Y80 Microprocessor

Technical Manual

Systemyde

Table of Contents

Revison History

Date	Changes	Page(s)

Introduction

This book documents the operation of the Y80 microprocessor. The Y80 design is supplied in Verilog HDL and can be implemented in any technology supported by a logic synthesis tool that accepts Verilog HDL. Included in the design package is a test bench that exercises all instructions, flag settings, and representative data patterns. The test patterns should achieve at least 95% fault coverage.

The Y80 CPU was designed in a clean-room environment and is compatible with the Zilog Z80 microprocessor. Only publicly available documentation was used to create this design so there may be minor differences where the public documentation is misleading or lacking. The instruction execution times are not identical between the two designs. The Y80 CPU operates with a consistent two-clock-cycle machine cycle, while the Z80 microprocessor uses machine cycles that vary from three to seven clock cycles.

This document should always be used as the final word on the operation of the Y80 CPU, but it is useful to refer to the Zilog documentation if the description given here is too cryptic. The Z80 architecture is over thirty years old, so it is assumed that it is already at least somewhat familiar to the reader.

The Y80 CPU is accompanied by full design documentation, in the form of a large spreadsheet, which describes nearly every facet of the internal operation of the processor. This provides knowledgeable users the opportunity to customize the design for unique application requirements.

Features

* Fully functional synthesizable Verilog HDL version of the Z80 CPU

* Vendor and technology independent

* Software compatible with several industry-standard processors

* 189 instructions

* Eight addressing modes

* 64K byte memory addressing capability

* Separate 64K byte I/O address space

* 16 bit ALU with bit, byte and BCD operations

* Powerful vectored interrupt capability with separate interrupt vector input bus

* Static, fully synchronous design uses no 3-state buses

* Uniform 2 clock-cycle machine cycle

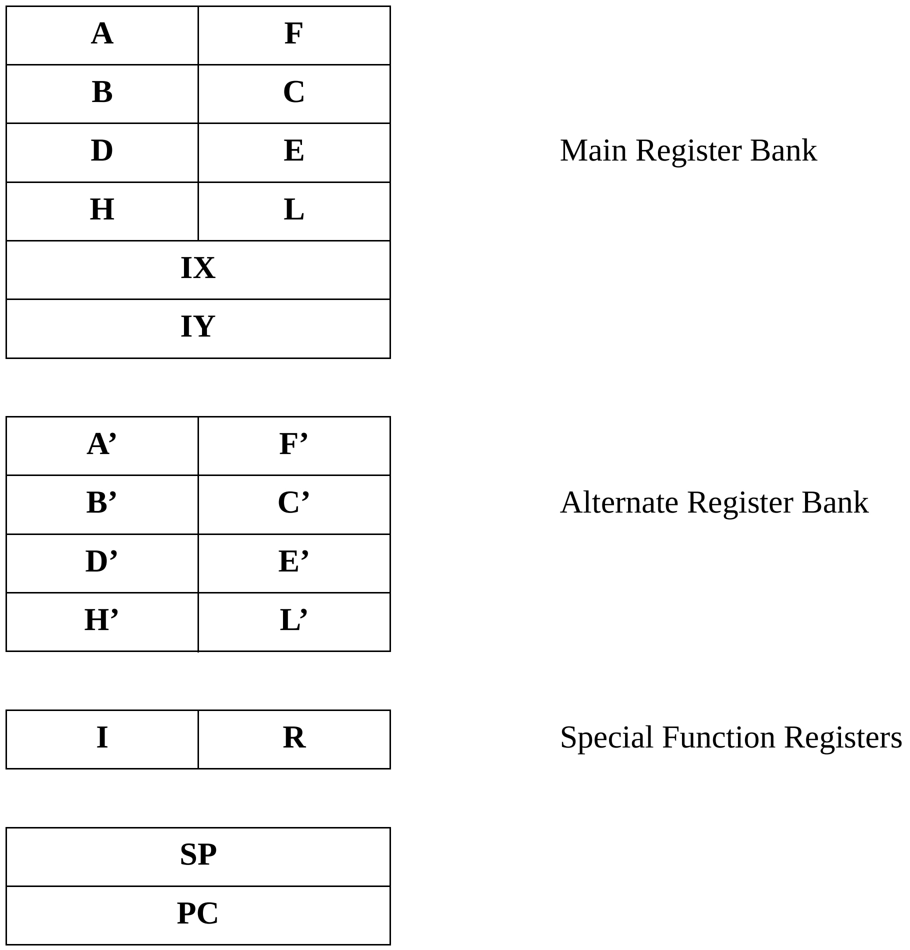
* Memory interface matches common FPGA and ASIC memory timing

* Separate I/O bus, compatible with AMBA Peripheral Bus

* Full design documentation included

* Verilog simulation and test suite included

Shown below are the registers visible to the programmer. The main registers have both a primary and an alternate version. The primary register set consists of A, F, B, C, D, E, H, and L, while the alternate register set consists of A', F', B', C', D', E', H', and L'. At any given time only one bank is active, and care must be used when switching between banks, as there is no way for the programmer to check which bank is active. The accumulator, A, is the destination for all 8-bit arithmetic and logic operations, while the Flag register F contains the flag results of arithmetic and logic operations. The other general-purpose registers can be paired, BC or DE or HL, to form 16-bit registers. There are two index registers, IX and IY, used for indexed addressing mode. The I register holds the upper eight bits of the interrupt vector table address for use in Interrupt Mode 2. The R register is left over from the original Z80 architecture, where it was used to hold a refresh address for DRAMs. In the Y80 it is just another general purpose register. The Stack pointer, SP, holds the address of the stack, and the Program Counter, PC, holds the address of the currently executing instruction.



Pin Descriptions

The Y80 design does not attempt to match the signals or timing present on the Z80 microprocessor. Rather, the interfaces and signals are optimized for use in either an ASIC or an FPGA.

Memory and I/O use separate address and data buses in addition to the separate control signals. The memory bus is designed to match typical ASIC and FPGA memory timing, although it can be used with stand-alone memory devices just as easily. A separate interrupt vector bus is provided for use with an interrupt controller. If desired, this interrupt vector bus can be tied to either the memory or I/O input bus for operation more closely resembling that of the original Z80.

The interface signals for the Y80 CPU are detailed below. Note that all inputs except the two resets are sampled by the rising edge of the clock and all outputs change in response to the rising edge of the clock.

clearb (input, active-Low) The Master (test) Reset signal is used to initialize all of the flip-flops that are not initialized by the user reset signal. Most user-visible registers are not affected by the user reset, so this signal allows full initialization for testing and simulation. This is an asynchronous signal that should be used for Power-On Reset.

clkc (input, active-High) The CPU Clock connects to all flip-flops in the design.

dma_ack (output, active-High) The DMA Acknowledge signal is activated to indicate that the processor has halted to allow another bus master to use the bus. The **iack_tran**, **io_addr_out**, **io_data_out**, **io_tran**, **mem_addr_out**, **mem_data_out**, **mem_tran**, **reti_tran** and **t1** signals are all inactive (Low) during this time. The processor will signal **dma_ack** while in the Halt state without de-asserting the **halt_tran** signal. Interrupts are not sampled while the **dma_ack** signal is active, so the exit from a coincident Halt state will be deferred until the **dma_ack** signal is no longer active.

dma_req (input, active-High) The DMA Request signal requests that the processor halt to allow another bus master to transfer data on the bus. The processor only releases the bus between instructions, rather than between individual bus transactions.

halt_tran (output, active-High) The Halt Transaction signal is activated by the Halt instruction. While in the Halt state the CPU freezes and waits for an interrupt. The **iack_tran**, **io_addr_out**, **io_data_out**, **io_tran**, **mem_addr_out**, **mem_data_out**, **mem_tran**, **reti_tran** and **t1** signals are all inactive (Low) during this time.

iack_tran (output, active-High) The Interrupt Acknowledge Transaction signal is activated to identify an interrupt acknowledge bus transaction. An interrupt acknowledge occurs in response to either a Non-Maskable Interrupt request or an enabled Maskable Interrupt request. During an interrupt acknowledge the interrupt vector data bus is sampled, although the sampled value is only used in Interrupt Mode 2 with a maskable interrupt request.

int_req (input, active-High) The Interrupt Request signal is the maskable interrupt request. Maskable interrupts can be enabled and disabled under program control. This interrupt request is not latched, so it should remain active until an interrupt acknowledge transaction occurs.

io_addr_out (output, 16-bit bus) The I/O Address Output bus carries the address of the I/O port during an I/O transaction. This bus holds the current value until the next I/O transaction or until the **dma_ack** signal is activated.

io_data_in (input, 8-bit bus) The I/O Data Input bus is sampled during the various I/O input instructions. A separate bus allows peripherals to be connected without loading the memory data bus.

io_data_out (output, 8-bit bus) The I/O Data Output bus carries the output data for I/O output instructions. This bus holds the current value until the next I/O transaction or until the **dma_ack** signal is activated.

io_read (output, active-High) The I/O Read signal indicates the direction of data transfer during I/O transactions. High signals read and Low signals write. This signal is valid only during I/O transactions.

io_strobe (output, active-High) The I/O Strobe signal is one clock cycle wide (in the absence of Wait states) and identifies the data transfer clock cycle for I/O transactions.

io_tran (output, active-High) The I/O Transaction signal is activated for all I/O transactions.

ivec_data_in (input, 8-bit bus) The Interrupt Vector Data Input bus is sampled during interrupt acknowledge transactions. If the interrupt acknowledge was for a maskable interrupt and the CPU is in Interrupt Mode 2, this vector is used as a pointer in the interrupt vector table to find the starting address of the interrupt service routine.

ivec_read (output, active-High) The Interrupt Vector Read signal is one clock cycle wide (in the absence of Wait states) and identifies the data transfer clock cycle for interrupt acknowledge transactions.

mem_addr_out (output, 16-bit bus) The Memory Address Output bus carries the address during memory read and write transactions. This bus holds the current value until the next I/O transaction or until the **dma_ack** signal is activated.

mem_data_in (input, 8-bit bus) The Memory Data Input bus is sampled during memory read transactions. A separate bus allows peripherals to be connected without loading the memory data bus.

mem_data_out (output, 8-bit bus) The Memory Data Output bus carries the output data for memory write transactions. This bus holds the current value until the next I/O transaction or until the **dma_ack** signal is activated.

mem_rd (output, active-High) The Memory Read signal is one clock cycle wide (in the absence of Wait states) and identifies the data transfer clock cycle for memory read transactions.

mem_tran (output, active-High) The Memory Transaction signal is activated for memory read and write transactions. The **mem_tran** signal is active during the Halt state but is inactive during DMA transfers.

mem_wr (output, active-High) The Memory Write signal is one clock cycle wide (in the absence of Wait states) and identifies the data transfer clock cycle for memory write transactions.

nmi_req (input, active-High) The Non-Maskable Interrupt Request signal unconditionally interrupts the CPU. This request is internally latched, so that it can be as short as one clock cycle wide.

resetb (input, active-Low) The User Reset signal is used to initialize all state flip-flops and some user registers (the I, R, PC and SP registers). This is an asynchronous signal.

reti_tran (output, active-High) The Return From Interrupt transaction signal is activated immediately after the second stack read transaction during the Return From

Interrupt (RETI) instruction. This signal may be used by an external interrupt controller to re-enable interrupts, for example.

t1 (output, active-High) The T1 signal is active during the first clock cycle of a bus transaction. This signal is inactive during the Halt state.

wait_req (input, active-High) The Wait Request signal temporarily halts the CPU, usually to wait for memory access time to be met. The wait request is not honored during the bus idle state, or while the **halt_tran** signal is active.

