IM8000 Technical Reference Manual

Document Version: 0.5
Date: December 2025

Table of Contents

---

1. Design Overview

1.1 General Description

The IM8000 is a high-performance 16-bit microprocessor conceived as an alternate-history successor to the Zilog Z80. It preserves the Z80's programming model while extending it into a clean, orthogonal 16/32-bit instruction set. The IM8000 offers both backward compatibility and forward-looking enhancements, delivering powerful computing capabilities for advanced microcomputer applications of the 1980s.

1.2 Key Features

- **Architecture**: 16-bit word size, select 32-bit operations
- **Address Space**: 24-bit physical addressing (up to 16 MB)
- **Clock Speed**: 4-8 MHz (preliminary)
- **Instruction Set**: Z80-inspired with expanded capabilities
- **Registers**: Seven 16-bit primary registers, supporting 32-bit register pairs
- **Interrupts**: Fully vectored system with 256 interrupt vectors, Z80 IM 2 compatible

1.3 Target Applications

- Home and business microcomputers
- Industrial control systems
- Educational and development systems

- Exactly one simulator created for myself by myself

1.4 Design Objectives

- **Enhanced Performance**: Improve computational performance and memory addressing flexibility
- **Familiar Programming**: Retain and extend Z80 programming conventions
- **Future Expansion**: Support for architectural extensions and advanced variants

---

2. Architecture Overview

2.1 Functional Units

2.1.1 Execution Unit

- 16-bit ALU with 32-bit ripple carry operations
- Microcode multi-bit shifts
- Microcode multiply/divide support
- Flag generation and condition testing

2.1.2 Register File

- Seven 16-bit primary registers (A, B, C, D, E, H, L)
- Seven 16-bit alternate registers (A', B', C', D', E', H', L')
- Two 32-bit index registers (IX, IY)
- Two 32-bit alternate index registers (IX', IY')
- 32-bit stack pointer with alternate (SP, SP')
- 16-bit flag register with system flags

2.1.3 Memory Controller

- Address calculation
- Stack pointer management
- External data bus interface
- Address bus interface
- Wait state generation
- Bus arbitration logic

---

3. Programming Model

3.1 Register Set

3.1.1 Primary Registers

The IM8000 maintains the familiar Z80 register architecture while extending all registers to 16 bits. Each register can be accessed as a complete 16-bit word or as an 8-bit half:

Register	Width	Function
A	16-bit	General Purpose
B	16-bit	General Purpose
C	16-bit	General Purpose
D	16-bit	General Purpose
E	16-bit	General Purpose
H	16-bit	General Purpose
L	16-bit	General Purpose

Byte operations target the lower byte of the register, preserving the upper byte unless specified.

3.1.2 Register Pairs

Registers may be combined in pairs for memory addressing and 32-bit operations:

Pair	Registers	Width	Primary Use
AF	A + F	32-bit	State save/restore
BC	B + C	32-bit	Counter operations, loop control
DE	D + E	32-bit	Block operation source pointer
HL	H + L	32-bit	Block operation destination pointer

Register Pair Formation: Register pairs are 32-bit register views formed by concatenating two 16-bit registers. The first named register forms the high-order word and the second forms the low-order word. For example, `BC = (B << 16) | C`.

Note: `AF` forms a register pair for architectural consistency, but cannot be treated as a general purpose register. `F` is the system flag register, and only `A` is suitable for arithmetic use.

3.1.3 index registers

Register	Width	Function
IX	32-bit	Index Register X
IY	32-bit	Index Register Y

Index registers are dedicated 32-bit registers primarily used for indexed memory addressing. All indirect access through IX, IY, and SP take an additional 16-bit displacement immediately after the instruction word and before any immediate operands.

3.1.4 System Registers

Register	Width	Function	Reset Value
PC	32-bit	Program Counter	Reset Vector
SP	32-bit	Stack Pointer	0x00000000
IVB	22-bit	Interrupt Vector Base	0x000000
R	16-bit	Refresh Counter	0x0000
F	16-bit	Flag Register	0x0000

3.2 Flag Register

The flag register maintains Z80 compatibility in the lower byte while adding the Interrupt Enable flag in the upper byte:

Bit	Symbol	Name	Function
0	C	Carry	Carry out / Borrow
1	N	Add/Subtract	Operation type indicator
2	PV	Parity/Overflow	Parity (logical) / Overflow (arithmetic)
3	-	Reserved	Must be zero
4	H	Half Carry	Carry from bit 3 to 4
5	-	Reserved	Must be zero
6	Z	Zero	Result is zero
7	S	Sign	Result is negative
8	IE (IFF1)	Interrupt Enable	Maskable interrupt enable
9-15	-	Reserved	Must be zero

Note: The Half-Carry and Add/Subtract flags are included for BCD support, and are deprecated. They may be removed in later revisions.
The numeric value of `F` should not be relied on.

3.2.3 Flag Size Semantics

ALU Flags always correspond to the size of the operand being used.
For 32-bit ops, the MSB is bit 31; for 16-bit ops, bit 15; for 8-bit ops, bit 7.

Byte:

- Carry is set when bit 7 overflows to carry
- Zero is set when the 8-bit result is 0
    - Byte operations ignore the upper half of the register
- Sign is set when bit 7 is 1
  Word:
- Carry is set when bit 15 overflows to carry
- Zero is set when the 16-bit result is 0
- Sign is set when bit 15 is 1
  Dword:
- Carry is set when bit 31 overflows to carry
- Zero is set when the 32-bit result is 0
- Sign is set when bit 31 is 1

3.3 Alternate Register Set

The IM8000 provides a complete alternate register set for fast context switching:

- **Primary registers**: A', B', C', D', E', H', L', F' (16-bit each)
- **index registers**: IX', IY' (32-bit each)
- **Stack pointer**: SP' (32-bit)

Exchange instructions provide rapid context switching:

- `EX r, r'` - Exchange register with alternate
- `EXX` - Exchange `BC, DE, HL` with `BC', DE', HL'`
- `EXI` - Exchange `IX, IY, SP` with `IX', IY', SP'`

3.4 Memory Organization

3.4.1 Endianness

The IM8000 uses little-endian byte ordering for memory and I/O devices.

3.4.2 Alignment

- The IM8000 does not require alignment. However, multi-byte accesses from odd addresses require multiple bus cycles, and are slower than accesses from even addresses.

3.4.1 Memory Address Space

- **Width**: 24-bit (16 MB total)
- Mapped directly to address pin A[23:0]
- Supports ROM, RAM, and memory mapped peripherals

3.4.2 I/O Address Space

- **Width**: 16-bit (64K I/O ports)

- Accessed using `IN, OUT` instructions

---

4. Addressing Modes

The IM8000 supports eight Addressing Modes, providing flexibility while maintaining Z80 compatibility.

