AltairDuino

Switch Functions

AUX1 DOWN

SW 7-0	Function
0000000	Print this list to serial interface
0000001	**Calculator**
0000010	**Kill-the-Bit**
0000011	**Pong** on front panel
0000100	**Pong** on serial terminal
0000101	**4K BASIC** (Sw 11: Down=SIO, Up=2SIO)
0000110	**MITS 16K BASIC** (Sw 15-12: 0000=2SIO, 0010=SIO) Memory size? **Press Enter** Lineprinter? **Answer 0**
0000111	**MITS Programming System** (Ed, Asm, Debug) Sw11: Down=SIO, Up=2SIO
0001000	**Combo Disk Boot Loader ROM**
0001001	**ALTAIR Turnkey Monitor**
0001010	**Music: "Daisy, Daisy…"**
0001011	**8080 CPU Diagnostic**
0001100	8080 **CPU Exerciser**
0001101	**Music System**
0001110	**Hard-Disk boot ROM v2.0 (mount first)**
0001111	**Enhanced Multi-Boot Loader V3.0** Sw 10-8 selects boot device: 000=2SIO; 010=SIO; 011=ACR; 110=2SIO, port 2
0010000	**Tarbell disk boot loader**
0100000	**Read Intel Hex Data from Primary Host Interface**
11xxxxxx	Save 256-byte memory page selected by Sw 15-8 to file #xxxxxx
10xxxxxx	Load 256-byte memory page selected by Sw 15-8 from file #xxxxxxx

AUX1 UP

STOP Sw	Function
Up	Configuration Editor
Down	Run program set in Config Ed (#3 set to HD boot)

AUX2 DOWN

MOUNT DISKS

MITS: 0001nnnnDDDDDDDD n=Drive, D=Disk
Tarbell: 0101xxnnDDDDDDDD n=Drive, D=Disk
Hard Drive: 0001uuppDDDDDDDD u=Unit, p=Platter, D=Disk

Play Back Captured Data or Demo Programs

SW	Function
15: Down	Use host primary serial output last used
15: Up	Use serial device selected on Sw 14-13
14-13:	00: 88-SIO (port 0x00/0x01) 01: 88-ACR (port 0x06/0x07) cassette 10: 88-2SIO, serial 1 (port 0x10/0x11) 11: 88-2SIO, serial 2 (port 0x12/0x13)
08: Down	Play back example program selected on Sw 7-0
08: Up	Play back captured data in file on Sw 7-0
07: Down	(Sw 8 down) Select **BASIC** examples
07: Up	(Sw 8 down) Select **Assembler** examples
6-0: 000000	List available examples
6-0: xxxxxx	AUX2 Down to transmit **example** xxxxxx AUX2 Down again to cancel
7-0:Xxxxxxxx	(Sw8 up) play back **data** from xxxxxxxx

AUX2 UP

UNMOUNT DISKS

MITS: 0001nnnnDDDDDDDD n=Drive, D=Disk
Tarbell: 0101xxnnDDDDDDDD n=Drive, D=Disk
Hard Drive: 0001uuppDDDDDDDD u=Unit, p=Platter, D=Disk

Capture Data (set serial device)

SW	Function
15: Down	Use host primary serial output last used
15: Up	Use serial device selected on Sw 14-13
14-13:	00: 88-SIO (port 0x00/0x01) 01: 88-ACR (port 0x06/0x07) cassette 10: 88-2SIO, serial 1 (port 0x10/0x11) 11: 88-2SIO, serial 2 (port 0x12/0x13)
12: 0	
7-0:xxxxxxxx	Store data in file xxxxxxxx

AltairDuino Disks

Hard disk images:

0001) HDSK01.DSK: Altair Hard Disk BASIC
0010) HDSK02.DSK: Altair Accounting System
0011) HDSK03.DSK: Mike Douglas' 88-HDSK CP/M
0100) HDSK04.DSK: Infocom Adventures CP/M

MITS Floppy disk images:

