UNICOMP
incorporated

MAR 1971

UNICAP-II

COMP 18

ASSEMBLER MANUAL

18219 PARTHENIA STREET, NORTHRIDGE, CALIFORNIA 91324 • (213) 886-7722

COMP-18

UNICAP-II ASSEMBLER MANUAL

TABLE OF CONTENTS

COMP-18 UNICAP-II

ASSEMBLY PROGRAM

This manual describes programming the COMP-18 computer using the UNICAP-II assembly program. The reader is assumed to be familiar with the COMP-18 computer, as described in the COMP-18 Reference Manual. Further, it is assumed the reader is knowledgeable of computer terminology, operation, and programming.

UNICAP-II is a programmer's tool for developing application programs using a coding language composed of symbols and mnemonic codes rather than octal or binary representation of COMP-18 commands. Incorporated in UNICAP-II are various aids for program housekeeping, data entry, memory addressing, and program checkout.

The UNICAP-II assembler provides for:

- substitution of mnemonic codes for octal equivalents
- symbolic definition of memory addresses
- pseudo operations to modify the assembler's functions
- macro commands which generate several machine commands

UNICAP-II operates in either a one pass conversational mode or a two pass mode.

UNICAP-II will operate on a minimum configuration of 4,096 words of memory and using the ASR-33 paper tape or keyboard for input, ASR-33 paper tape punch for object program output, and the teleprinter for program listing.

UNICAP-II is written to operate efficiently with expanded memory and will operate with higher speed input/output devices, such as, punched cards, magnetic tape, line printers, and high speed paper tape reader/punch.

UNICAP-II SOURCE PROGRAM LANGUAGE

UNICAP-II will translate source language symbolic instructions and addresses into binary formated object program commands for the COMP-18. A single computer command word is generated for most symbolic instructions. Certain macro instructions and pseudo operations result in more than one command being generated.

CHARACTER SET

The following characters are recognized by the UNICAP-II assembly program.

Alphabetic Characters

ABCDEFGHIJKLMNOPQRSTUVWXYZ

Numeric Characters

0123456789

Special Characters

(Space) !"#$%&' ( ) *+, - . / : ; <=> ? @[\]↑←

Regulations covering the character usage are defined in the following portions of this section.

When UNICAP-II input is from the ASR-33 keyboard or paper tape, a carriage return and line feed, (CR) and (LF) terminate an instruction. The ASR-33 characters are in ASCII code, Appendix 1.

When UNICAP-II input is from punched cards, each card contains a single instruction. The punched card codes are shown in Appendix 2.

Throughout the remainder of the manual, the alphabetic and numeric characters will be referred to as alphanumeric characters.

DATA FORMATS

UNICAP-II provides for the storage of data values in six different formats. The format is specified by a special character preceding, following, or within the quantity.

A data value may be stored in a single COMP-18 word, multiple COMP-18 words, or included in the L Code of the command word.

If a data value is to occupy the full COMP-18 word (s), the first digit must be in the first position of the Function Field.

The Label Field of a data value is assembled.

The following describes each method of format specification.

Octal

An octal integer is specified as a signed or unsigned numeric value. However, if the integer has seven octal digits, it must be signed in order to distinguish it from the command format. No decimal period is allowed within an octal integer. If the unsigned octal integer has seven digits, it will be converted as a command format, i.e. the 18 bits will be broken into two 8 and 10 bit bytes, and each byte converted to a 3 or 4 digit octal number from 000_8 to 377_8 and 000 to 1777. Hence, the largest command format represented by the 18 bit word is 3771777.

The following are examples of the octal command and numeric format.

	COMP-18 (binary)																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
COMMAND FORMAT 3540311	1	1	1	0	1	1	0	0	0	0	1	1	0	0	1	0	0	1
NUMERIC FORMAT -35431	1	1	1	1	0	0	0	1	0	0	1	1	1	0	0	1	1	1
NUMERIC FORMAT 35431	0	0	0	0	1	1	1	0	1	1	0	0	0	1	1	0	0	1

Fixed Point Decimal

The fixed point decimal format is specified by the use of a decimal point. If the number contains no letter, it is assumed to be single precision fixed point decimal number scaled as an integer.

For example:

100. assembles as **0000144**
-10.3 assembles **3771366** (notice the loss of fractional portion)

The binary scaling factor is allowed within the single precision fixed point numbers. The fixed point number may be followed by the letter B and a signed integer which specifies the scaling factor.

For example:

9.5B4 will assemble as **1140000**

The scale factor B4 moved the binary point from between bit position 1 and 2 to between bits 6 and 7.

3.14159B2 will assemble as **1440434**

Floating Point Decimal

The floating point format is specified by the letter E, followed by a decimal number between 63 and -64.

The numeric value preceding the letter E may be signed and it may contain a decimal point but in any case, it must contain at least one digit.

The floating point numbers are stored in two computer words as:

1	2 – 8	9 – 18
S	Exponent	Mantissa
Mantissa		

The exponent is stored in bits 2-8 of the first word. The sign of the exponent is bit position 2 of the first word. The Mantissa is in bit position 9 through 18 of the first word and 1-18 of the second word. The sign of the Mantissa is in bit position 1 of the first word.

The following is an example of a floating point format and the binary equivalent:

	COMP-18 (binary)																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1.E-1	0	1	1	1	1	0	0	0	0	0	0	0	1	0	0	1	1	0
	0	0	1	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0

Alphanumeric Constant

This format is specified by the use of a quote (") mark preceding the character. A single character will be stored in the COMP-16 word in ASCII code, reference Appendix 1. Only one character may follow the quote mark. The following is an example of the format and the binary equivalent.

	COMP-18 (binary)																	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
"A ;CODE	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
LAP 6 "A	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	1

UNICAP-II ELEMENTS

The UNICAP-II source program consists of lines of coding, each one representing either one machine program step or a so called pseudo operation which produces no machine coding but allows communication between the programmer and the assembler. Through this communication, the programmer can change the way his program is being assembled.

The machine instructions that the computer executes are binary words consisting of 18 bits. Description on how they are interpreted and executed can be found in the Reference Manual. Programs can be prepared for the computer by simply inserting the right combinations of 1's and 0's in computer words. This process, however, is extremely tedious, time consuming and sometimes simply impossible for the human programmer due to the size of the programs. UNICAP-II is an intermediary between the computer machine language form of programs and the programmer wanting to solve a problem. The assembler allows replacement of certain binary strings with easily remembered mnemonic words called symbols. This removes the need for memorizing a mass of numbers, easing considerably the task of programming. This, however, does not remove the stringent grammatical restrictions of the machine programming.

The programming instructions must follow those rules which are described in this manual.

SYMBOLS

A symbol is a string of alphanumeric characters of which only the first four are recognizable by the assembler. A symbol is considered ended by either four characters or by a non-alphanumeric character. The examples of legal symbols are:

BUF
TM34
C400
XYZX
SYMBOL
A
A1

Notice that in the word SYMBOL, only the first four characters SYMB are recognized by UNICAP-II. The programmer must be careful not to use long symbols whose first four characters are the same. For example, INTERVAL and INTEGRAL will be considered the same and produce the "doubly defined" error diagnostic. They should be abbreviated as: INVL and INGR.

There is another special symbol: * which always has the value of the present location counter.

SYMBOLS BEGINNING WITH $ SIGN

There is another privileged class of symbols which begins with $ and then follows the above rules for symbols. These symbols allow inter-program communications in the fact that they are saved by the assembler from program to program and allow tieing to many routines into a system. The usual symbols are valid only between two NEW pseudo operations. The action of NEW pseudo operations is described later in this manual.

LABELS

A label is a symbol which describes a program location value. This is a common way of defining a symbol. A label may be either a usual symbol or a "dollar sign" symbol.

EXPRESSIONS

Since symbols replace binary strings, it would be desirable to be able to perform certain arithmetic operations on the symbols with the assembler performing the same with the binary numbers. Such a capability is included in UNICAP-II. The operations are as follows:

+	Add
-	Subtract
'	Or
&	And
↑	Shift left

Expression consists of any number of symbols and operations in between. No parenthesis are allowed. Expressions are evaluated from left to right.

For example:

6032+ 5 - 2 has the value 6035

If the symbol AB has the value 03146, then AB & 37 has the value 00046.

In another example: 1↑9.+ 4 has the value 01004. The special symbol * may be used in the expressions and it will always have the value of present location counter.

INSTRUCTION FORMAT

The UNICAP-II source language program command is composed of five Fields: Function, Index, Variable, Label, and Comment. The following describes the use and conventions associated with each Field.

Function Field

The Function Field contains either a data value or a mnemonic code for a COMP-18 Function, assembler pseudo operation, or assembler macro. A data value is stated in the format specified for one of the various UNICAP-II data formats reference page 1-2. A list of allowable menmonic Function Field codes, and their meaning is shown in Appendix 3. The first character of the data value or mnemonic code must occupy the first Field position or an assembly error will be indicated. As a special case, an * in the first Function Field location will result in all Fields being treated as a Comment. The following is an example of Function Field codes written on the UNICOMP coding sheet.

FUNCTION	I	VARIABLE	LABEL	COM
ADD				
STA				
ORG				
3000				
13.4B-4				
* THIS IS A COMMENT				

Index Field

The Index Field specifies the mode of addressing that is to be used. A digit, 0 through 7, is used to designate the address mode, as follows:

Index	Meaning
0	Direct Address
1 through 6	Indexed Address
7	Indirect Address

This Field is identical to the meaning of the I Code in a COMP-18 command. The Index Field must be separated from the Function Field by a space.

Variable Field

The Variable Field is a four character Field. It is the operand, operand address, shift count, memory address, or peripheral device code. The Variable Field may be expressed as an octal or decimal integer or symbolically defined. Arithmetic operations of addition or subtraction and logic operations of AND or OR and SHIFT may be indicated as part of the Variable Field statement. When arithmetic or logic operations are used, the Variable Field will exceed four characters.

