C. REGISTER PAIR INSTRUCTIONS

The *ALTAIR 8800* has eight register pair instructions. Each instruction occupies a single byte. Five of the instructions, PUSH, POP, DAD, INX, and DCX, have four variances each. The variances are specified according to any desired register pair, and the following register pair bit patterns apply:

Register Pair	Bit Pattern
B and C	00
D and E	01
H and L	10
Flags and A	11

PUSH (PUSH DATA ONTO STACK) 11 (rp)0 101 (Byte 1)

Operation: The contents of the specified register pair (rp) are stored in two bytes of memory at an address indicated by the Stack Pointer. The contents of the first register are PUSHed into the address one less than the address in the Stack Pointer. The contents of the second register are PUSHed into the address two less than the address in the Stack Pointer.

If the Status Bit Register and Accumulator (register pair PSW) pair is specified, the first byte PUSHed into memory is the Status Bit Register. This byte has the following format:

Bit Position	Contents
7	Sign Bit
6	Zero Bit
5	0
4	Auxiliary Carry Bit
3	0
2	Parity Bit
1	1

Bit Position	Contents
0	Carry Bit

For example, if the Carry Bit is set to 1 and all remaining status bits are reset to 0, the Status Bit Register will contain the following byte: 00 000 011.

After the PUSH instruction is implemented, the Stack Pointer is automatically decremented by two.

Status Bits: Unaffected.

Example: Assume PUSH BC is implemented. The instruction byte will have the following format: 11 000 101. The contents of register pair BC will be stored in memory thusly: B will be stored at the address in the Stack Pointer less one; C will be stored at the address in the Stack Pointer less two. The Stack Pointer will then be decremented by two.

POP (POP DATA OFF STACK) 11 (rp)0 001 (Byte 1)

Operation: The contents of the specified register pair (rp) are retrieved from the two bytes of memory at an address indicated by the Stack Pointer. The contents of the memory byte at the Stack Pointer address are loaded into the second register of the pair, and the contents of the byte at the Stack Pointer address plus one are loaded into the first register of the pair.

If the Status Bit Register and Accumulator (register pair PSW) pair is specified, the contents of the byte at the Stack Pointer address plus one are used to set or reset the status bits according to the format provided in the description of the PUSH instruction.

After the POP instruction is implemented, the Stack Pointer is automatically incremented by two.

Status Bits Affected: None unless register pair PSW is specified.

Example: The inverse of the example provided under the PUSH instruction will illustrate operation of the POP instruction.

DAD (DOUBLE ADD) 00 (rp)1 001 (Byte 1)

Operation: The 16-bit number formed by the two bytes in the specified register pair (rp) is added to the 16-bit number formed by the two bytes in the H and L registers. The result is stored in the H and L register pair.

Status Bits Affected: Carry.

Example: Assume the 16-bit number formed by the two bytes in register pair BC is 00 101 111 01 111 111. Assume the contents of the H and L register pair form the 16-bit number 01 100 000 00 100 101. The instruction DAD BC (00 001 001) will add the two numbers and store the result in the H and L register pair. The result of the addition is: 10 001 111 10 100 100. Since no carry occurred, the Carry Bit is reset to 0.

INX (INCREMENT REGISTER PAIR) 00 (rp)0 011 (Byte 1)

Operation: The 16-bit number formed by the two bytes in the specified register pair (rp) is incremented by one.

Status Bits: Unaffected.

Example: Assume the INX instruction 00 100 011 is present. According to the table of register pair bit patterns, the 16-bit number formed by the two bytes in the H and L register pair will be incremented by one. If the initial 16-bit number is 10 001 111 10 100 100, the new 16-bit number will be 10 001 111 10 100 101.

DCX (DECREMENT REGISTER PAIR) 00 (rp)1 011 (Byte 1)

Operation: The 16-bit number formed by the two bytes in the specified register pair is decremented by one.

Status Bits: Unaffected.

Example: Assume the DCX instruction 00 101 011 is present. According to the table of register pair bit patterns, the 16-bit number formed by the two bytes in the H and L register pair will be decremented by one. If the initial 16-bit number is 10 001 111 10 100 101, the new 16-bit number will be 10 001 111 10 100 100.

