RECEIVED BY DTIE MAR 1 1 1970

MASTER

IS-2229

Physics (UC-34)
TID-4500

UNITED STATES ATOMIC ENERGY COMMISSION

Research and Development Report

KUBER: The Control Program for the
USAEC Ames Laboratory
On-Line Measuring System

by

A. H. Klein and W. J. Higby

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

December 1969

Ames Laboratory
at
Iowa State University of Science and Technology
Robert S. Hansen, Director
Contract W-7405 eng-82

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

IS-2229

This report is distributed according to the category Physics (UC-34) as listed in TID-4500.

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

- A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or
- B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

Printed in the United States of America
Available from
Clearinghouse for Federal Scientific and Technical Information
National Bureau of Standards, U. S. Department of Commerce
Springfield, Virginia 22151
Price: Printed Copy \$3.00; Microfiche \$0.65

TABLE OF CONTENTS

	Page
ABSTRACT	v
INTRODUCTION	1
THE PROGRAM ENVIRONMENT_____	2
PROGRAM OPERATION	4
Introduction	4
Initialization	4
The Job Selector	5
The Task Processor	5
The Operation Selector	6
The Interrupt Processor	6
Diagnostic Interrupt Processor	7
DETAILED DESCRIPTION OF THE TASK WORDS	8
ACKNOWLEDGMENTS	10
APPENDIX I	21
APPENDIX II	40

IS-2229

IS-2229

KUBER: The Control Program
for the USAEC Ames Laboratory On-Line Measuring System

A. H. Klein and W. J. Higby

ABSTRACT

A program called KUBER has been written to provide control for an on-line measuring machine system. The program interprets a table of task words for each measuring machine and performs the tasks specified. It supervises the input of data from the measuring machines and calls routines to process the data.

IS-2229

INTRODUCTION

The elements of the on-line measuring machine control and data processing system consist of the interpretive routine, KUBER,^{*} and a list of operations in a task word table, TAWTA. The TAWTA may be coded by the programmer directly in the form which can be interpreted by KUBER or it may be written in the language of the KUBER Expression Reducer and TRANslator, KERTRAN.¹ The KERTRAN compiler then produces an ASIST language listing which is assembled into a TAWTA which can be interpreted by KUBER.²

KUBER could operate with any number of measuring machines but in its present form it controls four on-line measuring machines.³ There is a "job" associated with each measuring machine. Each job consists of a series of "tasks" specified in a task word table or TAWTA. A task may involve some measuring machine operation or the processing of data. For each job, KUBER starts at the beginning of the TAWTA, reads a task word, and performs the task specified by it. KUBER then processes the next word. Eventually it will reach a point where it must pause and wait for some action on the part of the measuring machine operator before it can continue through the TAWTA. This particular job is then in a deactive state.

*KUBER: from the stem of Greek κυβερνᾶν, to steer or control, whence κυβερνήτης, helmsman, and the derivative adjective κυβερνήτικος, skilled in steering, whence Latin cyberneticus, whence the modern cybernetics; cognate to Latin gubernāre, to steer or pilot, whence French gouverner, whence English govern; also compare Sanskrit kubhanyús, dancing (steering one's steps).

Logically, there are four jobs to be performed. There must be a mechanism to switch from one to another. This switching is done by the "Job Selector." When a task for one job has been performed, KUBER goes to the Job Selector where another job is selected and its next task is performed. The Job Selector cycles through all the active jobs. Thus, the tasks of the jobs are interleaved in a temporal sequence although each job is logically independent and its tasks follow each other in strict logical order.

It is not unusual for all four jobs to reach a situation where each requires some operator action before it can continue. When this happens, the Job Selector will find no active job to be performed. It will then branch to an endless loop called "Lazyclock," to wait until some operator action interrupts and starts the next sequence of tasks.

Interrupts from the measuring machine are accepted by KUBER only when they are useful to the job. An acknowledge interrupt will be accepted only after the program has sent a message to the measurer and an enter interrupt will be accepted only when the job has reached a situation where it is ready to use input data. Thus, the measuring machine operator can hit an interrupt button at the wrong time or hit the wrong button without confusing the job. The incorrect interrupt will simply be ignored by the program.

THE PROGRAM ENVIRONMENT

The hardware consists of an EMR(ASI)6050 computer with 16K of memory. Peripheral equipment includes a five-megabit drum, a nine-track

tape, and a card reader on Channel 0. The four measuring machines are on Channel 1. There is also a teletypewriter on a programmed input/output channel.

There is a hardware trap which allows or disallows interrupts from the external devices and is under program control. Each external device has reserved for its use two memory locations. Whenever a hardware interrupt is received from an external device, the computer executes the instruction in the appropriate interrupt location in memory. This instruction is usually an RTJ^{*} to an interrupt routine. The execution of the RTJ instruction puts the computer in an "interrupted state." No other interrupts of equal or lower priority are permitted in this state. All external devices are of equal priority. Thus, no external device can interrupt the interrupt routine of any other. The interrupt routine is usually completed with a JDI[†] instruction. This instruction decrements the interrupt level at which interrupts are locked out. Thus, when the interrupt routine of one external device is completed, the JDI instruction at its end permits interrupts from other external devices to be recognized.

Each of the four measuring machines has two interrupts associated with it. The enter interrupt is used to inform the computer that data are ready to be transferred from the measuring machine to the computer. The acknowledge interrupt is used to respond to messages which the computer causes to be displayed on the measuring machine for the measurer's benefit.

^{*}Set return address and jump.

[†]Jump and decrement interrupt level.

Each measuring machine has two addresses. The computer uses one to read in data from the measuring machine. The other is used to send instructions to the measuring machine to display a binaview number or a message or to light a view.

The KUBER program is written in ASIST, Advanced Scientific Instruments Symbolic Translator (See Appendix II). The ASIST language consists of symbolic commands which are directly related to machine codes. It permits mnemonic labels and addresses. Specification of indexing and indirect addressing is possible. Comments are permitted and are extensively used in the KUBER program (See Appendix I).

PROGRAM OPERATION

INTRODUCTION

Figure 3 is a flowchart of the KUBER program. The flowchart is done at the conceptual level but some of the terms are drawn from those used in the actual coding. Table IV comprises the most important of these terms. In the following paragraphs, the words in brackets are labels of instructions in the actual KUBER program and are given to help correlate the description with the relevant part of the code.

INITIALIZATION

KUBER begins by setting up the interrupt locations for the measuring machines [KUBER]. These will transfer control to the Interrupt Processor when the computer receives a hardware interrupt from a measuring machine. The coding is set up to keep track of where the interrupt came from and what software operation was in progress when it occurred. An option is

set up which permits the restart of the system without reloading. Pointers and control indicators are initialized. KUBER makes a patch in the system monitor which permits using the operator interrupt button on the console to get diagnostic information out of the system or to tape-mark an output tape before removing it. Then it sets the beginning values for the task word pointers [INVAR]. Next it goes to the job selector [JOBSEL].

THE JOB SELECTOR

There is a job associated with each measuring machine. The Job Selector steps through the jobs in cyclic fashion [CYCLER]. If it finds an activated job, it goes to the Task Processor [TASPRO]; the job number is left set so that the next time through the Job Selector will first look at the next job in the cycle. If the Job Selector finds no active job as it goes around the cycle, it goes to check [LNKLOK] if a link has been interrupted and left unfinished. When a job calls a link, the job is deactivated; therefore, the Job Selector will not select a job which is in the middle of an unfinished link. If an unfinished link is found, KUBER restores the necessary link information [LRSTR] and returns to the link operation [LRETRN]. If no link is found unfinished, KUBER goes to the Lazyclock [LZYCLK] which merely counts loops until it is interrupted.

THE TASK PROCESSOR

The Task Processor immediately sets the interrupt controls to ignore all interrupts for this job. The current task word is examined [TSTEP]. If the task involves transferring a word to or from memory, the memory location involved is calculated [LOSTAR]; all indexing is done at this

point so that a simple location is produced for use by the operation. If no memory location is involved, control goes directly to the Operation Selector [OPSEL].

THE OPERATION SELECTOR

The Operation Selector looks at the operation code section of the task word and branches to the appropriate one of the thirty-one possible operations. When the operation is completed, KUBER either goes on to the next task for this job or returns to the Job Selector as specified by the end-of-task control switch [ENDTSK].

