8048/8041 FAMILY
CROSS-ASSEMBLER

USER'S MANUAL

XASM48 Assembler Version 1.60
Manual Revision 1.4 (17-MAR-81)

AVOCET SYSTEMS, INC.
804 SOUTH STATE ST.
DOVER, DELAWARE 19901

(302) 734-0151

Section 7 Format of Listings

1.0 Introduction

XASM48 is a cross-assembler designed to run on 8080- and Z80-based microcomputers, under the CP/M* operating system. It generates machine code for the Intel 8048 family of microprocessors, accepting standard 8048 mnemonics and syntax.

The 8048 family includes five different processor groups, as distinguished by slightly different instruction sets; these are represented by the 8048, 8041A, 8041, 8022, and 8021 chips. Within each group, various models provide a selection of memory and input/output configurations. XASM48 assembles instructions for all five groups, and has provisions for flagging instructions not available in a specified target machine.

XASM48 accepts its input from a CP/M text file, and generates as output an object code file, an assembly listing, and an alphabetized listing of all symbols defined in the assembly. The object file may be in Intel HEX format or may be omitted altogether, at the user's discretion. The listings are normally sent to the system's LIST device, but may be directed to the console or to a disk file.

The assembler features a variety of pseudo-operations. In addition to the usual storage-definition instructions, there are facilities for conditional assembly, for control of listing format, and for including multiple source files in an assembly. New mnemonics may be defined as synonyms for existing operations and pseudo-operations, thus easing the task of converting existing programs.

*Trademark of Digital Research.

This document is intended as a reference manual for the experienced user. As such, it assumes familiarity with the 8048 instruction set, with the operation of CP/M, and with assemblers in general. Users lacking this background information may obtain it from the CP/M manuals and from the references listed below:

MCS-48 USER'S MANUAL	Intel Corp.	No. 9800270
UPI-41 USER'S MANUAL	Intel Corp.	No. 9800504
UPI-41A USER'S MANUAL	Intel Corp.	No. 9800504-02
MCS-48 AND UPI-41 ASSEMBLY LANGUAGE MANUAL	Intel Corp.	No. 9800255

2.0 Running the Cross-Assembler

2.1 XASM48 Command Strings

The 8048 cross-assembler is invoked by typing

XASM48 [d:]filename[.ext] [d:] [switches]

The d:'s are optional drive specifiers (eg. A:, B:, etc.). The file extension (.ext) is also optional; if omitted, it defaults to .ASM. The second d: specifies a disk drive for the output files (object and listing); if it is omitted, these files are placed on the currently-logged drive. The switches are (optional) single characters which control the output of the assembler, as described below.

2.2 XASM48 Switches

The switches following the filename, if supplied, control various options as follows:

L Listing only; ie. no object file.

X No assembly listing.

Y No symbol-table listing

O Object only; equivalent to XY.

C Send listing, if any, to console instead of LST: device.

D Send listing, if any, to disk file with same name as source file and extension "PRN".

N Suppress pagination of assembly listing; ie. no page headings or page ejects.

The switches may be used in combination; for example,

```
XASM48 GRINCH YC
```

will suppress the symbol-table listing and send the assembly listing to the console. Note that errors are always listed, even if listing is turned off. Thus,

```
XASM48 GRINCH LXYC
```

will list errors only, on the console.

2.3 Summary of Defaults

The default output options, in the absence of their respective specifiers, are as follows:

The source file is assumed to have extension "ASM" and to reside on the currently-logged drive.

An object file is generated. It is in Intel HEX format and is placed on the currently-logged drive.

An assembly listing and a symbol-table listing are generated; they are sent to the CP/M LST: device.

3.0 Syntax of Assembler Source Files

3.1 Statements

Source files consist of a sequence of statements of one of the forms:

```
[label:] operator [arguments]    [;comment]

symbol   operator [arguments]    [;comment]

[;comment]
```

If a label or a leading symbol is present, it must begin in column 1. Labels must be followed by a colon. Blank lines are treated as comments. Elements of a statement may be separated by blanks or tabs.

3.2 Symbols

Symbols may be up to 8 characters in length, and may include any of the following characters:

```
A..Z   0..9   $   .
```

The first character of a symbol must be a letter or a period.

3.3 Numeric Constants

Numeric constants consist of a sequence of digits, optionally preceeded or followed by a radix specifier. The first character must be either a leading radix specifier or a decimal digit (0..9). The default radix is ten. For compatiblity with existing assemblers, other bases may be denoted by either a leading or a trailing specifier. The leading radix specifiers are:

```
%  (binary)     @  (octal)     $  (hex)
```

The trailing specifiers are:

```
B (binary)     Q (octal)     H (hex)
```

Thus, for example, the following are equivalent:

%1111111 1111111B @177 177Q $7F 7FH

3.4 Character Constants

Character constants may be used wherever a numeric value is allowed. A character constant consists of one or two characters enclosed in single or double quotes (' or "). The single quote may be used as a character between double quotes, and vice-versa.

Thus, the following are equivalent:

'A' "A" and 41H

'AB' "AB" and 4142H

3.5 Location-Counter Reference

The character "$" may be used as an element in expressions. Its value is the value of the location counter at the beginning of the current statement.