00001) DISK01.DSK: CP/M (63k)
00010) DISK02.DSK: ALTAIR DOS 1.0
00011) DISK03.DSK: ALTAIR Disk Basic
00100) DISK04.DSK: ALTAIR Disk Basic programs
00101) DISK05.DSK: Games (CP/M programs)
00110) DISK06.DSK: SuperCalc II (CP/M program)
00111) DISK07.DSK: WordStar (CP/M program)
01000) DISK08.DSK: Zork (CP/M game)
01001) DISK09.DSK: Time Sharing Basic V1.1
01010) DISK0A.DSK: Time Sharing Basic V2
01011) DISK0B.DSK: Time Sharing Basic V2 programs
01100) DISK0C.DSK: Altair Mini-Disk Basic
01101) DISK0D.DSK: Altair Mini-Disk Basic programs
01110) DISK0E.DSK: Altair Mini-Disk DOS
01111) DISK0F.DSK: Altair Mini-Disk DOS programs
10000) DISK10.DSK: Dazzler programs (boots CP/M)
10001) DISK11.DSK: VDM-1 programs (boots CP/M)
10010) DISK12.DSK: IMP modem executive (CP/M)

Serial Debugger Functions

Enable Serial Debug and Serial Input in Config Editor
*Requires STANDALONE to be set in config.h

Key	Action
0-9,a-f	Toggle SW0-15*
/	Prompt for value to set SW0-15*
r	Run
o	Stop
t	Step
R	Reset
!	Hard reset (STOP+RESET)
X/x	Examine/examine next
P/p	Deposit/deposit next
U/u	AUX1 up/AUX1 down
s	Capture serial data
l	Play back captured data or BASIC example
m	Mount (hard) disk image
Q	Protect
q	Unprotect
>	Run from address
B	Add breakpoint (only if breakpoints enabled
V	Delete last breakpoint
D	Disassemble
M	Dump memory
n	change number system (hex/oct/dec)
C	Enter configuration menu
L	Load a program or data through serial input
H	Load a program in Intel HEX format
h	Dump memory in Intel HEX format through serial

Hard Disk BOOTUP to CP/M

To WiFi: Sw 1-0: 11
To VGA: Sw 1-0: 01
DEPOSIT Up, **POWER** On (loads config)
STOP Down, **AUX1** Up (boots CP/M hard disk in config 3)

CP/M 2.2b for Hard Disk commands

ASM	Load assembler and assemble program
CCP	Allow line editing on command line entry ^C - reboot CP/M
DDT	Load debugger
DUMP	Dump contents of a file in Hex
DIR	Directory listing
ED	Load text editor
ERA	Erase file from current disk
LOAD	Load HEX file format and make executable
MOVCPM	Regenerate CP/M for a memory size
PIP	Load Peripheral Interchange Program PIP <dest>=<src1>,<src2>, …, <srcN>
REN	REName a file, REN fname1=fname2 fname2 is changed to fname1
SAVE n fn	Copy n 256-byte blocks from TPA into file fn
STAT	File storage and device assignment
SUBMIT	Execute a file of commands
SYSGEN	Create new CP/M Disk
USER n	Change user prefix for directory entries
TYPE fn	Display contents of fn on the console

CP/M Editor (ED)

n**A**	Appends next n source lines from the source file
+-**B**	Move CP to begin (+) or end (-) of buffer
+-n**C**	Move CP +/- n chars
+-n**D**	Delete n chars (+) ahead or (-) behind CP
E	Ends edit - copy all source to temporary
n**Fs**<cr>	Search for nth occurrence of string s in buffer
H	Move to head of new file, does E & opens new file
I	Begin inserting text
n**Js1^Zs2^s3**	Find s1, insert s2, delete chars to s3
+-n**K**	Delete +/- n lines of text from CP
+-n**L**	Move CP to start of line +/- lines from CP
n**M**	Macro definition
O	Reverts to original file
+-n**P**	Move and print pages
Q	Quit without saving
R	Read library file
n**Ss1^Zs2**<cr>	Substitute s2 for s1 n times or end of buffer
+-n**T**	Type current line +n lines before (-)/after (+)
U	Translate lower to upper case
V	Verify line numbers/(OV) show remaining space
n**W**	Writes n lines of buffer to the temporary file
^Z	Return to ED command mode
+-n	Same as +-nLT

