8085 Instructions

Data Transfer Group

Instructions which are used to transfer the data from a register to another register from memory to register or register to memory come under this group.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Flags** | **Addre-ssing** | **Machine Cycles** | **Example** |
| MOV r1, r2 [r1] ← [r2] | Move the content of the one register to another | 4 | none | Register | 1 | MOV A, B |
| MOV r, M [r]←[[H-L]] | Move the content of memory to register | 7 | none | Register Indirect | 2 | MOV B, M |
| MOV M, r [[H-L]]←[r] | Move the content of register to memory | 7 | none | Register Indirect | 2 | MOV M, C |
| MVI r, data [r] ←data | Move immediate data to register | 7 | None | Immediate Register | 3 | MVI M, 08 |
| LXI rp, data 16 [rp] ←data 16 bits, [rh] ←8 MSBs, [rl] ←8 LSBs of data | Load Register pair immediate | 10 | None | Immediate | 3 | LXI H, 2500H |
| LDA addr [A] ←[addr] | Load Accumulator direct | 13 | None | Direct | 4 | LDA 2400 H |
| STA Addr [addr] ←[A] | Store accumulator direct | 13 | None | Direct | 4 | STA 2000H |
| LHLD addr [L] ←[addr], [H] ← [addr + 1 ] | Load H-L pair direct | 16 | None | Direct | 5 | LHLD 2500H |
| SHLD addr [addr] ←[L], [addr +1] ← [H] | Store H-L pair direct | 16 | None | Direct | 5 | SHLD 2500 H |
| LDAX rp [A] ←[[rp]] | Load accumulator indirect | 7 | None | Register Indirect | 2 | LDAX B |
| STAX rp [[rp]] ←[A] | Store accumulator indirect | 7 | None | Register Indirect | 2 | STAX D |
| XCHG [H-L] ↔[D-E] | Change the contents of H-L with D-E pair | 4 | None | Register | 1 |  |

**Arithmetic Group**

**The instructions of this group perform arithmetic operations such as addition, subtraction, increment or decrement of the content of a register or a memory.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Flags** | **Addre-ssing** | **Machine Cycles** | **Example** |
| **ADD r [A] ←[A]+[r]** | **Add register to accumulator** | **4** | **All** | **Register** | **1** | **ADD K** |
| **ADD M [A] ← [A] + [[H-L]]** | **Add memory to accumulator** | **7** | **All** | **Register indirect** | **2** | **ADD K** |
| **ACC r [A] ← [A] + [r] + [CS]** | **Add register with carry to accumulator** | **4** | **All** | **Register** | **1** | **ACC K** |
| **ADC M [A] ← [A] + [[H-L]] [CS]** | **Add memory with carry to accumulator** | **7** | **All** | **Register indirect** | **2** | **ADC K** |
| **ADI data [A] ← [A] + data** | **Add immediate data to accumulator** | **7** | **All** | **Immediate** | **2** | **ADI 55K** |
| **ACI data [A] ← [A] + data + [CS]** | **Add with carry immediate data to accumulator** | **7** | **All** | **Immediate** | **2** | **ACI 55K** |
| **DAD rp [H-L] ←[H-L] + [rp]** | **Add register paid to H-L pair** | **10** | **CS** | **Register** | **3** | **DAD K** |
| **SUB r [A] ←[A]-[r]** | **Subtract register from accumulator** | **4** | **All** | **Register** | **1** | **SUB K** |
| **SUB M [A] ← [A] - [[H-L]]** | **Subtract memory from accumulator** | **7** | **ALL** | **Register indirect** | **2** | **SUB K** |
| **SBB r [A] ←[A]-[H-L]] - [CS]** | **Subtract memory from accumulator with borrow** | **7** | **All** | **Register indirect** | **2** | **SBB K** |
| **SUI data [A] ←[A]-data** | **Subtract immediate data from accumulator** | **7** | **All** | **Immediate** | **2** | **SUI 55K** |
| **SBI data [A] ←[A]-data-[CS]** | **Subtract immediate data from accumulator with borrow** | **7** | **All** | **Immediate** | **2** | **XCHG** |
| **INR r [r] ←[r]+1** | **Increment register content** | **4** | **All except carry flag** | **Register** | **1** | **INR K** |
| **INR M [[H-L]] ←[[H-L]]+1** | **Increment memory content** | **10** | **All except carry flag** | **Register indirect** | **3** | **INR K** |
| **DCR r [r] ←[r] -1** | **Decrement register content** | **4** | **All except carry flag** | **Register** | **1** | **DCR K** |
| **DCR M [[H-L]] ← [[H-L]]-1** | **Decrement memory content** | **10** | **All except carry flag** | **Register indirect** | **3** | **DCR K** |
| **INX rp [rp] ←[rp]+1** | **Increment memory content** | **6** | **None** | **Register** | **1** | **INX K** |
| **DCX rp [rp] ←[rp]-1** | **Decrement register pair** | **6** | **None** | **Register** | **1** | **DCX K** |
| **DAA** | **Decimal adjust accumulator** | **4** |  |  | **1** | **DAA** |

