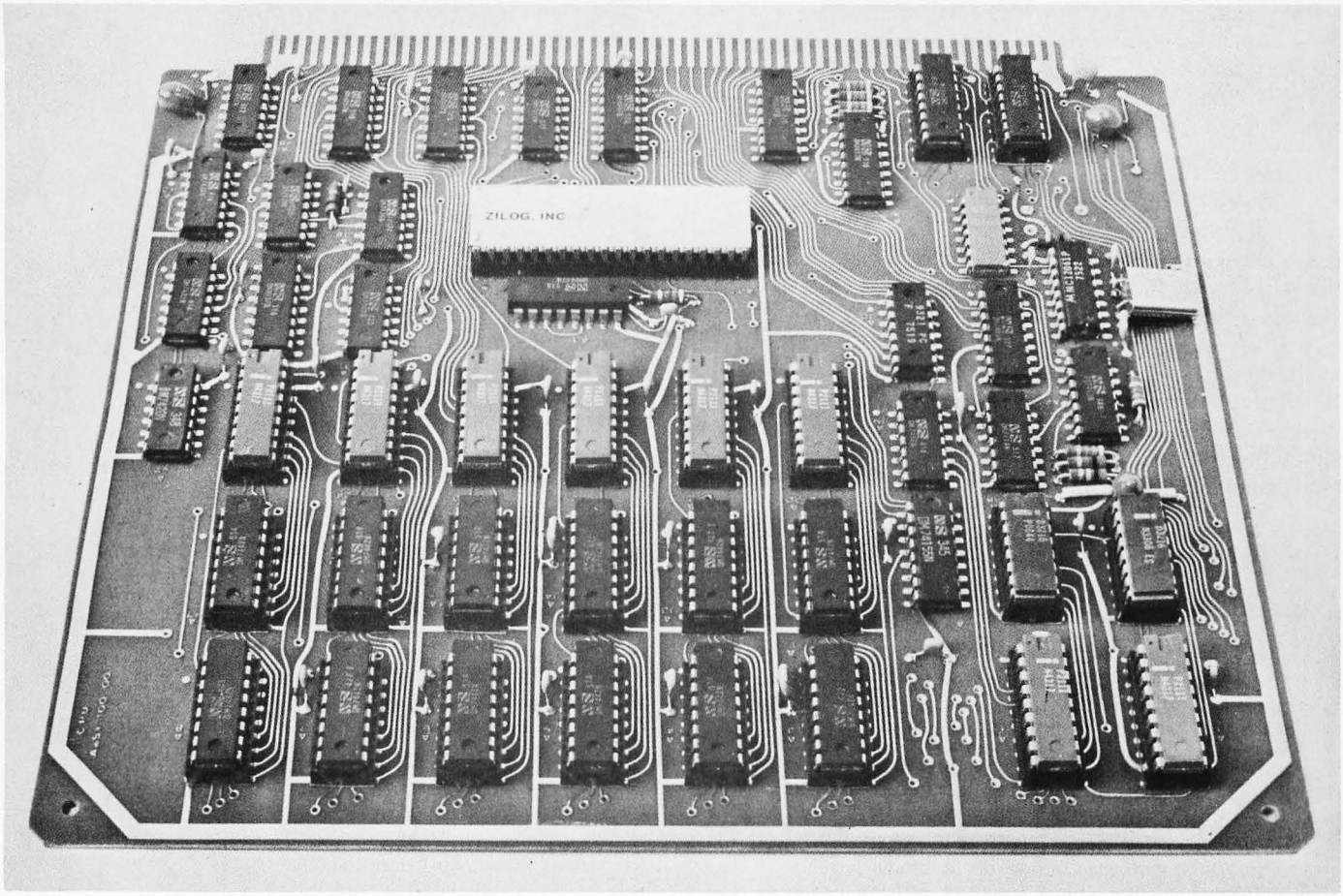

Photo 1: The Z80 microprocessor evaluation board.

Microprocessor Update: Zilog Z80

Burt Hashizume
PO Box 172
Placentia CA 92670

One feature of the Z80 not found in other 8 bit microprocessors is a built in dynamic MOS memory refresh algorithm which employs unused memory cycles to do hidden (from software timing) refresh operations.

Zilog, a fairly new company in Los Altos CA, has been sampling an 8 bit microprocessor, the Z80, since early this year. The Z80 is a "third generation," single chip, NMOS microprocessor, which is completely software compatible with Intel's 8080A. Its 158 instructions include the 8080A's 78 instructions as a subset. Because the 8080A is probably the most widely used 8 bit microprocessor on the market today and because of the Z80's upward software compatibility, this article evaluates the Z80 in comparison to the 8080A.