4.1 Implied Addressing

Format: No operands specified

Usage: Control instructions, register exchanges

Examples:

- `NOP` - No operation
- `EXX` - Exchange primary register sets
- `HALT` - Halt processor

4.2 Immediate Addressing

Format: Operand included in instruction

Sizes: 8-bit, 16-bit, or 32-bit immediate values

Examples:

- `LD A, 0x1234` - Load 16-bit immediate into A
- `ADD.b B, 0xFF` - Add 8-bit immediate to B
- `LD SP, 0x12345678` - Load 32-bit immediate into SP

4.3 Register Addressing

Format: Operand is a register

Usage: Register-to-register operations

Examples:

- `LD A, B` - Copy register B to A
- `ADD.b C, D` - Add lower byte of register D to lower byte of C
- `XOR E, H` - Exclusive-OR H with E

4.4 Indirect Addressing

Format: `(register_pair)`

Usage: Memory access via register contents

Examples:

- `LD A, (HL)` - Load from memory pointed to by HL
- `LD (DE), B` - Store B to memory pointed to by DE
- `ADD A, (BC)` - Add to A from memory pointed to by BC

4.5 Indexed Addressing

Format: `(register ± offset)`

Offset: Signed 16-bit displacement

Examples:

- `LD BC, (IX+10)` - Load from IX + 10
- `LD (SP-4), C` - Store C to SP - 4
- `BIT.B 3, (IY+0x100)` - Test bit 3 at HL + 256

4.6 Direct Addressing

Format: `(address)`

Address: 32-bit direct address

Examples:

- `LD A, (0x12345678)` - Load from direct address
- `CALL 0x1000` - Call subroutine at address
- `JP 0xFFFF0000` - Jump to direct address

4.7 Relative Addressing

Format: Signed offset from current PC

Range: ±255 bytes (8-bit displacement) or ±32767 bytes (16-bit displacement)

Examples:

- `JR Z, label` - Jump if zero to relative address
- `DJNZ loop` - Decrement and jump if not zero
- `JR -10` - Jump back 10 bytes

4.8 Stack Addressing

Format: Implied stack pointer usage

Operations: Push/pop, call/return

Examples:

- `PUSH AF` - Push AF onto stack
- `POP BC` - Pop from stack into BC
- `CALL subroutine` - Call with return address push
- `RET` - Return with address pop

---

5. Interrupt System

5.1 Interrupt Architecture

The IM8000 implements a vectored interrupt system supporting 256 interrupt vectors with full compatibility with Z80 peripherals.

5.1.1 Interrupt Types

Type	Priority	Maskable	Vector Source
Reset	1	No	`/RST` pin
Non-Maskable Interrupt	3	No	`/NMI` pin
Interrupt	4	Yes	`/INT` pin
Software Interrupt	5	No	`RST` instruction

5.1.2 Interrupt Enable Flags

- IFF1 and IFF2 are retained from the Z80
- IFF1 is exposed in the flag register as bit 8
  - Writing to IFF1 through F or via EI/DI will write to both IFF1 and IFF2
  - There is a 1 instruction delay between enabling IFF1 and the processor enabling interrupts
    - This allows for `EI : RETI` to work as a pair of instructions.
- On a non-maskable interrupt, IFF1 is copied to IFF2 and IFF1 is disabled
- On return from a non-maskable interrupt, IFF2 is copied to IFF1

5.1.3 Interrupt Vector Table

- **Location**: Base address in IVB register (22-bit)
- **Size**: 1024 bytes (256 × 4-byte entries)
- **Alignment**: 1024-byte boundary
- **Format**: Little-endian 32-bit addresses
- **Index Calculation**:
  - `(IVB << 10) + (Interrupt Number * 4)`

5.2 Interrupt Modes

5.2.1 Interrupt Mode 1 (IM 1)

- Fixed interrupt number (Vector 1)
- Compatible with simple Z80 interrupt hardware
- No vector acknowledgment required

5.2.2 Interrupt Mode 2 (IM 2)

- Vectored interrupts (256 32-bit vectors)
- Device supplies 8-bit interrupt number on data bus lines [7-0] during acknowledge
- Fully compatible with Z80 IM 2 peripherals

5.3 Interrupt Processing

5.3.1 Interrupt Detection

- Interrupts are sampled at end of each instruction.
- Read-Modify-Write operations and 32-bit operations sample interrupts at the end of full execution.
- Block operations sample interrupts at the end of each iteration.

5.3.2 Interrupt Acknowledge

- Standard Z80-compatible acknowledge cycle
- `/M1` and `/IORQ` asserted simultaneously
- 8-bit vector number read from data bus (IM 2 mode)

5.3.3 Context Preservation

- Program Counter pushed to stack
- Interrupt Enable Flag:
  - Unchanged (software interrupt)
  - Disabled (hardware interrupt)
  - Saved/Restored (non-maskable interrupt)
- Register preservation is software responsibility

5.4 Defined Vectors

Name	Vector	Description
Reset	0	System reset/power-on
Maskable Interrupt	1	IM1 hardware interrupt
Non-Maskable Interrupt	2	Non-maskable interrupt

Note: Vector 0 is not required to be the reset vector, but is a useful convention. The system reset vector is read at address `0000:0000`, which happens to overlap with vector 0 on reset.

Additionally, ports of Z80 software may assume `RST 0` resets the system.

---

6. Instruction Set Summary

Instruction Notation

Notation	Description
r	Any 16-bit primary register: A, B, C, D, E, H, L
.b	Instruction size suffix, byte operand
.w	Explicit instruction size suffix, word operand
.d	Explicit instruction size suffix, dword operand
rr	Any 32-bit address register: AF, BC, DE, HL, IX, IY, SP
n	Immediate 8-bit unsigned integer
nn	Immediate 16-bit unsigned integer
nnnn	Immediate 32-bit unsigned integer
dd	Signed 16-bit displacement
(rr)	Memory at address contained in register `rr`
(rr+dd)	Memory at address contained in `rr` plus signed displacement `dd`
(nnnn)	Memory at address given by a 32-bit immediate constant
cc	Condition code

Note: Indirect and Indexed notations are reversible. Although documented as `INS r, (rr)`, `INS (rr), r` is equally valid and selects the indirect value as the destination.

Operand Sizes

- **Byte**: 8-bit (accessed via instruction `.b` notation)
- **Word**: 16-bit (default register size)
- **Dword**: 32-bit (register pairs and index registers)

Flag Definitions

Flag	Name	Set When	Reset When
S	Sign	Result MSB is 1 (negative)	Result MSB is 0 (positive)
Z	Zero	Result equals zero	Result is non-zero
H	Half Carry	Carry/borrow from bit 3 to bit 4	No carry/borrow from bit 3
P/V	Parity/Overflow	Even parity OR signed overflow	Odd parity OR no signed overflow
N	Add/Subtract	Subtraction operation	Addition operation

Flag	Name	Set When	Reset When
C	Carry	Carry/borrow from MSB	No carry/borrow from MSB

Flag Notation:

- **✓** - Flag affected according to result
- **R** - Flag reset to 0
- **S** - Flag set to 1
- **-** - Flag unchanged
- **X** - Flag undefined

Timing Model

- **T-Cycle**: One full clock oscillation
- **M-Cycle**: Four T-Cycles

Estimation model:

- 1 base M-Cycle per instruction
- +1 M-Cycle per bus access
- I/O device access: +1 T-Cycle per word

Actual timing varies by implementation.