3.6 Expressions

Arithmetic expressions are composed of symbols, numeric constants, character constants, and operators. All operators except +, -, *, and / must be separated from their operands by at least one space. Symbols which are operators are reserved, and may not be redefined as user symbols.

A description of the operators follows:

3.61 Arithmetic Operators

These treat their operands as 16-bit unsigned quantities, and return 16-bit results. No overflow checking is performed.

+ and -	Sum and Difference. Operands and results may be thought of as unsigned or as twos-complement quantities.
unary +	+x is defined as 0+x
unary -	-x is defined as 0-x
* /	Product and Quotient (unsigned)
MOD	Remainder; x MOD y gives the remainder of x/y

3.62 Shift Operators

SHL	Binary left shift. x SHL y yields x shifted left y places (ie. x multiplied by 2**y).
SHR	Binary right shift, logical. x SHR y yields x shifted right y places (ie. x divided by 2**y).

If the right argument is negative, then the direction of the shift is reversed.

3.63 Byte-Extraction Operators

HIGH	Returns the value of the most significant byte of its argument.
LOW	Returns the value of the least significant byte of its argument

These are unary operators, taking an argument on the right. For example: HIGH 1122H is 11H, and LOW 1122H is 22H.

3.64 Boolean Operators

NOT — Unary logical negation. Complements all the bits in its argument.

AND — Logical product; ie. each bit of the result is obtained by ANDing together the corresponding bits in the arguments.

OR — Logical sum.

XOR — Exclusive-OR.

These are all bitwise operators; that is, the same operation is performed on each operand bit position, with no carry from one bit position to the next.

For example:
```
NOT 0 is 0FFFFH
101B AND 010B is 0
101B OR  010B is 111B
101B XOR 010B is 111B
101B XOR 100B is 001B
```

3.65 Relational Operators

These perform unsigned 16-bit comparisons of their operands, returning 1 for TRUE and 0 for FALSE.

For comparison x R y, where R is a relational operator, the results are as follows:

EQ	TRUE iff x and y are equal
NE	TRUE iff x and y are not equal
LE	TRUE iff x is less than or equal to y
LT	TRUE iff x is strictly less than y
GE	TRUE iff x is greater than or equal to y
GT	TRUE iff x is strictly greater than y.

3.66 Evaluation of Expressions

Parentheses may be used to specify the order of evaluation of subexpressions. In the absence of parentheses, this order is determined by operator precedence; higher-precedence operators are evaluated first. In the case of operators with equal precedence, evaluation proceeds from left to right. The operators are listed below in groups according to precedence. Operators in the same horizontal group have the same precedence:

```
unary +, unary -                    (HIGHEST PRECEDENCE)

HIGH  LOW

*  /  MOD  SHR  SHL

+  -

EQ  NE  LT  LE  GT  GE

NOT

AND

OR  XOR                             (LOWEST PRECEDENCE)
```

4.0 Instruction Set Idiosyncracies

While the 8048 family instruction set is not discussed in detail in this manual, some peculiarities of the jump and call instructions are worthy of note:

The JMP and CALL instructions in machine code contain a target address of 11 bits; a 12th bit is supplied via the memory bank selection feature. In the assembler, 12-bit addresses are accepted as operands for these instructions, but the high-order bit is discarded. Range checking is performed to ascertain that the target address is less than or equal to 4095.

No JMP instruction may begin at location 2047 or at 4095. No CALL instruction may begin at 2046, 2047, 4094, or 4095.

The short jump instructions (including DJNZ) contain a single byte operand representing an address within the current 256-byte page. The assembler accepts addresses in the range 0..4095 as operands for these instructions. Bits 8 thru 11 are discarded when assembling the instruction, but are compared with the corresponding bits of the location counter to make sure the target address is accessible. If a short jump begins at the last location of a page (ie. low order address byte is 255) then its operand must lie in the following page; this is enforced by the assembler.

5.0 Pseudo-Operations

5.1 Storage Definition

DB arg[,arg...]

Define Bytes. Each arg may be either an expression or a string. Expressions must evaluate to 8-bit values (high byte either 0 or 255). Strings may be delimited by single or double quotes, as for character constants.

For each expression, a single byte of storage is reserved, initialized to the low byte of the expression's value. For each string, the characters of the string are stored in sequential reserved bytes.

If a compound expression beginning with a character constant is used in a DB, then the expression must be enclosed in parentheses to keep it from being incorrectly parsed as a string. For example,

```
DB ('A'+1)
```

will give the expected result, while

```
DB 'A'+1
```

would be in error.

DW expression[,expression...]

Each expression reserves one word (2 bytes), initialized to the value of the expression. The value is placed in memory with the high-order byte first.

DS expression

Reserves n bytes, where n is the value of the expression. The bytes are not initialized.

5.2 ORG and END

ORG expression

Set program origin. This statement should precede the first code-generating statement in the source file. It sets the program counter initial value to the value of the expression, thus setting the location of code which follows. Additional ORG statements may be used to generate program segments which will load at different locations.

END [expression]

The last statement of the source file must be an END statement. An optional argument is allowed; if this is supplied, its value becomes the start address specified in the last record of the HEX object file.