XCHG (EXCHANGE REGISTERS) 11 101 011 (Byte 1)

Operation: The 16-bit number formed by the contents of the H and L registers is exchanged with the 16-bit number formed by the contents of the D and E registers.

Status Bits: Unaffected.

Example: Assume the H register byte is 10 001 111 and the L register byte is 10 000 011. Assume the D and E register bytes are both 00 000 000. Implementation of the XCHG instruction will exchange the contents of the two register pairs so that the H and L register bytes are both 00 000 000 and the D and E register bytes are, respectively, 10 001 111 and 10 000 011.

XTHL (EXCHANGE STACK) 11 100 011 (Byte 1)

Operation: The byte stored in the L register is exchanged with the memory byte addressed by the Stack Pointer. The byte stored in the H register is exchanged with the memory byte at the address one greater than that addressed by the Stack Pointer.

Status Bits: Unaffected.

Example: The example provided under the XCHG instruction is similar to the operation which occurs when the XTHL instruction is implemented.

SPHL (LOAD SP FROM H AND L) 11 111 001 (Byte 1)

Operation: The 16-bit contents of the H and L registers replace the contents of the Stack Pointer without affecting the contents of the H and L registers.

Example: Assume the H register byte is 10 001 111 and the L register byte is 10 000 011. Assume the Stack Pointer address is 00 001 100 01 111 111. Implementation of the SPHL instruction will load the Stack Pointer with: 10 001 111 10 000 011. The contents of the H and L registers will remain unchanged.

D. ROTATE ACCUMULATOR INSTRUCTIONS

This is a special set of four instructions which apply only to the *ALTAIR 8800*'s accumulator. Only one byte of instruction is required, and no memory or register variances apply.

RLC (ROTATE ACCUMULATOR LEFT) 00 000 111 (Byte 1)

Operation: The accumulator byte is rotated one bit position to the left. The 7 bit position now occupies the 0 bit position and the Carry Bit is set with the value of the 7 bit before rotation.

Status Bits Affected: Carry.

Example: Assume the accumulator byte is 10 001 000 and the RLC instruction is present. The Carry Bit is set to equal the value of the accumulator byte's 7 bit (1), and the contents of the accumulator are rotated one bit position to the left. The 7 bit now occupies the 0 bit: 00 010 001.

RRC (ROTATE ACCUMULATOR RIGHT) 00 001 111 (Byte 1)

Operation: The accumulator byte is rotated one bit position to the right. The 0 bit position now occupies the 7 bit position and the Carry Bit is set with the value of the 0 bit before rotation.

Status Bits Affected: Carry.

Example: Assume the accumulator byte is 10 001 000 and the RRC instruction is present. The Carry Bit is set equal to the value of the accumulator byte's 0 bit (0), and the contents of the accumulator are rotated one bit position to the right. The 0 bit now occupies the 7 bit: 01 000 100.

RAL (ROTATE ACCUMULATOR LEFT THROUGH CARRY) 00 010 111

Operation: The accumulator byte is rotated one bit position to the left through the Carry Bit. The 7 bit position then occupies the Carry Bit and the Carry Bit occupies the 0 bit position.

Status Bits Affected: Carry.

Example: Assume the accumulator byte is 10 001 000, the Carry Bit is 1, and the RAL instruction is present. The contents of the accumulator are rotated one bit left through

the Carry Bit. The 7 bit now occupies the Carry Bit (1) and the Carry Bit now occupies the 0 bit: 00 010 001.

RAR (ROTATE ACCUMULATOR RIGHT THROUGH CARRY) 00 011 111

Operation: The accumulator byte is rotated one bit position to the right through the Carry Bit. The 0 bit position now occupies the Carry Bit and the Carry Bit occupies the 7 bit position.

Status Bits Affected: Carry.

Example: Assume the accumulator byte is 10 001 000, the Carry Bit is 1, and the RAR instruction is present. The contents of the accumulator are rotated one bit position right through the Carry Bit. The 0 bit now occupies the Carry Bit, and the Carry Bit now occupies the 7 bit: 11 000 100.

