Zilog Z80 CPU Assembler Syntax

This page is typed and converted to HTML by Thomas Scherrer
I'm still working on this document, so you can come back later to see it develop..

ASM: Is the assembler source code.
OBJ: Is the object code (binary executable).

Select the Mnemonic you want detailed info about:

Mnemonic:	Description:
ADC	ADD WITH CARRY
ADD	ADD
AND	LOGICAL AND
BIT	BIT TEST
CALL	CALL SUB ROUTINE
CCF	COMPLEMENT CARRY FLAG
CP	COMPARE
CPD	COMPARE AND DECREMENT
CPDR	COMPARE DECREMENT AND REPEAT
CPI	COMPARE AND INCREMENT
CPIR	COMPARE INCREMENT AND REPEAT
CPL	COMPLEMENT ACCUMULATOR
DAA	DECIMAL ADJUST ACCUMULATOR
DEC	DECREMENT
DI	DISABLE INTERRUPTS
DJNZ	DEC JUMP NON-ZERO
EI	ENABLE INTERRUPTS
EX	EXCHANGE REGISTER PAIR
EXX	EXCHANGE ALTERNATE REGISTERS
HALT	HALT, WAIT FOR INTERRUPT OR RESET
IM	INTERRUPT MODE 0 1 2
IN	INPUT FROM PORT
INC	INCREMENT
IND	INPUT, DEC HL, DEC B
INDR	INPUT, DEC HL, DEC B, REPEAT IF B>0
INI	INPUT, INC HL, DEC B
INIR	INPUT, INC HL, DEC B, REPEAT IF B>0
JP	JUMP
JR	JUMP RELATIVE
LD	LOAD DATA TO/FROM REGISTERS/MEMORY
LDD	LOAD DECREMENT
LDDR	LOAD DECREMENT AND REPEAT
LDI	LOAD AND INCREMENT
LDIR	LOAD INCREMENT AND REPEAT
NEG	NEGATE ACCUMULATOR 2'S COMPLEMENT
NOP	NO OPERATION
OR	--
OTDR	OUTPUT, DEC HL, DEC B, REPEAT IF B>0
OTIR	OUTPUT, INC HL, DEC B, REPEAT IF B>0
OUT	OUTPUT TO PORT
OUTD	OUTPUT, DEC HL, DEC B
OUTI	OUTPUT, INC HL, DEC B
POP	POP FROM STACK
PUSH	PUSH INTO STACK
RES	RESET BIT
RET	RETURN FROM SUB ROUTINE
RETI	RETURN FROM INTERRUPT
RETN	RETURN FROM NON MASKABEL INTERRUPT
RL	ROTATE LEFT register
RLA	ROTATE LEFT ACUMULATOR

RLC	ROTATE LEFT THROUGH CARRY register
RLCA	ROTATE LEFT THROUGH CARRY ACCUMULATUR
RLD	ROTATE LEFT DIGIT
RR	ROTATE RIGHT register
RRA	ROTATE RIGHT ACCUMULATOR
RRC	ROTATE RIGHT CIRCULAR register
RRCA	ROTATE RIGHT CIRCULAR ACCUMULATOR
RRD	ROTATE RIGHT DIGIT
RST	RESTART
SBC	SUBTRACT WITH CARRY
SCF	SET CARRY FLAG
SET	SET BIT
SLA	SHIFT LEFT ARITHMETIC register
SRA	SHIFT RIGHT ARITHMETIC register
SRL	SHIFT RIGHT LOGICAL register
SUB	SUBTRACTION
XOR	EXCLUSIVE OR

Detailed info ADC

ADD WITH CARRY

Detailed info ADD

Detailed info AND

LOGICAL AND

Detailed info BIT

BIT TEST

Detailed info CALL

CALL a subroutine

Detailed info CCF

Opperation: CY = Inv CY

Instruction Format:

Opcode: CCF
OBJ: 3Fh

Description:

The Carry flag in the flags register is inverted.

Tstates: 4.

Flags:

S: Not affected.
Z: Not affected.
H: Previous carry will be copied to H.
P/V: Not affected.
N: Reset.
C: Set if carry was 0 before operation, reset otherwise.

