Registers

All processors have small memory banks located on the processor itself. These are for high speed data transfer. The processor loads a value from the RAM onto the high-speed data location, manipulates it, and then puts it back in its place in the RAM.
These small, high speed data locations are called registers. They have letter names to distinguish them. Right now, we will learn a few of the most commonly used registers: a, b, c, d, e, h, and l.

a b c d e h l

On the z80 they are 1 byte long (8 bits, 8 1's and 0's); however, they can be put into pairs (bc, de, and hl) so that then they can hold two bytes, or a word. These registers are called Register Pairs because it is two registers put together. They can only be grouped into bc, de, or hl. Bh and da don't exist.

a f b c d e h l

The a register is called the Accumulator because it is used the most, just as hl is called the Address Register Pair because it is used for referencing memory addresses mostly.
Putting parenthesis around a registered pair signifies that you are talking about the byte at the *address* held by the register pair.

Conditionals

Zero (Z, NZ) a subtraction resulted in a zero, a decrement ended up being zero, or there was zero difference between the two values compared.

Carry (C, NC) arg1 > a Carry is set, arg1 <= a Carry is not set, the carry flag tells us the resultant value was too big to be stored in the register.

Parity (PO, PE) byte has even number PE "even parity", odd number of 1 digits "odd parity" PO, Parity Overflow = 1 when even parity, 0 = odd parity.

Sign (P/M) P = not sign, M = sign (the number/result is negative)

Here are some general rules on using CP

Unsigned

If A == N, then Z flag is set.
If A != N, then Z flag is not set
If A < N, then C flag is set.
If A >= N, then C flag is not set

Signed

If A == N, then Z flag is set.
If A != N, then Z flag is not set
If A < N, then S and P/V are different.
if A >= N, then S and P/V are the same.

opcode	t-states	explanation
ADC A,(HL)	7	Add with carry location (HL) to acccumulator.
ADC A,(IX+d)	19	Add with carry location (IX+d) to accumulator.
ADC A,(IY+d)	19	Add with carry location (IY+d) to accumulator.
ADC A,n	7	Add with carry value n to accumulator.
ADC A,r	4	Add with carry register r to accumulator.
ADC HL,BC	15	Add with carry register pair BC to HL.
ADC HL,DE	15	Add with carry register pair DE to HL.
ADC HL,HL	15	Add with carry register pair HL to HL.
ADC HL,SP	15	Add with carry register pair SP to HL.
ADD A,(HL)	7	Add location (HL) to acccumulator.
ADD A,(IX+d)	19	Add location (IX+d) to accumulator.
ADD A,(IY+d)	19	Add location (IY+d) to accumulator.
ADD A,n	7	Add value n to accumulator.
ADD A,r	4	Add register r to accumulator.
ADD HL,BC	11	Add register pair BC to HL.
ADD HL,DE	11	Add register pair DE to HL.
ADD HL,HL	11	Add register pair HL to HL.
ADD HL,SP	11	Add register pair SP to HL.
ADD IX,BC	15	Add register pair BC to IX.
ADD IX,DE	15	Add register pair DE to IX.
ADD IX,IX	15	Add register pair IX to IX.
ADD IX,SP	15	Add register pair SP to IX.
ADD IY,BC	15	Add register pair BC to IY.
ADD IY,DE	15	Add register pair DE to IY.
ADD IY,IY	15	Add register pair IY to IY.
ADD IY,SP	15	Add register pair SP to IY.
AND (HL)	7	Logical AND of value at location (HL) to accumulator.
AND (IX+d)	19	Logical AND of value at location (IX+d) to accumulator.
AND (IY+d)	19	Logical AND of value at location (IY+d) to accumulator.
AND n	7	Logical AND of value n to accumulator.
AND r	4	Logical AND of register r to accumulator.
BIT b,(HL)	12	Test bit b of location (HL).
BIT b,(IX+d)	20	Test bit b of location (IX+d).
BIT b,(IY+d)	20	Test bit b of location (IY+d).
BIT b,r	8	Test bit b of register r.
CALL cc,nn	17,10	Call subroutine at location nn if condition CC is true.
CALL nn	17	Call subroutine at location.
CCF	4	Complement carry flag.
CP (HL)	7	Compare value at location (HL) with accumulator.
CP (IX+d)	19	Compare value at location (IX+d) with accumulator.
CP (IY+d)	19	Compare value at location (IY+d) with accumulator.
CP n	7	Compare value n with accumulator.
CP r	4	Compare register r with accumulator.