5.3 Symbol Definition

EQU and SET

These take the form:

```
symbol    EQU  expression
symbol    SET  expression
```

They cause the symbol to be defined and given the value of the argument expression. Symbols defined with EQU serve as symbolic constants, and may not have their values changed. Symbols defined with SET are treated as variables; their values may be changed by additional SET's.

5.4 Conditional Assembly

IF, ELSE, and ENDIF

The construct

```
IF expression
statement
.
.
.
ENDIF
```

behaves as follows: If the value of the expression is non-zero, then the statements between the IF and the ENDIF are assembled. Otherwise, these statements are ignored. Similarly,

```
IF expression
statement
.
.
.
ELSE
statement
.
.
.
ENDIF
```

causes the first sequence of statements to be processed if the expression is TRUE (non-zero), and the second sequence to be processed otherwise.

Conditionals may be nested to a depth of 10. The value of the expression in the IF statement must be known in Pass 1.

5.5 Listing Control

PAGE [expression]

If the argument is omitted, then this causes an immediate page eject (ie. a skip to the top of the next page). If an argument is supplied and has value n, then an eject occurs only if there are less than n lines remaining on the current page.

WIDTH expression

Sets the assumed width of the listing page. The value of the argument may be between 32 and 132, and defaults to 132.

PGLEN expression

Sets the number of lines which will be printed on each page of the listing. Note that the page heading takes up 7 of these lines. The value of the argument may be between 8 and 255, and defaults to 58.

TITLE dtextd

Causes the specified text to become the listing page title, beginning with the next page header printed. The delimiter d may be any printing character.

If no TITLE statement is used, then XASM48 supplies a default title consisting of the text "SOURCE FILE NAME: " followed by the name and extension of the input file.

SBTTL dtextd

Just like TITLE, but sets the listing page subtitle, which is printed on the line after the title line.

LIST and NOLIST

These allow selective listing of portions of a program. NOLIST turns off the assembly listing, and LIST turns it back on. If listing has been turned off with NOLIST, then the next LIST encountered will begin a new page. Command-line switches which disable listing (ie. X and O) will take precedence over LIST. NOLIST does not turn off listing of the symbol table.

5.6 External Source Files

INCLUD d:name.ext

This pseudo-op causes the specified file to be included as if it were present at this point in the source file. INCLUD's may not be nested; that is, the file read by INCLUD may not contain another INCLUD statement. The file must end with an END statement.

5.7 Operator Synonyms

The statement

```
symbol    OPSYN    operator
```

causes the given symbol to be defined as a synonym for the operator or pseudo-op specified as argument. This is particularly useful when assembling source files written for another assembler. For example, if a program uses .BYTE instead of DB, it could be correctly assembled by including the statement

```
.BYTE     OPSYN    DB
```

5.8 Target Machine Validation

XASM48 checks to see that each instruction assembled is in the target machine's instruction set. The default target instruction set is that of the 8048. Others may be specified by one of the following pseudo-ops:

```
MOD41A
MOD41
MOD22
MOD21
```

Only one of these may appear in a given program; it may only appear once, and it must precede all code-generating instructions.

The instruction sets for the five machine groups are listed in appendices C and D.

6.0 Errors

Fatal errors result in the printing of an error message on the console and immediate return to CP/M. Fatal errors may be caused by missing source files, inadequate disk space for output files, or overflow of the symbol table or various internal stacks.

Non-fatal errors are flagged with a character in the first column of the assembly listing. Lines containing errors are always listed, even if listing is turned off. A count is maintained of lines containing errors; if the count is non-zero at the end of the assembly, it is printed on the console in the message:

***** nnn LINES CONTAINED ERRORS *****

Only one error is listed per line; hence, if a line contains multiple errors some may not be caught until successive assembler runs.

The fatal error messages and non-fatal error flags are described in Appendix A.

7.0 Format of Listings

7.1 Page Headings

All listings begin with a heading consisting of seven lines:

(blank line)
(blank line)
(assembler name and version)
(blank line)
(title and page number)
(subtitle)
(blank line)

If no title is supplied in the source program, then the assembler provides a default title consisting of the message

SOURCE FILE NAME:

followed by the name of the input file in the form name.ext. The page number is listed at the right-hand end of this same line, always within the specified page width.

If no subtitle is supplied, the subtitle line is left blank. Both title and subtitle will be truncated, if necessary, to satisfy page width constraints.

7.2 Line Headings

Each code-generating line of the listing begins with the error flag (blank if no error) and the 4-digit hexadecimal value of the location counter as of the start of the line. This is followed by up to four bytes of generated code, also in hexadecimal with two digits per byte. Statements which generate more than 4 bytes will be assembled correctly, but only the first four bytes are listed. Lines which do not generate code but which evaluate an operand (such as EQU or WIDTH) list the operand value in their headers, in place of the location counter.