Detailed info CP

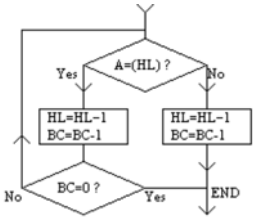
COMPARE

Detailed info CPD

COMPARE AND DECREMENT

Detailed info CPDR

COMPARE DECREMENT AND REPEAT



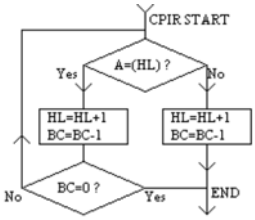

```
Flags: CY Z  PV S  N  H
       .  ~  ~  ~  1  ~ (~ = changes, . = no change)

And the flags: if byte found: Z=1; HL - address of next byte;
                                  BC - figure out by yourself.
                  not found: Z=0
```

Detailed info CPI

COMPARE AND INCREMENT

Detailed info CPIR

COMPARE INCREMENT AND REPEAT




```
Flags: CY Z  PV S  N  H
       .  ~  ~  ~  1  ~ (~ = changes, . = no change)

And the flags: if byte found: Z=1; HL - address of next byte;
                                  BC - figure out by yourself.
                  not found: Z=0
I believe, that if on exit BC=0 then PV=0 else PV=1, that's so with CPI
(CPD) command, so why shouldn't it be with CPIR (CPDR)?

So the difference between if a byte, is or is not found, is expressed
by the Zero flag. Sorry, don't know exactly how are S and H affected...
```

Detailed info CPL

COMPLEMENT ACCUMULATOR

Detailed info DAA

Instruction Format:

```
OPCODE                    CYCLES
--------------------------------
  27h                       4
```

Description:
This instruction conditionally adjusts the accumulator for BCD addition
and subtraction operations. For addition (ADD, ADC, INC) or subtraction
(SUB, SBC, DEC, NEC), the following table indicates the operation performed:

Operation	C Flag Before DAA	HEX value in upper digit (bit 7-4)	H Flag Before DAA	HEX value in lower digit (bit 3-0)	Number added to byte	C flag After DAA
	0	0-9	0	0-9	00	0
ADD	0	0-8	0	A-F	06	0
	0	0-9	1	0-3	06	0
ADC	0	A-F	0	0-9	60	1
	0	9-F	0	A-F	66	1
INC	0	A-F	1	0-3	66	1
	1	0-2	0	0-9	60	1
	1	0-2	0	A-F	66	1
	1	0-3	1	0-3	66	1
SUB	0	0-9	0	0-9	00	0
SBC	0	0-8	1	6-F	FA	0
DEC	1	7-F	0	0-9	A0	1
NEG	1	6-F	1	6-F	9A	1

Flags:

C: See instruction.

N: Unaffected.

P/V: Set if Acc. is even parity after operation, reset otherwise.

H: See instruction.

Z: Set if Acc. is Zero after operation, reset otherwise.

S: Set if most significant bit of Acc. is 1 after operation, reset otherwise.

Example:

If an addition operation is performed between 15 (BCD) and 27 (BCD), simple decimal
arithmetic gives this result:

```
   15
  +27
  ----
   42
```

But when the binary representations are added in the Accumulator according to
standard binary arithmetic:

```
   0001 0101  15
  +0010 0111  27
  --------------
   0011 1100  3C
```

The sum is ambiguous. The DAA instruction adjusts this result so that correct
BCD representation is obtained:

```
   0011 1100  3C result
  +0000 0110  06 +error
  --------------
   0100 0010  42 Correct BCD!
```

Detailed info DEC

DECREMENT

Detailed info **DI**

DISABLE INTERRUPTS

Detailed info **DJNZ**

DEC JUMP NON-ZERO

Detailed info **EI**

ENABLE INTERRUPTS

Detailed info **EX**

EXCHANGE REGISTER PAIR

ASM: EX DE,HL
OBJ: EBh
Tstates : 4.

ASM: EX AF,AF'
OBJ: 08h
Tstates : 4.

Description:
The 16bit value in the two registers are exchanged.
note: register pair AF consists of register A and the flags register F

ASM: EX (SP),HL
OBJ: E3h
Tstates : 19.

Description:
The low order byte contained in register pait HL is exchanged with the contents
of the memory address spefified by the contents of register pair SP (Stack Pointer),
and the high order byte of HL is exchanges with the next highest memory address (SP+1).
H <-> (SP+1) , L <-> (SP) this will not affect the contents of SP.