CPD	16	Comapre location (HL) and acc., decrement HL and BC,
CPDR	21,16	Perform a CPD and repeat until BC=0.
CPI	16	Compare location (HL) and acc., incr HL, decr BC.
CPIR	21,16	Perform a CPI and repeat until BC=0.
CPL	4	Complement accumulator (1's complement).
DAA	4	Decimal adjust accumulator.
DEC (HL)	11	Decrement value at location (HL).
DEC (IX+d)	23	Decrement value at location (IX+d).
DEC (IY+d)	23	Decrement value at location (IY+d).
DEC BC	6	Decrement register pair BC.
DEC DE	6	Decrement register pair DE.
DEC HL	6	Decrement register pair HL.
DEC IX	10	Decrement IX.
DEC IY	10	Decrement IY.
DEC r	4	Decrement register r.
DEC SP	6	Decrement register pair SP.
DI	4	Disable interrupts. (except NMI at 0066h)
DJNZ n	13,8	Decrement B and jump relative if B<>0.
EI	4	Enable interrupts.
EX (SP),HL	19	Exchange the location (SP) and HL.
EX (SP),IX	23	Exchange the location (SP) and IX.
EX (SP),IY	23	Exchange the location (SP) and IY.
EX AF,AF'	4	Exchange the contents of AF and AF'.
EX DE,HL	4	Exchange the contents of DE and HL.
EXX	4	Exchange the contents of BC,DE,HL with BC',DE',HL'.
HALT	4	Halt computer and wait for interrupt.
IM 0	8	Set interrupt mode 0. (instruction on data bus by int device)
IM 1	8	Set interrupt mode 1. (rst 38)
IM 2	8	Set interrupt mode 2. (vector jump)
IN A,(n)	11	Load the accumulator with input from device n.
IN r,(c)	12	Load the register r with input from device (C).
INC (HL)	11	Increment location (HL).
INC (IX+d)	23	Increment location (IX+d).
INC (IY+d)	23	Increment location (IY+d).
INC BC	6	Increment register pair BC.
INC DE	6	Increment register pair DE.
INC HL	6	Increment register pair HL.
INC IX	10	Increment IX.
INC IY	10	Increment IY.
INC r	4	Increment register r.
INC SP	6	Increment register pair SP.
IND	16	(HL)=Input from port (C). Decrement HL and B.
INDR	21,16	Perform an IND and repeat until B=0.
INI	16	(HL)=Input from port (C). HL=HL+1. B=B-1.

INIR	21,16	Perform an INI and repeat until B=0.
JP (HL)	4	Unconditional jump to location (HL).
JP (IX)	8	Unconditional jump to location (IX).
JP (IY)	8	Unconditional jump to location (IY).
JP cc,nn	10	Jump to location nn if condition cc is true.
JP nn	10	Unconditional jump to location nn
JR C,n	12,7	Jump relative to PC+n if carry=1.
JR n	12	Unconditional jump relative to PC+n.
JR NC,n	12,7	Jump relative to PC+n if carry=0.
JR NZ,n	12,7	Jump relative to PC+n if non zero (Z=0).
JR Z,n	12,7	Jump relative to PC+n if zero (Z=1).
LD (BC),A	7	Load location (BC) with accumulator.
LD (DE),A	7	Load location (DE) with accumulator.
LD (HL),n	10	Load location (HL) with value n.
LD (HL),r	7	Load location (HL) with register r.
LD (IX+d),n	19	Load location (IX+d) with value n.
LD (IX+d),r	19	Load location (IX+d) with register r.
LD (IY+d),n	19	Load location (IY+d) with value n.
LD (IY+d),r	19	Load location (IY+d) with register r.
LD (nn),A	13	Load location (nn) with accumulator.
LD (nn),BC	20	Load location (nn) with register pair BC.
LD (nn),DE	20	Load location (nn) with register pair DE.
LD (nn),HL	16	Load location (nn) with HL.
LD (nn),IX	20	Load location (nn) with IX.
LD (nn),IY	20	Load location (nn) with IY.
LD (nn),SP	20	Load location (nn) with register pair SP.
LD A,(BC)	7	Load accumulator with value at location (BC).
LD A,(DE)	7	Load accumulator with value at location (DE).
LD A,(nn)	13	Load accumulator with value at location nn.
LD A,I	9	Load accumulator with I.(interrupt vector register)
LD A,R	9	Load accumulator with R.(memory refresh register)
LD BC,(nn)	20	Load register pair BC with value at location (nn).
LD BC,nn	10	Load register pair BC with nn.
LD DE,(nn)	20	Load register pair DE with value at location (nn).
LD DE,nn	10	Load register pair DE with nn.
LD HL,(nn)	16	Load HL with value at location (nn). (L-first)
LD HL,nn	10	Load register pair HL with nn.
LD I,A	9	Load I with accumulator.
LD IX,(nn)	20	Load IX with value at location (nn).
LD IX,nn	14	Load IX with value nn.
LD IY,(nn)	20	Load IY with value at location (nn).
LD IY,nn	14	Load IY with value nn.
LD r,(HL)	7	Load register r with value at location (HL).
LD r,(IX+d)	19	Load register r with value at location (IX+d).

