

# D

## 8085A Instruction Set Reference Encyclopedia<sup>1</sup>

<sup>1</sup> Courtesy of Intel Corporation.

## INSTRUCTION SET ENCYCLOPEDIA

In the ensuing dozen pages, the complete 8085A instruction set is described, grouped in order under five different functional headings, as follows:

1. **Data Transfer Group** — Moves data between registers or between memory locations and registers. Includes moves, loads, stores, and exchanges.
2. **Arithmetic Group** — Adds, subtracts, increments, or decrements data in registers or memory.
3. **Logic Group** — ANDs, ORs, XORs, compares, rotates, or complements data in registers or between memory and a register.
4. **Branch Group** — Initiates conditional or unconditional jumps, calls, returns, and restarts.
5. **Stack, I/O, and Machine Control Group** — Includes instructions for maintaining the stack, reading from input ports, writing to output ports, setting and reading interrupt masks, and setting and clearing flags.

The formats described in the encyclopedia reflect the assembly language processed by Intel-supplied assembler, used with the Intellec® development systems.

### Data Transfer Group

This group of instructions transfers data to and from registers and memory. **Condition flags are not affected by any instruction in this group.**

#### MOV r1, r2 (Move Register)

(r1) ← (r2)

The content of register r2 is moved to register r1.

0	1	D	D	D	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1  
 States: 4 (8085), 5 (8080)  
 Addressing: register  
 Flags: none

#### MOV r, M (Move from memory)

(r) ← ((H) (L))

The content of the memory location, whose address is in registers H and L, is moved to register r.

0	1	D	D	D	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: none

#### MOV M, r (Move to memory)

((H) (L)) ← (r)

The content of register r is moved to the memory location whose address is in registers H and L.

0	1	1	1	0	S	S	S
---	---	---	---	---	---	---	---

Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: none

#### MVI r, data (Move Immediate)

(r) ← (byte 2)

The content of byte 2 of the instruction is moved to register r.

0	0	D	D	D	1	1	0
data							

Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: none

#### MVI M, data (Move to memory immediate)

((H) (L)) ← (byte 2)

The content of byte 2 of the instruction is moved to the memory location whose address is in registers H and L.

0	0	1	1	0	1	1	0
data							

Cycles: 3  
 States: 10  
 Addressing: immed./reg. indirect  
 Flags: none



**LXI rp, data 16** (Load register pair immediate)  
 (rh) ← (byte 3),  
 (rl) ← (byte 2)  
 Byte 3 of the instruction is moved into the high-order register (rh) of the register pair rp. Byte 2 of the instruction is moved into the low-order register (rl) of the register pair rp.

0	0	R	P	0	0	0	1
low-order data							
high-order data							

Cycles: 3  
 States: 10  
 Addressing: immediate  
 Flags: none

**LDA addr** (Load Accumulator direct)  
 (A) ← ((byte 3)(byte 2))  
 The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register A.

0	0	1	1	1	0	1	0
low-order addr							
high-order addr							

Cycles: 4  
 States: 13  
 Addressing: direct  
 Flags: none

**STA addr** (Store Accumulator direct)  
 ((byte 3)(byte 2)) ← (A)  
 The content of the accumulator is moved to the memory location whose address is specified in byte 2 and byte 3 of the instruction.

0	0	1	1	0	0	1	0
low-order addr							
high-order addr							

Cycles: 4  
 States: 13  
 Addressing: direct  
 Flags: none

**LHLD addr** (Load H and L direct)  
 (L) ← ((byte 3)(byte 2))  
 (H) ← ((byte 3)(byte 2) + 1)  
 The content of the memory location, whose address is specified in byte 2 and byte 3 of the instruction, is moved to register L. The content of the memory location at the succeeding address is moved to register H.

0	0	1	0	1	0	1	0
low-order addr							
high-order addr							

Cycles: 5  
 States: 16  
 Addressing: direct  
 Flags: none

**SHLD addr** (Store H and L direct)  
 ((byte 3)(byte 2)) ← (L)  
 ((byte 3)(byte 2) + 1) ← (H)  
 The content of register L is moved to the memory location whose address is specified in byte 2 and byte 3. The content of register H is moved to the succeeding memory location.

0	0	1	0	0	0	1	0
low-order addr							
high-order addr							

Cycles: 5  
 States: 16  
 Addressing: direct  
 Flags: none

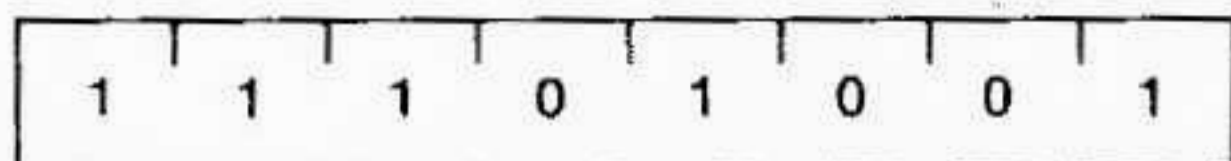
**LDAX rp** (Load accumulator indirect)  
 (A) ← ((rp))  
 The content of the memory location, whose address is in the register pair rp, is moved to register A. Note: only register pairs rp = B (registers B and C) or rp = D (registers D and E) may be specified.

0	0	R	P	1	0	1	0
---	---	---	---	---	---	---	---

Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: none



**PCHL** (Jump H and L indirect — move H and L to PC)  
 (PCH) ← (H)  
 (PCL) ← (L)  
 The content of register H is moved to the high-order eight bits of register PC. The content of register L is moved to the low-order eight bits of register PC.



Cycles: 1  
 States: 6 (8085), 5 (8080)  
 Addressing: register  
 Flags: none

## Stack, I/O, and Machine Control Group

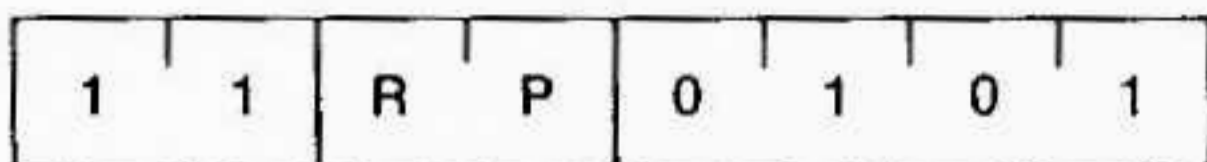
This group of instructions performs I/O, manipulates the Stack, and alters internal control flags.

Unless otherwise specified, **condition flags are not affected by any instructions in this group.**

**PUSH rp** (Push)

((SP) - 1) ← (rh)  
 ((SP) - 2) ← (rl)  
 ((SP) ← (SP) - 2

The content of the high-order register of register pair rp is moved to the memory location whose address is one less than the content of register SP. The content of the low-order register of register pair rp is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. **Note: Register pair rp = SP may not be specified.**

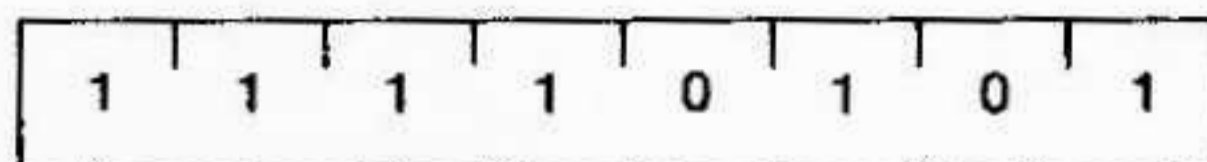


Cycles: 3  
 States: 12 (8085), 11 (8080)  
 Addressing: reg. indirect  
 Flags: none

**PUSH PSW** (Push processor status word)

((SP) - 1) ← (A)  
 ((SP) - 2)<sub>0</sub> ← (CY), ((SP) - 2)<sub>1</sub> ← X  
 ((SP) - 2)<sub>2</sub> ← (P), ((SP) - 2)<sub>3</sub> ← X  
 ((SP) - 2)<sub>4</sub> ← (AC), ((SP) - 2)<sub>5</sub> ← X  
 ((SP) - 2)<sub>6</sub> ← (Z), ((SP) - 2)<sub>7</sub> ← (S)  
 (SP) ← (SP) - 2                      X: Undefined.