7.3 Symbol-Table Listing

The symbol-table listing shows all symbols defined in the current assembly, with their hexadecimal values. Only user-defined symbols are listed. Symbols are in vertical columns, sorted alphabetically according to the ASCII collating sequence. The number of columns is adjusted automatically to fit in the specified page width. All pages of the symbol-table listing are automatically subtitled

```
---- SYMBOL TABLE ----
```

Because the sorting scheme "alphabetizes" symbol names even if they end in numeric characters, the listing order may not be what you expect. For example, a typical sequence of symbols might appear as follows:

```
SYM19
SYM2
SYM20
SYM21
SYM3
SYM4
```

APPENDIX A. Error Messages and Flags

Non-Fatal Error Flags

C	Conditional Err	Unmatched IF, ELSE, or ENDIF; or conditionals nested too deep.
I	INCLUD Err	File not found, or nested INCLUD's.
J	Jump Err	A jump or call instruction begins too near the end of a 2K page. Jumps and DJNZ may not begin at locations 2047 or 4095. CALL may not begin at locations 2046, 2047, 4094, or 4095.
L	Label Err	Label too long. (>8 chars)
M	Multiple Defn	Symbol already defined.
O	Operator Err	Undefined or illegal operator.
P	Phase Err	Symbol had different value on Pass 2 than on Pass 1.
R	Range Err	Argument out of bounds, branch out of range, or illegal register/port number.
S	Syntax Err	Ill-formed argument or expression.
U	Undefined	Undefined symbol(s) in operand field.
W	Warning	Instruction not available on specified target machine.

Fatal Error Messages

SOURCE FILE NOT FOUND	The specified source file doesn't exist.
UNABLE TO CREATE OUTPUT FILE	The directory is full on the disk specifed for output.
OUTPUT FILE WRITE ERROR	The output disk is full.
EVALUATION STACK FULL	An arithmetic expression was encountered which had too many levels of parentheses or of precedence nesting.
SYMBOL TABLE FULL	Not enough memory remains to create a table entry for a symbol being defined.

Non-Fatal Error Messages

NO ROOM FOR SYMBOL-TABLE SORT	Not enough memory is available to sort the symbol table. Symbol-table listing is therefore omitted.
END STATEMENT MISSING	End-of-File was reached in the source file, or in an include file, without encountering an END statement. The assembler inserts an END statement, flagged by a string of asterisks in the comment field.

APPENDIX B. Object File Format

Object files are in the Intel HEX format, which represents binary data bytes as two-digit ASCII hexadecimal numbers. An object file consists of a sequence of data records, followed by a single end record.

The record formats are:

Data Record:

Byte 1	Colon (:)
2..3	Number of binary data bytes in this record.
4..5	Load address for this record, high byte.
6..7	Load address, " " " low byte.
8..9	Unused, should be "00".
10..x	Data bytes, two characters each.
x+1..x+2	Checksum (2 characters).
x+3..x+4	CR/LF

End Record:

Like data record, but number of data bytes is zero and the load address field contains the program starting address.

The checksum is the two's complement of the 8-bit sum, without carry, of all the data bytes, the two bytes of load address, and the byte count.

APPENDIX C. Summary Of Instruction Sets

Mnemonic			8048	8041A	8041	8022	8021
ADD	A,#data		X	X	X	X	X
ADD	A,Rr	(r=0..7)	X	X	X	X	X
ADD	A,@Rr	(r=0..1)	X	X	X	X	X
ADDC	A,#data		X	X	X	X	X
ADDC	A,Rr	(r=0..7)	X	X	X	X	X
ADDC	A,@Rr	(r=0..1)	X	X	X	X	X
ANL	A,#data		X	X	X	X	X
ANL	A,Rr	(r=0..7)	X	X	X	X	X
ANL	A,@Rr	(r=0..1)	X	X	X	X	X
ANL	BUS,#data		X				
ANL	Pp,#data	(p=1..2)	X	X	X		
ANLD	Pp,A	(p=4..7)	X	X	X	X	X
CALL	addr		X	X	X	X	X
CLR	A		X	X	X	X	X
CLR	C		X	X	X	X	X
CLR	F0		X	X	X		
CLR	F1		X	X	X		
CPL	A		X	X	X	X	X
CPL	C		X	X	X	X	X
CPL	F0		X	X	X		
CPL	F1		X	X	X		
DA	A		X	X	X	X	X

Mnemonic			8048	8041A	8041	8022	8021
DEC	A		X	X	X	X	X
DEC	Rr	(r=0..7)	X	X	X		
DIS	I		X	X	X	X	
DIS	TCNTI		X	X	X	X	
DJNZ	Rr,addr	(r=0..7)	X	X	X	X	X
EN	DMA			X			
EN	FLAGS			X			
EN	I		X	X	X	X	
EN	TCNTI		X	X	X	X	
ENT0	CLK		X				
IN	A,DBB			X	X		
IN	A,P0					X	X
IN	A,Pp	(p=1..2)	X	X	X	X	X
INC	A		X	X	X	X	X
INC	Rr	(r=0..7)	X	X	X	X	X
INC	@Rr	(r=0..1)	X	X	X	X	X
INS	A,BUS		X				
JBb	addr	(b=0..7)	X	X	X		
JC	addr		X	X	X	X	X
JF0	addr		X	X	X		
JF1	addr		X	X	X		
JMP	addr		X	X	X	X	X
JMPP	@A		X	X	X	X	X
JNC	addr		X	X	X	X	X