LD r,(IY+d)	19	Load register r with value at location (IY+d).
LD R,A	9	Load R with accumulator.
LD r,n	7	Load register r with value n.
LD r,r'	4	Load register r with register r'.
LD SP,(nn)	20	Load register pair SP with value at location (nn).
LD SP,HL	6	Load SP with HL.
LD SP,IX	10	Load SP with IX.
LD SP,IY	10	Load SP with IY.
LD SP,nn	10	Load register pair SP with nn.
LDD	16	Load location (DE) with location (HL), decrement DE,HL,BC.
LDDR	21	,16 Perform an LDD and repeat until BC=0.
LDI	16	Load location (DE) with location (HL), incr DE,HL; decr BC.
LDIR	21	,17 Perform an LDI and repeat until BC=0.
NEG	8	Negate accumulator (2's complement).
NOP	4	No operation.
OR (HL)	7	Logical OR of value at location (HL) and accumulator.
OR (IX+d)	19	Logical OR of value at location (IX+d) and accumulator.
OR (IY+d)	19	Logical OR of value at location (IY+d) and accumulator.
OR n	7	Logical OR of value n and accumulator.
OR r	4	Logical OR of register r and accumulator.
OTDR	21,16	Perform an OUTD and repeat until B=0.
OTIR	21,16	Perform an OTI and repeat until B=0.
OUT (C),r	12	Load output port (C) with register r.
OUT (n),A	11	Load output port (n) with accumulator.
OUTD	16	Load output port (C) with (HL), decrement HL and B.
OUTI	16	Load output port (C) with (HL), incr HL, decr B.
POP AF	10	Load register pair AF with top of stack.
POP BC	10	Load register pair BC with top of stack.
POP DE	10	Load register pair DE with top of stack.
POP HL	10	Load register pair HL with top of stack.
POP IX	14	Load IX with top of stack.
POP IY	14	Load IY with top of stack.
PUSH AF	11	Load register pair AF onto stack.
PUSH BC	11	Load register pair BC onto stack.
PUSH DE	11	Load register pair DE onto stack.
PUSH HL	11	Load register pair HL onto stack.
PUSH IX	15	Load IX onto stack.
PUSH IY	15	Load IY onto stack.
RES b,(HL)	15	Reset bit b in value at location (HL).
RES b,(IX+d)	23	Reset bit b in value at location (IX+d).
RES b,(IY+d)	23	Reset bit b in value at location (IY+d).
RES b,r	8	Reset bit b of register r.
RET	10	Return from subroutine.
RET cc	11,5	Return from subroutine if condition cc is true.