The content of register A is moved to the memory location whose address is one less than register SP. The contents of the condition flags are assembled into a processor status word and the word is moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two.



Cycles: 3  
 States: 12 (8085), 11 (8080)  
 Addressing: reg. indirect  
 Flags: none

## FLAG WORD

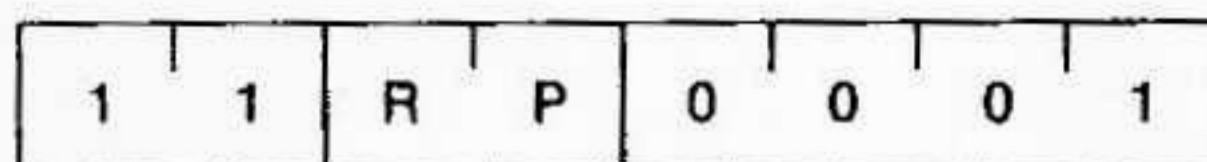
D <sub>7</sub>	D <sub>6</sub>	D <sub>5</sub>	D <sub>4</sub>	D <sub>3</sub>	D <sub>2</sub>	D <sub>1</sub>	D <sub>0</sub>
S	Z	X	AC	X	P	X	CY

X: undefined

**POP rp** (Pop)

(rl) ← ((SP))  
 (rh) ← ((SP) + 1)  
 (SP) ← (SP) + 2

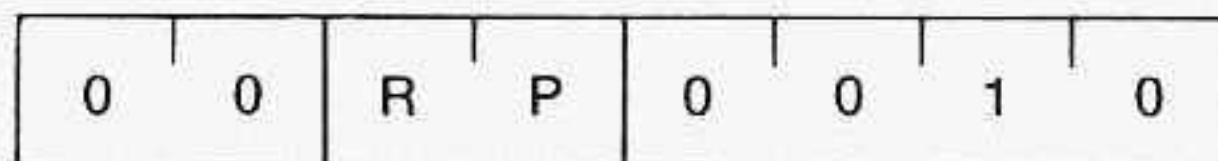
The content of the memory location, whose address is specified by the content of register SP, is moved to the low-order register of register pair rp. The content of the memory location, whose address is one more than the content of register SP, is moved to the high-order register of register pair rp. The content of register SP is incremented by 2. **Note: Register pair rp = SP may not be specified.**



Cycles: 3  
 States: 10  
 Addressing: reg.indirect  
 Flags: none

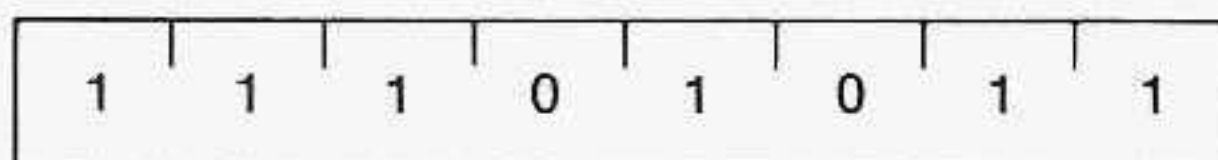


**STAX rp** (Store accumulator indirect)  
 $((rp)) \leftarrow (A)$   
 The content of register A is moved to the memory location whose address is in the register pair rp. Note: only register pairs  $rp = B$  (registers B and C) or  $rp = D$  (registers D and E) may be specified.



Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: none

**XCHG** (Exchange H and L with D and E)  
 $(H) \leftrightarrow (D)$   
 $(L) \leftrightarrow (E)$   
 The contents of registers H and L are exchanged with the contents of registers D and E.



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: none

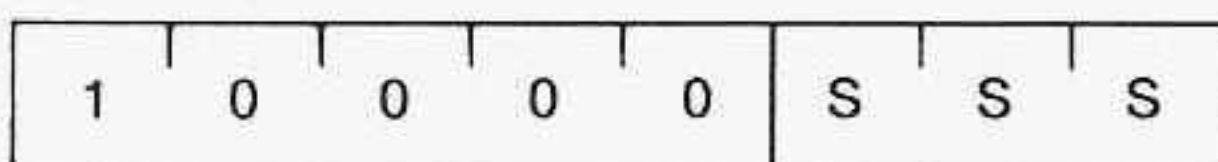
## Arithmetic Group

This group of instructions performs arithmetic operations on data in registers and memory.

**Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Carry, and Auxiliary Carry flags according to the standard rules.**

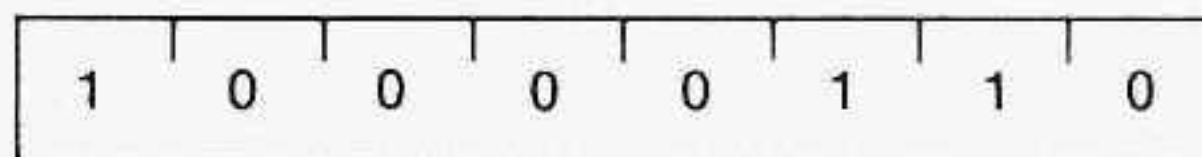
All subtraction operations are performed via two's complement arithmetic and set the carry flag to one to indicate a borrow and clear it to indicate no borrow.

**ADD r** (Add Register)  
 $(A) \leftarrow (A) + (r)$   
 The content of register r is added to the content of the accumulator. The result is placed in the accumulator.



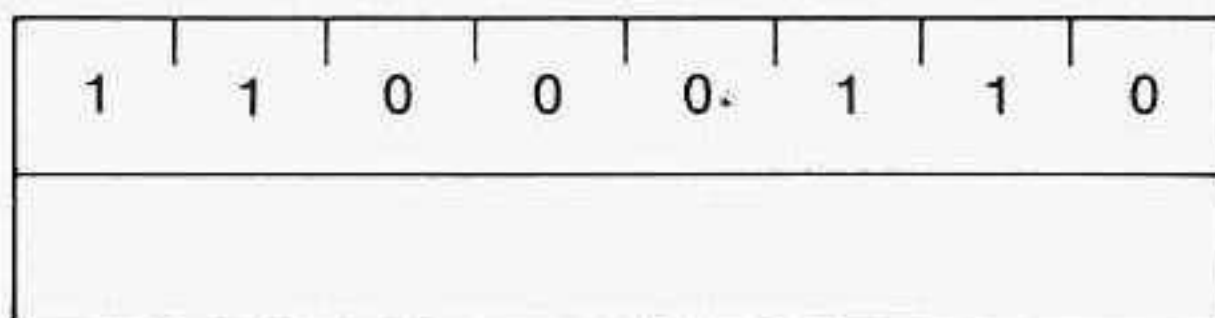
Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

**ADD M** (Add memory)  
 $(A) \leftarrow (A) + ((H) (L))$   
 The content of the memory location whose address is contained in the H and L registers is added to the content of the accumulator. The result is placed in the accumulator.



