Program execution then continues from the new address. If the Zero Bit is 0, a zero is not present and the program continues sequential operation.

Status Bits: Unaffected.

Example: Assume the Zero Bit is 1 (zero present) and a JZ 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.

JNZ (JUMP IF NOT ZERO)	11 000 010	(Byte 1)
	(Low Address)	(Byte 2)
	(High Address)	(Byte 3)

Operation: This is a conditional instruction. If the status of the Zero Bit is 0 (zero not present) and a JNZ instruction is present, the Program Counter jumps to the address specified in bytes 2 and 3. Program execution then continues from the new address. If the Zero Bit is 1, a zero is present, and the program continues sequential operation.

Status Bits: Unaffected.

Example: The inverse of the example provided under the JZ instruction will illustrate operation of the JNZ instruction.

JM (JUMP IF MINUS)	11 111 010	(Byte 1)
	(Low Address)	(Byte 2)
	(High Address)	(Byte 3)

Operation: This is a conditional instruction. If the status of the Sign Bit is 1 (a negative result), the Program Counter jumps to the address specified in bytes 2 and 3. Program execution then continues from the new address. If the Sign Bit is 0, the result is positive and the program continues sequential operation.

Status Bits: Unaffected.

Example: Assume the Sign Bit is 1 indicating a negative result and the JM instruction is present. The Program Counter will then jump to the address specified in bytes 2 and 3 of the instruction and the program will continue at the new address.

JP (JUMP IF POSITIVE) 11 110 010 (Byte 1)
(Low Address) (Byte 2)
(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Sign Bit is 0 (a positive result), the Program Counter jumps to the address specified in bytes 2 and 3. Program execution then continues from the new address. If the Sign Bit is 1, the result is negative and the program continues sequential operation.

Status Bits: Unaffected.

Example: The inverse of the example provided under the JM instruction will illustrate operation of the JP instruction.

JPE (JUMP IF PARITY IS EVEN) 11 101 010 (Byte 1)
(Low Address) (Byte 2)
(High Address) (Byte 3`

Operation: This is a conditional instruction. If the status of the Parity Bit is 1 (a result with even parity), the Program Counter jumps to the address specified in bytes 2 and 3. Program execution then continues from the new address. If the Parity Bit is 0, the parity is odd and the program continues sequential operation.

Status Bits: Unaffected.

Example: Assume the Parity Bit is 1 indicating the result has even parity and the JPE instruction is present. The Program Counter will jump to the address specified in bytes 2 and 3 and the program will continue at the new address.

JPO (JUMP IF PARITY ODD) 11 100 010 (Byte 1)
(Low Address) (Byte 2)
(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Parity Bit is 0 (a result with odd parity), the Program Counter jumps to the address specified by bytes 2 and 3. Program execution then continues from the new address. If the Parity Bit is 1, the parity is even and the program continues sequential operation.

Status Bits: Unaffected.

Example: The inverse of the example provided under the JPE instruction will illustrate operation of the JPO instruction.

2. CALL INSTRUCTIONS

CALL instructions cause a program to execute a subroutine stored at a specified location in memory. The CALL instruction may be either conditional or unconditional. Many subroutines are called unconditionally. For example, the calculation sequence for extracting a square root is relatively lengthy. In a program which requires frequent square root extractions, considerable programming time and memory space can be saved by writing a single square root extraction subroutine. This subroutine can then be stored in memory and called by the program each time it is needed.

Conditional CALL instructions are available also. They permit a great deal of flexibility since the programmer can instruct the computer to make logical decisions about the status of the program at any specified point. A subroutine can then be called if a specified condition is met.

When a subroutine has been executed, the Program Counter returns to the next step in the main program by means of a special RETURN instruction. This instruction is described in the next section.

All the CALL 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 first instruction of the subroutine.

