

Chapter 28 - The Z80 Processor

The Z80 was the dominant rival of the 6502 in the late 1970s and all through the 1980s. It powered the ZX Spectrum, the Amstrad CPC, the MSX, the Sega Master System and Game Gear, and a long list of arcade machines. Intuition Engine runs it natively.

It has the same 64-kilobyte address space as the 6502, the same bank-and-MMIO mapping into the low 32-bit window of the 64-bit Intuition Engine bus, and a much richer instruction set: an alternate register bank, two 16-bit index registers, block-copy and block-search instructions, separate 8-bit port I/O, and three interrupt modes.

28.1 Architecture

28.1.1 Main register set

Register	Width	Pair	Purpose
A	8	with F as AF	Accumulator. Arithmetic and logic.
F	8	with A as AF	Flags: S Z - H - P/V N C
B	8	with C as BC	Counter for block ops; general use
C	8	with B as BC	Port number for IN/OUT (C)
D	8	with E as DE	General purpose
E	8	with D as DE	General purpose
H	8	with L as HL	Most-used pointer register
L	8	with H as HL	Most-used pointer register

28.1.2 Alternate register set

Each register also has a primed shadow: A', F', B', C', D', E', H', L'. The shadows are not separately addressable. You exchange them in two atomic moves:

- EX AF, AF' swaps AF with AF'.
- EXX simultaneously swaps BC/DE/HL with BC'/DE'/HL'.

The conventional use is for fast interrupt response: the handler swaps banks on entry, does its work in the alternate set, and swaps back on exit, never touching the foreground set's contents.

28.1.3 Index, special, and control

Register	Width	Purpose
IX	16	Index register (use with displaced addressing)
IY	16	Index register (use with displaced addressing)
SP	16	Stack pointer
PC	16	Program counter

Register	Width	Purpose
I	8	Interrupt vector page register (used in IM 2)
R	8	Refresh counter - increments on every M1 cycle

28.1.4 Flag register

Bit	Name	Meaning
7	S	Sign - copy of bit 7 of result
6	Z	Zero - set if result was zero
5	-	Undefined (copy of bit 5 of result on most ops)
4	H	Half carry - carry from bit 3 to bit 4
3	-	Undefined (copy of bit 3 of result on most ops)
2	P/V	Parity (logic ops) / overflow (arithmetic)
1	N	Subtract - set after subtraction; used by DAA
0	C	Carry

28.2 The Intuition Engine memory map for the Z80

The Z80 sees the same 64-K layout as the 6502:

Range	Purpose
\$0000-\$1FFF	Free RAM
\$2000-\$3FFF	Bank 1 window (8 KiB)
\$4000-\$5FFF	Bank 2 window (8 KiB)
\$6000-\$7FFF	Bank 3 window (8 KiB)
\$8000-\$BFFF	VRAM bank window (16 KiB)
\$C000-\$EFFF	Free RAM
\$F000-\$FFF9	MMIO window (maps to \$F0000-\$F0FF9)

As on the 6502, the CPU adapter intercepts the bank-control bytes inside the MMIO window before bus translation: BANK1_REG_LO \$F700, _HI \$F701; BANK2_REG \$F702/\$F703; BANK3_REG \$F704/\$F705; VRAM_BANK_REG \$F7F0. Writes to those bytes change the matching banked range. All other addresses in \$F000-\$FFF9 reach the corresponding MMIO register on the low 32-bit window of the 64-bit bus.

There is no direct Z80 \$F700 path to TERM_OUT (\$F0700); that byte is captured by the bank intercept. Code that needs the system terminal can select TERM_IO_BANK into bank 1 with SET_TERMINAL_BANK, then use the Z80 terminal aliases at \$2700-\$27FF; TERM_OUT is \$2700 in that window and resolves to bus \$F0700.

