Instruções do microprocessador da INTEL 8085

Nomenclatura:

	Legenda
pr	Par de registos: HL, BC, DE, SP, PC
reg	Registo: A, B, C, D, E, H, L
M	Posição de memória
addr	Endereço de 16 bits de uma posição de memória
x	O bit do registo de flags é afetado
byte	Constante, ou expressão lógica/aritmética que representa um dado de 8 bits
double	Constante, ou expressão lógica/aritmética que representa um dado de 16 bits
[ ]	Conteúdo do que se encontra dentro de parênteses retos
[[ ]]	Conteúdo do conteúdo do que se encontra dentro de parênteses retos
CS	Flag de carry
label	Endereço de uma posição de memória
port	Endereço de um dispositivo I/O

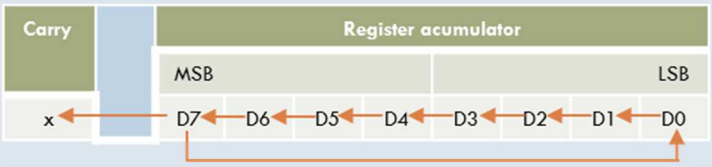
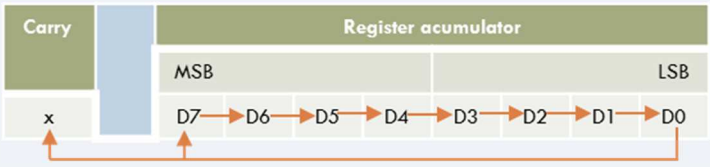
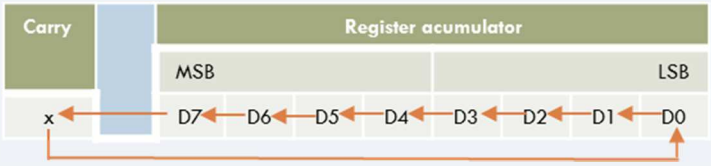
Grupo de transferência de dados

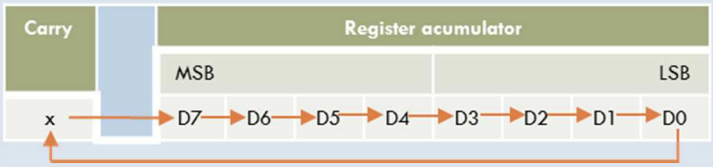
Instrução	Operandos	Status do registo de flags					Operação realizada
		CS	AC	Z	S	P	
LDAX	pr						[A] ← [[pr]] Load A using implied addressing by BC (pr=B) or DE (pr=D)
STAX	pr						[[pr]] ← [A] Store A using implied addressing by BC (pr=B) or DE (pr=D)
MOV	r,M						[r] ← [[HL]] Load any register using implied addressing by HL
MOV	M,r						[[HL]] ← [r] Store any register using implied addressing by HL
LDA	addr						[A] ← [addr] Load A using direct addressing
STA	addr						[addr] ← [A] Store A using direct addressing
LHLD	addr						[L] ← [addr] and [H] ← [addr+1] Load H and L registers using direct addressing
SHLD	addr						[addr] ← [L] and [addr+1] ← [H] Store H and L registers using direct addressing
MOV	r,r						[r] ← [r] Move any register to any register
XCHG							[D] ← → [H] and [E] ← → [L] Exchange DE with HL
SPHL							[HL] ← [SP] Move HL to SP
LXI	pr,double						[pr] ← double Load 16 bits immediate data into BC (pr=B), DE (pr=D), HL (pr=H), SP (pr=SP)
MVI	M,byte						[[HL]] ← byte Load 8 bit immediate data into memory location with implied addressing by HL
MVI	r,byte						[r] ← byte Load 8 bit immediate data into any register

Grupo aritmético, lógico e de rotação

Instrução	Operandos	Status do registo de flags					Operação realizada
		CS	AC	Z	S	P	
ADD	M	x	x	x	x	x	[A] ← [A] + [[HL]] Add register A with implied addressing by HL and store the result in register A
ADC	M	x	x	x	x	x	[A] ← [A] + [[HL]] + [CS] Add register A with carry with implied addressing by HL and store the result in register A
SUB	M	x	x	x	x	x	[A] ← [A] – [[HL]] Subtract register A with implied addressing by HL and store the result in register A
SBB	M	x	x	x	x	x	[A] ← [A] - [[HL]] - [CS] Subtract register A with carry with implied addressing by HL and store the result in register A
ANA	M	0	1	x	x	x	[A] ← [A] AND [[HL]] AND between register A with implied addressing by HL and store the result in register A
XRA	M	0	0	x	x	x	[A] ← [A] XOR [[HL]] Exclusive-OR between register A with implied addressing by HL and store the result in register A
ORA	M	0	0	x	x	x	[A] ← [A] OR [[HL]] OR between register A with implied addressing by HL and store the result in register A
CMP	M	x	x	x	x	x	[A] – [[HL]] Compare register A with implied addressing by HL If register A < [[HL]] than the carry flag is set (1) If register A = [[HL]] than the zero flag is set (1) If register A > [[HL]] than the carry and zero flags are reset (0)
INR	M	x	x	x	x	x	[[HL]] ← [[HL]] + 1 Increment memory
DCR	M	x	x	x	x	x	[[HL]] ← [[HL]] -1 Decrement memory
ADI	byte	x	x	x	x	x	[A] ← [A] + byte Add register A with 8 bit immediate data and store the result in register A
ACI	byte	x	x	x	x	x	[A] ← [A] + byte + [CS] Add register A with 8 bit immediate data with carry and store the result in register A
SUI	byte	x	x	x	x	x	[A] ← [A] – byte Subtract register A with 8 bit immediate data and store the result in register A