If the Variable Field contains a symbolic reference, then that symbol must be defined by the Label Field of another command.

The Variable Field may be separated from the Index Field by a space character or a comma. If a comma is used to separate the Fields, then the Variable Field must contain a numeric value, octal or decimal, less than 1777_8. The following is an example of Function, Index, and Variable Field statements written on a UNICOMP coding sheet.

FUNCTION	I	VARIABLE	LABEL	COMMENTS
LAP		BUF+14		
SLL		2		
STA		105		
JMP		*-5		
LAP	2,	5		
STA	4,	14		
ADD		XX+YY		
OUT	3,	6		
LAN	6,	75		
SUB	6	50.		

Label Field

The Label Field is used to symbolically identify the location of the source language command or a reserved memory location. The Label Field is five characters in length. The first character of the Label must be alphabetic or $ sign. Space characters are not allowed in the Label. The Label Field is separated from the Variable Field by a semi-colon (;). The semi-colon is not interpreted as part of the Label.

Comment Field

The Comment Field is used expressly for program notation and has no bearing on the assembly process. The Comment Field follows the Label Field and must be separated from that field by a space character.

The following are examples of various source language commands.

FUNCTION	I	VARIABLE	LABEL	COMMENTS
ORG		2000		PROGRAM STARTING LOCATION
JMP	7	200		JUMP INDIRECT
SLL		5	;SFTL	SHIFT LEFT 5
LAP	6,	4		LOAD ACC WITH 4
STA		XXX		STORE ACC IN XXX

INSTRUCTION ASSEMBLY

The source language commands are processed into COMP-18 commands in binary format. The commands are assembled in sequential order, as specified by the order of the source language commands. The COMP-18, **eighteen bit binary command word format is:**

1 2	3 4 5	6 7 8	9	10 11 12	13 14 15	16 17 18	**bit positions**
D_1	D_2	D_3	D_4	D_5	D_6	D_7	**octal digits**
F Code		I Code		**L Code**			**command codes**

and,

F Code - is the five bit code of one of the 31 Functions,

I Code - specifies direct or indirect addressing or designates the use of one of the six index registers,

L Code - is the base of the effective address.

During the assembly process, the Function Field of the source program command is converted from the menmonic code into the binary code of the COMP-18 Function. If the Function Field contains a pseudo operation or macro, the appropriate COMP-18 Functions, if any, are generated and placed in the command string sequence. If UNICAP-II does not recognize the Function Field Code as an allowable mnemonic, the error message, MN is indicated, reference Appendix 4.

The Index Field will be placed in the I Code portion of the command word. If the reference address setting macro, IND, has been used, then UNICAP-II will place the appropriate index register number in the I Code portion of the command word.

If no Index Field value is given for a Shift Function, and the shift count is numeric, the I Code will be set to zero.

The Variable Field contains the operand, operand address, shift count, peripheral device code, or memory address. UNICAP-II interprets the Variable Field into a quantity that is placed in the L Code of the command word. If the Variable Field contains an octal number, then that value is placed in the L Code. If the Variable Field contains a decimal number, specified by a decimal point, then the integer portion is converted to binary and placed in the L Code. If the Variable Field is a symbolic reference, then the difference, between the program location of the corresponding IND pseudo and the memory address referenced by the symbol is placed in the L Code portion of the command word.

The Character * in the Variable Field is used to mean the present address. Typically, the * is used with a positive or negative number to reference a location that many addresses removed from the present location.

The Label Field is used to symbolically define a memory location. The Label is assigned the memory address of the command. Use of the symbolic Label in the Variable Field of another command would result in a reference to the Labeled command. The Label address is computed from the origin of the program. If a program is assigned an origin of 3000_8 and the first instruction is Labeled STAR, then STAR would be given the address 3000_8.

The Comment Field is ignored by UNICAP-II.

The following are examples of source program commands and the corresponding assembled COMP-18 command word. Note the various Variable Field formats.

SOURCE COMMANDS

FUNCTION	I	VARIABLE	LABEL	COMMENTS	COMP-18 COMMANDS (Octal)
SPC		101		P.C. TO REG 1	0400101
LAP	6	147		L CODE IS OPERAND	1060147
SLL		2	;SFTL	LEFT SHIFT 2	3200002
JAN		SFTL		JUMP ACC NEGATIVE	2310001
JMP		*+3			2710006
RIN		101		L IS THE DEVICE CODE	0600101
ADD	6	14.		OPERAND IS DECIMAL	1260016

PSEUDO OPERATIONS

The following pseudo operations are available to the programmer as aids for performing program housekeeping operations. The pseudo operations are written in the Function Field of the source commands followed by the associated parameters. A Label, used with a pseudo operation, is written following the last parameter, and preceded by a semicolon. The semicolon is not interpreted as part of the Label.

The following described each pseudo operation in UNICAP-II.

BSS

A block of memory is reserved starting with the present location, and following, for the number of locations specified by the parameter. The Label Field is used to symbolically reference the reserved locations. The pseudo operation, BSS, is written in the Function Field. For example, the following would reserve 10_8 locations, with the first location symbolically named BUF,

```
BSS 10 ;BUF
```

BES

This pseudo operation is identical to BSS, except the location following the last location of the buffer reserved is assigned to the symbol in the Label Field. The pseudo operation is written in the Function Field. For example, the following would reserve 10_8 locations, with the last location plus one symbolically named SAV,

```
BES 10 ;SAV
```

SET

The format of the SET pseudo operation is as follows:

```
SET (label) = (an expression)
```

The label referenced in the SET pseudo before the sign will from now on have the value of the expression. In the following example, the symbol .ABUF will be assigned the value of symbol BUF plus two.

```
BSS 10 ;BUF
SET ABUF=BUF+2
```

In another example, if one wishes to have a symbol having a numerical value, use:

```
SET TTY = 100
```

From this point on, when symbol TTY is used, it will assemble as 100_8, for example:

OUT TTY will assemble as 0700100

NEW

The NEW pseudo operation signals UNICAP-II that a new symbol table is to be created. Labels separated by a NEW pseudo operation may be the same without the "doubly defined" error diagnostic. However, labels beginning with the $ sign are still useable and the NEW pseudo operation has no effect on them. The pseudo operation is written in the Function Field. There are no parameters with this pseudo operation.

One should then remember that no reference to usual symbols defined before the last NEW or after the next NEW should be made, or an "undefined symbol" error diagnostic will be listed. Reference to symbols beginning with the $ sign can be made without regard to NEW pseudos in the program.

NSYM

The NSYM pseudo operation signals UNICAP-II that a new program is to be assembled. All previous symbol tables, including the "dollar sign" symbols are destroyed. This pseudo will essentially reinitialize the assembler. It should be used when a new program is to be assembled, and the state of the previous assemblies is not known. There are no parameters with this pseudo operation.

END

The END pseudo operation signals the end of the source language commands for the current program. The symbol table compiled for the source language commands will be retained and used for assembling the following source programs. The END pseudo operation is written in the Function Field. There are no parameters with this pseudo operation.

ORG

The ORG is followed by an octal or decimal integer or a symbolic address. The parameter of the ORG defines the starting address of the program. If a symbol is the starting address, it must have been previously defined. An integer without a decimal point will be interpreted as an octal quantity. The following are examples of the use of the ORG pseudo operation.

ORG 400 . .	Location 400_8 will be the starting address of the program.
BSS 1 ;STAR	
ORG STAR . . .	The Location defined by the Label, STAR, will be the starting address of the program.
ORG 512 . . **END**	Location 1000_8 will be the starting address of the program.

CALL

The CALL pseudo operation is to reference a subroutine in the UNICAP system library. UNICAP-II will generate the necessary linkage to exit from the main program to the referenced subroutine. The name of the routine referenced by the CALL pseudo operation will be used to search a library tape for the subroutine. The pseudo operation is written in the Function Field followed by the parameter. For example, the following command would result in loading the subroutine into memory from a library tape and executing that program as a portion of a main program:

```
CALL  $EXPX
```

This will generate the JPR machine instruction with the address that was used by the assembler for storage of the subroutine. The UNICAP system library contains many fixed and floating subroutines already and is destined to grow in the future.

IND

The format of the IND pseudo operation is:

```
IND  (n) =  (an expression)
```

where n is an octal number from 1 to 6, the expression on the right will be evaluated and the value of it will be saved by the assembler for subsequent use in computing the addressing mode. Whenever the address called for is greater than direct addressing capability of the computer but it falls within the range of the nth index, this index will be assembled with appropriate displacement in the L portion of the instruction. Usually the IND pseudo operation follows directly the SPC instruction at the beginning of a program. All routines which reside in non-directly addressable memory position should start with the sequence as follows:

```
SPC  101
IND 1=*
```

The SPC 101 will at execution time store the Program Counter in 101 (which is the index 1). The IND1=* will do the same at the assembly time. This will suffice for programs shorter than 2000_8 program steps since the maximum displacement that can be used with any index is 1777_8.

For programs shorter than 4000_8 steps, one can use the following:

```
SPC  101
IND  1=*
IND  2=*+ 2000
SPC     102
INC  2  1777
```

This will prepare index 1 for addressing the first 2000 locations and index 2 for addressing the second 2000 locations.

LRA

The LRA macro generates the address modifier necessary for the correct execution of the INC executed jump. The parameter of the LRA is a defined Label. The assembler computes the difference between the location of the LRA and the location referenced by the Label. If the Label address is less than the LRA address, a LAN Function is generated. If the Label address is greater than the LRA address, a LAP Function is generated. The Function is generated with an I Code of 6 and the L Code containing the absolute value of the computed difference.