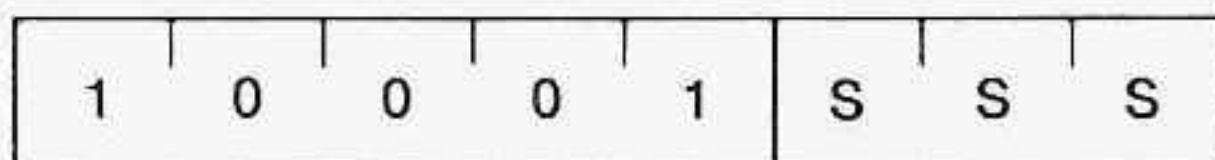
Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**ADI data** (Add immediate)  
 $(A) \leftarrow (A) + (\text{byte 2})$   
 The content of the second byte of the instruction is added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**ADC r** (Add Register with carry)  
 $(A) \leftarrow (A) + (r) + (CY)$   
 The content of register r and the content of the carry bit are added to the content of the accumulator. The result is placed in the accumulator.



Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

# THE INSTRUCTION SET

## ADC M (Add memory with carry)

$(A) \leftarrow (A) + ((H) (L)) + (CY)$

The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are added to the accumulator. The result is placed in the accumulator.

1	0	0	0	1	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

## SUB M (Subtract memory)

$(A) \leftarrow (A) - ((H) (L))$

The content of the memory location whose address is contained in the H and L registers is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	0	0	1	0	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

## ACI data (Add immediate with carry)

$(A) \leftarrow (A) + (\text{byte 2}) + (CY)$

The content of the second byte of the instruction and the content of the CY flag are added to the contents of the accumulator. The result is placed in the accumulator.

1	1	0	0	1	1	1	0
data							

Cycles: 2  
States: 7  
Addressing: immediate  
Flags: Z,S,P,CY,AC

## SUI data (Subtract immediate)

$(A) \leftarrow (A) - (\text{byte 2})$

The content of the second byte of the instruction is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	1	0	1	0	1	1	0
data							

Cycles: 2  
States: 7  
Addressing: immediate  
Flags: Z,S,P,CY,AC

## SUB r (Subtract Register)

$(A) \leftarrow (A) - (r)$

The content of register r is subtracted from the content of the accumulator. The result is placed in the accumulator.

1	0	0	1	0	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1  
States: 4  
Addressing: register  
Flags: Z,S,P,CY,AC

## SBB r (Subtract Register with borrow)

$(A) \leftarrow (A) - (r) - (CY)$

The content of register r and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.

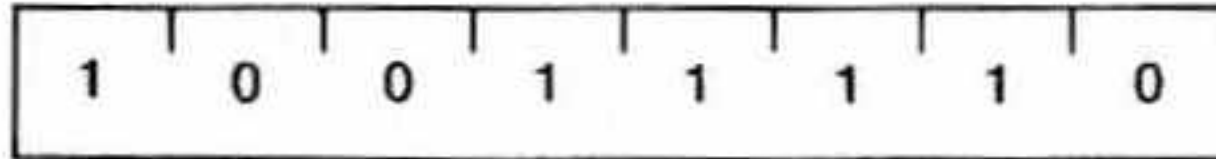
1	0	0	1	1	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1  
States: 4  
Addressing: register  
Flags: Z,S,P,CY,AC



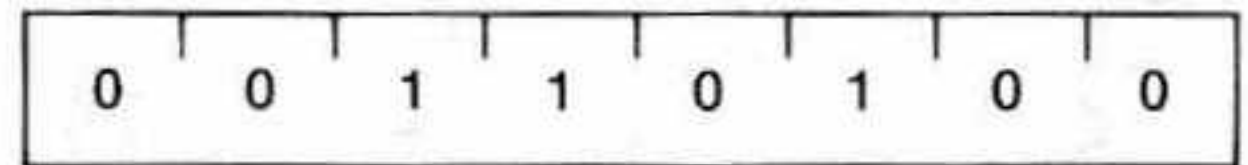
# THE INSTRUCTION SET

**SBB M** (Subtract memory with borrow)  
 $(A) \leftarrow (A) - ((H) (L)) - (CY)$   
 The content of the memory location whose address is contained in the H and L registers and the content of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



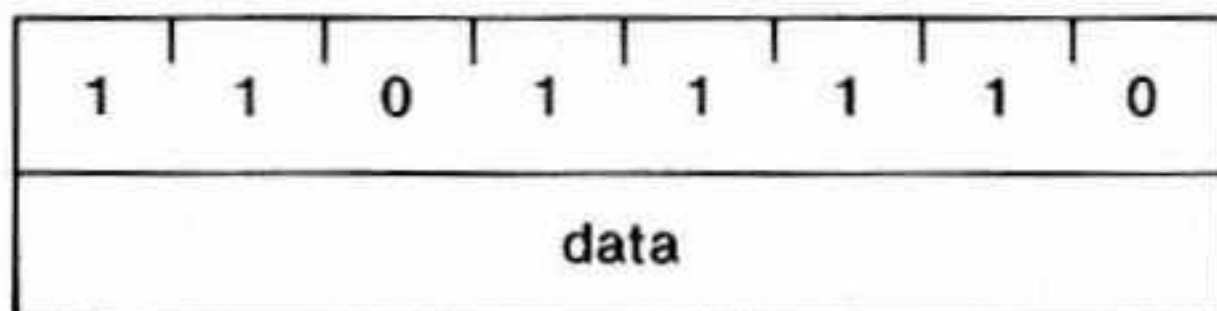
Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

**INR M** (Increment memory)  
 $((H) (L)) \leftarrow ((H) (L)) + 1$   
 The content of the memory location whose address is contained in the H and L registers is incremented by one. Note: All condition flags **except CY** are affected.



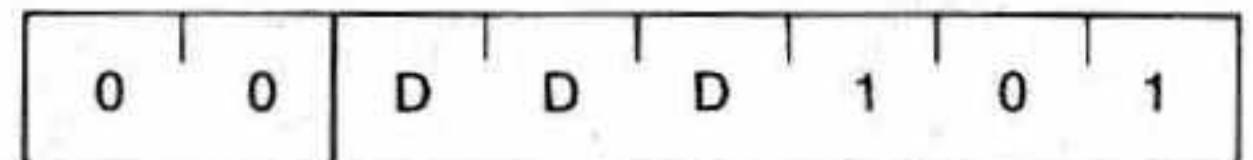
Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: Z,S,P,AC

**SBI data** (Subtract immediate with borrow)  
 $(A) \leftarrow (A) - (\text{byte 2}) - (CY)$   
 The contents of the second byte of the instruction and the contents of the CY flag are both subtracted from the accumulator. The result is placed in the accumulator.



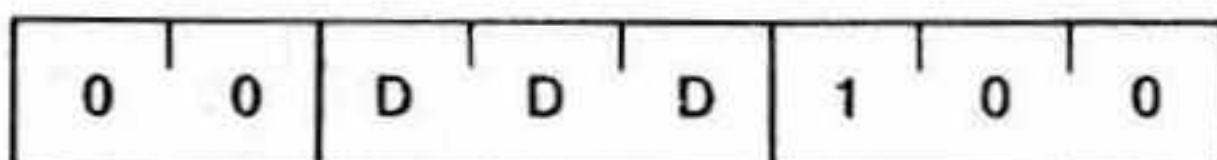
Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

**DCR r** (Decrement Register)  
 $(r) \leftarrow (r) - 1$   
 The content of register r is decremented by one. Note: All condition flags **except CY** are affected.