E. DATA TRANSFER INSTRUCTIONS

Data can be conveniently transferred between registers or between the memory and registers of the *ALTAIR 8800*. Certain of these operations are direct data transfers and no other operation is involved. For example, the MOV instruction causes a byte of data to be transferred from one register (the source register) to another register (the destination register). Other data transfers are accompanied by an arithmetic or logical operation. For example, the ADD instruction adds the contents of a specified register to the contents of the accumulator.

Still another class of data transfer instructions concerns only the accumulator and the H and L register pair. For example, the STA instruction causes the contents of the accumulator to replace the byte of data stored at a specified memory address.

This section describes fifteen separate data transfer instructions, but it is important to note that many other instructions also involve the transfer of data (e.g. PUSH, POP, DAD, XCHG, XTHL, SPHL, etc.). However, it is more appropriate to the efficient organization of this operating manual to describe these instructions elsewhere.

The data transfer instructions described in this section are grouped into three subdivisions. The first subdivision is Data Transfers (MOV, STAX, and LDAX). The second is Register/Memory to Accumulator Transfers (ADD, ADC, SUB, SBB, ANA, XRA, ORA, and CMP). And the third is Direct Addressing Transfers (STA, LDA, SHLD, and LHLD).

1. DATA TRANSFER INSTRUCTIONS

There are three data transfer instructions and each is unconditional. Each of the three instructions has at least two variances. The variances are determined by register or memory addresses which are specified by the programmer.

MOV (MOVE DATA) 01 DDD SSS (Byte 1)

Operation: The contents of SSS (the source register) are moved to DDD (the destination register). The contents of SSS remain unchanged. The following bit patterns for the source and destination registers apply:

Register	Bit Pattern
B	000
C	001
D	010
E	011
H	100
L	101
Memory Reference M	110
A	111

The source and destination registers cannot both equal 110.

Status Bits: Unaffected.

Example: Assume it is necessary to transfer the contents of register E to the accumulator. By referring to the register bit pattern table provided above, an appropriate MOV instruction can be formulated: 01 111 011.

STAX (STORE ACCUMULATOR) 00 0X0 010 (Byte 1)

Operation: The contents of the accumulator are stored in a memory address specified by registers B and C or registers D and E. Registers B and C are specified by a 0 at the 4 bit position (X). Registers D and E are specified by a 1 at the 4 bit position (X).

Status Bits: Unaffected.

Example: Assume it is necessary to store the contents of the accumulator at a memory address specified by registers D and E. The appropriate STAX instruction is: 00 010 010.

LDAX (LOAD ACCUMULATOR) 00 0X1 010 (Byte 1)

Operation: The contents of the memory address specified by registers B and C or by registers D and E replace the contents of the accumulator. Registers B and C are specified by a 0 at the 4 bit position (X). Registers D and E are specified by a 1 at the 4 bit position (X).

Status Bits: Unaffected.

Example: Assume it is necessary to load the accumulator with the contents of a memory address specified by registers B and C. The appropriate LDAX instruction is: 00 001 010.

2. REGISTER/MEMORY TO ACCUMULATOR TRANSFERS

There are eight Register/Memory to Accumulator Transfers and each is unconditional. Each of the eight instructions has eight variances determined by registers specified by the programmer. The following bit patterns for each of the registers apply:

Register	Bit Pattern
B	000
C	001
D	010
E	011
H	100
L	101
Memory Address M	110
A	111

Four of the instructions involve arithmetic (add or subtract) operations. The remaining four involve logical operations.

ADD (ADD REGISTER/ACCUMULATOR TO MEMORY) 10 000(reg) (Byte 1)

Operation: The contents of the specified register (reg) are added to the contents of the accumulator.

Status Bits Affected: Carry, Sign, Zero, Parity, and Auxiliary Carry.

Example: Assume it is necessary to add the contents of register B to the accumulator. Referring to the register bit pattern table given above, the appropriate instruction

is: 10 000 000. If the data bytes at register B and the accumulator are 11 010 100 and 01 100 010 respectively, the following addition will be performed:

```
    11 010 100    Register B Byte
    01 100 010    Accumulator Byte
    ----------
   100 110 110    New Accumulator Byte
```

Since the new accumulator byte has nine bits, the Carry Bit will be set to 1 to indicate a carry has occurred.

