

AD-A148 031

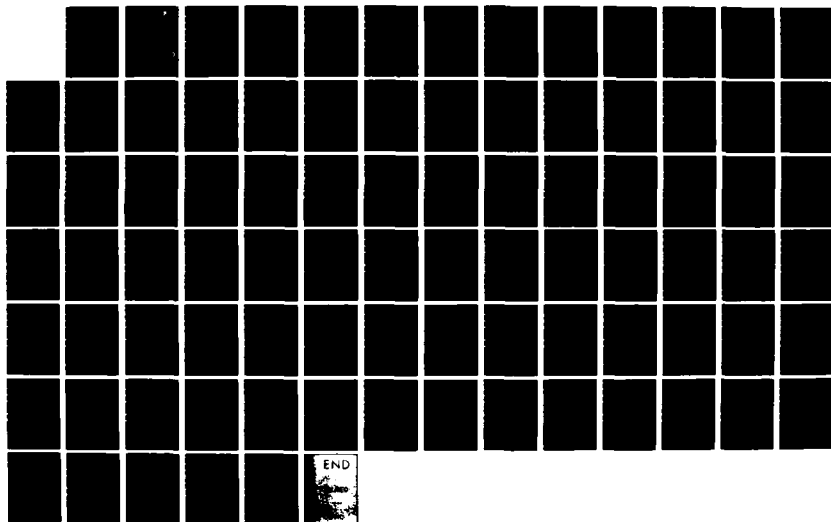
MC68CRX CROSS-ASSEMBLER USERS MANUAL(U) ROME AIR
DEVELOPMENT CENTER GRIFFISS AFB NY K D ROMANO SEP 84
RADC-TM-84-19

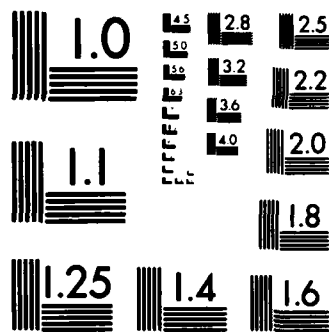
1/1

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

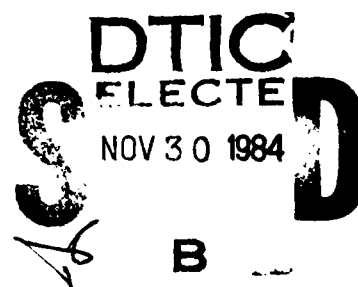
RADC-TM-84-19
In-House Report
September 1984



MC68CRX CROSS-ASSEMBLER USERS MANUAL

Ken D. Romano

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED



ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, NY 13441

84 11 19 110

AD-A148 031

DTIC FILE COPY

This report has been reviewed by the RADC Public Affairs Office (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

RADC-TM-84-19 has been reviewed and is approved for publication.

APPROVED:



THADEUS J. DOMURAT
Chief, Signal Intelligence Branch
Intelligence & Reconnaissance Division

APPROVED:



THADEUS J. DOMURAT
Acting Chief, Intelligence & Reconnaissance Division

FOR THE COMMANDER:



DONALD A. BRANTINGHAM
Plans Office

If your address has changed or if you wish to be removed from the RADC mailing list, or if the addressee is no longer employed by your organization, please notify RADC (IRAP) Griffiss AFB NY 13441. This will assist us in maintaining a current mailing list.

Do not return copies of this report unless contractual obligations or notices on a specific document requires that it be returned.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED		1b. RESTRICTIVE MARKINGS N/A		
2a. SECURITY CLASSIFICATION AUTHORITY N/A		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A				
4. PERFORMING ORGANIZATION REPORT NUMBER(S) N/A		5. MONITORING ORGANIZATION REPORT NUMBER(S) RADC-TM-84-19		
6a. NAME OF PERFORMING ORGANIZATION Rome Air Development Center		6b. OFFICE SYMBOL (If applicable) IRAP		7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441		7b. ADDRESS (City, State and ZIP Code)		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Rome Air Development Center		8b. OFFICE SYMBOL (If applicable) IRAP		9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER N/A
8c. ADDRESS (City, State and ZIP Code) Griffiss AFB NY 13441		10. SOURCE OF FUNDING NOS.		
		PROGRAM ELEMENT NO. 62702F	PROJECT NO. 4594	TASK NO. 15 WORK UNIT NO. LB
11. TITLE (Include Security Classification) MC68CRX CROSS-ASSEMBLER USERS MANUAL				
12. PERSONAL AUTHOR(S) Ken D. Romano				
13a. TYPE OF REPORT In-House		13b. TIME COVERED FROM May 84 TO Aug 84		14. DATE OF REPORT (Yr., Mo., Day) September 1984
15. PAGE COUNT 96				
16. SUPPLEMENTARY NOTATION N/A				
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)	
FIELD	GROUP	SUB. GR.	Microprocessor Cross-Assembler Users Guide	
09	02		Program Listing	
19. ABSTRACT (Continue on reverse if necessary and identify by block number)				
This in-house report is a technical user's manual containing all the information needed to utilize a Fortran Cross-Assembler (MC68CRX) for the Motorola MC68000 microprocessor. The Cross-Assembler was developed in-house at RADC (IRAP). A program listing (Fortran 77) is also included, along with information concerning hardware connections from the MC68000 to a DEC mainframe computer. 26 p i				
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS ☐			21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED	
22a. NAME OF RESPONSIBLE INDIVIDUAL Andrew J. Noga			22b. TELEPHONE NUMBER (Include Area Code) 315-330-3206	22c. OFFICE SYMBOL RADC (IRAP)

DD FORM 1473, 83 APR

EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

INTRODUCTION

147
This manual describes the MC68CRX cross-assembler and a Fortran transfer program, which were developed to facilitate programming of the Motorola MC68000 microprocessor, and development of MC68000 based systems. Both programs are written in Fortran 77, which allows the user to utilize the features of a mainframe computer, such as the DEC 11/70 or DEC 11/45. The cross-assembler translates MC68000 assembly language code into machine language. The transfer program downloads the machine code to the MC68000. ←

This manual is designed as a reference to the specifics of the cross-assembler and transfer program, and assumes that the user is familiar with MC68000 assembly language and the host system. For detailed information on MC68000 machine code, the user is encouraged to consult the sources listed in APPENDIX B.

A complete listing of the cross-assembler and transfer program is included in APPENDIX A.

RE: Legibility, Appendix A
Best copy available per Ms. Joyce Watkins,
RADC/TSR

Accession For	
NTIS GRA&I	☒
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution	
Availability Codes	
Avail and/or	
Dist	
A-1	

MC68CRX USERS MANUAL TABLE OF CONTENTS

Section	page
-----	-----
System Diagram.....	1
Running the cross assembler.....	2
Running programs on the MC68000.....	3
Assembler code file format.....	5
Addressing mode chart.....	6
Addressing mode details.....	7

ASSEMBLER MNEMONICS

DIRECTIVES

DC.....	9
DS.....	10
END.....	11
EQU.....	10

OPERATIONS

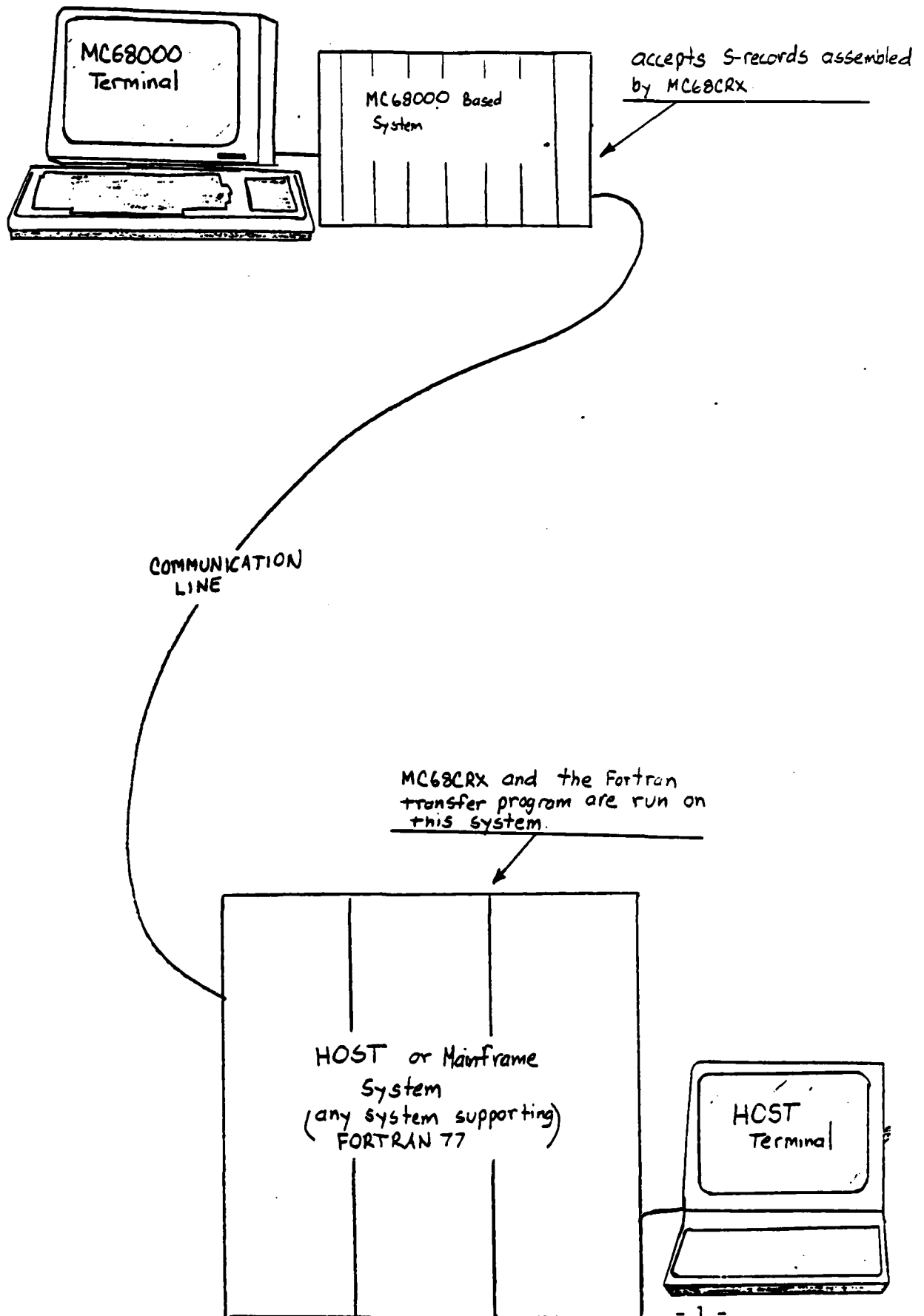
ADD.....	12
----------	----

ADDA.....	23
ADDQ.....	21
AND.....	14
ANDI.....	23
ASL.....	17
ASR.....	17
Bcc.....	19
BTST.....	28
CMP.....	16
CMPI.....	23
DIVU.....	26
DIVS.....	26
EOR.....	17
EORI.....	23
JMP.....	23
JSR.....	24
LSL.....	18
LSR.....	18
MOVE.....	15

MOVEA.....	22
MULS.....	25
MULU.....	25
NEG.....	20
NEX.....	20
NOP.....	27
ORR.....	15
ORRI.....	23
ROL.....	18
ROR.....	18
RTS.....	24
STOP.....	27
SUB.....	13
SUBA.....	22
SUBI.....	23
SWAP.....	27

MC68CRX FORTRAN PROGRAM SPECIFICS.....	29
Subroutines.....	31
Adding Mnemonics to the cross assembler.....	34
Program and subroutine listings.....	APPENDIX A
Bibliography.....	APPENDIX B
MC68000 Design Module to Host System	
Hardware Connections.....	APPENDIX C
MVME400 RS-232C" Module to Host System	
Hardware Connections.....	APPENDIX D
Compiling and Taskbuilding the Cross-assembler	APPENDIX E

SYSTEM DIAGRAM



I. RUNNING THE CROSS ASSEMBLER:

Begin by typing (on the host terminal):

RUN MC68CRX

program prompt: INPUT MEMORY LOCATION (HEX) AT WHICH TO
BEGIN PROGRAM STORAGE IN MC68000 RAM (<8000,>06FF)

user input: four character hex string, which will be the
program's starting address.

program prompt: INPUT MEMORY LOCATION (HEX) AT WHICH TO
BEGIN DATA STORAGE IN MC68000 RAM

user input: four character hex string, which will be the
starting address for data storage.

program prompt: INPUT NAME OF ASSEMBLER CODE FILE

user input: name of file containing assembler code.

program prompt: INPUT NAME OF OUTPUT (S RECORD) FILE TO BE
CREATED: XXXXX.M68

user input: name of file which will contain assembled S
records, to be sent to the MC68000. 5 letters.M68.

program prompt: INPUT NAME OF LIST FILE TO BE CREATED:
XXXXX.LST

user input: name of file which will contain assembler code
and its assembled hex code, useful for debugging.

II. TRANSFER OF S RECORDS TO THE MC68000

With MAXbug firmware monitor:

After the MC68CRX program has been run, the S record file (FNAME.M68) must be sent to the MC68000 using the MC68000 TRANSFER PROGRAM in file TRANSFER.FTN. This program should be taskbuilt with logical unit number 1 being assigned to the MC68000 terminal, and number 5 being the terminal being used on the host system.

user input on the host terminal: RUN TRANSFER

user input on the MC68000 terminal: RE;=FNAME or RE;X=FNAME
with the X option, S records will be displayed on the MC68000 terminal as they arrive.

prompt on host terminal: ENTER VERIFY;=FNAME OR *DONE ON
MC68000 TERMINAL

user input on MC68000 terminal: VERIFY;=FNAME or *DONE

*DONE completes the transfer process, VERIFY;=FNAME checks the S records again and displays any discrepancies, will cause a prompt of : ENTER VERIFY;=FNAME OR *DONE ON MC68000 TERMINAL on the host terminal again.

With VMEbug firmware monitor:

Use LO;=FNAME (load) instead of RE;=FNAME.

III. TRANSFER OF S-RECORDS USING A SINGLE TERMINAL

If the transfer program is taskbuilt with logical unit numbers 1 and 5 being assigned to the MC68000, transfer of S records can be done without using a terminal on the host system.

user input on the MC68000 terminal:

```
*HEL (account number)
*(password)
*RUN TRANSFER
RE;=FNAME or RE;X=FNAME
VERIFY;=FNAME or *DONE
(if previous command was verify,
now type *DONE)
*BYE
```

With VMEbug firmware monitor:

Use LO;=FNAME instead of RE;=FNAME

IV. RUNNING PROGRAMS ON THE MC68000

With MAXbug firmware monitor.

After the S records have been sent to the MC68000, the program can be run with the following commands:

user input on MC68000 terminal: PC xxxx , where xxxx is the program's starting address in hex. (Same as the program storage location on page 2)

G TILL yyyy , where yyyy is the address of the last assembler instruction, this can be obtained from the list file.

(for more details concerning running programs on the MC68000, consult: Motorola MC68000 DESIGN MODULE USER'S GUIDE [MEX68KDM(D3)] (MAXbug firmware) or VMEbug DEBUGGING PACKAGES USERS MANUAL [MVMEBUG/D2].)

With VMEbug firmware monitor.

Use .PC instead of PC, GT instead of G TILL.

THE ASSEMBLER CODE FILE

Programs in MC68000 assembly language must be contained in a file of 100 lines or less on the host system. Long programs can be broken into parts and put into memory in the proper order, remembering that jumps to labels in different sections will have to be modified.

The assembler code file is made up of 4 distinct fields. Each field starts in a column which is unique to the field. The four fields are LABEL, OPERAT, ADRES1, ADRES2 and have the following functions:

LABEL : columns 1-5, can be used to label lines, constants, or provide jump-to points in the program.

OPERAT : columns 20-25, contains the assembler operation or directive.