Mnemonic			8048	8041A	8041	8022	8021
JNI	addr		X				
JNIBF	addr			X	X		
JNTO	addr		X	X	X	X	
JNT1	addr		X	X	X	X	X
JNZ	addr		X	X	X	X	X
JOBF	addr			X	X		
JTF	addr		X	X	X	X	X
JTO	addr		X	X	X	X	
JT1	addr		X	X	X	X	X
JZ	addr		X	X	X	X	X
MOV	A,#data		X	X	X	X	X
MOV	A,PSW		X	X	X		
MOV	A,Rr	(r=0..7)	X	X	X	X	X
MOV	A,@Rr	(r=0..1)	X	X	X	X	X
MOV	A,T		X	X	X	X	X
MOV	PSW,A		X	X	X		
MOV	Rr,A	(r=0..7)	X	X	X	X	X
MOV	Rr,#data	(r=0..7)	X	X	X	X	X
MOV	@Rr,A	(r=0..1)	X	X	X	X	X
MOV	@Rr,#data	(r=0..1)	X	X	X	X	X
MOV	STS,A			X			
MOV	T,A		X	X	X	X	X
MOVD	A,Pp	(p=4..7)	X	X	X	X	X
MOVD	Pp,A	(p=4..7)	X	X	X	X	X

Mnemonic			8048	8041A	8041	8022	8021
MOVP	A,@A		X	X	X	X	X
MOVP3	A,@A		X	X	X		
MOVX	A,@Rr	(r=0..1)	X				
MOVX	@Rr,A	(r=0..1)	X				
NOP			X	X	X	X	X
ORL	A,#data		X	X	X	X	X
ORL	A,Rr	(r=0..7)	X	X	X	X	X
ORL	A,@Rr	(r=0..1)	X	X	X	X	X
ORL	BUS,#data		X				
ORL	Pp,#data	(p=1..2)	X	X	X		
ORLD	Pp,A	(p=4..7)	X	X	X	X	X
OUT	DBB,A			X	X		
OUTL	BUS,A		X				
OUTL	P0,A					X	X
OUTL	Pp,A	(p=1..2)	X	X	X	X	X
RAD						X	
RET			X	X	X	X	X
RETI						X	
RETR			X	X	X		
RL	A		X	X	X	X	X
RLC	A		X	X	X	X	X
RR	A		X	X	X	X	X
RRC	A		X	X	X	X	X
SEL	AN0					X	

Mnemonic			8048	8041A	8041	8022	8021
SEL	AN1					X	
SEL	MB0		X				
SEL	MB1		X				
SEL	RB0		X	X	X		
SEL	RB1		X	X	X		
STOP	TCNT		X	X	X	X	X
STRT	CNT		X	X	X	X	X
STRT	T		X	X	X	X	X
SWAP	A		X	X	X	X	X
XCH	A,Rr	(r=0..7)	X	X	X	X	X
XCH	A,@Rr	(r=0..1)	X	X	X	X	X
XCHD	A,@Rr	(r=0..1)	X	X	X	X	X
XRL	A,#data		X	X	X	X	X
XRL	A,Rr	(r=0..7)	X	X	X	X	X
XRL	A,@Rr	(r=0..1)	X	X	X	X	X