ADC (ADD REGISTER/MEMORY AND CARRY TO ACCUMULATOR) 10 001 (reg)

Operation: The contents of the specified register (reg) and the content of the Carry Bit are added to the accumulator.

Status Bits Affected: Carry, Sign, Zero, Parity, and Auxiliary Carry.

Example: Assume it is necessary to add the contents of register C and the content of the Carry Bit to the accumulator. Referring to the register bit pattern table given above, the appropriate instruction is: 10 001 001. If the data bytes at register C and the accumulator are 00 100 011 and 01 011 100 and the Carry Bit is 1, the following addition will be performed:


```
    00 100 011    Register C Byte
    01 011 100    Accumulator Byte
           1      Carry Bit
    ----------
    10 000 000    New Accumulator Byte
```

If the new accumulator byte had nine bits, the extra bit would set the Carry Bit to 1.

SUB (SUBTRACT REGISTER/MEMORY FROM ACCUMULATOR) 10 010 (reg)

Operation: The contents of the specified register are subtracted from the contents of the accumulator. The *ALTAIR 8800* achieves subtraction by means of a simple addition process called two's complement arithmetic. If there are only

eight bits in the result, no carry bit is present. This means a borrow occurred, and the Carry Bit is set to 1. Note that this operation is the inverse of what occurs in an ADD instruction.

Status Bits Affected: Carry Sign, Zero, Parity, and Auxiliary Carry.

Example: Assume it is necessary to clear the accumulator of its contents. An efficient way to achieve this requirement is to implement a SUB A instruction (10 010 111) where A specifies the accumulator variance of the SUB instruction. Implementation of this instruction will cause the contents of the accumulator to be subtracted from itself.

SBB (SUBTRACT REGISTER/MEMORY FROM ACCUMULATOR WITH BORROW)

10 011 (reg) (Byte 1)

Operation: The content of the Carry Bit is added to the contents of the specified register and the result is then subtracted from the accumulator using two's complement arithmetic.

Status Bits Affected: Carry, Sign, Zero, Parity, and Auxiliary Carry.

Example: Assume that the SBB instruction is implemented for the B variance (SBB B). The contents of register B will be added to the carry bit, and the result will then be subtracted from the accumulator. Status bits will be set or reset as appropriate.

ANA (LOGICAL AND REGISTER/MEMORY WITH ACCUMULATOR) 10 100 (reg)

Operation: The content of the specified register is logically ANDed with the contents of the accumulator. The Carry Bit is reset to 0.

Status Bits Affected: Carry, Zero, Sign, and Parity.

Example: Assume the content of register L is 10 001 100 and the content of the accumulator is 10 000 101. An ANA instruction will then cause the contents of the two registers to be ANDed with one another bit-by-bit. Since the logical ANDing of two bits is 1 only if both bits are 1, the following procedure occurs:

```
10 001 100    Register L
10 000 101    Accumulator
-----------
10 000 100    Register L AND Accumulator
```

XRA (LOGICAL EXCLUSIVE-OR REGISTER/MEMORY WITH ACCUMULATOR)

10 101 (reg)

Operation: The content of the specified register is logically EXCLUSIVE ORed with the contents of the accumulator. The Carry Bit is reset to 0.

Status Bits Affected: Carry, Sign, Zero, and Parity.

Example: Since the EXCLUSIVE-ORing of two bits is 1 only if the values of the bits are different, the XRA instruction can be used to clear the accumulator to 0. This function is implemented by means of the instruction XRA and the variance A. The resulting statement is 10 101 111 (see the table of register bit patterns given above).

The XRA instruction can also be used to monitor the status of individual bits in a byte which has been designated a condition byte. For example, assume a byte has been designated to record eight separate true-false conditions wherein a 1 is true and a 0 is false. In order to check whether or not any of the conditions have changed, the original data byte can be moved to the accumulator and EXCLUSIVE-ORed with the updated data byte. Conditions which have not changed will produce a 0 bit and conditions which have changed will produce a 1 bit.

ORA (LOGICAL OR REGISTER/MEMORY WITH ACCUMULATOR) 10 110 (reg)

Operation: The content of the specified register is logically ORed with the content of the accumulator. The Carry Bit is reset to zero.