The LRA macro must immediately precede the INC Function used to control an indexed loop. Further, control of the loop must be with an initial negative value setting of the index register.

The following example shows the use of the LRA. The program will sort ten consecutively stored values in ascending order. The method of sorting is to compare two consecutively stored values and reverse their locations if the greater value is found in the lesser memory address. Each value is compared with the following value. Nine comparisons are made. The complete comparison cycle is repeated nine times. Index register six controls comparison loop. Index register five controls the number of times the comparison loop is repeated. Index register two contains the memory reference to the ten values. Index register one is set by the IND pseudo and is the base of the relative address of the program.

The following page shows the UNICAP assembly listing of the program on TTY. The first column shows the absolute program location. Following it is the assembled instruction shown in octal command format. The remainder of the line is the original source statement as prepared by the programmer. Notice that certain pseudo instructions in the source language do not generate any instructions. Had there been any errors in the source program, the diagnostics would appear on the extreme left of the listing. (See section , Error Diagnostics, Page 1-18).

```
                      ORG     3000
   03000   0400101    SPC     101
                      IND   1 =*                  INDEX 1 IS SET TO 3001
   03001   1110026    LAN     BFSZ
   03002   1260001    ADD   6 ,1
   03003   0500105    STA   0 105
   03004   1000105    LAP     105         ;LOP1
   03005   0500106    STA   0 106
   03006   1010027    LAP     BUFA
   03007   0500102    STA   0 102             IR TWO IS 3027
   03010   1020001    LAP   2 1           ;LOP2   COMPARE SEQUENTIAL
   03011   1320000    SUB   2 0                   ENTRIES IN BUFFER
   03012   2210020    JAP     MOR                 IF OUT OF ORDER REVERSE
                      * NEXT SIX INSTRUCTIONS REVERSE 2,0 WITH 2,1
   03013   1020000    LAP   2 0
   03014   0500300    STA   0 300
   03015   1020001    LAP   2 001
   03016   0520000    STA   2 000
   03017   1000300    LAP   0 300
   03020   0520001    STA   2 001
   03021   0220001    INC   2 1           ;MOR    INCREMENT IR 2
   03022   1160014    LRA     LOP2                LRA GENERATE LAN 6 14
   03023   0260001    INC   6 1
   03024   1160022    LRA     LOP1                LRA GENERATE LAN 6 22
   03025   0250001    INC   5 1                   REPEAT COMPARISON LOOP
   03026   2770107    JMP   7 107                 EXIT VIA JPR ENTRY
   03027   0000012    10.                 ;BFSZ   BUFFER SIZE
   03030   0011031    BUFF                ;BUFA
                      BSS     BFSZ        ;BUFF   LOCATIONS RESERVED
                      END
```

EJCT

When UNICAP-II will encounter the EJCT pseudo operation, it will stop listing and output a page on the listing device. The normal listing will resume on the next page. There are no arguments with the EJCT pseudo operation.

TEXT

This pseudo operation allows insertion of ASCII coded text in the program. Two ASCII characters will be inserted in one computer word.

The first non-space character following the TEXT will be taken as a delimiting character for this TEXT instruction. The assembler will proceed and store two ASCII characters in computer words until it finds the same character delimiter which would define the end of the text. For example:

```
TEXT \ THE VALUE IS \
TEXT "THE VALUE IS"
```

No carriage return or line feed characters are allowed within the text.

If a label is used, it will have the value of the first location of ASCII characters stored.

The above example will assemble into:

```
06000    1240110    TEXT\THE VALUE IS\ ;MSGE
06001    1050040
06002    1260101
06003    1140125
06004    1050040
06005    1110123
```

The symbol MSGE will have the value of 6000_8.

TSIX

This pseudo operation has identical meaning as TEXT except that ASCII characters are trimmed to 6 bits and stored three per COMP-18 word. Table of the six bit codes can be found in the Appendix 2.

ERROR DIAGNOSTICS

During the assembly process, UNICAP-II does an extensive checking of the source statements. The errors in syntax, illegal expressions, impossible addresses, doubly defined symbols, and other kinds of errors will be found and printed on the assembly listing. Each type of error has been assigned a two letter mnemonic which will appear on the extreme left margin of the listing. The list of error diagnostics and their codes is shown in Appendix 4.

Below is a typical listing of a program with errors introduced to show the error diagnostics.

```
UNICAP-2.0 ASSEMBLER     09/ 11 /70                              PAGE  0001
                         ORG     2000
   02000  0400101        SPC     101
                         IND   1 =*
PS                       IND   1                      NO ARGUMENT WITH IND
PS                       SET     X=                   NO ARGUMENT IN SET PSEUDO
IA 02001  1000000        LAP     2000                 ADDRESS TOO LARGE
IA 02002  1300000        SUB     1737.       ;MAX     ADDRESS TOO LARGE
UD 02003  0500000        STA     YY                   UNDEFINED YY SYMBOL
IA 02004  1000000        LAP     BUFF                 OUT OF RANGE ADDRESS
UD 02005  0500000        STA     YY                   UNDEFINED YY SYMBOL
DD 02006  1060001        LAP   6 1           ;MAX     DOUBLY DEFINED MAX
MN 02007  0000000        LXR     1                    NONEXISTING MNEMONIC
LR 02010  0000000        LRA     BUFF                 LRA ARGUMENT OUT OF RANGE
   02011  0230001        INC   3,1
EX 02012  0000000        -13%JK                       ILLEGAL EXPRESSION
NR 02013  0000000        3FFC                         NO HEX CONSTANTS ALLOWED!
NR 02014  0000000        317B-5                       NO B SCALE WITH OCTAL #
                         ORG     6000
   06000  0000000        0                   ;BUFF
EX 06001  1000000        LAP     6#31                 NO SUCH OPERATION
MN 06002  0000000        STO     200                  NONEXISTING MNEMONIC
UD 06003  0300000        JPR     SUB                  UNDEFINED SUBROUTINE
                         END
```

COMP-18 FUNCTION SET

The following describes the operation of the COMP-18 Function Codes. The Function name and mnemonic and octal code are given for each Function.

Halt, HLT, F=00

Operation of this Function is dependent upon the setting of the HALT ENABLE switch. If the switch is off, the Function is executed as an unconditional jump. If the switch is on, the COMP-18 halts after executing the jump.

Increment Index, INC, F=02

The Variable Field value is added to the contents of the index register specified by the I Field and the result stored in the specified index register. The I Field may have a value, 0 through 7, which will reference memory locations 100_8 through 107_8.

If the result of the computation is negative, a jump will be executed to an address formed by adding the contents of the accumulator to the Program Counter. The new address will replace the previous contents of the Program Counter. Execution of this Function will affect the contents of the memory location specified and may affect the contents of the Program Counter.

Jump Return, JPR, F=03

The contents of the Program Counter are stored in location 107_8. The jump address is placed in the Program Counter and the command stored in that location is executed. Execution of this Function affects the Program Counter and memory location 107_8.

Store Program Counter, SPC, F=04

The contents of the Program Counter are stored in the memory location referenced by the effective address. The Program Counter will contain the address of the SPC command plus one, at the time the Function is executed. For further discussion of this Function, reference the IND macro, page

Execution of this Function affects the referenced memory location.

Read-In, RIN, F=06

Data from the Input/Output Data bus is placed in the accumulator. The effective address is used as the peripheral device code.

Execution of this Function affects the accumulator.

Output, OUT, F=07
The contents of the accumulator are placed on the Input/Output Data bus. The effective address is used as the peripheral device code.

Load Accumulator Positive, LAP, F=10
The operand, taken from memory or the L portion of the command word, is placed in the accumulator in 2's complement form.

Execution of this command replaces the contents of the accumulator.

Add, ADD, F=12
The contents of the accumulator and the operand are added together. The sum replaces the previous contents of the accumulator. If addition results in a carry out of bit position 1, the Exchange Bit is set to a one. If no carry occurs, the Exchange Bit is set to zero. If addition results in the loss of a significant bit from bit position 2, the overflow indicator is set.

Execution of this Function affects the accumulator and may affect the Exchange Bit and overflow indicator.

Subtract, SUB, F=13
The operand is subtracted from the contents of the accumulator and the remainder placed in the accumulator. Subtraction is accomplished by taking the 2's complement and adding. If the subtraction results in a carry out of bit position 1, the Exchange Bit is set to one. If no carry occurs, the Exchange Bit is zero. If the subtraction results in the loss of a significant bit from bit position 2, the overflow indicator is set.

Execution of this Function affects the accumulator and may affect the Exchange Bit and overflow indicator.

Logical AND, AND, F=14
The contents of the accumulator are ANDed bit-by-bit with the operand and the result placed in the accumulator.

Execution of this Function affects the accumulator.

Logical and Inverted, ANI, F=15

This Function is similar to the Logical AND except the operand is converted to 1's complement prior to the AND operation.

Execution of this Function **affects** the accumulator.

Logical OR, LOR, F=16

The contents of the accumulator are ORed, bit-by-bit, with the operand and the result placed in the accumulator.

Execution of this Function **affects** the accumulator.

Exclusive OR, EXO, F=17

The contents of the accumulator are exclusive-ORed with operand and the result placed in the accumulator.

Execution of this Function **affects** the accumulator.

Jump If Accumulator Zero, JAZ, F=20

If all accumulator bits are zero, the jump address is placed in the Program Counter and the command stored in that location is executed.

Jump If Accumulator Not Zero, JNZ, F=21

If any accumulator bit is a one, the jump address is placed in the Program Counter and the command stored in that location is executed.

Execution of this Function **affects** the Program Counter.

Jump If Accumulator Positive, JAP, F=22

If bit position 1 is a zero, then the jump address is placed in the Program Counter and the command stored in that location is executed.

Execution of this Function **affects** the Program Counter.