MBasic Commands & Statements

Command	Description
AUTO	Generate auto line number after each <CR> AUTO [<line number>[,<increment>]]
CALL	Call assembly language subroutine CALL <variable name>[(<arg list>)]
CHAIN	Call a program and pass variables **CHAIN [MERGE] <filename>[,[<line #>] [,ALL]** **[,DELETE<range>]]**
CLEAR	Clear variables to zero/null, set memory end and amount of stack space. CLEAR [,[<exp1>][,<exp2>]]
CLOAD[?|*]	Load program from cassette **? - verify against prgm in memory** * - load array into variable
CLOSE	Close disk file CLOSE [[#]<file number>[,[#]<file number>…>]]
COMMON	Pass variables to CHAINed program **COMMON <list of variables>**
CONT	Continue program after Control-C, STOP or END
CSAVE	Save program to cassette **CSAVE <filename> (only first char is used)**
DATA	Store constants accessed by READ statement **DATA <list of constants>**
DEF FN	Define a user specified function **DEF FN<name>[(<params>)]=<function def>**
DEF<type>	Declare variable types beginning with a range of letters as a specific type **DEF<type> <range of letters>** **Where type is INT, SNG, DBL, or STR**
DEF USR	Specify starting address of assembly subroutine DEF USR[<digit>]=<integer expression>
DELETE	Delete program lines **DELETE [<line number>][-<line number>]**
DIM	Specify maximum size of array variable **DIM <list of variables>**
EDIT	Edit a specified line **EDIT <line number>**
END	Terminate program execution.
ERASE	Eliminate arrays from a program. **ERASE <list of array variables>**
ERR/ERL	Variable containing error code and line number **IF ERR = <error code> THEN ….**
ERROR	Simulate error code or trap for error **ON ERROR GOTO … Trap error** **ERROR 15 Generate error code 15**
FIELD	Allocate space for variables in a file buffer **FIELD [#]<file number>, <field width> AS** **<string variable>**
FOR..NEXT	Loop **FOR <var> = X to Y [STEP z]** **NEXT [<var>][,<var>…]**
GET	Read a record from a random disk file **GET [#]<file number>[,<record number>]**
GOSUB	Branch to subroutine **GOSUB <line number>** … **RETURN**
GOTO	Unconditional Branch **GOTO <line number>**
IF/THEN/ELSE	Conditional branch **IF <exp> THEN <statement(s)> | <line #>** **[ELSE <statement(s)> | <line #>**
INPUT	Input from terminal device **INPUT[;][<"prompt">;]<list of vars>**
INPUT#	Read data from sequential file **INPUT#<file#>, <vars>**

MBasic Commands & Statements

KILL	Delete file from disk. **KILL <filename>**
LET	Assign value to variable **[LET] <var>=<exp>**
LINE INPUT	Input up to 254 chars without delimiters **LINE INPUT[;][<"prompt">;]<string var>**
LINE INPUT#	Read entire line from sequential file **LINE INPUT#<file#>, <string var>**
LIST	List all or part of program in memory **LIST [<line#>[-<line#>]]**
LLIST	List current program to line printer **LLIST [<line#>[-<line#>]]**
LOAD	Load file from disk, .BAS extension assumed **LOAD <filename>[,R]** **R option runs program after load**
LPRINT	Print data on the line printer device **LPRINT [<list of expr>]** **LPRINT USING <string exp>;<list of expr>**
LSET/RSET	**Move data from memory and a file buffer in Preparation for a PUT statement** **LSET left-justifies, RSET right-justifies** **LSET <str var>=<str expr>** **RSET <str var>=<str expr>**
MERGE	**Merge a specified disk file with pgm in mem** **MERGE <filename>**
MID$	Replace part of one string with another **MID$(<str exp1>,n[,m])=<str exp2>**
NAME	Change name of a disk file **NAME <old fname> AS <new fname>**
NEW	Delete current program, clear vars
NULL	Set number of nulls to be printed after NewLine
ON ERROR	Enable error trapping **ON ERROR GOTO <line #>**
ON..GOSUB	Branch on expression value
ON..GOTO	ON <expr> GOTO/GOSUB <list of line #s>
OPEN	Open a disk file for I/O OPEN <mode>,#<file#>,<filename>[,<reclen>]
OPTION BASE	Specify min value of an array index OPTION BASE n (n= 0 or 1)
OUT I,J	Send byte J to machine port I OUT <port>, <byte>
POKE	Write a byte to a memory location POKE <memory locn>,<byte>
PRINT[#] PRINT USING	Print on the terminal PRINT <list of expr>
PUT	Write record from random buffer to disk PUT[#]<file#>,[,<record #>]
RANDOMIZE	Reseed random number generator RANDOMIZE [<expr>]
READ	Read values from DATA statement READ <vars>
REM or ‘	Remark
RENUM	RENUM [[<new #>][,[<old #>][,<increment>]
RESTORE	Reset Data statements to a specified point RESTORE [<line#>]
RESUME	Continue execution after error RESUME [0 | NEXT | <line#>]
RUN	RUN <filename> [,R] - R doesn’t clear vars RUN [<line#>] start execution at line #
SAVE	Save program to disk SAVE <filename>[,A | ,P] A: ASCII FORMAT, P: Encoded format
STOP E	Terminate program execution
SWAP	Exchange value of two variables SWAP <var1>,<var2>