Status Bits Affected: Carry, Zero, Sign, and Parity.

Example: Since the ORing of two bits is 0 only if the value of each bit is 0, the ORA instruction can be used to set a group of bits to a series of 1s.

CMP (COMPARE REGISTER/MEMORY WITH ACCUMULATOR) 10 111 (reg)

Operation: The content of the specified register is compared with the content of the accumulator by subtracting the former from the latter. The contents of the register and accumulator are unaffected by this operation, and the status bits are set or reset as appropriate.

Status Bits Affected: Carry, Sign, Zero, and Parity (Note: The sense of the Carry Bit is reversed if one byte is plus and the other is minus).

Example: The CMP instruction is useful in determining when the content of any particular register equals that of the accumulator. If the two bytes are equal, the subtraction will give a 0 result, and the Zero Status Bit will be set to 1. If the register contents are greater than the accumulator contents, the Carry Bit will be set to 1 since a subtraction has occurred. If the register contents are less than the accumulator contents, the Carry Bit will be reset to 0.

3. DIRECT ADDRESSING INSTRUCTIONS

The four instructions described in this section are used to store the contents of the accumulator and the H and L registers in the memory or to load the accumulator and H and L registers with data from the memory. All four instructions require three bytes. The first byte is the specific instruction, and the second and third bytes provide the memory address.

STA (STORE ACCUMULATOR DIRECT) 00 110 010 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: The contents of the accumulator are stored in the memory at the address specified in bytes 2 and 3.

Status Bits: Unaffected.

Example: Assume the accumulator byte is 00 010 110 and a STA instruction is present with the following memory address:

01 000 000 (Byte 2)

01 000 001 (Byte 3)

The accumulator byte will then be stored at this memory address.

LDA (LOAD ACCUMULATOR DIRECT) 00 111 010 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: The accumulator is loaded with the contents of the byte at the memory address given by bytes 2 and 3 of the instruction.

Status Bits: Unaffected.

Example: The inverse of the example given in the STA instruction will illustrate operation of the LDA instruction.

SHLD (STORE H AND L DIRECT) 00 100 010 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: The contents of the L register are stored in the memory at the address specified in bytes 2 and 3. The contents of the H register are stored in the memory at the next higher address.

Status Bits: Unaffected.

Example: Assume the L register byte is 00 101 100, the H register byte is 00 101 111, and an SHLD instruction is present with the following address:

01 000 101 (Byte 2)

01 110 101 (Byte 3)

The L register byte will then be stored at this memory address, and the H register byte will be stored at the next highest address.

LHLD (LOAD H AND L DIRECT) 00 101 010 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: The L register is loaded with the contents of the byte at the memory address given by bytes 2 and 3. The H register is loaded with the contents of the byte at the next higher memory address.

Status Bits: Unaffected.

Example: The inverse of the example given in the SHLD instruction will illustrate operation of the LHLD instruction.

F. IMMEDIATE INSTRUCTIONS

The *ALTAIR 8800* has ten immediate instructions. These instructions cause the computer to process one or two bytes of data which form a part of the instruction. Immediate instructions are available to load two bytes of data into a specified register pair, move one byte of data into a specified register or memory address, and to perform arithmetic and logical operations with the contents of the accumulator and one byte of immediate data.

A typical byte of immediate data is a mathematical constant such as pi. Immediate data can also be a number or quantity specified by the programmer such as an actual or projected inventory count. For example, a program utilizing one or more immediate instructions will permit the computer to compare the actual inventory of a particular product with the desired inventory. At any inventory count specified in the program, the computer can notify the programmer or operator of the need to reorder.

LXI	(LOAD REGISTER PAIR IMMEDIATE)	00 (rp)0 001	(Byte 1)
		(Data)	(Byte 2)
		(Data)	(Byte 3)

Operation: Two bytes of immediate data are loaded into the register pair specified rp in byte 1 of the instruction. The first byte of data (the least significant 8 bits) is loaded into the second register of the specified pair, and the second byte of data (the most significant 8 bits) is loaded into the first register of the specified pair. This procedure is reversed if the Stack Pointer is the specified register pair. The bit patterns for the register pairs are as follows:

00	Registers B and C
01	Registers D and E
10	Registers H and L
11	Stack Pointer

Status Bits: Unaffected.