Jump If Accumulator Negative, JAN, F=23

If bit position 1 is a one, then the jump address is placed in the Program Counter and the command stored in that location is executed.

Execution of this Function **affects** the Program Counter.

Jump If Even Parity, JEP, F=24
If the accumulator contains an even number of one bits, the address will be placed in the Program Counter and the command stored in that location will be executed.

Execution of this Function **affects** the Program Counter.

Jump If Odd Parity, JOP, F=25
If the accumulator contains an odd number of one bits, the jump address will be placed in the Program Counter and the command stored in that location will be executed.

Execution of this Function **affects** the Program Counter.

Jump If Overflow, JOF, F=26
If the overflow indicator is on, the jump address will be placed in the Program Counter and the command stored in that location will be executed. The overflow indicator is reset to zero.

Execution of this Function **affects** the Program Counter and the overflow indicator.

Jump, JMP, F=27
The jump address is placed in the Program Counter and the command stored in that location will be executed. Execution of this Function effects the Program Counter.

Shift Left Arithmetically, SLA, F=30
Accumulator bit positions 2 through 18 are shifted left. Bit position 1 is unchanged. Bits shifted out of bit position 2 are lost. Zeros are shifted into bit position 18. Shifting a one bit of a positive number out of bit position 2 will set the overflow indicator. Shifting a zero bit of a negative number out of bit position 2 will set the overflow indicator.

Execution of this Function **affects** the accumulator.

Shift Left Through Exchange Bit, SLX, F=31
The accumulator bit and Exchange Bit are circulated left. The Exchange Bit enters the accumulator at bit position 18. The contents of bit position 1 enters the Exchange Bit.

Execution of this Function **affects** the accumulator and Exchange Bit.

Shift Left Logical, SLL, F=32
All accumulator bits are shifted left. Bits shifted out of bit position 1 are lost. Zeros are shifted into bit position 18.

Execution of this Function affects the accumulator.

Shift Left End-Around, SLE, F=33
The accumulator bits are circulated left to right. The content of bit position 1 is shifted into bit position 18.

Execution of this Function affects the accumulator.

Shift Right Arithmetically, SRA, F=34
Accumulator bit positions 2 through 18 are shifted right. Bits shifted out of bit position 18 are lost. Bits shifted into bit position 2 repeat the sign bit.

Execution of this Function affects the accumulator.

Shift Right Through Exchange Bit, SRX, F=35
The accumulator bits and Exchange Bits are circulated right. The Exchange Bit enters the accumulator at bit position 1. The contents of bit position 18 enters the Exchange Bit.

Execution of this Function affects the accumulator and Exchange Bit.

Shift Right Logical, SRL, F=36
All accumulator bits are shifted right. Bits shifted out of bit position 18 are lost. Zeros shifted into bit position 1.

Execution of this Function affects the accumulator.

Shift Right End-Around, SRE, F=37
The accumulator bits are circulated right. The content of bit position 18 is shifted into bit position 1.

Execution of this Function affects the accumulator.

UNICAP-II OPERATION

The UNICAP-II assembler will operate in either a one or two pass mode. In the one pass mode, the programmer has the advantage of special pseudo operations for program corrections using source language commands. Use of the one pass mode restricts the length of the source program being assembled since the source program commands, symbol table, object program, and the assembler must all reside in COMP-18 memory.

In the two pass mode, longer source programs can be assembler; however, correction of source program procedural errors must be manually incorporated into the source program sequence.

The following portions of this section present a sequential description for a one pass assembly and checkout of a source program using the ASR-33 for input and output.

SOURCE PROGRAM ASSEMBLY

The first phase of the assembly process is to load the UNICAP-II assembly program into COMP-18 memory. This is accomplished using a binary loader. The binary loader is loaded using the bootstrap loader. The following instructions will load the binary loader program from the ASR-33 paper tape reader.

On the COMP-18:

1) Turn the power on.
2) Press PROGRAM.
3) Turn the HALT ENABLE on.
4) Insure that the COMP-18 is in the idle mode.
5) Press and hold LOAD, momentarily press STEP, release LOAD.
6) Press STEP.
7) Press RUN.

The bootstrap loader is now running. Input is accepted from the ASR-33 paper tape reader or keyboard.

On the ASR-33:

1) Set the power switch to ON LINE.
2) Set the reader control switch to STOP.
3) Push the OFF button on the punch.

If the binary loader and assembler are to be input using the high speed paper tape reader, then, at the keyboard:

4) Depress CNTRL key and strike the R key.

Load the paper tape on the appropriate reader. If the programs are read from the ASR-33 paper tape reader, then:

5) Place the reader control switch to START.

The binary loader program is read into the COMP-18 starting at location 7700_8. The paper tape will read through the ASR-33 reader until tape is exhausted.

To begin execution of the binary loader enter, from the keyboard,

@ 07700

The display indicator will contain 007700 and the COMP-18 will be halted.

To load UNICAP-II, place the assembler program tape in the reader. On the COMP-18:

1) Press ACC NUMERIC.
2) Press STEP.

The program tape is read through the ASR-33 reader until tape is exhausted.

Upon completion of the load process, the COMP-18 will halt if a checksum error were detected. The accumulator will contain the checksum error. In the event of a checksum error, the assembly program should be re-loaded into memory. To re-load UNICAP-II, on the COMP-18:

1) Press RUN.
2) Press and hold LOAD, momentarily press STEP, release LOAD.
3) Press RUN.

At the ASR-33:

1) Place the assembler program tape in the reader.
2) Enter @07700 from the keyboard.

On the COMP-18:

1) Press STEP.

The assembler program tape will be loaded as described above.

To execute UNICAP-II, at the COMP-18:

1) Press RUN.
2) Press and hold START, momentarily press STEP, release START.
3) Press PROGRAM.

The display indicator should contain 000400, the starting location of UNICAP-II.

At the COMP-18:

4) Press STEP.
5) Press RUN.

UNICAP-II is now operating. Prior to loading a source language program for assembly, the input device, output device, and mode of assembly operation must be specified. These selections are specified by:

INP k where k is:
1 - Teletype Keyboard
2 - Teletype Paper Tape Reader
3 - High Speed Paper Tape Reader
4 - Card Reader

OTP n   where n is:
1 - Teletype Paper Tape Punch
2 - High Speed Paper Tape Punch
3 - Card Punch

MOD m   where m is:
0 - one pass
1 - two pass

Enter the following from the ASR-33 keyboard to specify a one pass mode using the ASR-33 paper tape input and output:

```
INP   2
OTP   1
MOD   0
```

To load the source language program tape, at the ASR-33:

1) Place the reader control switch to STOP.
2) Place the source program tape in the reader.
3) Place the reader control switch to START.

The source program tape will pass through the reader until tape is exhausted or the reader is stopped by setting the reader control switch to STOP.

When in the two pass mode, the source program tape is now re-loaded on the paper tape read. The tape is read a second time to complete the assembly.

In the one pass mode, the source program is now in memory and the following describes the assembly and checkout operations that may be performed.

CHECKOUT PSEUDO OPERATIONS

These pseudo operations provide a means of correcting source language commands without having to re-punch and re-load the source program. To use these pseudo operations, UNICAP-II must be in the one pass mode and the source program, assembler, and assembly generated symbol tables must be in memory.

These pseudo operations and parameters are entered from the ASR-33 keyboard.

The following describes the usage of each Checkout Pseudo Operation.

STO

This operation provides for adding a new command or value to a specific memory location. The memory location may be specified as an octal location or a symbolic address. Locations may be specified using arithmetic operations. The format is:

STO (memory reference) (space) (new item)

The parentheses are for illustrative purposes only. For example, assume LAP 6 12 ;XTA is in error and should be LAN 6 15 ;XTA, then the following would be entered from the keyboard:

STO XTA LAN 6 15 ;XTA

If the Command, SUB CONS, located four locations from XTA, were to be changed to, LOR CONS, then the following would be typed:

STO XTA - 4 LOR CONS

INS

This operation provides for inserting several commands into the program. Memory reference is to the location immediately preceding the location that is to be changed. Following the INS pseudo operation are the UNICAP-II source language statements in standard format. The pseudo operation, INS $END, is used to terminate the source language command string.

An example of INS usage is:

INS XXX
LAP 6 10
STA 0 105
INS $END

The previous contents of XXX, plus one, and all locations following, would be moved two memory locations to allow for insertion of the LAP and STA commands.

DEL

This pseudo operation provides for the removal of the contents of a specified number of locations. If a single value or command is to be removed, the format is:

DEL (address)

The parentheses are for illustrative purposes only. If consecutive commands are to be removed, the format is:

DEL (address) , (address)

The locations following the removed quantities are moved to form a continuous command string.

For example,

DEL XXX ,XXX+10

would result in the removal of 10_8 words starting at location XXX.

MOV

This pseudo operation will move a block of commands or values from one memory location to another. Three addresses are given as the parameter of this pseudo operation: starting address, ending address, new address. The format is:

MOV (address) , (address), (address)

For example,

MOV LOP1 ,LOP1+10, LOP7

would result in the 10_8 commands starting in location LOP1 to be moved to the starting location, LOP7.

OBJECT PROGRAM

Once the source language commands are considered correct for assembly, the pseudo operation, COM , is entered from the keyboard. UNICAP-II will assemble the source program commands into COMP-18 commands. If assembly errors are detected, the COMP-18 will halt with the error message number in the accumulator. The source language command being assembled at the time of error detection and the mnemonic code for the error will be printed on the Teleprinter. The error messages are given in Appendix 4.

To continue the assembly process after an error message, press STEP. If the error is due to the program and symbol tables exceeding memory then the assembly should be aborted.

Upon completion of the assembly, the names of the programs referenced by the CALL macro will be printed on the Teleprinter. To load these programs, place the appropriate program tape in the reader, place the reader control to START, and press STEP.