ASM: EX (SP),IX
OBJ: DDh, E3h
Tstates : 23.

ASM: EX (SP),IY
OBJ: FDh, E3h
Tstates : 23.

Description:
Does the same as above.
eg: IXH <-> (SP+1), IXL <-> (SP) IXH = IX height byte, IXL = IX low byte..

Detailed info EXX

EXCHANGE ALTERNATE REGISTERS

ASM: EXX
OBJ: D9h
Tstates : 4.

Description:
Each 16bit value in register pairs BC, DE, HL is exchanged with the 16bit value in BC', DE', HL' respectively.
eg:
BC <-> BC'
DE <-> DE'
HL <-> HL'

Detailed info HALT

HALT, WAIT FOR INTERRUPT OR RESET

ASM: HALT
OBJ: 76h
Tstates : 4.

Description:
The HALT instruction suspends CPU operation until a interrupt or reset is received.
While in the halted state, the processor will execute NOP's to maintain memory refresh logic.

Detailed info IM

INTERRUPT MODE 0 1 2

Detailed info IN

INPUT FROM PORT

Detailed info INC

INCREMENT

Detailed info IND

Instruction Format:

OPCODE	CYCLES
ED AA	16

Description:
The contents of register C are placed on the bottom half (A0-A7) of the address bus to select the I/O device. Register B may be used as a byte counter, and its contents are placed on the top half (A8-A15) af the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are pointer to store the redden byte in memory. The register pair HL is decremented, the byte counter B is decremented.

Flags:

C: Not affected.

N: Set.

P/V: UnKnown.

H: UnKnown.

Z: Set if B-1=0, reset otherwise.

S: UnKnown.

Example:

Reg C=07h
Reg B=10h
Reg HL=1000h
IO port nr. 07h contain data 7Bh

Then after the execution of
IND
The HL register pair will contain 0FFFh, and register B will contain 0Fh, and memory location 1000h will contain 7Bh.

Detailed info INDR

Instruction Format:

OPCODE	CYCLES
ED BA	16 if B=0 21 if B >< 0

Description:
The contents of register C are placed on the bottom half (A0-A7) of the address bus to select the I/O device. Register B is used as a byte counter, and its contents are placed on the top half (A8-A15) af the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are pointer to store the redden byte in memory. The register pair HL is decremented, the byte counter B is decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the instruction is repeated.

Flags:

C: Not affected.

N: Set.

P/V: UnKnown.

H: UnKnown.

Z: Set.

S: UnKnown.

Example:

Reg C=07h
Reg B=03h
Reg HL=1000h
IO port nr. 07h contain data 51h, A9h, 03h

Then after the execution of
INDR
The HL register pair will contain 0FFDh, and register B will contain zero, and memory locations will have contents as follows:

```
Location:    Contents:
0FFEh        03h
0FFFh        A9h
1000h        51h
```

Detailed info **INI**

Instruction Format:

```
OPCODE        CYCLES
--------------------
 ED  A2         16
```

Description:
The contents of register C are placed on the bottom half (A0-A7) of the address bus to select the I/O device. Register B may be used as a byte counter, and its contents are placed on the top half (A8-A15) af the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are pointer to store the redden byte in memory. Finally the byte counter B is decremented and register pair HL is incremented.

Flags:

C: Not affected.

N: Set.

P/V: UnKnown.

H: UnKnown.

Z: Set if B-1=0, reset otherwise.

S: UnKnown.

Example:

Reg C=07h
Reg B=12h
Reg HL=1000h
IO port nr. 07h contain data 7Bh

Then after the execution of
INI
Memory location 1000h will contain 7Bh, the HL register pair will contain 1001h, and register B will contain 11h.

Detailed info **INIR**

Instruction Format:

```
OPCODE                   CYCLES
-----------------------------------------------
 ED  B2         16 if B=0    21 if B >< 0
```

Description:
The contents of register C are placed on the bottom half (A0-A7) of the address bus to select the I/O device. Register B is used as a byte counter, and its contents are placed on the top half (A8-A15) af the address bus at this time. Then one byte from the selected port is placed on the data bus and written to the CPU. The contents of the HL register pair are pointer to store the redden byte in memory. The register pair HL is incremented, the byte counter B is decremented. If decrementing causes B to go to zero, the instruction is terminated. If B is not zero, the instruction is repeated.

