**8-bit Load**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| LD r, r | r = r | ● | ● | ● | ● | ● | ● | 1 | 4 | r = A,B,C,D,E,H y L |
| LD p,p’ | p = p’ | ● | ● | ● | ● | ● | ● | 2 | 8 | p = A,B,C,D,E,IXh y IXl |
| LD q,q’ | q = q’ | ● | ● | ● | ● | ● | ● | 2 | 8 | q = A,B,C,D,E,IYh y IYl |
| LD r,n | r = n | ● | ● | ● | ● | ● | ● | 2 | 7 | n = número 8 bits |
| LD r,[HL] | r = [HL] | ● | ● | ● | ● | ● | ● | 1 | 7 |  |
| LD [HL],r | [HL] = r | ● | ● | ● | ● | ● | ● | 1 | 7 |  |
| LD A,[BC] | A = [BC] | ● | ● | ● | ● | ● | ● | 1 | 7 |  |
| LD A,[DE] | A = [DE] | ● | ● | ● | ● | ● | ● | 1 | 7 |  |
| LD [BC],A | [BC] = A | ● | ● | ● | ● | ● | ● | 1 | 7 |  |
| LD [DE],A | [DE] = A | ● | ● | ● | ● | ● | ● | 1 | 7 |  |
| LD p,n | p = n | ● | ● | ● | ● | ● | ● | 3 | 11 |  |
| LD q,n | q = n | ● | ● | ● | ● | ● | ● | 3 | 11 |  |
| LD [HL],n | [HL] = n | ● | ● | ● | ● | ● | ● | 2 | 11 |  |
| LD A,[nn] | A = [nn] | ● | ● | ● | ● | ● | ● | 3 | 13 |  |
| LD [nn],A | [nn] = A | ● | ● | ● | ● | ● | ● | 3 | 13 |  |
| LD r,[IX+d] | r = [IX+d] | ● | ● | ● | ● | ● | ● | 3 | 19 |  |
| LD r,[IY+d] | r =[IY+d] | ● | ● | ● | ● | ● | ● | 3 | 19 |  |
| LD [IX+d],r | [IX+d] = r | ● | ● | ● | ● | ● | ● | 3 | 19 |  |
| LD [YX+d],r | [YX+d] = r | ● | ● | ● | ● | ● | ● | 3 | 19 |  |
| LD [IX+d],n | [IX+d] = n | ● | ● | ● | ● | ● | ● | 4 | 19 |  |
| LD [IY+d],n | [IY+d] = n | ● | ● | ● | ● | ● | ● | 4 | 19 | ● No toca nada |
| LD A,I | A = I | ↕ | ↕ | 0 | ? | 0 | ↕ | 2 | 9 | 0 Siempre a 0 |
| LD A,R | A = R | ↕ | ↕ | 0 | ? | 0 | ↕ | 2 | 9 | 1 Siempre a 1 |
| LD I,A | I = A | ● | ● | ● | ● | ● | ● | 2 | 9 | ↕ Cambio normal |
| LD R,A | R = A | ● | ● | ● | ● | ● | ● | 2 | 9 | ? Ni idea, no importa |

**16-bit Load**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| LD dd, nn | dd = nn | ● | ● | ● | ● | ● | ● | 3 | 10 | dd = BC,DE,HL y SP |
| LD IX, nn | IX = nn | ● | ● | ● | ● | ● | ● | 4 | 14 | nn = número 16 bits |
| LD IY, nn | IY = nn | ● | ● | ● | ● | ● | ● | 4 | 14 |  |
| LD HL, [nn] | H=[nn+1] L=nn | ● | ● | ● | ● | ● | ● | 3 | 16 |  |
| LD dd, [nn] | ddh=[nn+1] ddl=[nn] | ● | ● | ● | ● | ● | ● | 4 | 20 |  |
| LD IX, [nn] | IXh=[nn+1] IXl=[nn] | ● | ● | ● | ● | ● | ● | 4 | 20 |  |
| LD IY, [nn] | IYh=[nn+1] IYl=[nn] | ● | ● | ● | ● | ● | ● | 4 | 20 |  |
| LD [nn], HL | [nn+1]=H [nn]=L | ● | ● | ● | ● | ● | ● | 3 | 16 |  |
| LD [nn], dd | [nn+1]=ddh [nn]=ddl | ● | ● | ● | ● | ● | ● | 4 | 20 |  |
| LD [nn], IX | [nn+1]=IXh [nn]=IXl | ● | ● | ● | ● | ● | ● | 4 | 20 | ● No toca nada |
| LD [nn], IY | [nn+1]=IYh [nn]=IYl | ● | ● | ● | ● | ● | ● | 4 | 20 | 0 Siempre a 0 |
| LD SP, HL | SP = HL | ● | ● | ● | ● | ● | ● | 1 | 6 | 1 Siempre a 1 |
| LD SP, IX | SP = IX | ● | ● | ● | ● | ● | ● | 2 | 10 | ↕ Cambio normal |
| LD SP, IY | SP = IY | ● | ● | ● | ● | ● | ● | 2 | 10 | ? Ni idea, no importa |

**Push / Pop**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| PUSH qq | [SP-2]=qql [SP-1]=qqh | ● | ● | ● | ● | ● | ● | 1 | 11 | qq = BC,DE,HL y AF |
| PUSH IX | [SP-2]=IXl [SP-1]=IXh | ● | ● | ● | ● | ● | ● | 2 | 15 |  |
| PUSH IY | [SP-2]=IYl [SP-1]=IYh | ● | ● | ● | ● | ● | ● | 2 | 15 |  |
| POP qq | qqh=[SP+1] qql=[SP] | ● | ● | ● | ● | ● | ● | 1 | 10 |  |
| POP IX | IXh=[SP+1] IXl=[SP] | ● | ● | ● | ● | ● | ● | 2 | 14 |  |
| POP IY | IYh=[SP+1] IYl=[SP] | ● | ● | ● | ● | ● | ● | 2 | 14 |  |

