SYSTEM 2400

SOFTWARE

601 ASSEMBLER MANUAL

MOHAWK DATA SCIENCES CORP

SYSTEM 2400

601 ASSEMBLER MANUAL

EDITION 1

CORPORATE HEADQUARTERS · UTICA, NEW YORK 13503

REVISION PAGE			
EDITION/ ADDENDUM NUMBER	NUMBER OF PAGES AFFECTED	FORM NUMBER AND DATE	SOFTWARE LEVEL SUPPORTED
EDITION 1		PM-2625-1073	

FOREWORD

This manual describes the basic instruction repertoire, format, and detailed information for programming the 601 Microprocessor in machine code and assembly language.

Section I describes the I, S, and R registers and the paging system. Section II discusses the three different language levels: machine code, symbolic notation, and assembly language. Section III explains each assembly language statement, its machine code equivalent, and other information necessary for using the particular statement. Section IV contains the operating procedure.

TABLE OF CONTENTS

SECTION I

REGISTERS

The 601 Microprocessor is a functionally complete, high-performance module of the MDS 600 Series. This microprocessor uses the following registers:

- 16-bit instruction (I) register
- 16 eight-bit scratch (S) registers
- 16 eight-bit R registers

The 16-bit instruction (I) register holds the microinstruction during the instruction execution time. The least significant bits of the register contain the literal operands for instructions that use literal operands.

The 16 eight-bit scratch (S) registers (labeled S0 through $S17_8$) may contain the operands for the arithmetic and logical operations. The S registers may be referenced both as a source and as a destination. The S0 register becomes a dedicated register during the execution of a G statement. At all other times, the S0 register is not dedicated.

The 16 eight-bit R registers (labeled R0 through $R17_8$) also may contain the operands for the arithmetic and logical operations. In addition, the R registers have the following individual functions and limitations:

R0 (A Register) — A general-purpose register that can be referenced both as a source and as a destination. As a secondary function, its contents are used in a G statement to jump across a page boundary.

R1 (C Register) — A condition register used for sensing conditions during execution of certain instructions. In addition, certain bits can signal the occurrence of certain external events, as shown below:

Bit Position	Bit Name	Use
0	LINK	The LINK bit is set by a carryout of the adder from a previous arithmetic operation.

Bit Position	Bit Name	Use
1	SIGN	The SIGN bit is generally used to store the sign of one of the operands of a previous arithmetic operation.
2	POS	The POSITIVE bit is set when bit 7 of the result of an arithmetic or logical operation is Ø.
3	ZERO	The ZERO bit is set when the result of an arithmetic operation is Ø.
4	OVRFLW	The OVERFLOW bit is set when an arithmetic operation which could conceivably generate an overflow condition is performed.
5	PARITY	The PARITY ERROR bit is set when the R-register selected to the R bus appears to have erroneous parity. The parity may be odd or even.
6 7	EXT1 EXT2	The EXTERNAL CONDITION #1 and EXTERNAL CONDITION #2 bits are set by logic outside the microprocessor; their meaning and use are determined by the nature of their source. If these bits are not in use, they are set to 1.

Note that the C register has two types of inputs. It can be used as a normal destination when referenced by its R name, R1. Or the bits can be set by the conditions listed above. If both situations occur simultaneously, the C register is used as a destination and the condition bits are suppressed.

The C register can also be referenced as a normal source, in which case the value of the C register (used as an

operand) is the value before any bits are changed due to condition inputs from the current operation.

R2 (X Register) — A general-purpose register that can be referenced both as a source and as a destination. The X register bits are brought off of the microprocessor and can be used as output discrete bits to control other modules of the system.

R3 (Y Register) — A general-purpose register that can be referenced both as a source and as a destination. As a secondary function, its bits are available to modules outside the microprocessor and they can be used as output discrete bits to control other system components.

R4 (PU Register) — The four most significant bits of the P register. Can be referenced both as a source and as a destination. If it is referenced as a source, the actual reference is a value of zero.

R5 (PL Register) — The eight least significant bits of the P register. Can be referenced both as a source and as a destination. If it is referenced as a source, the actual reference is a value of zero.