Flags:

C: Not affected.

N: Set.

P/V: UnKnown.

H: UnKnown.

Z: Set.

S: UnKnown.

Example:

```
Reg C=07h
Reg B=03h
Reg HL=1000h
IO port nr. 07h contain data 51h, A9h, 03h

Then after the execution of
INIR
The HL register pair will contain 1003h, and register B will contain zero,
and memory locations will have contents as follows:

Location:    Contents:
1000h        51h
1001h        A9h
1002h        03h
```

Detailed info JP

JUMP to address.

Detailed info JR

JUMP to address, Relative

Detailed info LD

Instruction Format:

```
OPCODE  r    r'                              CYCLES
-----------------------------------------------------
  01   ddd  sss     LD   r,r'    dst,src      4
```

Introductory note -- Binary form of opcodes

  Example: LD r,r'

The 8-bit binary opcode is

```
        r  r'
     01dddsss
```

  ...where "ddd" is a three-bit field specifying the destination,
  and "sss" is a three-bit field specifying the source.

the value for ddd and sss is shown below:

```
Registers
--------------
 A = 111
 B = 000
 C = 001
 D = 010
 E = 011
```

```
 H = 100
 L = 101
```

Operation: dst <-- src

The contest of the source operand are loaded into the destination operand. The contents of the source operand are not affected.

Flags:

C: Unaffected.

N: Unaffected.

P/V: Unaffected.

H: Unaffected.

Z: Unaffected.

S: Unaffected.

Example: If register E, contains the value 10h, The statement:

ASM: LD H,E

OBJ: 63h

Loads the value 10H into register H.

Detailed info LDD

Instruction Format:

```
OPCODE          CYCLES
-----------------------------
 ED  A8           16
```

Description:
This instruction transfers a byte of data from memory location addressed
by the register pair HL, to the memory location addressed by the register pair DE.
Then both af these register pairs including the BC (byte counter) register pair
are decremented.

Flags:

C: Not affected.

N: reset.

P/V: Set if BC-1 is not 0, reset otherwise.

H: Reset.

Z: Not affected.

S: Not affected.

Detailed info LDDR

Instruction Format:

```
OPCODE          CYCLES
-----------------------------
 ED  A8           16
```

Description:
This instruction transfers a byte of data from memory location addressed
by the register pair HL, to the memory location addressed by the register pair DE.
Then both af these register pairs including the BC (byte counter) register pair
are decremented.

Flags:

C: Not affected.

N: reset.

P/V: Set if BC-1 is not 0, reset otherwise.

H: Reset.

Z: Not affected.

S: Not affected.

Detailed info LDI

LOAD AND INCREMENT

Detailed info LDIR

LOAD INCREMENT AND REPEAT

Detailed info NEG

NEGATE ACCUMULATOR 2'S COMPLEMENT

Detailed info NOP

NO OPERATION, not the most complicated instruction :-)

Detailed info OR

Detailed info OTDR

OUTPUT, DEC HL, DEC B, REPEAT IF B>0

Detailed info **OTIR**

OUTPUT, INC HL, DEC B, REPEAT IF B>0

Detailed info **OUT**

OUTPUT TO PORT

Detailed info **OUTD**

OUTPUT, DEC HL, DEC B

Detailed info **OUTI**

OUTPUT, INC HL, DEC B

Detailed info **POP**

POP FROM STACK

Detailed info **PUSH**

PUSH INTO STACK

Detailed info **RES**

RESET BIT

Detailed info **RET**

RETURN FROM SUB ROUTINE

Detailed info **RETI**

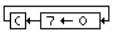
RETURN FROM INTERRUPT

Detailed info **RETN**

RETURN FROM NON MASKABEL INTERRUPT

Detailed info **RL**

Instruction Format:



ASM: RL B

OBJ: CBh, 10h

Tstates : 8.

ASM: RL C

OBJ: CBh, 11h

Tstates : 8.

ASM: RL D

OBJ: CBh, 12h

Tstates : 8.

ASM: RL E

OBJ: CBh, 13h

Tstates : 8.

ASM: RL H

OBJ: CBh, 14h

Tstates : 8.

