8 = 16^2$
$2^{12} = 16^3$
$2^{16} = 16^4$
$2^{20} = 16^5$
$2^{24} = 16^6$
$2^{28} = 16^7$
$2^{32} = 16^8$
$2^{36} = 16^9$
$2^{40} = 16^{10}$
$2^{44} = 16^{11}$
$2^{48} = 16^{12}$
$2^{52} = 16^{13}$
$2^{56} = 16^{14}$
$2^{60} = 16^{15}$

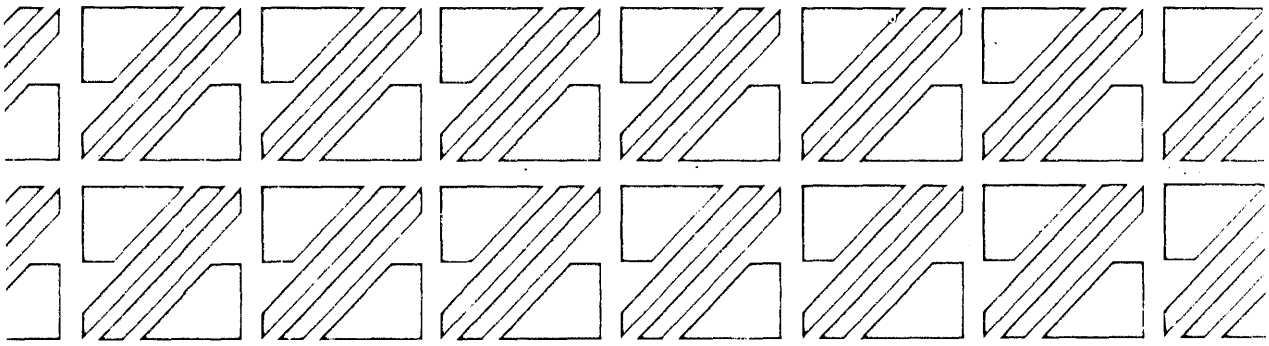
POWERS OF 16

16^n	n
1	0
16	1
256	2
4 096	3
65 536	4
1 048 576	5
16 777 216	6
268 435 456	7
4 294 967 296	8
68 719 476 736	9
1 099 511 627 776	10
17 592 186 044 416	11
281 474 976 710 656	12
4 503 599 627 370 496	13
72 057 594 037 927 936	14
1 152 921 504 606 846 976	15

Zilog

10460 Bubb Road
Cupertino, CA 95014
Tel: (408) 446-4666
TWX 910-338-7621

Price: $7.5



03-0002-01

Printed in USA