**Logical Group**

**The instructions in this group perform logical operation such as AND, OR, compare, rotate, etc.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **ANA r [A] ←[A]∧[r]** | **AND register with accumulator** | **4** | **All** | **Register** | **1** |
| **ANA M [A] ←[A]∧[[H-]]** | **AND memory with accumulator** | **4** | **All** | **Register indirect** | **2** |
| **ANI data [A] ← [A] ∧ [data]** | **AND immediate data with accumulator** | **7** | **All** | **Immediate** | **2** |
| **ORA r [A] ←[A]∨[r]** | **OR-register with accumulator** | **4** | **All** | **Register** | **1** |
| **ORA M [A] ←[A]∨[[H-L]]** | **OR-memory with accumulator** | **7** | **All** | **Register indirect** | **2** |
| **ORI data [A] ← [A] ∨ [data]** | **OR -immediate data with accumulator** | **7** | **All** | **Immediate** | **2** |
| **XRA r [A] ← [A]∀[r]** | **XOR register with accumulator** | **4** | **All** | **Register** | **1** |
| **XRA M [A] ← [A] ∀ [[H-L]]** | **XOR memory with accumulator** | **7** | **All** | **Register indirect** | **2** |
| **XRI data [A] ←[A] ∀ [data]** | **XOR immediate data with accumulator** | **7** | **All** | **Immediate** | **2** |
| **CMA [A] ←[A]** | **Complement the accumulator** | **4** | **None** | **Implicit** | **1** |
| **CMC [CS] ←[CS]** | **Complement the carry status** | **4** | **CS** |  | **1** |
| **STC [CS] ← 1** | **Set carry status** | **4** | **CS** |  | **1** |
| **CMP r [A]-[r]** | **Compare register with accumulator** | **4** | **All** | **Register** | **1** |
| **CMP M [A] - [[H-L]]** | **Compare memory with accumulator** | **7** | **All** | **Register indirect** | **2** |
| **CPI data [A] - data** | **Compare immediate data with accumulator** | **7** | **All** | **Immediate** | **2** |
| **RLC [An+1] ←[An], [A0] ←[A7], [CS] ←[A7]** | **Rotate accumulator left** | **4** | **Cs** | **Implicit** | **1** |
| **RRC [A7] ←[A0], [CS] ←[A0], [An] ←[An+1]** | **Rotate accumulator right** |  | **CS** | **Implicit** | **1** |
| **RAL [An+1] ←[An], [CS] ←[A7], [A0] ←[CS]** | **Rotate accumulator left through carry** |  | **CS** | **Implicit** | **1** |
| **RAR [An] ←[An+1], [CS] ←[A0], [A7] ←[CS]** | **Rotate accumulator right through carry** |  | **CS** | **Implicit** | **1** |

**Branch Control Group**

**This group contains the instructions for conditional and unconditional jump, subroutine call and return, and restart.**

**Unconditional Jump**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **JMP addr(label) [PC] ← Label** | **Unconditional jump: jump to the instruction specified by the address** | **10** | **None** | **Immediate** | **3** |