**EXchange, Block Transfer & Search**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| EX DE, HL | DE <-> HL cambia | ● | ● | ● | ● | ● | ● | 1 | 4 |  |
| EX AF, AF’ | AF <-> AF’ cambia | ● | ● | ● | ● | ● | ● | 1 | 4 |  |
| EXX | BC,DE,HL<->BC’,DE’HL’ | ● | ● | ● | ● | ● | ● | 1 | 4 |  |
| EX [SP],HL | H<->[SP+1] L<->[SP] | ● | ● | ● | ● | ● | ● | 1 | 19 |  |
| EX [SP],IX | IXh<->[SP+1] IXl<->[SP] | ● | ● | ● | ● | ● | ● | 2 | 23 |  |
| EX [SP],IY | IYh<->[SP+1] IYl<->[SP] | ● | ● | ● | ● | ● | ● | 2 | 23 |  |
| LDI | [DE]=[HL] copia  DE+=1; HL+=1; BC-=1 | ● | ● | 0 | ♠ | 0 | ● | 2 | 16 | ♠ P/V=0 si BC-1==0 |
| LDIR | [DE]=[HL]  DE+=1; HL+=1; BC-=1  hasta BC==0 | ● | ● | 0 | ♣ | 0 | ● | 2 | 21  16 | 21T si BC!=0  16T si BC==0  ♣ P/V=0 al acabar |
| LDD | [DE]=[HL] copia  DE-=1; HL-=1; BC-=1 | ● | ● | 0 | ♠ | 0 | ● | 2 | 16 |  |
| LDDR | [DE]=[HL] copia  DE-=1; HL-=1; BC-=1  hasta BC==0 | ● | ● | 0 | ♣ | 0 | ● | 2 | 21  16 | 21T si BC!=0  16T si BC==0 |
| CPI | (A-[HL]) busca  HL+=1; BC-=1 | ? | ♥ | ? | ♠ | 1 | ● | 2 | 16 | ♥ Z=1 si A==[HL] |
| CPDIR | (A-[HL]) busca  HL+=1; BC-=1  hasta (A==[HL])||(BC==0) | ? | ♥ | ? | ♠ | 1 | ● | 2 | 2116 | 21T si (BC!=0)&&(A!=[HL])  16T si (BC==0)||(A==[HL]) |
| CPD | (A-[HL]) busca  HL-=1; BC-=1 | ? | ♥ | ? | ♠ | 1 | ● | 2 | 16 |  |
| CPDR | (A-[HL]) busca  HL-=1; BC-=1  hasta (A==[HL])||(BC==0) | ? | ♥ | ? | ♠ | 1 | ● | 2 | 21  16 | 21T si (BC!=0)&&(A!=[HL])  16T si (BC==0)||(A==[HL]) |

**8-Bit Logical**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| AND r | A&=r | ↕ | ↕ | 1 | P | 0 | 0 | 1 | 4 | r = A,B,C,D,E,H y L |
| AND p | A&=p | ↕ | ↕ | 1 | P | 0 | 0 | 2 | 8 | p = A,B,C,D,E,IXh y IXl |
| AND q | A&=q | ↕ | ↕ | 1 | P | 0 | 0 | 2 | 8 | q = A,B,C,D,E,IYh y IYl |
| AND n | A&=n | ↕ | ↕ | 1 | P | 0 | 0 | 2 | 7 | n = numero 8 bits |
| AND [HL] | A&=[HL] | ↕ | ↕ | 1 | P | 0 | 0 | 1 | 7 |  |
| AND [IX+d] | A&=[IX+d] | ↕ | ↕ | 1 | P | 0 | 0 | 3 | 19 |  |
| AND [IY+d] | A&=[IY+d] | ↕ | ↕ | 1 | P | 0 | 0 | 3 | 19 |  |
| OR r | A|=r | ↕ | ↕ | 0 | P | 0 | 0 | 1 | 4 |  |
| OR p | A|=p | ↕ | ↕ | 0 | P | 0 | 0 | 2 | 8 |  |
| OR q | A|=q | ↕ | ↕ | 0 | P | 0 | 0 | 2 | 8 |  |
| OR n | A|=n | ↕ | ↕ | 0 | P | 0 | 0 | 2 | 7 |  |
| OR [HL] | A|=[HL] | ↕ | ↕ | 0 | P | 0 | 0 | 1 | 7 |  |
| OR [IX+d] | A|=[IX+d] | ↕ | ↕ | 0 | P | 0 | 0 | 3 | 19 |  |
| OR [IY+d] | A|=[IY+d] | ↕ | ↕ | 0 | P | 0 | 0 | 3 | 19 |  |
| XOR r | A^=r | ↕ | ↕ | 0 | P | 0 | 0 | 1 | 4 |  |
| XOR p | A^=p | ↕ | ↕ | 0 | P | 0 | 0 | 2 | 8 |  |
| XOR q | A^=q | ↕ | ↕ | 0 | P | 0 | 0 | 2 | 8 | ● No toca nada |
| XOR n | A^=n | ↕ | ↕ | 0 | P | 0 | 0 | 2 | 7 | 0 Siempre a 0 |
| XOR [HL] | A^=[HL] | ↕ | ↕ | 0 | P | 0 | 0 | 1 | 7 | 1 Siempre a 1 |
| XOR [IX+d] | A^=[IX+d] | ↕ | ↕ | 0 | P | 0 | 0 | 3 | 19 | ↕ Cambio normal |
| XOR [IY+d] | A^=[IY+d] | ↕ | ↕ | 0 | P | 0 | 0 | 3 | 19 | ? Ni idea, no importa |
| CPL | !A NOT(A) | ● | ● | 1 | ● | 1 | ● | 1 | 4 |  |
| NEG | ~A A=-A | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 8 |  |
| CCF | !CF NOT(CF) | ● | ● | ♠ | ● | 0 | ↕ | 1 | 4 | ♠ HF is like CF before |
| SCF | CF=1 | ● | ● | 0 | ● | 0 | 1 | 1 | 4 |  |
| DAA | *mejor no usar* | ↕ | ↕ | ↕ | P | ● | ↕ | 1 | 4 | Decimal adjust accumulator |