THE INTERRUPT PROCESSOR

As soon as the Interrupt Processor is entered [INTER], the external device trap is cleared to prevent any other hardware interrupts from occurring while the current one is being processed. Next the drum is checked; if it is busy, it is assumed that a drum read or write operation is in progress and control is returned immediately to that operation and the interrupting job is ignored. (This precaution is necessary because drum operations end with an interrupt from the drum. If the measuring machines are allowed to intervene while the drum operation is in progress, the drum interrupt may be lost. Then, when control returns to the interrupted link it will wait for the interrupt which signals the completion of the drum operation, but it will never arrive.)

If no drum operation is in progress, the reaction to the interrupt will depend on the interrupt control switch [ICNTRL] for the relevant machine and the level of software operation that was interrupted. If the

interrupt control switch is set to "ignore," KUBER returns to the interrupted operation; if the switch is set to "task," the next action depends on what kind of operation was interrupted.

From the Interrupt Processor's point of view, there is a three-level hierarchy of operations which may have been interrupted. A KUBER operation or a subroutine is of high priority and must be completed before any other job can be processed. Therefore, the Interrupt Processor merely activates the interrupting job and returns immediately to the interrupted operation. The newly active job is now eligible for selection by the Job Selector when it next looks for an active job.

A link operation is of medium priority. If interrupted it may have to wait while a KUBER operation or subroutine is processed. However, if an attempt is made to call another link while the interrupted link remains incompletd, the second link must wait until the first is completed.

When KUBER has nothing else to do, it branches to the Lazyclock which has the lowest level of priority. If an interrupt occurs while KUBER is in Lazyclock, the interrupting job is immediately processed.

DIAGNOSTIC INTERRUPT PROCESSOR

Scattered through KUBER there are sections of code which form a Diagnostic Information Table or DIT, which contains a record of the measuring machine and logical operations. The DIT is constantly updated so that it always contains the last fifty operations performed. When debug information is desired by the programmer, the Operator Interrupt button on the EMR 6050 console can be pushed to call a link which will print out the DIT in an easily legible form.

DETAILED DESCRIPTION OF THE TASK WORDS

Figure 1 shows part of a sample KERTRAN program. The ASIST language listing which would be produced from this program by the KERTRAN compiler is shown in the middle part of the figure. The right part of the figure shows the machine code which would be produced by the ASIST Assembler. It is this code which is actually interpreted by KUBER.

In the task word table each word specifies an operation and identifies the data word which is to be acted upon if there is one. The TAWTA operations are listed and briefly defined in Table V.

For detailed analysis each task word must be considered as a 24-bit word (See Table I). The operations are specified in bits 24-19 for the 24-bit word. Bit 18 is the end-of-task control. If it is set to one, KUBER proceeds directly to the next task in the TAWTA after completing this operation; if it is set to zero, KUBER goes to the Job Selector. For arithmetic operations, bits 17-16 specify the class of the root variable: 0, 1, 2, or 3.

For a class 0 variable, bits 15-1 specify the relative position within the KUBER table, KUTAB, where there is a parameter which contains additional information about the variable (See Table II and Figure 2). In KUTAB, bits 15-1 contain the base address of the variable. Bits 17-16 specify the number of subscripts which the variable has: 0, 1, 2, or 3. Bit 18 is one if the last subscript of the variable is the measuring machine number. If the variable has no subscripts or one subscript, bits 24-19 are unused. If the variable is doubly subscripted, bits 24-19 contain the range of the first subscript. If the variable is triply

subscripted, bits 24-19 contain a pointer to a position within another table, ANTAB, which contains information on the range of the first two subscripts (See Table III and Figure 2). The range of the first subscript is specified in bits 24-19 of the ANTAB word and the range of second subscript is specified in bits 15-1.

If the root variable is of class 1, it is from the IICOM section of common. In fact, from KUBER's point of view the variable is IICOM which is doubly subscripted with the measuring machine number as the second subscript and 100 as the range of the first subscript. Bits 15-1 of the task word contain the value of the first subscript. If the task word is followed by a subscript parameter (distinguished by bits 24-19 being all zero), the first subscript of the root variable is itself subscripted. This subscript is of class 1, 2, or 3 and the information in the subscript parameter is treated accordingly.

If the root variable is of class 2, it is from the IDEVNT section of common. Everything else about it is the same as for class 1.

If the root variable is of class 3, bits 15-1 contain the address of the root variable. Class 3 variables may be singly subscripted. The subscript may be of class 1, 2, or 3.

For logical transfer operations and the INVOKE operation, bits 15-1 specify the location within the TAWTA to which control is to be transferred, i.e., the label of the next task word to be interpreted.

The CALL operation uses bits 15-1 for the location of the resident subroutine which is being called.

The CALL LINK has the number of the link in bits 15-1.

The button board operations, x,y coordinate operations, return operation, and measuring machine operations do not use bits 15-1.

For the "set interrupt controls" operation, bits 15-1 contain the control information for the enter and acknowledge interrupts as $10 \times IENT + IACK$ where a zero value for IENT or IACK means ignore the relevant interrupt and a one value means accept the interrupt.

ACKNOWLEDGMENTS

We wish to express our appreciation to R. A. Jespersen and T. L. Schalk for their cooperation and patience while testing this program and to W. J. Kernan for his general guidance and encouragement.

REFERENCES

1. W. J. Higby and A. H. Klein, "Ames Laboratory On-Line System for Bubble Chamber Film Measuring," IS-2228 (December 1969). (This report discusses the rules of the KERTRAN language in detail. It also describes the system in general.)
2. C. L. Cate, R. L. Wagstaff, and W. J. Higby, "The KERTRAN Compiling System," IS-2230 (December 1969).
3. T. L. Schalk and W. J. Kernan, "Physics Use of the Ames Laboratory On-Line Measuring System for Bubble Chamber Events," IS-2231 (December 1969).

TABLE 1

TASK WORD FORMAT for operations 1-16

24		19	18	17	16	15		1
	OP	I	X				Y	

OP: operation to be performed*

I: end of task control; =1:go to next task of this job
 =0:go to job selector for any task

X: class of variable being acted on by the operation; 0,1,2, or 3

Y: for X=0: the relative position in KUTAB of further information
 on the variable

for X=1: the relative position in IICOM of the variable

for X=2: the relative position in IDEVNT of the variable

for X=3: the actual address of the variable

*If $\theta P=00$, the task word is a subscript parameter associated with a preceding operation. I is not used and X and Y specify the characteristics of the subscript. A class 0 root variable may have as many as three subscripts. Class 1, 2, or 3 root variables may be followed by one subscript parameter.

TABLE II

KUTAB FORMAT

OP	I	X	Y
----	---	---	---

Y: base address of the variable or array
 X: number of subscripts for variable; 0, 1, 2, or 3
 I: indicates that last index is machine number if I=1
 OP: for X=0 or 1: unused
 for X=2: range of first subscript
 for X=3: pointer to the position in ANTAB
 where subscript range information
 is found

TABLE III

ANTAB FORMAT

OP	-	-	Y
----	---	---	---

OP: range of first subscript
 Y: range of second subscript

TABLE IV

DEFINITION OF TERMS

IJOB	Job number or measuring machine number =1,2,3, or 4.
IThis(IJOB)	The location of task word being interpreted.
IDACT(IJOB)	The deactivation indicator used by job selector and binaview and interrupt control operations. It is zero if job is active, one if job is deactivated.
IENT(IJOB)	Specifies that an enter interrupt should be ignored if IENT is zero, acted upon if IENT is one.
IACK(IJOB)	Same as IENT but for acknowledge interrupts.
LNKJOB	The calling job number when a link is in progress; otherwise it is equal to zero.
KJOB	A pointer used by the job selector as it searches for an active job. If one is found, IJOB is set to KJOB and the next task is processed.
ILC	A lazyclock counter. It counts intervals of 9.525 μ sec.
ITIME	A lazyclock counter. It counts intervals of 79.9 sec.
WAIT	A switch set to one when KUBER is selecting a job or waiting for an interrupt. Otherwise it is zero.
LIND	A switch set to one when a link is in progress. It is zero when no link is in progress or a link has been interrupted and is waiting for other jobs to finish.
LINKNO	The number of the link being executed.
IRET	Location in TAWTA to which control will return after the series of tasks called by INVOKE has been completed.
ICNTRL	Switch which determines whether interrupt from a machine should be ignored or have its job activated.