APPENDIX D. Instruction Mnemonics and Opcodes

This is an excerpt from an actual XASM48 assembly listing, showing all of the instruction mnemonics, the available addressing modes for each, and the associated hex opcodes. Instructions not available on the 8048 proper have been flagged as "W" errors by the assembler.

```
0100                    ORG     100H
0005           GRINCH   EQU     5

0100 68                 ADD     A,R0
0101 69                 ADD     A,R1
0102 6A                 ADD     A,R2
0103 6B                 ADD     A,R3
0104 6C                 ADD     A,R4
0105 6D                 ADD     A,R5
0106 6E                 ADD     A,R6
0107 6F                 ADD     A,R7

0108 60                 ADD     A,@R0
0109 61                 ADD     A,@R1

010A 0305               ADD     A,#GRINCH

010C 78                 ADDC    A,R0
010D 79                 ADDC    A,R1
010E 7A                 ADDC    A,R2
010F 7B                 ADDC    A,R3
0110 7C                 ADDC    A,R4
0111 7D                 ADDC    A,R5
0112 7E                 ADDC    A,R6
0113 7F                 ADDC    A,R7

0114 70                 ADDC    A,@R0
0115 71                 ADDC    A,@R1

0116 1305               ADDC    A,#GRINCH

0118 58                 ANL     A,R0
0119 59                 ANL     A,R1
011A 5A                 ANL     A,R2
011B 5B                 ANL     A,R3
011C 5C                 ANL     A,R4
011D 5D                 ANL     A,R5
011E 5E                 ANL     A,R6
011F 5F                 ANL     A,R7
```

```
0120 50              ANL     A,@R0
0121 51              ANL     A,@R1

0122 5305            ANL     A,#GRINCH
0124 9817            ANL     BUS,#17H        ;8048 only
0126 9905            ANL     P1,#GRINCH      ;Not in 8021,8022
0128 9A05            ANL     P2,#GRINCH      ;Not in 8021,8022

012A 9C      ADDR1:  ANLD    P4,A
012B 9D              ANLD    P5,A
012C 9E              ANLD    P6,A
012D 9F              ANLD    P7,A

012E 342A            CALL    ADDR1

0130 27              CLR     A
0131 97              CLR     C
0132 85              CLR     F0              ;Not in 8021,8022
0133 A5              CLR     F1              ;Not in 8021,8022

0134 37              CPL     A
0135 A7              CPL     C
0136 95              CPL     F0              ;Not in 8021,8022
0137 B5              CPL     F1              ;Not in 8021,8022

0138 57              DA      A

0139 07              DEC     A

013A C8              DEC     R0              ;Not in 8021,8022
013B C9              DEC     R1              ;Not in 8021,8022
013C CA              DEC     R2              ;Not in 8021,8022
013D CB              DEC     R3              ;Not in 8021,8022
013E CC              DEC     R4              ;Not in 8021,8022
013F CD              DEC     R5              ;Not in 8021,8022
0140 CE              DEC     R6              ;Not in 8021,8022
0141 CF              DEC     R7              ;Not in 8021,8022

0142 15              DIS     I               ;Not in 8021
0143 35              DIS     TCNTI           ;Not in 8021

0144 E844    ADDR2:  DJNZ    R0,ADDR2
0146 E944            DJNZ    R1,ADDR2
0148 EA44            DJNZ    R2,ADDR2
014A EB44            DJNZ    R3,ADDR2
014C EC44            DJNZ    R4,ADDR2
014E ED44            DJNZ    R5,ADDR2
0150 EE44            DJNZ    R6,ADDR2
0152 EF44            DJNZ    R7,ADDR2
```

```
W0154 E5              EN      DMA             ;8041A only
W0155 F5              EN      FLAGS           ;8041A only
 0156 05              EN      I               ;Not in 8021
 0157 25              EN      TCNTI

 0158 75              ENTO    CLK             ;8048 only

W0159 08              IN      A,P0            ;8021,8022 only
 015A 09              IN      A,P1
 015B 0A              IN      A,P2
W015C 22              IN      A,DBB           ;8041/41A only

 015D 17              INC     A
 015E 18              INC     R0
 015F 19              INC     R1
 0160 1A              INC     R2
 0161 1B              INC     R3
 0162 1C              INC     R4
 0163 1D              INC     R5
 0164 1E              INC     R6
 0165 1F              INC     R7
 0166 10              INC     @R0
 0167 11              INC     @R1

 0168 08              INS     A,BUS           ;8048 only

 0169 1244            JB0     ADDR2           ;Not in 8022,8021
 016B 3244            JB1     ADDR2           ;Not in 8022,8021
 016D 5244            JB2     ADDR2           ;Not in 8022,8021
 016F 7244            JB3     ADDR2           ;Not in 8022,8021
 0171 9244            JB4     ADDR2           ;Not in 8022,8021
 0173 B244            JB5     ADDR2           ;Not in 8022,8021
 0175 D244            JB6     ADDR2           ;Not in 8022,8021
 0177 F244            JB7     ADDR2           ;Not in 8022,8021

 0179 F644            JC      ADDR2
 017B B644            JF0     ADDR2           ;Not in 8022,8021
 017D 7644            JF1     ADDR2           ;Not in 8022,8021
 017F
 017F 242A            JMP     ADDR1
 0181 B3              JMPP    @A

 0182 E682    ADDR3:  JNC     ADDR3
 0184 8682            JNI     ADDR3           ;8048 only
 0186 2682            JNT0    ADDR3           ;Not in 8021
 0188 4682            JNT1    ADDR3
 018A 9682            JNZ     ADDR3
 018C 1682            JTF     ADDR3
 018E 3682            JT0     ADDR3           ;Not in 8021
 0190 5682            JT1     ADDR3
 0192 C682            JZ      ADDR3
```

```
W0194 D682          JNIBF   ADDR3           ;8041/41A only
W0196 8682          JOBF    ADDR3           ;8041/41A only

 0198 237F          MOV     A,#7FH
 019A C7            MOV     A,PSW           ;Not in 8022,8021
 019B F8            MOV     A,R0
 019C F9            MOV     A,R1
 019D FA            MOV     A,R2
 019E FB            MOV     A,R3
 019F FC            MOV     A,R4
 01A0 FD            MOV     A,R5
 01A1 FE            MOV     A,R6
 01A2 FF            MOV     A,R7
 01A3 F0            MOV     A,@R0
 01A4 F1            MOV     A,@R1
 01A5 42            MOV     A,T
 01A6 D7            MOV     PSW,A           ;Not in 8022,8021
W01A7 90            MOV     STS,A           ;8041A only

 01A8 A8            MOV     R0,A
 01A9 A9            MOV     R1,A
 01AA AA            MOV     R2,A
 01AB AB            MOV     R3,A
 01AC AC            MOV     R4,A
 01AD AD            MOV     R5,A
 01AE AE            MOV     R6,A
 01AF AF            MOV     R7,A

 01B0 B805          MOV     R0,#GRINCH
 01B2 B905          MOV     R1,#GRINCH
 01B4 BA05          MOV     R2,#GRINCH
 01B6 BB05          MOV     R3,#GRINCH
 01B8 BC05          MOV     R4,#GRINCH
 01BA BD05          MOV     R5,#GRINCH
 01BC BE05          MOV     R6,#GRINCH
 01BE BF05          MOV     R7,#GRINCH

 01C0 A0            MOV     @R0,A
 01C1 A1            MOV     @R1,A
 01C2 B005          MOV     @R0,#GRINCH
 01C4 B105          MOV     @R1,#GRINCH
 01C6 62            MOV     T,A

 01C7 0C            MOVD    A,P4
 01C8 0D            MOVD    A,P5
 01C9 0E            MOVD    A,P6
 01CA 0F            MOVD    A,P7
```

```
01CB 3C             MOVD    P4,A
01CC 3D             MOVD    P5,A
01CD 3E             MOVD    P6,A
01CE 3F             MOVD    P7,A
01CF A3             MOVP    A,@A
01D0 E3             MOVP3   A,@A            ;Not in 8022, 8021

01D180              MOVX    A,@R0           ;8048 only
01D2 81             MOVX    A,@R1           ;8048 only
01D3 90             MOVX    @R0,A           ;8048 only
01D4 91             MOVX    @R1,A           ;8048 only

01D5 00             NOP

01D6 48             ORL     A,R0
01D7 49             ORL     A,R1
01D8 4A             ORL     A,R2
01D9 4B             ORL     A,R3
01DA 4C             ORL     A,R4
01DB 4D             ORL     A,R5
01DC 4E             ORL     A,R6
01DD 4F             ORL     A,R7

01DE 40             ORL     A,@R0
01DF 41             ORL     A,@R1
01E0 4305           ORL     A,#GRINCH
01E2 8817           ORL     BUS,#17H        ;8048 only
01E4 8907           ORL     P1,#7           ;Not in 8022,8021
01E6 8A07           ORL     P2,#7           ;Not in 8022,8021

01E8 8C             ORLD    P4,A
01E9 8D             ORLD    P5,A
01EA 8E             ORLD    P6,A
01EB 8F             ORLD    P7,A

W01EC 02            OUT     DBB,A           ;8041/41A only
 01ED 02            OUTL    BUS,A           ;8048 only
W01EE 90            OUTL    P0,A            ;8021, 8022 only
 01EF 39            OUTL    P1,A
 01F0 3A            OUTL    P2,A

W01F1 80            RAD                     ;8022 only
 01F2 83            RET
W01F3 93            RETI                    ;8022 only
 01F4 93            RETR                    ;Not in 8022,8021
```

```
 01F5 E7            RL      A
 01F6 F7            RLC     A
 01F7 77            RR      A
 01F8 67            RRC     A
W01F9 85            SEL     AN0             ;8022 only
W01FA 95            SEL     AN1             ;8022 only
 01FB E5            SEL     MB0             ;8048 only
 01FC F5            SEL     MB1             ;8048 only
 01FD C5            SEL     RB0             ;Not in 8022,8021
 01FE D5            SEL     RB1             ;Not in 8022,8021

 01FF 65            STOP    TCNT
 0200 45            STRT    CNT
 0201 55            STRT    T
 0202 47            SWAP    A

 0203 28            XCH     A,R0
 0204 29            XCH     A,R1
 0205 2A            XCH     A,R2
 0206 2B            XCH     A,R3
 0207 2C            XCH     A,R4
 0208 2D            XCH     A,R5
 0209 2E            XCH     A,R6
 020A 2F            XCH     A,R7

 020B 20            XCH     A,@R0
 020C 21            XCH     A,@R1

 020D 30            XCHD    A,@R0
 020E 31            XCHD    A,@R1

 020F D8            XRL     A,R0
 0210 D9            XRL     A,R1
 0211 DA            XRL     A,R2
 0212 DB            XRL     A,R3
 0213 DC            XRL     A,R4
 0214 DD            XRL     A,R5
 0215 DE            XRL     A,R6
 0216 DF            XRL     A,R7

 0217 D0            XRL     A,@R0
 0218 D1            XRL     A,@R1
 0219 D305          XRL     A,#GRINCH
```