RETI	14	Return from interrupt.
RETN	14	Return from non-maskable interrupt.
RL (HL)	15	Rotate left through value at location (HL).
RL (IX+d)	23	Rotate left through value at location (IX+d).
RL (IY+d)	23	Rotate left through value at location (IY+d).
RL r	8	Rotate left through register r.
RLA	4	Rotate left accumulator through carry.
RLC (HL)	15	Rotate location (HL) left circular.
RLC (IX+d)	23	Rotate location (IX+d) left circular.
RLC (IY+d)	23	Rotate location (IY+d) left circular.
RLC r	8	Rotate register r left circular.
RLCA	4	Rotate left circular accumulator.
RLD	18	Rotate digit left and right between accumulator and (HL).
RR (HL)	15	Rotate right through carry location (HL).
RR (IX+d)	23	Rotate right through carry location (IX+d).
RR (IY+d)	23	Rotate right through carry location (IY+d).
RR r	8	Rotate right through carry register r.
RRA	4	Rotate right accumulator through carry.
RRC (HL)	15	Rotate value at location (HL) right circular.
RRC (IX+d)	23	Rotate value at location (IX+d) right circular.
RRC (IY+d)	23	Rotate value at location (HL+d) right circular.
RRC r	8	Rotate register r right circular.
RRCA	4	Rotate right circular accumulator.
RRD	18	Rotate digit right and left between accumulator and (HL).
RST 00h	11	Restart to location 0000h.
RST 08h	11	Restart to location 0008h.
RST 10h	11	Restart to location 0010h.
RST 18h	11	Restart to location 0018h.
RST 20h	11	Restart to location 0020h.
RST 28h	11	Restart to location 0028h.
RST 30h	11	Restart to location 0030h.
RST 38h	11	Restart to location 0038h.
SBC A,(HL)	7	Subtract value at location (HL) from accu. with carry.
SBC A,(IX+d)	19	Subtract value at location (IX+d) from accu. with carry.
SBC A,(IY+d)	19	Subtract value at location (IY+d) from accu. with carry.
SBC A,n	7	Subtract value n from accumulator with carry.
SBC A,r	4	Subtract register r from accumulator with carry.
SBC HL,BC	15	Subtract register pair BC from HL with carry.
SBC HL,DE	15	Subtract register pair DE from HL with carry.
SBC HL,HL	15	Subtract register pair HL from HL with carry.
SBC HL,SP	15	Subtract register pair SP from HL with carry.
SCF	4	Set carry flag (C=1).
SET b,(HL)	15	Set bit b of location (HL).
SET b,(IX+d)	23	Set bit b of location (IX+d).

SET b,(IY+d)	23	Set bit b of location (IY+d).
SET b,r	8	Set bit b of register r.
SLA (HL)	15	Shift value at location (HL) left arithmetic.
SLA (IX+d)	23	Shift value at location (IX+d) left arithmetic.
SLA (IY+d)	23	Shift value at location (IY+d) left arithmetic.
SLA r	8	Shift register r left arithmetic.
SRA (HL)	15	Shift value at location (HL) right arithmetic.
SRA (IX+d)	23	Shift value at location (IX+d) right arithmetic.
SRA (IY+d)	23	Shift value at location (IY+d) right arithmetic.
SRA r	8	Shift register r right arithmetic.
SRL (HL)	15	Shift value at location (HL) right logical.
SRL (IX+d)	23	Shift value at location (IX+d) right logical.
SRL (IY+d)	23	Shift value at location (IY+d) right logical.
SRL r	8	Shift register r right logical.
SUB (HL)	7	Subtract location (HL) from accumulator.
SUB (IX+d)	19	Subtract location (IX+d) from accumulator.
SUB (IY+d)	19	Subtract location (IY+d) from accumulator.
SUB n	7	Subtract value n from accumulator.
SUB r	4	Subtract register r from accumulator.
XOR (HL)	7	Exclusive OR value at location (HL) and accumulator.
XOR (IX+d)	19	Exclusive OR value at location (IX+d) and accumulator.
XOR (IY+d)	19	Exclusive OR value at location (IY+d) and accumulator.
XOR n	7	Exclusive OR value n and accumulator.
XOR r	4	Exclusive OR register r and accumulator.
EXTENDED OPCODES		
LDIX LDIRX LDDX LDDRX	21/16	LDDX/LDDRX advance DE by incrementing it (like LDI), while HL is decremented (like LDD).
LDWS	14	Copies the byte pointed to by HL to the address pointed to by DE and increments only L and D.
LDPIRX	21/16	Similar to LDIRX except the source byte address is not just HL, but is obtained by using the top 13 bits of HL and the lower 3 bits of DE and HL does not increment during whole loop
OUTINB	16	Behaves like OUTI, but doesn't decrement B.
MUL D,E	16	Multiplies D by E, storing 16 bit result into DE
ADD HL,A, ADD DE,A, ADD BC,A	8	Add accumulator to reg pair