TABLE V

TAWTA OPERATIONS SUBSET

<u>Type</u>	<u>Op Code</u>	<u>Mnemonic</u>	<u>Description</u>
1	01	LDA	$M \rightarrow A$
2	02	LDE	$M \rightarrow E$
3	03	LMA	$-M \rightarrow A$
4	04	STA	$A \rightarrow M$
5	05	STE	$E \rightarrow M$
6	06	LBA	COL A OF BUTTON BOARD $\rightarrow A$
7	07	LBB	COL B OF BUTTON BOARD $\rightarrow A$
8	10	LBC	COL C OF BUTTON BOARD $\rightarrow A$
9	11	ADD	$A + M \rightarrow A$
10	12	MPY	$A * M \rightarrow A$
11	13	SUB	$A - M \rightarrow A$
12	14	DVD	$A \div M \rightarrow A$
13	15	LDX	X FROM MEAS. MACH. $\rightarrow A$
14	16	LDY	Y FROM MEAS. MACH. $\rightarrow A$
15	21	JMP	TRANSFER CONTROL
16	22	JMF	JUMP IF MEM. FAIL
17	23	JBD	JUMP IF BAD DATEX CODE
18	24	JET	JUMP IF END OF TRACK
31	25	JRF	JUMP IF READ FAIL
19	30	JLZ	JUMP IF $A < 0$
20	31	JGZ	JUMP IF $A > 0$
21	32	JEZ	JUMP IF $A = 0$
22	33	SRT	SET RETURN, TRANSFER WITHIN TAWTA
23	34	RET	RETURN WITHIN TAWTA
24	35	CAL	CALL CORE RESIDENT SUBROUTINE
25	36	LNK	CALL LINK
26	60	RDP	READ DATA POINT
27	61	LTV	LIGHT VIEW
28	62	DMS	DISPLAY MESSAGE
29	63	LBV	LIGHT BINAVIEW
30	77	SIC	SET INTERRUPT CONTROL

EQUIVALENCE CLASS 0 IVERTX,7 IVERTY,8		
EQUIVALENCE CLASS 1 NTRACK,15 NVERT,14 NVIEW,12		
EQUIVALENCE CLASS 2 ISTOP,32		
:		
STRT	IF(ISTOP(NTRACK).EQ.0) 0UT,NVRT	
:		
NVRT	NVERT=2	
	NVIEW=1	
LITE	LIGHT VIEW NVIEW NEXT	
	PROCEED IGNORE,TASK	
	INDAT	
	IVERTX(NVIEW,NVERT)=MMX	
	IVERTY(NVIEW,NVERT)=MMY	
	NVIEW=NVIEW+1	
	IF(NVIEW.LT.4)LITE,C0NT	
CONT	...	
	:	
	:	
0UT	...	
STRT LDA ISTOP,2 24000 01200040		
PAR NTRACK,1 00100017		
SUB =0,3 13325000		
JEZ 0UT 32026000		
JMP NVRT 21024005		
NVRT	LDA =2,3 24005 01325002	
	STA NVERT,1 04100016	
	LDA =1,3 01325001	
	STA NVIEW,1 04100014	
LITE	LDA* NVIEW,1 24011 01500014	
	LTV* 61400000	
	SIC 10 77000012	
	RDP 60000000	
	LDX 15000000	
	STA IVERTX,0 04000007	
	PAR NVIEW,1 00100016	
	PAR NVERT,1 00100014	
	LDY 16000000	
	STA IVERTY,0 04000010	
	PAR NVIEW,1 00100016	
	PAR NVERT,1 00100014	
	LDA NVIEW,1 01100016	
	ADD =1,3 11325001	
	STA NVIEW,1 04100016	
	LDA NVIEW,1 01100016	
	SUB =4,3 13325004	
	JLZ LITE 30024011	
	JMP C0NT 21024034	
CONT	...	24034 ...
	:	:
	0CT 0 25000 00000000	
	0CT 1 25001 00000001	
	0CT 2 25002 00000002	
	0CT 3 25003 00000003	
	0CT 4 25004 00000004	
	:	:
	:	:
0UT	...	26000 ...

Figure 1. Part of a sample KERTRAN program and its ASIST language and machine language equivalents. The EQUIVALENCE statements are used by the KERTRAN compiler to help specify the variables being used. The middle column is the ASIST language listing which would be produced from the statements at the left. (In reality the octal equivalents of the mnemonics LTV, SIC, RDP, LDX, and LDY would be produced by the compiler. The mnemonics are used here to correspond to the terms used in the KUBER listing as given in Appendix 1.) On the right is the machine code (in octal notation) as it would be generated by the ASIST Assembler. It is arbitrarily shown as starting in computer memory at location 24000. The table of literals is assumed to be at 25000 and some other part of the TAWTA, labeled 0UT, is assumed to start at 26000.

KUTAB	PAR	**	00000000
	00	ISYS,1	00030624
	03*	MFID,2	03632726
	03*	MVRT,2	03632742
	03*	IP0INT,2	03632754
	01*	IFIDX,3	01732766
	02*	IFIDY,3	02733046
	03*	IVERTX,3	03733126
	04*	IVERTY,3	04733222
	05*	IC0RDX,3	05733316
	06*	IC0RDY,3	06733506
	03*	NPT,2	03636726
	00	IJOB	00027414
ANTAB	PAR	**	00000000
	03	4	03000004
	03	4	03000004
	03	5	03000005
	03	5	03000005
	03	10	03000012
	03	10	03000012

Figure 2. The KUTAB and ANTAB listings as they presently are used by KUBER. The ASIST language listing is on the left and the equivalent machine language (in octal notation) is on the right.

Figure 3. Flowcharts of the KUBER program. The initial entry to the program is through KUBER START. When the interrupts have been enabled, the hardware may interrupt the program by initiating action starting at INTERRUPT or OP INTERRUPT. The program operations enclosed by the dotted lines are protected from further interruption by the hardware. The process JOBSEL starts when WAIT is set to one and ends when WAIT is set to zero. LOSTAR, which appears on page 19, is a predefined process. It is defined in the flowchart on page 20. GETSUB and LIMBO are also defined on page 20.

Figure 3

Figure 3 (continued)

Figure 3 (continued)

APPENDIX I. KUBER PROGRAM LISTING

The KUBER program is listed here in ASIST language mnemonics. Columns 1-6 are for labels, columns 7-12 for operations, and columns 13-17 for addresses, indices, and comments. Comments are separated from address and index information by one or more blanks. An asterisk in column 1 also specifies a comment card. Extensive comments are used in this listing to aid in following the logic of the program. A list of ASIST mnemonics with brief definitions is given in Appendix II.

```

*
* KUBER-THE CONTROL PROGRAM FOR THE USAEC AMES LABORATORY
* ON-LINE MEASURING SYSTEM
*
*

```

```

ENTRY KUBER
CMN DIT(50),IRDBUF(100),LNKJOB(1),LINKNO(1),IJOB(1),ITIME(1)
CMN IDACT(10),IENDT(10),ITHIS(10),IACK(5),IENT(5),IEMPT1(6)
CMN ISECT(50),JSECT(50)
CMN IEMPT2(100),IICOM(400),ISYS(50),CONMA(240)
CMN IOPTIC(400),IDEVNT(400),MFID(12),MVRT(12),IPOINT(12)
CMN IFIDX(48),IFIDY(48),IVERTX(60),IVERTY(60)
CMN ICORDX(120),ICORDY(120)
CMN IMTR(96),KMTR(1464),NPT(12)
EXT LINKER

```

```

*
KUBER LDA INOP          INITIALIZE
      STA $30
      ADD =2            INTERRUPT
      STA TEMP
      LDA =1            LOCATIONS
      ADM KUBER+1
      LDA TEMP          30 TO 37 FOR
      KAL INEND
      JMP *+2            MACHINES 1 TO 4
      JMP KUBER+1
      LDA RESTR          PUT RESTART INSTRUCTION
      STA 0              IN LOCATION ZERO
      LDA CHJMP
      STA $1247          SET DRUM TEST FOR CHAINF
      JMP INVAR