**8-Bit Add, Substract, Inc & Dec**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| ADD A, r | A+=r | ↕ | ↕ | ↕ | V | 0 | ↕ | 1 | 4 | r = A,B,C,D,E,H y L |
| ADD A, p | A+=p | ↕ | ↕ | ↕ | V | 0 | ↕ | 2 | 8 | p = A,B,C,D,E,IXh y IXl |
| ADD A, q | A+=q | ↕ | ↕ | ↕ | V | 0 | ↕ | 2 | 8 | q = A,B,C,D,E,IYh y IYl |
| ADD A, n | A+=n | ↕ | ↕ | ↕ | V | 0 | ↕ | 2 | 7 | n = numero 8 bits |
| ADD A, [HL] | A+=[HL] | ↕ | ↕ | ↕ | V | 0 | ↕ | 1 | 7 |  |
| ADD A,[IX+d] | A+=[IX+d] | ↕ | ↕ | ↕ | V | 0 | ↕ | 3 | 19 |  |
| ADD A,[IY+d] | A+=[IY+d] | ↕ | ↕ | ↕ | V | 0 | ↕ | 3 | 19 |  |
| ADC A, r | A+=r+CF | ↕ | ↕ | ↕ | V | 0 | ↕ | 1 | 4 |  |
| ADC A, p | A+=p+CF | ↕ | ↕ | ↕ | V | 0 | ↕ | 2 | 8 |  |
| ADC A, q | A+=q+CF | ↕ | ↕ | ↕ | V | 0 | ↕ | 2 | 8 |  |
| ADC A, n | A+=n+CF | ↕ | ↕ | ↕ | V | 0 | ↕ | 2 | 7 |  |
| ADC A, [HL] | A+=[HL]+CF | ↕ | ↕ | ↕ | V | 0 | ↕ | 1 | 7 |  |
| ADC A,[IX+d] | A+=[IX+d]+CF | ↕ | ↕ | ↕ | V | 0 | ↕ | 3 | 19 |  |
| ADC A,[IY+d] | A+=[IY+d]+CF | ↕ | ↕ | ↕ | V | 0 | ↕ | 3 | 19 |  |
| SUB A, r | A-=r | ↕ | ↕ | ↕ | V | 1 | ↕ | 1 | 4 |  |
| SUB A, p | A-=p | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 8 |  |
| SUB A, q | A-=q | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 8 |  |
| SUB A, n | A-=n | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 7 |  |
| SUB A, [HL] | A-=[HL] | ↕ | ↕ | ↕ | V | 1 | ↕ | 1 | 7 |  |
| SUB A,[IX+d] | A-=[IX+d] | ↕ | ↕ | ↕ | V | 1 | ↕ | 3 | 19 |  |
| SUB A,[IY+d] | A-=[IY+d] | ↕ | ↕ | ↕ | V | 1 | ↕ | 3 | 19 |  |
| SBC A, r | A-=r-CF | ↕ | ↕ | ↕ | V | 1 | ↕ | 1 | 4 |  |
| SBC A, p | A-=p-CF | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 8 |  |
| SBC A, q | A-=q-CF | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 8 |  |
| SBC A, n | A-=n-CF | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 7 |  |
| SBC A, [HL] | A-=[HL]-CF | ↕ | ↕ | ↕ | V | 1 | ↕ | 1 | 7 |  |
| SBC A,[IX+d] | A-=[IX+d]-CF | ↕ | ↕ | ↕ | V | 1 | ↕ | 3 | 19 |  |
| SBC A,[IY+d] | A-=[IY+d]-CF | ↕ | ↕ | ↕ | V | 1 | ↕ | 3 | 19 |  |
| INC r | ++r | ↕ | ↕ | ↕ | V | 0 | ● | 1 | 4 | r = A,B,C,D,E,H y L |
| INC p | ++p | ↕ | ↕ | ↕ | V | 0 | ● | 2 | 8 | p = A,B,C,D,E,IXh y IXl |
| INC q | ++q | ↕ | ↕ | ↕ | V | 0 | ● | 2 | 8 | q = A,B,C,D,E,IYh y IYl |
| INC [HL] | ++[HL] | ↕ | ↕ | ↕ | V | 0 | ● | 1 | 11 | n = numero 8 bits |
| INC [IX+d] | ++[IX+d] | ↕ | ↕ | ↕ | V | 0 | ● | 3 | 23 |  |
| INC [IY+d] | ++[IY+d] | ↕ | ↕ | ↕ | V | 0 | ● | 3 | 23 |  |
| DEC r | --r | ↕ | ↕ | ↕ | V | 1 | ● | 1 | 4 |  |
| DEC p | --p | ↕ | ↕ | ↕ | V | 1 | ● | 2 | 8 |  |
| DEC q | --q | ↕ | ↕ | ↕ | V | 1 | ● | 2 | 8 |  |
| DEC [HL] | --[HL] | ↕ | ↕ | ↕ | V | 1 | ● | 1 | 11 |  |
| DEC [IX+d] | --[IX+d] | ↕ | ↕ | ↕ | V | 1 | ● | 3 | 23 |  |
| DEC [IY+d] | --[IY+d] | ↕ | ↕ | ↕ | V | 1 | ● | 3 | 23 |  |

**16-Bit Add, Substract, Inc & Dec**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| ADD HL, ss | HL+=ss | ● | ● | ↕ | ● | 0 | ↕ | 1 | 11 | ss = BC, DE, HL y SP |
| ADD IX, pp | IX+=pp | ● | ● | ↕ | ● | 0 | ↕ | 2 | 15 | pp = BC, DE, IX y SP |
| ADD IY, qq | IY+=qq | ● | ● | ↕ | ● | 0 | ↕ | 2 | 15 | qq = BC, DE, Iy y SP |
| ADC HL, ss | HL+=ss+CF | ↕ | ↕ | ↕ | V | 0 | ↕ | 2 | 15 |  |
| SBC HL, ss | HL-=ss-CF | ↕ | ↕ | ↕ | V | 0 | ↕ | 2 | 15 |  |
| INC ss | ++ss | ● | ● | ● | ● | ● | ● | 1 | 6 |  |
| INC IX | ++IX | ● | ● | ● | ● | ● | ● | 2 | 10 | ● No toca nada |
| INC IY | ++IY | ● | ● | ● | ● | ● | ● | 2 | 10 | 0 Siempre a 0 |
| DEC ss | ++ss | ● | ● | ● | ● | ● | ● | 1 | 6 | 1 Siempre a 1 |
| DEC IX | ++IX | ● | ● | ● | ● | ● | ● | 2 | 10 | ↕ Cambio normal |
| DEC IY | ++IY | ● | ● | ● | ● | ● | ● | 2 | 10 | ? Ni idea, no importa |