After the assembly process is complete, the programmer may use the Checkout Pseudo Operations to correct any assembly errors.

If the program is to be debugged while resident in memory with UNICAP-II, then a memory map showing locations occupied by the object program, assembler, source program, symbol table, and loader is necessary. A memory map is printed on the Teleprinter after the pseudo operation, MAP, is entered from the keyboard.

Entry to the object program may be made by entering JUM, and the starting address, on the keyboard. The COMP-18 will halt with the address entered displayed in the indicators.

If program errors are detected during debugging, the Checkout Pseudo Operations may be used for corrections. Use of the Checkout Pseudo Operations requires a re-entry to UNICAP-II which is accomplished by taking the COMP-18 out of the run mode and, at the COMP-18:

1) Press and hold START, momentarily press STEP, release START.
2) Press STEP.
3) Press RUN.

The Checkout Pseudo Operations may now be used as previously described. After the source program has been corrected, the assembly and debugging may be repeated.

To produce a binary tape of the object program, on the ASR-33 paper tape punch:

1) Press the ON button on the paper tape punch unit.
2) Enter, BIN, on the keyboard.

After the tape has been punched, turn the punch unit off.

A program listing is obtained by entering the LST pseudo operation on the keyboard. Two types of listing can be obtained. A 0 following LST will result in a listing of commands in octal command format and the associated memory locations. A 1 following LST will provide the same listing as, LST 0, but will include the source commands as well.

If an object program tape is to be loaded, with UNICAP-II in memory and operating, the program tape is mounted on the ASR-33 paper tape reader and LBN is entered on the keyboard.

Appendix 5 shows the format of the object tape.
Appendix 8 shows the Teleprinter listing of source commands, assembled object program, and UNICAP-II control instructions.

RESERVED MEMORY LOCATIONS

Certain memory locations are reserved for UNICAP-II operations. Some of the locations are reserved for programming conventions, others contain the assembly program. Appendix 6 shows the reserved memory locations for UNICAP-II operation.

FLOATING POINT OPERATIONS

UNICAP-II provides subroutines to perform floating point arithmetic operations. The command format is identical to other source language commands. A three alphabetic mnemonic is used to reference the operation. A symbolic address is used to specify the operand. A pseudo accumulator has been defined as memory locations 110_8 and 111_8. Locations 112_8 and 113_8 are used by the floating point routines for temporary operand storage.

A single precision floating point value is stored in two COMP-18 words as,

1	2 — 8	9 — 18
S	Exponent	Mantissa
1 — 18		
Mantissa		

where, the mantissa is a 29 bit signed value and the exponent is a 7 bit signed value. The sign of the mantissa is in bit position 1 of the first word. The sign of the exponent is in bit position 2 of the second word.

UNICAP-II will assign two consecutive locations for all floating point quantities. A symbolic reference to a floating point quantity is assigned the memory address of the first word of the floating point value.

UNICAP-II assigns an octal value to the floating point mnemonic. In general, the assigned octal code will equal the COMP-18 Function Code for a similar operation. The command, FEN, is given prior to executing floating point operations. The FEN command signifies that the following Functions are to be interpreted as floating point operations. The command, FXT, is used to terminate the execution of Functions in the floating point mode.

Appendix 7 contains a list of the floating point mnemonics and a description of their operation.

The following is an example of a floating program which will compute:

$$Z=\sqrt{X^2+Y^2}$$

```
*   COMPUTE   Z=SQRT (X**2+Y**2)
ORG     2000
SPC     101
IND     1=*
FENT                                    ENTER FLOATING POINT
FLAP    X                               INTERRPRETIVE MODE
FMPY    X                               COMPUTE X SQUARED
FSTA    Z
FLAP    Y
FMPY    Y                               COMPUTE Y SQUARED
FADD    Z
FSQT                                    TAKE SQUARE ROOT OF SUM
FSTA    Z                               X SQUARED PLUS Y SQUARED
FEXT                                    EXIT FLOATING POINT
BSS     2             ;X
BSS     2             ;Y
BSS     2             ;Z
END
```

PROGRAMMING EXAMPLES

This section shows examples of programming techniques for the COMP-18 using UNICAP-II.

ADDRESSING

The effective address of a COMP-18 command is created as follows:

I CODE	EFFECTIVE ADDRESS
0	The L Code is the effective address
1 - 6	The contents of the specified index register are added to the L Code to form the effective address.
7	The L Code is used to reference a memory location whose contents are the effective address.

The use of the effective address varies with the Function Code. For arithmetic, logic, and jump Functions, the effective address is a memory address.

Since the L Code is **ten** bits, the maximum directly addressable location is 1777_8. Referencing memory locations above that address requires an index or indirect address.

UNICAP-II computes the L Code value of symbolic references and sets the I Code to index register one. Reference is made to a buffer area outside the 2000_8 **relative** addressed block. This address is stored in a location within the 2000_8 **block** by using the miltiple ORG'ed assemblies. The program loads a DMA register with the buffer address and the number of words to transfer.

```
                ORG     3000
03000  0400101  SPC     101
                IND     1=*                   INDEX REGISTER ONE
03001  1010004  LAP     BUFA                  CONTAINS REFERENCE ADDRESS
                *  OUTPUT STARTING ADDRESS TO DMA
                SET     XXX=170
03002  0700170  OUT   0 XXX
03003  1110005  LAN     CONS
                * OUTPUT NUMBER OF WORDS TO TRANSFER TO DMA
                SET     YYY=160
03004  0700160  OUT   0 YYY
                *  ADDRESS OF BUF (=6000) STORED IN 3005
03005  0030000  BUF                   ;BUFA
                * NUMBER OF WORDS TO TRANSFER IS 256.
03006  0000400  256.                  ;CONS
                ORG     6000
                BSS     256.          ;BUF
                END
```

The following example sets the interrupt processing subroutine address in location, 000_8, using multiple ORG'ed assemblies.

```
                *  AN INDIRECT JUMP IS ASSEMBLED INTO LOCATION 0001
                ORG     1
00001  2770200  JMP   7 INTA
                * LOCATION 200 IS ASSEMBLED TO CONTAIN THE ADDRESS
                * OF THE INTERRUPT PROGRAMMING, INT
                ORG     200
00200  0030001  INT                   ;INTA
00201  0000000  0                     ;AREG
                ORG     6000
06000  0400101  SPC     101
                IND     1=*
06001  0500201  STA     AREG          ;INT
                * THE FOLLOWING WOULD BE THE BODY OF THE INTERRUPT
                * PROCESSING PROGRAM
                END
```

INDEXING

The operation of the index register is that of address modification and control of a repeated programming loop.

In the following example, index register 1 is used for address modification, index register 3 controls the loop. The program will store index register 4,5,6, and memory location 107_8 in four reserved locations, labeled IRS.

```
                ORG     3000
03000  0400101  SPC     101
                IND     1=*
03001  1160004  LAN   6 4                   IR 3=-4
03002  0500103  STA     103
03003  1010012  LAP     IRSA
03004  0500102  STA     102                 IR 2=3300
03005  1030110  LAP   3 110         ;LOOP
03006  0520001  STA   2 1
03007  0220001  INC   2 1
                * MACRO WILL SET ACCUMULATOR FOR JUMP TO LOOP
                * UPON EXECUTION OF INC
03010  1160005  LRA     LOOP
03011  0230001  INC   3 1
03012  2770107  JMP   7 107
03013  0011014  IRS                 ;IRSA
                BSS     4           ;IRS
                END
```

In the following example, the Interrupt Status word will be read into the accumulator and shifted left until a one bit is in the sign position. When the one bit is in the sign position, index register 4 will contain the bit location of the first one bit in the Interrupt Status word.

```
                ORG     3500
03500  0400101  SPC     101
                IND     1=*
03501  1060000  LAP  6 0
                * INDEX REGISTER 3 IS SET TO ZERO
03502  0500104  STA  0 104
                * INTERRUPT STATUS WORD IS READ INTO ACCUMULATOR
03503  0600020  RIN  0 020
03504  2310007  JAN     *+4
                * JUMP WHEN ONE IS IN BIT POSITION 1
03505  3200001  SLL     1
03506  0240001  INC  4 1
03507  2710003  JMP     *-3                 REPEAT TEST
```

The value in Index register 4 could be used to reference a table of interrupt processing subroutine addresses and jump to that routine, as follows:

```
03510  1010013  LAP     ADT1
03511  1200104  ADD     104
03512  0500200  STA     200
03513  2770200  JMP  7 200
                * THE FOLLOWING LOCATIONS CONTAIN STARTING ADDRESS
                * OF SIXTEEN INTERRUPT PROCESSING SUBROUTINES
                SET     ADT1=*
                END
```

MULTIPLY/DIVIDE SUBROUTINE

When the hardware Multiply, Divide, Square Root (MDSR) option is not available, the software Multiply and Divide subroutines are used.

The Multiply subroutine uses the contents of location 110_8 as the multiplicant and the contents of the accumulator as the multiplier. Upon completion of the multiply, the most significant bits of the product are in location 110_8, the least significant bits are in the accumulator. The sign of the product is in the sign bit of location 110_8.