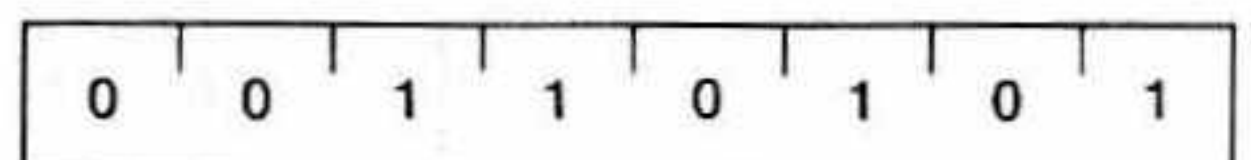
Cycles: 1  
 States: 4 (8085), 5 (8080)  
 Addressing: register  
 Flags: Z,S,P,AC

**INR r** (Increment Register)  
 $(r) \leftarrow (r) + 1$   
 The content of register r is incremented by one. Note: All condition flags **except CY** are affected.



Cycles: 1  
 States: 4 (8085), 5 (8080)  
 Addressing: register  
 Flags: Z,S,P,AC

**DCR M** (Decrement memory)  
 $((H) (L)) \leftarrow ((H) (L)) - 1$   
 The content of the memory location whose address is contained in the H and L registers is decremented by one. Note: All condition flags **except CY** are affected.



Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: Z,S,P,AC

**INX rp** (Increment register pair)  
 $(rh) (rl) \leftarrow (rh) (rl) + 1$   
 The content of the register pair rp is incremented by one. Note: **No condition flags are affected.**

0	0	R	P	0	0	1	1
---	---	---	---	---	---	---	---

Cycles: 1  
 States: 6 (8085), 5 (8080)  
 Addressing: register  
 Flags: none

**DCX rp** (Decrement register pair)  
 $(rh) (rl) \leftarrow (rh) (rl) - 1$   
 The content of the register pair rp is decremented by one. Note: **No condition flags are affected.**

0	0	R	P	1	0	1	1
---	---	---	---	---	---	---	---

Cycles: 1  
 States: 6 (8085), 5 (8080)  
 Addressing: register  
 Flags: none

**DAD rp** (Add register pair to H and L)  
 $(H) (L) \leftarrow (H) (L) + (rh) (rl)$   
 The content of the register pair rp is added to the content of the register pair H and L. The result is placed in the register pair H and L. Note: **Only the CY flag is affected.** It is set if there is a carry out of the double precision add; otherwise it is reset.

0	0	R	P	1	0	0	1
---	---	---	---	---	---	---	---

Cycles: 3  
 States: 10  
 Addressing: register  
 Flags: CY

**DAA** (Decimal Adjust Accumulator)  
 The eight-bit number in the accumulator is adjusted to form two four-bit Binary-Coded-Decimal digits by the following process:

1. If the value of the least significant 4 bits of the accumulator is greater than 9 or if the AC flag is set, 6 is added to the accumulator.
2. If the value of the most significant 4 bits of the accumulator is now greater than 9, or if the CY flag is set, 6 is added to the most significant 4 bits of the accumulator.

NOTE: All flags are affected.

0	0	1	0	0	1	1	1
---	---	---	---	---	---	---	---

Cycles: 1  
 States: 4  
 Flags: Z,S,P,CY,AC

## Logical Group

This group of instructions performs logical (Boolean) operations on data in registers and memory and on condition flags.

Unless indicated otherwise, all instructions in this group affect the Zero, Sign, Parity, Auxiliary Carry, and Carry flags according to the standard rules.

**ANA r** (AND Register)  
 $(A) \leftarrow (A) \wedge (r)$   
 The content of register r is logically ANDed with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set (8085). The CY flag is cleared and AC is set to the OR'ing of bits 3 of the operands (8080).**

1	0	1	0	0	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC



## ANA M (AND memory)

$(A) \leftarrow (A) \wedge ((H) (L))$

The contents of the memory location whose address is contained in the H and L registers is logically ANDed with the content of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set (8085). The CY flag is cleared and AC is set to the OR'ing of bits 3 of the operands (8080).**

1	0	1	0	0	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

## ANI data (AND immediate)

$(A) \leftarrow (A) \wedge (\text{byte } 2)$

The content of the second byte of the instruction is logically ANDed with the contents of the accumulator. The result is placed in the accumulator. **The CY flag is cleared and AC is set (8085). The CY flag is cleared and AC is set to the OR'ing of bits 3 of the operands (8080).**

1	1	1	0	0	1	1	0
data							

Cycles: 2  
States: 7  
Addressing: immediate  
Flags: Z,S,P,CY,AC

## XRA r (Exclusive OR Register)

$(A) \leftarrow (A) \vee (r)$

The content of register r is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	0	1	0	1	S	S	S
---	---	---	---	---	---	---	---

Cycles: 1  
States: 4  
Addressing: register  
Flags: Z,S,P,CY,AC

## XRA M (Exclusive OR Memory)

$(A) \leftarrow (A) \vee ((H) (L))$

The content of the memory location whose address is contained in the H and L registers is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---

Cycles: 2  
States: 7  
Addressing: reg. indirect  
Flags: Z,S,P,CY,AC

## XRI data (Exclusive OR immediate)

$(A) \leftarrow (A) \vee (\text{byte } 2)$

The content of the second byte of the instruction is exclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	1	1	0	1	1	1	0
data							

Cycles: 2  
States: 7  
Addressing: immediate  
Flags: Z,S,P,CY,AC

## ORA r (OR Register)

$(A) \leftarrow (A) \vee (r)$

The content of register r is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

1	0	1	1	0	S	S	S
---	---	---	---	---	---	---	---

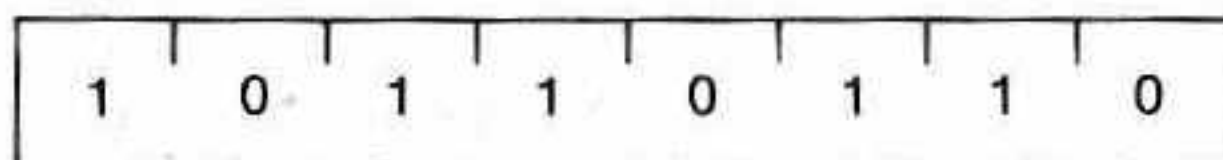
Cycles: 1  
States: 4  
Addressing: register  
Flags: Z,S,P,CY,AC



## ORA M (OR memory)

(A) ← (A) V ((H) (L))

The content of the memory location whose address is contained in the H and L registers is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

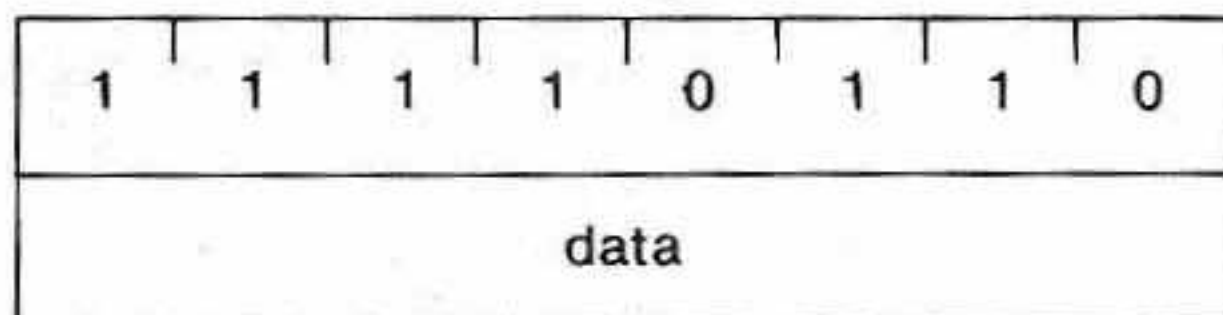


Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

## ORI data (OR Immediate)

(A) ← (A) V (byte 2)