6.1 Data Transfer

LD - Load

Description: Copies a value from the source to the destination.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`LD r, r`	R	-	-	-	-	-	-
Immediate	`LD r, nn`	R	-	-	-	-	-	-
Indirect	`LD r, (rr)`	RM	-	-	-	-	-	-
Indexed	`LD r, (rr+dd)`	RM	-	-	-	-	-	-
Direct	`LD r, (nnnn)`	RM	-	-	-	-	-	-

Operand Sizes: Byte, Word, Dword

EX - Exchange

Description: Exchanges the contents of the source and destination.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`EX r, r`	R	*	*	*	*	*	*
Indirect	`EX r, (rr)`	RM	*	*	*	*	*	*
Indexed	`EX r, (rr+dd)`	RM	*	*	*	*	*	*

Operand Sizes: Byte, Word, Dword
Flags Affected: If `F` is a target, all flags exchange with new value.

EX r, r' - Exchange with Alternate

Description: Exchanges the contents of the destination register and its alternate register.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`EX r, r'`	UR	*	*	*	*	*	*

Operand Sizes: Word, Dword
Flags Affected: If `F` is a target, all flags exchange with new value.

EXX - Exchange Primary with Alternate

Description: Exchanges the contents of `BC, DE, HL` with `BC', DE', HL'`.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`EXX`	N	-	-	-	-	-	-

EXI - Exchange Index with Alternate

Description: Exchanges the contents of `IX, IY, SP` with `IX', IY', SP'`

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`EXI`	N	-	-	-	-	-	-

EXH - Exchange Halves

Description: Exchanges the upper half of a target with the lower half

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`EXH r`	UR	-	-	-	-	-	-
Memory	`EXH (rr)`	UM	-	-	-	-	-	-

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Indexed	`EXH (rr+dd)`	UM	-	-	-	-	-	-

Operand Sizes: Byte, Word, Dword
Flags Affected: If `F` is a target, all flags exchange with new value.

PUSH - Push

Description: Pushes the value of the target to the stack.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Stack	`PUSH r`	UR	-	-	-	-	-	-
Immediate	`PUSH nnnn`	UR	-	-	-	-	-	-

Operand Sizes: Byte, Word, Dword
Notes: Always pushes a Dword, and sign-extends smaller values to fit.

POP - Pop

Description: Pops the top of the stack into a register.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Stack	`POP rr`	UR	*	*	*	*	*	*

Operand Sizes: Dword
Flags Affected: If `AF` is the target, all flags are replaced with the popped value.

IN - Input from Port

Description: Reads the value from an I/O device to a register.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`IN r, (r)`	R	-	-	-	-	-	-
Immediate	`IN r, (n)`	R	-	-	-	-	-	-

Operand Sizes: Byte, Word

OUT - Output to Port

Description: Writes the value from a register to an I/O device.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`OUT (r), r`	R	-	-	-	-	-	-
Immediate	`OUT (n), r`	R	-	-	-	-	-	-

Operand Sizes: Byte, Word

6.2 Block Operations

LDI - Load and Increment

Description: The value at `(HL)` is copied to `(DE)`. `HL, DE` are incremented by the operand size, `BC` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`LDI.b`	N	-	-	R	✓	R	-
Implied	`LDI`	N	-	-	R	✓	R	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if `BC ≠ 0` after operation

LDIR - Load, Increment, and Repeat

Description: Repeats `LDI` until `BC = 0`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`LDIR.B`	N	-	-	R	R	R	-
Implied	`LDIR`	N	-	-	R	R	R	-

Operand Sizes: Byte, Word

LDD - Load and Decrement

Description: The value at `(HL)` is copied to `(DE)`. `HL, DE` are decremented by the operand size, `BC` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`LDD.b`	N	-	-	R	✓	R	-
Implied	`LDD`	N	-	-	R	✓	R	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if `BC ≠ 0` after operation

LDDR - Load, Decrement, and Repeat

Description: Repeats `LDD` until `BC = 0`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`LDDR.b`	N	-	-	R	R	R	-
Implied	`LDDR`	N	-	-	R	R	R	-

Operand Sizes: Byte, Word

CPI - Compare and Increment

Description: Compares the value at `(HL)` with the value in `A`. `HL` is incremented by the operand size, `BC` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`CPI.b`	N	✓	✓	✓	✓	S	-
Implied	`CPI`	N	✓	✓	✓	✓	S	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if `BC ≠ 0` after operation

CPIR - Compare, Increment, and Repeat

Description: Repeats `CPI` until `BC = 0` or `(HL) = A`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`CPIR.b`	N	✓	✓	✓	✓	S	-
Implied	`CPIR`	N	✓	✓	✓	✓	S	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if terminated by `BC = 0`, reset if terminated by match

CPD - Compare and Decrement

Description: Compares the value at `(HL)` with the value in `A`. `HL` is decremented by the operand size, `BC` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`CPD.b`	N	✓	✓	✓	✓	S	-

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`CPD`	N	✓	✓	✓	✓	S	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if `BC ≠ 0` after operation

CPDR - Compare, Decrement, and Repeat

Description: Repeats `CPD` until `BC = 0` or `(HL) = A`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`CPDR.b`	N	✓	✓	✓	✓	S	-
Implied	`CPDR`	N	✓	✓	✓	✓	S	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if terminated by `BC = 0`, reset if terminated by match

TSI - Test and Increment

Description: Tests the value at `(HL)` with the value in `A`. `HL` is incremented by the operand size, `BC` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`TSI.b`	N	✓	✓	✓	✓	S	-
Implied	`TSI`	N	✓	✓	✓	✓	S	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if `BC ≠ 0` after operation

TSIR - Test, Increment, and Repeat

Description: Repeats `TSI` until `BC = 0` or `(HL) = A`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`TSIR.b`	N	✓	✓	✓	✓	S	-
Implied	`TSIR`	N	✓	✓	✓	✓	S	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if terminated by `BC = 0`, reset if terminated by match

TSD - Test and Decrement

Description: Tests the value at `(HL)` with the value in `A`. `HL` is decremented by the operand size, `BC` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`TSD.b`	N	✓	✓	✓	✓	S	-
Implied	`TSD`	N	✓	✓	✓	✓	S	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if `BC ≠ 0` after operation

TSDR - Test, Decrement, and Repeat

Description: Repeats `TSD` until `BC = 0` or `(HL) = A`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`TSDR.b`	N	✓	✓	✓	✓	S	-
Implied	`TSDR`	N	✓	✓	✓	✓	S	-

Operand Sizes: Byte, Word
Flags: `P/V` is set if terminated by `BC = 0`, reset if terminated by match

INI - Input and Increment

Description: Reads the value from the I/O device `(C)` to `(HL)`. `HL` is incremented by the operand size, `B` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`INI.b`	N	-	✓	-	-	S	-
Implied	`INI`	N	-	✓	-	-	S	-