**Jump**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| CP A, r | (A-r) compara | ↕ | ↕ | ↕ | V | 1 | ↕ | 1 | 4 | si 0 Z=1 C=0 |
| CP A, p | (A-p) | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 8 | si >0 Z=0 C=0 |
| CP A, q | (A-q) | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 8 | si <0 Z=0 C=1 |
| CP A, n | (A-n) | ↕ | ↕ | ↕ | V | 1 | ↕ | 2 | 7 | n = numero 8 bits |
| CP A, [HL] | (A-[HL]) | ↕ | ↕ | ↕ | V | 1 | ↕ | 1 | 7 | nn = numero 16 bits |
| CP A,[IX+d] | (A-[IX+d]) | ↕ | ↕ | ↕ | V | 1 | ↕ | 3 | 19 | cc=Z,NZ,C,NC,PO,PE,P,M |
| CP A,[IY+d] | (A-[IY+d]) | ↕ | ↕ | ↕ | V | 1 | ↕ | 3 | 19 | **Z** Cero **NZ** No Cero |
| JP nn | goto nn; PC=nn | ● | ● | ● | ● | ● | ● | 3 | 10 | **C** Acarreo **NC** No Acarreo |
| JP [HL] | goto [HL] | ● | ● | ● | ● | ● | ● | 1 | 4 | **PE** Paridad Par |
| JP [IX] | goto [IX] | ● | ● | ● | ● | ● | ● | 2 | 8 | **PO** Paridad Impar |
| JP [IY] | goto [IY] | ● | ● | ● | ● | ● | ● | 2 | 8 | **P** Positivo |
| JP cc, nn | if(cc) goto nn | ● | ● | ● | ● | ● | ● | 3 | 10 | **M** Negativo |
| JR e | goto PC+e; PC+=e | ● | ● | ● | ● | ● | ● | 2 | 12 | e = 129 a -127 |
| JR ss, e | if(ss) goto PC+e | ● | ● | ● | ● | ● | ● | 2 | 127 | ss = C, NC, Z y NZ  12T si (ss==true) |
| DJNZ e | --B;  if(B!=0) goto PC+e | ● | ● | ● | ● | ● | ● | 2 | 13  8 | 13T si (B!=0)  8T si (B==0) |
| CALL nn | funcion(nn) | ● | ● | ● | ● | ● | ● | 3 | 17 | nn = numero 16 bits |
| CALL cc, nn | if(cc) funcion(nn) | ● | ● | ● | ● | ● | ● | 3 | 17  10 | 17T si (cc==true)  10T si (cc==false) |
| RET | return | ● | ● | ● | ● | ● | ● | 1 | 10 |  |
| RET cc | if(cc) return | ● | ● | ● | ● | ● | ● | 1 | 11  5 | 11T si (cc==true)  5T si (cc==false) |
| RST p | funcion(p) | ● | ● | ● | ● | ● | ● | 1 | 11 | p=$0,$8,$10,$18,$20,$28,$30,$38 |
| NOP | nada | ● | ● | ● | ● | ● | ● | 1 | 4 |  |
| HALT | espera interrupción | ● | ● | ● | ● | ● | ● | 1 | 4 | HALT + DI = CUELGUE |
| DI | desactiva interrup. | ● | ● | ● | ● | ● | ● | 1 | 4 | ganamos velocidad |
| EI | activa interrup. | ● | ● | ● | ● | ● | ● | 1 | 4 |  |
| IM 0 | *no usado por MSX* | ● | ● | ● | ● | ● | ● | 2 | 8 | *no usado por el MSX* |
| IM 1 | interrup. tipo 1 | ● | ● | ● | ● | ● | ● | 2 | 8 | salta a $0038 |
| IM 2 | *no usado por MSX* | ● | ● | ● | ● | ● | ● | 2 | 8 | *no usado por el MSX* |
| RETI | return interrup. | ● | ● | ● | ● | ● | ● | 2 | 14 | retorna de interrupción |
| RETN | *no usado por MSX* | ● | ● | ● | ● | ● | ● | 2 | 14 | *no usado por el MSX* |

**Bit Set, Reset & Test**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| BIT b, r | ZF=!(rb) | ↕ | ↕ | 1 | ↕ | 0 | ● | 2 | 8 | r=A,B,C,D,E,H,L y A |
| BIT b, [HL] | ZF=!(HLb) | ↕ | ↕ | 1 | ↕ | 0 | ● | 2 | 12 | b = bit  000 0  001 1  010 2  011 3  100 4  101 5  110 6  111 7 |
| BIT b,[IX+d] | ZF=!(IX+d)b | ↕ | ↕ | 1 | ↕ | 0 | ● | 4 | 20 |
| BIT b,[IY+d] | ZF=!(IY+d)b | ↕ | ↕ | 1 | ↕ | 0 | ● | 4 | 20 |
| SET b, r | rb=1 | ● | ● | ● | ● | ● | ● | 2 | 8 |
| SET b, [HL] | [HL]b=1 | ● | ● | ● | ● | ● | ● | 2 | 15 |
| SET b,[IX+d] | [IX+d]b=1 | ● | ● | ● | ● | ● | ● | 4 | 23 |
| SET b,[IY+d] | [IY+d]b=1 | ● | ● | ● | ● | ● | ● | 4 | 23 |
| SET b,[IX+d],r | r=[IX+d]  rb=1; [IX+d]=r | ● | ● | ● | ● | ● | ● | 4 | 23 |
| SET b,[IY+d],r | r=[IY+d]  rb=1; [IY+d]=r | ● | ● | ● | ● | ● | ● | 4 | 23 |
| RES b, r | rb=0 | ● | ● | ● | ● | ● | ● | 2 | 8 | ● No toca nada  0 Siempre a 0  1 Siempre a 1  ↕ Cambio normal  ? Ni idea, no importa |
| RES b, [HL] | [HL]b=0 | ● | ● | ● | ● | ● | ● | 2 | 15 |
| RES b,[IX+d] | [IX+d]b=0 | ● | ● | ● | ● | ● | ● | 4 | 23 |
| RES b,[IY+d] | [IY+d]b=0 | ● | ● | ● | ● | ● | ● | 4 | 23 |
| RES b,[IX+d],r | r=[IX+d]  rb=0; [IX+d]=r | ● | ● | ● | ● | ● | ● | 4 | 23 |
| RES b,[IY+d],r | r=[IY+d]  rb=0; [IY+d]=r | ● | ● | ● | ● | ● | ● | 4 | 23 |