The P register (R4 and R5) holds the address of a microinstruction during its access from program memory. Before each instruction access, the P register is incremented by 1.

R6 through R17 (External Registers) — These registers are external to the microprocessor and reside in the adaptors of those modules that must communicate with the microprocessor. The R6 through R17 external registers are enabled by the data contained in the X and Y registers. Register numbers R6 through R17 may be duplicated within the same system, but each register that is enabled will have a unique number in the X and Y register. For example, a system may have three R13 registers on three different modules, but

only one R13 register can be enabled at one time to communicate with the processor. The functional characteristics of these R registers depend on the nature of the modules or devices of which they are a part. Each portion of a module or assembly that must be able to communicate directly with the microprocessor is assigned an R-register address from 6 through 17_8.

PAGING

The 601 Microprocessor divides its memory into pages. Each page has 377_8 bytes of memory, as follows:

Page	Bytes
1	000-000 through 000-377
2	001-000 through 001-377
3	002-000 through 002-377
⋮	⋮ ⋮

The G statement can jump across a page boundary. The GD and GND statements must remain within their original boundaries.

SECTION II

LANGUAGE LEVELS

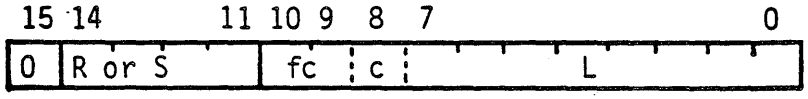
The three levels at which a user can interface with the 601 Microprocessor are:

- Machine Code
- Symbolic Notation
- Assembly Language

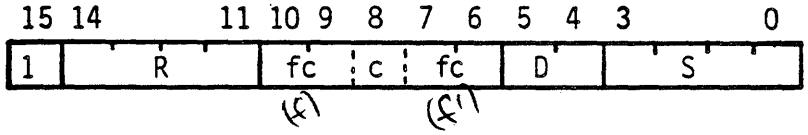
An example of the same instruction in the three language levels is:

230042	Machine Code: Tri-octal
S2 + R3 → R	Symbolic Notation
LABEL ADD S2,R3,R	Assembly Language

In machine code, the 16-bit instructions (may be written as 6 tri-octal digits) can be divided into two major divisions: literal and non-literal. The format of these instructions is:



Literal Instruction



Non-Literal Instruction

The symbols used in these instructions are explained below:

Bit 15 0 = Literal Instruction
1 = Non-Literal Instruction
R = number of R register (Bits 11-14)
S = number of S register (Bit 11-14 of Literal, Bits 0-3 of Non-Literal)
L = Literal value (Bits 0-7 of Literal)

D = destination (Bits 4-5 of Non-Literal): Specifies the location where the result of the operation is to be stored:

00 result to PU, A (R0) to PL
01 result to specified S register
10 result to specified R register
11 result to both the specified S and R registers

c = Setting of C register (Bit 8 in arithmetic and logical statements): Determines whether this statement will affect the C register.

c = 0 Execution of this statement will set the bits of the C register.

c = 1 C register is not affected.

fc = function code (Bits 8-10 of Literal, Bits 6-10 of Non-Literal, except for bit 8 of arithmetic and logical statements, as described above): Specifies the operation of the statement.

The second level, symbolic notation, uses the following symbols:

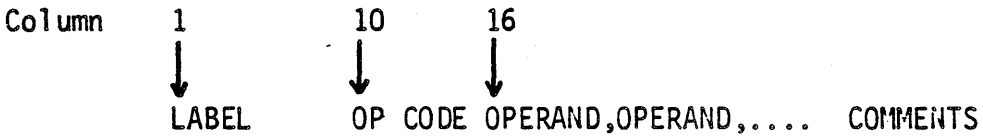
+	plus
-	minus
→	result stored at
∪	Logical OR
⊕	Logical Exclusive OR
∩	Logical AND
S_N	number of S register
R_N	number of R register
D	destination
$\bar{a}$	1's complement of a
C_N	bit number of the C register
Inh C	inhibit C register
RS	right shift
LS	left shift
LK	link bit

The highest level, the assembly language, is a symbolic programming language that makes programming easier and faster by allowing the programmer to use mnemonic representation for function codes, registers, and addresses.