Operand Sizes: Byte, Word
Flags: `Z` is set if `B = 0` after operation

INIR - Input, Increment, and Repeat

Description: Repeats `INI` until `B = 0`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`INIR.b`	N	-	S	-	-	S	-

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`INIR`	N	-	S	-	-	S	-

Operand Sizes: Byte, Word

IND - Input and Decrement

Description: Reads the value from the I/O device `(C)` to `(HL)`. `HL` is decremented by the operand size, `B` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`IND.b`	N	-	✓	-	-	S	-
Implied	`IND`	N	-	✓	-	-	S	-

Operand Sizes: Byte, Word
Flags: `Z` is set if `B = 0` after operation

INDR - Input, Decrement, and Repeat

Description: Repeats `IND` until `B = 0`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`INDR.b`	N	-	S	-	-	S	-
Implied	`INDR`	N	-	S	-	-	S	-

Operand Sizes: Byte, Word

OUTI - Output and Increment

Description: Writes the value from `(HL)` to the I/O device at port `(C)`. `HL` is incremented by the operand size, `B` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`OUTI.b`	N	-	✓	-	-	S	-
Implied	`OUTI`	N	-	✓	-	-	S	-

Operand Sizes: Byte, Word
Flags: `Z` is set if `B = 0` after operation

OTIR - Output, Increment, and Repeat

Description: Repeats `OUTI` until `B = 0`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`OTIR.b`	N	-	S	-	-	S	-
Implied	`OTIR`	N	-	S	-	-	S	-

Operand Sizes: Byte, Word

OUTD - Output and Decrement

Description: Writes the value from `(HL)` to the I/O device at port `(C)`. `HL` is decremented by the operand size, `B` is decremented by 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`OUTD.b`	N	-	✓	-	-	S	-
Implied	`OUTD`	N	-	✓	-	-	S	-

Operand Sizes: Byte, Word
Flags: `Z` is set if `B = 0` after operation

OTDR - Output, Decrement, and Repeat

Description: Repeats `OUTD` until `B = 0`. Interrupts may occur while this instruction is executing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`OTDR.b`	N	-	S	-	-	S	-
Implied	`OTDR`	N	-	S	-	-	S	-

Operand Sizes: Byte, Word

6.3 Arithmetic

ADD - Add

Description: Adds the source to the destination, result is stored in destination.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`ADD r, r`	R	✓	✓	✓	✓	R	✓
Immediate	`ADD r, nn`	R	✓	✓	✓	✓	R	✓
Indirect	`ADD r, (rr)`	RM	✓	✓	✓	✓	R	✓
Indexed	`ADD r, (rr+dd)`	RM	✓	✓	✓	✓	R	✓
Direct	`ADD r, (nnnn)`	RM	✓	✓	✓	✓	R	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates signed overflow

ADC - Add with Carry

Description: Adds the source and carry to the destination, result is stored in destination.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`ADC r, r`	R	✓	✓	✓	✓	R	✓
Immediate	`ADC r, nn`	R	✓	✓	✓	✓	R	✓
Indirect	`ADC r, (rr)`	RM	✓	✓	✓	✓	R	✓
Indexed	`ADC r, (rr+dd)`	RM	✓	✓	✓	✓	R	✓
Direct	`ADC r, (nnnn)`	RM	✓	✓	✓	✓	R	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates signed overflow

SUB - Subtract

Description: Subtracts the source from the destination, result is stored in destination.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`SUB r, r`	R	✓	✓	✓	✓	S	✓
Immediate	`SUB r, nn`	R	✓	✓	✓	✓	S	✓
Indirect	`SUB r, (rr)`	RM	✓	✓	✓	✓	S	✓
Indexed	`SUB r, (rr+dd)`	RM	✓	✓	✓	✓	S	✓
Direct	`SUB r, (nnnn)`	RM	✓	✓	✓	✓	S	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates signed overflow

SBC - Subtract with Carry

Description: Subtracts the source and carry flag from the destination, result is stored in destination.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`SBC r, r`	R	✓	✓	✓	✓	S	✓
Immediate	`SBC r, nn`	R	✓	✓	✓	✓	S	✓
Indirect	`SBC r, (rr)`	RM	✓	✓	✓	✓	S	✓
Indexed	`SBC r, (rr+dd)`	RM	✓	✓	✓	✓	S	✓

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Direct	`SBC r, (nnnn)`	RM	✓	✓	✓	✓	S	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates signed overflow

CP - Compare

Description: Subtracts the source from the destination, result is discarded and flags are set.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`CP r, r`	R	✓	✓	✓	✓	S	✓
Immediate	`CP r, nn`	R	✓	✓	✓	✓	S	✓
Indirect	`CP r, (rr)`	RM	✓	✓	✓	✓	S	✓
Indexed	`CP r, (rr+dd)`	RM	✓	✓	✓	✓	S	✓
Direct	`CP r, (nnnn)`	RM	✓	✓	✓	✓	S	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates signed overflow

INC - Increment

Description: Adds 1 to the operand.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`INC r`	UR	✓	✓	✓	✓	R	-
Indirect	`INC (rr)`	UM	✓	✓	✓	✓	R	-
Indexed	`INC (rr+dd)`	UM	✓	✓	✓	✓	R	-
Direct	`INC (nnnn)`	UM	✓	✓	✓	✓	R	-

Operand Sizes: Byte, Word, Dword

DEC - Decrement

Description: Subtracts 1 from the operand.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`DEC r`	UR	✓	✓	✓	✓	S	-
Indirect	`DEC (rr)`	UM	✓	✓	✓	✓	S	-
Indexed	`DEC (rr+dd)`	UM	✓	✓	✓	✓	S	-

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Direct	`DEC (nnnn)`	UM	✓	✓	✓	✓	S	-

Operand Sizes: Byte, Word, Dword

DAA - Decimal Adjust Accumulator

Description: Adjusts A for BCD addition and subtraction operations.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`DAA`	N	✓	✓	✓	✓	-	✓

Operand Sizes: Byte
Note: Deprecated, included for Z80 compatibility

NEG - Negate

Description: Two's complement the operand. Equivalent to `0 - operand`.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`NEG r`	UR	✓	✓	✓	✓	S	✓
Indirect	`NEG (rr)`	UM	✓	✓	✓	✓	S	✓
Indexed	`NEG (rr+dd)`	UM	✓	✓	✓	✓	S	✓
Direct	`NEG (nnnn)`	UM	✓	✓	✓	✓	S	✓

Operand Sizes: Byte, Word, Dword

EXT - Sign Extend Value

Description: Sign-extends the lower half of the target to the full target width

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`EXT r`	UR	-	-	-	-	-	-
Memory	`EXT (rr)`	UM	-	-	-	-	-	-
Indexed	`EXT (rr+dd)`	UM	-	-	-	-	-	-

Operand Sizes: Word, Dword
Note:

- Word operands extend the lower byte into a word
- Dword operands extend the lower word into a dword.

- To extend a byte into a dword, two instructions are needed: `EXT r`, and `EXT rr`.