**Conditional Jump**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Machine Cycles** |
| **Jump addr (label) [PC] ← Label** | **Conditional jump: jump to the instruction specified by the address if the specified condition is fulfilled** | **10, if true and 7, if not true** | **3, if true and 2, if not true** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **Status** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **JZ addr (label) [PC] ← address (label)** | **Jump, if the result is zero** | **Jump if Z=1** | **7/10** | **None** | **Immediate** | **2/3** |
| **JNZ addr (label) [PC] ← address (label)** | **Jump if the result is not zero** | **Jump if Z=0** | **7/10** | **None** | **Immediate** | **2/3** |
| **JC addr (label) [PC] ← address (label)** | **Jump if there is a carry** | **Jump if CS =1** | **7/10** | **None** | **Immediate** | **2/3** |
| **JNC addr (label) [PC] ← address (label)** | **Jump if there is no carry** | **Jump if CS =0** | **7/10** | **None** | **Immediate** | **2/3** |
| **JP addr (label) [PC] ← address (label)** | **Jump if result is plus** | **Jump if S=0** | **7/10** | **None** | **Immediate** | **2/3** |
| **JM addr (label) [PC] ← address (label)** | **Jump if result is minus** | **Jump if S=1** | **7/10** | **None** | **Immediate** | **2/3** |
| **JPE addr (label) [PC] ← address (label)** | **Jump if even parity** | **The parity status P =1** | **7/10** | **None** | **Immediate** | **2/3** |
| **JPO addr (label) [PC] ← address (label)** | **Jump if odd parity** | **The parity status P =0** | **7/10** | **None** | **Immediate** | **2/3** |

**Unconditional CALL**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **CALL addr (label) [SP]-1] ← [PCH] ,[[SP-2] ← [PCL], [SP] ← [SP]-2, [PC] ← addr(label)** | **Unconditional CALL: Call the subroutine identified by the address** | **18** | **None** | **Immediate /register** | **5** |

**Conditional CALL**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Machine Cycles** |
| **CALL addr (label) [SP]-1] ← [PCH] , [[SP-2] ← [PCL], [PC] ← addr (label), [SP] ← [SP]-2** | **Unconditional CALL: Call the subroutine identified by the address if the specified condition is fulfilled** | **18, if true and 9, if not true** | **5, if true and 2, if not true** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **Status** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **CC addr(label)** | **Call subroutine if carry status CS=1** | **CS =1** | **9/18** | **None** | **Immediate /register** | **2/5** |
| **CNC addr (label)** | **Call subroutine if carry status CS=0** | **CS =0** | **9/18** | **None** | **Immediate /register** | **2/5** |
| **CZ addr (label)** | **Call Subroutine if the result is zero** | **Zero status Z=1** | **9/18** | **None** | **Immediate /register** | **2/5** |
| **CNZ addr (label)** | **Call Subroutine if the result is not zero** | **Zero status Z=0** | **9/18** | **None** | **Immediate /register** | **2/5** |
| **CP addr (label)** | **Call Subroutine if the result is plus** | **Sign status S=0** | **9/18** | **None** | **Immediate /register** | **2/5** |
| **CM addr (label)** | **Call Subroutine if the result is minus** | **Sign status S= 1** | **9/18** | **None** | **Immediate /register** | **2/5** |
| **CPE addr(label)** | **Call subroutine if even parity** | **Parity Status P=1** | **9/18** | **None** | **Immediate /register** | **2/5** |
| **CPO addr(label)** | **Call subroutine if odd parity** | **Parity Status P= 0** | **9/18** | **None** | **Immediate /register** | **2/5** |

**Unconditional Return**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **RET [PCL] ← [[SP]], [PCH] ← [[SP] + 1], [SP] ← [SP] + 2** | **Unconditional RET: Return from subroutine** | **10** | **None** | **Indirect** | **3** |