```

```

*
TSTDRM NOP
      EXD DRMTST        DRUM TEST FOR CHAINF
      JMP *-2
      JMP $1252
DRMTST OCT 44100004
CHJMP JMP TSTDRM

```

```

*
INVAR ZOO              ZERO CONTROL VARIABLES
      LOX 44,1          IN COMMON
      STA LNKJOB-1,1
      JDX *-1,1
      DST KJOB
      STA LIND
      LDA =-1           INITIALIZE PUSHDOWN STACK COUNTERS
      ATE
      DST NRET
      DST NRET+2
      STA NRET+4
      LDA =4            SET JOBMAX FOR
      STA JOBMAX        FOUR MACHINES
      CTP EDT           CLEAR ED TRAP
      STP OPT

```

LDA	OPINT	PREPARE OPERATOR INTERRUPT BUTTON
STA	\$101	FOR LINK 13 PRINTOUT
LDA	OPINOP	
STA	\$620	
DLD	ISYS+29	SET TASK WORD TABLES
DST	ITHIS+1	TO
DLD	ISYS+31	BEGINNING ADDRESSES
DST	ITHIS+3	
JMP	SETRAP	
*		
IENT1	PAR **	CODING TO
RTJ	INTER	
IACK1	PAR **	KEEP TRACK
RTJ	INTER	
IENT2	PAR **	OF SOURCE
RTJ	INTER	
IACK2	PAR **	OF INTERRUPT
RTJ	INTER	
IENT3	PAR **	
RTJ	INTER	
IACK3	PAR **	
RTJ	INTER	
IENT4	PAR **	
RTJ	INTER	
IACK4	PAR **	
RTJ	INTER	
*		
INTER	PAR **	INTERRUPT FROM MEASURING MACHINE
CTP	EDT	CLEAR ED TRAP
EXD	CHZT	CHECK IF LINK ED OPERATION WAS INTERRUPTED
JMP	BSYXIT	IT WAS--RETURN IMMEDIATELY
DST	IREG	SAVE A AND E
SXM	IR1,1	SAVE INDEX ONE
SXM	IR2,2	TWO
SXM	IR3,3	THREE
LDA	IJOB	SAVE IJOB
STA	IJSV	
RTJ	INUIZN	UPDATE DIT
LDA	INTER	GET INTERRUPT ADDRESS
SOA	IENT1-2	SUBTRACT BASE ADDRESS
RLD	25	
DVD	=2	GET MACHINE NUMBER IN E
NOP		A=ZERO FOR ENT. A=ONE FOR ACK
STE	IJOB	SET IJOB TO NUMBER OF INTERRUPTING MACHINE
XAE		
LAX	,1	
ADD	=\$43	FORM DIT CODE FOR INTERRUPT
LLE	8	POSITION INTERRUPT IDENTIFIER
LLD	17	
STA*	DITI	
ATE		
RLE	1	
AND	=1	RETRIEVE INTERRUPT IDENTIFIER
JEZ	IE	GO TO PROCESS ENTER INTERRUPT
LDA	IACK,1	SET ICNTRL TO IACK

```

      JMP      ICNTRL
*
CHZT  44      $21000,1
*
BSYXITSTA    IREG
      LDA     INTER      PREPARE FOR RETURN
      SUB     =2         TO DRUM OPERATION
      SAM     **+1
      LDA     **
      SAM     BRETRN
      LDA     IREG
      STP     EDT
BRETRNJDI    **
*
RSTR  RTJ     INUIZN      UPDATE DIT
      LDA     IJSAV
      STA     IJOB
      ADD     =$3         FORM DIT CODE FOR RESTORE
      LLA     17
      STA*    DITI
      LDA     INTER
      SUB     =2         PREPARE RETURN TO INTERRUPTED PROGRAM
      SAM     **+1
      LDA     **
      SAM     RETURN
      SAM*    DITI        PUT RESTORE ADDRESS IN DIT
      DLD     IREG        RESTORE A AND E REGISTERS
IR1   LOX     *,1         AND INDEX ONE
IR2   LOX     *,2         TWO
IR3   LOX     *,3         THREE
      STP     EDT        SET ED TRAP
RETURNJDI    **         RETURN TO INTERRUPTED PROGRAM
*
IE      LDA     IENT,1    SET ICNTRL TO IENT
ICNTRLATE
      AMA*    DITI
      STA*    DITI        INSERT ICNTRL IN DIT
      ETA
      JEZ     WATEST      RETRIEVE ICNTRL
      ZOA
      STA     IDACT,1
WATESTLDA    WAIT
      JGZ     SETRAP
LNKTSTLDA    LIND
      JEZ     RSTR        WAS PROGRAM ON ACTION LEVEL WHEN INTERRUPTED
      DLD     IREG        NO-HEAD FOR JOB SELECTOR
      DST     LREG        YES-WAS THERE A LINK INTERRUPTED
      LDA     IR1         NO-GO TO RESTORE KUBER ACTION
      SAM     LR1         YES-SAVE REGISTERS FOR LINK RETURN
      LDA     IR2
      SAM     LR2
      LDA     IR3
      SAM     LR3
      LDA     INTER      GET LINK RETURN ADDRESS
      SUB     =2

```


SAM	**+1	
LDA	**	
SAM	LRETRN	
ZOM	LIND	RESET LINK INDICATOR TO ZERO
SETRAPSTP	EDT	
JDI	**+1	
JOBSELLDA	=1	
STA	WAIT	
CYCLE ZOM	CYCLEC	INITIALIZE CYCLE COUNT
CYCLERIMT	CYCLEC	
LDA	=5	TAKE FIVE ROMAN STEPS AROUND CYCLE
KAH	CYCLEC	
JMP	LNKLOK	LOOK FOR UNFINISHED LINK IF ALL JOBS INACTIVE
LDA	KJOB	
ADD	=1	INCREMENT KJOB
KAL	=5	IS IT OUT OF RANGE
JMP	RECYCL	YES-GO TO SET KJOB TO ONE
SETKJBSTA	KJOB	NO-STORE
LAX	,1	
STA	IJOB	
LDA	IDACT,1	IS KJOB DEACTIVATED
JEZ	WATSET	NO-GO TO PROCESS JOB
JMP	CYCLER	YES-LOOK AT NEXT JOB
*		
LNKLOKLDA	LNKJOB	IS THERE AN INTERRUPTED LINK TO FINISH
JEZ	LZYCLK	NO-GO TO LAZYCLOCK
LRSTR ZOM	WAIT	YES-RESTORE REGISTERS FOR RETURN TO LINK
STA	IJOB	
RTJ	UIZN	
LDA	IJOB	
ADD	=\$15	SET LINK RESTORE CODE
LDE	LRETRN	GET RETURN ADDRESS
LLE	7	
LLD	17	POSITION DIAGNOSTIC INFORMATION
STA*	DITI	PUT IN DIT
LR1 LOX	*,1	
LR2 LOX	*,2	
LR3 LOX	*,3	
DLD	LREG	
IMT	LIND	SET LINK INDICATOR TO ONE
LRETRNJMP	**	RETURN TO INTERRUPTED LINK
*		
RECYCLLDA	=1	RESET KJOB TO ONE
JMP	SETKJB	
*		
LZYCLKIMT	ILC	LAZY CLOCK TO COUNT TIME NOT SPENT
JMP	*-1	ON MEASURING MACHINE JOBS
IMT	ITIME	EACH LOOP TAKES 79.9 SECONDS
JMP	*-3	
*		
WATSETZOA		SET ACTION LEVEL TO ZERO
STA	WAIT	
TASPROZOA		
STA	IENT,1	SET INTERRUPT CONTROLS
STA	IACK,1	TO IGNORE

```

      LDA*   ITHIS,1
NOP    NOP
      LLA    6          DEBUG TASKWORD CHECKPOINT
                        GET END OF TASK INDICATOR
      RLA    23
      STA    IENDT,1    SET FOR IJOB
TSTEP  LDA    ITHIS,1
      SAM    POINTR     SET OPERATION POINTER TO TASK WORD POINTER
      LDA*   POINTR
      RLA    18
      KAL    =$15       IS THIS A MEMORY OPERATION CODE
                        NO
      JMP    OPSEL      YES  IS IT ARITH
      KAL    =$11       YES
      JMP    LOSTAR     YES
      KAL    =6         NO  IS IT LOAD OR STORE
      JMP    OPSEL      NO
LOSTAR LDA*   POINTR   YES
      LLA    7
      RLA    22         IS ROOT CLASS ZERO
      JGZ    ADDVAR     NO- GO TO FORM ADDRESS FOR VARIABLE
*
* ADDRESS CONSTRUCTION SECTION
*
ADDCONLDE* POINTR     FORM ADDRESS FOR CLASS ZERO ROOT
      AND    =$37777    GET ADDRESS PORTION
      AOA    KUTAB
      SAM    KUVAR
KUVAR  LDA    **
      LLA    7          LOOK AT NUMBER OF SUBSCRIPTS
      RLA    22
      JGZ    SUBVAR     GO TO SUBSCRIPTED VARIABLE PROCEDURE
      LDA*   KUVAR      GET VARIABLE ADDRESS
      SAM    OPERA
      JMP    OPSEL      GO TO OPERATION SELECTION
*
* SECTION TO PROCESS 1,2,OR 3 SUBSCRIPTS
*
SUBVARSTA INNO        SET NUMBER OF INDICES
      ADM    ITHIS,1
      ADM    POINTR     SET TAWTA POINTER TO LAST PARAMETER
      ZOA
      STA    I          INITIALIZE
      STA    J          INDEX
      LDA    =1         AND
      STA    IR         RANGE
      STA    JR         CELLS
      LDA*   KUVAR
      LLA    6
      RLA    23         IS K THE MACHINE NUMBER
      JEZ    SUBK       NO
      LDA    IJOB       YES
      SUB    =1
      STA    K          SET K=IJOB
      LDA    =-1
      ADM    ITHIS,1    RESET TAWTA POINTER
      ADD    INNO       DECREMENT INDEX NUMBER