MLT - Multiply

Description: Multiplies the upper half of the target with the lower half of the target, result stored in the full target.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`MLT r`	UR	-	-	X	✓	X	✓
Memory	`MLT (rr)`	UM	-	-	X	✓	X	✓
Indexed	`MLT (rr+dd)`	UM	-	-	X	✓	X	✓

Operand Sizes: Byte, Word
Flags: `C` set if result doesn't fit in the original operand width, `P/V` set if full result is negative
Note:

- Address Register targets will multiply the upper and lower words together
  - C is set if result is larger than 65535
  - `P/V` is set if bit 31 is 1
- Register targets will multiply the upper and lower bytes together
  - C is set if result is larger than 255
  - `P/V` is set if bit 15 is 1

DIV - Unsigned Divide

Description: Divides the upper half of the target by the lower half of the target. Quotient stored in the lower half, remainder stored in the upper half.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`DIV r`	UR	X	X	X	✓	X	X
Memory	`DIV (rr)`	UM	X	X	X	✓	X	X
Indexed	`DIV (rr+dd)`	UM	X	X	X	✓	X	X

Operand Sizes: Byte, Word
Flags:

- `P/V` is set if the divisor is 0
  Note:
- Address Register targets will divide the upper word by the lower word
- Register targets will divide the upper byte by the lower byte

SDIV - Signed Divide

Description: Divides the upper half of the target by the lower half of the target as signed values. Quotient stored in the lower half, remainder stored in the upper half.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`SDIV r`	UR	X	X	X	✓	X	X
Memory	`SDIV (rr)`	UM	X	X	X	✓	X	X
Indexed	`SDIV (rr+dd)`	UM	X	X	X	✓	X	X

Operand Sizes: Byte, Word
Flags:

- `P/V` is set if the divisor is 0
  Note:
- Address Register targets will divide the upper word by the lower word
- Register targets will divide the upper byte by the lower byte

6.4 Logical

AND - Bitwise AND

Description: Logically ANDs the source and destination, result is stored in destination.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`AND r, r`	R	✓	✓	S	✓	R	R
Immediate	`AND r, nn`	R	✓	✓	S	✓	R	R
Indirect	`AND r, (rr)`	RM	✓	✓	S	✓	R	R
Indexed	`AND r, (rr+dd)`	RM	✓	✓	S	✓	R	R
Direct	`AND r, (nnnn)`	RM	✓	✓	S	✓	R	R

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity, `H` is set

OR - Bitwise OR

Description: Logically ORs the source and destination, result is stored in destination.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`OR r, r`	R	✓	✓	R	✓	R	R
Immediate	`OR r, nn`	R	✓	✓	R	✓	R	R
Indirect	`OR r, (rr)`	RM	✓	✓	R	✓	R	R
Indexed	`OR r, (rr+dd)`	RM	✓	✓	R	✓	R	R

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Direct	`OR r, (nnnn)`	RM	✓	✓	R	✓	R	R

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity

XOR - Bitwise Exclusive OR

Description: Logically XORs the source and destination, result is stored in destination.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`XOR r, r`	R	✓	✓	R	✓	R	R
Immediate	`XOR r, nn`	R	✓	✓	R	✓	R	R
Indirect	`XOR r, (rr)`	RM	✓	✓	R	✓	R	R
Indexed	`XOR r, (rr+dd)`	RM	✓	✓	R	✓	R	R
Direct	`XOR r, (nnnn)`	RM	✓	✓	R	✓	R	R

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity

TST - Test

Description: Logically ANDs the source and destination, result is discarded and flags are set.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`TST r, r`	R	✓	✓	S	✓	R	R
Immediate	`TST r, nn`	R	✓	✓	S	✓	R	R
Indirect	`TST r, (rr)`	RM	✓	✓	S	✓	R	R
Indexed	`TST r, (rr+dd)`	RM	✓	✓	S	✓	R	R
Direct	`TST r, (nnnn)`	RM	✓	✓	S	✓	R	R

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity

CPL - Complement

Description: One's complement (invert) the operand. Equivalent to `XOR operand, -1`.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`CPL r`	UR	✓	✓	S	✓	S	-

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Indirect	`CPL (rr)`	UM	✓	✓	S	✓	S	-
Indexed	`CPL (rr+dd)`	UM	✓	✓	S	✓	S	-
Direct	`CPL (nnnn)`	UM	✓	✓	S	✓	S	-

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity

BIT - Test Bit

Description: Tests the specified bit of the operand. Source selects bit number, modulo operand size.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`BIT r, r`	R	-	✓	S	-	R	-
Immediate	`BIT n, r`	R	-	✓	S	-	R	-
Indirect	`BIT r, (rr)`	RM	-	✓	S	-	R	-
Indexed	`BIT r, (rr+dd)`	RM	-	✓	S	-	R	-
Direct	`BIT r, (nnnn)`	RM	-	✓	S	-	R	-

Operand Sizes: Byte, Word, Dword
Flags: `Z` set if bit is 0

SET - Set Bit

Description: Sets the specified bit of the destination. Source selects bit number, modulo operand size.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`SET r, r`	R	-	-	-	-	-	-
Immediate	`SET n, r`	R	-	-	-	-	-	-
Indirect	`SET r, (rr)`	RM	-	-	-	-	-	-
Indexed	`SET r, (rr+dd)`	RM	-	-	-	-	-	-
Direct	`SET r, (nnnn)`	RM	-	-	-	-	-	-

Operand Sizes: Byte, Word, Dword

RES - Reset Bit

Description: Resets the specified bit of the destination. Source selects bit number, modulo operand size.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`RES r, r`	R	-	-	-	-	-	-
Immediate	`RES n, r`	R	-	-	-	-	-	-
Indirect	`RES r, (rr)`	RM	-	-	-	-	-	-
Indexed	`RES r, (rr+dd)`	RM	-	-	-	-	-	-
Direct	`RES r, (nnnn)`	RM	-	-	-	-	-	-

Operand Sizes: Byte, Word, Dword

RLC - Rotate Left Circular

Description: Rotates the destination left by the number of positions specified by source. MSB copied to carry and bit 0.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`RLC r, r`	R	✓	✓	R	✓	R	✓
Immediate	`RLC n, r`	R	✓	✓	R	✓	R	✓
Indirect	`RLC r, (rr)`	RM	✓	✓	R	✓	R	✓
Indexed	`RLC r, (rr+dd)`	RM	✓	✓	R	✓	R	✓
Direct	`RLC r, (nnnn)`	RM	✓	✓	R	✓	R	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity, `C` receives MSB

RRC - Rotate Right Circular

Description: Rotates the destination right by the number of positions specified by source. Bit 0 copied to carry and MSB.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`RRC r, r`	R	✓	✓	R	✓	R	✓
Immediate	`RRC n, r`	R	✓	✓	R	✓	R	✓
Indirect	`RRC r, (rr)`	RM	✓	✓	R	✓	R	✓
Indexed	`RRC r, (rr+dd)`	RM	✓	✓	R	✓	R	✓
Direct	`RRC r, (nnnn)`	RM	✓	✓	R	✓	R	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity, `C` receives bit 0