**Conditional Return**

|  |  |  |  |
| --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Machine Cycles** |
| **RET [PCL] ← [[SP]], [PCH] ← [[SP] + 1], [SP] ← [SP] + 2** | **Conditional RET: Return from subroutine** | **12, if true and 6, if not true** | **3, if true and 1, if not true** |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **Status** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **RC** | **Return from subroutine if carry status is zero.** | **CS =1** | **6/12** | **None** | **Register indirect** | **1/3** |
| **RNC** | **Return from subroutine if carry status is not zero.** | **CS = 0** | **6/12** | **None** | **Register indirect** | **1/3** |
| **RZ** | **Return from subroutine if result is zero.** | **Zero status Z=1** | **6/12** | **None** | **Register indirect** | **1/3** |
| **RNZ** | **Return from subroutine if result is not zero.** | **Zero status Z= 0** | **6/12** | **None** | **Register indirect** | **1/3** |
| **RP** | **Return from subroutine if result is not plus.** | **Sign Status S= 0** | **6/12** | **None** | **Register indirect** | **1/3** |
| **RM** | **Return from subroutine if result is not minus.** | **Sign Status S= 0** | **6/12** | **None** | **Register indirect** | **1/3** |
| **RPE** | **Return from subroutine if even parity.** | **Parity Status P= 1** | **6/12** | **None** | **Register indirect** | **1/3** |
| **RPO** | **Return from subroutine if odd parity.** | **Parity Status P= 1** | **6/12** | **None** | **Register indirect** | **1/3** |

**Restart**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **RST [[SP]-1] ← [PCH], [[SP]-2] ← [PCL], [SP] ← [SP] - 2, [PC] ← 8 times n** | **Restart is a one word CALL instruction.** | **12** | **None** | **Register Indirect** | **3** |

**The restart instructions and locations are as follows:**

|  |  |  |
| --- | --- | --- |
| **Instruction** | **Opcode** | **Restart Locations** |
| **RST 0** | **C7** | **0000** |
| **RST 1** | **CF** | **0008** |
| **RST 2** | **D7** | **0010** |
| **RST 3** | **DF** | **0018** |
| **RST 4** | **E7** | **0020** |
| **RST 5** | **EF** | **0028** |
| **RST 6** | **F7** | **0030** |
| **RST 7** | **FF** | **0038** |

**PCHL**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **PCHL [PC] ← [H-L], [PCH] ←[H], [PCL] ←[L]** | **Jump address specified by H-L pair** | **6** | **None** | **Register** | **1** |

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

**This group contains the instructions for input/output ports, stack and machine control.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Instruction Set** | **Explanation** | **States** | **Flags** | **Addressing** | **Machine Cycles** |
| **IN port - address [A] ← [Port]** | **Input to accumulator from I/O port** | **10** | **None** | **Direct** | **3** |
| **OUT port-address [Port] ← [A]** | **Output from accumulator to I/O port** | **10** | **None** | **Direct** | **3** |
| **PUSH rp [[SP] - 1] ← [rh], [[SP] - 2] ← [rh], [SP] ← [SP] - 2** | **Push the content of register pair to stack** | **12** | **None** | **Register(source)/register Indirect(destination)** | **3** |
| **PUSH PSW [SP]-1] ← [A], [[SP] -2] ← PSW, [SP] ← [SP] - 2** | **Push processor word** | **12** | **None** | **Register(source)/register Indirect(destination)** | **3** |
| **POP rp [rl] ← [ [ SP ] ], [rh] ← [[SP]+1], [SP] ← [SP] + 2** | **Pop the content of register pair, which was saved, from the stack** | **10** | **None** | **Register(source)/register Indirect(destination)** | **3** |
| **HLT** | **Halt** | **5** | **None** |  | **1** |
| **XTHL [L] ↔ [[SP]], [H] ↔ [[SP] + 1]** | **Exchange top stack with H-L** | **16** | **None** | **Register indirect** | **5** |
| **SPHL [H-L] → [SP]** | **Moves the contents of H-L pair to stack pointer** | **6** | **None** | **Register** | **1** |
| **EI** | **Enable Interrupts** | **4** | **None** |  | **1** |
| **SIM** | **Set Interrupts Masks** | **4** | **None** |  | **1** |
| **RIM** | **Read Interrupts Masks** | **4** | **None** |  | **1** |
| **NOP** | **No Operation** | **4** | **None** |  | **1** |