Example: The following LXI instruction is inputed to the computer:

00 010 001 (Byte 1)

01 111 111 (Byte 2)

01 111 110 (Byte 3)

Bit positions 4 and 5 of byte 1 specify that the data in bytes 2 and 3 is to be loaded into registers D and E. Byte 2 is loaded into D and byte 3 is loaded into E.

MVI (MOVE IMMEDIATE DATA) 00 (reg)110 (Byte 1)

(Data) (Byte 2)

Operation: One byte of immediate data is moved into the specified register or memory byte. The following register bit patterns apply:

Register	Bit Pattern
B	000
C	001
D	010
E	011
H	100
L	101
Memory Address M	110
A	111

Status Bits: Unaffected.

Example: The following MVI instruction is inputed to the computer:

00 011 110 (Byte 1)

11 111 111 (Byte 2)

The immediate data in byte 2 is moved into register E.

ADI (ADD IMMEDIATE TO ACCUMULATOR) 11 000 110 (Byte 1)

(Data) (Byte 2)

Operation: The immediate data in byte 2 is added to the contents of the accumulator.

Status Bits Affected: Carry, Sign, Zero, Parity, and Auxiliary Carry.

Example: Assume the accumulator byte is 11 110 000 and the ADI instruction is present. The immediate data in the ADI instruction is 10 000 000. Implementation of the ADI instruction will leave 01 110 000 in the accumulator and the Carry Bit will be set to 1. All other status bits will be reset.

ACI (ADD IMMEDIATE AND CARRY TO ACCUMULATOR) 11 001 110 (Byte 1)

(Data) (Byte 2)

Operation: The data in byte 2 and the content of the Carry Bit are added to the contents of the accumulator.

Status Bits Affected: Carry, Sign, Zero, Parity, and Auxiliary Carry.

Example: Assume the accumulator byte is 11 110 000, the Carry Bit is set to 1, and the ACI instruction is present. The immediate data in the ACI instruction is 00 101 100. Implementation of the ACI instruction will leave the sum 00 011 101 in the accumulator and both the Carry and Parity Bits will be set to 1. The remaining status bits will be reset to 0.

SUI (SUBTRACT IMMEDIATE FROM ACCUMULATOR) 11 010 110 (Byte 1)

(Data) (Byte 2)

Operation: The data in byte 2 is subtracted from the contents of the accumulator using two's complement arithmetic. Since the *ALTAIR 8800* implements subtraction by means of addition, the Carry Bit is set to 1 if no carry occurred since this means a borrow occurred. If a borrow did not occur, a carry did occur, and the Carry Bit is reset to 0. Note that this operation is the reverse of what occurs in an ADI or ACI instruction.

Status Bits Affected: Carry, Sign, Zero, Parity, and Auxiliary Carry.

Example: Assume it is necessary to subtract 00 000 100 from the accumulator. The resulting instruction would be as follows:

11 010 110 (Byte 1)

00 000 100 (Byte 2)

If the accumulator byte is 00 001 010, implementation of the SUI instruction will leave 00 000 110 in the accumulator. Since this is a subtraction operation and no carry is present, a borrow occurred and the Carry Bit is set to 1. The Parity Bit is also set to 1, and the remaining status bits are reset to 0.

SBI (SUBTRACT IMMEDIATE PLUS CARRY FROM ACCUMULATOR) 11 011 110

(Data)

Operation: The data in byte 2 is added to the content of the Carry Bit and the result is subtracted from the accumulator using two's complement arithmetic.

Status Bits Affected: Carry, Sign, Zero, Parity, and Auxiliary Carry.

Example: Assume it is necessary to implement the SBI instruction. The contents of the data byte will then be added to the Carry Bit and the result subtracted from the accumulator. Since this is a subtraction operation, the Carry Bit will be set to 1 if no carry occurred (meaning a borrow occurred) and reset to 0 if a carry occurred (meaning a borrow did not occur).