CALL (CALL)	11 001 101	(Byte 1)
	(Low Address)	(Byte 2)
	(High Address)	(Byte 3)

Operation: The Program Counter unconditionally moves to the Memory address specified in bytes 2 and 3. The subroutine at the new location is then executed.

Status Bits: Unaffected.

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

```
11 001 101      (Byte 1)

10 101 111      (Byte 2)

11 111 010      (Byte 3)
```

The Program Counter will move to the address in Memory specified by bytes 2 and 3 and the subroutine at that location will then be executed.

```
CC    (CALL IF CARRY)                 11 011 100  (Byte 1)

                                  (Low Address)  (Byte 2)

                                  (High Address) (Byte 3)
```

Operation: This a conditional instruction. If the status of the Carry Bit is 1, a carry has occurred and the Program Counter moves to the address specified in bytes 2 and 3. The subroutine at this location is then executed. 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 the CC instruction is present. The Program Counter will then jump to the address specified in bytes 2 and 3 and the subroutine at that location will be executed.

```
CNC   (CALL IF NO CARRY)              11 010 100  (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, a carry has not occurred, and the

Program Counter moves to the address specified in bytes 2 and 3. The subroutine at that location is then executed. If the Carry Bit is 1, a carry has occurred, and the Program Counter continues sequential execution.

Status Bits: Unaffected.

Example: The inverse of the example provided under the CC instruction will illustrate operation of the CNC instruction.

CZ (CALL IF ZERO) 11 001 100 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Zero Status Bit is 1, a zero is present, and the Program Counter moves to the address specified in bytes 2 and 3. The subroutine at this location is then executed. If the Zero Status Bit is 0, no zero is present, and the program continues sequential execution.

Status Bits: Unaffected.

Example: Assume the Zero Status Bit is 1 and the CZ instruction is present. The Program Counter will then move to the address specified in bytes 2 and 3, and the subroutine at that location will be executed.

CNZ (CALL IF NOT ZERO) 11 000 100 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Zero Status Bit is 0, a zero is not present, and the Program Counter moves to the address specified in bytes 2 and 3. The subroutine at this location is then executed. If the Zero Status Bit is 1, a zero is present, and the program continues sequential execution.

Status Bits: Unaffected.

Example: The inverse of the example provided under the CZ instruction will illustrate operation of the CNZ instruction.

CM (CALL IF MINUS) 11 111 100 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Sign Bit is 1 (a negative result), the Program Counter moves to the address specified in bytes 2 and 3. The subroutine at this location is then executed. If the Sign Bit is 0, the result is positive, and the program continues sequential execution.

Status Bits: Unaffected.

Example: Assume the Sign Bit is 1 and the CM instruction is present. The Program Counter will then move to the address specified in bytes 2 and 3, and the subroutine at that location will be executed.

CP (CALL IF PLUS) 11 110 100 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Sign Bit is 0 (a positive result), the Program Counter moves to the address specified in bytes 2 and 3. The subroutine at this location is then executed. If the Sign Bit is 1, the result is negative, and the program continues sequential execution.

Status Bits: Unaffected.

Example: The inverse of the example provided under the CM instruction will illustrate operation of the CP instruction.