ASM: RL L

OBJ: CBh, 15h

Tstates : 8.

ASM: RL A

OBJ: CBh, 17h

Tstates : 8.

ASM: RL (HL)

OBJ: CBh, 16h

Tstates : 15.

ASM: RL (IX+D)

OBJ: DDh, CBh, XX, 16h

Tstates : 23.

ASM: RL (IY+D)

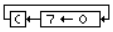
OBJ: FDh, CBh, XX, 16h

Tstates : 23.

XX is the offset value.

Detailed info RLA

Instruction Format:



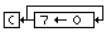
ASM: RLA

OBJ: 17h

Tstates : 4.

Detailed info RLC

Instruction Format:



ASM: RLC B

OBJ: CBh, 00h

Tstates : 8.

ASM: RLC C

OBJ: CBh, 01h

Tstates : 8.

ASM: RLC D

OBJ: CBh, 02h

Tstates : 8.

ASM: RLC E

OBJ: CBh, 03h

Tstates : 8.

ASM: RLC H

OBJ: CBh, 04h

Tstates : 8.

ASM: RLC L

OBJ: CBh, 05h

Tstates : 8.

ASM: RLC A

OBJ: CBh, 07h

Tstates : 8.

ASM: RLC (HL)

OBJ: CBh, 06h

Tstates : 15.

ASM: RLC (IX+D)

OBJ: DDh, CBh, XX, 06h

Tstates : 23.

ASM: RLC (IY+D)

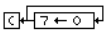
OBJ: FDh, CBh, XX, 06h

Tstates : 23.

XX is the offset value.

Detailed info **RLCA**

Instruction Format:



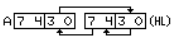
ASM: RLCA

OBJ: 07h

Tstates : 4.

Detailed info **RLD**

Instruction Format:



ASM: RLD

OBJ: EDh, 6Fh

Tstates : 18.

Detailed info **RR**

Instruction Format:

ASM: RR B

OBJ: CBh, 18h

Tstates : 8.

ASM: RR C

OBJ: CBh, 19h

Tstates : 8.

ASM: RR D

OBJ: CBh, 1Ah

Tstates : 8.

ASM: RR E

OBJ: CBh, 1Bh

Tstates : 8.

ASM: RR H

OBJ: CBh, 1Ch

Tstates : 8.

ASM: RR L

OBJ: CBh, 1Dh

Tstates : 8.

ASM: RR A

OBJ: CBh, 1Fh

Tstates : 8.

ASM: RR (HL)

OBJ: CBh, 1Eh

Tstates : 15.

ASM: RR (IX+D)

OBJ: DDh, CBh, XX, 1Eh

Tstates : 23.

ASM: RR (IY+D)

OBJ: FDh, CBh, XX, 1Eh

Tstates : 23.

XX is the offset value.

Detailed info RRA

Instruction Format:

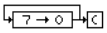
ASM: RRA

OBJ: 1Fh

Tstates : 4.

Detailed info RRC

Instruction Format:



ASM: RRC B

OBJ: CBh, 08h

Tstates : 8.

ASM: RRC C

OBJ: CBh, 09h

Tstates : 8.

ASM: RRC D

OBJ: CBh, 0Ah

Tstates : 8.

ASM: RRC E

OBJ: CBh, 0Bh

Tstates : 8.

ASM: RRC H

OBJ: CBh, 0Ch

Tstates : 8.

ASM: RRC L

OBJ: CBh, 0Dh

Tstates : 8.

ASM: RRC A

OBJ: CBh, 0Fh

Tstates : 8.

ASM: RRC (HL)

OBJ: CBh, 0Eh

Tstates : 15.

ASM: RRC (IX+D)

OBJ: DDh, CBh, XX, 0Eh

Tstates : 23.

ASM: RRC (IY+D)

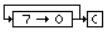
OBJ: FDh, CBh, XX, 0Eh

Tstates : 23.

XX is the offset value.

Detailed info RRCA

Instruction Format:



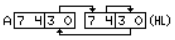
ASM: RRCA

OBJ: 0Fh

Tstates : 4.

Detailed info RRD

Instruction Format:



ROTATE RIGHT DIGIT

ASM: RRD

OBJ: EDh, 67h

Tstates : 18.