The starting address of the Multiply subroutine is stored in location 114_8. Use of the Multiply subroutine is shown in the following example.

```
                        ORG    3000
03000    0400101        SPC    101
                        IND    1=*
                        * THE CONTENTS OF MULT WILL BE MULTIPLIED
                        * BY THE CONTENTS OF MULR
03001    1010007        LAP    MULT
03002    0500110        STA    110
03003    1010010        LAP    MULR
03004    2770114        JMP  7 114                 JUMP TO MULTIPLY SUBROUTINE
                        * THE LEAST SIGNIFICANT BITS OF THE PRODUCT
                        * ARE STORED IN THE LSB  THE MOST SIGNIFICANT BITS OF
                        * THE PRODUCT ARE STORED IN THE MSB
03005    0510011        STA    LSB
03006    1000110        LAP    110
03007    0510012        STA    MSB
                        * THE PROGRAM WOULD CONTINUE IN
                        * THE FOLLOWING LOCATIONS
                        BSS    1           ;MULT
                        BSS    1           ;MULR
                        BSS    1           ;LSB
                        BSS    1           ;MSB
                        END
```

The Divide subroutine uses the contents of 110_8 and 111_8 as a signed, 35 bit dividend and the contents of the accumulator as the signed, 17 bit divisor. After the divide is executed, the signed quotient is in the accumulator.

The starting address of the Divide subroutine is stored in location 115_8.

Use of the Divide subroutine is shown in the following example.

```
                     ORG    3000
03000    0400101     SPC    101
                     IND    1=*
                     * A 15 BIT , SIGNED DIVIDEND IS STORED IN THE MOST
                     * SIGNIFICANT HALF OF PSEUDO ACCUMULATOR
                     * THE LEAST SIGNIFICANT HALF OF THE WORD IS ZERO
03001    1010007     LAP    DIVD
03002    0500110     STA    110
03003    1010010     LAP    DIVS
03004    2770115     JMP  7 115
03005    0510011     STA    QT
03006    1000110     LAP    110
03007    0510012     STA    REM
                     * PROGRAM WOULD CONTINUE IN THE FOLLOWING LOCATIONS
                     BSS    1           ;DIVD
                     BSS    1           ;DIVS
                     BSS    1           ;QT
                     BSS    1           ;REM
                     END
```

$SAVE/ $UNSV SUBROUTINES

These subroutines will store and re-store the contents of the accumulator and all index registers. The routines are used by closed subroutines which will alter the contents of working registers during execution.

Location 00004_8 contains the address of $SAVE location 00005_8 contains the address of $UNSV. Location 00007_8 contains the starting location of the subroutine prior to entering $SAVE. The contents of accumulator are saved in location 000006 by the $SAVE. routine.

The following is an example of $SAVE/$UNSV usage:

```
                SET    $SAVE=4
                SET    $UNSV=5
                SET    $SBST=7
                ORG    3000
03000  0400101  SPC    101
                IND    1=*
03001  2710005  JMP    SUB          ;X
                *THE PROGRAM WOULD CONTINUE
                *IN THE FOLLOWING LOCATIONS
03002  0000000  0
03003  0000000  0
03004  0000000  0
03005  0310005  JPR    SUB
                *WHEN SUBROUTINE IS ENTERED
                *LINK TO SAVE IS EXECUTED
03006  0400007  SPC    $SBST        ;SUB
03007  0370004  JPR  7 $SAVE
                BSS    8.                  STORAGE FOR INDEXES
                *THE SUBROUTINE WOULD BE
                *IN THE FOLLOWING LOCATIONS
03020  0000000  0
03021  0000000  0
03022  0000000  0
03023  0000000  0
03024  0000000  0
03025  2770005  JMP  7 $UNSV               LINKAGE TO UNSAVE
                END
```

INTERRUPT PROCESSING

Appendix 7 contains the source and object program listings of a program for processing multiple interrupts. For each interrupt, the working registers and indicator bits are stored in a variable length program buffer. When the buffer is full, interrupts will be disabled until processing progresses to the point that buffer space is available to store the registers. The number of locations reserved for the buffer determines the number of interrupts within interrupts which may be processed.

The registers and indicators are stored in the buffer, labeled BUF, in the following order:

Overflow Indicator
Exchange Bit
Index Register 1
Index Register 2
Index Register 3
Index Register 4
Index Register 5
Index Register 6
Location 107_8
Location 100_8
Accumulator
Interrupt Status Word
Location of first one bit in Interrupt Status Word

Location 000_8 must be initialized so that when an interrupt occurs, enter is made to the routine INTI. This routine saves all index registers, accumulator, overflow indicator, Exchange Bit, contents of location 100_8, and the Interrupt Status. The routine interrogates the Interrupt Status word, starting with bit position 1, for the line causing the interrupt. From an ordered table, CON, the address of the appropriate interrupt servicing subroutine will be found. Within the same table is the enable/disable mask that is to be in effect while the interrupt servicing routine is being executed. The last commands of INTI output the enable/disable mask to the Interrupt Mask register and jump to the interrupt servicing routine.

The interrupt servicing routine exits to the subroutine INTO. In order that this routine not be interrupts,the interrupts are disable. This routine restores the working registers and indicator bits and sets the Program Counter to the location that would have been executed had the interrupt not occurred. The Interrupt Status word is checked to see if an interrupt occurred during the execution of INTO. If an interrupt did occur, the disable mask is left in place and entry is made to INTI. When no interrupt occurs during the INTO execution, control is returned to the last interrupted program.

In the sample program, interrupts are processed on a first-in-last-out basis. To change to a first-in-first-out basis, requires that INTI and INTO not use a common BUF address point for storing and retrieving working registers.

APPENDIXES

ASCII TELETYPE CODE

TELETYPE CHARACTER	OCTAL CODE
SPECIAL CHARACTERS	
RUBOUT	377
NUL (CTRL & SHIFT P)	000
SOM (CTRL A)	201
EOA (CTRL B)	202
EOM (CTRL C)	003
EOT (CTRL D)	204
WRU (CTRL E)	005
RU (CTRL F)	006
BEL (CTRL G)	207
FE (CTRL H)	210
H TAB (CTRL I)	011
LINE FEED (or CTRL J)	012
V TAB (CTRL K)	213
FORM (CTRL L)	014
RETURN (or CTRL M)	215
SO (CTRL N)	216
SI (CTRL O)	017
DCO (CTRL P)	
X-ON (CTRL Q)	021
TAPE AUX	
ON (CTRL R)	022
X-OFF (CTRL S)	223
TAPE AUX	
OFF (CTRL T)	024
ERROR (CTRL U)	225
SYNC (CTRL V)	226
LEM (CTRL W)	027
SO (CTRL X)	030
SI (CTRL Y)	231
S2 (CTRL Z)	232
S3 (CTRL K)	033
S4 (CTRL L)	234
S5 (CTRL M)	035
S6 (CTRL N)	036
S7 (CTRL O)	237

APPENDIX 2

REPRESENTATIONS OF GRAPHIC CODES

Character	7 Bit ASCII	6 Bit UniCap CODE	029 PUNCHED CARD CODES	Character	7 Bit ASCII	6 Bit UniCap CODE	029 PUNCHED CARD CODES
SPACE	040	40	No Punch	@	100	00	8,4
!	041	41	11,8,2	A	101	01	12,1
"	042	42	8,7	B	102	02	12,2
#	043	43	8,3	C	103	03	12,3
$	044	44	11,8,3	D	104	04	12,4
%	045	45	0,8,4	E	105	05	12,5
&	046	46	12	F	106	06	12,6
'	047	47	8,5	G	107	07	12,7
(	050	50	12,8,5	H	110	10	12,8
)	051	51	11,8,5	I	111	11	12,9
*	052	52	11,8,4	J	112	12	11,1
+	053	53	12,8,6	K	113	13	11,2
,	054	54	0,8,3	L	114	14	11,3
-	055	55	11	M	115	15	11,4
.	056	56	12,8,3	N	116	16	11,5
/	057	57	0,1	O	117	17	11,6
0	060	60	0	P	120	20	11,7
1	061	61	1	Q	121	21	11,8
2	062	62	2	R	122	22	11,9
3	063	63	3	S	123	23	0,2
4	064	64	4	T	124	24	0,3
5	065	65	5	U	125	25	0,4
6	066	66	6	V	126	26	0,5
7	067	67	7	W	127	27	0,6
8	070	70	8	X	130	30	0,7
9	071	71	9	Y	131	31	0,8
:	072	72	8,2	Z	132	32	0,9
;	073	73	11,8,6	[	133	33	12,8,2
<	074	74	12,8,4	\	134	34	11,8,7
=	075	75	8,6	]	135	35	0,8,2
>	076	76	0,8,6	↑	136	36	12,8,7
?	077	77	0,8,7	←	137	37	0,8,5

Note: Teletype character is the 7 bit ASCII code with odd parity character added as the 8th bit.