ADRES1 : columns 40-48, contains the source address or immediate data.

ADRES2 : columns 50-58, contains the destination address.

EXAMPLE FILE:

AB30	EORIW	#FFFFFF	D0
	SWAP	D0	
	EORIW	#FFFFFF	D0
	SWAP	D0	
	ADD1L	#1	D0
POSTIV	ADD1L	D0	D1
	SUB1W	#1	D2
	CMPIW	#0	D2
	BGT	(VARIAT	
	DIVU	#1	D1
	MULU	#100	D1
	DIVU	\$7F02	D1
	MOVEW	D1	\$7F08
	MOVEAW	\$7EFC	A0
	MOVEW	D1	*A0
	ADDQW	#2	\$7EFC
	CMPIW	#\$74D0	\$7EFC
	BEG	(STOP	
	MOVEW	#0	\$7F00
	RTS		
STOP	STOP		
	END		

ADDRESSING MODES:

The MC68CRX cross-assembler supports nine of the twelve addressing modes available on the MC68000. The user specifies which mode is being used by the first one or two characters in the source (ADRES1) and destination (ADRES2) fields.

ADDRESSING MODE	assembler code file source/dest. field	Motorola RTL notation
Data register direct	Dn	Dn
Address register direct	An	An
Address register indirect	@An	@An
Postincrement register indirect	+An	An+
Predecrement register indirect	-An	An-
Register indirect with integer displacement	%IIIIAn or %-IIIIAn	An(d)
Register indirect with hex displacement	%\$HHHAn	An(d)
Program counter relative with integer displacement	PCIIIII or PC-IIIII	PC(d)
Program counter relative with hex displacement	PC\$HHHH	PC(d)
Immediate integer	#I or #II...#IIIII	#xxxx
Immediate hex	#\$HHHH	#xxxx
Absolute short	\$HHHH or (label	xxx.W

NOTES: n = register number
IIIII = 5 place integer
HHHH = 4 place hex

ADDRESSING MODE DETAILS:

Data Register Direct - Dn

The operand is stored in data register n.

Address Register Direct -@An

The operand is stored in address register n.

Address Register Indirect - An

The operand is stored in the memory location which is stored in address register n.

Postincrement Register Indirect - +An

The operand is stored in the memory location which is stored in address register n. After the instruction is executed, the location stored in An is incremented by 1,2, or 4, depending on the operation data size.

Predecrement Register Indirect - -An

Same as Postincrement register indirect, except that the location stored in An is decremented by 1,2, or 4, before the operation is executed.

Register Indirect With Displacement - %(displacement)An

The operand is stored in the location stored in An plus the displacement.

Program Counter Relative With Displacement - PC(displacement)

The location of the operand is the sum of the program counter and the displacement.

Immediate - #(data)

The operand is '(data)', either hex or integer.

Absolute Short - \$(location) or (label

The operand is stored in memory location '(location)', or the location associated with 'label'.

NON-ASSEMBLY LANGUAGE COMMANDS: DIRECTIVES

Since the user specifies the memory locations where data and program storage are to begin, the need for an origin (ORG) command is eliminated. However, some useful data storage directives are supported by the assembler. These include DC, DS and EQU which may be before, after or buried within the assembler source file and will have no effect on the program storage.

DIRECTIVE: DC - define constant

field	LABEL	OPERAT	ADRES1

(as would appear in assembly code file)	[label]	DCL	[-]constant
	[label]	DCW	([-]constant)('character')
	[label]	DCB	(constant)('character')

NOTE: [] - enclosed is optional

() - one of the enclosed types must be
used

Stores the value in ADRES1 field in the next available data storage location. Automatically increments data count to assure word or long word data begins on an even memory location. Note that signed data is not allowed for DCB (byte storage). Data counter incremented by 2 for DCW (word storage), 4 for DCL (long word storage), 1 for DCB.

CALL STATEMENT IN MAIN PROGRAM:

CALL DC(LABEL, OPERAT, ADRES1, DCOUNT, NCK)

DIRECTIVE: EQU - equate

field	LABEL	OPERAT	ADRES1

	[label]	EQU	\$HHHH

Equates label with memory location shown in ADRES1 field.
Adds nothing to memory.

CALL STATEMENT IN MAIN PROGRAM:
CALL EQU(LABEL, ADRES1, NCK)

DIRECTIVE: DS - define storage space

field	LABEL	OPERAT	ADRES1

	[label]	DSL	
	[label]	DSW	
	[label]	DSB	

Keeps space open for data storage. Four bytes for DSL,
two bytes for DSW, one byte for DSB.

CALL STATEMENT IN MAIN PROGRAM:
CALL DS(LABEL, OPERAT, DCOUNT, NCK)

DIRECTIVE: END - Ends assembler file.

field	LABEL	OPERAT

	[label]	END

ASSEMBLER OPERATIONS

OPERATION: ADD - add binary. Adds source (ADRES1) to destination (ADRES2) and stores in destination.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	ADD1(B)(W)(L)	(source)	D(n)
	[label]	ADD2(B)(W)(L)	D(n)	(destination)

ADD1B - Data register is destination - byte data

W - " " " " - word data

L - " " " " - long word data

ADD2B - Data register is source - byte data

W - " " " " - word data

L - " " " " - long word data

ADDRESSING MODES SUPPORTED:

ADD1

Source - all supported
Destination - data register direct

ADD2

Source - data register direct
Destination - all except:
 address register indirect
 immediate
 PC relative with disp.

CALL STATEMENT IN MAIN PROGRAM:

CALL ANDADD(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT, NWORDS,
HEXM, BIN1, BIN2)

OPERATION: SUB - Subtract source (ADRES1) from destination (ADRES2) and stores result in destination.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	SUB1(B)(W)(L)	(source)	D(n)
	[label]	SUB2(B)(W)(L)	D(n)	(destination)

SUB1B - data register destination - byte data

W - " " " - word data

L - " " " - long word data

SUB2B - data register source - byte data

W - " " " - word data

L - " " " - long word data

ADDRESSING MODES SUPPORTED:

SUB1

source - all
destination - data register direct

SUB2

source - data register direct
destination - all except:
data register direct
address " "
PC relative with disp.
immediate

CALL STATEMENT IN MAIN PROGRAM:

same as ADD

OPERATION: AND - Logical AND bit by bit between source (ADRES1) and destination (ADRES2), result stored in destination.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	AND1(B)(W)(L)	(source)	D(n)
	[label]	AND2(B)(W)(L)	D(n)	(destination)

AND1B - Data register destination - byte data

W - " " " - word data

L - " " " - long word data

AND2B - Data register source - byte data

W - " " " - word data

L - " " " - long word data

ADDRESSING MODES SUPPORTED:

AND1

Source - all except:
 address register direct
 Destination - data register direct

AND2

Source - data register direct
 Destination - all except:
 address register direct
 immediate
 PC relative with disp.

CALL STATEMENT IN MAIN PROGRAM:

CALL ANDADD(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT, NWORDS,
 HEXM, BIN1, BIN2)

OPERATION: ORR - Inclusive OR bit by bit between source (ADRES1) and destination (ADRES2), stored in destination.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	ORR1(B)(W)(L)	(source)	D(n)
	[label]	ORR2(B)(W)(L)	D(n)	(destination)

ORR1B - Data register destination - byte data

W - " " " - word data

L - " " " - long word data

ORR2B - Data register source - byte data

W - " " " - word data

L - " " " - long word data

For addressing mode details see AND.

CALL STATEMENT , same as AND.

OPERATION: MOVE - Move data from source (ADRES1) to destination (ADRES2).

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	MOVE(B)(W)(L) (specifies data size)	(source)	(destination)

ADDRESSING MODES SUPPORTED:

source - all except: PC relative with displacement

destination - all except: address register direct
immediate
PC relative with disp.

CALL STATEMENT IN MAIN PROGRAM:

CALL MOVE(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT, NWORDS,
HEXM, BIN1, BIN2, BIN3)

OPERATION: CMP - Subtract the source (ADRES1) operand from the
destination (ADRES2) operand and set the condition codes
according to result. Neither operand is changed.

field	LABEL	OPERAND	ADRES1	ADRES2

	[label]	CMP(B)(W)(L)	(source)	D(n)

CMPB - byte data

W - word data

L - long word data

ADDRESSING MODES SUPPORTED:

Source - All

Destination - data register direct

CALL STATEMENT IN MAIN PROGRAM:

CALL CMP(OPERAT)

CALL ANDADD(...)

OPERATION: EOR - Exclusive OR logical. Performs an exclusive or, bit by bit between the source (ADRES1) and destination (ADRES2), and stores the result in the destination operand.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	EOR(B)(W)(L) (specifies data size)	D(n)	(destination)

DESTINATION ADDRESSING MODES ALLOWED:

all except:
 address register direct
 PC relative with displacement
 immediate

OPERATION: ASL, ASR - Arithmetic shift left, right. Arithmetically shifts contents of register or memory location by a specified number of bits.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	ASLD(B)(W)(L)	D(n)	D(m)
	[label]	ASRD(B)(W)(L) (shifts contents of destination)	D(n) (contains number of bit shifts)	D(m) (data register to be shifted)
	[label]	ASLM	(address)	
	[label]	ASRM (shifts data in memory one bit)	(address) (address of data to be shifted)	

ADDRESSING SUPPORTED:

ASLD, ASRD : as shown above

ASLM, ASRM : address register indirect
post increment register indirect
predecrement " "
register indirect with displacement
absolute short

CALL STATEMENT IN MAIN PROGRAM:

CALL AS(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT, NWORDS,
HEXM, BIN1, BIN2)

OPERATION: LSL,LSR - Same as ASL, ASR, except LSR places a
zero in the most significant bit of the operand, where ASR
keeps it intact.

(See ASL, ASR for details on addressing and format of
operation.)

OPERATION: ROL, ROR - Rotates data to the left or right by a
specified number of bits.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	ROLD(B)(W)(L)	D(n)	D(m)
	[label]	RORD(B)(W)(L)	D(n)	D(m)

(see ASL, ASR)

[label] ROLM (address)

[label] RORM (address)

ADDRESSING MODES SUPPORTED:

same as ASL, ASD

CALL STATEMENT IN MAIN PROGRAM:

CALL AS(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT, HEXM,
BIN1, BIN2)

OPERATION: Bcc - Conditional branch. cc is condition code.
If condition is met, control is transferred to location
specified by ADRES1.

field	LABEL	OPERAT	ADRES1

	[label]	B(cc)	(location)

ADDRESSING MODES SUPPORTED:

program counter relative with displacement
absolute short

(if PC relative with disp. is used the displacement
should be decreased by two if the desired displacement
is counted from the location of the Bcc instruction.)

CONDITION CODES:

! code !	condition
!-----!	-----
! HI !	high
!-----!	-----
! LS !	low or same

CC	carry clear
NE	not equal
CS	carry set
EQ	equal
VC	overflow clear
VS	overflow set
PL	plus
MI	minus
GE	greater or equal
LT	less than
GT	greater than
LE	less or equal
RA	branch always
SR	branch to subroutine

NOTE ON USING Bcc WITH CMP: If Bcc is used after a CMP type instruction, the relation tested is:

DESTINATION condition SOURCE

Where destination and source are from the CMP instruction line.

OPERATION:NEG,NEX - Negate, negate with extend. NEg subtracts the contents of source (ADRES1) operand from zero using two's complement arithmetic. NEX subtracts the source operand and the value of the extend flag from zero. Results are stored in source (ADRES1).

field	LABEL	OPERAT	ADRES1

	[label]	NEG(B)(W)(L)	(source)
	[label]	NEX(B)(L)(W)	(source)
		(specifies data size)	

SOURCE ADDRESSING MODES SUPPORTED:

all except:
 address register direct
 PC relative with displacement
 immediate

OPERATION: ADDQ - Add quick. Adds immediate data of 1-8 to the destination operand and stores result in the destination. Immediate data is in ADRES1 field.

field	LABEL	OPERAT	ADRES1	ADRES2

	[label]	ADDQ(B)(W)(L)	\$(data)	(destination)
		(specifies data size)		

ADDRESSING MODES SUPPORTED FOR DESTINATION:

all except:
 PC relative with displacement
 immediate

CALL STATEMENT IN MAIN PROGRAM:

CALL QADD(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT,
NWORDS, HEXM, BIN1, BIN2)

ADDRESS REGISTER DIRECT ADDRESSING OPERATIONS: Perform same operations as MOVE, ADD, and SUB, with the destination (ADRES2) being an address register addressed directly.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	MOVEA(W)(L)	(source)	A(n)
	[label]	ADDA(W)(L)	(source)	A(n)
	[label]	SUBA(W)(L)	(source)	A(n)
		(note byte data is not allowed)		

SOURCE ADDRESSING MODES SUPPORTED:

All

WITH SIZE SPEC 'L':

all except immediate

CALL STATEMENT IN MAIN PROGRAM:

CALL OPTA(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT,
NWORDS, HEXM, BIN1, BIN2)

IMMEDIATE OPERATIONS: ANDI, ORRI, EORI, SUBI, CMPI, use immediate data as the source operand.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	ANDI(B)(W)	#[\$(data)	(destination)
	"	ORRI(B)(W)	"	"
	"	EORI(B)(W)	"	"
	"	SUBI(B)(W)	"	"
	"	CMPI(B)(W) (specifies size spec.)	"	"

Perform same functions as AND, ORR, EOR, SUB, CMP.
Note that long word data cannot be used.

ADDRESSING MODES SUPPORTED:

source - immediate
destination - all except:
 address register direct
 PC relative with displacement
 immediate

CALL STATEMENT IN MAIN PROGRAM:

CALL IMME(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT,
NWORDS, HEXM, BIN1, BIN2, BIN3)

OPERATION: JMP - Unconditional jump to specified memory address.

field	LABEL	OPERAT	ADRES1

[label] JMP (address)

Previous value of program counter is lost.

ADDRESSING MODES SUPPORTED:

all except : data register direct
address register direct
postincrement register indirect
predecrement " "
immediate data

OPERATION: JSR - Jump to subroutine and save old value of
program counter on system stack.

field	LABEL	OPERAT	ADRES1

[label] JSR (address)

ADDRESSING MODES SUPPORTED:

same as JMP

CALL STATEMENT IN MAIN PROGRAM (both JMP and JSR):

CALL JUMP(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT,
NWORDS, HEXM, BIN1, BIN2)

OPERATION: RTS - Return from subroutine to location stored

on stack.

field	LABEL	OPERAT

	[label]	RTS

Will not affect status flags.

OPERATION: MUL - Signed or unsigned multiply. Multiplies two 16-bit operands and yields a 32-bit result which is stored in the data register destination. MULU (unsigned) uses unsigned binary arithmetic, and MULS uses two's complement signed binary arithmetic.

field	LABEL	OPERAT	ADRES1	ADRES2

	[label]	MULS	(source)	D(n)
	[label]	MULU	(source)	D(n)

ADDRESSING MODES SUPPORTED:

source - all except:
 address register direct

destination - data register direct

CALL STATEMENT IN MAIN PROGRAM:

CALL MULDIV(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT,
NWORDS, HEXM, BIN1, BIN2)

OPERATION: DIV - Signed or unsigned divide. Divides destination (ADRES2) by source (ADRES1), result stored in destination with the quotient in the least significant word and the remainder in the most significant word. DIVU (unsigned) uses binary arithmetic and DIVS uses signed two's complement arithmetic.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	DIVS	(source)	D(n)
	[label]	DIVU	(source)	D(n)

ADDRESSING MODES SUPPORTED:

source - all except:
 address register direct

destination - data register direct

CALL STATEMENT IN MAIN PROGRAM:
 same as MUL

OPERATION: NOP - No operation. Increments program counter.

field	LABEL	OPERAT

	[label]	NOP

CALL STATEMENT IN MAIN PROGRAM:

CALL NOP(LABEL, OPERAT, PCOUNT, NWORDS, HEXM, BIN1)

OPERATION: STOP - Loads next memory word into status register and stops processor.

field	LABEL	OPERAT

	[label]	STOP

CALL STATEMENT IN MAIN PROGRAM:

same as NOP

OPERATION: SWAP - Swaps data register halves.

field	LABEL	OPERAT	ADRES1

	[label]	SWAP	D(n)

ONLY ADDRESSING IS AS SHOWN

CALL STATEMENT IN MAIN PROGRAM:

CALL SWAP(LABEL, OPERAT, ADRES1, PCOUNT, NWORDS, HEXM)

OPERATION: BTST, Test a specified bit in the destination operand and set the zero status flag according to result.

field	LABEL	OPERAT	ADRES1	ADRES2
	[label]	BTST	\$(bit no.)	(desination)

DESTINATION ADDRESSING MODES SUPPORTED:

all except :
address register direct
immediate data

MC68CRX FORTRAN PROGRAM SPECIFICS:

Sections of the MC68CRX Main Program
(See APPENDIX A, program listing)

INITIALIZATION SECTION - Sets up the program and data start locations. START is the input variable which is converted to PCOUNT, the program counter.