CPE (CALL IF PARITY EVEN) 11 101 100 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Parity Bit is 1 (a result with even parity), the Program Counter moves to the address specified in bytes 2 and 3. The subroutine at this location is then executed. If the Parity Bit is 0, the parity is odd, and the program continues sequential execution.

Status Bits: Unaffected.

Example: Assume the status of the Parity Bit is 1 and a CPE instruction is present. The Program Counter will then move to the address specified in bytes 2 and 3, and the subroutine at that location will be executed.

CPO (CALL IF PARITY ODD) 11 100 100 (Byte 1)

(Low Address) (Byte 2)

(High Address) (Byte 3)

Operation: This is a conditional instruction. If the status of the Parity Bit is 0 (a result with odd parity), the Program Counter moves to the address specified in bytes 2 and 3. The subroutine at this location is then executed. If the Parity Bit is 1, the parity is even, and the program continues sequential execution.

Status Bits: Unaffected.

Example: The inverse of the example provided under the CPE instruction will illustrate operation of the CPO instruction.

3. RETURN INSTRUCTIONS

When a CALL subroutine instruction is executed, the address of the next sequential instruction in the program is automatically pushed onto the stack. The subroutine may have one or more RETURN statements. An unconditional RETURN instruction is included at the end of most subroutines. This instruction pops the last address stored in the stack by the CALL instruction from the stack and onto the Program Counter. When the subroutine has been executed, the program resumes sequential execution at the address following the initial CALL subroutine instruction.

Conditional RETURN instructions may be scattered throughout a subroutine. If the required condition is met, the program resumes sequential execution in the manner just described.

Since the program address to which the Program Counter returns upon receiving a RETURN instruction is already stored on the stack, RETURN instructions require only one byte. The last bit in the byte is 1 for an unconditional RETURN and 0 for conditional RETURNS.

RET (RETURN) 11 001 001 (Byte 1)

Operation: The subroutine is completed, and the Program Counter automatically and unconditionally returns to the next address following the initial CALL subroutine instruction.

Status Bits: Unaffected.

Example: Assume two of the instruction statements in an *ALTAIR 8800* program are as follows:

CALL	11 001 101	(Byte 1)
	(Low Address)	(Byte 2)
	(High Address)	(Byte 3)
CMA	00 101 111	(Byte 1)

Upon receiving the CALL instruction, the Program Counter moves to the address in Memory specified by bytes 2 and 3. Simultaneously, the address of the next sequential instruction (CMA) is pushed onto the stack.

The final instruction in the subroutine must be an unconditional RETURN (only if you wish to return). When execution of the subroutine is complete and the RET instruction is reached, the Program Counter automatically receives the address of the next instruction in the main program from the stack (CMA), and sequential execution resumes.

RC (RETURN IF CARRY) 11 011 000 (Byte 1)

Operation: This is a conditional instruction which may be inserted before the end of a subroutine. If the status of the Carry Bit is 1, a carry has occurred and the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction.

Status Bits: Unaffected.

Example: Assume three of the instructions in a subroutine are as follows:

RAL	00 10 111	(Byte 1)
RC	11 011 000	(Byte 1)
STAX	00 000 010	(Byte 1)