The content of the second byte of the instruction is inclusive-OR'd with the content of the accumulator. The result is placed in the accumulator. **The CY and AC flags are cleared.**

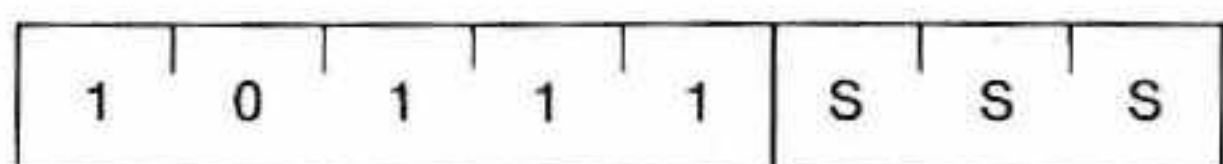


Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

## CMP r (Compare Register)

(A) ← (r)

The content of register r is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = (r). The CY flag is set to 1 if (A) < (r).**

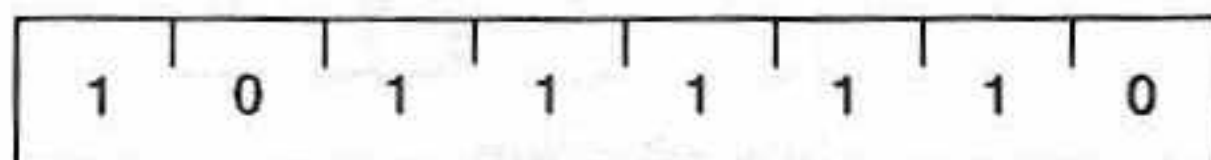


Cycles: 1  
 States: 4  
 Addressing: register  
 Flags: Z,S,P,CY,AC

## CMP M (Compare memory)

(A) ← ((H) (L))

The content of the memory location whose address is contained in the H and L registers is subtracted from the accumulator. The accumulator remains unchanged. The condition flags are set as a result of the subtraction. **The Z flag is set to 1 if (A) = ((H) (L)). The CY flag is set to 1 if (A) < ((H) (L)).**

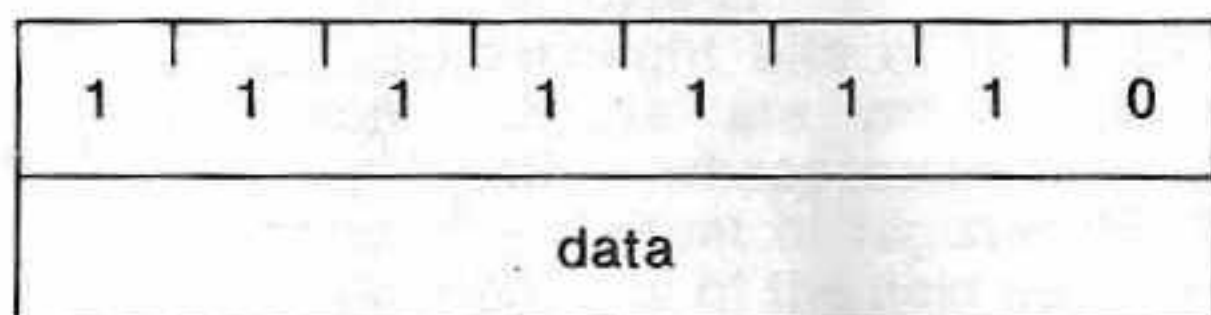


Cycles: 2  
 States: 7  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

## CPI data (Compare immediate)

(A) ← (byte 2)

The content of the second byte of the instruction is subtracted from the accumulator. The condition flags are set by the result of the subtraction. **The Z flag is set to 1 if (A) = (byte 2). The CY flag is set to 1 if (A) < (byte 2).**



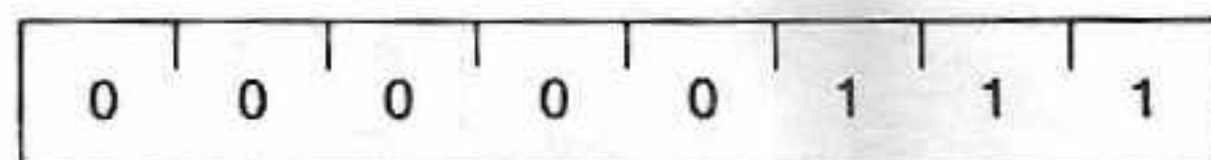
Cycles: 2  
 States: 7  
 Addressing: immediate  
 Flags: Z,S,P,CY,AC

## RLC (Rotate left)

(A<sub>n+1</sub>) ← (A<sub>n</sub>); (A<sub>0</sub>) ← (A<sub>7</sub>)

(CY) ← (A<sub>7</sub>)

The content of the accumulator is rotated left one position. The low order bit and the CY flag are both set to the value shifted out of the high order bit position. **Only the CY flag is affected.**



Cycles: 1  
 States: 4  
 Flags: CY

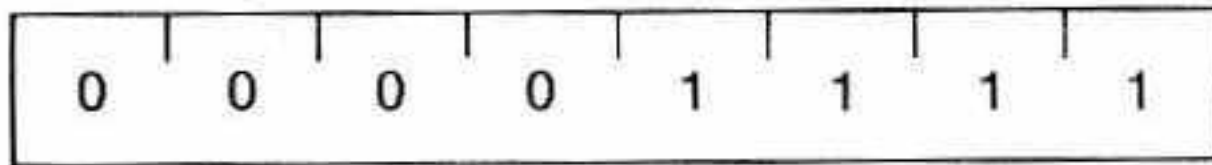


# THE INSTRUCTION SET

## RRC (Rotate right)

$(A_n) \leftarrow (A_{n+1}); (A_7) \leftarrow (A_0)$   
 $(CY) \leftarrow (A_0)$

The content of the accumulator is rotated right one position. The high order bit and the CY flag are both set to the value shifted out of the low order bit position. **Only the CY flag is affected.**

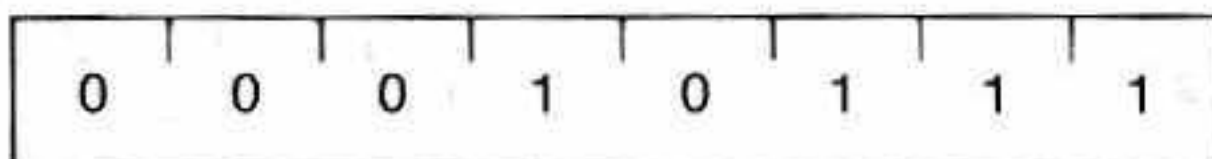


Cycles: 1  
 States: 4  
 Flags: CY

## RAL (Rotate left through carry)

$(A_{n+1}) \leftarrow (A_n); (CY) \leftarrow (A_7)$   
 $(A_0) \leftarrow (CY)$

The content of the accumulator is rotated left one position through the CY flag. The low order bit is set equal to the CY flag and the CY flag is set to the value shifted out of the high order bit. **Only the CY flag is affected.**