Physical and Electrical Characteristics

The Z80 processor is packaged in the standard 40 pin dual in line package; however, even though the Z80 is software compatible with the 8080A, it is most definitely not pin compatible. (See figure 1 and table 1 for pinout definitions.) There are numerous differences between the two processors as far as electrical characteristics are concerned.

The 8080A requires three voltage levels, +12, +5, and −5 V. A high voltage two phase clock is also required. Maximum speed is a 480 ns clock period. Finally, some sort of system controller is needed to separate the system control signals from the data bus. This all makes for a fairly complex system design around the 8080A.

On the other hand, it is very easy to design a system around the Z80. It requires only a single +5 V power supply because the

technology used is of the same type used by Motorola in its 6800 microprocessor, which also requires a single 5 V power supply. The Z80 requires a single phase 5 V clock. Maximum frequency is 2.5 MHz for a 400 ns clock period. System control signals, such as memory read and write, have separate pins from the processor and are not time shared with the data bus. An additional feature not found on any other microprocessor at the time of this writing is the capability to refresh dynamic memory.

Because the Z80 is upward software compatible with the 8080A, the internal architectures are similar. (See the register configuration in figure 2.) Both have 16 bit program counters and stack pointers as well as a register array of six general purpose registers, (B, C, D, E, H and L), an accumulator (A), and a flag register (F).

The Z80 has numerous additional characteristics. It has an additional duplicate register array consisting of 8 registers (A', F', B', C', D', E', H' and L'). These can be switched with the primary register array for fast interrupt processing. There are also two 16 bit index registers (IX and IY) for increased addressing capability and easier data manipulation. An 8 bit interrupt vector register (I) expands the capability and increases the power and speed of interrupt handling by the processor. Finally, an 8 bit memory refresh register (R) automatically increments after every instruction fetch and refreshes memory while the processor is not using the bus. Thus the execution time of the system is not increased due to refresh overhead.

Software

Now that we have seen the hardware aspects of the Z80 and how it compares to the 8080A, let's take a look at its instruction set. The fact that the Z80 has 158 instructions versus the 8080A's 78 gives only a small indication of its technological superiority in this area. The instruction set can be broken up into two aspects, addressing modes and instruction groups.

Since the Z80 is software compatible with 8080A, it necessarily has the same addressing modes as the 8080A. The modes in common are register addressing, register indirect addressing, direct addressing, and immediate addressing.

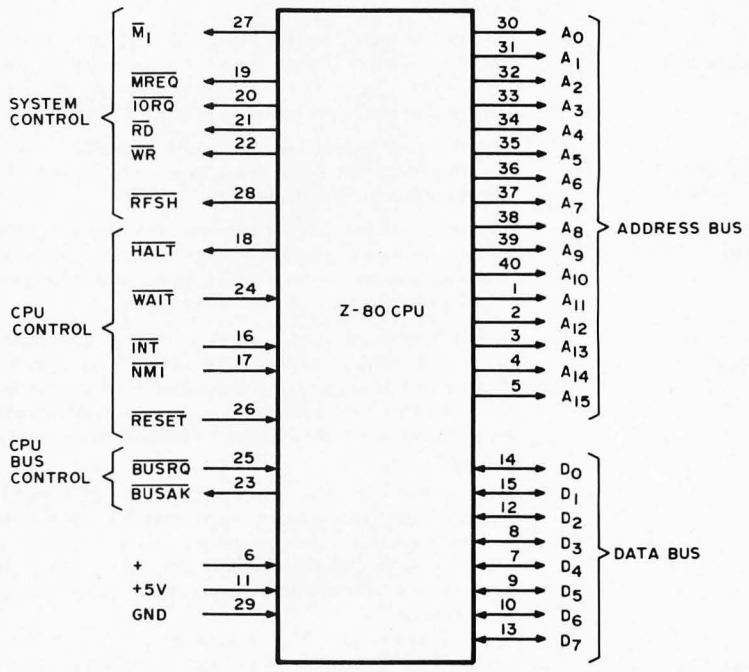

Figure 1: Pin configuration of the Z80 processor. Of particular note to custom hardware hackers is the "M_1" line which gives users the possibility of identifying instruction cycles.

Table 1: Signal list for the Z80 processor. This table lists each active pin of the Z80 with a short explanation of its purpose.