If the status of the Carry Bit is 1 when the RC instruction is reached, a carry has occurred and the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction. If the status of the Carry Bit is 0, the subroutine continues sequential execution by implementing the STAX instruction.

RNC (RETURN IF NO CARRY) 11 010 000 (Byte 1)

Operation: This is a conditional instruction which may be inserted before the end of a subroutine. If the status of the Carry Bit is 0, a carry has not occurred and the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction. If the status of the Carry Bit is 1, a carry has occurred, and the subroutine continues sequential execution.

Status Bits: Unaffected.

Example: The inverse of the example provided under the RC instruction will illustrate operation of the RNC instruction.

RZ (RETURN IF ZERO) 11 001 000 (Byte 1)

Operation: This is a conditional instruction which may be inserted before the end of a subroutine. If the status of the Zero Status Bit is 1, a 0 is present and the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction. If the status of the Zero Status Bit is 0, a zero is not present and the subroutine continues sequential execution.

Status Bits: Unaffected.

Example: Assume three of the instructions in a subroutine are as follows:

ADD	10 000 101	(Byte 1)
RZ	11 001 000	(Byte 1)
LDAX	00 011 010	(Byte 1)

If the status of the Zero Status Bit is 1 when the RZ instruction is reached, a zero result is present and the Program Counter automatically returns to the next sequential address in the main program following the initial CALL instruction. If the status of the Zero Status Bit is 0, the subroutine continues execution by implementing the LDAX instruction.

RNZ (RETURN IF NOT ZERO) 11 000 000 (Byte 1)

Operation: This is a conditional instruction which may be inserted before the end of a subroutine. If the status of the Zero Status Bit is 0, a zero result is not present and the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction. If the status of the Zero Status Bit is 1, a zero result is present, and the subroutine continues sequential execution.

Status Bits: Unaffected.

Example: The inverse of the example provided under the RZ instruction will illustrate operation of the RNZ instruction.

RM (RETURN IF MINUS) 11 111 000 (Byte 1)

Operation: This is a conditional instruction which may be inserted before the end of a subroutine. If the status of the Sign Bit is 1 (a negative result), the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction. If the status of the Sign Bit is 0 (a positive result), the subroutine continues sequential execution.

Status Bits: Unaffected.

Example: Assume three of the instructions in a subroutine are as follows:

SUB	10 010 001	(Byte 1)
RM	11 111 000	(Byte 1)
LDAX	00 011 010	(Byte 1)

If the status of the Sign Bit is 1 when the RM instruction is reached, a negative result is present, and the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction. If the status of the Sign Bit is 0, the subroutine continues sequential execution by implementing the LDAX instruction.

RP (RETURN IF PLUS) 11 110 000 (Byte 1)

Operation: This is a conditional instruction which may be inserted before the end of a subroutine. If the status of the Sign Bit is 0 (a positive result), the Program Counter automatically returns to the next sequential address in the program following the initial CALL subroutine instruction. If the status of the Sign Bit is 1 (a negative result), the subroutine continues sequential execution.

Status Bits: Unaffected.

Example: The inverse of the example provided under the RM instruction will illustrate operation of the RP instruction.

RPE (RETURN IF PARITY EVEN) 11 101 000 (Byte 1)