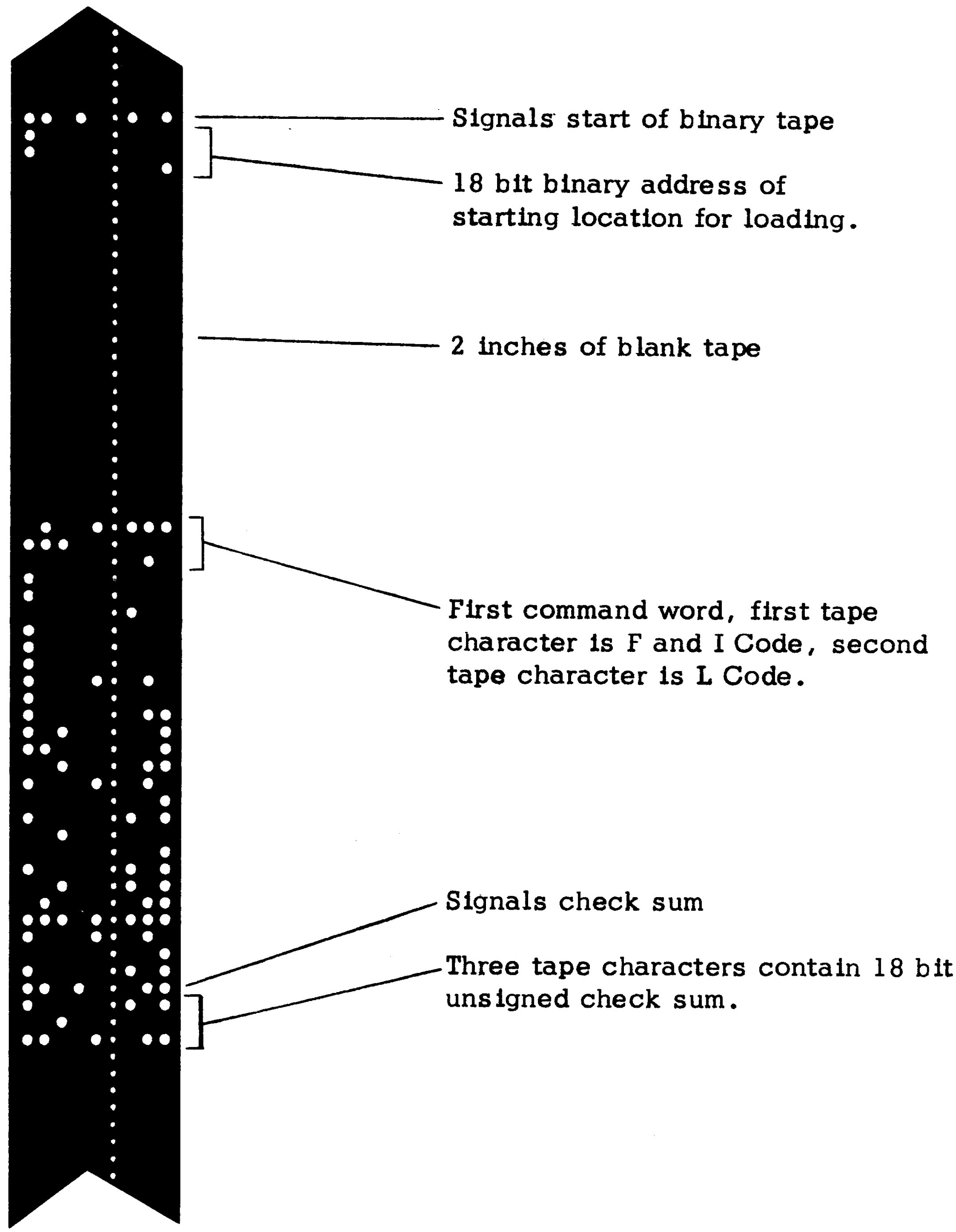
APPENDIX 3

MNEMONIC CODE	DESCRIPTION
COMP-18 FUNCTION CODES	
HLT	Jump and if halt switch on, halt.
INC	Increment index register
JPR	Store Program Counter in 107_8 and jump.
SPC	Store the Program Counter.
STA	Store the contents of the accumulator.
RIN	Read data into the accumulator.
OUT	Output data from the accumulator.
LAP	Load accumulator positive.
LAN	Load accumulator with 2's complement.
ADD	Add operand to contents of accumulator.
SUB	Subtract operand from contents of accumulator .
AND	AND operand and accumulator.
ANI	AND operand and accumulator and take 1's complement.
LOR	Logical OR operand and accumulator.
EXO	Exclusive OR operand and accumulator.
JAZ	Jump if accumulator is zero.
JNZ	Jump if accumulator is not zero.
JAP	Jump if accumulator is positive.
JAN	Jump if accumulator is negative.
JEP	Jump if even parity in accumulator.
JOP	Jump if odd parity in accumulator
JOF	Jump if overflow indicator is set.
JMP	Jump unconditionally.
SLA	Shift accumulator left, arithmetic.
SLX	Shift accumulator through exchange bit, right.
SLL	Shift accumulator left, logically.
SLE	Circulate accumulator left.
SRA	Shift accumulator right, logically.
SRX	Shift accumulator through exchange bit, left.
SRL	Shift accumulator right, logically.
SRE	Circulate accumulator right.

ASSEMBLY ERROR MESSAGES

MNEMONIC CODE	MEANING
MN	Unrecognizable mnemonic Function Code
LN	The program and symbol table exceed memory capacity segment the program.
DD	Doubly defined symbol in the label field.
UD	Undefined symbol used in the variable field.
IA	Symbol or address used in the variable field can not be computed. Probably it is outside of range set by IND pseudo.
LR	The argument in the LRA macro is further than 1777_8 locations away from the present program counter, or it is an undefined symbol.
NR	Numeric error. Either octal or decimal format rules are violated.
EX	Illegal expression. At least one of the rules of expression syntax is violated.
PS	Pseudo instruction syntax error.

OBJECT PROGRAM TAPE



UNICAP-II RESERVED MEMORY LOCATIONS

LOCATION (OCTAL)	USAGE
001	Reserved for hardware operation.
004	Contains the starting address of SAVE.
005	Contains the starting address of UNSAVE.
006	Contains the starting address of the Floating Point Package.
007	Contains the contents of the Accumulator for SAVE.
010	Contains the format for printing FLOATING POINT values.
100 through 107	Reserved for hardware operations.
110 111	Two locations used as a pseudo accumulator, for Floating Point operations.
112 113	Used with the following location to contain a floating point operand.
114	Contains the starting address of the fixed point multiply program.
115	Contains the starting address of the fixed point divide program.
7700 through 7743	Location of object program loader.

FLOATING POINT COMMANDS

FLOATING POINT MNEMONIC CODE	FUNCTION
FENT	Entry to the floating point interpretative mode.
FLAP	Load an operand into the floating point accumulator, location 110 and 111.
FSTA	Store the floating accumulator into the specified operand address. The specified location and the following location will be filled.
FNOR	The contents of the floating accumulator will be normalized.
FADD	The specified operand will be added to the floating accumulator.
FSUB	The specified operand will be subtracted from the contents of the floating accumulator.
FMPY	The specified operand will be multiplied by the contents of the floating accumulator.
FDIV	The contents of the floating accumulator will be divided by specified operand.
FEXT	Exit from the floating point interpretative mode.
FFLT	The specified operand will be converted to a floating point value.
FSIN	The sine of the value in the floating accumulator, in radians, is placed in the floating accumulator.
FCOS	The cosine of the value in the floating accumulator, in radians, is placed in the floating accumulator.
FATN	The arc tangent of the value in the floating accumulator, in radians, is placed in the floating accumulator.
FEXP	The floating exponent of the value in the floating accumulator is placed in the floating accumulator.
FLOG	The floating natural logarithm of the value in the floating accumulator is placed in the floating accumulator.
FSQT	The square root of the value in the floating accumulator is placed in the floating accumulator.