RL - Rotate Left

Description: Rotates the destination left by the number of positions specified by source. Carry copied to bit 0, MSB copied to carry.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`RL r, r`	R	✓	✓	R	✓	R	✓
Immediate	`RL n, r`	R	✓	✓	R	✓	R	✓
Indirect	`RL r, (rr)`	RM	✓	✓	R	✓	R	✓
Indexed	`RL r, (rr+dd)`	RM	✓	✓	R	✓	R	✓
Direct	`RL r, (nnnn)`	RM	✓	✓	R	✓	R	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity, `C` receives MSB

RR - Rotate Right

Description: Rotates the destination right by the number of positions specified by source. Carry copied to MSB, bit 0 copied to carry.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`RR r, r`	R	✓	✓	R	✓	R	✓
Immediate	`RR n, r`	R	✓	✓	R	✓	R	✓
Indirect	`RR r, (rr)`	RM	✓	✓	R	✓	R	✓
Indexed	`RR r, (rr+dd)`	RM	✓	✓	R	✓	R	✓
Direct	`RR r, (nnnn)`	RM	✓	✓	R	✓	R	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity, `C` receives bit 0

SLA - Shift Left Arithmetic

Description: Shifts the destination left by the number of positions specified by source. MSB copied to carry, 0 shifted into bit 0.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`SLA r, r`	R	✓	✓	R	✓	R	✓
Immediate	`SLA n, r`	R	✓	✓	R	✓	R	✓
Indirect	`SLA r, (rr)`	RM	✓	✓	R	✓	R	✓
Indexed	`SLA r, (rr+dd)`	RM	✓	✓	R	✓	R	✓
Direct	`SLA r, (nnnn)`	RM	✓	✓	R	✓	R	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity, `C` receives MSB

SRA - Shift Right Arithmetic

Description: Arithmetic shift right by the number of positions specified by source. Bit 0 copied to carry, MSB replicated.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`SRA r, r`	R	✓	✓	R	✓	R	✓
Immediate	`SRA n, r`	R	✓	✓	R	✓	R	✓
Indirect	`SRA r, (rr)`	RM	✓	✓	R	✓	R	✓
Indexed	`SRA r, (rr+dd)`	RM	✓	✓	R	✓	R	✓
Direct	`SRA r, (nnnn)`	RM	✓	✓	R	✓	R	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity, `C` receives bit 0

SRL - Shift Right Logical

Description: Logical shift right by the number of positions specified by source. Bit 0 copied to carry, 0 shifted into MSB.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`SRL r, r`	R	✓	✓	R	✓	R	✓
Immediate	`SRL n, r`	R	✓	✓	R	✓	R	✓
Indirect	`SRL r, (rr)`	RM	✓	✓	R	✓	R	✓
Indexed	`SRL r, (rr+dd)`	RM	✓	✓	R	✓	R	✓
Direct	`SRL r, (nnnn)`	RM	✓	✓	R	✓	R	✓

Operand Sizes: Byte, Word, Dword
Flags: `P/V` indicates parity, `C` receives bit 0

RLD - Rotate Left Decimal

Description: Rotates BCD digits left between A and (HL). High nibble of (HL) → low nibble of A.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`RLD`	N	✓	✓	R	✓	R	-

Operand Sizes: Byte
Note: Deprecated, included for Z80 compatibility

RRD - Rotate Right Decimal

Description: Rotates BCD digits right between A and (HL). Low nibble of (HL) → low nibble of A.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`RRD`	N	✓	✓	R	✓	R	-

Operand Sizes: Byte
Note: Deprecated, included for Z80 compatibility

6.5 Flow Control

Note: All branch instructions support conditional execution. Format: `INS cc, operand` (e.g., `JP NZ, address`, `CALL C, subroutine`)

NOP - No Operation

Description: Performs no operation, advances PC by 1. Useful for alignment and timing.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`NOP`	SB	-	-	-	-	-	-

JP - Jump

Description: Transfers execution to the specified address.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`JP rr`	B	-	-	-	-	-	-
Direct	`JP nnnn`	B	-	-	-	-	-	-

Operand Sizes: Dword

JR - Jump Relative

Description: Transfers execution to an address relative to the current PC.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`JR r`	B	-	-	-	-	-	-
Immediate	`JR nn`	B	-	-	-	-	-	-

Operand Sizes: Byte (±127), Word (±32767)

DJNZ - Decrement and Jump if Not Zero

Description: Decrements B and jumps to relative address if B ≠ 0.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Immediate	`DJNZ n`	SB	-	-	-	-	-	-

Operand Sizes: Byte (±127)

JANZ - Jump if A Not Zero

Description: Performs `TST A, A` and jumps to relative address if A ≠ 0.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Immediate	`JANZ n`	SB	-	-	-	-	-	-

Operand Sizes: Byte (±127)

CALL - Call Subroutine

Description: Pushes return address onto stack and transfers execution to the specified address.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`CALL rr`	B	-	-	-	-	-	-
Direct	`CALL nnnn`	B	-	-	-	-	-	-

Operand Sizes: Dword

CALLR - Call Subroutine Relative

Description: Pushes return address onto stack and transfers execution to a relative address.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Register	`CALLR r`	B	-	-	-	-	-	-
Immediate	`CALLR nn`	B	-	-	-	-	-	-

Operand Sizes: Byte (±127), Word (±32767)

RET - Return

Description: Pops return address from stack and transfers execution to that address.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`RET`	B	-	-	-	-	-	-

RETI - Return from Interrupt

Description: Returns from interrupt service routine and re-enables interrupts.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`RETI`	B	-	-	-	-	-	-

RETN - Return from Non-Maskable Interrupt

Description: Returns from NMI service routine and restores interrupt state.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`RETN`	B	-	-	-	-	-	-

RST - Reset to Trap

Description: Generates a software interrupt with specified vector number.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Immediate	`RST n`	N	-	-	-	-	-	-

SCF - Set Carry Flag

Description: Sets the carry flag to 1.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`SCF`	N	-	-	R	-	R	S

CCF - Complement Carry Flag

Description: Inverts the carry flag.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`CCF`	N	-	-	✓	-	R	✓

Flags: `H` receives previous carry value, `C` is complemented

---

6.6 System

EI - Enable Interrupts

Description: Enables maskable interrupts. Effect delayed by one instruction.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`EI`	N	-	-	-	-	-	-

System Flags: Sets IE flag in upper byte of F register

DI - Disable Interrupts

Description: Disables maskable interrupts.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`DI`	N	-	-	-	-	-	-

System Flags: Resets IE flag in upper byte of F register

IM 1 - Interrupt Mode 1

Description: Sets interrupt mode 1 (all interrupts use vector 1).

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`IM 1`	N	-	-	-	-	-	-

IM 2 - Interrupt Mode 2

Description: Sets interrupt mode 2 (vectored interrupts with device-supplied vector).

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`IM 2`	N	-	-	-	-	-	-

HALT - Halt Processor

Description: Halts processor execution until interrupt or reset occurs.