Operation: This is a conditional instruction which may be inserted before the end of a subroutine. If the status of the Parity Bit is 1 (a result with even parity), the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction. If the status of the Parity Bit is 0 (a result with odd parity), the subroutine continues sequential execution.

Status Bits: Unaffected.

Example: Assume three of the instructions in a subroutine are as follows:

CMP	10 111 001	(Byte 1)
RPE	11 101 000	(Byte 1)

RLC 00 000 111 (Byte 1)

If the status of the Parity Bit is 1 when the RPE instruction is reached, the parity of the result is even, and the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction. If the status of the Parity Bit is odd, the subroutine continues sequential execution by implementing the RLC instruction.

RPO (RETURN IF PARITY ODD) 11 100 000 (Byte 1)

Operation: This is a conditional instruction which may be inserted before the end of a subroutine. If the status of the Parity Bit is 0 (a result with odd parity), the Program Counter automatically returns to the next sequential address in the main program following the initial CALL subroutine instruction. If the status of the Parity Bit is 1 (a result with odd parity), the subroutine continues sequential execution.

Status Bits: Unaffected.

Example: The inverse of the example provided under the RPE instruction will illustrate operation of the RPO instruction.

APPENDIX. ALTAIR 8800 INSTRUCTION SET

Definitions:

DDD	Destination Register
SSS	Source Register
rp	Register Pair

Register Designations:

Register (SSS or DDD)	Bit Pattern
B	000
C	001
D	010
E	011
H	100
L	101
Memory	110
Accumulator	111

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

A. COMMAND INSTRUCTIONS

1. Input/Output Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
In	2	3	11 011 011	333
Out	2	3	11 010 011	323

2. Interrupt Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
EI	1	1	11 111 011	373
DI	1	1	11 110 011	363
HLT	1	1	01 110 110	166
RST	1	3	11 exp 111	3(exp)7

3. Carry Bit Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
CMC	1	1	00 111 111	077
STC	1	1	00 110 111	067

4. No Operation Instruction

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
NOP	1	1	00 000 000	000

B. SINGLE REGISTER INSTRUCTIONS

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
INR	1	3	00 DDD 100	0(DDD)4
DCR	1	3	00 DDD 101	0(DDD)5
CMA	1	1	00 101 111	057
DAA	1	1	00 100 111	047

APPENDIX. ALTAIR 8800 INSTRUCTION SET

Definitions:

DDD	Destination Register
SSS	Source Register
rp	Register Pair

Register Designations:

Register (SSS or DDD)	Bit Pattern
B	000
C	001
D	010
E	011
H	100
L	101
Memory	110
Accumulator	111

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

A. COMMAND INSTRUCTIONS

1. Input/Output Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
In	2	3	11 011 011	333
Out	2	3	11 010 011	323

2. Interrupt Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
EI	1	1	11 111 011	373
DI	1	1	11 110 011	363
HLT	1	1	01 110 110	166
RST	1	3	11 exp 111	3(exp)7

3. Carry Bit Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
CMC	1	1	00 111 111	077
STC	1	1	00 110 111	067

4. No Operation Instruction

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
NOP	1	1	00 000 000	000

B. SINGLE REGISTER INSTRUCTIONS

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
INR	1	3	00 DDD 100	0(DDD)4
DCR	1	3	00 DDD 101	0(DDD)5
CMA	1	1	00 101 111	057
DAA	1	1	00 100 111	047

C. REGISTER PAIR INSTRUCTIONS

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
PUSH	1	3	11 (rp)0 101	3(rp)5
POP	1	3	11 (rp)0 001	3(rp)1
DAD	1	3	00 (rp)1 001	0(rp)1
INX	1	1	00 (rp)0 011	0(rp)3
DCX	1	1	00 (rp)1 011	0(rp)3
XCHG	1	1	11 101 011	353
XTHL	1	5	11 100 011	343
SPHL	1	1	11 111 001	371

D. ROTATE ACCUMULATOR INSTRUCTIONS

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
RLC	1	1	00 000 111	007