FILE NAMING AND OPENING SECTION - User inputs the names of all files to be manipulated in the cross assembly process. The list file is opened and a header is printed in that file.

READ ASSEMBLY LINE SECTION - Opens assembler code file and .M68 file. Reads one line of assembler code into variables LABEL, OPERAT, ADRES1, ADRES2.

CALL SECTION - Matches OPERAT to a string and calls proper subroutine.

PASS CHECK SECTION - Checks variable NSTOP to see if an END has been encountered in the assembly code file. If so, increment NPASS by 1. If NPASS = 3, assembly is complete.

WRITE S-RECORD SECTION - Converts the binary instruction string BIN1 to hex and inserts it into the S-record array. If NWORDS is greater than one, BIN2 and BIN3 (if used) are also converted to hex and inserted into the S-record array. The hex memory location HEXM array is inserted into the S-record array.

COUNT AND CHECKSUM SECTION - Sets up the count and checksum sections of the S-record and inserts them into the S-record array.

SUBROUTINES:

File	Subroutine	Function/Mnemonic
MC68CRX.FTN		Main Program
OPTSUB2.FTN	ANDADD	ADD,AND,ORR,CMP,SUB,EOR
	MOVE	MOVE
	CMP	CMP
	EOR	EOR
	AS	ASL,ASR,LSL,LSR,ROL,ROR
	Bcc	Bcc
	QMOVE	MOVEQ
	QADD	ADDQ
	IMME	ADDI,ANDI,ORRI,EORI
SUBDIR.FTN	EQU	EQU
	DS	DS
	END	END
	OPTA	ADDA,SUBA,MOVEA
	NOP	NOP,STOP
	JUMP	JSR,JMP
	MULDIV	MULU,MULS,DIVU,DIVS
	NEG	NEG,NEX
UTLSUB.FTN	KSTRIN	Separates four character string into a 4 element array, rightmost character becomes the first element.
	TCOMP	Performs a two's complement on the 16 or 32 array sent to the sub-

routine. (Complements
and adds 1 with carry)

OCOMP

Complements all bits of
array sent.

CKSUM

Computes the checksum
for each S record and
adds it to the S record
array. Also generates
list file.

LABTAB

First pass: sets up table
(two parallel arrays)
of labels and their lo-
cations.
Second pass: returns the
location of a label name.

LABAD

Used in conjunction with
LABTAB during second pass
of assembler.

TEST

BTST

DCSUB.FTN

DC

DC

SUBS1.FTN

BINDIG

Converts a 16 element
character array of binary
(1's and 0's) into a 4
byte integer. Element
1 of array is 'ones'
place.

DIGHEX

Integer (4 byte) to four
element character array.

SUBS2.FTN

HEXNUM

Hex array to 4 byte
integer conversion,

NUMBIN

4 byte integer to 16 or
32 element character
array of binary 1's and
0's.

SUBS3.FTN

ADRLOC

Returns a 3 element
character array when
sent a single character
which is a numeral from

0-7. The 3 element array
is a binary representation
of the numeral sent.

TADR

Returns the necessary
addressing information
when sent the contents
of an address field.

ADDING MNEMONICS TO THE MC68CRX CROSS ASSEMBLER:

The MC68000 supports over sixty instructions. The most commonly used mnemonics are assembled by the MC68CRX cross assembler. However, if a programming situation occurs which requires an operation not currently in the library of the MC68CRX program, a subroutine (the operation subroutine) can be added to assemble the instruction.

The operation subroutine must contain the following:

NPASS, the variable which counts the number of passes the assembler has made, must be declared as COMMON/BLOCK1/NPASS at the start of the subroutine.

Each operation subroutine must have the variables PCOUNT, NWORDS, and HEXM passed to it. NWORDS must be set to an integer which is the number of memory words the instruction will write (1-3). PCOUNT, the program counter, must be incremented by two for each memory word, preferably just prior to the return statement. After the COMMON statement and type declarations, the following lines must be included.

```
IF(NPASS.NE.1)GO TO 100
  CALL LABTAB(LABEL,PCOUNT,NA)
  IF(ADRES1(1:1).EQ.'(')ADRES1='$0000'
  IF(ADRES2(1:1).EQ.'(')ADRES2='$0000'
  GO TO 150
100 CALL LABAD(ADRES1,ADRES2)
150 CALL DIGHEX(PCOUNT,HEXM)
```

[These three lines
only needed if
ADRES1,ADRES2 were sent
to the subroutine]

(PCOUNT should be incremented AFTER these lines)

The subroutine should generate and return up to three words of binary code stored in 16 element, single character arrays. The code for the effective address fields of many instructions can be easily obtained by using the subroutine TADR, which is called by:

CALL TADR(ADRES, MODE, REG, NUM, TYPE, FLG)

ADRES is ADRES1 or ADRES2 (this is the only variable sent TO the subroutine)

MODE and REG are 3 element, single character arrays containing the binary code for the effective address field.

NUM is a 4-byte integer variable which contains an integer equivalent to the value of the 16-bit binary extension word used by some addressing modes. If NUM is to be used, the integer variable FLG will be set to 1 by TADR. NUM can be converted to a binary word array by using:

CALL NUMBIN(NUM, BIN32, BIN2, NZ)

(BIN32 is a 32 element array not used. NZ is a single character variable not used)

TYPE is a single character variable not used.

Each operation/mnemonic subroutine must be called in the CALL SECTION of the main program (MC68CRX).

APPENDIX A
PROGRAM LISTINGS

```

C      MC68CRX
C      MC68000 CROSS ASSEMBLER      KEN ROMANO,IRAP,JUNE 1984
C      COMMENTS REFER TO SECTIONS EXPLAINED IN MC68CRX USERS MANUAL
C

```

```

COMMON/BLOCK1/NPASS
COMMON/BLOCK2/LABEL,OPERAT,ADRES1,ADRES2
CHARACTER*6 LABEL,OPERAT
CHARACTER*9 ADRES1,ADRES2,FOUT,FLST
CHARACTER*1 BIN1(16),BIN2(16),BINL(32)
CHARACTER*1 BIN3(16),BIN4(16),LOC(3),HEXM(4)
CHARACTER*1 HEX2(4),HEX3(4),HEX4(4),SREC(30)
CHARACTER*15 FNAME
CHARACTER*4 START,NSTOP,DSTART
INTEGER*4 NUMBER,DCOUNT,PCOUNT,PCONT2,DCONT2
INTEGER NCK,NWORDS

```

```

C
C      INITIALIZATION SECTION
C

```

```

      WRITE(5,100)
100      FORMAT(1X,'INPUT MEMORY LOCATION (HEX) AT WHICH TO BEGIN PROGRAM
      & STORAGE IN MC68000 RAM (<8000,>06FF)')
      READ(5,111)START
111      FORMAT(A4)
      WRITE(5,150)
150      FORMAT(1X,'INPUT MEMORY LOCATION (HEX) AT WHICH TO BEGIN DATA
      & STORAGE IN MC68000 RAM')
      READ(5,222)DSTART
222      FORMAT(A4)
      DO 152 J=1,4
          HEXM(-1*J+5)=START(J:J)
          HEX4(-1*J+5)=DSTART(J:J)
152      CONTINUE
      CALL HEXNUM(HEXM,PCOUNT)
      PCONT2=PCOUNT
      CALL HEXNUM(HEX4,DCOUNT)
      DCONT2=DCOUNT

```

```

C
C      FILE NAMING AND OPENING SECTION
C

```

```

      WRITE(5,200)
200      FORMAT(1X,'INPUT NAME OF ASSEMBLER CODE FILE')
      READ(5,225)FNAME
225      FORMAT(A15)
      WRITE(5,300)
300      FORMAT(1X,'INPUT NAME OF OUTPUT (S RECORD) FILE TO BE
      & CREATED: XXXXX.M68')
      READ(5,230)FOUT
230      FORMAT(A9)
      WRITE(5,310)
310      FORMAT(1X,'INPUT NAME OF LIST FILE TO BE CREATED: XXXXX.LST')
      READ(5,240)FLST
240      FORMAT(A9)
      OPEN(UNIT=11,FILE=FLST,STATUS='NEW')
      WRITE(11,312)
312      FORMAT(1X,'LABEL',T9,'OPERAT',T17,'ADRES1',T28,'ADRES2',
      ST40,'LOCATION',T50,'HEX DATA')
      WRITE(11,314)
314      FORMAT(2X)

```

```

C
C      INITIALIZE ASSEMBLER
C

```

```

      NPASS=1
      A1

```



```

CALL LABIAB('XSTART',PCOUNT,1)
C
C
C
350 OPEN(UNIT=3,FILE=FNAME,READONLY,STATUS='OLD')
    OPEN(UNIT=4,FILE=FOUT,STATUS='NEW')
    REWIND 3
    REWIND 4
    NSTOP='GO'
    PCOUNT=PCONT2
    DCOUNT=DCONT2
    WRITE(4,360)
360  FORMAT(1X,'S0')
400  READ(3,440)LABEL,OPERAT,ADRES1,ADRES2
440  FORMAT(T1,A6,T20,A6,T40,A9,T50,A9)
    NCK=0
    DO 441 KK=1,30
        SREC(KK)=' '
441  CONTINUE
C
C
C
CALL SECTION
IDENTIFY MNEMONIC AND CALL PROPER SUBROUTINES

    IF(OPERAT(1:4).NE.'SUBA')GO TO 371
370  CALL OPTA(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,
    SHEXM,BIN1,BIN2)
    GO TO 498
371  IF(OPERAT(1:4).NE.'ADDA')GO TO 372
    GO TO 370
372  IF(OPERAT(1:5).NE.'MOVEA')GO TO 373
    GO TO 370
373  IF(OPERAT(1:5).NE.'MOVEQ')GO TO 374
    CALL QMOVE(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,
    SHEXM,BIN1)
    GO TO 498
374  IF(OPERAT(1:4).NE.'ADDQ')GO TO 375
    CALL QADD(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,
    SHEXM,BIN1,BIN2)
    GO TO 498
375  IF(OPERAT(1:3).NE.'NOP')GO TO 377
376  CALL NOP(LABEL,OPERAT,PCOUNT,NWORDS,HEXM,BIN1)
    GO TO 498
377  IF(OPERAT(1:4).NE.'STOP')GO TO 378
    GO TO 376
378  IF(OPERAT(1:3).EQ.'JSR')GO TO 380
    IF(OPERAT(1:3).EQ.'RTS')GO TO 380
    IF(OPERAT(1:3).NE.'JMP')GO TO 381
380  CALL JUMP(LABEL,OPERAT,ADRES1,PCOUNT,NWORDS,
    SHEXM,BIN1,BIN2)
    GO TO 498
381  IF(OPERAT(1:3).EQ.'MUL')GO TO 382
    IF(OPERAT(1:3).NE.'DIV')GO TO 383
382  CALL MULDIV(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,
    SNWORDS,HEXM,BIN1,BIN2)
    GO TO 498
383  IF(OPERAT(1:3).EQ.'NEG')GO TO 384
    IF(OPERAT(1:3).NE.'NEX')GO TO 385
384  CALL NEG(LABEL,OPERAT,ADRES1,PCOUNT,NWORDS,HEXM,
    SBIN1,BIN2)
    GO TO 498
385  IF(OPERAT(1:4).NE.'SWAP')GO TO 386
    CALL SWAP(LABEL,OPERAT,ADRES1,PCOUNT,NWORDS,HEXM,
    SBIN1)
    GO TO 498
386  IF(OPERAT(1:4).NE.'EORI')GO TO 387
    GO TO 409

```

```

387 IF (OPERAT(1:4).NE.'BTST') GO TO 399
    CALL TEST(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,
$HEXM,BIN1,BIN2,BIN3)
    GO TO 498
399 IF (OPERAT(1:4).NE.'ADDI') GO TO 401
409 CALL IMME(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,
$HEXM,BIN1,BIN2,BIN3)
    GO TO 498
401 IF (OPERAT(1:4).NE.'ANDI') GO TO 402
    GO TO 409
402 IF (OPERAT(1:4).NE.'ORRI') GO TO 403
    GO TO 409
403 IF (OPERAT(1:3).NE.'EOR') GO TO 404
    CALL EOR(OPERAT)
    CALL ANDADD(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,
$HEXM,BIN1,BIN2)
    GO TO 498
404 IF (OPERAT(1:4).NE.'SUBI') GO TO 405
    GO TO 409
405 IF (OPERAT(1:4).NE.'CMPI') GO TO 410
    GO TO 409
410 IF (OPERAT(1:2).NE.'DC') GO TO 411
    CALL DC(LABEL,OPERAT,ADRES1,DCOUNT,NCK)
    GO TO 498
411 IF (OPERAT.NE.'END') GO TO 412
    CALL END(PCONT2,NSTOP)
    GO TO 498
412 IF (OPERAT(1:2).NE.'DS') GO TO 413
    CALL DS(LABEL,OPERAT,DCOUNT,NCK)
    GO TO 498
413 IF (OPERAT.NE.'EQU') GO TO 414
    CALL EQU(LABEL,ADRES1,NCK)
    GO TO 498
414 IF (OPERAT(1:3).NE.'AND') GO TO 415
    CALL ANDADD(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,HEXM,BIN1,
$BIN2)
    GO TO 498
415 IF (OPERAT(1:3).NE.'ADD') GO TO 416
    CALL ANDADD(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,HEXM,BIN1,
$BIN2)
    GO TO 498
416 IF (OPERAT(1:3).NE.'ORR') GO TO 417
    CALL ANDADD(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,HEXM,BIN1,
$BIN2)
    GO TO 498
417 IF (OPERAT(1:3).NE.'CMP') GO TO 418
    CALL CMP(OPERAT)
    CALL ANDADD(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,HEXM,BIN1,
$BIN2)
    GO TO 498
418 IF (OPERAT(1:4).NE.'MOVE') GO TO 419
    CALL MOVE(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,HEXM,BIN1,
$BIN2,BIN3)
    GO TO 498
419 IF (OPERAT(1:2).NE.'AS') GO TO 420
    CALL AS(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,HEXM,
$BIN1,BIN2)
    GO TO 498
420 IF (OPERAT(1:2).NE.'LS') GO TO 442
    CALL AS(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,HEXM,
$BIN1,BIN2)
    GO TO 498
442 IF (OPERAT(1:3).NE.'SUB') GO TO 452
    CALL ANDADD(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,
$HEXM,BIN1,BIN2)
    GO TO 498