**Input & Output**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| IN A, [n] | A=[n] | ● | ● | ● | ● | ● | ● | 2 | 11 | r=A,B,C,D,E,H,L y A |
| IN r, [C] | r=[C] | ↕ | ↕ | 0 | P | 0 | ● | 2 | 12 |  |
| IN F, [n] | <-[C] | ↕ | ↕ | 0 | P | 0 | ● | 2 | 12 |  |
| INI | [HL]=[C]; ++HL; --B | ♠ | ♠ | ? | ? | ♣ | ? | 2 | 16 | 1 ♠ igual que en DEC B |
| INIR | [HL]=[C]; ++HL; --B  hasta (B==0) | 0 | 1 | ? | ? | ♣ | ? | 2 | 21  16 | T21 si (B!=0)  T15 si (B==0) |
| IND | [HL]=[C];  --HL; --B | ♠ | ♠ | ? | ? | ♣ | ? | 2 | 16 | 2 ♣ copia del bit 7 |
| INDR | [HL]=[C]; --HL; --B  hasta (B==0) | 0 | 1 | ? | ? | ♣ | ? | 2 | 21  16 | T21 si (B!=0)  T16 si (B==0) |
| OUT [n], A | [n]=A | ● | ● | ● | ● | ● | ● | 2 | 11 |  |
| OUT [C], r | [C]=r | ● | ● | ● | ● | ● | ● | 2 | 12 |  |
| OUT [C], 0 | [C]=0 | ● | ● | ● | ● | ● | ● | 2 | 12 |  |
| OUTI | [C]=[HL]; ++HL; --B | ♠ | ♠ | ? | ? | ♣ | ? | 2 | 16 |  |
| OTIR | [C]=[HL]; ++HL; --B  hasta (B==0) | 0 | 1 | ? | ? | ♣ | ? | 2 | 21  16 | T21 si (B!=0)  T16 si (B==0) |
| OUTD | [C]=[HL]; --HL; --B | ♠ | ♠ | ? | ? | ♣ | ? | 2 | 16 |  |
| OTDR | [C]=[HL]; --HL; --B  hasta (B==0) | 0 | 1 | ? | ? | ♣ | ? | 2 | 21  16 | T21 si (B!=0)  T16 si (B==0) |

**Rotate and Shift**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Nemónico** | **Operación** | **Banderas** | | | | | | **B** | **T** | **Comentarios** |
| S | Z | H | PV | N | C |
| SLA m |  | ↕ | ↕ | 0 | P | 0 | ↕ | 2 | 8 | r=A,B,C,D,E,H,L y A |
| SLL m |  | ↕ | ↕ | 0 | P | 0 | ↕ | 2 | 8 | m=r,[HL],[IX+d],[IY+d] |
| SRA m |  | ↕ | ↕ | 0 | P | 0 | ↕ | 2 | 8 |  |
| SRL m |  | ↕ | ↕ | 0 | P | 0 | ↕ | 2 | 8 |  |
| RLCA |  | ● | ● | 0 | ● | 0 | ↕ | 1 | 4 |  |
| RLC r |  | ↕ | ↕ | 0 | P | 0 | ↕ | 2 | 8 |  |
| RLC [HL] |  | ↕ | ↕ | 0 | P | 0 | ↕ | 2 | 15 |  |
| RLC [IX+d] |  | ↕ | ↕ | 0 | P | 0 | ↕ | 4 | 23 |  |
| RLC [IY+d] |  | ↕ | ↕ | 0 | P | 0 | ↕ | 4 | 23 |  |
| RLC [IX+d],r | r=[IX+d]; RLC r  [IX+d]=r | ↕ | ↕ | 0 | P | 0 | ↕ | 4 | 23 |  |
| RLC [IY+d],r | r=[IY+d]; RLC r  [IY+d]=r | ↕ | ↕ | 0 | P | 0 | ↕ | 4 | 23 |  |
| RRCA |  | ● | ● | 0 | ● | 0 | ↕ | 1 | 4 |  |
| RRC m |  | ↕ | ↕ | 0 | P | 0 | ↕ | 2 | 8 |  |
| RLA |  | ● | ● | 0 | ● | 0 | ↕ | 1 | 4 |  |
| RL m |  | ↕ | ↕ | 0 | P | 0 | ↕ | 2 | 8 | ● No toca nada |
| RRA |  | ● | ● | 0 | ● | 0 | ↕ | 1 | 4 | 0 Siempre a 0 |
| RR m |  | ↕ | ↕ | 0 | P | 0 | ↕ | 2 | 8 | 1 Siempre a 1 |
| RLD |  | ↕ | ↕ | 0 | P | 0 | ● | 2 | 18 | ↕ Cambio normal |
| RRD |  | ↕ | ↕ | 0 | P | 0 | ● | 2 | 18 | ? Ni idea, no importa |

\* **CPL**: negación. NOT 0 = 1, NOT 1 = 0, cambia todos los bits.

\* **NEG**: Complemento a dos niega todos los bits y les suma uno. (CPL + INC)

\* **AND**: producto lógico, vale uno sólo si los dos bits son uno:

0 AND 0 = 0

0 AND 1 = 0

1 AND 0 = 0

1 AND 1 = 1

\* **OR**: suma lógica, vale uno si al menos uno de los dos bits vale uno.

0 OR 0 = 0

0 OR 1 = 1

1 OR 0 = 1

1 OR 1 = 1

\* **XOR**: OR exclusiva. El resultado es uno si sólo uno de los bits es uno.

0 XOR 0 = 0

0 XOR 1 = 1

1 XOR 0 = 1

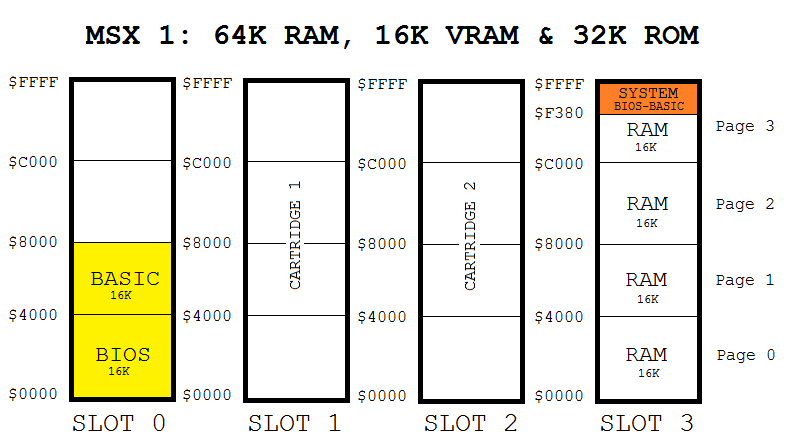
1 XOR 1 = 0

**Los Registros del Z80**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | F | <- **EX** AF,AF' -> | A' | F' | Acumulador y Flags (banderas) |
| B | C |  | B' | C' |  |
| D | E | <- **EXX** -> | D' | E' |  |
| H | L |  | H' | L' |  |
| I | R | Interrupción y Refresco de Memoria |  |  |  |
| IX | | <- Con instrucciones -> | IXh | IXl | Registros de Indexación de 16 bits Ejemp: **LD** H,(IX+n) |
| IY | |  | IYh | IYl |
| SP | | Stack Pointer (puntero de pila) |  |  |  |
| PC | | Program Counter (contador de programa) |  |  |  |