RRC	1	1	00 001 111	017
RAL	1	1	00 010 111	027
RAR	1	1	00 011 111	037

E. DATA TRANSFER INSTRUCTIONS

1. Data Transfer Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
MOV	1	1 or 2	01 DDD SSS	1(DDD)(SSS)
STAX	1	2	00 0X0 010*	0(X)2
LDAX	1	2	00 0X0 010*	0(X)2

*NOTE: Register Pair B and C -- 0 at X
Register Pair D and E -- 1 at X

2. Register/Memory to Accumulator Transfers

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
ADD	1	1	10 000 SSS	20 (SSS)
ADC	1	1	10 001 SSS	21 (SSS)
SUB	1	1	10 010 SSS	22 (SSS)
SBB	1	1	10 011 SSS	23 (SSS)
ANA	1	1	10 100 SSS	24 (SSS)
XRA	1	1	10 101 SSS	25 (SSS)
ORA	1	1	10 110 SSS	26 (SSS)
CMP	1	1	10 111 SSS	27 (SSS)

3. Direct Addressing Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
STA	3	4	00 110 010	062
LDA	3	4	00 111 010	072
SHLD	3	5	00 100 010	042
LHLD	3	5	00 101 010	052

F. IMMEDIATE INSTRUCTIONS

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
LXI	3	3	00 (rp)0 001	0(rp)1
MVI	2	2 or 3	00 SSS 110	0(SSS)6
ADI	2	2	11 000 110	306
ACI	2	2	11 001 110	316
SUI	2	2	11 010 110	326
SBI	2	2	11 011 110	336
ANI	2	2	11 100 110	346
XRI	2	2	11 101 110	356
ORI	2	2	11 110 110	366
CPI	2	2	11 111 110	376

G. BRANCHING INSTRUCTIONS

1. Jump Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
PCHL	1	1	11 101 001	351
JMP	3	3	11 000 011	303
JC	3	3	11 011 010	332
JNC	3	3	11 010 010	322
JZ	3	3	11 001 010	312
JNZ	3	3	11 000 010	302
JM	3	3	11 111 010	372
JP	3	3	11 110 010	362
JPE	3	3	11 101 010	352
JPO	3	3	11 100 010	342

2. Call Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
CALL	3	5	11 001 101	315
CC	3	3 or 5	11 011 100	334
CNC	3	3 or 5	11 010 100	324
CZ	3	3 or 5	11 001 100	314
CNZ	3	3 or 5	11 000 100	304
CM	3	3 or 5	11 111 100	374
CP	3	3 or 5	11 110 100	364
CPE	3	3 or 5	11 101 100	354
CPO	3	3 or 5	11 100 100	344

3. Return Instructions

Mnemonic	Bytes	Cycles	Binary Code	Octal Code
RET	1	3	11 001 001	311
RC	1	1 or 3	11 011 000	330
RNC	1	1 or 3	11 010 000	320
RZ	1	1 or 3	11 001 000	310
RNZ	1	1 or 3	11 000 000	300
RM	1	1 or 3	11 111 000	370
RP	1	1 or 3	11 110 000	360
RPE	1	1 or 3	11 101 000	350
RPO	1	1 or 3	11 100 000	340

SERVICE

Should you have a problem with your computer, it can be returned to MITS for repair. If the unit is still under warranty, any defective part will be replaced free of charge. The purchaser is responsible for all postage. In no case should a unit be shipped back without the outer case fully assembled.

If you need to return the unit to us for any reason, remove the top cover of your computer and secure the cards in their sockets with tape and fill the space between the case top and the cards with packing material. Secure cover and pack the unit in a sturdy cardboard container and surround it on all sides with a thick layer of packing material. You can use shredded newspaper, foamed plastic or excelsior. The packed carton should be neatly sealed with gummed tape and tied with a stout cord. Be sure to tape a letter containing your name and address, a description of the malfunction, and the original invoice (if the unit is still under warranty) to the outside of the box.

Mail the carton by parcel post or UPS--for extra fast service, ship by air parcel post. Be sure to insure the package.

SHIP TO: MITS, Inc.
6328 Linn Ave. N.E.
Albuquerque, New Mexico 87108

All warranties are void if any changes have been made to the basic design of the machine or if the internal workings have been tampered with in any way.

MITS ALTAIR 8800

Price List

January 1, 1976

Part Number	Description	Kit	Assem	Days Delivery
8800	Altair 8800 Computer	$ 439.00	$ 621.00	60
COMTER II	Terminal w/Audio Cassette I/O	780.00	920.00	60
CT-256	Comter 256 Terminal	745.00	885.00	45-60
CT257,8 or 9	Pages 2, 3, or 4 for CT-256	95.00	105.00	45-60
CT-8096	CRT Terminal	TBD	TBD	TBD