The basic unit of a source program coded in the assembly language is the statement. A statement occupies one line on the programming coding form and, in most cases, represents one instruction. Each statement consists of a string of characters, grouped into fields:

- label field (usually optional)
- operator field (op code)
- operand field
- comment field (optional)

Statements are entered on the coding form in a relatively fixed format, with certain fields required to start in specific column positions:



The label field must start in column 1, the op code field in column 10, and the operand field in column 16. Comments, if any, must be separated from the last operand by at least one space. Operands within the operand field are separated by a comma.

LABEL FIELD

A label uniquely identifies a statement or value so that it can be referenced by other statements. The label is optional; except for the EQU statement which requires a label, and the END statement which cannot have a label.

The label contains from two to six alphanumeric characters, the first of which must be an alphabetic character. The assembler automatically defines certain labels, such as the S and R register names, destination codes, designator names, and commonly used literals. The following labels are defined by the assembler and must not be redefined by the user. Note that assembler-defined labels may be only one character, but user-defined labels must be 2-6 characters.

Assembler-Defined Label	Octal value of label
R0 through $R17_8$	0 through 17_8
S0 through $S17_8$	0 through 17_8
A	0
C	1
X	2

Assembler-Defined Label	Octal value of label
Y	3
PU	4
PL	5
LINK	0
SIGN	1
POS	2
ZERO	3
OVRFLW	4
PARITY	5
EXT1	6
EXT2	7
P	00
R	10
S	01
RS	11
SPACE	100
NULL	000

A label on an assembly language statement has the value of the memory address at which the statement is stored. The programmer may also define a label with the EQU statement.

OPERATOR FIELD

The operator field may contain one of the 24 mnemonics or one of the six assembler directive mnemonics. The operator field defines how the other fields are to be interpreted and what absolute coding (if any) is to result. The individual mnemonics are explained in Section III.

OPERAND FIELD

The operand field has from 0 through 4 operands separated by commas. Spaces are not allowed in the operand field. See Section III for the correct order and type of operand for each statement. The following symbols may be used in the operand field:

S_N = number of S register, where N is an octal digit from 0 through 17, or a label that has been EQUated to an S register name.

R_N = number of R register, where N is an octal digit from 0 through 17, or a label that has been EQUated to an R register name.

D = destination: location where the result of the operation is to be stored:

P result to PU, A (R0) to PL
S result to specified S register
R result to specified R register
RS result to specified S and R registers

L = literal value: An 8-bit value that is to be defined as part of the instruction. A literal may be coded as an octal value (000 through 377) or as a label that has been EQUated to an octal value. If a label is EQUated to a 16-bit value, an asterisk indicates whether the literal value is to be taken from the high-order or low-order 8 bits of the label's 16-bit value.

An asterisk preceding the label indicates that the high-order bits of the literal are to be used; a label without an asterisk indicates the low-order bits.

SC = Statement will not affect the S register.

N = number or name of C-register designator bit:

Name	Number
LINK	0
SIGN	1
POS	2
ZERO	3
OVRFLW	4
PARITY	5
EXT1	6
EXT2	7

COMMENT FIELD

The Comment field must be separated from the last operand by one or more spaces. The field may extend to column 71. This field is included for documentation only

and is ignored by the assembler. A comment statement, identified by an asterisk (*) in column 1, can be used to insert longer comments into a program. In a comment statement, all characters in positions 2 through 71 are treated as comments.

Coding of Assembly Language Statements

Creating a program requires three steps:

- coding the instructions on a standard coding form,
- transferring the instructions to punched cards or tape for input to the assembler, and
- entering the program into the assembler for translation into machine code.

The assembler produces the object code on magnetic tape or punched cards and also generates an assembly listing showing the source language input, and the generated machine language code (in tri-octal). The assembler also prints one of the following error flags beside each statement in which a syntax error is found:

M Multiple labels: two labels have the same spelling.

O Operand error: illegal character in the operand field.

U Undefined label: label in an operand has not been defined.