MBasic Commands & Statements

SYSTEM	Return to CP/M
TRON/TROFF	Trace execution for debugging
WAIT	Suspend execution until pattern on input port WAIT <port#>, I[,J] J if specified, is XORd with I
WHILE/WEND	Loop while condition is true WHILE <expr>… WEND
WIDTH	Set line width for line printer WIDTH [LPRINT] <integer expr
WRITE[#]	Output to terminal or file with commas WRITE [<list of expr>]

MBASIC FUNCTIONS

Function	Description
ABS(X)	Absolute Value
ASC(X$)	ASCII Numerical value of a character
ATN(X)	Arctangent of X in radians
CDBL(X)	Convert X to double-precision
CHR$(I)	Convert ASCII value I to character
CINT(X)	Convert X to integer
COS(X)	Cosine of X in radians
CSNG(X)	Convert X to single-precision
CVI,CVS,CVD	Convert string to Integer, Single, Double
EOF(file#)	Returns true if end of seq file reached
EXP(X)	Returns e^X
FIX(X)	Returns integer portion of X (does not round)
FRE(X)	Returns number of free bytes
FRE(“”)	Forces garbage collection
HEX$(X)	Returns string with hex value of X
INP(I)	Returns byte read from port I
INPUT$(X)	INPUT(X[,[#]Y]) Returns string of X chars from terminal or file
INSTR	INSTR([I,],X$,Y$) - returns pos where Y$ is found in X$, starting from position I
INT(X)	Returns largest integer <=X
LEFT$(X$,I)	Returns string with leftmost I chars of X$
LEN(X$)	Returns number of chars in X$
LOC(<file>)	Returns next record to be used by GET/PUT
LOG(X)	Returns natural log of X
LPOS(X)	Returns current position of line printer head
MID$(X$,I,J)	Returns string of J chars from X from pos I
MKI$,MKS$, MKD$	Convert numeric to string. MKI: 2-bytes, MKS, 4-bytes, MKD: 8-bytes
OCT$(X)	Returns string with Octal value of X
PEEK(I)	Returns byte read from memory loc I
POS(I)	Returns current cursor position
RIGHT$(X$,I)	Returns I rightmost chars of X$
RND[(X)]	Returns random number between 0 and 1
SGN(X)	If X>0 returns 1, =0 returns 0, <0, returns -1
SIN(X)	Returns sine of X in radians
SPACE$(X)	Returns a string of X spaces
SPC(I)	Prints I spaces on terminal
SQR(X)	Returns square root of X
STR$(X)	Returns string representation of X
STRING$(I,J)	Returns string of len I of chars with ASCII J
TAB(I)	Spaces to position I on terminal
TAN(X)	Returns tangent of X in radians
USR	Calls user assembler subroutine
VAL(X$)	returns numeric value of X$
VARPTR(vn)	Returns memory address of variable name OR Starting address of disk I/O buffer

Control Characters

KEY	Function
^A	Edit mode on current line
^C	Interrupt program execution
^O	Halts output, execution continues
^R	Retypes current line
^S	Suspend execution
^Q	Resume execution
^U	Delete current line
ESC	Exit edit mode

8080 Instruction Set Reference

Conventions in instruction source:

```
        D = Destination register (8 bit)
        S = Source register (8 bit)
        RP = Register pair (16 bit)
        # = 8 or 16 bit immediate operand
        a = 16 bit Memory address
        p = 8 bit port address
        ccc = Conditional
```

Conventions in instruction encoding

```
        db  = Data byte (8 bit)
        lb  = Low byte of 16 bit value
        hb  = High byte of 16 bit value
        pa  = Port address (8 bit)
```

Dest and Source reg fields:

```
        111=A (Accumulator)
        000=B
        001=C
        010=D
        011=E
        100=H
        101=L
        110=M (Memory reference through address in H:L)
```

Register pair 'RP' fields:

```
        00=BC (B:C as 16 bit register)
        01=DE (D:E as 16 bit register)
        10=HL (H:L as 16 bit register)
        11=SP (Stack pointer,
              refers to PSW (FLAGS:A) for PUSH/POP)
```