```

```

452 IF (OPERAT(1).NE.'B')GO TO 476
CALL BCC(LABEL,OPERAT,ADRES1,PCOUNT,NWORDS,HEXM,BIN1,BIN2)
GO TO 498
496 WRITE(5,500)OPERAT
500 FORMAT(1X,'INVALID COMMAND : ',A6,' -EXECUTION TERMINATED')
STOP

C
C PASS CHECK SECTION
C
498 IF(NSTOP.NE.'STOP')GO TO 510
NPASS=NPASS+1
IF(NPASS.EQ.3)STOP
CLOSE (UNIT=3,STATUS='KEEP')
CLOSE (UNIT=4,STATUS='DELETE')
GO TO 350

C
C WRITE/NO WRITE CHECK
C
510 IF(NCK.EQ.1)GO TO 400

C
C WRITE S-RECORD SECTION
C
SREC(1)='S'
SREC(2)='1'
CALL BINDIG(BIN1,NUMBER)
CALL DIGHEX(NUMBER,HEX2)
DO 1000 J=1,4
SREC(J+8)=HEX2(-1*J+5)
1000 CONTINUE
IF(NWORDS.EQ.1)GO TO 1010
CALL BINDIG(BIN2,NUMBER)
CALL DIGHEX(NUMBER,HEX3)
DO 1001 J=1,4
SREC(J+12)=HEX3(-1*J+5)
1001 CONTINUE
IF(NWORDS.EQ.2)GO TO 1010
CALL BINDIG(BIN3,NUMBER)
CALL DIGHEX(NUMBER,HEX4)
DO 1002 J=1,4
SREC(J+16)=HEX4(-1*J+5)
1002 CONTINUE
1010 DO 1003 J=1,4
SREC(J+4)=HEXM(-1*J+5)
1003 CONTINUE

C
C COUNT AND CHEKSUM SECTION
C
IF(NWORDS.NE.3)GO TO 1020
SREC(3)='0'
SREC(4)='9'
CALL CKSUM(SREC,3)
GO TO 1050
1020 IF(NWORDS.NE.2)GO TO 1030
SREC(3)='0'
SREC(4)='7'
CALL CKSUM(SREC,2)
GO TO 1050
1030 SREC(3)='0'
SREC(4)='5'
CALL CKSUM(SREC,1)

C
C WRITE S RECORDS TO ,M68 FILE
1050 WRITE(4,5900)(SREC(J),J=1,30)
5900 FORMAT(1X,30A1)
GO TO 400

```

C
C

CLOSE FILES

9999

CONTINUE

CLOSE(UNIT=3,STATUS='KEEP')

CLOSE(UNIT=4,STATUS='KEEP')

CLOSE(UNIT=11,STATUS='KEEP')

STOP

END

```

C
C
C      BINARY TO DECIMAL CONVERSION
SUBROUTINE BINDIG(BINARY,NUMBER)
CHARACTER*1 BINARY(16)
INTEGER*4 NUMBER,K
INTEGER*4 MULT
NUMBER=0
DO 50 K=1,16
    IF(BINARY(K).NE.'1')GO TO 50
    MULT=2**(K-1)
    NUMBER=NUMBER+MULT
50  CONTINUE
RETURN
END

```

```

C
C
C      DECIMAL TO HEX CONVERSION SUBROUTINE
SUBROUTINE DIGHEX(NUMBER,HEX)
INTEGER*4 NUMBER
CHARACTER*1 HEX(4)
ANUM=FLOATJ(NUMBER)
DO 100 K=4,1,-1
    DIVID=ANUM/(16**(K-1))
    NREM=IINT(DIVID)
    IF(NREM.GT.15)GO TO 999
    IF(NREM.NE.15)GO TO 5
        HEX(K)='F'
        GO TO 99
5    IF(NREM.NE.14)GO TO 10
        HEX(K)='E'
        GO TO 99
10   IF(NREM.NE.13)GO TO 15
        HEX(K)='D'
        GO TO 99
15   IF(NREM.NE.12)GO TO 20
        HEX(K)='C'
        GO TO 99
20   IF(NREM.NE.11)GO TO 25
        HEX(K)='B'
        GO TO 99
25   IF(NREM.NE.10)GO TO 30
        HEX(K)='A'
        GO TO 99
30   IF(NREM.NE.9)GO TO 35
        HEX(K)='9'
        GO TO 99
35   IF(NREM.NE.8)GO TO 40
        HEX(K)='8'
        GO TO 99
40   IF(NREM.NE.7)GO TO 45
        HEX(K)='7'
        GO TO 99
45   IF(NREM.NE.6)GO TO 50
        HEX(K)='6'
        GO TO 99
50   IF(NREM.NE.5)GO TO 55
        HEX(K)='5'
        GO TO 99
55   IF(NREM.NE.4)GO TO 60
        HEX(K)='4'
        GO TO 99
60   IF(NREM.NE.3)GO TO 65
        HEX(K)='3'

```

```

        GO TO 99
65      IF(NREM,NE,2)GO TO 70
        HEX(K)='2'
        GO TO 99
70      IF(NREM,NE,1)GO TO 75
        HEX(K)='1'
        GO TO 99
75      IF(NREM,NE,0)GO TO 80
        HEX(K)='0'
        GO TO 99
80      WRITE(5,111)
111     FORMAT(1X,'NOT HEX ERROR - FATAL')
        STOP
99      REM=FLOATI(NREM)
        ANUM=ANUM-REM*(16,**(K-1))
100     CONTINUE
        RETURN
999     WRITE(5,222)
222     FORMAT(1X,'OUT OF BOUNDS IN HEX SUB, - FATAL')
        STOP
        END

```

```

C
C
HEX TO DECIMAL CONVERSION SUBROUTINE
SUBROUTINE HEXNUM(HEX,NUMBER)
CHARACTER*1 HEX(4)
INTEGER*4 NUMBER,MULT,NDIG,K
NUMBER=0
DO 200 K=1,4
  IF(HEX(K).NE,'F')GO TO 5
    NDIG=15
    GO TO 111
5    IF(HEX(K).NE,'E')GO TO 10
    NDIG=14
    GO TO 111
10   IF(HEX(K).NE,'D')GO TO 15
    NDIG=13
    GO TO 111
15   IF(HEX(K).NE,'C')GO TO 20
    NDIG=12
    GO TO 111
20   IF(HEX(K).NE,'B')GO TO 25
    NDIG=11
    GO TO 111
25   IF(HEX(K).NE,'A')GO TO 30
    NDIG=10
    GO TO 111
30   IF(HEX(K).NE,'9')GO TO 35
    NDIG=9
    GO TO 111
35   IF(HEX(K).NE,'8')GO TO 40
    NDIG=8
    GO TO 111
40   IF(HEX(K).NE,'7')GO TO 45
    NDIG=7
    GO TO 111
45   IF(HEX(K).NE,'6')GO TO 50
    NDIG=6
    GO TO 111
50   IF(HEX(K).NE,'5')GO TO 55
    NDIG=5
    GO TO 111
55   IF(HEX(K).NE,'4')GO TO 60
    NDIG=4
    GO TO 111
60   IF(HEX(K).NE,'3')GO TO 65
    NDIG=3
    GO TO 111
65   IF(HEX(K).NE,'2')GO TO 70
    NDIG=2
    GO TO 111
70   IF(HEX(K).NE,'1')GO TO 75
    NDIG=1
    GO TO 111
75   IF(HEX(K).NE,'0')GO TO 80
    NDIG=0
    GO TO 111
80   WRITE(5,100)
100  FORMAT(1X,'NOT HEX ERROR - FATAL')
    STOP
111  MULT=(16**(K-1))*NDIG
    NUMBER=NUMBER+MULT
200  CONTINUE
    RETURN
    END

```

C

```

DECIMAL TO BINARY SUBROUTINE
SUBROUTINE NUMBIN(NUMBER,BIN32,BIN16,NFLAG)
CHARACTER*1 BIN32(32),BIN16(16)
INTEGER*4 NUMBER,N,NUM2
NFLAG=0
NUM2=NUMBER
DO 60 K=1,32
    BIN32(K)='0'
60  CONTINUE
DO 70 K=1,16
    BIN16(K)='0'
70  CONTINUE
DO 100 N=31,1,-1
    IF(NUMBER.LT.(2**(N-1)))GO TO 100
    BIN32(N)='1'
    NUMBER=NUMBER-(2**(N-1))
100 CONTINUE
DO 200 N=16,1,-1
    IF(NUM2.LT.(2**(N-1)))GO TO 200
    BIN16(N)='1'
    NUM2=NUM2-(2**(N-1))
200 CONTINUE
IF(NUM2.GT,0)NFLAG=1
RETURN
END

```


SUBS3.FTN

```

C
C      ADDRESS LOCATION SUBROUTINE
SUBROUTINE ADRLOC(NUM,LOC)
CHARACTER*1 NUM,LOC(3)
DO 33 K=1,3
    LOC(K)='0'
33    CONTINUE
    IF(NUM.NE.'0')GO TO 5
        GO TO 250
5      IF(NUM.NE.'1')GO TO 10
        LOC(1)='1'
        GO TO 250
10     IF(NUM.NE.'2')GO TO 15
        LOC(2)='1'
        GO TO 250
15     IF(NUM.NE.'4')GO TO 20
        LOC(3)='1'
        GO TO 250
20     IF(NUM.NE.'3')GO TO 50
        LOC(1)='1'
        LOC(2)='1'
        GO TO 250
50     IF(NUM.NE.'5')GO TO 100
        LOC(1)='1'
        LOC(3)='1'
        GO TO 250
100    IF(NUM.NE.'6')GO TO 150
        LOC(2)='1'
        LOC(3)='1'
        GO TO 250
150    IF(NUM.NE.'7')GO TO 200
        LOC(1)='1'
        LOC(2)='1'
        LOC(3)='1'
        GO TO 250
200    WRITE(5,333)NUM
333    FORMAT(1X,A1,' IS NOT A VALID REGISTER NUMBER - EXECUTION
      s  TERMINATED')
      STOP
250    RETURN
      END

```

```

C
C      ADDRESS MODE AND LOCATION SUBROUTINE
SUBROUTINE TADR(ADRES,MODE,REG,NUM,TYPE,FLG)
CHARACTER*9 ADRES,SWITCH,ADRES1
CHARACTER*1 MODE(3),REG(3),TYPE,R,HEX(4)
CHARACTER*1 BIN16(16),BIN32(32)
INTEGER FLG,NO
INTEGER*4 NUM,NUMBER
IF(ADRES(:1).EQ.'D')GO TO 100
IF(ADRES(:1).EQ.'A')GO TO 200
IF(ADRES(:1).EQ.'#')GO TO 300
IF(ADRES(:1).EQ.'+')GO TO 400
IF(ADRES(:1).EQ.'-')GO TO 450
IF(ADRES(:1).EQ.'%')GO TO 500
IF(ADRES(:1).EQ.'P')GO TO 700
IF(ADRES(:1).EQ.'S')GO TO 600
IF(ADRES(:1).EQ.'*')GO TO 800
WRITE(5,77)ADRES(1:1)
77    FORMAT(1X,'IMPROPER ADRESSING SPECIFIER : ',A1,' FATAL')
      STOP

```

```

C
C      DATA REGISTER DIRECT      A10

```

```

100  TYPE='0'
    MODE(1)='0'
    MODE(2)='0'
    MODE(3)='0'
    GO TO 250

C
C  ADDRESS REGISTER DIRECT
200  TYPE='1'
    MODE(1)='1'
    MODE(2)='0'
    MODE(3)='0'
    250  R=ADRES(2:2)
        CALL ADRLOC(R,REG)
        FLG=0
        GO TO 900

C
C  ADDRESS REGISTER INDIRECT
300  TYPE='1'
    R=ADRES(3:3)
    CALL ADRLOC(R,REG)
    MODE(1)='0'
    MODE(2)='1'
    MODE(3)='0'
    FLG=0
    GO TO 900

C
C  ADDRESS REGISTER IND, WITH POST INCREMENT
400  TYPE='1'
    R=ADRES(3:3)
    CALL ADRLOC(R,REG)
    MODE(1)='1'
    MODE(2)='1'
    MODE(3)='0'
    FLG=0
    GO TO 900

C
C  ADDRESS REGISTER IND, WITH PRE-DECREMENT
450  TYPE='1'
    R=ADRES(3:3)
    CALL ADRLOC(R,REG)
    MODE(1)='0'
    MODE(2)='0'
    MODE(3)='1'
    FLG=0
    GO TO 900

C
C  ADDRESS REGISTER INDIRECT WITH DISPLACEMENT
500  TYPE='1'
    NO=0
    IF(ADRES(2:2).EQ.'S')GO TO 520
    IF(ADRES(2:2).NE.'-')GO TO 505
        ADRES1=ADRES(3:7)
        NO=1
        GO TO 510
505  ADRES1=ADRES(2:6)
510  OPEN(UNIT=2,FILE='TEMP.DAT',STATUS='NEW')
    WRITE(2,515)ADRES1
515  FORMAT(1X,A9)
    REWIND 2
    READ(2,517)NUM
517  FORMAT(I9)
    CLOSE(UNIT=2,STATUS='DELETE')
    GO TO 540
520  M=4
    DO 525 J=3,6
        HEX(M)=ADRES(J:J)

```

```

MEM=1
525  CONTINUE
      CALL HEXNUM(HEX,NUM)
540  IF(ADRES(2:2).NE.'-')GO TO 570
      CALL NUMBIN(NUM,BIN32,BIN16,NF)
      CALL TCOMP(BIN16,BIN32,NF)
      CALL BINDIG(BIN16,NUM)
570  FLG=1
      KL=8+NO
      R=ADRES(KL:KL)
      CALL ADRLOC(R,REG)
      MODE(1)='1'
      MODE(2)='0'
      MODE(3)='1'
      GO TO 900

C
C  ABSOLUTE SHORT
600  TYPE='1'
      FLG=1
      MODE(1)='1'
      MODE(2)='1'
      MODE(3)='1'
      CALL ADRLOC('0',REG)
      CALL KSTRIN(ADRES(2:5),HEX)
      CALL HEXNUM(HEX,NUM)
      GO TO 900

C
C  PC AND DISPLACEMENT
700  TYPE='1'
      FLG=1
      MODE(1)='1'
      MODE(2)='1'
      MODE(3)='1'
      CALL ADRLOC('2',REG)
      NO=0
      IF(ADRES(3:3).EQ.'S')GO TO 720
      IF(ADRES(3:3).NE.'-')GO TO 705
      ADRES1=ADRES(4:8)
      NO=1
      GO TO 710
705  ADRES1=ADRES(3:7)
710  OPEN(UNIT=2,FILE='TEMP.DAT',STATUS='NEW')
      WRITE(2,715)ADRES1
715  FORMAT(1X,A9)
      REWIND 2
      READ(2,717)NUM
717  FORMAT(I9)
      CLOSE(UNIT=2,STATUS='DELETE')
      GO TO 740
720  M=4
      DO 725 J=4,7
          HEX(M)=ADRES(J:J)
          M=M+1
725  CONTINUE
740  IF(ADRES(3:3).NE.'-')GO TO 770
      CALL NUMBIN(NUM,BIN32,BIN16,NF)
      CALL TCOMP(BIN16,BIN32,NF)
      CALL BINDIG(BIN16,NUM)
770  GO TO 900

C
C  IMMEDIATE
800  TYPE='1'
      FLG=1
      MODE(1)='1'
      MODE(2)='1'
      MODE(3)='1'

```

```

      CALL ADRLLOC(14, REB)
      IF(ADRES(2:2), NE, 'S') GO TO 850
      CALL KSTRIN(ADRES(3:6), HEX)
      CALL HEXNUM(HEX, NUM)
      GO TO 900
850   IF(ADRES(2:2), NE, '-') GO TO 855
      ADRES1=ADRES(3:7)
      GO TO 860
855   ADRES1=ADRES(2:6)
860   OPEN(UNIT=2, FILE='TEMP.DAT', STATUS='NEW')
      WRITE(2, 862) ADRES1
862   FORMAT(1X, A9)
      REWIND 2
      READ(2, 844) NUM
844   FORMAT(I9)
      CLOSE(UNIT=2, STATUS='DELETE')
      IF(ADRES(2:2), NE, '-') GO TO 900
      CALL NUMBIN(NUM, BIN32, BIN16, NF)
      CALL TCOMP(BIN16, BIN32, NF)
      CALL BINDIG(BIN16, NUM)
900   RETURN
      END