L Label error: label in an operand has not been defined as an S or R register, although the statement format requires an S or R register.

I Illegal instruction: instruction mnemonic is spelled incorrectly.

F Format error: format of the statement is in error.

P Jump error: short jump (GD or GND) crosses a page boundary.

The assembly program itself is coded in 2400 Assembly Language.

SECTION III

ASSEMBLY LANGUAGE STATEMENTS

This section explains each assembly language statement, including the following information:

Purpose: What the statement is intended to accomplish. The symbolic notation is also included.

Format: The format of the machine code and assembly language, including a brief description of the operands.

Notes: Any additional information pertaining to the statement.

C Register: How the statement affects the C register.

AND/OR Operation: A simple table illustrating the AND or OP operation.

Example: A typical example that illustrates the statement's use.

The statements are presented in two groups: Assembler Directive statements and Assembler statements. Each group is in alphabetical order.

EJECT Statement

Purpose: To start a new page on the assembly listing. This statement may precede the START statement.

Format: Machine Code: None

Assembly Language: EJECT

Note:

1. This statement has no label and no operand.
2. This statement will not appear in the listing.

Example:

```
    .
    .
    .
EJECT
START 4000
    .
    .
    .
EJECT
    .
    .
    .
```

Purpose: To specify the end of the source program.

Format: Machine Code: None

Assembly Language: END

Note: The END statement must be the last statement in a source program. Only one END statement can appear in each source program. The END statement does not require a label or an operand. If either is specified, the Assembler ignores it.

Example:

```
    .
    .
    .
END
```

Purpose: To equate a label to a numeric value or to another label.

Format: Machine Code: None

Assembly Language: label EQU value / label

label (in name field) = required label

value = tri-octal value

label (in operand field) = Assembler-assigned label, such as an R or S register, or a user-assigned label that was previously defined

Example:

```
VAL1      EQU    10
          .
          .
          .
ADDRA     EQU    5377
          .
          .
          .
QUOT      EQU    S1
DIVISR    EQU    S2
DIVDND    EQU    S3
ANSW      EQU    DIVDND
          .
          .
          .
```

G Statement (Go)

Purpose: To perform an unconditional long jump, that is, across a page boundary.

Format: Machine Code: (Generates three machine code words, described in the note below.)

Assembly Language: label G label

label (in name field) = optional label

label (in operand field) = value of the location to which the program jumps. This label is used in the three statements explained in the note below.

Note: The G statement is a macro instruction that generates the following three assembly language statements:

```
ML   label,R0
ML   *label,S0
M    S0,,P
```

The M and ML statements are explained on pages 3-19 and 3-27.

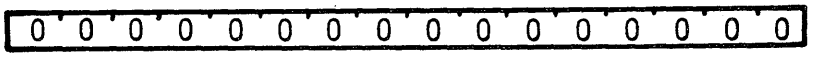
Example:

```
LOGS2   EQU    200100
        :
BRAN    G      LOGS2
        :
```

NOP Statement (No Operation)

Purpose: To create a memory gap of two bytes with zero fill.

Format: Machine Code:



Assembly Language: NOP

Notes:

1. This statement has no label and no operand.
2. In effect, this statement is equivalent to an ORL statement with a literal zero and the R0 register as operands. At execution time, an inclusive OR of the literal zero and the contents of the R0 register is performed and the result is stored at R0.

C Register: The C register is not affected by this statement.

Example:

```
:
NOP
:
```

Purpose: To specify the starting address in core memory for the instructions which follow.

Format: Machine Code: None

Assembly Language: label START address

label = optional label

address = core memory address in tri-octal, or a previously defined label.

Note: The first statement of a source program must be a START statement. Multiple START statements are allowed in a program. A START statement resets the program counter to the value of the address operand, therefore allowing the programmer to start routines at particular locations within program memory.

Example:

```
RTE1    START   3251
          .
          .
          .
        START   4000
          .
          .
          .
```

TITLE Statement

Purpose: To set up a descriptive title which will be printed as the first line of each page of the listing.