External Timing

The Y80 CPU uses a uniform two-clock-cycle machine cycle. This consistent timing simplifies the design of logic external to the CPU makes it easier to track the state of the CPU.

The memory interface timing and signals are designed to make it easy to interface to standard ASIC and FPGA memories. It uses separate read and write strobes.

The I/O interface is very close to the AMBA Peripheral Bus (APB) to allow connection to APB peripherals with a minimum of logic. It uses a single strobe with a separate direction control. The only difference relative to the APB is the setup time for the write data. In the APB the write data is setup one clock before the strobe; in this interface the write data changes coincident with the leading edge of the strobe. In most cases this will not be a problem.

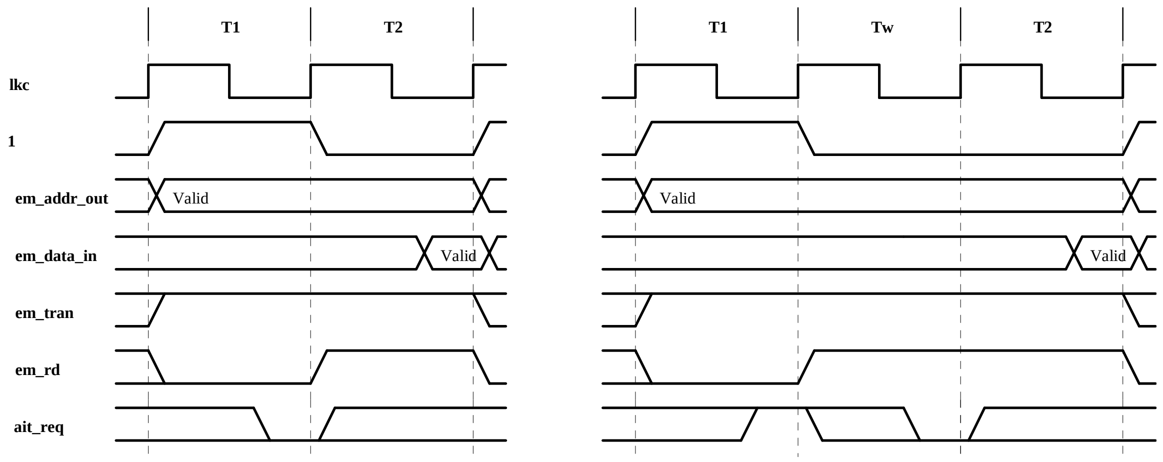
The separate interrupt vector bus provides an easy way to connect to the optional interrupt controller. The interrupt vector bus is used for Mode 2 maskable interrupts, so if this mode is not used the vector input bus can be tied to ground and the vector strobe output ignored.

In the diagrams below only the relevant signals are shown for each transaction. All other signals are either inactive or hold the previous value. Note that only one of the transaction identifiers (**mem_tran, io_tran, iack_tran, reti_tran,** and **halt_tran**) can be active at a time. If all are inactive, an idle bus transaction (usually for address calculation) is in progress. The **dma_ack** signal also indicates that the bus is idle, in response to the **dma_req** signal. The **dma_ack** signal can be active while **halt_tran** is active.

The **wait_req** input is only sampled for memory, I/O and interrupt acknowledge transactions and is ignored in all other cases. Wait states will disrupt the two-clock-cycle machine cycle rule. If this feature is important but wait states must be used, two wait states per transaction is recommended. If memory access time is an issue it might be better to stretch the first clock cycle of a transaction rather than add Wait states. The uniform two-clock machine cycle makes it relatively straightforward to do this.

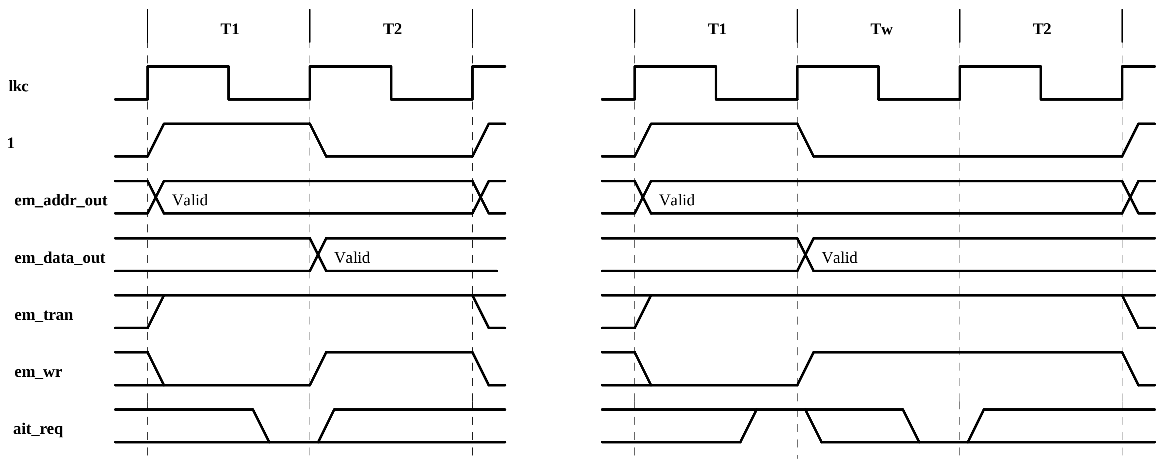
Memory Read

The figure below shows the memory read transaction, without Wait states and with one Wait state. Memory read transactions are used for both instruction and data fetch. There is no separate instruction/data status indicator, although this status exists internally if it is needed.



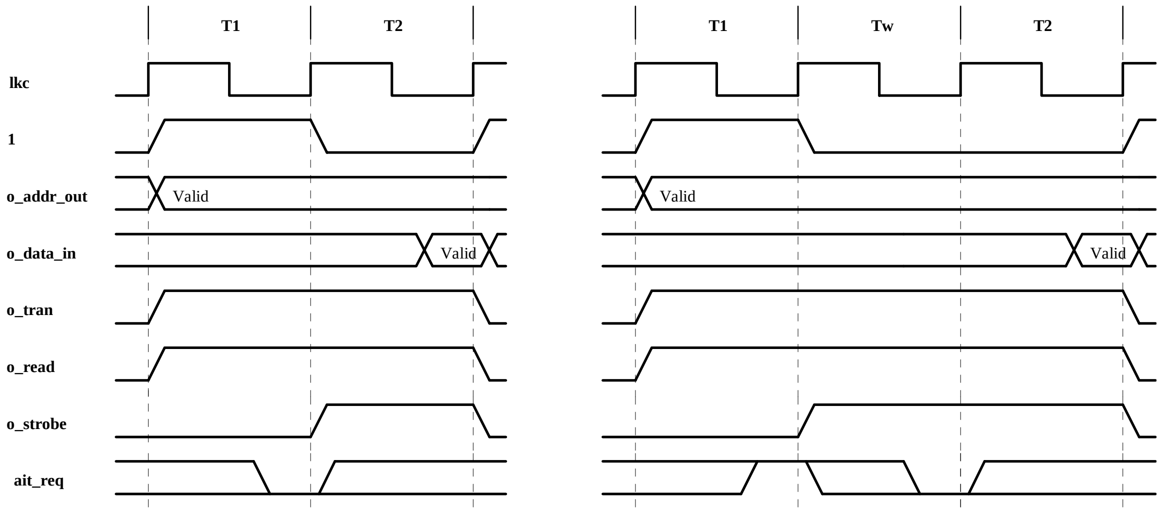
Memory Write

The figure below shows the memory write transaction, without Wait states and with one Wait state.



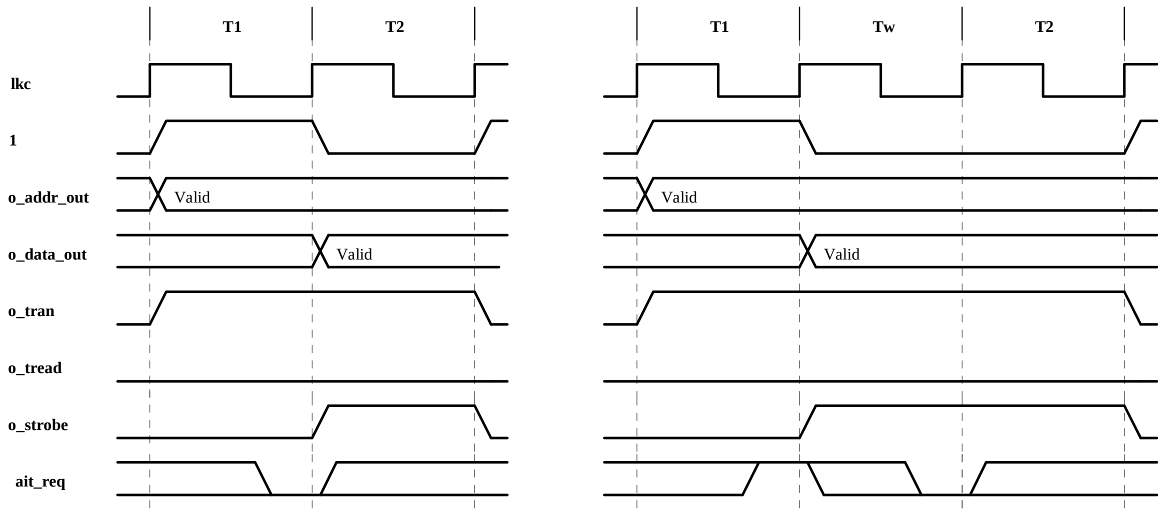
I/O Read

The figure below shows an I/O read transaction, without Wait states and with one Wait state.