```

```

C DIRECTIVES : DCB,DCL,DCW
SUBROUTINE DC(LABEL,OPERAT,ADRES1,DCOUNT,NCK)
CHARACTER*6 LABEL,OPERAT
CHARACTER*9 ADRES1,SWITCH
INTEGER Z,NCK,K,NF,NASC,J
INTEGER*4 DCOUNT,CONST,DSAVE,HOLD
INTEGER*4 INTEX,INTEY,INTEZ
BYTE IVAR
CHARACTER*1 SREC(30),ASCII,HEXM(4),HEX(4),HEX2(4)
CHARACTER*1 BIN16(16),BIN32(32),BINT(16),VAR
EQUIVALENCE (IVAR,VAR)
IF(OPERAT(3:3).EQ.'B')GO TO 50
  ACOUNT=FLOATJ(DCOUNT)
  AXX=(ACOUNT/2.)*10.
  INTEX=JINT(ACOUNT/2.)
  INTEZ=10*INTEX
  INTEY=JINT(AXX)
IF(INTEZ,NE,INTEY)DCOUNT=DCOUNT+1
50 Z=0
  IVAR=39
  IF(ADRES1(:1),NE,VAR)GO TO 100
    ASCII=ADRES1(2:2)
    NASC=ICHAR(ASCII)
    REAL=FLOATI(NASC)
    CONST=JIFIX(REAL)
    GO TO 300
100 IF(ADRES1(:1),NE.'-')GO TO 200
    SWITCH=ADRES1
    ADRES1=SWITCH(2:)
    Z=1
200 OPEN(UNIT=2,FILE='TEMP.DAT',STATUS='NEW')
    REWIND 2
    WRITE(2,150)ADRES1
150 FORMAT(1X,A9)
    REWIND 2
    READ(2,111)CONST
111 FORMAT(I9)
    CLOSE(UNIT=2,STATUS='DELETE')
300 DSAVE=DCOUNT
    CALL LABTAB(LABEL,DSAVE,K)
    SREC(1)='S'
    SREC(2)='1'
    IF(OPERAT(3:3).NE.'B')GO TO 400
    CALL DIGHEX(CONST,HEX)
    SREC(3)='0'
    SREC(4)='4'
    SREC(9)=HEX(2)
    SREC(10)=HEX(1)
    CALL DIGHEX(DCOUNT,HEXM)
    DO 310 J=1,4
      SREC(J+4)=HEXM(-1*J+5)
310 CONTINUE
    CALL CKSUM(SREC,0)
    DCOUNT=DCOUNT+1
    GO TO 999
400 IF(OPERAT(3:3).NE.'W')GO TO 500
    CALL DIGHEX(DCOUNT,HEXM)
    SREC(3)='0'
    SREC(4)='5'
    IF(Z,NE,1)GO TO 405
    CALL NUMBIN(CONST,BIN32,BIN16,NF)
    CALL TCOMP(BIN16,BIN32,NF)
    CALL BINDIG(BIN16,CONST) A14

```

```

405      CALL DIGHEX (CONST, HEX)
        DO 410 J=1,4
          SREC(J+4)=HEXM(-1*J+5)
          SREC(J+8)=HEX(-1*J+5)
410      CONTINUE
        CALL CKSUM(SREC,1)
        DCOUNT=DCOUNT+2
        GO TO 999
500      CALL NUMBIN(CONST,BIN32,BIN16,NF)
        IF(Z,NE,1)GO TO 505
        NF=1
        CALL TCOMP(BIN16,BIN32,NF)
505      DO 510 J=17,32
          BINT(J-16)=BIN32(J)
510      CONTINUE
        CALL BINDIG(BINT,HOLD)
        CALL DIGHEX(HOLD,HEX)
        DO 520 J=1,16
          BIN16(J)=BIN32(J)
520      CONTINUE
        CALL BINDIG(BIN16,HOLD)
        CALL DIGHEX(HOLD,HEX2)
        SREC(3)='0'
        SREC(4)='7'
        CALL DIGHEX(DCOUNT,HEXM)
        DO 530 J=1,4
          SREC(J+4)=HEXM(-1*J+5)
          SREC(J+8)=HEX(-1*J+5)
          SREC(J+12)=HEX2(-1*J+5)
530      CONTINUE
        CALL CKSUM(SREC,2)
        DCOUNT=DCOUNT+4
C
999      NCK=1
        WRITE(4,1000)(SREC(J),J=1,30)
1000     FORMAT(1X,30A1)
        RETURN
        END

```

SUBDIR.FTN

```

C      DIRECTIVE : EQU
      SUBROUTINE EQU(LABEL,ADRES,NCK)
      CHARACTER*6 LABEL
      CHARACTER*9 ADRES
      CHARACTER*1 REG(3),MODE(3)
      INTEGER*4 NUM
      IF(NPASS.EQ.2)GO TO 100
      CALL TADR(ADRES,MODE,REG,NUM)
      CALL LABTAB(LABEL,NUM,NB)
100    NCK=1
      RETURN
      END

C
C      DIRECTIVE : DS
      SUBROUTINE DS(LABEL,OPERAT,DCOUNT,NCK)
      CHARACTER*6 LABEL,OPERAT
      INTEGER NCK,K
      INTEGER*4 DCOUNT,INTEZ,INTEY,DSAVE
      IF(OPERAT(3:3).EQ.'B')GO TO 100
      ACOUNT=FLOATJ(DCOUNT)
      AXX=(ACOUNT/2.)*10.
      INTEZ=10*JINT(ACOUNT/2.)
      INTEY=JINT(AXX)
      IF(INTEZ.NE.INTEY)DCOUNT=DCOUNT+1
100    DSAVE=DCOUNT
      CALL LABTAB(LABEL,DSAVE,K)
      IF(OPERAT(3:3).NE.'B')GO TO 200
      DCOUNT=DCOUNT+1
      GO TO 500
200    IF(OPERAT(3:3).NE.'L')GO TO 300
      DCOUNT=DCOUNT+4
      GO TO 500
300    DCOUNT=DCOUNT+2
500    NCK=1
      RETURN
      END

C
C      DIRECTIVE : END
      SUBROUTINE END(PCONT2,NSTOP)
      INTEGER*4 PCONT2
      CHARACTER*4 NSTOP
      CHARACTER*1 SREC(30),HEX(4)
      SREC(1)='S'
      SREC(2)='9'
      DO 50 J=1,6
      SREC(J+2)='0'
50    CONTINUE
      WRITE(4,100)(SREC(J),J=1,8)
100    FORMAT(1X,8A1)
      NSTOP='STOP'
      RETURN
      END

C
C
C      ADDRESS DIRECT DEST, ADD,SUB
      SUBROUTINE OPTA(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,
$NWORDS,HEXM,BIN1,BIN2)
      COMMON/BLOCK1/NPASS
      CHARACTER*6 LABEL,OPERAT
      CHARACTER*9 ADRES1,ADRES2
      CHARACTER*1 BIN1(16),BIN2(16),BIN32(32),HEXM(4)
      CHARACTER*1 REG(3),DREG(3),MODE(3),TYPE
      INTEGER FLG,NWORDS
      .....
      A16

```

```

      INTEGER*4 PCOUNT,NUM
C
      IF(NPASS.NE.1)GO TO 20
      CALL LABTAB(LABEL,PCOUNT,NK)
      IF(ADRES1(1:1).EQ.'(')ADRES1='$0000'
      GO TO 30
20    CALL LABAD(ADRES1,ADRES2)
30    CALL DIGHEX(PCOUNT,HEXM)
C
      IF(ADRES2(1:1).NE.'A')GO TO 50
C
      IF(OPERAT(1:5).NE.'MOVEA')GO TO 34
      DO 32 J=1,16
      BIN1(J)='0'
32    CONTINUE
      BIN1(14)='1'
      BIN1(7)='1'
      IF(OPERAT(6:6).NE.'W')GO TO 33
      BIN1(13)='1'
      GO TO 58
33    IF(OPERAT(6:6).NE.'L')GO TO 50
      GO TO 58
34    DO 35 J=1,16
      BIN1(J)='1'
35    CONTINUE
      IF(OPERAT(5:5).NE.'W')GO TO 40
      BIN1(9)='0'
      GO TO 55
40    IF(OPERAT(5:5).NE.'L')GO TO 50
      GO TO 55
50    WRITE(5,52)OPERAT
52    FORMAT(1X,'IMPROPER SIZE SPEC OR ADDRESS MODE FOR :',A6)
      STOP
55    BIN1(14)='0'
      IF(OPERAT(1:3).EQ.'SUB')BIN1(15)='0'
C
58    CALL TADR(ADRES2,MODE,DREG,NUM,TYPE,FLG)
      CALL TADR(ADRES1,MODE,REG,NUM,TYPE,FLG)
      DO 60 J=1,3
      BIN1(J+9)=DREG(J)
      BIN1(J)=REG(J)
      BIN1(J+3)=MODE(J)
60    CONTINUE
      NWORDS=1
      PCOUNT=PCOUNT+2
      IF(FLG.EQ.0)GO TO 100
      NWORDS=2
      PCOUNT=PCOUNT+2
      CALL NUMBIN(NUM,BIN32,BIN2,NZ)
100   RETURN
      END
C
C
      NO OPERATION,STOP
      SUBROUTINE NOP(LABEL,OPERAT,PCOUNT,NWORDS,HEXM,BIN1)
      COMMON/BLOCK1/NPASS
      CHARACTER*6 LABEL,OPERAT
      INTEGER*4 PCOUNT
      CHARACTER*1 BIN1(16),HEXM(4)
      IF(NPASS.NE.1)GO TO 20
      CALL LABTAB(LABEL,PCOUNT,K)
20    CALL DIGHEX(PCOUNT,HEXM)
      NWORDS=1
      PCOUNT=PCOUNT+2
      DO 30 J=1,16
      BIN1(J)='0'

```



```

30  CONTINUE
    BIN1(15)='1'
    BIN1(12)='1'
    BIN1(11)='1'
    BIN1(10)='1'
    BIN1(7)='1'
    BIN1(6)='1'
    BIN1(5)='1'
    BIN1(1)='1'
    IF(OPERAT(1:4).NE.'STOP')GO TO 40
        BIN1(2)='1'
        BIN1(1)='0'
40  CONTINUE
    RETURN
    END

```

C
C
C

```

    JUMP,JUMP TO SUBROUTINE (JMP,JSR)
    SUBROUTINE JUMP(LABEL,OPERAT,ADRES1,PCOUNT,NWORDS,
$HEXM,BIN1,BIN2)
    COMMON/BLOCK1/NPASS
    CHARACTER*1 BIN1(16),BIN2(16),BIN32(32),HEXM(4)
    CHARACTER*1 MODE(3),REG(3),TYPE
    CHARACTER*6 LABEL,OPERAT
    CHARACTER*9 ADRES1,ADRES0
    INTEGER*4 PCOUNT,NUM
    INTEGER FLG
    ADRES0='$0000'
    IF(NPASS.NE.1)GO TO 20
        CALL LABTAB(LABEL,PCOUNT,NA)
        IF(ADRES1(1:1).EQ.'(')ADRES1='$0000'
        GO TO 30
20  CALL LABAD(ADRES1,ADRES0)
30  CALL DIGHEX(PCOUNT,HEXM)
    DO 40 J=1,16
        BIN1(J)='1'
40  CONTINUE
    BIN1(16)='0'
    BIN1(14)='0'
    BIN1(13)='0'
    BIN1(9)='0'
    IF (OPERAT(1:3).NE.'RTS')GO TO 45
        FLG=0
        BIN1(2)='0'
        BIN1(4)='0'
        BIN1(8)='0'
        GO TO 50
45  CALL TADR(ADRES1,MODE,REG,NUM,TYPE,FLG)
    DO 50 J=1,3
        BIN1(J)=REG(J)
        BIN1(J+3)=MODE(J)
50  CONTINUE
    NWORDS=1
    PCOUNT=PCOUNT+2
    IF(FLG.NE.1)GO TO 70
        NWORDS=2
        PCOUNT=PCOUNT+2
        CALL NUMBIN(NUM,BIN32,BIN2,NZ)
70  IF(OPERAT(2:3).EQ.'SR')BIN1(7)='0'
    RETURN
    END

```

C
C
C

```

    SUBROUTINE MULDIV,MULTIPLY,DIVIDE
    SUBROUTINE MULDIV(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,
$NWORDS,HEXM,BIN1,BIN2)
    A18

```

```

COMMON/BLOCK1/NPASS
CHARACTER*6 LABEL, OPERAT
CHARACTER*9 ADRES1, ADRES2
CHARACTER*1 DREG(3), HEXM(4), BIN32(32), MODE(3), REG(3)
CHARACTER*1 TYPE, BIN1(16), BIN2(16)
INTEGER*4 NUM, PCOUNT
INTEGER FLG

```

```

C
  IF(NPASS, NE, 1) GO TO 20
    CALL LABTAB(LABEL, PCOUNT, NA)
    IF(ADRES1(1:1), EQ, '(') ADRES1='S0000'
    GO TO 30
20  CALL LABAD(ADRES1, ADRES2)
30  CALL DIGHEX(PCOUNT, HEXM)
    DO 40 J=1, 16
      BIN1(J)='1'
40  CONTINUE
    BIN1(13)='0'
    BIN1(14)='0'
    IF(OPERAT(1:3), EQ, 'DIV') BIN1(15)='0'
    IF(OPERAT(4:4), EQ, 'S') GO TO 50
    BIN1(9)='0'
50  IF(ADRES2(1:1), EQ, 'D') GO TO 60
    WRITE(5, 555) OPERAT
555  FORMAT(1X, 'IMPROPER ADDRESSING FOR :', A6)
    STOP
60  CALL TADR(ADRES2, MODE, DREG, NUM, TYPE, FLG)
    CALL TADR(ADRES1, MODE, REG, NUM, TYPE, FLG)
    DO 70 J=1, 3
      BIN1(J)=REG(J)
      BIN1(J+3)=MODE(J)
      BIN1(J+9)=DREG(J)
70  CONTINUE
    NWORDS=1
    PCOUNT=PCOUNT+2
    IF(FLG, NE, 1) GO TO 80
    NWORDS=2
    PCOUNT=PCOUNT+2
    CALL NUMBIN(NUM, BIN32, BIN2, NZ)
80  RETURN
    END

```

```

C
C
C
  NEX, NEG
  SUBROUTINE NEG(LABEL, OPERAT, ADRES1, PCOUNT, NWORDS,
    $HEXM, BIN1, BIN2)
    COMMON/BLOCK1/NPASS
    CHARACTER*1 BIN1(16), BIN2(16), BIN32(32), HEXM(4)
    CHARACTER*1 MODE(3), REG(3), TYPE
    INTEGER FLG
    INTEGER*4 PCOUNT, NUM
    CHARACTER*9 ADRES1, DUMMY
    CHARACTER*6 LABEL, OPERAT
    DUMMY='S0000'
    IF(NPASS, NE, 1) GO TO 100
    CALL LABTAB(LABEL, PCOUNT, NK)
    IF(ADRES1(1:1), EQ, '(') ADRES1='S0000'
    GO TO 150
100  CALL LABAD(ADRES1, DUMMY)
150  CALL DIGHEX(PCOUNT, HEXM)
    DO 200 J=1, 16
      BIN1(J)='0'
200  CONTINUE
    IF(OPERAT(4:4), EQ, 'B') GO TO 240
    IF(OPERAT(4:4), NE, 'W') GO TO 210
    BIN1(7)='1'

```

```

      GO TO 240
10  IF(OPERAT(4:4).NE,'L')GO TO 220
      BIN1(8)='1'
      GO TO 240
20  WRITE(5,225)OPERAT
25  FORMAT(1X,'IMPROPER SIZE SPEC FOR ',A6)
      STOP
40  CALL TADR(ADRES1,MODE,REG,NUM,TYPE,FLG)
      DO 250 J=1,3
          BIN1(J)=REG(J)
          BIN1(J+3)=MODE(J)
50  CONTINUE
      NWORDS=1
      PCOUNT=PCOUNT+2
      CALL NUMBIN(NUM,BIN32,BIN2,NZ)
60  BIN1(15)='1'
      IF(OPERAT(1:3).EQ,'NEG')BIN1(11)='1'
      RETURN
      END