**Las banderas del registro F**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **bit 7** | **bit 6** | **bit 5** | **bit 4** | **bit 3** | **bit 2** | **bit 1** | **bit 0** |
| **S** | **Z** | **5** | **H** | **3** | **P/V** | **N** | **C** |
| signo | cero | bit 5 | acarreo  mitad | bit 3 | paridad  desbord. | resta | acarreo |

**CF** = 1 **SCF** **CF** = 0 **OR** A **ZF** = 1 **CP** A, A **ZF** = 0 **AND** 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **D** | **H** | **Bin** | **Nombre** | **R** | **G** | **B** |
| 0 | 0 | 0000 | Transp. | 0 | 0 | 0 |
| 1 | 1 | 0001 | Negro | 0 | 0 | 0 |
| 2 | 2 | 0010 | Verde | 33 | 200 | 66 |
| 3 | 3 | 0011 | Verde  Claro | 94 | 220 | 120 |
| 4 | 4 | 0100 | Azul  Oscuro | 84 | 85 | 237 |
| 5 | 5 | 0101 | Azul  Claro | 125 | 118 | 252 |
| 6 | 6 | 0110 | Rojo  Oscuro | 212 | 82 | 77 |
| 7 | 7 | 0111 | Azul  Cielo | 66 | 235 | 245 |
| 8 | 8 | 1000 | Rojo | 252 | 85 | 84 |
| 9 | 9 | 1001 | Rojo  Claro | 255 | 121 | 120 |
| 10 | A | 1010 | Amarillo | 212 | 193 | 84 |
| 11 | B | 1011 | Amarillo  Claro | 230 | 206 | 128 |
| 12 | C | 1100 | Verdo  Oscuro | 33 | 179 | 59 |
| 13 | D | 1101 | Púrpura  Magenta | 201 | 91 | 186 |
| 14 | E | 1110 | Gris | 204 | 204 | 204 |
| 15 | F | 1111 | Blanco | 255 | 255 | 255 |

****

**PPI POR PUERTOS**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PPI (A) Port In/Out $A8 Configura los SLOTS | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Page 3  $C000 - $FFFF | | Page 2  $8000 - $BFFF | | Page 1  $4000 - $7FFF | | Page 0  $0000 - $3FFF | |
| nº de slot | | nº de slot | | nº de slot | | nº de slot | |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | PPI (B) Port In $A9 Keyboard Column | | | | | | | | |
| PPI (C)  Port Out  $AA  Keyboard  Row |  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0000 | &  7 | ^  6 | %  5 | $  4 | #  3 | @  2 | !  1 | )  0 |
| 0001 | :  ; | }  ] | {  [ | |  \ | +  = | - | (  9 | \*  8 |
| 0010 | B | A | ACENT | ?  / | >  . | <  , | ~  ` | “  ‘ |
| 0011 | J | I | H | G | F | E | D | C |
| 0100 | R | Q | P | O | N | M | L | K |
| 0101 | Z | Y | X | W | V | U | T | S |
| 0110 | F3 | F2 | F1 | CODE | CAPS | GRAPH | CTRL | SHIFT |
| 0111 | RETURN | SELECT | BS | STOP | TAB | ESC | F5 | F4 |
| 1000 | → | ↓ | ↑ | ← | DEL | INS | HOME  CLS | SPACE |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PPI (C) Port Out $AA | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Key Click | Cap  LED | Cas  Out | Cas  Motor | Keyboard Row | | | |
| 0=On  1=Off | 0=On  1=Off | 0=On  1=Off | 0=On  1=Off | [ 0000 ~ 1000 ] | | | |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PPI Mode Port $ABH | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 1 | Mode A &  C upper four bits | | A $A8  Dir | C $AA  Dir upper four bits | Mode B &  C lower four bits | B $A9  Dir | C $AA  Dir lower four bits |
| 00=Normal Mode  01=Strobed Mode  10=Bidirectional | | 0=Output  1=Input | 0=Output  1=Input | 0=Normal Mode 1=Strobed Mode | 0=output  1=Input | 0=Output  1=Input |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PPI Mode Port $ABH | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | Not used | | | Bit Number | | | Set |
|  | | | 00=0  01=1  10=2  11=3 | | | 1=Set  0=Reset |

**PSG POR PUERTOS**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| REGISTRO / BIT | | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | PERIODO CANAL A | AJUSTE FINO | | | | | | | |
| 1 |  |  |  |  | AJUSTE GRUESO | | | |
| 2 | PERIODO CANAL B | AJUSTE FINO | | | | | | | |
| 3 |  |  |  |  | AJUSTE GRUESO | | | |
| 4 | PERIODO CANAL C | AJUSTE FINO | | | | | | | |
| 5 |  |  |  |  | AJUSTE GRUESO | | | |
| 6 | PERIODO RUIDO |  |  |  | 5 BITS AJUSTE (0~31) | | | | |
| 7 | ACTIVACIÓN | E/S | | RUIDO | | | TONO | | |
| IOB | IOA | C | B | A | C | B | A |
| 8 | VOLUMEN A |  |  |  | M | 4 BITS VOLUMEN (0~15) | | | |
| 9 | VOLUMEN B |  |  |  | M | 4 BITS VOLUMEN (0~15) | | | |
| 10 | VOLUMEN C |  |  |  | M | 4 BITS VOLUMEN (0~15) | | | |
| 11 | PERIODO  ENVOLVENTE | 8 BITS AJUSTE FINO | | | | | | | |
| 12 | 8 BITS AJUSTE GRUESO | | | | | | | |
| 13 | FORMA ENVOLVENTE |  |  |  |  | CONT | ATT | ALT | HOLD |
| 14 | PUERTO A JOYSTICK | CAS | KDB | B | A | → | ← | ↓ | ↑ |
| 15 | PUERTO B JOYSTICK | INP | MODE | B | A | → | ← | ↓ | ↑ |

Frecuencia = ( 1789772.5 Hz ) / 16 \* ( 256 \* Grueso + Fino )

X = INT( 1789772.5 / ( 16 \* Frecuencia )

Fino = INT( X / 256 )

Grueso = X – Fino

HOLD: Sostenimiento o mantenimiento, determina si la envolvente mantiene o no, el valor final del primer ciclo.

0 - La envolvente se repetirá periódicamente.

1 - El volumen del sonido se sostendrá.

ALT: Alternativamente, si está a uno se sucederán alternativamente rampas ascendentes y descendentes que determinarán una envolvente triangular.