```

	JEZ	SUBL	IF ZERO GO TO L FORMATION
	JMP	SUBJ	OTHERWISE GO TO J FORMATION
*			
GETSUBPAR	**		
LDA*	POINTR		GET TASK WORD
LLA	7		
RLA	22		GET PARAMETER TYPE
RTJ	LIMBO		CALCULATE LOCATION OF INDEX VARIABLE
SAM	**+1		
LDA	**		GET INDEX VALUE
SUB	=1		
ATE			
LDA	INNO		
SUB	=1		DECREMENT INDEX NUMBER
JMP*	GETSUB		RETURN WITH SUBSCRIPT VALUE IN E, INNO IN A
*			
SUBK	RTJ	GETSUB	GET K
	STE	K	SET K
	JEZ	SUBL	IF INNO IS ZERO GO TO L FORMATION
SUBJ	STA	INNO	IF NONZERO FORM J
	KAQ	=2	
	JMP	JROP	
	LDA*	KUVAR	CALCULATE LOCATION
	RLA	18	OF INDEX RANGE INFORMATION
	AOA	ANTAB	FOR TRIPLY SUBSCRIPTED ARRAY(I,J,K)
	SAM	ANVAR	
ANVAR	LDA	**	
	SAM	JR	SET J RANGE
	RLA	18	
	SAM	IR	SET I RANGE
	JMP	SETJ	
JROP	LDA*	KUVAR	
	RLA	18	
	SAM	JR	SET J RANGE FOR ARRAY(J,K)
SETJ	LDA	=-1	
	ADM	POINTR	DECREMENT POINTER
	RTJ	GETSUB	GET J
	STE	J	SET J
	JEZ	SUBL	IF INNO IS ZERO GO TO L FORMATION
SUBI	LDA	=-1	
	ADM	POINTR	DECREMENT POINTER
	RTJ	GETSUB	GET I
	STE	I	SET I
SUBL	LDA	K	CALCULATE
	MPY	JR	$L = (I + IR * (J + JR * K))$
	ETA		
	ADD	J	IF ARRAY HAS FEWER THAN THREE
	MPY	IR	SUBSCRIPTS THE CALCULATION WILL
	ETA		BE DONE WITH INITIALIZED VALUES
	ADD	I	WHICH HAVE NO EFFECT ON L
	STA	L	
	LDE*	KUVAR	
	AND	=\$37777	GET ARRAY BASE ADDRES
	ADD	L	ADD INDEX
	SAM	OPERA	SET VARIABLE ADDRESS

```

DPOINT LDA    =-1
ADM    POINTR  DECREMENT POINTER
JMP    OPSEL   GO TO OPERATION SELECTION
*
ADDVARRTJ LIMBO  CALCULATE ADDRESS OF VARIABLE
SAM    OPERA   SET OPERA TO ADDRESS
IMT    POINTR
LDA*   POINTR  LOOK AT NEXT TASK WORD
RLA    18      IS IT SUBSCRIPT
JGZ    DPOINT  NO-GO TO DECREMENT POINTER
IMT    ITHIS,1 YES-INCREMENT TAWTA POINTER
LDA*   POINTR
LLA    7
RLA    22      GET PARAMETER TYPE
RTJ    LIMBO   CALCULATE LOCATION OF INDEX VARIABLE
SAM    **+1    PREPARE LOAD
LDA    **      GET SUBSCRIPT
SUB    =1
ADD    OPERA   ADD TO ARRAY ADDRESS
SAM    OPERA   AND SET OPERATION ADDRESS
JMP    DPOINT  GO TO RESET POINTER FROM PAR TO OP
*
*   ENTER WITH TYPE IN A.  EXIT WITH LOCATION IN A
*
LIMBO PAR    **
KAH    =2      IS ROOT TYPE THREE
JMP    COMID  NO
LDA*   POINTR  YES
JMP*   LIMBO
COMID KAL    =2      IS ROOT TYPE ONE
JMP    VIDVNT GO TO IDEVNT VARIABLE ADDRESS CONSTRUCTION
LOA    IICOM  GET IICOM BASE ADDRESS
JMP    **+2
VIDVNTLOA IDEVNT  GET IDEVNT BASE ADDRESS
STA    TBASE
LDA    IJOB
SUB    =1
MPY    =100    MPY IDEVNT RANGE TIMES IJOB-1
ONE TWOSTE TRANGE
LDE*   POINTR
AND    =$37777
SRR    =1
ADD    TRANGE  FORM DOUBLE SUBSCRIPT
ADD    TBASE
JMP*   LIMBO
*
INC    IMT    ITHIS,1  INCREMENT TAWTA POINTR
ENDTSK LDA IENDT,1    GET END OF TASK CONTROL
JEZ    JOBSEL  GO TO ANY TASK
JMP    TASPRO  GO TO NEXT STEP
*
*   CODE TO PERFORM OPERATION SELECTION
*
ERR    HLT    11      HALT IF OPERATION CODE IS INVALID
OPSEL LDA*   POINTR

```

RLA	18	ISOLATE NUMERIC OP CODE
KAL	=\$33	SKIP IF NOT SUBROUTINE OR MACHINE OPERATION
JMP	DIP	GO TO DIAGNOSTIC INFORMATION PREPARATION
SIFSELLAX	,2	SET INDEX TWO FOR OPERATION SELECTION
JMP*	**+1,2	JUMP TO OPERATION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	LOA	01 M TO A
PAR	LDE	02 M TO E
PAR	LMA	03 -M TO A
PAR	STA	04 A TO M
PAR	STE	05 E TO M
PAR	LBA	06 COL A OF BUTTONBOARD TO A
PAR	LBB	07 COL B OF BUTTONBOARD TO A
PAR	LBC	10 COL C OF BUTTONBOARD TO A
PAR	ADD	11 A+M TO A
PAR	MPY	12 A*M TO A
PAR	SUB	13 A-M TO A
PAR	DVD	14 A/M TO A
PAR	LDX	15 X FROM MEAS MACH TO A
PAR	LDY	16 Y FROM MEAS MACH TO A
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	JMP	21 TRANSFER CONTROL
PAR	JMF	22 JUMP IF MEM FAIL
PAR	JBD	23 JUMP IF BAD DATEX CODE
PAR	JET	24 JUMP IF END OF TRACK
PAR	JRF	25 JUMP IF READ FAIL
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	JLZ	30 JUMP IF A LT 0
PAR	JGZ	31 JUMP IF A GT 0
PAR	JEZ	32 JUMP IF A EQ 0
PAR	SRT	33 SET RETURN, TRANSFER WITHIN TAWTA
PAR	RET	34 RETURN WITHIN TAWTA
PAR	CAL	35 CALL CORE RESIDENT SUBROUTINE
PAR	LNK	36 CALL LINK
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	ERR	00 ERROR-INVALID INSTRUCTION
PAR	RDP	60 READ DATA POINT
PAR	LTV	61 LIGHT VIEW