The Z80 also has the architectural 256-port I/O space. Intuition Engine maps useful devices into ports as well as into memory:

Ports	Device
\$A0-\$AD	VGA registers
\$B0-\$B7	Voodoo register and texture gateway
\$D0/\$D1	POKEY select/data
\$D4/\$D5	ANTIC select/data
\$D6/\$D7	GTIA select/data
\$E0/\$E1	SID select/data
\$E4/\$E5	SN76489 data/status
\$F0/\$F1	PSG select/data
\$F2/\$F3	TED select/data
\$FE/\$FD/\$BE/\$FA-\$FC	ULA control and VRAM data port

For select/data pairs, write the register number to the select port, then write or read the data port. The PSG example below uses that exact pattern.

28.3 Addressing modes

Z80 addressing modes:

Mode	Example	Meaning
Immediate	LD A, n	Literal 8-bit
Immediate extended	LD HL, nn	Literal 16-bit
Modified zero page	RST 38h	Call to a fixed 8-byte slot in page 0
Relative	JR e	Signed 8-bit PC offset (used by JR/DJNZ)
Extended	LD A, (nn)	16-bit absolute address
Indexed	LD A, (IX+d)	IX/IY plus signed 8-bit displacement
Register	LD A, B	Register direct
Register indirect	LD A, (HL)	Byte at the address in a register pair
Implied	CCF	No operand
Bit	BIT b, r	Single bit of a register or (HL)

The two index registers IX and IY give the Z80 a true displaced addressing mode that the 6502 lacks.

28.4 Instruction groups

The Z80 has about 158 distinct mnemonics with 694 opcode forms. Appendix G has the full opcode table; this section names the groups and what each does.

28.4.1 8-bit load group

LD r, r' (any 8-bit register to any 8-bit register); LD r, n (immediate); LD r, (HL); LD (HL), r; LD A, (BC); LD A, (DE); LD A, (nn); LD (BC), A; LD (DE), A; LD (nn), A; LD A, I; LD A, R; LD I, A; LD R, A.

28.4.2 16-bit load group

LD rp, nn (pair immediate); LD HL, (nn); LD rp, (nn); LD (nn), HL; LD (nn), rp; LD SP, HL; PUSH rp; POP rp.

28.4.3 Exchange, block transfer, block search

Mnemonic	Operation
EX DE, HL	Swap DE and HL
EX AF, AF'	Swap AF with AF'
EXX	Swap BC/DE/HL with the shadow set
EX (SP), HL	Swap HL with the top word of the stack
LDI/LDIR/LDD/LDDR	Block move (HL to DE, BC bytes)
CPI/CPIR/CPD/CPDR	Block compare (search for A)

LDIR and CPIR repeat until BC reaches zero. LDDR and CPDR are their decrementing counterparts.

28.4.4 8-bit arithmetic and logic

ADD A, r/ADC A, r/SUB r/SBC A, r/AND r/OR r/XOR r/CP r/INC r/DEC r. Each is available with any of the addressing modes that produce an 8-bit operand.

DAA adjusts the accumulator after a BCD operation.

28.4.5 16-bit arithmetic

ADD HL, rp; ADC HL, rp; SBC HL, rp; ADD IX, rp; ADD IY, rp; INC rp; DEC rp.

28.4.6 Rotate and shift

RLCA, RLA, RRCA, RRA operate on A. The CB-prefix group RLC r, RL r, RRC r, RR r, SLA r, SRA r, SRL r, SLL r (SLL is undocumented but supported) operates on any 8-bit register or (HL)/(IX+d)/(IY+d).

28.4.7 Bit set, reset, and test

BIT b, r (set Z from bit b), SET b, r, RES b, r. Bit number b is 0-7; register r is any 8-bit register or (HL)/(IX+d)/(IY+d).

28.4.8 Jump, call, return, restart

Mnemonic	Operation
JP nn	Unconditional absolute jump
JP cc, nn	Conditional absolute jump
JP (HL)/JP (IX)/JP (IY)	Register-indirect jump

Mnemonic	Operation
JR e	Unconditional relative jump (-126 to +129)
JR cc, e	Conditional relative jump (cc in NZ/Z/NC/C)
DJNZ e	Decrement B, branch if non-zero
CALL nn/CALL cc, nn	Call (push PC, jump)
RET/RET cc	Return
RETI	Return from interrupt (signals the device)
RETN	Return from non-maskable interrupt
RST p	Call to a fixed page-0 slot (p in 00h/08h/10h/18h/20h/28h/30h/38h)

The condition codes for JP cc, JR cc, CALL cc, RET cc: NZ, Z, NC, C, PO, PE, P, M.