ATT: Es ataque es el periodo inicial de la producción del sonido

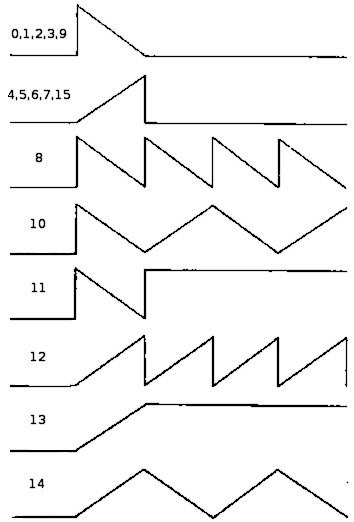
0 – Ascendente

1 - Descendente

CONT: La envolvente se repite periódicamente o no después del primer ciclo.

0 – Al terminar el primer ciclo el volumen caerá a cero.

1 – La envolvente se repite.



**ESCRIBIR**

**LD** A, n ; n. registro (0~15)

**OUT** [$A0], A ; al PSG

**LD** A, v ; valor registro (0~255)

**OUT** [$A1], A ;

**LEER**

**LD** A, n ; n. registro (0~15)

**OUT** [$A0], A ; al PSG

**IN** A,[$A2] ; lee el valor devuelto

**LD** [v],A ; guardamos el valor

**VDP POR PUERTOS**

**Puerto de Datos (I/O $98) “Data Port”**

* El Puerto de Datos se utiliza para leer o escribir bytes individuales a la VRAM.
* El VDP posee un Registro de Dirección interno que apunta a una localización en la VRAM.
* Leyendo o escribiendo el Puerto de Datos accedemos al byte de la VRAM indicado por el Registro de Dirección.
* Después de una lectura o escritura el Registro de Direcciones se incrementa automáticamente para que apunte a la siguiente posición de VRAM.
* Bytes secuenciales pueden ser accedidos simplemente leyendo o escribiendo continuamente el Puerto de Datos.

**Puerto de Comandos (I/O $99) “Command Port”**

El Puerto de Comandos se utiliza para tres propósitos:

1. Para configurar el Registro de Direcciones del Puerto de Datos.
2. Para leer el Registro de Estado del VDP

**IN** A, [$99] ;

1. Para escribir en uno de los Registros de Modo del VDP

**El Registro de Direcciones:**

* El Registro de Dirección del Puerto de Datos debe estar configurado de diferentes maneras dependiendo de si el acceso posterior va a ser una lectura o una escritura.
* El Registro de Dirección puede ajustarse a cualquier valor de 0000H a 3FFFH escribiendo primero el LSB (byte menos significativo) y luego el MSB (byte más significativo) en el Puerto de Comandos.
* Los bits 6 y 7 del MSB son utilizados por el VDP para determinar si el Registro de Dirección está configurado para lectura o escritura de la siguiente manera:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LECTURA | LSB  $00-$FF | | | | | | | | 0 | 0 | MSB  $00-$3F | | | | | |
| ESCRITURA | 0 | 1 |

Es importante que no se produzcan otros accesos al VDP entre la escritura del LSB y del MSB, ya que esto alteraría su sincronización. El Gestor de Interrupciones de la ROM BIOS está leyendo continuamente el estado del Registro de Estado como una tarea en segundo plano por lo que las interrupciones se deben desactivar y activar cuando sea necesario. Para esto contamos con las instrucciones **DI** y **EI** respectivamente.

Escribir:

**IN** A, [$99] ; Leemos el Registro de Estado (para sincronizar)

**LD** A, LSB ; Dirección LSB de la VRAM

**OUT** [$99], A ; La mandamos al Puerto de Comandos

**LD** A, MSB + 64 ; Dirección MSB de la VRAM, (bit 7 = 0, bit 6 = 1)

**OUT** [$99], A ; La mandamos al Puerto de Comandos

**LD** A, DATO ; El DATO a escribir en la VRAM

**OUT** [$98], A ; Mandamos al Puerto de Datos

Leer:

**IN** A, [$99] ; Leemos el Registro de Estado (para sincronizar)

**LD** A, LSB ; Dirección LSB de la VRAM

**OUT** [$99], A ; La mandamos al Puerto de Comandos

**LD** A, MSB ; Dirección MSB de la VRAM, (bit 7 = 0, bit 6 = 0)

**OUT** [$99], A ; La mandamos al Puerto de Comandos

**IN** A, [$98] ; Leemos del Puerto de Datos

**El Registro de Estado:**

Leyendo el Puerto de Comandos tenemos acceso al Registro de Estado que contiene la siguiente información:

**IN** A, [$99] ; Leemos el Registro de Estado (para sincronizar)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| F  Frame Int. | 5S  5º Sprite | C  Colisión | Número del 5º Sprite (0 a 31) | | | | |

* La bandera de (F) Frame está normalmente a 0, pero se pone a 1 al llegar al final de la última línea activa de video. Si la frecuencia es de 50Hz y esto ocurre cada 20ms, y si es de 60Hz cada 16,666ms. Leyendo el Registro de Estado esta bandera se pone a 0.
* La bandera de (5S) 5º Sprite se activa a 1 cuando hay más de cuatro Sprites en la misma línea de video. Leyendo el Registro de Estado esta bandera se pone a 0.
* La bandera de (C) Colisión se activa cuando los Sprites tienen un o más pixels superpuestos, si los Sprites en movimiento son tan rápidos que pasan por encima uno del otro entre los Frames no se detectará ninguna colisión.
* Los bits del 0 al 4 contienen el número del Sprite que ha desencadenado la bandera de 5º Sprite.
* Leyendo el Registro de Estado las banderas se ponen a 0.

**Registros de Modo:**

El VDP tiene ocho Registros de Modo que son de sólo escritura numerados del 0 al 7. Un Registro en particular se escribe mandando un byte de datos en el Puerto de Datos y a continuación un byte de selección de Registro al Puerto de Comandos. El byte de selección de Registro contiene el número de registro en los tres bits más bajos y su bit más alto debe ser 1.

**LD** A, DATO ; El DATO a escribir en la Registro

**OUT** [$98], A ; Mandamos al Puerto de Datos

**LD** A, núm\_reg + 128 ; Número de Registro de 0 a 7 + 128

**OUT** [$99], A ; Seleccionamos un registro

**Registro de Modo 0:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | M3 | EV |

* El bit de señal de video Externa VDP determina si la entrada del VDP externa debe ser activada o desactivada:
  + 0 = Desactivado,
  + 1 = Habilitado.
* El bit M3 se usa en unión con el M1 y el M2 del Registro de Modo 1.