PAR	DMS	62	DISPLAY MESSAGE
PAR	LBV	63	LIGHT BINAVIEW
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	ERR	00	ERROR-INVALID INSTRUCTION
PAR	SIC	77	SET INTERRUPT CONTROL

*

* DIAGNOSTIC INFORMATION PREPARATION ROUTINE

*

DIP	ATE		
	RTJ	UIZN	UPDATE DIT
	ETA		RETRIEVE OP CODE
	LLA	2	
	ADD	IJOB	INSERT MACHINE NUMBER
	SUB	=1	
	LLA	17	POSITION DIT CODE AND SHIFT OFF BIT 6
	STA*	DITI	
	ETA		RETRIEVE OP CODE
	JMP	SIFSEL	

*

* UPDATE INDEX AND ZERO NEXT WORD OF DIT

*

INUIZN	PAR	**	ENTRY FROM INTERRUPT PROTECTED LEVEL
	IMT	SWITCH	SET INTERRUPT LEVEL INDICATO SWITCH
	JMP	**+2	
UIZN	PAR	**	
	IMT	DITI	INCREMENT DIT INDICES
	IMT	DITIP	
	LOA	DITI	
	KAH	DITRNG	IS DITI OUT OF RANGE
	JMP	**+3	NO
	LOA	DIT	YES-RESET TO BEGINNING OF TABLE
	SAM	DITI	
	LOA	DITIP	
	KAH	DITRNG	IS DITIP OUT OF RANGE
	JMP	**+3	NO
	LOA	DIT	YES-RESET TO BEGINNING OF TABLE
	SAM	DITIP	
	ZOA		ZERO NEXT WORD
	STA*	DITIP	
	KAL	SWITCH	IS INTERRUPT INDUCATOR SWITCH ON
	JMP*	UIZN	NO-RETURN TO STANDARD CALL
	ZOM	SWITCH	YES-RESET SWITCH TO ZERO
	JMP*	INUIZN	RETURN TO INTERRUPT LEVEL CALL

*

* CODING FOR REGISTER AND ARITHMETIC OPERATIONS

*

LDA	LDA*	OPERA	GET WORD FROM MEMORY
PUT	STA	AREG-1,1	PUT IN SOFTWARE A REGISTER FOR THIS JOB
	JMP	INC	RETJRN TO JOB CONTROL
LDE	LDA*	OPERA	
	STA	EREG-1,1	LOAD EREG FROM MEMORY
	JMP	INC	
LMA	LDA*	OPERA	
	MNA		LOAD AREG WITH NEGATIVE OF MEMORY WORD
	JMP	PUT	
STA	LDA	AREG-1,1	
	STA*	OPERA	STORE AREG IN MEMORY
	JMP	INC	
STE	LDA	EREG-1,1	
	STA*	OPERA	STORE EREG IN MEMORY
	JMP	INC	
LBA	LDE	IRJPAR-1,1	PUT COLUMN ONE IN A
	AND	=\$17	
	JMP	PUT	
LBB	LDE	IRJPAR-1,1	PUT COLUMN TWO IN A
	RLE	8	
	JMP	LBA+1	
LBC	LDE	IRJPAR-1,1	PUT COLUMN THREE IN A
	RLE	16	
	JMP	LBA+1	
ADD	LDA*	OPERA	
	ADD	AREG-1,1	ADD M TO A
	JMP	PUT	
MPY	LDA*	OPERA	
	MPY	AREG-1,1	MULTIPLY A BY M
	STE	AREG-1,1	
	JMP	INC	
SUB	LDA	AREG-1,1	
	SUB*	OPERA	SUBTRACT M FROM A
	JMP	PUT	
DVD	ZDA		
	LDE	AREG-1,1	
	DVD*	OPERA	DIVIDE A BY M
	NOP		
	STE	AREG-1,1	
	JMP	INC	
LDX	LDA	IX-1,1	
	JMP	PUT	PUT X IN A
LDY	LDA	IY-1,1	
	JMP	PUT	PUT Y IN A
*			
* CODING FOR LOGIC OPERATIONS			
*			
JMP	LDA*	ITHIS,1	TRANSFER
	SAM	ITHIS,1	PUT NEW ADDRESS IN ITHIS
	JMP	ENDTSK	
JMF	LDA	IMF-1,1	GET MEMORY FAIL IDICATOR
	JEZ	INC	
	ZOE		
	STE	IMF-1,1	RESET INDICATOR
	JMP	JMP	

JBD	LDA	IBD-1,1	BAD DATA TRANSFER
	JEZ	INC	
	ZOE		
	STE	IBD-1,1	RESET INDICATOR
	JMP	JMP	
JET	LDA	IET-1,1	TRANSFER IF END OF TRACK INDICATOR IS SET
	JEZ	INC	
	ZOE		
	STE	IET-1,1	RESET INDICATOR
	JMP	JMP	
JRF	LDA	IRF-1,1	TRANSFER IF READFAIL INDICATOR IS SET
	JEZ	INC	
	ZOE		
	STE	IRF-1,1	RESET INDICATOR
	JMP	JMP	
JLZ	ZOA		
	KAH	AREG-1,1	IS AREG LESS THAN ZERO
	JMP	INC	NO
	JMP	JMP	YES
JGZ	ZOA		
	KAL	AREG-1,1	IS AREG GREATER THAN ZERO
	JMP	INC	NO
	JMP	JMP	YES
JEZ	ZOA		
	KAQ	AREG-1,1	COMPARE AREG TO ZERO
	JMP	INC	UNEQUAL
	JMP	JMP	EQUAL

*

* CODING FOR TRANSFER OPERATIONS

*

SRT	IMT	NRET,1	INCREMENT RETURN STACK POINTER
	NOP		
	LDA	NRET,1	SET FOR RETURN AND TRANSFER
	LLA	14	ADJUST TRANSFER STACK COUNTER
	AMA*	DITI	
	STA*	DITI	
	LDA	IJOB	
	SUB	=1	
	MPY	=10	
	ETA		
	ADD	NRET,1	CALCULATE LOCATION OF
	ANA	IRFT	IRFT(NRET(IJOB), IJOB)
	SAM	SETRET	SET OPERAND ADDRESS
	LDA	ITHIS,1	GET POINTER FOR
	ADD	=1	NEXT TASK WORD
SETRET	SAM	**	PUT IN RETURN STACK
	LDE*	ITHIS,1	
	AND	=63777	
	SAM	ITHIS,1	PUT NEW ADDRESS IN ITHIS
	AMA*	DITI	
	STA*	DITI	
	JMP	ENDTSK	
RET	LDA	IJOB	CALCULATE RETURN ADDRESS LOCATION
	SUB	=1	
	MPY	=10	

