A look at instructions, step by step

In this chapter, I look at various Z80 instructions step by step, showing what actually happens inside the chip as data moves around.

An overview of the instruction timing may help. One surprising thing is that instruction fetch is overlapped with execution of the previous instruction. An instruction isn't fetched until M1T3. During the first half of the M1 cycle, the previous instruction is still finishing up.

The fetch/execute overlap provides a speed increase. If one instruction completed before the next one was fetched, this would require additional cycles. This can be viewed as a simple form of pipelining, although modern processors have much deeper pipelines. For instance, the Intel Sandy Bridge processor has an instruction pipeline that is 14-19 stages deep: http://en.wikipedia.org/wiki/Sandy_Bridge

Instructions all start out the same way:
M1T1l:
The PC goes to the incrementer latch.

M1T1h:
The incrementer latch goes to the address pin latches, and thus the address pins.
The M1 pin goes low.

M1T2l:
MREQ and RD go low.
The PC receives the incremented value from the incrementer.

M1T2h:
The data pins are read into the data pin latch for the instruction fetch.

M1T3l:
The IR register goes to the incrementer latch.
The data pins continue to be read into the data pin latch for the instruction

fetch.

M1T3h:
RD and MREQ go high. RFSH goes low. The incrementer latch goes to the address pins.
The data pin latch writes to the data bus and into the instruction register. The instruction flows through the PLA and associated logic, starting the instruction decode.

M1T4l:
The IR register is updated from the incrementer.
MREQ goes low for the refresh.

M1T4h:
MREQ remains low.

Let's look at an 8-bit operation:
DEC B 0x05

In M1T3h, the instruction decode PLA activates lines 66 (inc/dec) and 75 (dec). (The specific register selection happens later based on the bits of the instruction.)

Things start to get interesting in M1T4l:
AF is read out of the register file onto the register buses. A is stored in the ALU's op2 latch. The flags are stored in the flag latches. Neither A nor the flags are used in this instruction, though.

At this point the next instruction starts.
In M1T1l, the B register goes to the upper register bus, which is connected to the ALU bus. The ALU bus is latched into the ALU op1 register. The ALU op2 latch is cleared to 0.

In M1T1l and M1T1, the ALU takes the lower 4 bits of op2 (complemented) and adds the lower 4 bits of op1, storing the result in the ALU 4-bit output latch.

In M1T2h and M1T3l of the next instruction, the ALU takes the upper 4 bits of op2l (complemented) and adds the upper 4 bits of op1. The result, along with the 4 lower bits from the output latch, is written to the upper register bus.

In M1T3h, the result (on the upper register bus) is written to the B register.

Some interesting points about this operation:
The instruction execution doesn't start until M1T1l, which is the start of the next instruction. This provides the fetch/execute overlap. The instruction execution doesn't finish until M1T3h of the next instruction. The instruction doesn't require any additional cycles beyond the minimum 4 T states in M1, even though the instruction doesn't even start its execution until the next M1 state. The instruction finishes up in M1T3h, which leaves the buses free to move AF for the next instruction in M1T4l.

Note that the ALU operates 4 bits at a time. It takes two half-cycles to compute each 4-bit result. There is a half-cycle in between, which means the first 4-bit operation happens in l/h clock, while the second happens in h/l clock.

The ALU has two 8-bit latches for two input arguments, and a 4-bit output latch to hold the lower half of the output while the upper half is computed. The 4-bit ALU is an unusual design choice; processors of that time normally used an 8-bit ALU.

Let's look at a 16-bit operation and how it proceeds
DEC BC 0x0b

In M1T3h, the instruction decode PLA activates lines 9 (inc/dec rr) and 14 (dec rr). (The specific register selection happens later based on the bits of the instruction.)

In M1T5l:
BC is read into the incrementer latch, passing through the gate between the

two parts of the register file.

In M1T6l:
The decremented value from the incrementer is written back to BC.

The interesting thing about this instruction is that because the incrementer is used for the PC in T1/T2, and used for the IR in T3/T4, this instruction has to use the incrementer in T5/T6. This instruction, unlike most instructions, can't overlap fetch and execution because it requires the incrementer.

Note that this instruction uses the incrementer/decrementer, rather than the ALU to perform the decrement.

Now let's look at a prefixed operation, the 16-bit ADC HL, BC: 0xED 0x4a

In the first M1 cycle, the prefix is fetched. This is decoded to the PLA line 51, which is latched in the ED prefix latch.

In the next M1 cycle, the opcode is fetched. This is decoded to PLA line 68: which indicates a 16-bit ADC/SBC.

In M1T4l, the flags are read from F into the flag latches. This provides the carry flag as input to the sum.

In order to handle the 16-bit operation, this instruction requires additional cycles. It gets these through two extra M cycles - M4 and M5 - which don't access memory, unlike normal M cycles.

M4T1l: L goes to the register bus and is latched into the ALU op1 register. The carry is latched into the half-carry flag (which is a bit surprising), and from there to the ALU's carry-in.

M4T2l: C goes to the register bus and is latched into the ALU op2 register.
The ALU computes the low-4-bit sum during M4T2l and M4T2h
The carry out is grabbed by a pass transistor during M4T2h and the half-carry latch is updated during M4T3l, to be used in the high-4-bit addition.

The ALU computes the high-4-bit sum during M4T3h and M4T4l.
The sum is stored in L during M4T4l, and the ALU's carry-out is stored in the carry flag latch.

M5T1l: D goes to the register bus and is latched into the ALU op1 register.
The carry flag is latched into the half-carry flag, and from there to the ALU's carry-in.

M5T2l: B goes to the register bus and is latched into the ALU op2 register.
The ALU computes the low-4-bit sum during M5T2l and M5T2h.
The carry out is grabbed by a pass transistor during M5T2h and the half-carry latch is updated during M5T3l, to be used in the high-4-bit addition.
The ALU computes the high-4-bit sum during M5T3h and M1T1l (of the next instruction.)
The sum is stored into H during M1T1l.
In M1T3l the ALU's carry-out is stored in the carry flag latch.
The flag latches are written to the register bus in M1T3h, M1T4l and stored back to the F register in M1T4l.

There are several things of interest in this instruction. First, the two extra M cycles are required to fit in the two 8-bit additions (i.e. four 4-bit additions), and the multiple data moves between the register file and the ALU. Second, the carry flag and half carry flag updates need to be handled carefully.