A_0–A_{15} (Address Bus)	Tri-state output, active high. A_0–A_{15} constitute a 16 bit address bus. The address bus provides the address for memory (up to 64 K bytes) data exchanges and for IO device data exchanges. IO addressing uses the 8 lower address bits to allow the user to directly select up to 256 input or 256 output ports. A_0 is the least significant address bit. During refresh time, the lower 7 bits contain a valid refresh address.
D_0–D_7 (Data Bus)	Tri-state input and output, active high. D_0–D_7 constitute an 8 bit bidirectional data bus. The data bus is used for data exchanges with memory and IO devices.
$\overline{M_1}$ (Machine Cycle one)	Output, active low. $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution.
$\overline{MREQ}$ (Memory Request)	Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.
$\overline{IORQ}$ (Input/Output Request)	Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid IO address for a IO read or write operation. An $\overline{IORQ}$ signal

Table 1 (continued).

	is also generated when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus. Interrupt Acknowledge operations occur during M_1 time while IO operations never occur during M_1 time.
$\overline{RD}$ (Memory Read)	Tri-state output, active low. $\overline{RD}$ indicates that the processor wants to read data from memory or an IO device. The addressed IO device or memory should use this signal to gate data onto the processor data bus.
$\overline{WR}$ (Memory Write)	Tri-state output, active low. $\overline{WR}$ indicates that the processor data bus holds valid data to be stored in the addressed memory or IO device.
$\overline{RFSH}$ (Refresh)	Output, active low. $\overline{RFSH}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories.
$\overline{HALT}$ (Halt state)	Output, active low. $\overline{HALT}$ indicates that the processor has executed a HALT software instruction and is awaiting either a non maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the processor executes NOPs to maintain memory refresh activity.
$\overline{WAIT}$ (Wait)	Input, active low. $\overline{WAIT}$ indicates to the Z80 processor that the addressed memory or IO devices are not ready for a data transfer. The processor continues to enter wait states for as long as this signal is active. This signal allows memory or IO devices of any speed to be synchronized to the processor.
$\overline{INT}$ (Interrupt Request)	Input, active low. The Interrupt Request signal is generated by IO devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip flop (IFF) is enabled and if the $\overline{BUSRQ}$ signal is not active. When the processor accepts the interrupt, an acknowledge signal ($\overline{IORQ}$ during M_1 time) is sent out at the beginning of the next instruction cycle. The processor can respond to an interrupt in three different modes that are described in detail in the Zilog documentation.
$\overline{NMI}$ (Non Maskable Interrupt)	Input, active low. The non maskable interrupt request line has a higher priority than $\overline{INT}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip flop. $\overline{NMI}$ automatically forces the Z80 processor to restart to location 0066 hexadecimal. The program counter is automatically saved in the external stack so that the user can return to the program that was interrupted.
$\overline{RESET}$	Input, active low. $\overline{RESET}$ forces the program counter to zero and initializes the processor. The processor initialization includes: 1) Disable the interrupt enable flip flop 2) Set Register I = 00 3) Set Register R = 00 During reset time, the address bus and data bus go to a high impedance state and all control output signals go to the inactive state.
$\overline{BUSRQ}$ (Bus Request)	Input, active low. The bus request signal is used to request the processor address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these buses. When $\overline{BUSRQ}$ is activated, the processor will set these buses to a high impedance state as soon as the current processor machine cycle is terminated.
$\overline{BUSAK}$ (Bus Acknowledge)	Output, active low. Bus acknowledge is used to indicate to the requesting device that the processor address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

- *Register addressing.* The opcode itself specifies a register or register pair in which the data is contained. An example would be to load the data in register B into register D.
- *Register indirect addressing.* The opcode specifies a register pair which contains a 16 bit address. This address points to the data in memory or is an address to be loaded into the program counter (PC). An example would be to load the accumulator with data in memory pointed to by the HL register pair.
- *Direct addressing.* The opcode is followed by two bytes of operand. These two bytes are either a 16 bit address pointing to data in memory or a 16 bit address to be loaded into the PC. For example, in a jump instruction, the two bytes indicate an address to which program control is transferred.
- *Immediate addressing.* The opcode is followed by one or two bytes of operand. This operand is the data itself to be used. An example is load accumulator immediate which moves an 8 bit operand into the accumulator.

To these addressing modes, the Z80 has added three more powerful modes. These are indexed addressing, relative addressing, and bit addressing. The first two are somewhat similar to index and relative addressing in the Motorola 6800 microprocessor.