ETA		
ADD	NRET,1	
AOA	IRET	
SAM	*+1	
LDA	**	GET RETURN ADDRESS FROM STACK
SAM	ITHIS,1	
SAM*	DITI	INSERT RETURN ADDRESS IN DIT
LDA	NRET,1	
SUB	=1	DECREMENT RETURN STACK POINTER
STA	NRET,1	
LLA	14	ADJUST TRANSFER STACK COUNTER
AMA*	DITI	
STA*	DITI	
JMP	ENDTSK	
CAL	LDA* ITHIS,1	CAL RESIDENT SUBROUTINE
	SAM* DITI	PUT SUBROUTINE ADDRESS IN DIT
	SAM EXSTEP	
EXSTEP	RTJ **	EXECUT STEP
	RTJ UIZN	UPDATE DIT
	LOA ,1	SET MACHINE NUMBER
	ADD =\$123	FORM DIT CODE FOR SUBROUTINE FINISH
	LLA 17	
	STA* DITI	
	JMP INC	
LNK	LDA LNKJOB	ANY LINK STILL IN PROCESS
	JEZ NULINK	NO-SET NEW LINK
	JMP LRSTR	YES-GO TO RESTORE REGISTERS AND RETURN
NULINK	LDA =1	
	STA IDACT,1	DEACTIVATE IJOB
	LDA IJOB	
	STA LNKJOB	
	LDA* ITHIS,1	GET LINK NUMBER
	SAM LINKNO	SET LOGICAL LINK NUMBER
	SAM* DITI	PUT LINK NUMBER IN DIT
	IMT LIND	SET LINK INDICATOR TO ONE
	RTJ LINKER	GO TO ACTUAL LINK
	ZOM LIND	RESET LINK INDICATOR TO ZERO
	RTJ UIZN	UPDATE DIT
	LDE LINKNO	SET LINK NUMBER
	LLE 7	
	LDA LNKJOB	SET MACHINE NUMBER
	STA IJOB	RESTOR IJOB
	LAX ,1	
	ADD =\$127	FORM DIT CODE FOR LINK FINISH
	LLD 17	
	STA* DITI	
	ZOA	
	STA LNKJOB	ZERO LNKJOB
	STA IDACT,1	ACTIVATE IJOB
	JMP INC	
*		
*	CODING FOR MEASURING MACHINE OPERATIONS	
*		
DMS	LDE AREG-1,1	PUT MESSAGE NUMBER IN E
	ETA	

```

      SAM* DITI          PUT MESSAGE NUMBER IN DIT
      LDA  IJOB          PUT MACHINE NUMBER IN A
      RTJ  MLM           DISPLAY MESSAGE
      JMP  INC           GO TO TASK
RDP    LDA  IJOB          PREPARE TO
      RTJ  MGP           READ DATA POINT
      JMP  INC           GO TO NEXT TASK
LTV    LDE  AREG-1,1     GET VIEW NUMBER
      ETA
      SAM* DITI          PUT VIEW NUMBER IN DIT
      LDA  IJOB          GET MACHINE NUMBER
      RTJ  MLV           GO TO LIGHT VIEW
      JMP  INC           GO TO NEXT TASK
*
*   LBV    IMT  IDACT,1   DEACTIVATE IJOB
*           IMT  IACK,1   SET ACKNOWLEDGE INTERRUPT
*           SXM  SAVI,1   SAVE IJOB
*           LDE  EREG-1,1 GET POSITION NUMBER IN E
*           ETA
*           LLA  4        POSITION BINAVIEW POSITION NUMBER
*           ADD  AREG-1,1  ADD DISPLAY NUMBER
*           SAM* DITI      PUT IN DIT
*           LDA  AREG-1,1
*           LAX  ,1        PUT DISPLAY NUMBER IN INDEX ONE
*           LDA  SAVI      GET MACHINE NUMBER IN A
*           RTJ  MLB       GO TO LIGHT BINAVIEW
*           LDA  SAVI
*           LAX  ,1        RETURN IJOB TO INDEX ONE
*           LDA  =0
*           STA  IDACT,1   ACTIVATE JOB
*           IMT  ITHIS,1
*           JMP  JOBSEL
*
*   *
*   *   INTERRUPT CODING
*   *
*   SIC    LDE* ITHIS,1   GET TASK WORD
*           AND  =$37      ISOLATE DECIMAL COCED ENT AND ACK CONTROLS
*           SAM* DITI      PUT ENT AND ACK CONTROLS IN DIT
*           ATE           PREPARE FOR
*           ZOA           DIVIDE OPERATION
*           DVD  =10
*           NOP           GET ENT IN E AND ACK IN A
*           STE  IENT,1
*           STA  IACK,1
*           IMT  IDACT,1   DEACTIVATE IJOB
*           IMT  ITHIS,1
*           JMP  JOBSEL
*
*   * ROUTINE TO READ IN A DATUM FROM THE SPECIFIED MACHINE
*   * COME INTO ROUTINE WITH MACHINE NO IN A
*   *
*   MGP    PAR  **
*           LLA  19        CREATE THE EDCW FOR THE
*           ADD  =$26000000 SPECIFIED MACHINE
*           STA  MIN+2

```

```

        AMA    = $100000
        STA    MTEST
MBAD    LDA    $103        PREPARE I/O PARITY FAIL INTERRUPT
        STA    MIOF
        LDA    MIOFIN
        STA    $103
        ZDA
        STA    IBD-1,1     REINITIALIZE THE
        STA    IMF-1,1     DATA QUALITY INDICATORS
        STA    IRF-1,1
        NOP
EXMIN    EXD    MIN        READ DATUM
        JMP    *-2
        NOP
        EXD    MTEST       CHECK THAT READIN IS COMPLETED
        JMP    *-2
MOUT    LDA    MIOF        RESET I/O INT LOC
        STA    $103
*
* ROUTINE TO CONVERT X AND Y VALUES IN DATEX CODE INTO ASI INTERNAL
* FIXED POINT FORM
*
        LDA    $102        PREPARE FOR TRANSLATION
        STA    MIMFSV       MEMORY FAIL
        LDA    MIMFIN
        STA    $102
        LDA    INPUT        MOVE REJECT CODE WORD
        STA    IRJPAR-1,1   TO INDEXED ARRAY
        LDA    INPUT+1
        RLA    23           SHIFT SIGN BIT TO POSITION ONE
        STA    IET-1,1     SET END OF TRACK INDICATOR
        LDE    INPUT+1      GET X
        RTJ    NONON        TRANSLATE
        STA    IX-1,1       STORE IN X ARRAY
        LDE    INPUT+2      GET Y
        RTJ    NONON        TRANSLATE
        STA    IY-1,1       STORE IN Y ARRAY
MFRST    LDA    MIMFSV
        STA    $102
        JMP*    MGP        RETURN
*
MIORESZDA    I/O PARITY FAIL RESTORE ROUTINE
        STA    INPUT       ZERO DATA AREA
        STA    INPUT+1
        STA    INPUT+2
        LDA    =1
        STA    IRF-1,1     SET FAIL INDICATOR
        SAM*    DITI
        LDA    MIOF
        STA    $103
        JDI*    MGP
MFSET    LDA    =1        DECREMENT INTERRUPT LEVEL AND RETURN
        STA    IMF-1,1
        LLA    1
        SAM*    DITI