Detailed info RST

Instruction Format:

OPCODE	CALL ADDRESS	MNEMONIC
C7h	0000h	RST 0
CFh	0008h	RST 8
D7h	0010h	RST 16
DFh	0018h	RST 24
E7h	0020h	RST 32
EFh	0028h	RST 40
F7h	0030h	RST 48
FFh	0038h	RST 56

CYCLES

Detailed info SBC

SUBTRACT WITH CARRY

Detailed info SCF

Opperation: CY = 1

Instruction Format:

Opcode: SCF

OBJ: 37h

Description:

The Carry flag in the flags register is set (1).

Tstates: 4.

Flags:

S: Not affected.

Z: Not affected.

H: Reset

P/V: Not affected.

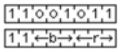
N: Reset.

C: Set

Detailed info SET b, r

Operation: rb = 1

Instruction Format:



CBh

Bit b in register r is set. Operands b and r are specified as follows:

Bit	b	Register	r
0	000	B	000
1	001	C	001
2	010	D	010
3	011	E	011
4	100	H	100
5	101	L	101
6	110	A	111
7	111		

Tstates: 8.

Flags: None Affected

Example:

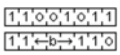
```
ASM:    SET 3,D
```

OBJ: CBh, DAh

Detailed info SET b, (HL)

Operation: (HL)b = 1

Instruction Format:


CBh

Bit b in the memory location addressed by the contents of register pair HL is set. Operands b is specified as follows:

Bit	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Tstates: 15.

Flags: None Affected

Example:

ASM: SET 5,(HL)

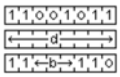
OBJ: CBh, EEh

Detailed info SET b, (IX+d)

Operation: (IX+d)b = 1

Instruction Format:


DDh


CBh

Bit b is set, in the memory location addressed by the sum of the contents of register IX and the two's complement integer d. (d = adr. offset)

Operands b is specified as follows:

Bit	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Tstates: 23.

Flags: None Affected

Example:

ASM: SET 0,(IX+3h)

OBJ: DDh, CBh, 03h, C6h

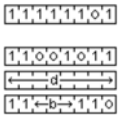
If the contents of index Register IX are 2000h, then bit 0 in memory location 2003h will be 1 (set).

Bit 0 is the least significant bit)

Detailed info SET b, (IY+d)

Operation: (IY+d)b = 1

Instruction Format:



Bit b is set, in the memory location addressed by the sum of the contents of register IY and the two's complement integer d. (d = adr. offset)

Operands b is specified as follows:

Bit	b
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Tstates: 23.

Flags: None Affected

Example:

ASM: SET 0,(IY+47h)

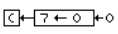
OBJ: DDh, CBh, 47h, C6h

If the contents of index Register IX are 2000h, then bit 0 in memory location 2047h will be 1 (set).

Bit 0 is the least significant bit)

Detailed info SLA

Instruction Format:



ASM: SLA B

OBJ: CBh, 20h

Tstates : 8.

ASM: SLA C

OBJ: CBh, 21h

Tstates : 8.

ASM: SLA D

OBJ: CBh, 22h

Tstates : 8.

ASM: SLA E

OBJ: CBh, 23h

Tstates : 8.

ASM: SLA H

OBJ: CBh, 24h

Tstates : 8.

ASM: SLA L

OBJ: CBh, 25h

Tstates : 8.

ASM: SLA A

OBJ: CBh, 27h

Tstates : 8.

ASM: SLA (HL)

OBJ: CBh, 26h

Tstates : 15.

ASM: SLA (IX+D)

OBJ: DDh, CBh, XX, 26h

Tstates : 23.

ASM: SLA (IY+D)

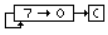
OBJ: FDh, CBh, XX, 26h

Tstates : 23.

XX is the offset value.

Detailed info SRA

Instruction Format:



ASM: SRA B

OBJ: CBh, 28h

Tstates : 8.

ASM: SRA C

OBJ: CBh, 29h

Tstates : 8.

ASM: SRA D

OBJ: CBh, 2Ah

Tstates : 8.

ASM: SRA E

OBJ: CBh, 2Bh

Tstates : 8.

ASM: SRA H

OBJ: CBh, 2Ch

Tstates : 8.