```

```

      SWAP
      SUBROUTINE SWAP(LABEL,OPERAT,ADRES1,PCOUNT,NWORDS,
$HEXM,BIN1)
      COMMON/BLOCK1/NPASS
      CHARACTER*1 MODE(3),REG(3),HEXM(4),BIN1(16)
      CHARACTER*1 BIN32(32),TYPE
      INTEGER FLG
      INTEGER*4 PCOUNT,NUM
      CHARACTER*6 LABEL,OPERAT
      CHARACTER*9 ADRES1
      IF(ADRES1(1:1).NE,'D')GO TO 60
      IF(NPASS.NE.1)GO TO 20
          CALL LABTAB(LABEL,PCOUNT,NA)
20  CALL DIGHEX(PCOUNT,HEXM)
      DO 30 J=1,16
          BIN1(J)='0'
50  CONTINUE
      BIN1(15)='1'
      BIN1(12)='1'
      BIN1(7)='1'
      CALL TADR(ADRES1,MODE,REG,NUM,TYPE,FLG)
      DO 40 J=1,3
          BIN1(J)=REG(J)
40  CONTINUE
      PCOUNT=PCOUNT+2
      NWORDS=1
      RETURN
60  WRITE(5,70)OPERAT
70  FORMAT(1X,'IMPROPER ADDRESS FOR ',A6)
      END

```

```

C      SEPARATES STRING INTO ARRAY
      SUBROUTINE KSTRIN(SOLID,SEP)
      CHARACTER*4 SOLID
      CHARACTER*1 SEP(4)
      DO 100 J=1,4
         K=5-J
         SEP(J)=SOLID(K:K)
100    CONTINUE
      RETURN
      END

```

```

C
C      16 OR 32 BIT TWO'S COMPLIMENT
      SUBROUTINE TCOMP(BIN16,BIN32,NF)
      CHARACTER*1 BIN16(16),BIN32(32)
      INTEGER NF
      IF(NF,NE,0)GO TO 100
         DO 50 K=1,16
            IF(BIN16(K).EQ.'0')GO TO 40
               BIN16(K)='0'
               GO TO 50
            BIN16(K)='1'
40         CONTINUE
         DO 75 K=1,16
            IF(BIN16(K).NE.'0')GO TO 60
               BIN16(K)='1'
               GO TO 90
            BIN16(K)='0'
60         CONTINUE
         GO TO 200
90         DO 150 K=1,32
            IF(BIN32(K).EQ.'0')GO TO 140
               BIN32(K)='0'
               GO TO 150
            BIN32(K)='1'
140        CONTINUE
         DO 175 K=1,32
            IF(BIN32(K).NE.'0')GO TO 160
               BIN32(K)='1'
               GO TO 200
            BIN32(K)='0'
160        CONTINUE
175        RETURN
200        END

```

```

C
C      16 OR 32 BIT ONE'S COMPLIMENT
      SUBROUTINE OCOMP(BIN16,BIN32,NF)
      CHARACTER*1 BIN16(16),BIN32(32)
      INTEGER NF
      IF(NF,NE,0)GO TO 100
         DO 50 K=1,16
            IF(BIN16(K).EQ.'0')GO TO 40
               BIN16(K)='0'
               GO TO 50
            BIN16(K)='1'
40         CONTINUE
         GO TO 200
100        DO 150 K=1,32
            IF(BIN32(K).EQ.'0')GO TO 140
               BIN32(K)='0'
               GO TO 150
            BIN32(K)='1'
140        CONTINUE
150        RETURN
200

```

END

C
C

```
CHECKSUM SUBROUTINE
SUBROUTINE CKSUM(SREC,LENGTH)
COMMON/BLOCK1/NPASS
COMMON/BLOCK2/LABEL,OPERAT,ADRES1,ADRES2
INTEGER LENGTH,NZ,D,NPASS
CHARACTER*1 SREC(30),HEX(4),BIN16(16),BIN32(32),SPEC(30)
CHARACTER*6 LABEL,OPERAT
CHARACTER*9 ADRES1,ADRES2
INTEGER*4 SUM,NUM,CSUM
HEX(4)='0'
HEX(3)='0'
SUM=0
N=2
```

```
IF(LENGTH.EQ.0)NZ=4
IF(LENGTH.EQ.1)NZ=5
IF(LENGTH.EQ.2)NZ=7
IF(LENGTH.EQ.3)NZ=9
DO 100 J=1,NZ
```

```
    HEX(2)=SREC(N+1)
    HEX(1)=SREC(N+2)
    CALL HEXNUM(HEX,NUM)
    SUM=SUM+NUM
    N=N+2
```

```
100  CONTINUE
    CALL NUMBIN(SUM,BIN32,BIN16,D)
    CALL OCOMP(BIN16,BIN32,0)
    CALL BINDIG(BIN16,CSUM)
    CALL DIGHEX(CSUM,HEX)
    SREC(N+1)=HEX(2)
    SREC(N+2)=HEX(1)
    IF(NPASS.NE.2)GO TO 200
```

```
    DO 110 JK=1,25
        SPEC(JK)=SREC(JK)
```

```
110  CONTINUE
    SPEC(N+1)=' '
    SPEC(N+2)=' '
    WRITE(11,120)LABEL,OPERAT,ADRES1,ADRES2,(SPEC(JZ),JZ=5,20)
120  FORMAT(1X,A6,T9,A6,T17,A9,T28,A9,T40,4A1,T50,2A1,1X,2A1,
```

```
51X,2A1,1X,2A1,1X,2A1,1X,2A1)
```

```
200  RETURN
    END
```

C
C

```
LABEL/LOCATION
SUBROUTINE LABTAB(LABEL,PLACE,NK)
CHARACTER*6 LABEL,LARRY(100)
INTEGER*4 PLACE,LOCAT(100)
COMMON/BLOCK1/NPASS
IF(LABEL.EQ.'XSTART')N=1
IF(NPASS.EQ.2)GO TO 100
    LARRY(N)=LABEL
    LOCAT(N)=PLACE
    N=N+1
    GO TO 200
```

```
100  DO 150 K=1,100
    IF(LABEL.NE.LARRY(K))GO TO 150
        PLACE=LOCAT(K)
        GO TO 200
```

```
150  CONTINUE
200  RETURN
    END
```

C
C

```
LABEL/ADRESS SUBROUTINE
SUBROUTINE LABAD(ADRES1,ADRES2)
CHARACTER*1 HEX(4),HEX2(4)
```

```

      INTEGER*4 PLACE,PLACE2
      CHARACTER*9 ADRES1,ADRES2
      IF(ADRES1(1:1).NE.'(')GO TO 110
        CALL LABTAB(ADRES1(2:),PLACE,NK)
        CALL DIGHEX(PLACE,HEX)
        OPEN(UNIT=2,FILE='TMP,DAT',STATUS='NEW')
        WRITE(2,120)(HEX(J),J=4,1,-1)
120      FORMAT(1X,'S',4A1)
        REWIND 2
        CLOSE (UNIT=2,STATUS='KEEP')
        OPEN (UNIT=2,FILE='TMP,DAT',STATUS='OLD')
        READ(2,130)ADRES1
130      FORMAT(T2,A9)
        CLOSE(UNIT=2,STATUS='DELETE')
110      IF(ADRES2(1:1).NE.'(')GO TO 140
        CALL LABTAB(ADRES2(2:),PLACE2,NK)
        CALL DIGHEX(PLACE2,HEX2)
        OPEN(UNIT=2,FILE='TCP,DAT',STATUS='NEW')
        WRITE(2,135)(HEX2(J),J=4,1,-1)
135      REWIND 2
        FORMAT(1X,'S',4A1)
        REWIND 2
        READ(2,138)ADRES2
138      FORMAT(T2,A9)
        CLOSE(UNIT=2,STATUS='DELETE')
140      RETURN
      END

```

```

C      BTST = BIT TEST
      SUBROUTINE TEST(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,
      SHEXM,BIN1,BIN2,BIN3)
      COMMON/BLOCK1/NPASS
      CHARACTER*1 BIN1(16),BIN2(16),BIN32(32),HEXM(4)
      CHARACTER*1 MODE(3),REG(3),TYPE
      CHARACTER*6 LABEL,OPERAT
      CHARACTER*9 ADRES1,ADRES2
      INTEGER*4 PCOUNT,NUM
      INTEGER FLG
      IF(NPASS.NE.1)GO TO 20
        CALL LABTAB(LABEL,PCOUNT,NA)
        IF(ADRES2(1:1).EQ.'(')ADRES2='S0000'
        GO TO 30
20      CALL LABAD(ADRES1,ADRES2)
30      CALL DIGHEX(PCOUNT,HEXM)
      DO 40 J=1,16
        BIN1(J)='0'
40      CONTINUE
      BIN1(12)='1'
      CALL TADR(ADRES1,MODE,REG,NUM,TYPE,FLG)
      CALL NUMBIN(NUM,BIN32,BIN2,NZ)
      NWORDS=2
      CALL TADR(ADRES2,MODE,REG,NUM,TYPE,FLG)
      DO 50 J=1,3
        BIN1(J)=REG(J)
        BIN1(J+3)=MODE(J)
50      CONTINUE
      PCOUNT=PCOUNT+4
      IF(FLG.NE.1)GO TO 70
      NWORDS=3
      PCOUNT=PCOUNT+2
      CALL NUMBIN(NUM,BIN32,BIN3,NZ)
70      RETURN
      END

```

```

C      OPERATION CODE SUBROUTINES
C
C      ADD, AND, ORR, CMP, SUB
C
      SUBROUTINE ANDADD(LABEL, OPERAT, ADRES1, ADRES2, PCOUNT, NWORDS,
&HEXM, BIN1, BIN2)
      COMMON/BLOCK1/NPASS
      CHARACTER*1 BIN1(16), BIN2(16), HEXM(4), SD, BIN32(32)
      CHARACTER*1 REG1(3), REG2(3), MODE1(3), MODE2(3), TYPE
      INTEGER NWORDS, FLG1, FLG2
      INTEGER*4 PCOUNT, NUM1, NUM2
      CHARACTER*9 ADRES1, ADRES2
      CHARACTER*6 LABEL, OPERAT
C
      IF(NPASS.NE.1)GO TO 100
      CALL LABTAB(LABEL, PCOUNT, NA)
      IF(ADRES1(1:1).EQ.'(')ADRES1='$0000'
      IF(ADRES2(1:1).EQ.'(')ADRES2='$0000'
      GO TO 150
100    CALL LABAD(ADRES1, ADRES2)
150    CALL DIGHEX(PCOUNT, HEXM)
C
      IF(OPERAT(4:4).EQ.'1')BIN1(9)='0'
      IF(OPERAT(4:4).EQ.'2')BIN1(9)='1'
      IF(OPERAT(5:5).NE.'B')GO TO 160
      BIN1(7)='0'
      BIN1(8)='0'
      GO TO 170
160    IF(OPERAT(5:5).NE.'L')GO TO 165
      BIN1(7)='0'
      BIN1(8)='1'
      GO TO 170
165    IF(OPERAT(5:5).NE.'W')GO TO 167
      BIN1(7)='1'
      BIN1(8)='0'
      GO TO 170
167    WRITE(5,169)OPERAT
169    FORMAT(1X,'IMPROPER SIZE SPEC IN : ',A6,' INSTRUCTION')
      STOP
170    CALL TADR(ADRES1, MODE1, REG1, NUM1, TYPE, FLG1)
      CALL TADR(ADRES2, MODE2, REG2, NUM2, TYPE, FLG2)
      IF(BIN1(9).NE.'0')GO TO 200
      DO 180 J=1,3
      BIN1(J)=REG1(J)
      BIN1(J+3)=MODE1(J)
180    CONTINUE
      IF(FLG1.NE.1)GO TO 190
      NWORDS=2
      PCOUNT=PCOUNT+4
      CALL NUMBIN(NUM1, BIN32, BIN2, NZ)
      GO TO 195
190    NWORDS=1
      PCOUNT=PCOUNT+2
195    DO 199 J=1,3
      BIN1(J+9)=REG2(J)
199    CONTINUE
      GO TO 250
200    DO 210 J=1,3
      BIN1(J)=REG2(J)
      BIN1(J+3)=MODE2(J)
210    CONTINUE
      IF(FLG2.NE.1)GO TO 215

```

```

      NWORDS=2
      PCOUNT=PCOUNT+4
      CALL NUMBIN(NUM2,BIN32,BIN2,NZ)
      GO TO 225
215  NWORDS=1
      PCOUNT=PCOUNT+2
225  DO 229 J=1,3
      BIN1(J+9)=REG1(J)
229  CONTINUE
250  BIN1(13)='1'
      BIN1(14)='0'
      BIN1(15)='1'
      BIN1(16)='1'
C
      IF(OPERAT(2:2).EQ.'N')BIN1(13)='0'
C
      IF(OPERAT(2:2).NE.'U')GO TO 700
      BIN1(15)='0'
      GO TO 900
C
700  IF(OPERAT(3:3).NE.'R')GO TO 800
      BIN1(13)='0'
      BIN1(15)='0'
      GO TO 900
800  IF(OPERAT(3:3).NE.'P')GO TO 850
      BIN1(14)='1'
      BIN1(15)='0'
      GO TO 900
850  CONTINUE
900  RETURN
      END
C
C
C
      MOVE COMMAND
      SUBROUTINE MOVE(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,HEXM,
&BIN1,BIN2,BIN3)
      COMMON/BLOCK1/NPASS
      CHARACTER*1 BIN1(16),BIN2(16),BIN3(16),BIN32(32),HEXM(4)
      CHARACTER*1 TYPE,REG1(3),REG2(3),MODE1(3),MODE2(3)
      INTEGER NWORDS,FLG1,FLG2
      INTEGER*4 PCOUNT,NUM1,NUM2,NTRAC
      CHARACTER*9 ADRES1,ADRES2
      CHARACTER*6 LABEL,OPERAT
      IF(NPASS.NE.1)GO TO 100
      CALL LABTAB(LABEL,PCOUNT,NK)
      IF(ADRES1(1:1).EQ.'(')ADRES1='$0000'
      IF(ADRES2(1:1).EQ.'(')ADRES2='$0000'
      GO TO 150
100  CALL LABAD(ADRES1,ADRES2)
150  CALL DIGHEX(PCOUNT,HEXM)
      IF(OPERAT(5:5).NE.'B')GO TO 160
      BIN1(13)='1'
      BIN1(14)='0'
      GO TO 170
160  IF(OPERAT(5:5).NE.'L')GO TO 165
      BIN1(13)='0'
      BIN1(14)='1'
      GO TO 170
165  IF(OPERAT(5:5).NE.'W')GO TO 167
      BIN1(13)='1'
      BIN1(14)='1'
      GO TO 170
167  WRITE(5,169)OPERAT
169  FORMAT(1X,'IMPROPER SIZED SPEC IN :',A6,' INSTRUCTION')
      STOP
170  BIN1(15)='0'

```



```

BIN1(16)='0'
CALL TADR(ADRES1,MODE1,REG1,NUM1,TYPE,FLG1)
CALL TADR(ADRES2,MODE2,REG2,NUM2,TYPE,FLG2)
DO 175 J=1,3
  BIN1(J)=REG1(J)
  BIN1(J+3)=MODE1(J)
  BIN1(J+6)=MODE2(J)
  BIN1(J+9)=REG2(J)

```

```

175 CONTINUE
  IF(FLG1,NE,1)GO TO 200
  NTRAC=4
  NWORDS=2
  CALL NUMBIN(NUM1,BIN32,BIN2,NZ)
  GO TO 210
200 NTRAC=2
  NWORDS=1
210 IF(FLG2,NE,1)GO TO 250
  NTRAC=NTRAC+2
  NWORDS=NWORDS+1
  IF(NWORDS,EQ,3)GO TO 240
  CALL NUMBIN(NUM2,BIN32,BIN2,NZ)
  GO TO 250
240 CALL NUMBIN(NUM2,BIN32,BIN3,NZ)
250 PCOUNT=PCOUNT+NTRAC
  RETURN
  END

```

C
C

```

SUBROUTINE CMP(OPERAT)
CHARACTER*6 OPERAT
IF(OPERAT(4:4),NE,'B')GO TO 50
  OPERAT='CMP1B'
  GO TO 100
50 IF(OPERAT(4:4),NE,'L')GO TO 60
  OPERAT='CMP1L'
  GO TO 100
60 OPERAT='CMP1W'
100 RETURN
  END

```

C
C
C

```

EOR (BIARY CODE SIMILAR TO CMP)
SUBROUTINE EOR(OPERAT)
CHARACTER*6 OPERAT
IF(OPERAT(4:4),NE,'B')GO TO 50
  OPERAT='CMP2B'
  GO TO 100
50 IF(OPERAT(4:4),NE,'L')GO TO 60
  OPERAT='CMP2L'
  GO TO 100
60 OPERAT='CMP2W'
100 RETURN
  END

```

C

```

ARITH, SHIFT LEFT , RIGHT / LOGICAL SHIFTS
SUBROUTINE AS(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,NWORDS,
$HEXM,BIN1,BIN2)
COMMON/BLOCK1/NPASS
CHARACTER*1 HEXM(4),BIN1(16),BIN2(16),BIN32(32)
CHARACTER*1 TYPE,MODE(3),REG(3)
CHARACTER*6 LABEL,OPERAT
CHARACTER*9 ADRES1,ADRES2
INTEGER*4 PCOUNT,NUM
INTEGER NWORDS,FLG

```

C

```

  IF(NPASS,NE,1)GO TO 20
  CALL LABTAB(LABEL,PCOUNT,NK) A26
  .....