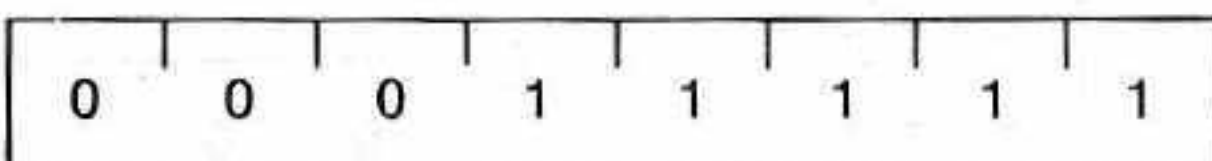


Cycles: 1  
 States: 4  
 Flags: CY

## RAR (Rotate right through carry)

$(A_n) \leftarrow (A_{n+1}); (CY) \leftarrow (A_0)$   
 $(A_7) \leftarrow (CY)$

The content of the accumulator is rotated right one position through the CY flag. The high order bit is set to the CY flag and the CY flag is set to the value shifted out of the low order bit. **Only the CY flag is affected.**

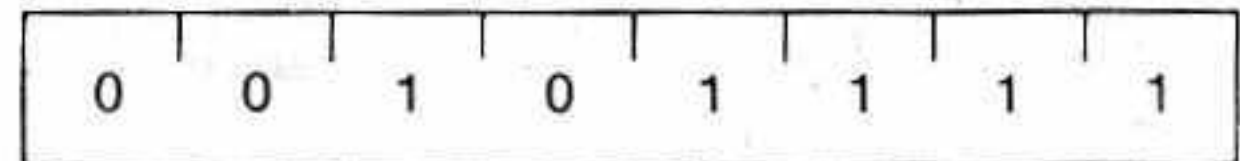


Cycles: 1  
 States: 4  
 Flags: CY

## CMA (Complement accumulator)

$(A) \leftarrow (\bar{A})$

The contents of the accumulator are complemented (zero bits become 1, one bits become 0). **No flags are affected.**

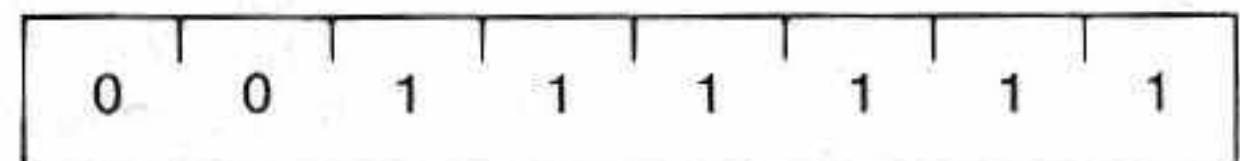


Cycles: 1  
 States: 4  
 Flags: none

## CMC (Complement carry)

$(CY) \leftarrow (\bar{CY})$

The CY flag is complemented. **No other flags are affected.**

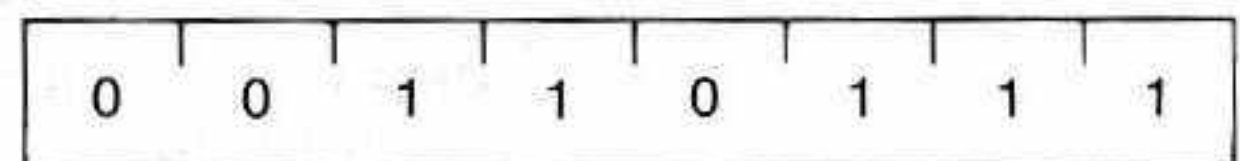


Cycles: 1  
 States: 4  
 Flags: CY

## STC (Set carry)

$(CY) \leftarrow 1$

The CY flag is set to 1. **No other flags are affected.**



Cycles: 1  
 States: 4  
 Flags: CY

## Branch Group

This group of instructions alter normal sequential program flow.

**Condition flags are not affected** by any instruction in this group.

The two types of branch instructions are unconditional and conditional. Unconditional transfers simply perform the specified operation on register PC (the program counter). Conditional transfers examine the status of one of the four processor flags to determine if the specified branch is to be executed. The conditions that may be specified are as follows:

CONDITION	CCC
NZ — not zero ( $Z = 0$ )	000
Z — zero ( $Z = 1$ )	001
NC — no carry ( $CY = 0$ )	010
C — carry ( $CY = 1$ )	011
PO — parity odd ( $P = 0$ )	100
PE — parity even ( $P = 1$ )	101
P — plus ( $S = 0$ )	110
M — minus ( $S = 1$ )	111

## JMP addr (Jump)

(PC) ← (byte 3) (byte 2)

Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

1	1	0	0	0	0	1	1
low-order addr							
high-order addr							

Cycles: 3  
States: 10  
Addressing: immediate  
Flags: none

## Jcondition addr (Conditional jump)

If (CCC),

(PC) ← (byte 3) (byte 2)

If the specified condition is true, control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction; otherwise, control continues sequentially.

1	1	C	C	C	0	1	0
low-order addr							
high-order addr							

Cycles: 2/3 (8085), 3 (8080)  
States: 7/10 (8085), 10 (8080)  
Addressing: immediate  
Flags: none

## CALL addr (Call)

((SP) - 1) ← (PCH)

((SP) - 2) ← (PCL)

(SP) ← (SP) - 2

(PC) ← (byte 3) (byte 2)

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by 2. Control is transferred to the instruction whose address is specified in byte 3 and byte 2 of the current instruction.

1	1	0	0	1	1	0	1
low-order addr							
high-order addr							

Cycles: 5  
States: 18 (8085), 17 (8080)  
Addressing: immediate/  
reg. indirect  
Flags: none



## Ccondition addr (Condition call)

If (CCC),

$((SP) - 1) - (PCH)$

$((SP) - 2) - (PCL)$

$(SP) - (SP) - 2$

$(PC) - (\text{byte 3}) (\text{byte 2})$

If the specified condition is true, the actions specified in the CALL instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	1	0	0
low-order addr							
high-order addr							

Cycles: 2/5 (8085), 3/5 (8080)

States: 9/18 (8085), 11/17 (8080)

Addressing: immediate/  
reg. indirect

Flags: none

## RET (Return)

$(PCL) - ((SP));$

$(PCH) - ((SP) + 1);$

$(SP) - (SP) + 2;$

The content of the memory location whose address is specified in register SP is moved to the low-order eight bits of register PC. The content of the memory location whose address is one more than the content of register SP is moved to the high-order eight bits of register PC. The content of register SP is incremented by 2.

1	1	0	0	1	0	0	1
---	---	---	---	---	---	---	---

Cycles: 3

States: 10

Addressing: reg. indirect

Flags: none

## Rcondition (Conditional return)

If (CCC),

$(PCL) - ((SP))$

$(PCH) - ((SP) + 1)$

$(SP) - (SP) + 2$

If the specified condition is true, the actions specified in the RET instruction (see above) are performed; otherwise, control continues sequentially.

1	1	C	C	C	0	0	0
---	---	---	---	---	---	---	---

Cycles: 1/3

States: 6/12 (8085), 5/11 (8080)

Addressing: reg. indirect

Flags: none

## RST n (Restart)

$((SP) - 1) - (PCH)$

$((SP) - 2) - (PCL)$

$(SP) - (SP) - 2$

$(PC) - 8 * (NNN)$

The high-order eight bits of the next instruction address are moved to the memory location whose address is one less than the content of register SP. The low-order eight bits of the next instruction address are moved to the memory location whose address is two less than the content of register SP. The content of register SP is decremented by two. Control is transferred to the instruction whose address is eight times the content of NNN.