```

	JDI	MFRST	SET FAIL INDICATOR AND END TRANSLATION
*			
NONON	PAR	**	NO NONSENSE ROUTINE
	LLE	4	SHIFT OFF NONSENSE
	ZOA		
	STA	DECEN	CLEAR DECIMAL CELL
	LDA	=-5	
	STA	COUNT	SET DIGIT COUNTER
TRANS	ZOA		CLEAR A REGISTER
	LLD	4	SHIFT NEXT DIGIT INTO A
	STE	MTEMP	STORE REMAINING DIGITS
	LOX	9,2	SET TRANSLATION INDEX
	KAQ	MDATEX,2	MATCH DATEX DIGIT WITH CODE TABLE
	JDX	*-1,2	
	STA	MVAL	
	LOA	,2	GET ASI EQUIVALENT
	JEZ	MZERO	CHECK LEGALITY IF NOT ONE THRU NINE
MGOOD	SAM	ODDINV	SAVE ASI NUMBER
	ADD	DECEN	ADD HIGHER ORDER DIGITS
	IMT	COUNT	ADVANCE DIGIT COUNTER
	JMP	**2	THERE ARE MORE DIGITS
	JMP*	NONON	TRANSLATION COMPLETE-RETURN
	MPY	=10	
	STE	DECEN	PREPARE DECIMAL CELL FOR NEXT DIGIT
	LDE	MTEMP	GET REMAINING CODE
	LDA	=-3	
	KAQ	COUNT	TEST FOR NONINVERTING COLUMN THREE
	JMP	**2	GO TO CHECK IF NUMBER WAS ODD
	JMP	TRANS	NO COLUMN INVERSION-CONTINUE TRANSLATION
ODDINVLOA	**		RETRIEVE ASI NUMBER
	LSA	22	ISOLATE ODD BIT
	JEZ	TRANS	NOT ODD-CONTINUE TRANSLATION
	MNE		ODD-INVERT FIRST BIT OF NEXT DATEX DIGIT
	JMP	TRANS	CONTINUE TRANSLATION
MZERO	LDA	MVAL	DATEX CODE WAS NOT 1 THRU 9
	KAQ	=\$16	IF IT IS NOT ZERO
	JMP	ILLEGL	IT IS ILLEGAL
	LOA	0,2	
	JMP	MGOOD	
*			
ILLEGL	LLDA	=1	
	STA	IBD-1,1	SET BAD DATA INDICATOR
	LLA	2	
	SAM*	DITI	
	JMP*	MGP	RETURN
*			
*	ROUTINE TO LIGHT SPECIFIED VIEW		
*	ENTER ROUTINE WITH MACHINE NUMBER IN A AND VIEW NUMBER IN E		
*			
MLV	PAR	**	
	LLA	19	FORM LIGHT VIEW INSTRUCTION
	ADD	=\$27000070	
	STA	MVIEW	
	ETA		
	ADM	MVIEW	ADD VIEW NUMBER OR MESSAGE NUMBER

```

      NOP
      EXD   MVIEW
      JMP   *-2
      JMP*  MLV
*
* ROUTINE TO DISPLAY OPERATOR MESSAGE
* ENTER ROUTINE WITH MACHINE NUMBER IN A AND MESSAGE NUMBER IN E
*
MLM   PAR   **
      LLA   19           FORM MESSAGE INSTRUCTION
      ADD   =$27000000
      STA   MVIEW
      LDA   MLM
      STA   MLV
      JMP   MLV+4
*
* ROUTINE TO LIGHT BINAVIEWS
* ENTER ROUTINE WITH MACHINE NUMBER IN A, POSITION NUMBER IN E,
* AND DISPLAY NUMBER IN INDEX ONE
*
MLB   PAR   **
      SXM   MI,1
      STA   MOPE
      LLA   19           FORM DISPLAY BINAVIEW INSTRUCTION
      ADD   =$27000000
      STA   MBIN
      ETA
      LLA   6
      AMA   MBIN
      AMA   MI
      STA   MBIN
      LDA   MOPE         TEST FOR MACHINE A OR B
      LLA   23
      AMA   =1
      JLZ   *+4          MACHINE A SELECT
      LDA   =$2000       MACHINE B SELECT
      AMA   MBIN
      STA   MBIN
      NOP
      EXD   MBIN         LIGHT BINAVIEW
      JMP   *-2
      LDA   TIME
      STA   TIMER
      IMT   TIMER
      JMP   *-1
      JMP*  MLB
*
OPINT JMP   LINKEX
LINKEXJDI *+1
      LDA   =13          SET UP LINK 13
      STA   LINKNO
      RTJ   LINKER       GO TO TAPE MARK OR DIAGNOSTIC PRINTOUT
      ZOA
      STA   LNKJOB
      JMP   JOBSEL

```

```

*
OPINOPSTE    $102
INOP  RTJ    IENT1
INEND RTJ    IENT1+16
RESTRJMP    INVAR
MIN  ARCW    1,LBM,INPUT
      ARCW    1,LBL,INPUT+3
      PAR     **

```

```

*
*   CONSTANTS AND RESERVED AREAS FOR VARIABLES
*

```

```

TEMP  RES    1
IREG  RES    2
LREG  RES    2
WAIT  OCT    0
KJOB  OCT    0
ILC   RES    1
LIND  OCT    0
JOBMAXRES  1
CYCLECOCT  0
POINTROCT  0
OPERA  OCT    0
TBASE  OCT    0
TRANGE OCT    0
I      OCT    0
J      OCT    0
IR     OCT    0
JR     OCT    0
INNO   OCT    0
K      OCT    0
L      OCT    0
AREG   RES    4
EREG   RES    4
IRJPARR  4
IX     RES    4
IY     RES    4
NRET   OCT    -1,-1,-1,-1,-1
IRET   RES    40
IJSV   RES    1
INPUT  RES    3
SAVI   RES    1
IBD    OCT    0,0,0,0
IET    OCT    0,0,0,0
IMF    OCT    0,0,0,0
IRF    OCT    0,0,0,0
MTEST  RES    1
MIOFINJMP MIORES
MIOF   RES    1
MIMFSVRES 1
MIMFINJMP MFSET
MVAL   RES    1
DECEN  RES    1
MTEMP  RES    1
MDATEX OCT    16,14,15,11,13,3,1,5,4,6
MVIEW  RES    1

```

```

MBIN RES 1
MOPE RES 1
MI OCT 0
COUNT RES 1
TIMER RES 1
TIME DEC -7000
DITI PAR DIT
DITIP PAR DIT+1
DITRNGPAR DIT+49
SWITCHOCT 0
*
KUT AB PAR **
00 ISYS,1
03* MFID,2
03* MVRT,2
03* IPOINT,2
01* IFIDX,3
02* IFIDY,3
03* IVERTX,3
04* IVERTY,3
05* ICORDX,3
06* ICORDY,3
03* NPT,2
00 IJOB
*
ANT AB PAR **
03 4
03 4
03 5
03 5
03 10
03 10
*
END

```

APPENDIX II. TABLE OF ASIST MNEMONICS

<u>Mnemonic</u>	<u>Operation</u>
ADD	Add to A
ADM	Add to Memory
AMA	Add Magnitude to A
AND	Logical 'And'
AOA	Add Address to A
ASR	Assembly Register Command
ATE	Copy A in E
AUX	Augment Index
AVA	Absolute Value A
AVE	Absolute Value E
AXK	A to External Line and Skip if Ready
CMA	Complement A
CMD	Complement AE
CME	Complement E
CSA	Circular Left Shift A
CSD	Circular Left Shift AE
CSE	Circular Left Shift E
CTP	Clear Flip-Flops
DAD	Add to AE
DLD	Load AE
DST	Store AC
DVD	Divide
ETA	Copy E in A
EXD	External Device Command
FAD	Floating Point Add
FCM	Floating Point Compare
FDV	Floating Point Divide
FMP	Floating Point Multiply
FSB	Floating Point Subtract
HLT	Halt

<u>Mnemonic</u>	<u>Operation</u>
IMT	Increment Memory and Test
IXT	Increment Index and Test
JDI	Jump and Decrement Interrupt Level
JDX	Jump and Decrement Index
JEZ	Jump if A Equal Zero
JGZ	Jump if A Greater Than Zero
JLZ	Jump if A Less Than Zero
JMP	Jump
JPO	Jump to Address Plus One
KAH	Skip if A Higher Than Memory
KAL	Skip if A Lower Than Memory
KAQ	Skip if A Equal to Memory
KEX	Skip on External Signal
KIF	Conditional Skip
KON	Skip if Address is One
KOZ	Skip if Address is Zero
KXH	Skip if Index is High
LAX	Load A in Index
LDA	Load A
LDE	Load E
LLA	Logical Left Shift A
LLD	Logical Left Shift AE
LLE	Logical Left Shift E
LOA	Load Address
LOR	Logical 'Or'
LOX	Load Address into Index
LSA	Left Shift A
LSD	Left Shift AE
LSE	Left Shift E
LXP	Load Exponent
MNA	Minus A
MND	Minus AE
MNE	Minus E

<u>Mnemonic</u>	<u>Operation</u>
MXK	Memory to External and Skip
NOP	No Operation
NRM	Normalize AE
RLA	Logical Right Shift A
RLD	Logical Right Shift AE
RLE	Logical Right Shift E
RND	Round off A
RSA	Right Shift A
RSD	Right Shift AE
RSE	Right Shift E
RTJ	Set Return Address and Jump
SAE	Sign of A to E
SAM	Store A Address in Memory
SCT	Save and Clear Flip-Flops
SEA	Sign of E to A
SEN	Send External Signal
SOA	Subtract Address from A
SST	Save and Set Flip-Flops
STA	Store A
STE	Store E
STP	Set Flip-Flops
SUB	Subtract
SXM	Store Index
SXP	Store Exponent
XAE	Exchange A and E
XAK	External Lines to A and Skip
XEC	Execute Instruction
XMK	External Lines to Memory and Skip
XOR	Exclusive 'Or'
ZOA	Zero A
ZOD	Zero AE
ZOE	Zero E
ZOM	Zero Memory
ZSA	Clear A to Sign of E