- *Indexed addressing.* The opcode is followed by an 8 bit displacement. This displacement is a *signed* two's complement number to be added to the contents of one of the two index registers. The result is a 16 bit effective address. The contents of the index register are unchanged.
- *Relative addressing.* The opcode is followed by an 8 bit signed two's complement number. The number is added to the contents of the program counter and the result placed back in the PC. This results in being able to execute program jumps within a range of +129 to −126 bytes using only a two byte instruction. Since most programs have a lot of jumps to locations relatively close to current locations, using relative addressing will significantly reduce program size. Another advantage is the ability to write relocatable code using relative addressing.
- *Bit addressing.* Three bits in the opcode itself specify one of eight bits in a byte to be addressed. This byte

could be the contents of a register or of a memory location. An example would be to set bit 6 in memory pointed to by index register, IX, displaced by −20.

The Z80 instruction set's increase of 80 instructions over the 8080A's didn't come from just increasing the number of addressing modes. There are instructions which don't exist in any other microprocessor. The instruction set will be broken up into groups by their function.

Load and Exchange Instructions

This group includes all the instructions that move data to and from registers, such as load B from D, load C from memory, store HL into memory, push IX into stack, and exchange AF with A′F′. The 8080A has most of the same instructions.

Block Transfer and Search Instructions

This group has several useful and unique instructions. The load and increment instruction moves one byte of data from memory pointed to by HL to another memory location pointed to by DE. Both register pairs are automatically incremented and the byte counter, BC, is decremented. This instruction is extremely valuable in moving blocks of data around.

Another instruction repeats the load and increment instruction automatically until the byte counter reaches zero. Thus, in one instruction, a block of data, up to 64 K bytes in length, can be moved anywhere in memory. Each byte of data transferred requires only 8.4 μs.

In the compare and increment instruction, the contents of the accumulator are compared with that of memory pointed to by HL. The appropriate flag bits are set, HL is automatically incremented, and the byte counter is decremented.

The instruction compare, increment, and repeat repeats the above instruction until either a match is found or the counter reaches zero.

The 8080A has no analogy to these instructions. It would have to execute three to ten separate instructions to achieve the same result. The number of bytes would be several times larger and the execution time would be several times longer.

Arithmetic and Logical Instructions

These instructions include all the adds and subtracts, increments, compares, exclusive-ors, etc. What the Z80 has added to the 8080A instructions is the indexed addressing mode and double precision add with carry and subtract with carry.

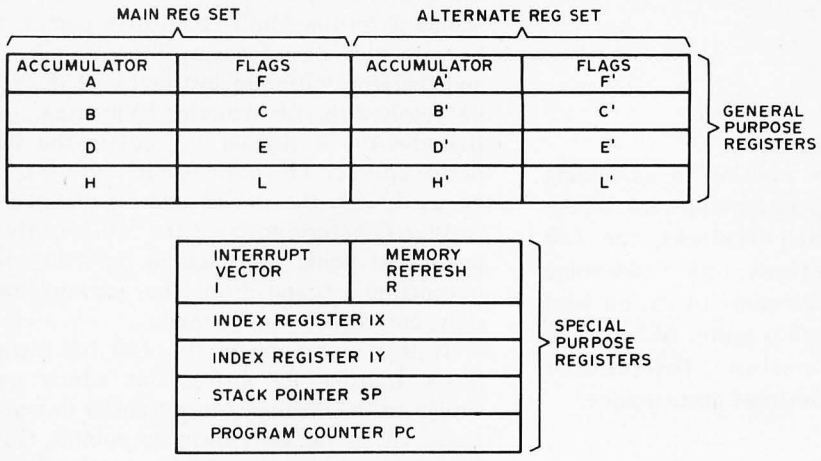

Figure 2: Programmable registers of the Z80. Considerable improvement over the 8080 design is found in the alternate register set, and the addition of two index registers, interrupt vector and memory refresh registers.

Rotate and Shift Instructions

Here the Z80 has taken the four 8080A rotate accumulator instructions and increased the possible addressing modes as well as included logical shifts and arithmetic shifts. On top of this there are a couple of rotate digit instructions. With these a digit (4 bits) can be rotated with two digits in a memory location, which is great for BCD arithmetic.

Bit Manipulation Instructions

There are three basic operations, test bit, set bit, and reset bit. With the various addressing modes, a powerful group of instructions is generated. For instance, if several memory locations are used for IO devices, status bits can be individually tested and control bits individually set or reset. The 8080A (nor any other 8 bit microprocessor) has no such capability to manipulate bits.

The Z80 should be a natural for string manipulation software with its pair of full 16 bit index registers and powerful multi-byte operations such as block move, memory search and block IO instructions.

Jump, Call, and Return

Both the 8080A and Z80 have numerous conditional and unconditional jumps, calls, and returns. In addition, the Z80 has several jump relative instructions using relative addressing. One of special interest decrements the B register, and jumps relative if B is not zero. This is especially useful in program loop control; it would take the 8080A two instructions to perform the same task.