**Registro de Modo 1:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 4/16K | Blank | IE | M1 | M2 | 0 | Size | Mag |

* La bandera de (Mag) Magnificación duplica el tamaño de los Sprites:
  + 0=Normal
  + 1=Doble
* La bandera (Size) de Tamaño determina el tamaño de los Sprites:
  + 0=8x8 bits
  + 1=16x16 bits
* Las banderas M1, M2 y M3 determinan el modo de video:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| M1 | M2 | M3 | Nº | DESCRIPCION | PATRONES  MODELOS | NOMBRES  CARACTER | TABLA  COLOR | SPRITES  PATRONES | SPRITES  ATRIBUTOS |
| 1 | 0 | 0 | 0 | Texto 40x24 | 256x8=  2048 bytes | 40x24=  960 bytes | 2 colores | NO | NO |
| $0800  $0FFF | $0000  $03BF | Registro Modo 7 |
| 0 | 0 | 0 | 1 | Texto 32x24 | 256x8=  2048 bytes | 32x24=  768 bytes | 16 colores  (4x3)x3=  32 bytes | 2028 bytes | 128 bytes |
| $0000  $07FF | $1800  $1AFF | $2000  $201F | $3800  $3FFF | $1B00  $1B7F |
| 0 | 0 | 1 | 2 | Gráficos 32x24  (256x192 pixels) | 768x8=  6144 bytes | 32x24=  768 bytes | 16 colores  768x8=  6144 bytes | 2028 bytes | 128 bytes |
| $0000  $17FF | $1800  $1AFF | $2000  $37FF | $3800  $3FFF | $1B00  $1B7F |
| 0 | 1 | 0 | 3 | Modo Multicolor  (64x24 pixels) | 1563 bytes | 768 bytes | 16 colores | 2028 bytes | 128 bytes |
| $0000  $05FF | $0800  $0AFF | Incluida en Patrones | $3800  $3FFF | $1B00  $1B7F |
| Color del Borde: Registro de Modo 7 (azul) | | | | | | | | | |

* La bandera (IE) Activa o Desactiva la Interrupción del VDP cuando llega a la última línea dibujable, el VDP saltará automáticamente a la dirección de memoria ROM $38, si está activada la BIOS esta permite un gancho en la dirección de memoria $FD9A a $FD9E donde puedes poner CALL mi\_rutina; RET; RET; y a funcionar:
  + 0=Desactiva
  + 1=Activa
* La bandera (Blank) Activa o Desactiva el video en pantalla, cuando esta desactivado se muestra el color del borde, no le veo mucha utilidad asi que déjalo siempre a 1:
  + 0=Desactiva
  + 1=Activa
* La bandera (4/16K) selecciona el tipo de chip de memoria del VDP es mejor que lo dejes siempre a 1:
  + 0=4K
  + 1=16K

**Registro de Modo 2:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | Name Table Base | | | |

Este registro tiene un valor de 0 a 15, que al multiplicarse por $400, da la dirección de comienzo en la VRAM de la Tabla de Nombres.

**Registro de Modo 3:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Colour Table Base | | | | | | | |

Este registro tiene un valor entre 0 y 255, que al multiplicarse por $40 nos da la dirección de comienzo en la VRAM de la Tabla de Colores.

**Registro de Modo 4:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | Character Pattern | | |

Este registro tiene un valor entre 0 y 7, que al multiplicarse por $800 nos da la dirección de comienzo en la VRAM de la Tabla de Patrones.

Los registros 3 y 4 en el modo gráfico de alta resolución 256x192 se comportan diferente, ver tabla anexa:

**Tablas Registro 3 TABLA COLOR Registro 4 TABLA PATRONES**

1 $1F - $0000, $0000, $0000 $00 - $0000, $0000, $0000

2 $3F - $0000, $0800, $0000 $01 - $0000, $0800, $0000

2 $5F - $0000, $0000, $1000 $02 - $0000, $0000, $1000

3 $7F - $0000, $0800, $1000 $03 - $0000, $0800, $1000

1 $9F - $2000, $2000, $2000 $04 - $2000, $2000, $2000

2 $BF - $2000, $2800, $2000 $05 - $2000, $2800, $2000

2 $DF - $2000, $2000, $3000 $06 - $2000, $2000, $3000

3 $FF - $2000, $2800, $3000 $07 - $2000, $2800, $3000

**Registro de Modo 5:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | Sprite Attribute Base | | | | | | |

Este registro tiene un valor entre 0 y 127, que al multiplicarse por $80 nos da la dirección de comienzo en la VRAM de la Tabla de Atributos de Sprites.

**Registro de Modo 6:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | Sprite Pattern | | |

Este registro tiene un valor entre 0 y 7, que al multiplicarse por $800 nos da la dirección de comienzo en la VRAM de la Tabla de Patrones de Sprites.

**Registro de Modo 7:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Color del Texto u Tinta  En Modo 0 bits a 1 | | | | Color del Borde  En Modo 0 bits a 0 | | | |

**Sprites**

Menos en el modo 0, el VDP puede manejar 32 sprites en todos los modos y el funcionamiento es idéntico.

La Tabla de Atributos ocupa 128 bytes de VRAM, contiene 32 bloques de 4 bytes para cada sprite. El primer bloque es en número cero y es el sprite más al frente. El 31 es el último sprite y el más al fondo.

El formato de cada bloque es el siguiente:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bits | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Byte 0 | Posición Vertical de -32 ~ 191 | | | | | | | |
| Byte 1 | Posición Horizontal 0 ~ 255 | | | | | | | |
| Byte 2 | Número de Patrón | | | | | | | |
| Byte 0 | EC | 0 | 0 | 0 | Código de Color | | | |

La posición se cuenta desde el tope izquierdo de la pantalla.

Y = 208 Hace que todos los sprites de menor prioridad desaparezcan de la pantalla.

Y = 209 Desaparece el sprite.

EC (Early Clock Bit) si se pone a 1 desplaza el sprite 32 unidades a la izquierda.

La Tabla de Patrones de Sprite contiene 256 bloques de 8x8 pixels o 64 bloques de 16x16 pixels. No se pueden mezclar estos dos tipos.

* Registro de Modo 1 bit 0: Magnificación duplica el tamaño de los Sprites:
  + 0=Normal
  + 1=Doble
* Registro de Modo 1 bit 1: Tamaño determina el tamaño de los Sprites:
  + 0=8x8 bits
  + 1=16x16 bits

|  |  |  |  |
| --- | --- | --- | --- |
| 8 Bytes |  | 16 Bytes | |
| Block A | A | C |
| Block B |
| Block C | B | D |
| Block D |