SBI	byte	x	x	x	x	x	[A] ← [A] - byte - [CS] Subtract register A with 8 bit immediate data with carry and store the result in register A
ANI	byte	0	1	x	x	x	[A] ← [A] AND byte AND between register A with 8 bit immediate data and store the result in register A
XRI	byte	0	0	x	x	x	[A] ← [A] XOR byte Exclusive-OR between register A with 8 bit immediate data and store the result in register A
ORI	byte	0	0	x	x	x	[A] ← [A] OR byte OR between register A with 8 bit immediate data and store the result in register A
CPI	byte	x	x	x	x	x	[A] – byte Compare register A with 8 bit immediate data If register A < byte than the carry flag is set (1) If register A = byte than the zero flag is set (1) If register A > byte than the carry and zero flags are reset (0)
ADD	r	x	x	x	x	x	[A] ← [A] + [r] Add register A with any register and store the result in register A
ADC	r	x	x	x	x	x	[A] ← [A] + [r] + [CS] Add register A with any register with carry and store the result in register A
SUB	r	x	x	x	x	x	[A] ← [A] – [r] Subtract register A with any register and store the result in register A
SBB	r	x	x	x	x	x	[A] ← [A] – [r] – [CS] Subtract register A with any register with carry and store the result in register A
ANA	r	0	1	x	x	x	[A] ← [A] AND [r] AND between register A with any register and store the result in register A
XRA	r	0	0	x	x	x	[A] ← [A] XOR [r] Exclusive-OR between register A with any register and store the result in register A
ORA	r	0	0	x	x	x	[A] ← [A] OR [r] OR between register A with any register and store the result in register A
CMP	r	x	x	x	x	x	[A] – [r] Compare register A with any register If register A < r than the carry flag is set (1) If register A = r than the zero flag is set (1) If register A > r than the carry and zero flags are reset (0)
INR	r		x	x	x	x	[r] ← [r] + 1 Increment any register

DCR	r		x	x	x	x	[r] ← [r] – 1 Decrement any register
CMA							[A] ← $[\bar{A}]$ Complement register A
DAA		x	x	x	x	x	The contents of the accumulator are changed from a binary value to two 4-bit binary coded decimal (BCD) digits. This is the only instruction that uses the auxiliary flag (AC) to perform the binary to BCD conversion, and the conversion procedure is described below. If the value of the low-order 4-bits in the accumulator is greater than 9 or if AC flag is set (1), the instruction adds 6 to the low-order four bits. If the value of the high-order 4-bits in the accumulator is greater than 9 or if the Carry flag (CS) is set (1), the instruction adds 6 to the high-order four bits.
RLC		x					Each binary bit of the register accumulator is rotated left by one position. Bit D7 is placed in the position of D0 as well as in the Carry flag. The carry flag (CS) is modified according to bit D7. 
RRC		x					Each binary bit of the register acumulator is rotated right by one position. Bit D0 is placed in the position of D7 as well as in the Carry flag. The carry flag (CS) is modified according to bit D0. 
RAL		X					Each binary bit of the register acumulator is rotated left by one position trought the carry flag. Bit D_7 is placed in the carry flag, and the carry flag is placed in the least significant position D_0. The carry flag (CS) is modified according to bit D7. 
RAR		x					Each binary bit of the register acumulator is rotated right by one position trought the carry flag. Bit D_0 is placed in the carry flag, and the carry flag is placed in the least significant position D_7. The carry flag (CS) is modified according to bit D0.

							
DAD	pr	x					[HL] ← [HL] + [pr] Add HL to a register pair BC (pr=B), DE (pr=D), HL (pr=H), SP (pr=SP) and store the result in HL
INX	pr						[pr] ← [pr] + 1 Increment register pair BC (pr=B), DE (pr=D), HL (pr=H), SP (pr=SP) and store the result in a register pair BC (pr=B), DE (pr=D), HL (pr=H), SP (pr=SP)
DCX	pr						[pr] ← [pr] – 1 Decrement register pair BC (pr=B), DE (pr=D), HL (pr=H), SP (pr=SP) and store the result in a register pair BC (pr=B), DE (pr=D), HL (pr=H), SP (pr=SP)