ASM: SRA L

OBJ: CBh, 2Dh

Tstates : 8.

ASM: SRA A

OBJ: CBh, 2Fh

Tstates : 8.

ASM: SRA (HL)

OBJ: CBh, 2Eh

Tstates : 15.

ASM: SRA (IX+D)

OBJ: DDh, CBh, XX, 2Eh

Tstates : 23.

ASM: SRA (IY+D)

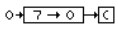
OBJ: FDh, CBh, XX, 2Eh

Tstates : 23.

XX is the offset value.

Detailed info SRL

Instruction Format:



ASM: SRL B

OBJ: CBh, 38h

Tstates : 8.

ASM: SRL C

OBJ: CBh, 39h

Tstates : 8.

ASM: SRL D

OBJ: CBh, 3Ah

Tstates : 8.

ASM: SRL E

OBJ: CBh, 3Bh

Tstates : 8.

ASM: SRL H

OBJ: CBh, 3Ch

Tstates : 8.

ASM: SRL L

OBJ: CBh, 3Dh

Tstates : 8.

ASM: SRL A

OBJ: CBh, 3Fh

Tstates : 8.

ASM: SRL (HL)

OBJ: CBh, 3Eh

Tstates : 15.

ASM: SRL (IX+D)

OBJ: DDh, CBh, XX, 3Eh

Tstates : 23.

ASM: SRL (IY+D)

OBJ: FDh, CBh, XX, 3Eh

Tstates : 23.

XX is the offset value.

Detailed info SUB

Operation: A = A - s

The s operand is subtracted from the contents of the accumulator, and the result is stored in the Accumulator.

ASM: SUB B

OBJ: 90h

Tstates : 4.

ASM: SUB C

OBJ: 91h

Tstates : 4.

ASM: SUB D

OBJ: 92h

Tstates : 4.

ASM: SUB E

OBJ: 93h

Tstates : 4.

ASM: SUB H

OBJ: 94h

Tstates : 4.

ASM: SUB L

OBJ: 95h

Tstates : 4.

ASM: SUB A

OBJ: 97h

Tstates : 4.

ASM: SUB n

OBJ: D6h, n

Tstates : 7.

ASM: SUB (HL)

OBJ: 96h

Tstates : 7.

ASM: SUB (IX+d)

OBJ: DDh, 96h, d

Tstates : 19.

ASM: SUB (IY+d)

OBJ: FDh, 96h, d

Tstates : 19.

Flags:

S: Set if result is negative, reset otherwise.

Z: Set if result is Zero, reset otherwise.

H: Set if borrow from Bit4, reset otherwise.

P/V: Set if overflow, reset otherwise.

N: Reset.

C: Set if borrow, reset otherwise.

Example: If Accumulator contains 29h, and register D contains 11h, after the execution of:

ASM: SUB D

OBJ: 92h

The Accumulator will contain 18h.

Detailed info XOR

Operation: A = A xor s

A logical exclusive-OR opperation is performed between the byte specified by the s opperand
and the byte contained in the Accumulator, the result is stored in the Accumulator.

ASM: XOR B

OBJ: A8h

Tstates : 4.

ASM: XOR C

OBJ: A9h

Tstates : 4.

ASM: XOR D

OBJ: AAh

Tstates : 4.

ASM: XOR E

OBJ: ABh

Tstates : 4.

ASM: XOR H

OBJ: ACh

Tstates : 4.

ASM: XOR L

OBJ: ADh

Tstates : 4.

ASM: XOR A

OBJ: AFh

Tstates : 4.

ASM: XOR n

OBJ: EEh, n

Tstates : 7.

ASM: XOR (HL)

OBJ: AEh

Tstates : 7.

ASM: XOR (IX+d)

OBJ: DDh, AEh, d

Tstates : 19.

ASM: XOR (IY+d)

OBJ: FDh, AEh, d

Tstates : 19.

Flags:

S: Set if result is negative, reset otherwise.

Z: Set if result is Zero, reset otherwise.

H: Reset.

P/V: Set if parity even, reset otherwise.

N: Reset.

C: Reset.

Example: If the Accumulator contains 96h (10010110),
and register D contains 5Dh (01011101) , after the execution of:

ASM: XOR D

OBJ: AAh

The Accumulator will contain CBh (11001011).