Condition code 'CCC' fields: (FLAGS: S Z x A x P x C)

```
        000=NZ (Zero flag not set)
        001=Z (Zero flag set)
        010=NC (Carry flag not set)
        011=C (Carry flag set)
        100=PO (Parity flag not set - ODD)
        101=PE (Parity flag set - EVEN)
        110=P (Sign flag not set - POSITIVE)
        111=M (Sign flag set - MINUS)
```

CP/M Assembler parameters:

ASM XYZ.p1p2p3

P1: disk letter containing source
P2: disk letter for hex file or Z:none
P3: disk letter for print or X:console/Z:none

Assembler directives:

Directive	Meaning
ORG	set the program or data origin
END	end program, optional start address
EQU	numeric equate
SET	numeric set
IF	begin conditional assembly
ENDIF	end of conditional assembly
DB	define data bytes
DW	define data words
DS	define data storage area

Assembler Error Codes:

Code	Meaning
D	Data error – element cannot be placed there
E	Expression Error – ill formed expression
L	Label Error – can't appear here or duplicate
N	Not implemented feature
O	Overflow – too complicated to be computed
P	Phase error – label not consistent between passes
R	Register error – reg val not compatible with op code
S	Syntax error
Y	Value error – operand improperly formed

SESSION:
A>ASM XYZ.AAA – generates PRN and HEX files on A
A>SAVE n XYZ.COM – saves n 256-byte pages to disk file

8080 Instructions by Mnemonic

Inst	Encoding	Flags	Description
ACI #	11001110 db	ZSCPA	Add immediate to A w/carry*
ADC S	10001SSS	ZSCPA	Add register to A w/carry
ADD S	10000SSS	ZSPCA	Add register to A
ADI #	11000110 db	ZSCPA	Add immediate to A
ANA S	10100SSS	ZSCPA	AND register with A
ANI #	11100110 db	ZSPCA	AND immediate with A
CALL a	11001101 lb hb	-	Uncond. subroutine call
Cccc a	11CCC100 lb hb	-	Conditional subroutine call
CMA	00101111	-	Complement A
CMC	00111111	C	Complement Carry flag
CMP S	10111SSS	ZSPCA	Compare register with A
CPI #	11111110	ZSPCA	Compare immediate with A
DAA	00100111	ZSPCA	Decimal Adjust accumulator
DAD RP	00RP1001	C	Add register pair to HL*
DCR D	00DDD101	ZSPA	Decrement register*
DCX RP	00RP1011	-	Decrement register pair
DI	11110011	-	Disable interrupts
EI	11111011	-	Enable interrupts
HLT	01110110	-	Halt processor
IN p	11011011 pa	-	Read input port into A
INR D	00DDD100	ZSPA	Increment register
INX RP	00RP0011	-	Increment register pair
Jccc a	11CCC010 lb hb	-	Conditional jump*
JMP a	11000011 lb hb	-	Unconditional jump*
LDA a	00111010 lb hb	-	Load A from memory
LDAX RP	00RP1010 *1	-	Load indirect via BC or DE
LHLD a	00101010 lb hb	-	Load H:L from memory*
LXI RP,#	00RP0001 lb hb	-	Load register pair immed*
MOV D,S	01DDDSSS	-	Move register to register*
MVI D,#	00DDD110 db	-	Move immediate to register*
NOP	00000000	-	No operation
ORA S	10110SSS	ZSPCA	OR register with A
ORI #	11110110	ZSPCA	OR immediate with A
OUT p	11010011 pa	-	Write A to output port
PCHL	11101001	-	Jump to address in H:L
POP RP	11RP0001 *2	*2	Pop register pair from stack
PUSH RP	11RP0101 *2	-	Push register pair on stack
RAL	00010111	C	Rotate A left through carry*
RAR	00011111	C	Rotate A right through carry
Rccc	11CCC000	-	Conditional return from sub
RET	11001001	-	Unconditional ret from sub
RLC	00000111	C	Rotate A left
RRC	00001111	C	Rotate A right
RST n	11NNN111	-	Restart (Call n*8)
SBB S	10011SSS	ZSCPA	Subtract reg from A w/borrow
SBI #	11011110 db	ZSCPA	Subtr immed from A w/ borrow
SHLD a	00100010 lb hb	-	Store H:L to memory*
SPHL	11111001	-	Set SP to content of H:L
STA a	00110010 lb hb	-	Store A to memory
STAX RP	00RP0010 *1	-	Store indirect via BC or DE
STC	00110111	C	Set Carry flag
SUB S	10010SSS	ZSCPA	Subtract register from A
SUI #	11010110 db	ZSCPA	Subtract immediate from A
XCHG	11101011	-	Exchange DE and HL content
XRA S	10101SSS	ZSPCA	Exclusive OR register with A
XRI #	11101110 db	ZSPCA	Exclusive OR immediate w/A
XTHL	11100011		Swap H:L with top of stack