INTERRUPT PROCESSING PROGRAM

UNICAP-II CONTROL INSTRUCTIONS

```
INP 2
OTP 1
MOD 0
```

SOURCE PROGRAM LISTING

```
NEW
ORG     2500
SPC     101
IND     1=*                     SET REFERENCE ADDRESS
STA     200                     STORE ACC
LAP     101
STA     201                     STORE IR1
LAP     102
STA     202                     STORE IR2
LAP     203
STA     102                     IR2 CONTAINS BUFFER ADDRESS
SUB     CEB                     TEST TO STORE REGISTERS
JAZ     INA                     ROOM FOR REGISTERS
LAP     201                     NO ROOM FOR REGISTERS
STA     101                     CAN NOT PROCESS
LAP     202                     RESTORE REGISTERS AND EXIT
STA     102                     TO POINT OF INTERRUPTION
LAP     200
JMP   7 107
JOF     *+2          ;INA       ACC IS NEGATIVE
LAP   6 0                       STORE 0 IF OVERFLOW IS OFF
STA   2 0                       STORE NEG
STA   2 0                       STORE NEG VALUE IF OF IS ON
SLX     1                       EX BIT TO ACC BIT 16
STA   2 1
LAP     201                     IR1 TO BUF+2
STA   2 2
LAP     202                     IR2 TO BUF+3
STA   2 ,3
LAP   0 103                     IR3 TO BUF=4
STA   2 4
LAN   6 4                       IR3=-4
STA     103                     LOOP TO STORE LOC 104-107
LAP   3 110                     TO BUF+5 TO BUF+7
STA   2 4            ;LOOP
INC   2 1
LRA     LOOP                    LRA MACRO
INC   3 1
```

```
LAP      100                   STORE CONTENTS OF 100
STA   2 5                      IN BUF+11
LAP      200                   STORE ORIGINAL CONTENTS
STA   2 6                      OF ACC IN BUF+12
INC   2 7                      SET IR2 TO BUF+13
SET      OLD=20
RIN      OLD                   INPUT INTERRUPT LINES
SLL      1          ;INTA      STATUS WORD LINES1-16
JAN      *+3
INC   3 1
JMP      *+3
STA   2 0                      STORE STATUS WORD
INC   2 1
LAP      103                   STORE NUMBER PLACES SHIFTED
STA   2 0
INC   2 1
LAP      102
STA      203                   STORE BUFFER ADDRESS
SLL      1
ADD      CONA
STA      103                   IR3 SET TO ADDRESS OF
LAP   3 0                      TABLE OF INTERRUPT
STA      201                   ADDRESSES
LAP   3 1
OUT      020                   OUTPUT ENABLE/DISABLE MASK
JMP   7 201                    EXIT TO INTERRUPT ROUTINE
RIN      20         ;INTO      ENTRY FROM INTERRUPT
STA      205                   STORE INT STATUS WORD
LAP   6 ,0                     DISABLE INTERRUPT
OUT      20
LAP      204                   SET IR2 TO REFERENCE ADDRESS
STA      101
LAP      203                   BUFFER ADDRESS
SUB   6 ,1
STA      102                   IR2 TO BUFFER ADDRESS
LAP   2 ,0
STA      103                   RESET IR'S TO INT1
LAP      102                   VALUES AT INTA
SUB   6 ,1
STA      102
SUB   6 ,13
STA      203
LAP   2 ,0
JNZ      INTA                  IF MULTIPLE INTERRUPTS GO
LAP   2 ,0                     TO INTERRUPT PROCESSOR
JOF      *+1                   RESTORE REGISTER PRIOR
JAZ      *+3                   TO EXECUTE FROM INT ROUTINE
LAM   6 377                    SET OVERFLOW
SLA      8.
LAP   2 ,1
SRR      1                     SET EX BIT
```

```
LAN   6 ,4
STA     103                           IR3=-4
LAP   2 ,2                            IR1 TO 201
STA     201
LAP   2 ,3                            IR2 TO 202
STA     202
LAP   2 ,5                   ;LOP1    IR4 IR5 IR6 IR7
STA   3 110
INC   2 ,1
LRA     LOP1
INC   3 ,1
LAP   2 ,5                            (100)
STA   0 ,100                          IR1 TO 200
LAP   2 ,6
STA     200
LAP   0 202
STA     102
LAP   2 ,0                            IR3
STA     103
RIN     20                            TEST IF ANY INT'S SINCE
SUB     205                           DISABLE
JAZ     *+2
JMP   7 204
LAP     201
STA     101
LAN   6 ,1
OUT     20
LAP     200                           RESTORE ACC
JMP   7 100
CON                          ;CONA
BUF+107                      ;CEB
BSS     40.                  ;CON
BSS     170                  ;BUF
ORG     204
END
```

```
UNICAP-2.0 ASSEMBLER     09/ 14 /70          UNICAP   PAGE  0001
                         NEW
                         ORG     2500
    02500   0400101      SPC     101
                         IND     1=*                 SET REFERENCE ADDRESS
    02501   0500200      STA     200                 STORE ACC
    02502   1000101      LAP     101
    02503   0500201      STA     201                 STORE IR1
    02504   1000102      LAP     102
    02505   0500202      STA     202                 STORE IR2
    02506   1000203      LAP     203
    02507   0500102      STA     102                 IR2 CONTAINS BUFFER ADDRESS
    02510   1310162      SUB     CEB                 TEST TO STORE REGISTERS
    02511   2010017      JAZ     INA                 ROOM FOR REGISTERS
    02512   1000201      LAP     201                 NO ROOM FOR REGISTERS
    02513   0500101      STA     101                 CAN NOT PROCESS
    02514   1000202      LAP     202                 RESTORE REGISTERS AND EXIT
    02515   0500102      STA     102                 TO POINT OF INTERRUPTION
    02516   1000200      LAP     200
    02517   2770107      JMP   7 107
    02520   2610021      JOF     *+2         ;INA    ACC IS NEGATIVE
    02521   1060000      LAP   6 0                   STORE 0 IF OVERFLOW IS OFF
    02522   0520000      STA   2 0                   STORE NEG
    02523   0520000      STA   2 0                   STORE NEG VALUE IF OF IS ON
    02524   3100001      SLX     1                   EX BIT TO ACC BIT 16
    02525   0520001      STA   2 1
    02526   1000201      LAP     201                 IR1 TO BUF+2
    02527   0520002      STA   2 2
    02530   1000202      LAP     202                 IR2 TO BUF+3
    02531   0520003      STA   2 ,3
    02532   1000103      LAP   0 103                 IR3 TO BUF+4
    02533   0520004      STA   2 4
    02534   1160004      LAN   6 4                   IR3=-4
    02535   0500103      STA     103                 LOOP TO STORE LOC 104-107
    02536   1030110      LAP   3 110                 TO BUF+5 TO BUF+7
    02537   0520004      STA   2 4           ;LOOP
    02540   0220001      INC   2 1
    02541   1160004      LRA     LOOP                LRA MACRO
    02542   0230001      INC   3 1
    02543   1000100      LAP     100                 STORE CONTENTS OF 100
    02544   0520005      STA   2 5                   IN BUF+11
    02545   1000200      LAP     200                 STORE ORIGINAL CONTENTS
    02546   0520006      STA   2 6                   OF ACC IN BUF+12
    02547   0220007      INC   2 7                   SET IR2 TO BUF+13
                         SET     OLD=20
    02550   0600020      RIN     OLD                 INPUT INTERRUPT LINES
    02551   3200001      SLL     1           ;INTA   STATUS WORD LINES1-16
    02552   2310054      JAN     *+3
    02553   0230001      INC   3 1
    02554   2710056      JMP     *+3
```

```
UNICAP-2.0 ASSEMBLER      09/ 14 /70          UNICAP  PAGE  0002
  02555  0520000  STA  2 0                    STORE STATUS WORD
  02556  0220001  INC  2 1
  02557  1000103  LAP    103                  STORE NUMBER PLACES SHIFTED
  02560  0520000  STA  2 0
  02561  0220001  INC  2 1
  02562  1000102  LAP    102
  02563  0500203  STA    203                  STORE BUFFER ADDRESS
  02564  3200001  SLL    1
  02565  1210161  ADD    CONA
  02566  0500103  STA    103                  IR3 SET TO ADDRESS OF
  02567  1030000  LAP  3 0                    TABLE OF INTERRUPT
  02570  0500201  STA    201                  ADDRESSES
  02571  1030001  LAP  3 1
  02572  0700020  OUT    020                  OUTPUT ENABLE/DISABLE MASK
  02573  2770201  JMP  7 201                  EXIT TO INTERRUPT ROUTINE
  02574  0600020  RIN    20           ;INTO   ENTRY FROM INTERRUPT
  02575  0500205  STA    205                  STORE INT STATUS WORD
  02576  1060000  LAP  6 ,0                   DISABLE INTERRUPT
  02577  0700020  OUT    20
  02600  1000204  LAP    204                  SET IR2 TO REFERENCE ADDRESS
  02601  0500101  STA    101
  02602  1000203  LAP    203                  BUFFER ADDRESS
  02603  1360001  SUB  6 ,1
  02604  0500102  STA    102                  IR2 TO BUFFER ADDRESS
  02605  1020000  LAP  2 ,0
  02606  0500103  STA    103                  RESET IR'S TO INT1
  02607  1000102  LAP    102                  VALUES AT INTA
  02610  1360001  SUB  6 ,1
  02611  0500102  STA    102
  02612  1360013  SUB  6 ,13
  02613  0500203  STA    203
  02614  1020000  LAP  2 ,0
  02615  2110050  JNZ    INTA                 IF MULTIPLE INTERRUPTS GO
  02616  1020000  LAP  2 ,0                   TO INTERRUPT PROCESSOR
  02617  2610117  JOF    *+1                  RESTORE REGISTER PRIOR
  02620  2010122  JAZ    *+3                  TO EXECUTE FROM INT ROUTINE
  02621  1160377  LAN  6 377                  SET OVERFLOW
  02622  3000010  SLA    8.
  02623  1020001  LAP  2 ,1
  02624  3500001  SRX    1                    SET EX BIT
  02625  1160004  LAN  6 ,4
  02626  0500103  STA    103                  IR3=-4
  02627  1020002  LAP  2 ,2                   IR1 TO 201
  02630  0500201  STA    201
  02631  1020003  LAP  2 ,3                   IR2 TO 202
  02632  0500202  STA    202
  02633  1020005  LAP  2 ,5           ;LOP1   IR4 IR5 IR6 IR7
  02634  0530110  STA  3 110
  02635  0220001  INC  2 ,1
```

```
UNICAP-2.0 ASSEMBLER     09/ 14 /70                    UNICAP  PAGE  0003
   02636  1160005  LRA     LOP1
   02637  0230001  INC   3 ,1
   02640  1020005  LAP   2 ,5                          (100)
   02641  0500100  STA   0 ,100                        IR1 TO 200
   02642  1020006  LAP   2 ,6
   02643  0500200  STA     200
   02644  1000202  LAP   0 202
   02645  0500102  STA     102
   02646  1020000  LAP   2 ,0                          IR3
   02647  0500103  STA     103
   02650  0600020  RIN     20                          TEST IF ANY INT'S SINCE
   02651  1300205  SUB     205                         DISABLE
   02652  2010153  JAZ     *+2
   02653  2770204  JMP   7 204
   02654  1000201  LAP     201
   02655  0500101  STA     101
   02656  1160001  LAN   6 ,1
   02657  0700020  OUT     20
   02660  1000200  LAP     200                         RESTORE ACC
   02661  2770100  JMP   7 100
   02662  0010664  CON                    ;CONA
   02663  0011043  BUF+107                ;CEB
                   BSS     40.            ;CON
                   BSS     170            ;BUF
                   ORG     204
                   END
```

TABLE OF POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 755 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25

OCTAL-DECIMAL INTEGER CONVERSION TABLE

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

0000 to 0777 (Octal) — 0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

1000 to 1777 (Octal) — 0512 to 1023 (Decimal)

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

2000 to 2777 (Octal) — 1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2100	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1519	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

3000 to 3777 (Octal) — 1536 to 2047 (Decimal)

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

4000 to 4777 (Octal) 2048 to 2559 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

5000 to 5777 (Octal) 2560 to 3071 (Decimal)

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

6000 to 6777 (Octal) — 3072 to 3583 (Decimal)

Octal	Decimal
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	3132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

7000 to 7777 (Octal) — 3584 to 4095 (Decimal)

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3496	3497	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3758	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3635	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
036	.058593	.136	183593	.236	308593	336	433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	277	373046	.377	498046

OCTAL-DECIMAL FRACTION CONVERSION TABLE (Cont'd)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

OCTAL-DECIMAL FRACTION CONVERSION TABLE (Cont'd)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.001098	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

UNI COMP SYSTEM CODING FORM

PROGRAM

ROUTINE

UNICOMP incorporated

NAME

PAGE

DATE

FUNCTION	I	VARIABLE	LABEL	COMMENTS	IDENT