Contents Of The Diskette

1. XASM48.COM ..... 8048/8041 Family Cross-Assembler

2. TEST48.ASM ..... Demonstration source code file, showing 8048/8041 family instruction set.

Current Versions

Assembler: V1.64 (Released 04-Oct-83)
Manual: Rev. 1.4

Enhancements In V1.64

1. SYNONYMS FOR PSEUDO-OPS

   The pseudo-op EJECT has been added as a synonym for PAGE. INCLUDE has been added as a synonym for INCLUD.

2. FORWARD REFERENCES IN PSEUDO-OPS

   An 'F' error will occur if a forward reference is present in the operand of any of the following pseudo-ops:

   DS
   EQU
   IF
   LOC
   ORG
   SET

   In general, if an 'F' error is present in your assembly you can expect obscure problems; usually, some symbols will have values which don't correspond to the location counter. Thus, it's best to get rid of all illegal forward references before trusting the generated code.

3. DEFAULT LISTING DESTINATION

   The assembly listing now defaults to a .PRN file, rather than to the printer. It may be diverted to the printer by specifying the 'P' switch in the assembler command line. The 'D' switch is no longer recognized.

Bugs Fixed In V1.64

1. The END statement is now recognized even if an unterminated conditional (ie. IF without ENDIF) is present.

2. When the assembly listing is sent directly to the printer, the page headings no longer contain LF characters with the high-order bit set.

3. The SHR operator has been fixed.

4. Symbols beginning with a period (".") are now accepted.

Known Bugs Remaining In V1.64

1. The presence of a non-printing ASCII character in a source line may cause an 'S' error, or may cause the line to be interpreted incorrectly.

2. Quasi-Bug: Assembler input (ie. source file) must be upper-case, except within quoted strings and character constants.