1	1	N	N	N	1	1	1
---	---	---	---	---	---	---	---

Cycles: 3

States: 12 (8085), 11 (8080)

Addressing: reg. indirect

Flags: none

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	N	N	N	0	0

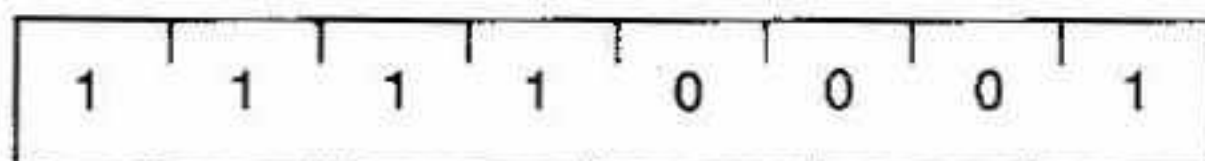
Program Counter After Restart

# THE INSTRUCTION SET

## POP PSW (Pop processor status word)

$(CY) \leftarrow ((SP))_0$   
 $(P) \leftarrow ((SP))_2$   
 $(AC) \leftarrow ((SP))_4$   
 $(Z) \leftarrow ((SP))_6$   
 $(S) \leftarrow ((SP))_7$   
 $(A) \leftarrow ((SP) + 1)$   
 $(SP) \leftarrow (SP) + 2$

The content of the memory location whose address is specified by the content of register SP is used to restore the condition flags. The content of the memory location whose address is one more than the content of register SP is moved to register A. The content of register SP is incremented by 2.

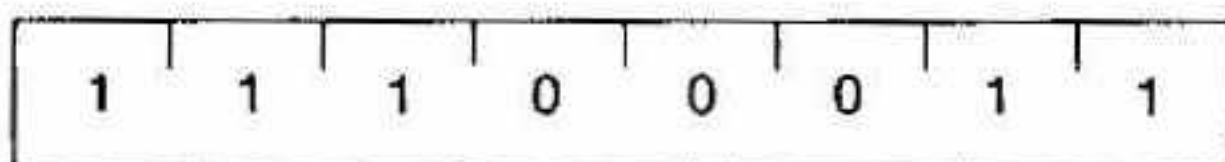


Cycles: 3  
 States: 10  
 Addressing: reg. indirect  
 Flags: Z,S,P,CY,AC

## XTHL (Exchange stack top with H and L)

$(L) \leftrightarrow ((SP))$   
 $(H) \leftrightarrow ((SP) + 1)$

The content of the L register is exchanged with the content of the memory location whose address is specified by the content of register SP. The content of the H register is exchanged with the content of the memory location whose address is one more than the content of register SP.

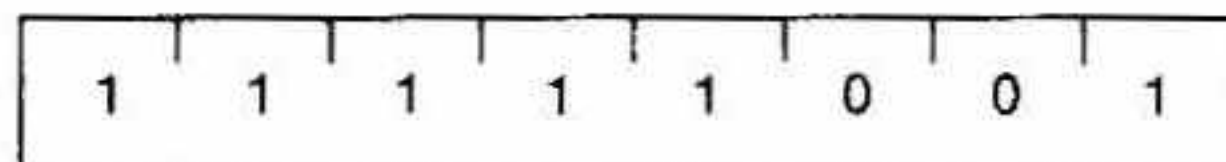


Cycles: 5  
 States: 16 (8085), 18 (8080)  
 Addressing: reg. indirect  
 Flags: none

## SPHL (Move HL to SP)

$(SP) \leftarrow (H) (L)$

The contents of registers H and L (16 bits) are moved to register SP.

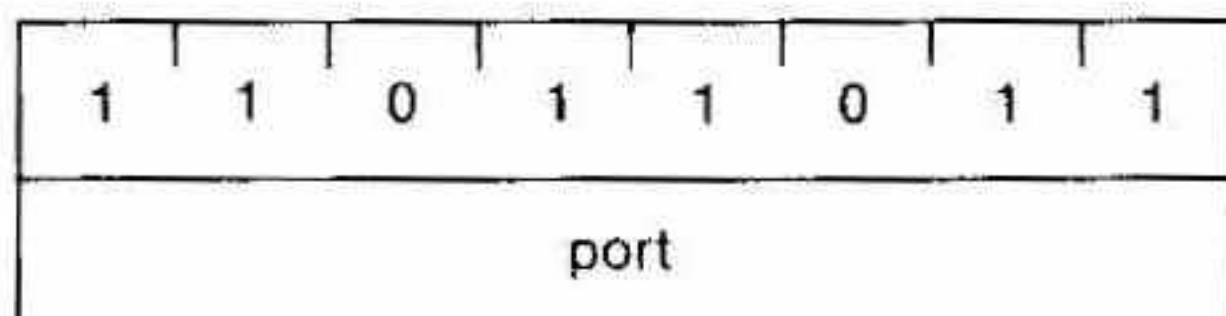


Cycles: 1  
 States: 6 (8085), 5 (8080)  
 Addressing: register  
 Flags: none

## IN port (Input)

$(A) \leftarrow \langle \text{data} \rangle$

The data placed on the eight bit bi-directional data bus by the specified port is moved to register A.

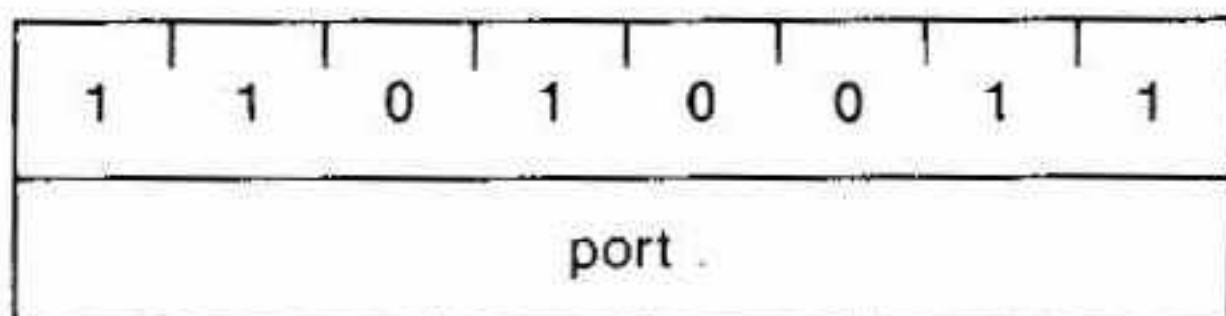


Cycles: 3  
 States: 10  
 Addressing: direct  
 Flags: none

## OUT port (Output)

$\langle \text{data} \rangle \leftarrow (A)$

The content of register A is placed on the eight bit bi-directional data bus for transmission to the specified port.

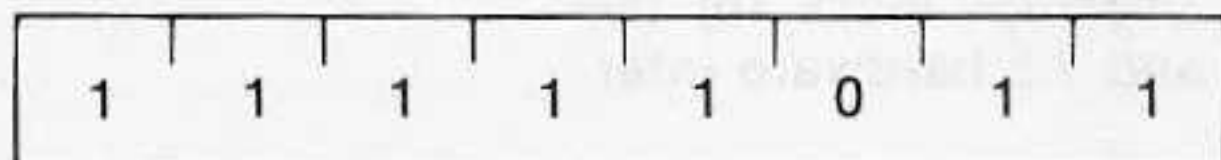


Cycles: 3  
 States: 10  
 Addressing: direct  
 Flags: none



# THE INSTRUCTION SET

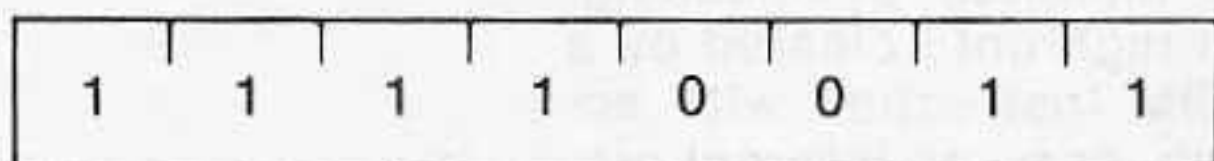
**EI** (Enable interrupts)  
The interrupt system is enabled **following the execution of the next instruction**. Interrupts are not recognized during the EI instruction.