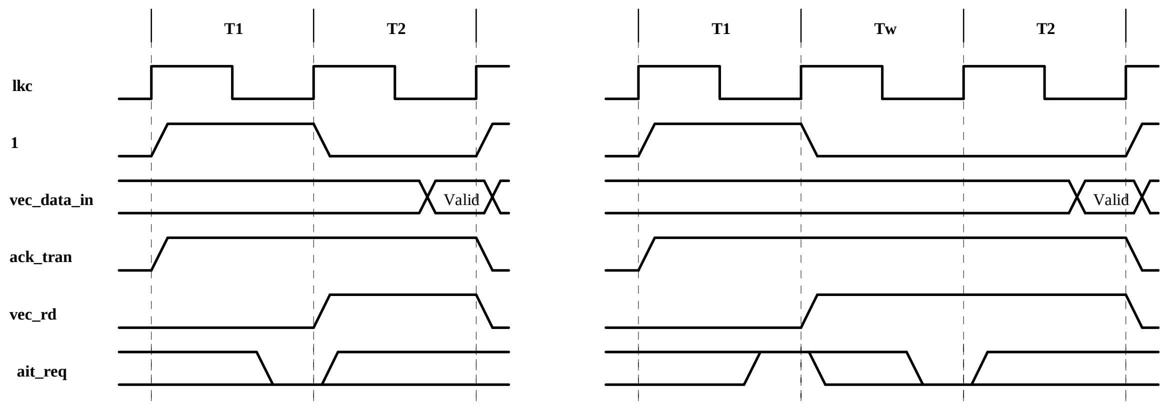
I/O Write

The figure below shows an I/O write transaction, without Wait states and with one Wait state.



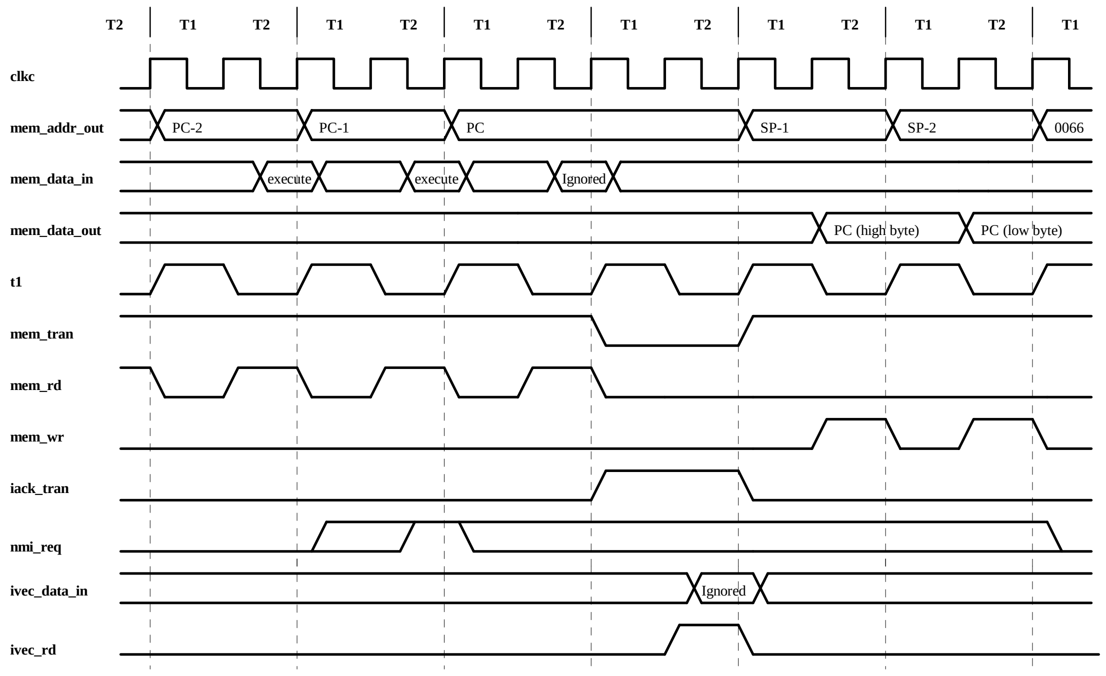
Interrupt Acknowledge

The figure below shows the interrupt acknowledge transaction, without Wait states and with one Wait state.



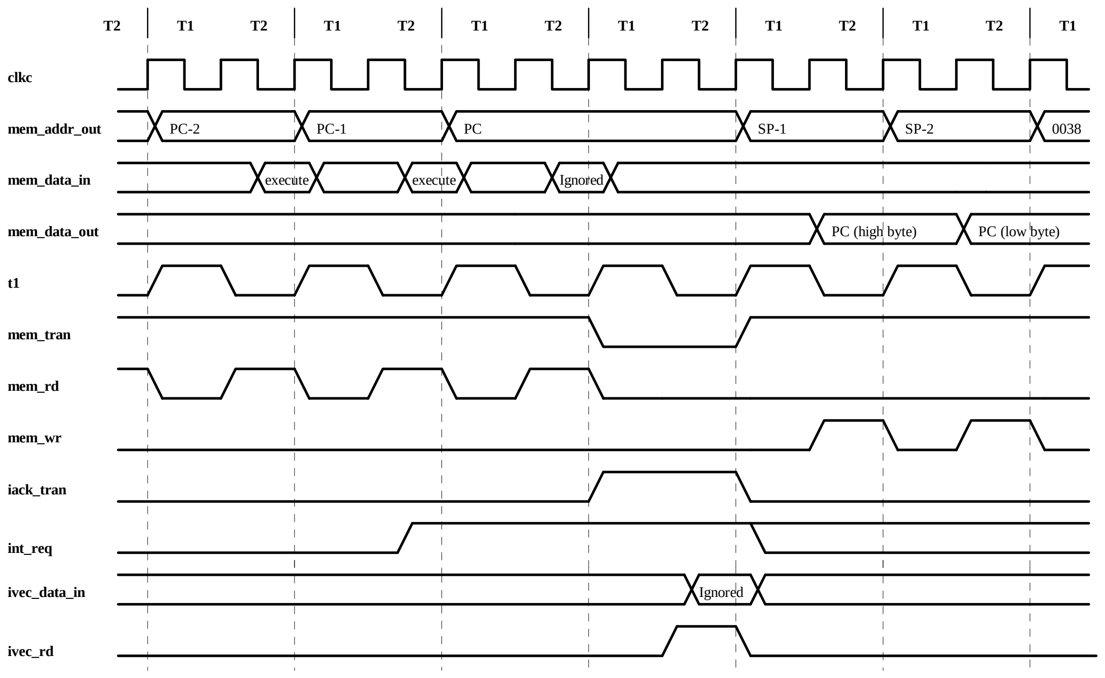
Non-maskable Interrupt

The timing of a non-maskable interrupt acknowledge transaction is shown below. The **nmi_req** input cannot be masked by software. This input must be sampled active by a rising edge of **clkc** to be recognized by the processor, but does not need to remain active until the interrupt acknowledge transaction. In fact, to prevent an endless loop of acknowledges, the **nmi_req** input must be de-asserted before the start of the fetch of the first instruction of the service routine. The acknowledge sequence consists of an aborted instruction fetch, the interrupt acknowledge, and two writes to push the contents of the program counter onto the stack. Execution then begins at 0x0066 with an instruction fetch. The non-maskable interrupt service routine must end with the RETN instruction to properly restore the state of the interrupt enable flag prior to the non-maskable interrupt.



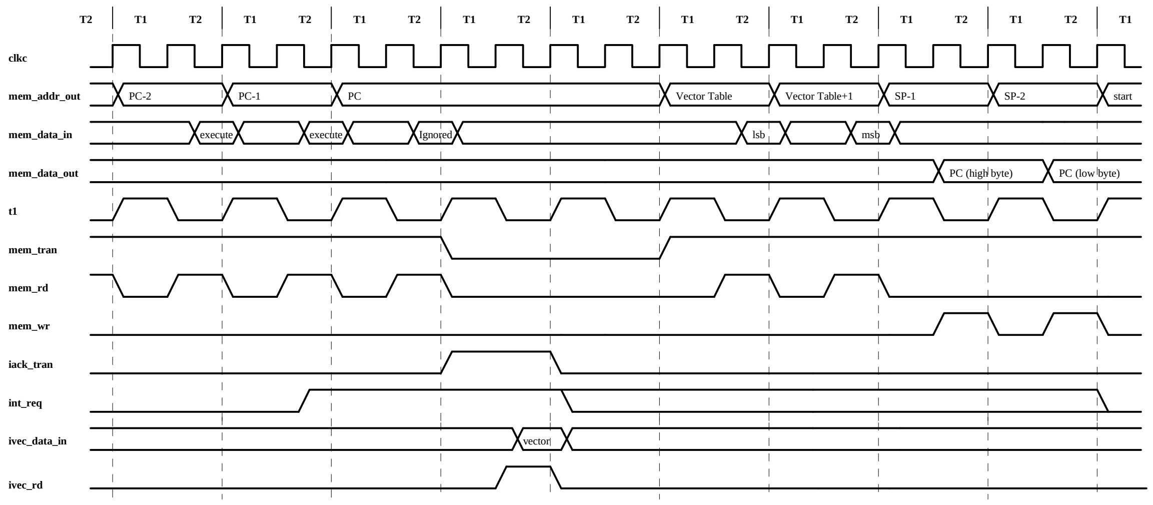
Interrupt Mode 0 or 1

The timing of a Mode 0 or Mode 1 interrupt acknowledge cycle is shown below. The **int_req** input needs to remain active until the interrupt acknowledge transaction. The acknowledge sequence consists of an aborted instruction fetch, the interrupt acknowledge, and two writes to push the contents of the program counter onto the stack. Execution then begins at address 0x0038 with an instruction fetch.