88-VLCT	Low Cost Terminal	129.00	169.00	45-60
88-80LP	Line Printer & Controller	1,750.00	1,975.00	60
88-TTY	Teletype ASR-33	1,500.00	1,500.00	60
88-MM	Adds 256 words to 88-MCS	14.00	26.00	30
88-1MCS	1K Static Memory	97.00	139.00	30
88-4MCD	4K Dynamic Memory	195.00	275.00	60
88-DCDD	Disc Controller & 1 Drive	1,480.00	1,980.00	60
88-DISC	Disc Drive in Cabinet	1,180.00	1,600.00	60
88-DMAC	Direct Memory Access Cont.	98.00	149.00	TBD
88-DMAE	Direct Memory I/O Channel	126.00	186.00	TBD
88-DMAI	Direct Memory I/O Channel	123.00	183.00	TBD
88-4PIO	4 Port Parallel I/O	86.00	112.00	30
88-PP	Extra Port on 4PIO	30.00	39.00	30
88-2SIO	2 Port Serial Board (State I/O)	115.00	144.00	30
88-SP	Extra Port for 2SIO Board	24.00	35.00	30
88-EC	Expander Mother Board only	16.00	31.00	30
88-MB	88-EC inc. connectors and card guides	65.00	138.00	30
88-EBC	Expander Cabinet	394.00	485.00	60
88-EXC	Extender Card	57.00	83.00	30
88-ACR	Audio Cassette Record Interface	128.00	174.00	30
88-VI	Vectored Interrupt	126.00	179.00	90
88-RTC	Real Time Clock	53.00	84.00	90
88-PPCB	Prototype PC Board	57.00	84.00	30
88-FAN	Cooling Fan	16.00	20.00	30
88-PMC	PROM Memory Card (no PROM's)	65.00	128.00	60
88-PROM	PROM's (256 x 8 Bytes)	25.00	37.00	60
88-PPC	PROM Programmer Card	CONTACT	FACTORY	90
25DB	I/O Socket for Cabinet Case	11.00	25.00	30
MS-416	MITScope--4 channel scope	127.00	189.00	30
680F	680 MPU Unit (Assem state I/O)	345.00	420.00	60
680T	680 Less Front Panel	280.00		60
680 CPU Bd	CPU Board w/microprocessor chip	195.00	275.00	60
680 PROM	256 x 8-Bit PROM	25.00	37.00	60
680FAN	Peewee Fan Option	16.00	20.00	60
680Socket	680 IC Socket Option	29.00	42.00	60

NOTE Prices, specifications, development and delivery all subject to change without notice.

Suggested 8800 System Prices		Kit	Assem	Days Delivery
System I	ALTAIR Basic I	1,712.00	2,265.00	60
System II	ALTAIR Extended Basic II	1,893.00	2,566.00	60
System III	ALTAIR DOS/Basic III	4,714.00	6,397.00	90
System IV	ALTAIR Extended Engr/Acctg IV	7,938.00	9,985.00	120

(To substitute teletype for COMTER II add $720.00 to kit or $580.00 to assembled price.)

Postage and Handling for systems will be subject to quotation.

Software for 8800 Systems

		Prices for Purchasers of 8800 plus:		
4K Basic	$150.00	4K memory, I/O	$ 60.00	30
8K Basic	$200.00	w/ 8K memory, I/O	$ 75.00	30
EXT Basic	$350.00	w/12K memory, I/O	$ 150.00	30
Package #1	$175.00	w/ 8K memory, I/O	$ 30.00	30
DOS	$500.00	w/12K memory, I/O	$ 150.00	60
DEBUG	$100.00	4K memory, I/O	$ 25.00	60

$15.00 copying charge for update copy or second copy of above software. Copying charge in addition to update charge will be imposed for those updating their software.

PLEASE SPECIFY PAPER TAPE OR AUDIO TAPE WHEN ORDERING SOFTWARE except for DOS which is available only on DISC.

Manuals

CT-256	Operator's	6.50	BASIC Language Documentation 10.00
	Assembly	10.00	
	Theory of Operation	10.00	
8800	Operator's	7.50	Special Offer--All Three Manuals in a Binder--$15.00
	Assembly	9.00	
	Theory of Operation	9.00	
680	Operator's	7.50	Special Offer--All Three Manuals in a Binder--$14.50
	Assembly	7.50	
	Theory of Operation	10.00	

Postage & Terms

Terms: Cash with order, Mastercharge or Bankamericard

Postage & Handling:

1. Add $8.00 for each terminal, computer, line printer and disc
2. Add for Modular Boards
   a. -0- if ordered with computer
   b. $3.00 if ordered separately
3. Postage included in price of manuals
4. Teletype orders will be sent truck freight charges collect.
5. Canada, Hawaii & Alaska postage charges subject to quotation.

Above applies to domestic shipments in U.S.A. only. Overseas shipment, unless otherwise specified are usually made by airfreight via our shipping agents, Emery Airfreight, on airfreight collect basis.