```

```

        IF (ADRES1(1:1),EQ,'(')ADRES1='50000'
        IF (ADRES2(1:1),EQ,'(')ADRES2='50000'
        GO TO 30
20      CALL LABAD(ADRES1,ADRES2)
30      CALL DIGHEX(PCOUNT,HEXM)
C
        IF (OPERAT(5:5),NE,'B')GO TO 50
        BIN1(7)='0'
        BIN1(8)='0'
        GO TO 100
50      IF (OPERAT(5:5),NE,'L')GO TO 70
        BIN1(7)='0'
        BIN1(8)='1'
        GO TO 100
70      BIN1(7)='1'
        BIN1(8)='0'
100     IF (OPERAT(3:3),EQ,'L')BIN1(9)='1'
        IF (OPERAT(3:3),EQ,'R')BIN1(9)='0'
        BIN1(13)='0'
        BIN1(14)='1'
        BIN1(15)='1'
        BIN1(16)='1'
        IF (OPERAT(4:4),EQ,'M')GO TO 200
        BIN1(6)='1'
        IF (ADRES1(1:1),NE,'D')GO TO 800
        IF (ADRES2(1:1),NE,'D')GO TO 800
        CALL TADR(ADRES1,MODE,REG,NUM,TYPE,FLG)
        DO 120 J=1,3
            BIN1(J+9)=REG(J)
120     CONTINUE
        CALL TADR(ADRES2,MODE,REG,NUM,TYPE,FLG)
        DO 130 J=1,3
            BIN1(J)=REG(J)
130     CONTINUE
        BIN1(4)='0'
        BIN1(5)='0'
        IF (OPERAT(1:1),NE,'R')GO TO 135
        BIN1(4)='1'
        BIN1(5)='1'
135     CONTINUE
        NWORDS=1
        PCOUNT=PCOUNT+2
        IF (OPERAT(1:1),EQ,'L')BIN1(4)='1'
        GO TO 700
200     DO 210 J=1,3
        BIN1(J+9)='0'
210     CONTINUE
        BIN1(7)='1'
        BIN1(8)='1'
        IF (OPERAT(1:1),EQ,'L')BIN1(10)='1'
        IF (OPERAT(1:1),NE,'R')GO TO 215
        BIN1(10)='0'
        BIN1(11)='0'
215     CONTINUE
        CALL TADR(ADRES1,MODE,REG,NUM,TYPE,FLG)
        DO 220 J=1,3
            BIN1(J)=REG(J)
            BIN1(J+3)=MODE(J)
220     CONTINUE
        NWORDS=1
        PCOUNT=PCOUNT+2
        IF (FLG,EQ,0)GO TO 700
        CALL NUMBIN(NUM,BIN32,BIN2,NZ)
        NWORDS=2
        PCOUNT=PCOUNT+2
700     RETURN

```

```

800 WRITE(5,850)OPERAT
850 FORMAT(1X,'IMPROPER ADDRESSING MODE FOR : ',A4)
STOP
END

```

```

C
C      CONDITIONAL BRANCH/ UNCONDITIONAL BRANCH
SUBROUTINE BCC(LABEL,OPERAT,ADRES1,PCOUNT,NWORDS,HEXM,
$BIN1,BIN2)
COMMON/BLOCK1/NPASS
CHARACTER*1 HEXM(4),BIN1(16),BIN2(16),BIN32(32),TYPE
CHARACTER*1 REG(3),MODE(3)
CHARACTER*6 LABEL,OPERAT
CHARACTER*9 ADRES1,DUMMY
INTEGER*4 PCOUNT,NUM,RESULT,NUM2,ABSRES
INTEGER NWORDS,NPASS,FLG
DUMMY='S0000'

```

```

C
IF(NPASS,NE,1)GO TO 20
CALL LABTAB(LABEL,PCOUNT,NK)
IF(ADRES1(1:1).EQ,'(')ADRES1='S0000'
GO TO 30
20 CALL LABAD(ADRES1,DUMMY)
30 CALL DIGHEX(PCOUNT,HEXM)
DO 50 J=1,13
BIN1(J)='0'
50 CONTINUE
BIN1(14)='1'
BIN1(15)='1'
BIN1(16)='0'
IF(OPERAT(2:3).NE,'RA')GO TO 60
GO TO 88
60 IF(OPERAT(2:3).NE,'HI')GO TO 62
BIN1(10)='1'
GO TO 88
62 IF(OPERAT(2:3).NE,'LS')GO TO 64
BIN1(10)='1'
BIN1(9)='1'
GO TO 88
64 IF(OPERAT(2:3).NE,'SR')GO TO 66
BIN1(9)='1'
GO TO 88
66 IF(OPERAT(2:3).NE,'CC')GO TO 68
BIN1(11)='1'
GO TO 88
68 IF(OPERAT(2:3).NE,'CS')GO TO 70
BIN1(11)='1'
BIN1(9)='1'
GO TO 88
70 IF(OPERAT(2:3).NE,'NE')GO TO 72
BIN1(10)='1'
BIN1(11)='1'
GO TO 88
72 IF(OPERAT(2:3).NE,'VC')GO TO 74
BIN1(12)='1'
GO TO 88
74 IF(OPERAT(2:3).NE,'VS')GO TO 76
BIN1(9)='1'
BIN1(12)='1'
GO TO 88
76 DO 77 J=1,4
BIN1(J+8)='1'
77 CONTINUE
79 IF(OPERAT(2:3).NE,'EQ')GO TO 80
BIN1(12)='0'
GO TO 88
80 IF(OPERAT(2:3).NE,'PL')GO TO 81      A28

```

```

      BIN1(7)='0'
      BIN1(11)='0'
      GO TO 88
81    IF(OPERAT(2:3).NE.'MI')GO TO 82
      BIN1(11)='0'
      GO TO 88
82    IF(OPERAT(2:3).NE.'GE')GO TO 83
      BIN1(9)='0'
      BIN1(10)='0'
      GO TO 88
83    IF(OPERAT(2:3).NE.'LT')GO TO 84
      BIN1(10)='0'
      GO TO 88
84    IF(OPERAT(2:3).NE.'GT')GO TO 85
      BIN1(9)='0'
      GO TO 88
85    IF(OPERAT(2:3).EQ.'LE')GO TO 88
      WRITE(5,86)OPERAT
86    FORMAT(1X,'IMPROPER BRANCH CONDITION :',A6)
      STOP
C
88    IF(ADRES1(1:1).EQ.'S')GO TO 100
      IF(ADRES1(1:1).EQ.'P')GO TO 200
      WRITE(5,90)OPERAT
90    FORMAT(1X,'INVALID ADDRESS FOR :',A6)
      STOP
C
100   CALL TADR(ADRES1,MODE,REG,NUM,TYPE,FLG)
      RESULT=NUM-PCOUNT-2
      ABSRES=JABS(RESULT)
      CALL NUMBIN(ABSRES,BIN32,BIN2,NF)
      IF(RESULT.GE.0)GO TO 150
      CALL TCOMP(BIN2,BIN32,NF)
150   PCOUNT=PCOUNT+4
      NWORDS=2
      GO TO 300
200   CALL TADR(ADRES1,MODE,REG,NUM,TYPE,FLG)
      CALL NUMBIN(NUM,BIN32,BIN2,NF)
      PCOUNT=PCOUNT+4
      NWORDS=2
300   RETURN
      END
C
C
C    MOVEQ INSTRUCTION
      SUBROUTINE QMOVE(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,
      SNWORDS,HEXM,BIN1)
      COMMON/BLOCK1/NPASS
      CHARACTER*6 LABEL,OPERAT
      CHARACTER*9 ADRES1,ADRES2
      CHARACTER*1 BIN1(16),BIN32(32),HEXM(4),TYPE,BIN2(16)
      CHARACTER*1 REG1(3),REG2(3),MODE1(3),MODE2(3)
      INTEGER NWORDS,FLG
      INTEGER*4 PCOUNT,NUM1,NUM2
C
      IF(NPASS.NE.1)GO TO 100
      CALL LABTAB(LABEL,PCOUNT,K)
100   CALL DIGHEX(PCOUNT,HEXM)
      IF(ADRES1(1:1).NE.'#')GO TO 200
      IF(ADRES2(1:1).NE.'D')GO TO 200
C
      CALL TADR(ADRES1,MODE1,REG1,NUM1,TYPE,FLG)
      CALL TADR(ADRES2,MODE2,REG2,NUM2,TYPE,FLG)
      BIN1(16)='0'
      BIN1(15)='1'

```

```

BIN1(14)='1'
BIN1(13)='1'
BIN1(9)='0'
DO 110 J=1,3
    BIN1(J+9)=REG2(J)
110 CONTINUE
CALL NUMBIN(NUM1,BIN32,BIN2,NF)
DO 120 J=1,8
    BIN1(J)=BIN2(J)
120 CONTINUE
PCOUNT=PCOUNT+2
NWORDS=1
RETURN
200 WRITE(5,210)
210 FORMAT(1X,'IMPROPER ADDRESS FOR MOVED COMMAND')
STOP
END

C
C
C ADDQ
SUBROUTINE GADD(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,
SNWORDS,HEXM,BIN1,BIN2)
COMMON/BLOCK1/NPASS
CHARACTER*6 LABEL,OPERAT
CHARACTER*9 ADRES1,ADRES2
CHARACTER*1 BIN1(16),BIN2(16),BIN32(32),HEXM(4)
CHARACTER*1 TYPE,DAT(3),REG(3),MODE(3)
INTEGER FLG
INTEGER*4 PCOUNT,NUM

C
DO 10 J=1,16
    BIN1(J)='0'
10 CONTINUE
C
IF(NPASS,NE,1)GO TO 20
CALL LABTAB(LABEL,PCOUNT,NK)
IF(ADRES2(1:1).EQ,'(')ADRES2='S0000'
GO TO 30
20 CALL LABAD(ADRES1,ADRES2)
30 CALL DIGHEX(PCOUNT,HEXM)
C
IF(ADRES1(1:1).NE,'#')GO TO 90
DO 35 J=2,9
    M=J
    IF(ADRES1(J:J).NE,'0')GO TO 45
35 CONTINUE
45 IF(ADRES1(M:M).NE,'8')GO TO 50
    DO 48 J=1,3
        DAT(J)='0'
48 CONTINUE
GO TO 55
50 CALL ADRLOC(ADRES1(M:M),DAT)
55 CALL TADR(ADRES2,MODE,REG,NUM,TYPE,FLG)
C
DO 60 J=1,3
    BIN1(J)=REG(J)
    BIN1(J+3)=MODE(J)
    BIN1(J+9)=DAT(J)
60 CONTINUE
BIN1(13)='1'
BIN1(15)='1'
C
IF(OPERAT(5:5).NE,'B')GO TO 70
GO TO 95
70 IF(OPERAT(5:5).NE,'L')GO TO 80
    BIN1(8)='1'

```

```

      GO TO 95
80  IF(OPERAT(5:5),NE,'W')GO TO 90
      BIN1(7)='1'
      GO TO 95
90  WRITE(5,100)OPERAT
100  FORMAT(1X,'IMPROPER SIZE SPEC OR ADDRESSING MODE FOR :',A6)
      STOP
95  NWORDS=1
      PCOUNT=PCOUNT+2
      IF(FLG,NE,1)GO TO 150
          CALL NUMBIN(NUM,BIN32,BIN2,NZ)
          NWORDS=2
          PCOUNT=PCOUNT+2
150  RETURN
      END

```

C
C
C

```

      IMMEDIATE ADD,AND,ORR,EOR
      SUBROUTINE IMME(LABEL,OPERAT,ADRES1,ADRES2,PCOUNT,
SNWORDS,HEXM,BIN1,BIN2,BIN3)
      COMMON/BLOCK1/NPASS
      CHARACTER*6 LABEL,OPFRAT
      CHARACTER*9 ADRES1,ADRES2
      CHARACTER*1 BIN1(16),BIN2(16),BIN3(16),BIN32(32)
      CHARACTER*1 HEXM(4),REG(3),MODE(3),TYPE
      INTEGER FLG,NWORDS,NPASS
      INTEGER*4 PCOUNT,NUM1,NUM2
      IF(NPASS,NE,1)GO TO 20
          CALL LABTAB(LABEL,PCOUNT,NK)
          IF(ADRES2(1:1),EQ,'(')ADRES2='$0000'
          GO TO 30
20  CALL LABAD(ADRES1,ADRES2)
30  CALL DIGHEX(PCOUNT,HEXM)
      IF(ADRES1(1:1),NE,'#')GO TO 200
      DO 40 J=1,16
          BIN1(J)='0'
40  CONTINUE

```

C

```

      IF(OPERAT(1:3),NE,'ORR')GO TO 45
          GO TO 100
45  IF(OPERAT(1:3),NE,'EOR')GO TO 50
          BIN1(12)='1'
          BIN1(10)='1'
          GO TO 100
50  IF(OPERAT(1:3),NE,'CMP')GO TO 60
          BIN1(11)='1'
          BIN1(12)='1'
          GO TO 100
60  IF(OPERAT(1:3),NE,'AND')GO TO 70
          BIN1(10)='1'
          GO TO 100
70  IF(OPERAT(1:3),NE,'ADD')GO TO 75
          BIN1(10)='1'
          BIN1(11)='1'
          GO TO 100
75  IF(OPERAT(1:3),NE,'SUB')GO TO 80
          BIN1(11)='1'
          GO TO 100
80  WRITE(5,90)OPERAT
90  FORMAT(1X,'UNRECOGNIZED COMMAND :',A6)
      STOP
100 NWORDS=2
      PCOUNT=PCOUNT+4
      IF(OPERAT(5:5),NE,'B')GO TO 110
          GO TO 135
110 IF(OPERAT(5:5),NE,'W')GO TO 200  A31

```

```

      BIN1(7)='1'
35  CALL TADR(ADRES1,MODE,REG,NUM1,TYPE,FLG)
      CALL TADR(ADRES2,MODE,REG,NUM2,TYPE,FLG)
      CALL NUMBIN(NUM1,BIN32,BIN2,NZ)
      DO 140 J=1,3
          BIN1(J)=REG(J)
          BIN1(J+3)=MODE(J)
40  CONTINUE
      IF(FLG.EQ.0)GO TO 160
          CALL NUMBIN(NUM2,BIN32,BIN3,NZ)
          NWORDS=3
          PCOUNT=PCOUNT+2
60  RETURN
00  WRITE(5,210)OPERAT
10  FORMAT(' IMPROPER SIZE SPEC OR ADDRESSING MODE FOR :',A6)
      STOP
      END

```

APPENDIX B

Bibliography

Kane, Hawkins, Leventhal, 68000 Assembly Language Programming
McGraw-Hill, Berkely, California, 1981.