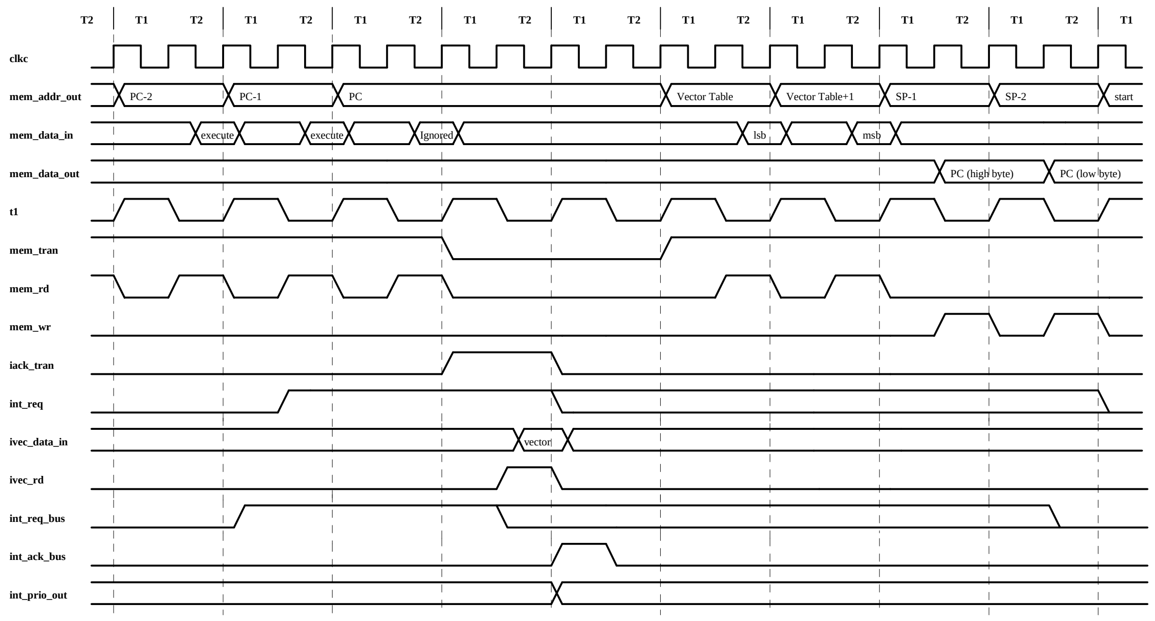


Interrupt Mode 2

The timing of a Mode 2 maskable interrupt acknowledge is shown below. The **int_req** input needs to remain active until the interrupt acknowledge transaction. The acknowledge sequence consists of an aborted instruction fetch, the interrupt acknowledge, an address calculation cycle, two reads of the interrupt vector table and two writes to push the contents of the program counter onto the stack. The processor automatically jumps to the address fetched from the interrupt vector table for the service routine. The upper eight bits of the interrupt vector table starting address are held in the I register in the processor. Note that the vector must be an even number. That is, the least significant bit of the vector must be a zero.



The interrupt controller in the Y80 MPU necessarily samples the **int_req_bus** inputs, which changes the timing slightly. The diagram below illustrates this change for Interrupt Mode 2. Also shown is the timing of the **int_ack_bus** and the **int_prio_out** signals.



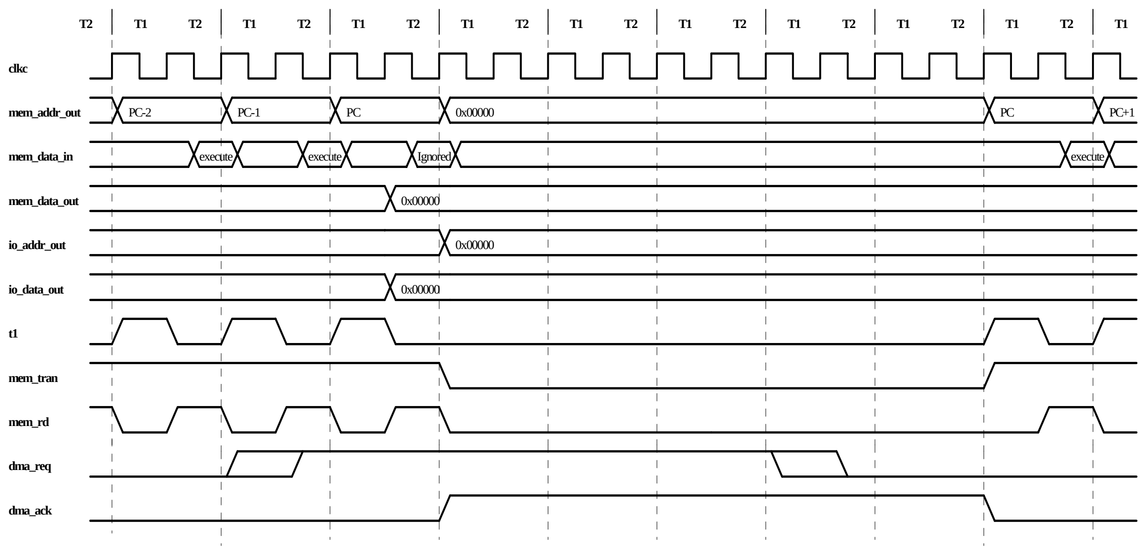
Although Interrupt Mode 2 is the preferred mode for use with the interrupt controller, the design allows the use of any interrupt mode. In Interrupt Mode 0 or 1 it will be necessary to use the **int_ack_bus** signals to externally latch the information about which interrupt is being acknowledged. This is because Interrupt Modes 0 and 1 will branch to a common interrupt service routine, rather than the individual routines possible in Interrupt Mode 2.

DMA Request/Acknowledge

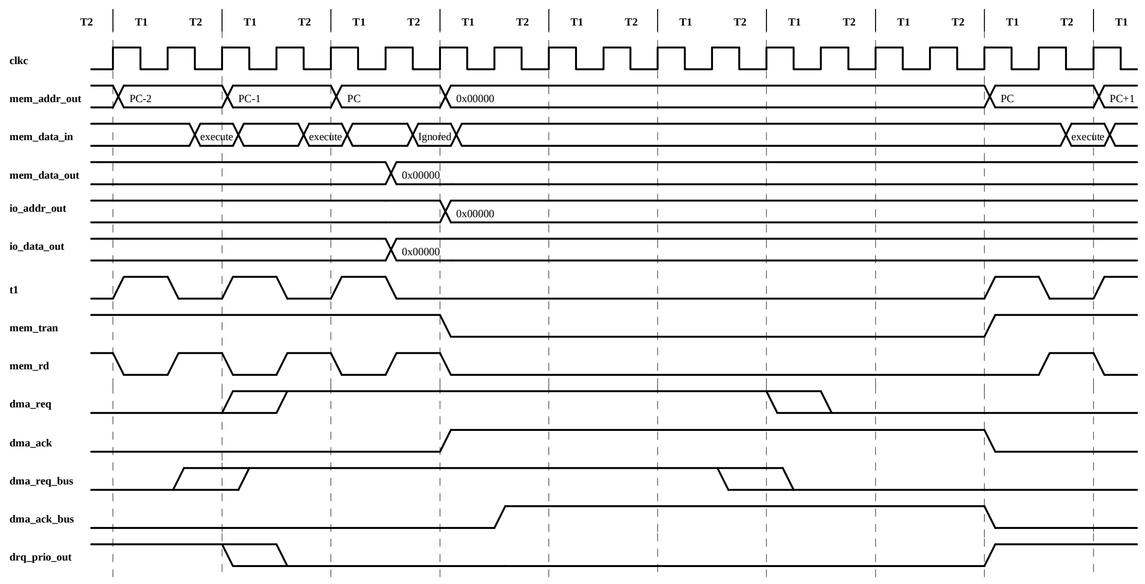
The timing of a DMA request and acknowledge is shown below. Note that like an interrupt, the **dma_req** signal is only sampled at the end of instructions. This guarantees that all instructions are atomic.

The delay from the **dma_req** signal to the **dma_ack** signal is always at least one bus cycle, irrespective of whether the processor is running, in the Halt state or in the Sleep state. This implies that it is more efficient to transfer multiple bytes each time that the **dma_req** signal is activated.

The **dma_req** signal can be asserted during the Halt or Sleep states. In this case the active **dma_req** signal will take precedence over **int_req** or **nmi_req** and inhibit either of these signals from causing an exit from the Halt or Sleep state. Once the **dma_req** signal is deasserted any pending or future interrupt request will cause the exit from the Halt or Sleep state.



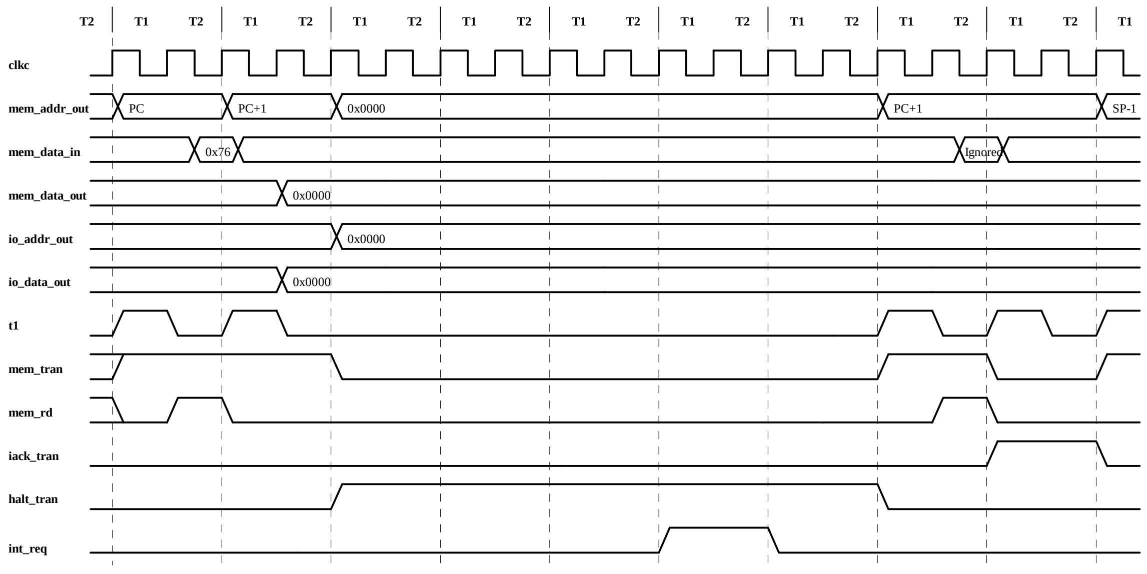
The DMA request controller in the Y80 MPU necessarily samples the **dma_req_bus** inputs, which changes the timing slightly. The diagram below illustrates this change. Also shown is the timing of the **dma_ack_bus** and the **dma_prio_out** signals. In particular, note that the leading edge of the **dma_ack_bus** signals are delayed by one clock cycle from the normal **dma_ack** timing. The timing of the trailing edge of these signals is not affected.



Halt state

The Halt state is entered when the HALT instruction is executed, as shown below. In the Halt state the processor freezes, for an unlimited number of two clock cycle machine cycles, with the **halt_tran** output active. The only way to exit the Halt state is with either an interrupt (either **nmi_req** or **int_req**) or via reset. Note that **int_req** can only be used to exit the Halt mode if interrupts are enabled when the HALT instruction is executed. The timing for exiting the Halt state with an interrupt is also shown below.

If the Halt state is exited by an interrupt, the processor will resume instruction execution (after the interrupt service routine) at the address of the instruction following the HALT instruction. The minimum width of the **halt_tran** signal is two clock cycles.