Input/Output Instructions

The 8080A has two IO instructions, input and output to and from the accumulator. The device address is in the second byte of the instruction, which means that each

In addition to expanding operations upward to the level of blocks, the Z80 refines its addressing downward to the bit level with a group of bit manipulation instructions which are quite unique.

device must have its own IO routine. One standard routine can't be used in common because each device has a different address and therefore different instruction. The Z80 has resolved this by including IO instructions that use the C register to contain the IO device address. Therefore one IO routine can be used with the device address placed in register C before entering the routine. Also instead of being restricted in inputting or outputting to and from the accumulator only, any register can be used.

If this isn't enough, the Z80 has eight block transfer IO instructions which are similar to the memory block transfer instructions. HL is the only memory pointer, C is the device pointer, and B is the byte counter. Therefore, an IO block transfer can handle up to 256 bytes. Essentially these commands are a processor implementation of direct memory access (DMA), invoked by a software sequence.

Miscellaneous Features

These instructions include no-operation, halt, enable and disable interrupts, decimal adjust accumulator, set carry, and complement carry. The Z80 can also select one of three interrupt modes.

Interrupts on the Z80

The Z80 simplifies the hardware required to implement a system as compared to the original 8080 design. Aside from the instruction enhancements, here is a way to get an 8080 instruction set with the ease of interfacing until now only available (in 8 bits) with processors like the 6800 and 6502.

The 8080A has one input for interrupts; the Z80 has two. One is a nonmaskable interrupt (similar to the Motorola 6800 or MOS Technology 6502) which cannot be disabled by the software. The other is a maskable interrupt which can be selectively enabled or disabled by the program. The maskable interrupt is analogous to the single 8080A interrupt.

A nonmaskable interrupt will be accepted at all times by the Z80 processor. When one occurs, the processor will execute a restart to hexadecimal location 0066. The nonmaskable interrupt is used for very important functions that must be serviced immediately, such as a power failure routine.

The Z80 has three programmable modes for processor response to a maskable interrupt. There are three instructions that will select these three modes.

Mode 0 is identical to the 8080A single interrupt response mode. The interrupting device places an instruction on the data bus, and the processor executes it. The instruction will often be a restart. This mode is also the default mode for the Z80 upon a reset.

In mode 1, the processor will respond to an interrupt by executing a restart to location 0056. The response in this mode is similar to the response to a nonmaskable interrupt except for the restart location.

In mode 2, a table of 16 bit starting addresses for every interrupt routine must be maintained. This table can be anywhere in memory. When an interrupt is accepted, a 16 bit address is formed from the contents of the 8 bit I register and the 8 bits on the data bus. The I register contains the upper 8 bits of the address and the 8 bit data on the data bus from the peripheral device constitutes the lower 8 bits of the address. This 16 bit address points to a location in the interrupt vector table. The processor fetches the 16 bit address found at the selected table location (in two bytes) and loads the program counter with its value. This whole process takes 19 clock periods, or just 7.6 μs.

The peripheral devices in the Z80 microcomputer family all have daisy chain interrupt structures. They automatically supply a programmed vector to the processor during interrupt acknowledge. Only the highest priority device interrupting the processor sees the interrupt acknowledge because of the daisy chain structure. With these devices, IO interfacing becomes quite a simple task, and is as powerful as the IO techniques used in many minicomputers.

Conclusion

What does the Z80 have going for it? It's easy to interface; one chip does the job of several 8080A family chips. It's as easy, if not easier, to design an entire system around than any other microprocessor on the market today, and the Z80 is software compatible with the 8080A, the most widely used and known 8 bit microprocessor. Its instruction set is much more powerful than the 8080A's or any other 8 bit microprocessor's instruction set.

Is there anything negative about the Z80? As of this writing (March), it is not yet in production and therefore not readily available to the personal computing experimenter. The price tag for unit samples is $200, but there are numerous price breaks with larger quantities. For instance, the price is $80 for quantities of 25 – 99. This is still more expensive, however, than either the 8080A, 6800 or 6502, and is about the same as 16 bit microprocessors.

The result is a tradeoff of cost versus performance. Much of the cost difference relative to other 8 bit processors is made up by the Z80's better memory utilization and (with respect to the 8080A) by the fact that fewer parts are needed to get a minimum system going. Although the Z80 processor is priced higher than the 8080A, when the cost of all the support devices the 8080A requires are included, the costs are comparable.■

For more information on the Z80 CPU and other Z80 parts contact Zilog Inc, 170 State St, Ste 260A, Los Altos CA 94022, (415) 941-5055.■