Motorola Inc., MC68000 Design Module Users Guide
(MEX68KDM), 1980.

APPENDIX C

the MC68000 Design Module and its associated connectors.

Connection to DEC 11/45:

P3 PIN		DB25 PIN
1	↔	1
3	↔	3
5	↔	2
13	↔	7
9 ↔ 14		
		4 ↔ 20

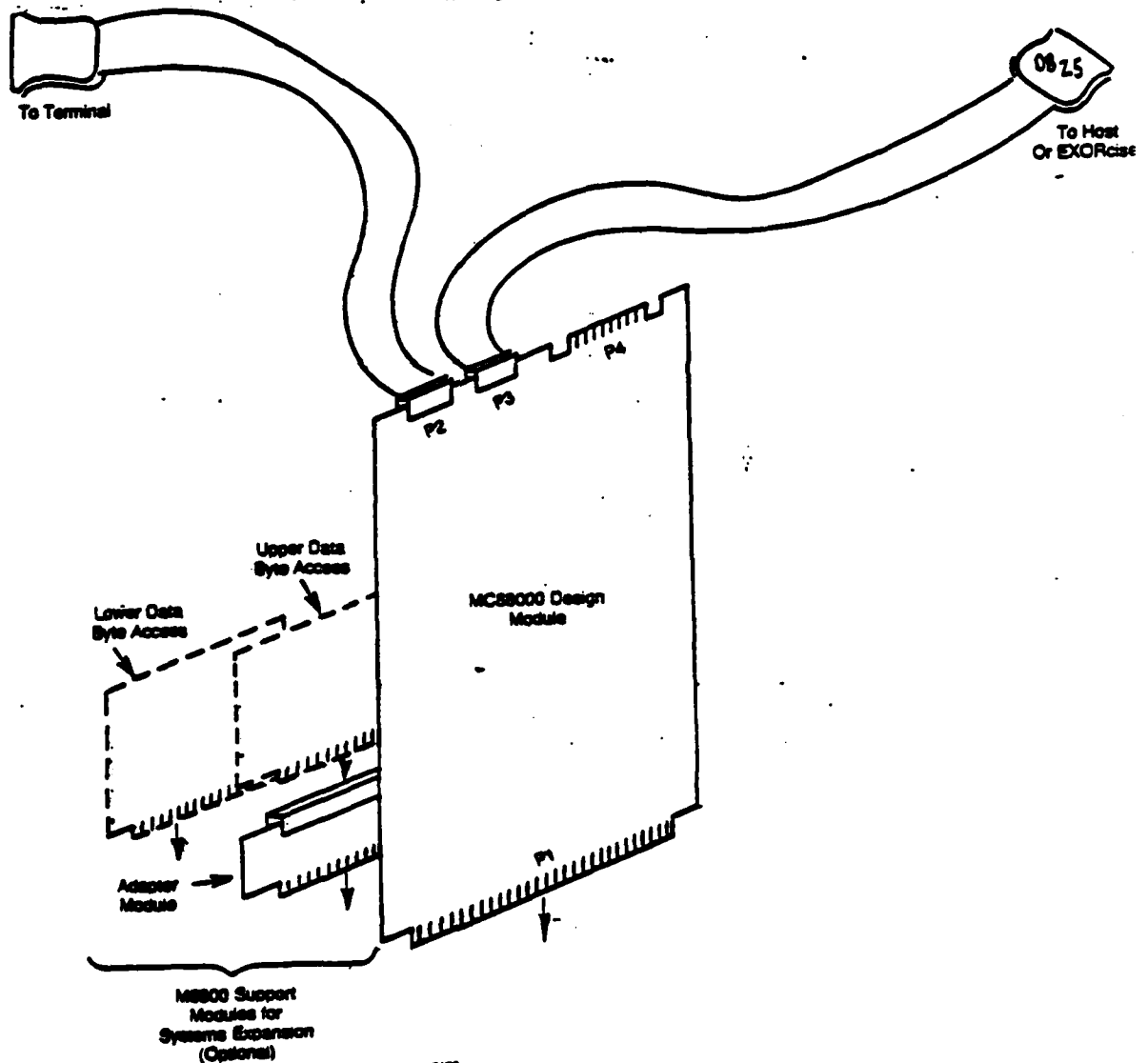


Diagram from: MC68000 Design Module
 USERS GUIDE (MEX6844M(92)), pg. 2-6

MC68000 Design Module

APPENDIX D

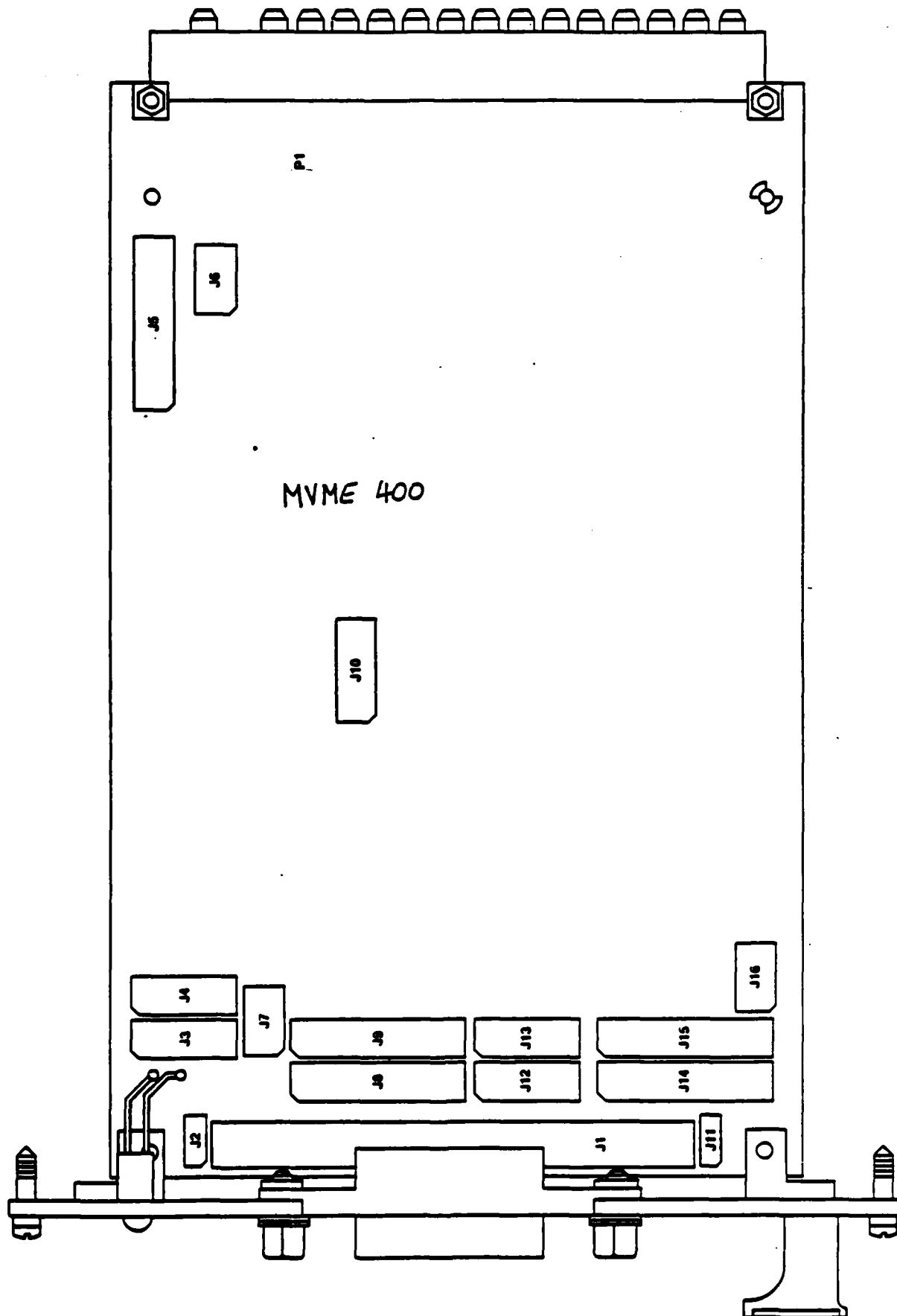
DSP RS-232C Connector Pin Assignments*

FRONT PANEL PORTS 1 & 2 PIN NUMBER	MODULE J1 PIN NUMBER	SIGNAL MNEMONIC	SIGNAL NAME AND DESCRIPTION
2	3,28	TxD	TRANSMITTED DATA - Serial binary data output.
3	5,30	RxD	RECEIVED DATA - Serial binary data input.
4	7,32	RTS	REQUEST TO SEND - A signal denoting terminal has data to send.
5	9,34	CTS	CLEAR TO SEND - A signal that indicates the terminal can transmit data.
6	11,36	DSR	DATA SET READY - A signal denoting the modem is ready (off the hook).
7	13,38	SIG GND	SIGNAL GROUND
8	15,40	DCD	DATA CARRIER DETECT - A signal that indicates to the terminal that a carrier is present.
15	4,29	TxC	TRANSMITTER CLOCK - (DCE Source) A signal that provides timing information for transmitted data.
17	8,33	RxC	RECEIVER CLOCK - A signal that provides timing information for received data.
20	14,39	DTR	DATA TERMINAL READY - A signal that denotes the terminal is ready to transmit or receive data.
22	18,43	RI	RING INDICATOR - A signal to DTE that denotes the modem is receiving a ringing signal.
24	22,47	TxC	TRANSMITTER CLOCK - (DTE Source) A signal that provides timing information for transmitted data.

* Chart from: MVME400 Dual RS-232C Serial Port Module
User's Manual (MVME400/02), p. 5-4

MVME400 INTERNAL JUMPERS *

HEADER	FUNCTION	JUMPER CONFIGURATION
J2	Port 2 TxC select	1-2
J3	Port 2 external clock select	No jumpers
J4	Port 2 internal clock select	1-2, 3-4, 9-10, 11-12
J5	Interrupt level select	3-5, 9-11, 15-17
J6	Base address select	7-8
J7	Port 2 CTS flow control	5-7, 6-8
J8	Port 2 to modem select	13-14
J9	Port 2 to terminal select	1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14, 15-16, 17-18, 19-20
J10	Baud rate port 1 and 2 select	3-4, 5-6, 9-10, 11-12
J11	Port 1 TxC select	1-2
J12	Port 1 external clock select	No jumpers
J13	Port 1 internal clock select	1-2, 3-4, 9-10, 11-12
J14	Port 1 to modem select	13-14
J15	Port 1 to terminal select	1-2, 3-4, 5-6, 7-8, 9-10, 11-12, 13-14, 15-16, 17-18, 19-20
J16	Port 1 CTS flow control	5-7, 6-8



DSP Module Header Location Diagram

MVME400/02, p 5-4

Compiling the Cross-assembler on the Host System:

The cross-assembler and its subroutines

SUBS2.FTN

SUBS3.FTN

DCSUB.FTN

MC68CRX.FTN

These must be compiled in FORTRAN 77 prior to taskbuilding.

Taskbuilding:

After compiling, the subroutines and main program must be taskbuilt or linked. On the DEC PDP11/45 with the RSX-11 operating system, the following taskbuilding session may be used:

TKB

MC68CRX/CP/FP=MC68CRX,DCSUB,SUBDIR,UTLSUB,SUBS1,SUBS2,
SUBS3,OPTSUB2

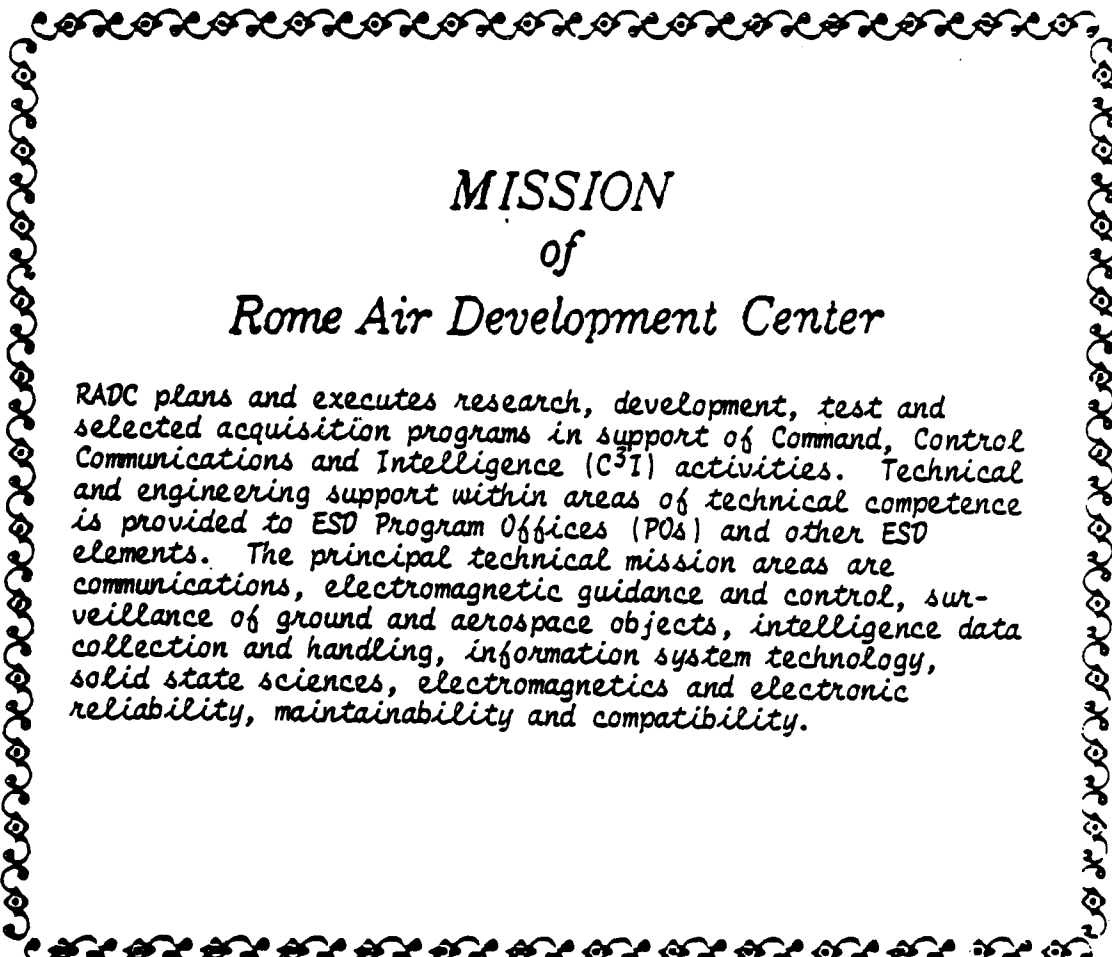
/

UNITS=12

ACTFIL=6

ASG=SY0:2:3:4:11,TIO:5

//



MISSION of Rome Air Development Center

RADC plans and executes research, development, test and selected acquisition programs in support of Command, Control Communications and Intelligence (C³I) activities. Technical and engineering support within areas of technical competence is provided to ESD Program Offices (POs) and other ESD elements. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, solid state sciences, electromagnetics and electronic reliability, maintainability and compatibility.