Cycles: 1  
States: 4  
Flags: none

NOTE: Placing an EI instruction on the bus in response to  $\overline{INTA}$  during an  $\overline{INA}$  cycle is prohibited. (8085)

**DI** (Disable interrupts)  
The interrupt system is disabled **immediately following the execution of the DI instruction**. Interrupts are not recognized during the DI instruction.



Cycles: 1  
States: 4  
Flags: none

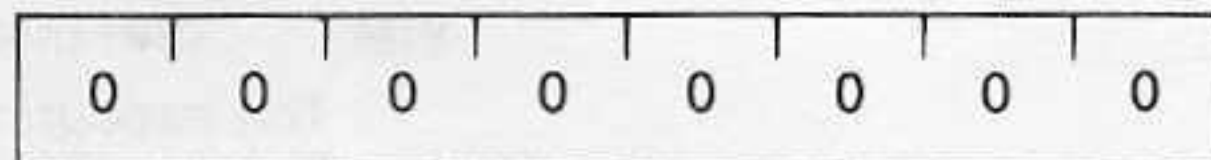
NOTE: Placing a DI instruction on the bus in response to  $\overline{INTA}$  during an  $\overline{INA}$  cycle is prohibited. (8085)

**HLT** (Halt)  
The processor is stopped. The registers and flags are unaffected. (8080) A second ALE is generated during the execution of HLT to strobe out the Halt cycle status information. (8085)



Cycles: 1 + (8085), 1 (8080)  
States: 5 (8085), 7 (8080)  
Flags: none

**NOP** (No op)  
No operation is performed. The registers and flags are unaffected.



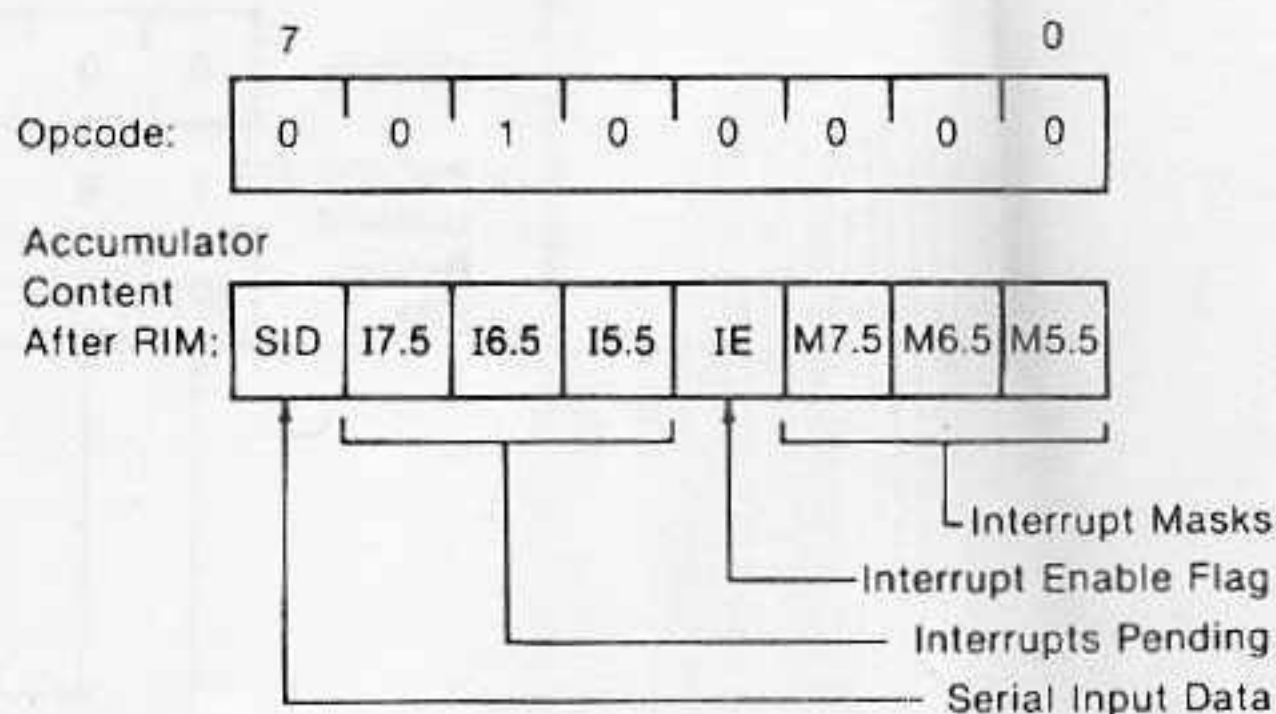
Cycles: 1  
States: 4  
Flags: none

**RIM** (Read Interrupt Masks) (8085 only)

The RIM instruction loads data into the accumulator relating to interrupts and the serial input. This data contains the following information:

- Current interrupt mask status for the RST 5.5, 6.5, and 7.5 hardware interrupts (1 = mask disabled)
- Current interrupt enable flag status (1 = interrupts enabled) except immediately following a TRAP interrupt. (See below.)
- Hardware interrupts pending (i.e., signal received but not yet serviced), on the RST 5.5, 6.5, and 7.5 lines.
- Serial input data.

Immediately following a TRAP interrupt, the RIM instruction must be executed as a part of the service routine if you need to retrieve current interrupt status later. Bit 3 of the accumulator is (in this special case only) loaded with the interrupt enable (IE) flag status that existed prior to the TRAP interrupt. Following an RST 5.5, 6.5, 7.5, or INTR interrupt, the interrupt flag flip-flop reflects the current interrupt enable status. Bit 6 of the accumulator (I7.5) is loaded with the status of the RST 7.5 flip-flop, which is always set (edge-triggered) by an input on the RST 7.5 input line, even when that interrupt has been previously masked. (See SIM Instruction.)



Cycles: 1  
States: 4  
Flags: none



## SIM (Set Interrupt Masks) (8085 only)

The execution of the SIM instruction uses the contents of the accumulator (which must be previously loaded) to perform the following functions:

- Program the interrupt mask for the RST 5.5, 6.5, and 7.5 hardware interrupts.
- Reset the edge-triggered RST 7.5 input latch.
- Load the SOD output latch.

To program the interrupt masks, first set accumulator bit 3 to 1 and set to 1 any bits 0, 1, and 2, which disable interrupts RST 5.5, 6.5, and 7.5, respectively. Then do a SIM instruction. If accumulator bit 3 is 0 when the SIM instruction is executed, the interrupt mask register will not change. If accumulator bit 4 is 1 when the SIM instruction is executed, the RST 7.5 latch is then reset. RST 7.5 is distinguished by the fact that its latch is always set by a rising edge on the RST 7.5 input pin, even if the jump to service routine is inhibited by masking. This latch remains high until cleared by a **RESET IN**, by a SIM Instruction with accumulator bit 4 high, or by an internal processor acknowledge to an RST 7.5 interrupt subsequent to the removal of the mask (by a SIM instruction). The **RESET IN** signal always sets all three RST mask bits.

If accumulator bit 6 is at the 1 level when the SIM instruction is executed, the state of accumulator bit 7 is loaded into the SOD latch and thus becomes available for interface to an external device. The SOD latch is unaffected by the SIM instruction if bit 6 is 0. SOD is always reset by the **RESET IN** signal.