28.4.9 Port I/O

Mnemonic	Operation
IN A, (n)	Read port n, address bits 8-15 = A
OUT (n), A	Write A to port n
IN r, (C)	Read port BC, set flags from result
OUT (C), r	Write r to port BC
INI/INIR/IND/INDR	Block input (port to memory)
OUTI/OTIR/OUTD/OTDR	Block output (memory to port)

28.4.10 CPU control

Mnemonic	Operation
NOP/HALT	No op / wait for interrupt
DI/EI	Disable / enable interrupts
IM 0/IM 1/IM 2	Set interrupt mode
SCF/CCF	Set / complement carry
CPL	Ones-complement of A
NEG	Twos-complement of A

28.5 Interrupts

The Z80 has three interrupt modes:

- **IM 0** - the interrupting device puts an 8-bit opcode on the data bus. The CPU executes it. The convention is a RST n, giving a fixed 8-slot vector.
- **IM 1** - every IRQ is handled as RST 38h. The vector is the handler entry at \$0038. This is the mode used by the ZX Spectrum.

- **IM 2** - the device supplies a vector byte. The CPU forms the address $(I \ll 8) | \text{vector}$ and reads a 16-bit handler pointer from there. This gives 128 independent vectors.

NMI always jumps to \$0066, ignoring the interrupt mode and the global enable flag.

Two enable flip-flops IFF1 and IFF2 track maskable-interrupt state across NMI handlers: NMI copies IFF1 to IFF2 and clears IFF1; RETN copies IFF2 back into IFF1. The ordinary handler uses EI and RETI.

28.6 Undocumented opcodes

The Z80 silicon decodes a number of opcodes that were left undocumented on the original machines. Intuition Engine implements the common ones, including:

- The undocumented SLL r (shift left logical, alias SL1).
- IXH, IXL, IYH, IYL as separate 8-bit halves of the index registers, accepted as operands by ordinary LD, ADD, SUB, etc.
- The ED-prefix block IN F, (C) and OUT (C), 0.

Appendix G has the complete listing.

28.7 A small example

This Z80 byte-entry program plays a three-voice chord on the AY-style PSG. It uses ordinary memory for AUDIO_CTRL and Z80 port I/O for the PSG's register-select/data pair. Unlike the 6502 example in the previous chapter, this one uses HL as a pointer to a small table and B as a loop counter, which is a natural Z80 way to feed a device:

```
(z80)> w 1000 3E 01 32 00 F8 21 17 10 06 0A 7E D3 F0 23 7E D3
(z80)> w 1010 F1 23 10 F6 C3 14 10 07 38 00 71 01 00 02 5F 03
(z80)> w 1020 00 04 47 05 00 08 0F 09 0C 0A 09
(z80)> d 1000 #12
1000: 3E 01          LD A, $01
1002: 32 00 F8      LD ($F800), A
1005: 21 17 10     LD HL, $1017
1008: 06 0A      LD B, $0A
T 100A: 7E      LD A, (HL)
100B: D3 F0     OUT ($F0), A
100D: 23      INC HL
100E: 7E      LD A, (HL)
100F: D3 F1     OUT ($F1), A
1011: 23      INC HL
1012: 10 F6     DJNZ $100A <- LOOP
T 1014: C3 14 10  JP $1014
(z80)> r pc 1000
(z80)> b 1014
(z80)> g
(z80)> m 1017 2
1017: 07 38 00 71 01 00 02 5F 03 00 04 47 05 00 08 0F  .8.q..._...G....
1027: 09 0C 0A 09 00 00 00 00 00 00 00 00 00 00 00 00  .....
(z80)> m FC00 1
FC00: 71 00 5F 00 47 00 00 38 0F 0C 09 00 00 00 00 00  q._.G..8.....
(z80)> bc 1014
```

The first two instructions turn audio on. The next two set up the loop: HL points at the register/value table at \$1017, and B contains \$0A, the number of PSG registers to write. The body of the loop reads one table byte, sends it to port \$F0 as the

PSG register number, advances HL, reads the next table byte, sends it to port \$F1 as the register value, advances HL again, and uses DJNZ to repeat until all ten pairs have been written.

