PROGRAMMING NOTES FOR THE ALTAIR 680 μCOMPUTER

BY H. EDWARD ROBERTS AND PAUL VAN BAALEN

The 6800 microprocessor IC is the central processor (MPU or CPU) for the Altair 680 computer. Other components in the system, such as the memory and the terminal connected to the I/O port do not initiate data transfers. They respond only to requests from the MPU to send or receive data. Through the interrupt facility, a device (such as a terminal) can signal that it has completed a data transfer or that it has new data available.

The MPU initiates a data transfer by placing an address on the address bus, setting the read/write flag to signal whether it is sending or receiving, raising a VMA (valid memory address), and putting data on the data lines if it is sending. A device on the bus must recognize when an address that refers to it has been sent out and take appropriate action. In the case of a read from a memory address, the memory will examine the appropriate memory cells and set the data lines, depending on what is stored in those cells.

All data is 8-bits wide (eight binary digits), which means that data has 256 (2^8) possible values. Addresses are 16-bits wide, giving the system up to 65,536 (2^{16}) possible locations to use for memory locations and device registers. The idea of using addresses to refer to I/O devices as well as memory locations, as opposed to having special I/O instructions and a special I/O bus, is relatively new. It has greatly simplified the computer's structure.

The MPU has a number of internal registers: the program counter (PC), A and B registers, condition code flags, index register, and stack pointer. The 16-bit PC is used to keep track of which instruction the computer will execute next. Once the computer is started and the PC initialized, the MPU will start executing instruction cycles. An instruction cycle begins when the PC is sent out on the address lines. When this data comes back to the MPU, the instruction is decoded and the MPU performs the appropriate operation.

There are 59 instructions, 197 of the possible 256 codes being unassigned. These 197 instructions are grouped into 72 different types. Some instructions require one or two additional arguments. These arguments occupy the memory locations following the instruction. When the instruction is fetched, and as each argument is fetched, the PC is incremented by 1, so that after all the arguments are fetched, the PC will give the address of the next instruction to be executed (unless the execution of the current instruction modifies the PC). When the current instruction is completed, a new instruction cycle is initiated. The MPU will continue to execute instructions until the RUN/HALT switch is set to the HALT position or the power is turned off.

The MPU has two 8-bit accumulators, designated A and B. They are called accumulators because they are used for storing arithmetic results. An example of some instructions that operate on the accumulators and take no additional arguments aside from the instruction itself are:

```
FND  PRT   LDX   PLIST    Initializes index
                          register to point
                          at parts list
          CMP   A     X   See if A = con-
                          tents of location
                          index pointed to.
                          A is not changed,
                          but zero flag is
                          set if equal
          BEQ   HAVPRT    If equal, check
                          count
          TST         X   Set zero flag if in-
                          dex register points
                          at 0, which means
                          list ended
          BEQ     PERR    If so, go to PERR
          INX             Point to next
          INX             Part number
          BRA   FNDPRT    Unconditionally
                          branch back,
                          search more
HAVPRT    CMP   B   1,X   1,X gives address
                          of availability
                          count; see if enough
          BLS   PORDER    If not, order more
          PUSH  A         Save part number
          LDA   A   1,X   A = old stock count
          SBA             A = A - B = new
                          stock count
          STA   A   1,X   Store new count
          POP       A     Restore part number
          RTS             Return to caller
```

DEC A Decrement value of A by 1
TAB Set B = A
CLR B Set B = 0
NEG A Set A = O − A
COM B Complement B [B = B − (B + 1)]
SBA A = A − B
ADD A = A + B

Since only a small amount of data can be stored in the MPU itself, there are a number of instructions that specify addresses in memory as operands. The method an instruction uses for specifying a memory address is called the addressing mode of the instruction. There are seven different addressing modes for the 6800 chip.

The simplest addressing mode is extended addressing. An instruction that employs the extended mode takes two arguments and is, therefore, a three-byte instruction. The first argument gives the high-order eight bits of the address to use and the second argument gives the low-order eight bits. For example:

LDA A 3000 Load A with contents of location 3000
CLR 2000 Set location 2000 to 0
STA B 2001 Store B in location 2001

Extended addressing is the most general form of addressing since it can refer to any possible address. However, instructions employing extended addressing require three locations in memory; so, other less general but more compact addressing modes are provided.

In the direct mode of addressing, an instruction takes a single argument, which is taken to be the low eight bits of the address of the operand. The high eight bits of the address are assumed to be zero.

In the immediate mode, an instruction's single argument to the instruction is used as an operand of the instruction. For example:

LDA A 30 Set A = 30
ADD B 20 Add 20 to B

The indexed mode of addressing takes a single argument that is added to the 16-bit index register to give the address of the operand. The index register can be loaded (LDX), stored (STX), incremented (INX), decremented (DEX), and compared with another 16-bit quantity (CPX).

All of the common arithmetic and logic operations can be performed (that is, add, subtract, and, or, exclusive-or). One of the accumulators is specified as the first operand and used to store the result. The second operand can be specified using one of the addressing modes. Arithmetic operations also set/clear the condition codes, depending on the result of the operation. (That is, if the result of an operation is zero, the zero flag is set; if the result of an addition is greater than 256, the carry flag is set.) These condition codes can be used for multi-precision arithmetic or for the conditional branching described below. For example:

SUB A 1456 Subtract contents of location 1456 from A. The zero flag will be set if the result is 0.
EOR B 1,X Set B = exclusive-or of B and contents of location given by adding 1 to index register.

Unless a branch, jump, JSR, BSR, or RTS instruction is executed, the computer will execute instructions sequentially out of memory. The jump and branch instructions change the PC, which changes the address from which the next instruction will be fetched. A jump instruction is unconditional and stores its two eight-bit arguments in the 16-bit PC, using the first argument for the high-order bits in the PC. Branching, which can be executed conditionally, based on the state of the condition codes, takes a single argument that is added to the PC. The argument is taken as a signed number, which means you can branch to any location that is less than 130 locations beyond the current location or any location that is less than 127 locations behind the current location. For locations further away, the jump instruction must be used.

The 16-bit stack pointer (SP) can be decremented, incremented, loaded, stored, and transferred to or loaded from the index register. The importance of the SP comes from its use in the JSR (jump to subroutine), BSR (branch to subroutine), RTS (return from subroutine), PUSH (save an accumulator on the stack), POP (fetch a value from the stack), and the handling of interrupts. The PUSH A instruction, which takes no arguments, stores A in the address contained in the SP and decrements the SP. By setting SP to point to an area of free RAM, a programmer can save and restore temporary values with 1-byte instructions. When a JSR or BSR is executed, the PC is saved on the stack by storing the low eight bits of the PC at the address the SP contains, decrementing the SP, storing the low eight bits of the PC, and decrementing the SP again. Other than pushing the PC onto the stack, BSR is exactly like branch always, and JSR is exactly like jump.

An RTS fetches the PC off the stack in a similar manner. The use of the stack for saving return addresses for subroutine calls allows for subroutines that call themselves and does not require that locations be set aside to store the return address of every subroutine. The stack is invaluable in making the programming of the Altair 680 easy.

The sample subroutine shown on the preceding page is called with a part number in the A register and the number wanted in the B register. It searches the parts list that starts at memory location PLIST and consists of a part number followed by the quantity of that part in stock. If the desired part is not in the list, the subroutine branches to memory location PERR. If the part is located in the list, the quantity in stock is checked to make sure there are enough of the part available. If there is not a sufficient quantity in stock, the sequence at location PORDER gains control. Otherwise, B is subtracted from the number of parts in stock and the subroutine returns.