The Halt state in this design is slightly different from that in the Z80 microprocessor. In that design the processor continues to fetch the Halt instruction during the Halt state, leading to continued power dissipation. Since this operation requires the special step of "rewinding" the PC, no attempt was made to match this operation. Rather, the Halt state reduces the power consumption to a minimum by minimizing the number of signals that are transitioning during this state.

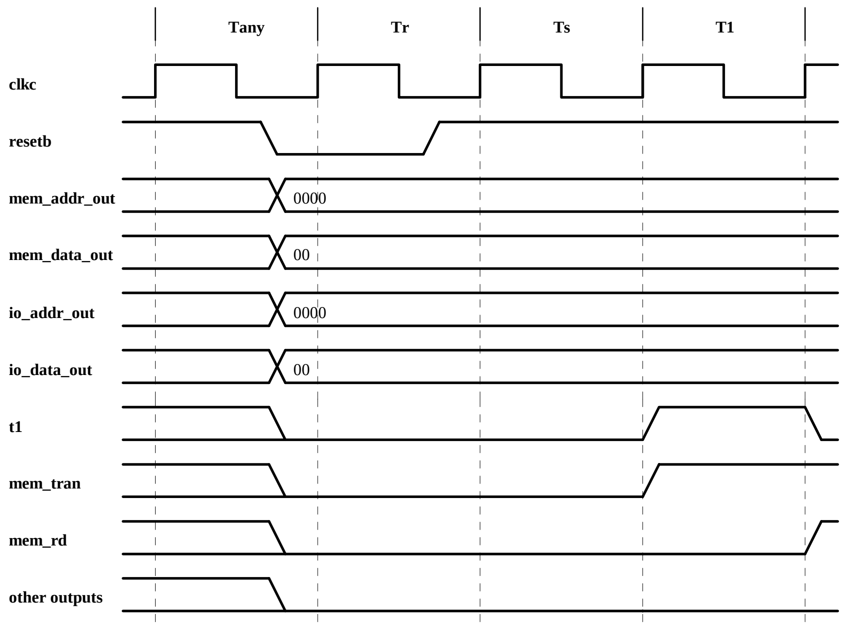
Reset

The Reset state is entered immediately when the **resetb** signal goes Low, independent of the current state, and this state continues until the first rising edge of **clkc** after the **resetb** signal is de-asserted. At this rising edge there is a one clock cycle transient state to set up the internal pipeline controls, and on the next clock the processor begins fetching the first instruction from address 0x0000.

Software starting at location 0x0000 must be able to distinguish between reset, execution of an RST 0 instruction, a trap, or watch-dog time-out. All of these cases cause the Program Counter to be reset to 0x0000. In the case of the Y90 MPU this information is available in the System Status Block.

The minimum width of the **resetb** signal is set by the flip-flops used in the design. The setup time for the **resetb** signal to the rising edge of the **clkc** signal is likewise determined by the flip-flops used in the design.

The **clearb** signal has the same timing requirements as the **resetb** signal. The **clearb** signal should only be used in the power-on case, and only affects those flip-flops not affected by the **resetb** signal.



Instruction Set

This chapter presents the assembly language syntax, addressing modes, flag settings, binary encoding, and execution time for the Y80 instruction set. The entire instruction set is presented in alphabetical order.

The assembly language syntax is identical to that used by the original Zilog assembler. Different assembler programs may or may not use identical syntax. The syntax is presented generically at the beginning of each instruction, with the details presented for each addressing mode later in each entry.

The operation of each instruction is specified in a format similar to Verilog HDL for minimum ambiguity, but no descriptive text or examples are included.

The effect of the instruction on each flag is listed, with a brief description. Normally the flags are updated by the main operation of the instruction, but for some complex instructions different flags may be affected by different parts of the instruction. This is specified in the description. The flags are organized as below in the F (Flag) register:

S	**Z**	**U5**	**H**	**U3**	**P/V**	**N**	**C**

These flags have the following meanings:

Flag	**Meaning**
S	Sign (a copy of the MSB of the result).
Z	Zero (indicating that the result was zero).
U5	Unused Bit 5 (an unused Flag register bit).
H	Half-Carry (carry out of the lower nibble, used for BCD math).
U3	Unused Bit 3 (an unused Flag register bit).
P/V	Parity/Overflow (parity of the result, or arithmetic overflow; depends on the instrcuction)
N	Negative (add/subtract flag, necessary for BCD math)
C	Carry (arithmetic carry, or shift linkage bit)

Fields in the instruction are listed using shortcuts for common fields. These shortcuts should be self-explanatory in most cases, but will be detailed here for completeness.

The most common field in the instruction specifies a CPU register, employing the following encoding:

rrr	**Register Selected**
000	B
001	C
010	D
011	E
100	H
101	L
111	A (Accumulator)

Word registers are similarly encoded, although the exact encoding depends on the instruction:

dd, ss, tt, xx or yy	**dd, ss Register**	**tt Register**	**xx Register**	**yy Register**
00	BC	BC	BC	BC
01	DE	DE	DE	DE
10	HL	HL	IX	IY
11	SP	AF	SP	SP

The execution time for instructions is always a multiple of two clocks.

ADC

Add With Carry

ADC A, src

src: R, IM, IR, X

Operation: A <= A + src + CF

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic carry out of bit 3; cleared otherwise.
P/V: Set if arithmetic overflow; cleared otherwise.
N: Cleared.
C: Set if arithmetic carry out of bit 7; cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	ADC A, r	10001rrr	2
IM:	ADC A, n	11001110	4
		----n---	
IR:	ADC A, (HL)	10001110	6
X:	ADC A, (IX+d) or ADC A, (IY+d)	11y11101	10
		10001110	
		----d---	

Notes:

1. The **rrr** field uses the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY

ADC

Add With Carry (Word)

ADC HL, src src: RR

Operation: HL <= HL + src + CF

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic carry out of bit 11; cleared otherwise.
P/V: Set if arithmetic overflow; cleared otherwise.
N: Cleared.
C: Set if arithmetic carry out of bit 15; cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
RR:	ADC HL, ss	11101101 01ss1010	4

Notes:

1. The **ss** field uses the standard word register encoding.

ADD

Add

ADD A, src src: R, IM, IR, X

Operation: A <= A + src

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic carry out of bit 3; cleared otherwise.
P/V: Set if arithmetic overflow; cleared otherwise.
N: Cleared.
C: Set if arithmetic carry out of bit 7; cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	ADD A, r	10000rrr	2
IM:	ADD A, n	11000110	4
		----n---	
IR:	ADD A, (HL)	10000110	6
X:	ADD A, (IX+d) or ADD A, (IY+d)	11y11101	10
		10000110	
		----d---	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY

ADD

Add (Word)

ADC dst, src

dst: HL, IX, IY
src: RR

Operation: dst <= dst + src

Flags:

S: Unaffected.
Z: Unaffected.
H: Set if arithmetic carry out of bit 11; cleared otherwise.
P/V: Unaffected.
N: Cleared.
C: Set if arithmetic carry out of bit 15; cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
RR:	ADD HL, ss	00ss1001	2
	ADC IX, xx	11011101 01xx1001	4
	ADC IY, yy	11111101 01yy1001	4

Notes:

1. The **ss**, **xx** and **yy** fields use the standard word register select encodings.

AND

Logical AND

AND A, src

src: R, IM, IR, X

Operation: A <= A & src

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set.
P/V: Set if parity of result even; cleared otherwise.
N: Cleared.
C: Cleared.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	AND A, r	10100rrr	2
IM:	AND A, n	11100110	4
		----n---	
IR:	AND A, (HL)	10100110	6
X:	AND A, (IX+d) or AND A, (IY+d)	11y11101	10
		10100110	
		----d---	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY

BIT

Bit Test

BIT b, src　　　　　src: R, IR, X

Operation: Z <= ~src[b]

Flags:
S: Unaffected.
Z: Set if tested bit is zero; cleared otherwise.
H: Set.
P/V: Unaffected.
N: Cleared.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	BIT b, r	11001011 01bbbrrr	4
IR:	BIT b, (HL)	10100110 01bbb1110	8
X:	BIT b, (IX+d) or BIT b, (IY+d)	11y11101 11001011 ----d--- 01bbb110	10

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

3. The **bbb** field uses normal binary encoding.

4. For the original Z80, the **S** and **C** flags are undefined.

CALL

Call Subroutine

CALL dst dst: DA

Operation:
SP <= SP - 2
(SP) <= PC
PC <= dst

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
DA:	CALL mn	11001101 ----n--- ----m---	10

CALL

Conditional Call Subroutine

CALL cc, dst

dst: DA

Operation:

```
if (cc = true) begin
    SP <= SP - 2
    (SP) <= PC
    PC <= dst
    end
```

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
DA:	CALL cc, mn	11fff100 ----n--- ----m---	10/6 (taken/not taken)

Notes:

1.

Mnemonic	Encoding (fff)	Meaning	Flag case
NZ	000	Non-zero	Z = 0
Z	001	Zero	Z = 1
NC	010	Non-carry	C = 0
C	011	Carry	C = 1
PO	100	Parity Odd	P/V = 0
PE	101	Parity Even	P/V = 1
P	110	Plus	S = 0
M	111	Minus	S = 1

CCF

Complement Carry Flag

CCF

Operation: CF <= ~CF

Flags:

S: Unaffected.
Z: Unaffected.
H: Copy of previous value of Carry flag.
P/V: Unaffected.
N: Cleared.
C: Set if previous Carry flag was zero; cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
IM:	CCF	00111111	2

CP

Compare

CP A, src — src: R, IM, IR, X

Operation: A - src

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 3; cleared otherwise.
P/V: Set if arithmetic overflow; cleared otherwise.
N: Set.
C: Set if arithmetic borrow out of bit 7; cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	CP A, r	10111rrr	2
IM:	CP A, n	11111110 ----n---	4
IR:	CP A, (HL)	10111110	6
X:	CP A, (IX+d) or CP (IY+d)	11y11101 10111110 ----d---	10

Notes:

1. The **rrr** field uses the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY

CPD

Compare and Decrement

CPD

Operation: A - (HL)
HL <= HL - 1
BC <= BC - 1

Flags:
S: Set if result of compare is negative, cleared otherwise.
Z: Set if result of compare is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 3 during compare; cleared otherwise.
P/V: Set if result of BC decrement is non-zero; cleared otherwise.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	CPD	11101101 10101001	10

CPDR

Compare, Decrement and Repeat

CPDR

Operation: A - (HL)
HL <= HL - 1
BC <= BC - 1
repeat if BC != 0 and A - (HL) != 0

Flags:
S: Set if result of compare is negative, cleared otherwise.
Z: Set if result of compare is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 3 during compare; cleared otherwise.
P/V: Set if result of BC decrement is non-zero; cleared otherwise.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	CPDR	11101101 10111001	8 + 4i

Notes:

1. This instruction can be interrupted after each iteration. The address saved on the stack in this case is the address of this instruction, allowing completion of the instruction after the interrupt service routine.