Each instruction in the executable part is one, two, or three bytes:

Address	Bytes	Meaning
\$1000	3E 01	LD A, \$01. Opcode \$3E loads an immediate byte into A; \$01 enables audio.
\$1002	32 00 F8	LD (\$F800), A. Opcode \$32 stores A to an absolute address; the address is low byte first.
\$1005	21 17 10	LD HL, \$1017. HL points to the first table pair. The 16-bit operand is low byte first.
\$1008	06 0A	LD B, \$0A. Ten register/value pairs follow.
\$100A	7E	LD A, (HL). Reads the next PSG register number from the table.
\$100B	D3 F0	OUT (\$F0), A. Port \$F0 is the PSG register-select port.
\$100D	23	INC HL. Advances to the value byte.
\$100E	7E	LD A, (HL). Reads the value for that PSG register.
\$100F	D3 F1	OUT (\$F1), A. Port \$F1 writes the selected PSG register.
\$1011	23	INC HL. Advances to the next register/value pair.
\$1012	10 F6	DJNZ \$100A. Decrement B; if it is not zero, branch back ten bytes to the loop body.
\$1014	C3 14 10	JP \$1014, a self-loop that leaves the PSG chord active.

The bytes at \$1017 are data, not opcodes:

Pair	Meaning
07 38	Register 7, the mixer. \$38 enables tone A, B, and C, and disables noise on all three channels.
00 71, 01 00	Channel A divider \$071.
02 5F, 03 00	Channel B divider \$05F, a higher pitch than channel A.
04 47, 05 00	Channel C divider \$047, higher again.
08 0F, 09 0C, 0A 09	Channel A, B, and C fixed volumes.

Port \$F0 selects a PSG register and port \$F1 writes its data. Registers 0/1, 2/3, and 4/5 hold the three 12-bit tone dividers. Lower divider values give higher pitches. The `m FC00 11` command proves that the PSG register window contains the values from the table after the loop has run.

28.8 ANTIC graphics example

The Z80's block-copy instruction is useful for video setup. This example copies a small ANTIC display list and a single high resolution bitmap row into RAM with LDIR, then uses Z80 port I/O to set the ANTIC and GTIA registers. The display list repeats the same mode-15 row eight times, so the pattern appears as a bright 8-line stripe rather than a single thin line.