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`HALT`	N	-	-	-	-	-	-

LD I - Load Interrupt Vector Base Register

Description: Loads an immediate value into the `I` register

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`LD I, nn`	N	-	-	-	-	-	-

LD R - Load Refresh Register

Description: Loads the `R` register to/from `A`

Mode	Mnemonic	Format	S	Z	H	P/V	N	C
Implied	`LD R, A`	N	-	-	-	-	-	-
Implied	`LD A, R`	N	-	-	-	-	-	-

---

7. Timing and Bus Cycles

The IM8000 architecture retains the Z80 concept of T-states. A T-state represents one rising-edge-to-rising-edge interval of the system clock. All timing is defined relative to T-states.

An M-cycle represents one bus read/write cycle. An M-cycle is typically 3 T-states, but may be extended by wait states or I/O timing.

Exact timing may vary between implementations. Complex instructions such as `MLT` and `DIV` may have variable timing based on operand values.

7.1 Instruction Throughput Notes

Instructions have a base timing of 1-2 M-cycles for the instruction word. Any additional immediate values or displacements will cost 1 additional M-cycle per bus cycle. This depends on the model of the processor and alignment of the instruction.

- **IM8008**:
  - **Instruction Word**: 2 M-cycles
  - **Immediate Values**: 1 M-cycle per byte, 2 M-cycles per word
- **IM8000:**
  - **Aligned:**

    - **Instruction Word**: 1 M-cycle
    - **Immediate Values**: 1 M-cycle per byte, 1 M-cycle per word
- **Unaligned:**
    - **Instruction Word**: 2 M-cycles
    - **Immediate Values**: 1 M-cycle per byte, 2 M-cycles per word

---

Appendix A - Instruction Encoding

A.1 Field Encoding

A.1.1 Register Encoding (3 bits)

Code	Register	Description
000	A	General register A
001	B	General register B
010	C	General register C
011	D	General register D
100	E	General register E
101	H	General register H
110	L	General register L
111	-	Immediate

Note:

- Displacements/Absolute Addresses are stored immediately after the instruction word
- Immediate values are stored immediately after the Displacement/Absolute Address (if exists)
- Encoding with immediate: `[Instruction Word][Immediate]`
- Encoding with displacement and immediate: `[Instruction Word][Displacement][Immediate]`

A.1.2 Address Register Encoding (3 bits)

For formats using index register fields (32-bit operands):

Code	Register	Description
000	AF	Register pair AF
001	BC	Register pair BC
010	DE	Register pair DE
011	HL	Register pair HL

Code	Register	Description
100	IX	Index register IX
101	IY	Index register IY
110	SP	Stack pointer
111	Direct/Immediate	Direct addressing or Immediate value

A.1.3 Size Field Encoding

Code	Size	Description
00	Byte	8-bit operation
01	Word	16-bit operation
10	Dword	32-bit operation
11	-	-

A.1.4 Condition Field Encoding

Code	Mnemonic	Condition	Flag Test
0000	NZ	Not Zero	Z = 0
0001	Z	Zero	Z = 1
0010	NC	No Carry	C = 0
0011	C	Carry	C = 1
0100	PO	Parity Odd/No Overflow	PV = 0
0101	PE	Parity Even/Overflow	PV = 1
0110	P	Plus	S = 0
0111	M	Minus	S = 1
1000-1110	-	Reserved	-
1111	-	Always (unconditional)	-

A.1.5 Direction Field Encoding

Code	Description
0	Register to Memory (`LD (rr), r`)
1	Memory to Register (`LD r, (rr)`)

A.2 Groups and Instruction Assignments

A.2.1 Encoding Legend

Character	Field
`S`	Source Register
`D`	Destination Register
`A`	Address Register
`R`	Register
`s`	Size
`d`	Direction
`c`	Condition

A.2.2 Format R - Register-Register - (Group 00)

Field Positions:

- *b0-b1* - Group (00)
- *b2-b7* - Opcode (6 bits)
- *b8-b9* - Size (2 bits)
- *b10-b12* - Destination Register (3 bits)
- *b13-b15* - Source Register (3 bits)

Opcode	Instruction	Full Encoding
000000	LD	`SSSDDDss 00000000`
000001	EX	`SSSDDDss 00000100`
000010	IN	`SSSDDDss 00001000`
000011	OUT	`SSSDDDss 00001100`
000100	ADD	`SSSDDDss 00010000`
000101	ADC	`SSSDDDss 00010100`
000110	SUB	`SSSDDDss 00011000`
000111	SBC	`SSSDDDss 00011100`
001000	CP	`SSSDDDss 00100000`
001001	AND	`SSSDDDss 00100100`
001010	OR	`SSSDDDss 00101000`
001011	XOR	`SSSDDDss 00101100`
001100	TST	`SSSDDDss 00110000`
001101	BIT	`SSSDDDss 00110100`
001110	SET	`SSSDDDss 00111000`
001111	RES	`SSSDDDss 00111100`

Opcode	Instruction	Full Encoding
010000	RLC	`SSSDDDss 01000000`
010001	RRC	`SSSDDDss 01000100`
010010	RL	`SSSDDDss 01001000`
010011	RR	`SSSDDDss 01001100`
010100	SLA	`SSSDDDss 01010000`
010101	SRA	`SSSDDDss 01010100`
010110	SRL	`SSSDDDss 01011000`

A.2.3 Format RM - Register-Memory - (Group 01)

Field Positions:

- *b0-b1* - Group (01)
- *b2-b6* - Opcode (5 bits)
- *b7* - Direction (1 bit)
- *b8-b9* - Size (2 bits)
- *b10-b12* - Register (3 bits)
- *b13-b15* - Address register (3 bits)

Opcode	Instruction	Full Encoding
00000	LD	`AAARRRss d0000001`
00001	EX	`AAARRRss d0000101`
00010	-	`AAARRRss d0001001`
00011	-	`AAARRRss d0001101`
00100	ADD	`AAARRRss d0010001`
00101	ADC	`AAARRRss d0010101`
00110	SUB	`AAARRRss d0011001`
00111	SBC	`AAARRRss d0011101`
01000	CP	`AAARRRss d0100001`
01001	AND	`AAARRRss d0100101`
01010	OR	`AAARRRss d0101001`
01011	XOR	`AAARRRss d0101101`
01100	TEST	`AAARRRss d0110001`
01101	BIT	`AAARRRss d0110101`
01110	SET	`AAARRRss d0111001`
01111	RES	`AAARRRss d0111101`
10000	RLC	`AAARRRss d1000001`

Opcode	Instruction	Full Encoding
10001	RRC	`AAARRRss d1000101`
10010	RL	`AAARRRss d1001001`
10011	RR	`AAARRRss d1001101`
10100	SLA	`AAARRRss d1010001`
10101	SRA	`AAARRRss d1010101`
10110	SRL	`AAARRRss d1011001`

A.2.4 Format UR - Unary Register - (Group 10-00)

Field Positions:

- *b0-b1* - Group (10)
- *b2-b3* - Sub-Group (00)
- *b4-b7* - Opcode (4 bits)
- *b8-b9* - Size (2 bits)
- *b10-b12* - Register (3 bits)
- *b13-b15* - Function (3 bits)

Opcode	Function	Instruction	Full Encoding
0000	000	EX r, r'	`000RRRss 00000010`
0000	001	EXH	`001RRRss 00000010`
0000	010	PUSH	`010RRRss 00000010`
0000	011	POP	`011RRRss 00000010`
0000	100	INC	`100RRRss 00000010`
0000	101	DEC	`101RRRss 00000010`
0000	110	NEG	`110RRRss 00000010`
0000	111	EXT	`111RRRss 00000010`
0001	000	MLT	`000RRRss 00010010`
0001	001	DIV	`001RRRss 00010010`
0001	010	SDIV	`010RRRss 00010010`
0001	011	CPL	`011RRRss 00010010`

A.2.5 Format UM - Unary Memory - (Group 10-01)

Field Positions:

- *b0-b1* - Group (10)
- *b2-b3* - Sub-Group (01)
- *b4-b7* - Opcode (4 bits)

- *b8-b9* - Size (2 bits)
- *b10-b12* - Function (3 bits)
- *b13-b15* - Address register (3 bits)

Opcode	Function	Instruction	Full Encoding
0000	000	-	`AAA000ss 00000110`
0000	001	EXH	`AAA001ss 00000110`
0000	010	-	`AAA010ss 00000110`
0000	011	-	`AAA011ss 00000110`
0000	100	INC	`AAA100ss 00000110`
0000	101	DEC	`AAA101ss 00000110`
0000	110	NEG	`AAA110ss 00000110`
0000	111	EXT	`AAA111ss 00000110`
0001	000	MLT	`AAA000ss 00010110`
0001	001	DIV	`AAA001ss 00010110`
0001	010	SDIV	`AAA010ss 00010110`
0001	011	CPL	`AAA011ss 00010110`

A.2.6 Format B - Branch - (Group 10-10)

Field Positions:

- *b0-b1* - Group (10)
- *b2-b3* - Sub-Group (10)
- *b4-b8* - Opcode (5 bits)
- *b9-b12* - Condition (4 bits)
- *b13-b15* - Address register (3 bits)

Opcode	Instruction	Full Encoding
00000	JP	`AAAcccc0 00001010`
00001	JR s8	`AAAcccc0 00011010`
00010	JR s16	`AAAcccc0 00101010`
00011	-	`AAAcccc0 00111010`
00100	CALL	`AAAcccc0 01001010`
00101	CALLR s8	`AAAcccc0 01011010`
00110	CALLR s16	`AAAcccc0 01101010`
00111	-	`AAAcccc0 01111010`
01000	RET	`000cccc0 10001010`
01001	RETI	`000cccc0 10011010`

Opcode	Instruction	Full Encoding
01010	RETN	`000cccc0 10101010`

A.2.7 Format N - Nullary - (Group 10-11)

Field Positions:

- *b0-b1* - Group (10)
- *b1-b2* - Sub-Group (11)
- *b3-b7* - Opcode (4 bits)
- *b8-b15* - Function (8 bits)

Byte Block Operations (Opcode 0000)

Opcode	Function	Instruction	Full Encoding
0000	00000000	LDI	`00000000 00001110`
0000	00000001	LDIR	`00000001 00001110`
0000	00000010	LDD	`00000010 00001110`
0000	00000011	LDDR	`00000011 00001110`
0000	00000100	CPI	`00000100 00001110`
0000	00000101	CPIR	`00000101 00001110`
0000	00000110	CPD	`00000110 00001110`
0000	00000111	CPDR	`00000111 00001110`
0000	00001000	TSI	`00001000 00001110`
0000	00001001	TSIR	`00001001 00001110`
0000	00001010	TSD	`00001010 00001110`
0000	00001011	TSDR	`00001011 00001110`
0000	00001100	INI	`00001100 00001110`
0000	00001101	INIR	`00001101 00001110`
0000	00001110	IND	`00001110 00001110`
0000	00001111	INDR	`00001111 00001110`
0000	00010000	OUTI	`00010000 00001110`
0000	00010001	OTIR	`00010001 00001110`
0000	00010010	OUTD	`00010010 00001110`
0000	00010011	OTDR	`00010011 00001110`

Word Block Operations (Opcode 0001)

Opcode	Function	Instruction	Full Encoding
0001	00000000	LDI	`00000000 00011110`
0001	00000001	LDIR	`00000001 00011110`
0001	00000010	LDD	`00000010 00011110`
0001	00000011	LDDR	`00000011 00011110`
0001	00000100	CPI	`00000100 00011110`
0001	00000101	CPIR	`00000101 00011110`
0001	00000110	CPD	`00000110 00011110`
0001	00000111	CPDR	`00000111 00011110`
0001	00001000	TSI	`00001000 00011110`
0001	00001001	TSIR	`00001001 00011110`
0001	00001010	TSD	`00001010 00011110`
0001	00001011	TSDR	`00001011 00011110`
0001	00001100	INI	`00001100 00011110`
0001	00001101	INIR	`00001101 00011110`
0001	00001110	IND	`00001110 00011110`
0001	00001111	INDR	`00001111 00011110`
0001	00010000	OUTI	`00010000 00011110`
0001	00010001	OTIR	`00010001 00011110`
0001	00010010	OUTD	`00010010 00011110`
0001	00010011	OTDR	`00010011 00011110`

Exchange (Opcode 0100)

Opcode	Function	Instruction	Full Encoding
0100	00000000	EXX	`00000000 01001110`
0100	00000001	EXI	`00000001 01001110`

Control Flow (Opcode 0101)

Opcode	Function	Instruction	Full Encoding
0101	00000000	RST	`00000000 01011110`
0101	00000001	SCF	`00000001 01011110`
0101	00000010	CCF	`00000010 01011110`

BCD (Opcode 0110)

Opcode	Function	Instruction	Full Encoding
0110	00000000	DAA	`00000000 01101110`
0110	00000001	RLD	`00000001 01101110`
0110	00000010	RRD	`00000010 01101110`

System (Opcode 1000)

Opcode	Function	Instruction	Full Encoding
1000	00000000	HALT	`00000000 10001110`
1000	00000001	EI	`00000001 10001110`
1000	00000010	DI	`00000010 10001110`
1000	00000101	IM 1	`00000101 10001110`
1000	00000110	IM 2	`00000110 10001110`
1000	00000111	LD I	`00000111 10001110`
1000	00001000	LD R, A	`00001000 10001110`
1000	00001001	LD A, R	`00001001 10001110`

A.2.8 Single Byte (Group 11-11)

Field Positions:

- *b0-b1* - Group (11)
- *b2-b3* - Sub-Group (11)
- *b4-b7* - Opcode

Opcode	Instruction	Full Encoding
0000	NOP	`00001111`
0001	DJNZ	`00011111`
0010	JANZ	`00101111`

Document Information

- **Revision**: 0.5
- **Date**: December 2025
- **Pages**: 50
- **Classification**: Preliminary