2. Interrupts are sampled during each memory read operation.

CPI

Compare and Increment

CPI

Operation: A - (HL)
HL <= HL + 1
BC <= BC - 1

Flags:
S: Set if result of compare is negative, cleared otherwise.
Z: Set if result of compare is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 3 during compare; cleared otherwise.
P/V: Set if result of decrementing BC is non-zero; cleared otherwise.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	CPI	11101101 10100001	10

CPIR

Compare, Increment and Repeat

CPIR

Operation:

A - (HL)
HL <= HL + 1
BC <= BC - 1
repeat if BC != 0 and A - (HL) != 0

Flags:

S: Set if result of compare is negative, cleared otherwise.
Z: Set if result of compare is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 3 during compare; cleared otherwise.
P/V: Set if result of decrementing BC is non-zero; cleared otherwise.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	CPIR	11101101 10110001	8 + 4i

Notes:

1. This instruction can be interrupted after each iteration. The address saved on the stack in this case is the address of this instruction, allowing completion of the instruction after the interrupt service routine

2. Interrupts are sampled during each memory read operation.

CPL

Complement

CPL

Operation: A <= ~A

Flags:

S: Unaffected.
Z: Unaffected.
H: Set.
P/V: Unaffected.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	CPL	00101111	2

DAA

Decimal Adjust Accumulator

DAA

Operation: A <= Decimal Adjust A

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: See table below.
P/V: Set if result has even parity; cleared otherwise.
N: Unaffected.
C: See table below.

Addressing Modes	**Assembly Syntax**	**Encoding**	**Clocks**
	DAA	00100111	2

Notes:

Instruction	C before DAA	A[7:4] before DAA	H before DAA	A[3:0] before DAA	Number added to A	C after DAA
ADC, ADD or 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
DEC, NEG, SUB or SBC	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

DEC

Decrement

DEC dst — dst: R, IR, X

Operation: dst <= dst - 1

Flags:
S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 3; cleared otherwise.
P/V: Set if arithmetic overflow; cleared otherwise.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	DEC r	00rrr101	2
IR:	DEC (HL)	00110101	8
X:	DEC (IX+d) or DEC (IY+d)	11y11101 00110101 ----d---	12

Notes:

1. The **rrr** field uses the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY

DEC

Decrement (Word)

DEC dst   dst: RR, IX, IY

Operation: dst <= dst - 1

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
RR:	DEC dd	00dd1011	2
IX, IY	DEC IX or DEC IY	11y11101 00101011	4

Notes:

1. The **dd** field uses the standard word register encoding.

DI

Disable Interrupt

DI

Operation: IFF1 <= 0
IFF2 <= 0

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	DI	11110011	2

Notes:

1. Interrupts are last sampled during the machine cycle that fetches this instruction.

DJNZ

Decrement, Jump if Non-zero

DJNZ e

Operation: B <= B - 1
if (B != 0) PC <= PC + e (where PC is the PC of this instruction)

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	DJNZ e	00010000 --(e-2)-	6

Notes:

1. Relative to the address of this instruction, the jump range is -126 to +129. Relative to the address of the next instruction, the jump range is -128 to +127.

EI

Enable Interrupt

EI

Operation: IFF1 <= 1
IFF2 <= 1

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	EI	11111011	2

Notes:

1. Interrupts are first sampled during the fetch of the next instruction. If an interrupt is pending this instruction fetch will be ignored and an interrupt acknowledge cycle started.

EX

Exchange with Top-of-Stack

EX (SP), src src: HL, IX, IY

Operation: (SP) <=> L or IXL or IYL
(SP+1) <=> H or IXH or IYH

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	EX (SP), HL	11100011	12
	EX (SP), IX or EX (SP), IY	11y11101 11100011	14

Notes:

1. **y** = 0 selects IX and **y** = 1 selects IY

EX AF, AF’

Exchange Accumulator

EX AF, AF’

Operation: AF <=> AF’

Flags:

S: Replaced by alternate flag.
Z: Replaced by alternate flag.
H: Replaced by alternate flag.
P/V: Replaced by alternate flag.
N: Replaced by alternate flag.
C: Replaced by alternate flag.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	EX AF, AF’	00001000	2

Notes:

1. No data is actually moved. Instead the registers are renamed.

EX

Exchange (Word)

EX DE, HL

Operation: DE <=> HL

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	EX DE, HL	11101011	2

EXX

Exchange Register Bank

EXX

Operation:	BC <=> BC’
	DE <=> DE’
	HL <=> HL’

Flags:	**S:** Unaffected.
	Z: Unaffected.
	H: Unaffected.
	P/V: Unaffected.
	N: Unaffected.
	C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	EXX	11011001	2

Notes:

1. No data is actually moved. Instead the registers are renamed.

HALT

Halt

HALT

Operation: activate Halt signal and wait for interrupt

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	HALT	01110110	4 + 2n

Notes:

1. The CPU halts with an idle bus until an interrupt is requested. The address pushed to the stack during the interrupt acknowledge is the address of the next instruction. During Halt the **mem_addr_out** and **io_addr_out** are driven with 0x0000, and the **mem_data_out** and **io_data_out** are driven with 0x00.

IM

Interrupt Mode

IM i

Operation: Set Interrupt Mode i

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	IM 0	11101101 01000110	4
	IM 1	11101101 01010110	4
	IM 2	11101101 01011110	4

Notes:

1. Interrupt Mode 0 always jumps to location 0x0038 in response to a maskable interrupt request.

2. Interrupt Mode 1 always jumps to location 0x0038 in response to a maskable interrupt request.

3. Interrupt Mode 2 uses the interrupt vector returned on the **ivec_bus** during an interrupt acknowledge cycle, along with the contents of the I register, to access an interrupt vector table in memory. The address stored at the selected location in the interrupt vector table is the starting addess of the interrupt service routine. Note that the least-significant bit of the interrupt vector must be zero to account for the two-byte entries in the interrupt vector table.

IN

Input

IN A, src

src: DA

Operation: A <= I/O(A:n)

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	IN A, (n)	11011011 ----n---	8

IN

Input

IN r, (C) dst: R

Operation: r <= I/O(BC)

Flags:

S: Set if the input data is negative; cleared otherwise.
Z: Set if the input data is zero; cleared otherwise.
H: Cleared.
P/V: Set if the parity of the input data is even; cleared otherwise.
N: Cleared.
C: Unaffected.

Addressing Modes	**Assembly Syntax**	**Encoding**	**Clocks**
	IN r, (C)	11101101 01rrr000	8

Notes:

1. The **rrr** field uses the standard register select encoding

INC

Increment

INC dst dst: R, IR, X

Operation: dst <= dst + 1

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic carry out of bit 3; cleared otherwise.
P/V: Set if arithmetic overflow; cleared otherwise.
N: Cleared.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	INC r	00rrr100	2
IR:	INC (HL)	00110100	8
X:	INC (IX+d) or INC (IY+d)	11y11101	12
		00110100	
		----d---	

Notes:

1. The **rrr** field uses the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY

INC

Increment (Word)

INC dst

dst: RR, IX, IY

Operation: dst <= dst + 1

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
RR:	INC ss	00dd0011	2
IX, IY	INC IX or INC IY	11y11101 00100011	4

Notes:

1. The **dd** field uses the standard word register encoding.

IND

Input and Decrement

IND

Operation: (HL) <= I/O(BC)
B <= B - 1
HL <= HL -1

Flags:
S: Unaffected.
Z: Set if result of decrementing B is zero; cleared otherwise.
H: Unaffected.
P/V: Unaffected.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	IND	11101101 10101010	10

Notes:

1. For the original Z80, the **S**, **H** and **P/V** flags are undefined.

INDR

Input, Decrement and Repeat

INDR

Operation: (HL) <= I/O(BC)
B <= B - 1
HL <= HL -1
repeat if B != 0

Flags:
S: Unaffected.
Z: Set if result of decrementing B is zero; cleared otherwise.
H: Unaffected.
P/V: Unaffected.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	INDR	11101101 10111010	8 + 4i

Notes:

1. This instruction can be interrupted after each iteration. The address saved on the stack in this case is the address of this instruction, allowing completion of the instruction after the interrupt service routine

2. Interrupts are sampled during each I/O read operation.

3. For the original Z80, the **S**, **H** and **P/V** flags are undefined.

INI

Input and Increment

INI

Operation: (HL) <= I/O(BC)
B <= B - 1
HL <= HL + 1

Flags:
S: Unaffected.
Z: Set if result of decrementing B is zero; cleared otherwise.
H: Unaffected.
P/V: Unaffected.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	INI	11101101 10100010	10

Notes:

1. For the original Z80, the **S**, **H** and **P/V** flags are undefined.

INIR

Input, Increment and Repeat

INIR

Operation:

```
(HL) <= I/O(BC)
B <= B - 1
HL <= HL + 1
repeat if B != 0
```

Flags:

S: Unaffected.
Z: Set if result of decrementing B is zero; cleared otherwise.
H: Unaffected.
P/V: Unaffected.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	INIR	11101101 10110010	8 + 6i

Notes:

1. This instruction can be interrupted after each iteration. The address saved on the stack in this case is the address of this instruction, allowing completion of the instruction after the interrupt service routine

2. Interrupts are sampled during each I/O read operation.

3. For the original Z80, the **S**, **H** and **P/V** flags are undefined.

JP

Jump

JP dst dst: IM, IR

Operation: PC <= dst

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
IR:	JP (HL)	11101001	4
	JP (IX) or JP (IY)	11y11101	6
		11101001	
IM:	JP mn	11000011	8
		----n---	
		----m---	

Notes:

1. The indirect jumps use the contents of the register directly for the jump address.

JP

Conditional Jump

JP cc, mn

Operation: if (cc = true) PC <= mn

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
IM:	JP cc, mn	11fff010 ----n--- ----m---	8 (taken) 6 (not taken)

Notes:

1.

Mnemonic	Encoding (fff)	Meaning	Flag case
NZ	000	Non-zero	Z = 0
Z	001	Zero	Z = 1
NC	010	Non-carry	C = 0
C	011	Carry	C = 1
PO	100	Parity Odd	P/V = 0
PE	101	Parity Even	P/V = 1
P	110	Plus	S = 0
M	111	Minus	S = 1

JR

Jump Relative

JR e

Operation: PC <= PC + e (where PC is the PC of this instruction)

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	JR e	00011000	6
		--(e-2)-	

Notes:

1. Relative to the address of this instruction, the jump range is -126 to +129. Relative to the address of the next instruction, the jump range is -128 to +127.