The display list is copied to \$2000. It points ANTIC at screen data at \$201B, directly after the display list. GTIA playfield colour 0 becomes the dark background, playfield colour 1 becomes the bright ink, and the background register sets the border colour:

```

(z80)> w 1100 21 46 11 11 00 20 01 43 00 ED B0 3E 00 D3 D6 3E
(z80)> w 1110 02 D3 D7 3E 01 D3 D6 3E CE D3 D7 3E 04 D3 D6 3E
(z80)> w 1120 24 D3 D7 3E 02 D3 D4 3E 00 D3 D5 3E 03 D3 D4 3E
(z80)> w 1130 20 D3 D5 3E 00 D3 D4 3E 22 D3 D5 3E 0E D3 D4 3E
(z80)> w 1140 01 D3 D5 C3 43 11 4F 1B 20 4F 1B 20 4F 1B 20 4F
(z80)> w 1150 1B 20 4F 1B 20 4F 1B 20 4F 1B 20 4F 1B 20 41 18
(z80)> w 1160 20 81 42 24 18 18 24 42 81 FF 00 FF 00 3C 42 81
(z80)> w 1170 42 3C 00 18 3C 7E FF 7E 3C 18 00 55 AA 55 AA F0
(z80)> w 1180 0F F0 0F 99 66 99 66 C3 3C
(z80)> d 1100 #33
1100: 21 46 11          LD HL, $1146
1103: 11 00 20          LD DE, $2000
1106: 01 43 00          LD BC, $0043
1109: ED B0             LDIR
110B: 3E 00             LD A, $00
110D: D3 D6             OUT ($D6), A
110F: 3E 02             LD A, $02
1111: D3 D7             OUT ($D7), A
1113: 3E 01             LD A, $01
1115: D3 D6             OUT ($D6), A
1117: 3E CE             LD A, $CE
1119: D3 D7             OUT ($D7), A
111B: 3E 04             LD A, $04
111D: D3 D6             OUT ($D6), A
111F: 3E 24             LD A, $24
1121: D3 D7             OUT ($D7), A
1123: 3E 02             LD A, $02
1125: D3 D4             OUT ($D4), A
1127: 3E 00             LD A, $00
1129: D3 D5             OUT ($D5), A
112B: 3E 03             LD A, $03
112D: D3 D4             OUT ($D4), A
112F: 3E 20             LD A, $20
1131: D3 D5             OUT ($D5), A
1133: 3E 00             LD A, $00
1135: D3 D4             OUT ($D4), A
1137: 3E 22             LD A, $22
1139: D3 D5             OUT ($D5), A
113B: 3E 0E             LD A, $0E
113D: D3 D4             OUT ($D4), A
113F: 3E 01             LD A, $01
1141: D3 D5             OUT ($D5), A
T 1143: C3 43 11       JP $1143
(z80)> r pc 1100
(z80)> b 1143
(z80)> g
(z80)> m 2000 2
2000: 4F 1B 20 4F 1B 20 4F 1B 20 4F 1B 20 4F 0. 0. 0. 0. 0. 0
2010: 1B 20 4F 1B 20 4F 1B 20 41 18 20 81 42 24 18 18 . 0. 0. A. .B$.
(z80)> m 201B 3
201B: 81 42 24 18 18 24 42 81 FF 00 FF 00 3C 42 81 42 .B$. $B.....<B.B
202B: 3C 00 18 3C 7E FF 7E 3C 18 00 55 AA 55 AA F0 0F <...<~.~<..U.U...
203B: F0 0F 99 66 99 66 C3 3C 00 00 00 00 00 00 00 00 ...f.f.<.....
(z80)> bc 1143

```

The first four instructions perform the copy. HL points at the table inside the program, DE points at \$2000, and BC gives the byte count, \$0043. LDIR copies one byte, advances HL and DE, decrements BC, and repeats until BC reaches zero.

After the copy, the program uses two select/data port pairs:

Ports	Device	Use in this example
\$D6/\$D7	GTIA	Select colour registers 0, 1, and 4, then write their colour bytes.
\$D4/\$D5	ANTIC	Select DLISTL, DLISTH, DMACTL, and ENABLE, then write their values.

The table copied by LDIR is not executed. It is video data:

Bytes	Meaning
4F 1B 20 repeated eight times	Eight display-list entries. \$4F is LMS + mode 15; the following address bytes point each line at \$201B.
41 18 20	JVB, jump and wait for vertical blank, pointing back to the JVB entry at \$2018.
The 40 bytes at \$201B	One high-resolution mode-15 bitmap row. Set bits use playfield colour 1; clear bits use playfield colour 0.

When the breakpoint is reached, ANTIC is enabled with display-list DMA, the display list starts at \$2000, and the first eight visible scanlines show the repeated motif. The m 2000 27 and m 201B 40 commands prove that the display list and bitmap row were copied before the hardware was started.

28.9 What comes next

Chapter 29 covers the M68K, the 16/32-bit processor family that replaced both the 6502 and the Z80 in the late 1980s machines: the Atari ST, the Amiga, and the early Apple Macintosh. Intuition Engine's M68K is MC68020-class, with 32-bit data and address registers and a flat 32-bit client view of the Intuition Engine bus. It sees the low-window MMIO map directly and needs no banking machinery.