Grupo de controlo e de salto

Instrução	**Operandos**	**Status do registo de flags**					**Operação realizada**
		CS	AC	Z	S	P	
JMP	label						[PC] ← label Jump to instruction at address label
PCHL							[PC] ← [HL] Jump to instruction at address contained in HL
CALL	label						[[SP]] ← [PC] , [PC] ← label, [SP] ← [SP] – 2 Jump to subrotine starting at address label
CC	label						[[SP]] ← [PC] , [PC] ← label, [SP] ← [SP] – 2 Jump to subrotine starting at address label if the carry flag (CS) equal to 1
CNC	label						[[SP]] ← [PC] , [PC] ← label, [SP] ← [SP] – 2 Jump to subrotine starting at address label if the carry flag (CS) equal to 0
CZ	label						[[SP]] ← [PC] , [PC] ← label, [SP] ← [SP] – 2 Jump to subrotine starting at address label if the zero flag (Z) equal to 1
CNZ	label						[[SP]] ← [PC] , [PC] ← label, [SP] ← [SP] – 2 Jump to subrotine starting at address label if the zero flag (Z) equal to 0
CP	label						[[SP]] ← [PC] , [PC] ← label, [SP] ← [SP] – 2 Jump to subrotine starting at address label if the sign flag (S) equal to 0

CM	label						[[SP]] ← [PC] , [PC] ← label, [SP] ← [SP] – 2 Jump to subrotine starting at address label if the sign flag (S) equal to 1
CPE	label						[[SP]] ← [PC] , [PC] ← label, [SP] ← [SP] – 2 Jump to subrotine starting at address label if the parity flag (P) equal to 1
CPO	label						[[SP]] ← [PC] , [PC] ← label, [SP] ← [SP] – 2 Jump to subrotine starting at address label if the parity flag (P) equal to 0
RET							[PC] ← [[SP]] , [SP] ← [SP] + 2 Return from subrotine
RC							[PC] ← [[SP]] , [SP] ← [SP] + 2 Return from subrotine if the carry flag (CS) equal to 1
RNC							[PC] ← [[SP]] , [SP] ← [SP] + 2 Return from subrotine if the carry flag (CS) equal to 0
RZ							[PC] ← [[SP]] , [SP] ← [SP] + 2 Return from subrotine if the zero flag (Z) equal to 1
RNZ							[PC] ← [[SP]] , [SP] ← [SP] + 2 Return from subrotine if the zero flag (Z) equal to 0
RM							[PC] ← [[SP]] , [SP] ← [SP] + 2 Return from subrotine if the sign flag (S) equal to 0
RP							[PC] ← [[SP]] , [SP] ← [SP] + 2 Return from subrotine if the sign flag (S) equal to 1
RPE							[PC] ← [[SP]] , [SP] ← [SP] + 2 Return from subrotine if the parity flag (P) equal to 1
RPO							[PC] ← [[SP]] , [SP] ← [SP] + 2 Return from subrotine if the parity flag (P) equal to 0
JC	label						[PC] ← label Jump to instruction at address label if the carry flag (CS) equal to 1
JNC	label						[PC] ← label Jump to instruction at address label if the carry flag (CS) equal to 0
JZ	label						[PC] ← label Jump to instruction at address label if the zero flag (Z) equal to 1
JNZ	label						[PC] ← label Jump to instruction at address label if the zero flag (Z) equal to 0
JP	label						[PC] ← label Jump to instruction at address label if the sign flag (S) equal to 0

JM	label						[PC] ← label Jump to instruction at address label if the sign flag (S) equal to 1
JPE	label						[PC] ← label Jump to instruction at address label if the parity flag (P) equal to 1
JPO	label						[PC] ← label Jump to instruction at address label if the parity flag (P) equal to 0
RST	n						The RST instruction is equivalent to a 1-byte call instruction to one of eight memory locations depending upon the number. The instructions are generally used in conjunction with interrupts and inserted using external hardware. However these can be used as software instructions in a program to transfer program execution to one of the eight locations. The addresses are: Instruction Restart Address RST 0 0000H RST 1 0008H RST 2 0010H RST 3 0018H RST 4 0020H RST 5 0028H RST 6 0030H RST 7 0038H

Grupo de controlo do CPU, I/O e da Pilha

Instrução	**Operandos**	**Status do registo de flags**					**Operação realizada**
		CS	AC	Z	S	P	
IN	port						[A] ← [port] Input to register acumulator (A) from I/O port
OUT	port						[port] ← [A] Ouput from register acumulator (A) to I/O port
PUSH	pr						[[SP]] ← [pr] , [SP] ← [SP] – 2 Push register pair BC (pr=B), DE (pr =D), H (pr=HL), PSW (pr=PSW) contentes onto stack
POP	pr						[pr] ← [[SP]] , [SP] ← [SP] + 2 Pop stack into register pair BC (pr=B), DE (pr =D), H (pr=HL), PSW (pr=PSW)
XTHL							[HL] ← [[SP]] Exchange HL with top of stack

EI							Enable interrupts following execution of next instruction
DI							Disable interrupts
SIM							Set interrupt mask
RIM							Read interrupt mask
NOP							[PC] ← [PC] +1 No operation but program counter (PC) is incremented
HLT							HALT Stop CPU operation