JR

Conditional Jump Relative

JR cc, e

Operation: if (cc = true) PC <= PC + e (where PC is the PC of this instruction)

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	JR cc, e	001cc000	6 (taken)
		--(e-2)-	4 (not taken)

Notes:

1. Relative to the address of this instruction, the jump range is -126 to +129. Relative to the address of the next instruction, the jump range is -128 to +127.

1.	Mnemonic	Encoding (cc)	Meaning	Flag case
	NZ	00	Non-zero	Z = 0
	Z	01	Zero	Z = 1
	NC	10	Non-carry	C = 0
	C	11	Carry	C = 1

LD

Load Accumulator from Memory

LD A, src	src: DA, IR

Operation: A <= src

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
DA:	LD A, (mn)	00111010	10 (8)
		----n---	
		----m---	
IR:	LD A, (BC)	00001010	6
	LD A, (DE)	00011010	6

LD

Load Accumulator from Special Register

LD A, src　　　　src: special register

Operation: A <= src

Flags:
S: Set if the contents of the Special Register is negative; cleared otherwise.
Z: Set if the contents of the Special Register is zero; cleared otherwise.
H: Cleared.
P/V: Loaded with the contents if the IFF2 interrupt enable flag.
N: Cleared.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	LD A, I	11101101	4
		01010111	
	LD A, R	11101101	4
		01011111	

LD

Load Memory from Accumulator

LD dst, A   dst: DA, IR

Operation: dst <= A

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
DA:	LD (mn), A	00110010	10 (8)
		----n---	
		----m---	
IR:	LD (BC), A	00000010	6
	LD (DE), A	00010010	6

LD

Load Memory with Immediate

LD dst, n

dst: IR, X

Operation: dst <= n

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
IR:	LD (HL), n	00110110	6
		----n---	
X:	LD (IX+d), n or LD (IY+d), n	11y11101	10
		00110110	
		----d---	
		----n---	

Notes:

1. **y** = 0 selects IX and **y** = 1 selects IY

LD

Load Memory from Register

LD dst, r | dst: IR, X

Operation: dst <= r

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
IR:	LD (HL), r	01110rrr	6
X:	LD (IX+d), r or LD (IY+d), r	11y11101 01110rrr	10

Notes:

1. The **rrr** field uses the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY

LD

Load Memory from Register (Word)

LD (mn), src

src: HL, RR, IX, IY

Operation: (mn) <= src

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
HL:	LD (mn), HL	00100010	12
		----n---	
		----m---	
RR:	LD (mn), ss	11101101	14
		01ss0011	
		----n---	
		----m---	
IX, IY:	LD (mn), IX or LD (mn), IY	11y11101	14
		00100010	
		----n---	
		----m---	

Notes:

1. The **ss** field uses the standard word register encoding.

LD

Load Register

LD r, src　　　dst: R, IM, IR, X

Operation: r <= src

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	LD rd, rs	01rdrrsr	2
IM	LD r, n	00rrr110	4
		----n---	
IR:	LD r, (HL)	01rrr110	6
X:	LD r, (IX+d) or LD r, (IY+d)	11y11101	10
		01rrr110	
		----d---	

Notes:

1. The **rdr, rsr** and **rrr** fields use the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY

LD

Load Register Immediate (Word)

LD dst, mn — dst: RR, IX, IY

Operation: dst <= mn

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
IM:	LD dd, mn	00dd0001	6
		----n---	
		----m---	
	LD IX, mn or LD IY, mn	11y11101	8
		00100001	
		----n---	
		----m---	

Notes:

1. The **dd** field uses the standard word register encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY

LD

Load Register (Word)

LD dst, (mn) dst: RR, IX, IY

Operation: dst <= (mn)

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
DA:	LD HL, (mn)	00101010	12
		----n---	
		----m---	
	LD dd, (mn)	11101101	14
		01dd1011	
		----n---	
		----m---	
	LD IX, (mn) or LD IY, (mn)	11y11101	14
		00101010	
		----n---	
		----m---	

Notes:

1. The **dd** field uses the standard word register encoding.

LD

Load Special Register from Accumulator

LD dst, A

dst: special register

Operation: dst <= A

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	LD I, A	11101101 01000111	4
	LD R, A	11101101 01001111	4

LD

Load Stack pointer

LD SP, src   src: HL, IX, IY

Operation: SP <=src

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	LD SP, HL	11111001	2
	LD SP, IX or LD SP, IY	11y11101	4
		11111001	

Notes:

2. **y** = 0 selects IX and **y** = 1 selects IY

LDD

Load and Decrement

LDD

Operation: (DE) <= (HL)
BC <= BC - 1
DE <= DE - 1
HL <= HL -1

Flags:
S: Unaffected.
Z: Unaffected.
H: Cleared.
P/V: Set if result of decrementing BC is non-zero; cleared otherwise.
N: Cleared.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	LDD	11101101 10101000	10

LDDR

Load, Decrement and Repeat

LDDR

Operation:

```
(DE) <= (HL)
BC <= BC - 1
DE <= DE - 1
HL <= HL -1
repeat if BC != 0
```

Flags:

S: Unaffected.
Z: Unaffected
H: Cleared.
P/V: Set if result of decrementing BC is non-zero; cleared otherwise.
N: Cleared.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	LDDR	11101101 10111010	8 + 4i

Notes:

1. This instruction can be interrupted after each iteration. The address saved on the stack in this case is the address of this instruction, allowing completion of the instruction after the interrupt service routine

2. Interrupts are sampled during each memory read operation.

LDI

Load and Increment

INI

Operation:

(DE) <= (HL)
BC <= BC - 1
DE <= DE + 1
HL <= HL + 1

Flags:

S: Unaffected.
Z: Unaffected.
H: Cleared.
P/V: Set if result of decrementing BC is non-zero; cleared otherwise.
N: Cleared.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	LDI	11101101 10100000	10

LDIR

Input, Increment and Repeat

LDIR

Operation:

(DE) <= (HL)
BC <= BC - 1
DE <= DE + 1
HL <= HL + 1
repeat if BC != 0

Flags:

S: Unaffected.
Z: Unaffected.
H: Cleared.
P/V: Set if result of decrementing BC is non-zero; cleared otherwise.
N: Cleared.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	LDIR	11101101 10110000	8 + 4i

Notes:

1. This instruction can be interrupted after each iteration. The address saved on the stack in this case is the address of this instruction, allowing completion of the instruction after the interrupt service routine

2. Interrupts are sampled during each memory read operation.

NEG

Operation: A <= 0 - A

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 3; cleared otherwise.
P/V: Set if arithmetic overflow (A was 0x80 before inst); cleared otherwise.
N: Cleared.
C: Set if arithmetic borrow out of bit 7 (A was not 0x00 before inst); cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	NEG	11101101 00100100	4

NOP

No Operation

NOP

Operation: none

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	NOP	00000000	2

OR

Logical OR

OR A, src

src: R, IM, IR, X

Operation: A <= A | src

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Cleared.
P/V: Set if parity of result is even; cleared otherwise.
N: Cleared.
C: Cleared.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	OR A, r	10110rrr	2
IM:	OR A, n	11110110	4
		----n---	
IR:	OR A, (HL)	10110110	6
X:	OR A, (IX+d) or OR A, (IY+d)	11y11101	10
		10110110	
		----d---	

Notes:

1. The **rrr** field uses the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY

OTDR

Output, Decrement and Repeat

OTDR

Operation:

```
I/O(BC) <= (HL)
B <= B - 1
HL <= HL -1
repeat if B != 0
```

Flags:

S: Unaffected.
Z: Set if result of decrementing B is zero; cleared otherwise.
H: Unaffected.
P/V: Unaffected.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	OTDR	11101101 10111011	8 + 4i

Notes:

1. This instruction can be interrupted after each iteration. The address saved on the stack in this case is the address of this instruction, allowing completion of the instruction after the interrupt service routine

2. Interrupts are sampled during each memory read operation.

3. For the original Z80, the **S**, **H** and **P/V** flags are undefined.

OTIR

Output, Increment and Repeat

OTIR

Operation:

I/O(BC) <= (HL)
B <= B - 1
HL <= HL + 1
repeat if B != 0

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Set if result of decrementing B is zero; cleared otherwise.
N: Cleared.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	OTIR	11101101 10110011	8 + 4i

Notes:

1. This instruction can be interrupted after each iteration. The address saved on the stack in this case is the address of this instruction, allowing completion of the instruction after the interrupt service routine

2. Interrupts are sampled during each memory read operation.

3. For the original Z80, the **S**, **H** and **P/V** flags are undefined.

OUT

Output

OUT dst, A

dst: DA

Operation: I/O(A:n) <= A

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	OUT (n), A	11010011 ----n---	8

OUT

Output

OUT (C), r src: R

Operation: I/O(BC) <= r

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	OUT (C), r	11101101 01rrr001	8

Notes:

1. The **rrr** field uses the standard register select encoding

OUTD

Output and Decrement

OUTD

Operation: I/O(BC) <= (HL)
B <= B - 1
HL <= HL -1

Flags:
S: Unaffected.
Z: Set if result of decrementing B is zero; cleared otherwise.
H: Unaffected.
P/V: Unaffected.
N: Set.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	OUTD	11101101 10101011	10

Notes:

1. For the original Z80, the **S**, **H** and **P/V** flags are undefined.

OUTI

Output and Increment

OUTI

Operation: I/O(BC) <= (HL)
B <= B - 1
HL <= HL + 1

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Set if result of decrementing B is zero; cleared otherwise.
N: Cleared.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	OUTI	11101101 10100011	10

Notes:

1. For the original Z80, the **S**, **H** and **P/V** flags are undefined.

POP

Pop from Stack

POP dst

dst: RR, IX, IY

Operation:

dst[lsb] <= (SP)
dst[msb] <= (SP+1)
SP <= SP + 2

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
RR:	POP tt	11tt0001	8
IX, IY	POP IX or POP IY	11y11101	10
		11100001	

Notes:

1. The **tt** field uses the standard word register encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY

PUSH

Push to Stack

PUSH src

src: RR, IX, IY

Operation:

(SP-1) <= src[msb]
(SP-2) <= src[lsb]
SP <= SP - 2

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
RR:	PUSH tt	11tt0101	8
IX, IY	PUSH IX or PUSH IY	11y11101	10
		11100101	

Notes:

1. The **tt** field uses the standard word register encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY

RES

Bit Reset

RES b, dst src: R, IR, X

Operation: dst[b] <= 0

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	RES b, r	11001011	4
		10bbbrrr	
IR:	RES b, (HL)	10100110	10
		10bbb1110	
X:	RES b, (IX+d) or RES (IY+d)	11y11101	12
		11001011	
		----d---	
		10bbb110	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

3. The **bbb** field uses normal binary encoding.

RET

Return from Subroutine

RET

Operation: PC[lsb] <= (SP)
PC[msb] <= (SP+1)
SP <= SP + 2

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	RET	11001001	10

RET

Conditional Return from Subroutine

RET cc

Operation:

```
if (cc = true) begin
    PC[lsb] <= (SP)
    PC[msb] <= (SP+1)
    SP <= SP + 2
    end
```

Flags:

S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	**Assembly Syntax**	**Encoding**	**Clocks**
	RET cc	11fff000	10 (taken) 2 (not taken)

Notes:

1.