Updates To XASM48 Manual

1. FORMAT OF OBJECT FILE (Appendix B)

The END record in object files produced by XASM48 is now:

```
Byte 1          :
2..3    00      (no. of data bytes)
4..7    0000    (or execution address, if
                 specified in END statement)
8..9    01      (record type)
10      FF      (checksum)
```

This change reflects Intel's revised definition of the HEX format.

2. RE-DEFINITION OF HARDWARE REGISTERS (V1.63 and Higher)

By popular request, XASM48 now supports the assignment of symbolic names to the hardware registers. Thus one can write, for example,

```
FOO     EQU     R0
BAR     EQU     R5

        MOV     A,@FOO
        MOV     BAR,#27H
```

3. INTERRUPTING ASSEMBLY

Assembler operation may be terminated at any time by striking a control-C.

Contents Of The Diskette

1. XASM48.CMD (CP/M-86) ..... 8048/8041 Family Cross-Assembler
   or
   XASM48.COM (MSDOS)

2. TEST48.ASM ..... Demonstration source code file, showing 8048/8041 family instruction set.

Current Versions

CP/M-86: 1.64C, Released 20-Jul-83
MSDOS (PCDOS): 1.64M, Released 20-Jul-83
Manual: 1.4

Known Bugs in Current Version

1. The presence of a non-printing ASCII character in a source line may cause an 'S' error, or may cause the line to be interpreted incorrectly.

2. Forward references are not allowed in ORG, DS, IF, EQU, but are not flagged as errors if present.

3. Quasi-Bug: Assembler input (ie. source file) must be upper-case.

Updates To XASM48 Manual (Revision 1.4)

1. ASSEMBLY LISTING DESTINATION (Section 2)

   The assembly listing now goes to a .PRN file (not to the printer) unless otherwise specified. The 'D' switch is no longer recognized. The 'P' switch has been added. It causes the assembly listing to be sent to the printer.

2. FORMAT OF OBJECT FILE (Appendix B)

   The END record in object files produced by XASM51 is now:

Byte 1		:
2..3	00	(no. of data bytes)
4..7	0000	(or execution address, if specified in END statement)
8..9	01	(record type)
10	FF	(checksum)

   This change reflects Intel's revised definition of the HEX format.

3. RE-DEFINITION OF HARDWARE REGISTERS (General)

By popular request, XASM48 now supports the assignment of symbolic names to the hardware registers. Thus one can write, for example,

```
FOO     EQU     R0
BAR     EQU     R5

        MOV     A,@FOO
        MOV     BAR,#27H
```

4. INTERRUPTING ASSEMBLY

Assembler operation may be terminated at any time by striking a control-C.

8048 Cross-Assembler Notes

Contents Of The Diskette

1. XASM48.COM ..... 8048/8041 Family Cross-Assembler

2. TEST48.ASM ..... Demonstration source code file, showing 8048/8041 family instruction set.

Current Versions

CP/M-80: 1.64 (Released 04-Oct-83)

Enhancements In V1.64

1. SYNONYMS FOR PSEUDO-OPS

   The pseudo-op EJECT has been added as a synonym for PAGE. INCLUDE has been added as a synonym for INCLUD.

2. FORWARD REFERENCES IN PSEUDO-OPS

   An 'F' error will occur if a forward reference is present in the operand of any of the following pseudo-ops:

   DS
   EQU
   IF
   LOC
   ORG
   SET

   In general, if an 'F' error is present in your assembly you can expect obscure problems; usually, some symbols will have values which don't correspond to the location counter. Thus, it's best to get rid of all illegal forward references before trusting the generated code.

3. DEFAULT LISTING DESTINATION

   The assembly listing now defaults to a .PRN file, rather than to the printer. It may be diverted to the printer by specifying the 'P' switch in the assembler command line. The 'D' switch is no longer recognized.

Bugs Fixed In V1.64

1. The END statement is now recognized even if an unterminated conditional

(ie. IF without ENDIF) is present.

2. When the assembly listing is sent directly to the printer, the page headings no longer contain LF characters with the high-order bit set.

3. The SHR operator has been fixed.

4. Symbols beginning with a period (".") are now accepted.

Known Bugs Remaining In V1.64

1. The presence of a non-printing ASCII character in a source line may cause an 'S' error, or may cause the line to be interpreted incorrectly.

2. Quasi-Bug: Assembler input (ie. source file) must be upper-case, except within quoted strings and character constants.

Updates To XASM48 Manual

1. FORMAT OF OBJECT FILE (Appendix B)

The END record in object files produced by XASM48 is now:

```
Byte 1          :
2..3    00      (no. of data bytes)
4..7    0000    (or execution address, if
                 specified in END statement)
8..9    01      (record type)
10      FF      (checksum)
```

This change reflects Intel's revised definition of the HEX format.

2. RE-DEFINITION OF HARDWARE REGISTERS (V1.63 and Higher)

By popular request, XASM48 now supports the assignment of symbolic names to the hardware registers. Thus one can write, for example,

```
FOO     EQU     R0
BAR     EQU     R5

        MOV     A,@FOO
        MOV     BAR,#27H
```

3. INTERRUPTING ASSEMBLY

Assembler operation may be terminated at any time by striking a control-C.