ANI (AND IMMEDIATE WITH ACCUMULATOR) 11 100 110 (Byte 1)

(Data) (Byte 2)

Operation: The contents of the data byte are logically ANDed with the contents of the accumulator. The Carry Bit is reset to 0.

Status Bits Affected: Carry, Sign, Zero, and Parity.

Example: Assume the content of the data byte is 00 111 011 and the content of the accumulator is 11 101 110. An ANI instruction will then cause the contents of both bytes to be ANDed together bit-by-bit. Since the logical ANDing of two bits is 1 only if both bits are 1, the following procedure occurs:

```
00 111 011      (Data Byte)
11 101 110      (Accumulator)
-----------
00 101 010      Data Byte AND Accumulator
```

XRI (EXCLUSIVE-OR IMMEDIATE WITH ACCUMULATOR) 11 101 110 (Byte 1)

(Data) (Byte 2)

Operation: The data in byte 2 of the instruction is EXCLUSIVE-ORed with the accumulator byte. The Carry Bit is reset to 0.

Status Bits Affected: Carry, Sign, Zero, and Parity.

Example: A bit is unchanged when EXCLUSIVE-ORed with a 0 and complemented when EXCLUSIVE-ORed with a 1. Therefore the EXCLUSIVE-ORed function can be used to complement any or all of the bits in the accumulator. For example, to complement all but the 7 position bit in the accumulator would require the following data byte: 01 111 111. If the accumulator byte is 10 110 001, the following operation will occur upon implementation of the XRI instruction:


```
01 111 111      (Data Byte)
10 110 001      (Accumulator)
-----------
11 001 110      Data Byte EXCLUSIVE-OR
                Accumulator
```

ORI (LOGICAL OR IMMEDIATE WITH ACCUMULATOR) 11 110 110 (Byte 1)

(Data) (Byte 2)

Operation: The data in byte 2 of the instruction is logically ORed with the accumulator byte. The Carry Bit is reset to 0.

Status Bits Affected: Carry, Sign, Zero, and Parity.

Example: The ORI instruction can be used to add 1 to the accumulator. Assume the accumulator byte is 10 000 100 and an ORI instruction is present. Since the ORing of two bits produces a 0 only if the value of the two bits is 0, the data byte 00 000 001 will add 1 to the accumulator if the 0 position bit is 0. Otherwise the accumulator byte will be unchanged.

CPI (COMPARE IMMEDIATE WITH ACCUMULATOR) 11 111 110 (Byte 1)

(Data) (Byte 2)

Operation: The data in byte 2 of the instruction is compared with the content of the accumulator by subtracting the former from the latter. The contents of the accumulator and data byte are unaffected by this operation, and the Status Bits are set or reset as appropriate.

Status Bits Affected: Carry, Zero, Sign, Parity, and Auxiliary Carry.

70 Example: The CPI instruction is useful in determining when the content of the accumulator equals that of the data byte. If the two bytes are equal, the subtraction process will give a 0 result, and the Zero Status Bit will be set to 1. If the data byte contents are greater than the accumulator contents, the Carry Bit will be set to 1 since a subtraction has occurred. If the Data byte contents are less than the accumulator contents, the Carry Bit will be reset to 0.

G. BRANCHING INSTRUCTIONS

The *ALTAIR 8800* has an extensive branching capability. Branching permits the computer to jump from one step in the program to another. Branching also permits the computer to call a specified set of instructions from memory and insert it into the program. A return feature permits the computer to resume normal operation after the specified instruction set is executed.

Branching is one of the most important capabilities of a computer. Jumping from one point in the program to another, for example, saves time, and calling a special set of instructions from memory means a frequently used instruction sequence need be stored at only one place in memory. The result is an important increase in computer processing speed and efficiency. Branching also adds to the economy of a computer since less memory is required to accomplish complex programs. And the ability to call frequently used instruction sets from memory can save considerable programming time.

The term subroutine is used to describe a special set of instructions stored in memory. Typical subroutines might include instruction sets for calculating trigonometric functions and square roots or making complex logical comparisons. Each of these subroutines can be quite lengthy. If a program requires a dozen or more trigonometric operations and several square root extractions, it is obvious that the use of subroutines can save considerable programming time and memory space.