*1 = Only RP=00(BC) and 01(DE) are allowed for LDAX/STAX
*2 = RP=11 refers to PSW for PUSH/POP (cannot push/pop SP).
When PSW is POP'd, ALL flags are affected.

8080 Instructions by Encoding

Encoding	Inst	Flags	Description
00000000	NOP	-	No operation
00000111	RLC	C	Rotate A left
00001111	RRC	C	Rotate A right
00010111	RAL	C	Rotate A left through carry*
00011111	RAR	C	Rotate A right through carry
00100010 lb hb	SHLD a	-	Store H:L to memory*
00100111	DAA	ZSPCA	Decimal Adjust accumulator
00101010 lb hb	LHLD a	-	Load H:L from memory*
00101111	CMA	-	Complement A
00110010 lb hb	STA a	-	Store A to memory
00110111	STC	C	Set Carry flag
00111010 lb hb	LDA a	-	Load A from memory
00111111	CMC	C	Complement Carry flag
00DDD100	INR D	ZSPA	Increment register
00DDD101	DCR D	ZSPA	Decrement register*
00DDD110 db	MVI D,#	-	Move immediate to register*
00RP0001 lb hb	LXI RP,#	-	Load register pair immed*
00RP0010 *1	STAX RP	-	Store indirect through BC/DE
00RP0011	INX RP	-	Increment register pair
00RP1001	DAD RP	C	Add register pair to HL*
00RP1010 *1	LDAX RP	-	Load indirect through BC/DE
00RP1011 D	CX RP	-	Decrement register pair
01110110	HLT	-	Halt processor
01DDDSSS	MOV D,S	-	Move register to register*
10000SSS	ADD S	ZSPCA	Add register to A
10001SSS	ADC S	ZSCPA	Add register to A with carry
10010SSS	SUB S	ZSCPA	Subtract register from A
10011SSS	SBB S	ZSCPA	Subtract reg from A w/borrow
10100SSS	ANA S	ZSCPA	AND register with A
10101SSS	XRA S	ZSPCA	Exclusive OR register with A
10110SSS	ORA S	ZSPCA	OR register with A
10111SSS	CMP S	ZSPCA	Compare register with A
11000011 lb hb	JMP a	-	Unconditional jump*
11000110 db	ADI #	ZSCPA	Add immediate to A
11001001	RET	-	Uncond return from sub
11001101 lb hb	CALL a	-	Uncond subroutine call
11001110 db	ACI #	ZSCPA	Add immed to A w/carry*
11010011 pa	OUT p	-	Write A to output port
11010110 db	SUI #	ZSCPA	Subtract immediate from A
11011011 pa	IN p	-	Read input port into A
11011110 db	SBI #	ZSCPA	Sub immed from A w/borrow
11100011	XTHL	-	Swap H:L with top of stack
11100110 db	ANI #	ZSPCA	AND immediate with A
11101001	PCHL	-	Jump to address in H:L
11101011	XCHG	-	Exchange DE and HL content
11101110 db	XRI #	ZSPCA	Exclusive OR immed with A
11110011	DI	-	Disable interrupts
11110110	ORI #	ZSPCA	OR immediate with A
11111001	SPHL	-	Set SP to content of H:L
11111011	EI	-	Enable interrupts
11111110	CPI #	ZSPCA	Compare immediate with A
11CCC000	Rccc	-	Cond return from subroutine
11CCC010 lb hb	Jccc a	-	Conditional jump*
11CCC100 lb hb	Cccc a	-	Conditional subroutine call
11NNN111	RST n	-	Restart (Call n*8)
11RP0001 *2	POP RP	*	Pop register pair from stack
11RP0101 *2	PUSH RP	-	Push register pair to stack

*1 = Only RP=00(BC) and 01(DE) are allowed for LDAX/STAX
*2 = RP=11 refers to PSW for PUSH/POP (cannot push/pop SP).
When PSW is POP'd, ALL flags are affected.