ADD HL,$im16
ADD DE,$im16
ADD BC,$im16
	16	Add 16 bit immediate number to reg pair

SWAPNIB	8	Swaps the high and low nibbles of the accumulator.
MIRROR A	8	Mirrors (reverses the order) of bits in the accumulator.

PUSH $im16	23	Push an address on the stack
NEXTREG $im8,$im8 NEXTREG $im8,A	20 17	Sets hardware registers
PIXELDN	8	Takes E and D as the X,Y coordinate of a point and calculates the address of the byte containing this pixel in the pixel area of standard ULA screen 0, storing it in HL.
PIXELAD	8	Updates the address in HL to move down by one line of pixels.
SETAE	8	takes the bit number to set from E (only the low 3 bits) and sets whole A to value of that bit, but counted from top to bottom (E=0 will produce A:=$80, E=7 will produce A:=$01). This works as pixel mask for ULA bitmap modes, when E is 0..255 x-coordinate.
TEST $im8	11	Similar to CP, but performs an AND instead of a subtraction.
BSLA DE,B	8	Shift instructions use only bits 4..0 of B, BSLA shifts DE left, BSRA/BSRL/BSRF shifts DE right in arithmetic/logical/fill-one way. BRLC rotates DE left by B places, uses only bits 3..0 of B (to rotate right, use B=16-places).
BSRA DE,B	8	
BSRL DE,B	8	
BSRF DE,B	8	
BRLC DE,B	8	
JP (C)	13	The JP (C) sets bottom 14 bits of current PC* to value read from I/O port: (can be used to execute code block read from a disk stream

The ULA behaves exactly like on the original machines. It generates its display from memory located in 16K banks 5 and 7.

The 48K Spectrum gathers pixel information from address 0x4000 and attribute information from address 0x5800.

On 128K Spectrums this is 16K bank 5. Pixel information comes from offset 0 in bank 5 and attribute data from offset 0x1800 in bank 5. (The memory map is ROM, BANK 5, BANK 2, BANK 0 when machine code programs start; the 128K allows the top 16K to hold different memory banks).

The 128K Spectrums also introduced a second display file like the original at offset 0 in bank 7. You can tell the ula to read its display data from bank 5 or bank 7 by setting an appropriate bit in i/o port 0x7ffd. The display format is the same with pixel information coming from offset 0 in bank 7 and attribute information coming from offset 0x1800 in bank 7. With the port 0x7ffd mapping ability you can put bank 5 or bank 7 in the top 16K. If you put bank 5 up there you could read the normal display in two places: - at 0x4000 and at 0xc000.

The Spectrum Next adds a new layer to this because it can place any memory bank anyplace in the 64K in 8K units. You could place the ULA's bank 5 in the bottom 16K by writing 10 to mmu0 and 11 to mmu1 (page numbers are doubled because they refer to 8K pages and not 16K banks). Likewise, you could place bank 7 down there by writing 14 to mm0 and 15 to mmu1. Or you could make either of the display files appear anyplace else or even more than once in the memory map.

You can completely remove the ula display from the 64K space and have ram instead -- this is needed by cpm for example. It's not necessary to be this flexible for the ula -- it's just a side effect of the banking mechanism just like it's a side effect that bank 5 can appear at 0x4000 and 0xc000 at the same time in the 128K's banking.

The ula is one display layer. The tilemap and layer 2 are some other display layers. They stack on top of each other when pixels are generated and the display order (what's on top of what) comes from the nextreg. Unfortunately, the number of layer orderings did not grow with the capabilities of the machine so the ula and tilemap are coupled in that they must be adjacent to each other, one of them on top of the other except for an exception made for sprites which is documented in the nextreg that controls orderings. Hopefully some of this will be unwound in future hw updates.