Branching instructions can be either conditional or unconditional. A conditional branch means a particular branching operation is accomplished only if a specified condition is met. For example, a typical conditional branch instruction is CZ (CALL IF ZERO). If the zero bit is indeed zero when the CZ instruction is processed, the Program Counter will automatically move to the address in memory specified in the two address bytes which follow the CZ instruction in the program.

Unconditional branching causes a branch to occur without the necessity for meeting certain specified conditions.

Branching instructions require either one or three bytes per instruction. The first byte is the actual instruction while the second and third bytes are, respectively, the low and high memory addresses. The address bytes tell the Program

Counter where to move. The instructions which require only one byte need no memory addresses since some of the bits in the byte refer the Program Counter to certain registers or the Stack Pointer, either of which contains the necessary addressing information.

1. JUMP INSTRUCTIONS

JUMP instructions permit the normal execution sequence of a program to be either conditionally or unconditionally altered. For example, a program might include a set of instructions to be executed if the result of a previous operation is greater than zero. If, however, the result is zero, the set of instructions becomes superfluous and unnecessary. The program, therefore, includes a JUMP statement which instructs the computer to advance to any specified address past the instruction set. Since the jump would be implemented only if the result of the preceeding operation were zero, this would be a conditional branching operation. The actual machine language mnemonic for this particular instruction is JZ (JUMP IF ZERO).

All but one of the ten JUMP instructions require three bytes. The first byte is the specific machine language instruction, while the second and third bytes are, respectively, the low and high memory addresses for the portion of the program to be selected by the Program Counter if a jump is implemented. The PCHL instruction requires only the initial machine language instruction byte since the memory locations to which the program jumps are known by the computer. The memory locations in this case happen to be the H and L Registers, the contents of which are placed into the Program Counter.

With the exception of the PCHL and JMP instructions, all JUMP instructions are conditional. If a specified condition is true, the Program Counter automatically advances to the address specified in the instruction. If the specified condition is not true, the program continues its sequential execution and a jump does not occur.

PCHL (LOAD PROGRAM COUNTER) 11 101 001 (Byte 1)

Operation: The Program Counter jumps to the Memory address specified by the contents of the H and L Registers. The most significant 8 bits of the Program Counter are loaded with the contents of the H Register and the least significant 8

bits of the Program Counter are loaded with the contents of the L Register.

Status Bits: Unaffected.

Example: Assume the contents of the H and L Registers are as follows:

H: 10 111 000

L: 11 010 110

Instruction PCHL will automatically transfer this Memory address to the Program Counter as shown below:

	Most Significant	Least Significant
Program Counter:	10 111 000	11 010 110

The program will now continue to execute after having jumped to the new address specified in the Program Counter.

JMP (JUMP) 11 000 011 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: The Program Counter jumps unconditionally to the Memory address specified in bytes 2 and 3 and the program continues to execute from the new location.

Status Bits: Unaffected.

Example: Assume the JMP instruction and address bit pattern is as follows:

11 000 011 (Byte 1)

10 111 000 (Byte 2)

11 010 110 (Byte 3)

The Program Counter will jump to the address in Memory specified by bytes 2 and 3 and program execution will continue from the new address.

JC (JUMP IF CARRY) 11 011 010 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Carry Bit is 1, a carry has occurred and the Program Counter jumps to the address specified in bytes 2 and 3. Program execution then continues from the new address. If the Carry Bit is 0, no carry has occurred and the program continues sequential execution.

Status Bits: Unaffected.

Example: Assume the Carry Bit is 1 and a JC instruction is present. The Program Counter will then jump to the address specified in bytes 2 and 3 and the program will continue at the new address.

JNC (JUMP IF NO CARRY) 11 010 010 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Carry Bit is 0, no carry has occurred, and the Program Counter jumps to the address specified in bytes 2 and 3. Program execution then continues from the new address. If the Carry Bit is 1, a carry has occurred and the program continues sequential execution.

Status Bits: Unaffected.

Example: The inverse of the example provided under the JC instruction will illustrate operation of the JNC instruction.

JZ (JUMP IF ZERO) 11 001 010 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Zero Bit is 1, a zero is present and the Program Counter jumps to the address specified in bytes 2 and 3.