Mnemonic	Encoding (fff)	Meaning	Flag case
NZ	000	Non-zero	Z = 0
Z	001	Zero	Z = 1
NC	010	Non-carry	C = 0
C	011	Carry	C = 1
PO	100	Parity Odd	P/V = 0
PE	101	Parity Even	P/V = 1
P	110	Plus	S = 0
M	111	Minus	S = 1

RETI

Return from Interrupt

RETI

Operation: PC[lsb] <= (SP)
PC[msb] <= (SP+1)
SP <= SP + 2

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	RETI	11101101 01001101	12

Notes:

1. This instruction activates the dedicated RETI signal out of the core.

RETN

Return from Non-Maskable Interrupt

RETN

Operation: PC[lsb] <= (SP)
PC[msb] <= (SP+1)
SP <= SP + 2
IFF2 <= IFF1

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	RETN	11001001	12
		01000101	

RL

Rotate Left

RL src src: R, IR, X

Operation: {CF, src} <= {src, CF}

Flags: **S:** Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Cleared.
P/V: Set if parity of result is even; cleared otherwise.
N: Cleared.
C: Data from bit 7.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	RL r	11001011	4
		00010rrr	
IR:	RL (HL)	10100110	10
		00010110	
X:	RL (IX+d) or RL (IY+d)	11y11101	12
		11001011	
		----d---	
		00010110	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

RLA

Rotate Left Accumulator

RLA

Operation: {CF, A} <= {A, CF}

Flags:
S: Unaffected
Z: Unaffected.
H: Cleared.
P/V: Unaffected.
N: Cleared.
C: Data from bit 7.

Addressing Modes	**Assembly Syntax**	**Encoding**	**Clocks**
	RLA	00010111	2

RLC

Rotate Left Circular

RLC src   src: R, IR, X

Operation: {CF, src} <= {src, src[7]}

Flags:
S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Cleared.
P/V: Set if parity of result is even; cleared otherwise.
N: Cleared.
C: Data from bit 7.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	RLC r	11001011	4
		00000rrr	
IR:	RLC (HL)	10100110	10
		00000110	
X:	RLC (IX+d) or RLC (IY+d)	11y11101	12
		11001011	
		----d---	
		00000110	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

RLCA

Rotate Left Circular Accumulator

RLCA

Operation: {CF, A} <= {A, A[7]}

Flags:
S: Unaffected
Z: Unaffected.
H: Cleared.
P/V: Unaffected.
N: Cleared.
C: Data from bit 7.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	RLCA	00000111	2

RLD

Rotate Left Digit

RLD

Operation: {A, (HL)} <= {A[7:4], (HL), A[3:0]}

Flags:
S: Set if A is negative after the operation; cleared otherwise.
Z: Set if A is zero after the operation; cleared otherwise.
H: Cleared.
P/V: Set if parity of A is even after the operation; cleared otherwise.
N: Cleared.
C: Unaffected

Addressing Modes	Assembly Syntax	Encoding	Clocks
	RLD	11101101	10
		01101111	

RR

Rotate Right

RR src src: R, IR, X

Operation: {src, CF} <= {CF, src}

Flags:
S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Cleared.
P/V: Set if parity of result is even; cleared otherwise.
N: Cleared.
C: Data from bit 0.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	RR r	11001011	4
		00011rrr	
IR:	RR (HL)	10100110	10
		00011110	
X:	RR (IX+d) or RR (IY+d)	11y11101	12
		11001011	
		----d---	
		00011110	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

RRA

Rotate Right Accumulator

RRA

Operation: {A, CF} <= {CF, A}

Flags:

S: Unaffected
Z: Unaffected.
H: Cleared.
P/V: Unaffected.
N: Cleared.
C: Data from bit 0.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	RRA	00011111	2

RRC

Rotate Right Circular

RRC src

src: R, IR, X

Operation: {src, CF} <= {src[0], src}

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Cleared.
P/V: Set if parity of result is even; cleared otherwise.
N: Cleared.
C: Data from bit 0.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	RRC r	11001011	4
		00001rrr	
IR:	RRC (HL)	10100110	10
		00001110	
X:	RRC (IX+d) or RRC (IY+d)	11y11101	12
		11001011	
		----d---	
		00001110	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

RRCA

Operation: {A, CF} <= {A[0], A}

Flags:
S: Unaffected
Z: Unaffected.
H: Cleared.
P/V: Unaffected.
N: Cleared.
C: Data from bit 0.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	RRCA	00001111	2

RRD

Rotate Right Digit

RRD

Operation: {A, (HL)} <= {A[7:4], (HL)[3:0], A[3:0], (HL)[7:4]}

Flags:

S: Set if A is negative after the operation; cleared otherwise.
Z: Set if A is zero after the operation; cleared otherwise.
H: Cleared.
P/V: Set if parity of A is even after the operation; cleared otherwise.
N: Cleared.
C: Unaffected

Addressing Modes	Assembly Syntax	Encoding	Clocks
	RRD	11101101 01100111	10

RST

Restart

RST v

Operation: SP <= SP - 2
(SP) <= PC
PC <= v

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	RST v	11vvv111	8

Notes:

1.

Mnemonic	Encoding (vvv)	Restart Address
0	000	0x0000
0x8	001	0x0008
0x10	010	0x0010
0x18	011	0x0018
0x20	100	0x0020
0x28	101	0x0028
0x30	110	0x0030
0x38	111	0x0038

SBC

Subtract With Carry

SBC A, src　　　　src: R, IM, IR, X

Operation: A <= A - src - CF

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 3; cleared otherwise.
P/V: Set if arithmetic overflow; cleared otherwise.
N: Cleared.
C: Set if arithmetic borrow out of bit 7; cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	SBC A, r	10011rrr	2
IM:	SBC A, n	11011110 ----n---	4
IR:	SBC A, (HL)	10011110	6
X:	SBC A, (IX+d) or SBC A, (IY+d)	11y11101 10011110 ----d---	10

Notes:

1. The **rrr** field uses the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY

SBC

Subtract With Carry (Word)

SBC HL, src

src: RR

Operation: HL <= HL - src - CF

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 11; cleared otherwise.
P/V: Set if arithmetic overflow; cleared otherwise.
N: Cleared.
C: Set if arithmetic carry out of bit 15; cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
RR:	SBC HL, ss	11101101 01ss0010	4

Notes:

1. The **ss** field uses the standard word register encoding.

SCF

Set Carry Flag

CCF

Operation: CF <= 1

Flags:

S: Unaffected.
Z: Unaffected.
H: Cleared.
P/V: Unaffected.
N: Cleared.
C: Set.

Addressing Modes	Assembly Syntax	Encoding	Clocks
	SCF	$\boxed{00110111}$	2

SET

Bit Set

SET b, dst src: R, IR, X

Operation: dst[b] <= 1

Flags:
S: Unaffected.
Z: Unaffected.
H: Unaffected.
P/V: Unaffected.
N: Unaffected.
C: Unaffected.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	SET b, r	11001011	4
		11bbbrrr	
IR:	SET b, (HL)	10100110	10
		11bbb1110	
X:	SET b, (IX+d) or SET b, (IY+d)	11y11101	12
		11001011	
		----d---	
		11bbb110	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

3. The **bbb** field uses normal binary encoding.

SLA

Shift Left Arithmetic

SLA src   src: R, IR, X

Operation: {CF, src} <= {src, 0}

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Cleared.
P/V: Set if parity of result is even; cleared otherwise.
N: Cleared.
C: Data from bit 7.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	SLA r	11001011	4
		00100rrr	
IR:	SLA (HL)	10100110	10
		00100110	
X:	SLA (IX+d) or SLA (IY+d)	11y11101	12
		11001011	
		----d---	
		00100110	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

SRA

Shift Right Arithmetic

SRA src src: R, IR, X

Operation: {src, CF} <= {src[7], src}

Flags:
S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Cleared.
P/V: Set if parity of result is even; cleared otherwise.
N: Cleared.
C: Data from bit 0.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	SRA r	11001011	4
		00101rrr	
IR:	SRA (HL)	10100110	10
		00101110	
X:	SRA (IX+d) or SRA (IY+d)	11y11101	12
		11001011	
		----d---	
		00101110	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

SRL

Shift Right Logical

SRL src src: R, IR, X

Operation: {src, CF} <= {0, src}

Flags:

S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Cleared.
P/V: Set if parity of result is even; cleared otherwise.
N: Cleared.
C: Data from bit 0.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	SRL r	11001011	4
		00111rrr	
IR:	SRL (HL)	10100110	10
		00111110	
X:	SRL (IX+d) or SRL (IY+d)	11y11101	12
		11001011	
		----d---	
		00111110	

Notes:

1. The **rrr** field uses the standard register select encoding.

2. **y** = 0 selects IX and **y** = 1 selects IY.

SUB

Subtract

SUB A, src　　　　src: R, IM, IR, X

Operation: A <= A - src

Flags:
S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Set if arithmetic borrow out of bit 3; cleared otherwise.
P/V: Set if arithmetic overflow; cleared otherwise.
N: Cleared.
C: Set if arithmetic borrow out of bit 7; cleared otherwise.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	SUB A, r	10010rrr	2
IM:	SUB A, n	11010110	4
		----n---	
IR:	SUB A, (HL)	10010110	6
X:	SUB A, (IX+d) or SUB A, (IY+d)	11y11101	10
		10010110	
		----d---	

Notes:

1. The **rrr** field uses the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY

XOR

Logical Exclusive-OR

XOR A, src　　　　src: R, IM, IR, X

Operation: A <= A ^ src

Flags:
S: Set if result is negative; cleared otherwise.
Z: Set if result is zero; cleared otherwise.
H: Cleared.
P/V: Set if parity of result is even; cleared otherwise.
N: Cleared.
C: Cleared.

Addressing Modes	Assembly Syntax	Encoding	Clocks
R:	XOR A, r	10101rrr	2
IM:	XOR A, n	11101110	4
		----n---	
IR:	XOR A, (HL)	10101110	6
X:	XOR A, (IX+d) or XOR A, (IY+d)	11y11101	10
		10101110	
		----d---	

Notes:

1. The **rrr** field uses the standard register select encoding

2. **y** = 0 selects IX and **y** = 1 selects IY