Format: Machine Code: None

Assembly Language: TITLE 'text'

text = 1-40 characters, enclosed in apostrophes, printed as the first line of each page.

Note: The TITLE statement is printed on the source code listing as a comment record (an asterisk followed by the text).

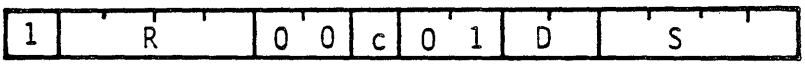
Example:

```
TITLE '601 ASSEMBLER PROGRAM'
:
:
```

ADC Statement (Add Binary plus Carry)

Purpose: To add (in binary) the values of the specified S and R registers plus any carry from a previous operation, and to store the result in the location specified by D. $S_N+R_N+LINK \rightarrow D$

Format: Machine Code:



Assembly Language: label ADC S_N,R_N,D,SC

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: S, R, RS, or P

SC = optional: if used, C register not affected by this operation

C Register: If the destination (D) is not the C register and the SC operand is omitted, the designator bits in the C register are set as follows:

- LINK: set only if this instruction generated a carry-out from the adder.
- SIGN: Set only the initial contents of the R register were positive.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only if the signs of the initial contents of the S and R registers were alike and the signs of the result and the initial contents of R are unlike.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.

- If 9 bits have odd number of 1's, PARITY bit is cleared.

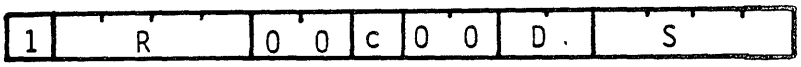
- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Example:

```
LAB1    ADC     S2,R7,S
        :
        :
        ADC     S7,X,R,SC
        :
        :
```

Purpose: To add (in binary) the values of the specified S and R registers and to store the result in the location specified by D. $S_N + R_N \rightarrow D$

Format: Machine Code:



Assembly Language: label ADD S_N,R_N,D,SC

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

SC = optional: C register not affected by this operation

C Register: If the destination (D) is not the C register and the SC operand is omitted, the designator bits in the C register are set as follows:

- LINK: set only if this instruction generated a carry-out from the adder.
- SIGN: set only if the initial contents of the R register were positive.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only if the signs of the initial contents of S and R registers were alike and the signs of the result and the initial contents of R are unlike.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.

- If 9 bits have odd number of 1's, PARITY bit is cleared.

- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Example:

```
RTE3    ADD    S1,R3,S,SC
        :
        :
        ADD    S5,R1,R
        :
        :
```

Purpose: To perform a logical AND on the contents of the specified R and S registers and to store the result in the location specified by D. $S_N \frown R_N \rightarrow D$

Format: Machine Code:

Assembly Language: label AND S_N,R_N,D,SC

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

SC = optional: C register not affected by this operation

C Register: If the destination (D) is not the C register and the SC operand is omitted, the designator bits in the C register are set as follows:

- LINK: unaffected.
- SIGN: unaffected.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: always cleared.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.

- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Logical AND Operation:

```
  1 1 0 0
  1 0 1 0
  -------
  1 0 0 0
```

Example:

```
LOG1      AND    S3,R2,RS
          :
STATUS    EQU    S2
          :
          AND    STATUS,R2,S,SC
          :
          AND    S3,R2,P,SC
          :
```

GD Statement (GOTO on Designator)

Purpose: To test a designator bit in the C register to determine whether the program is to branch to another address within the same memory page. The branch is made if the bit is set. $L \rightarrow R_5\ C_N$

Format: Machine Code:

Assembly Language: label GD N,L

label = optional label

N = number or name of C register designator bit to be tested:

Bit	Name
0	LINK
1	SIGN
2	POS
3	ZERO
4	OVRFLW
5	PARITY
6	EXT1
7	EXT2

L = literal value or label equated to literal value: placed in P_1 if N is set. An asterisk preceding the label indicates that the high-order eight bits of the literal are to be used; a label without an asterisk indicates the low-order bits.

Note: If the specified designator bit is set (=1), the literal value replaces the contents of the lower eight bits of the P register, and the next instruction executed is taken from this revised address.

If the specified designator bit is not set (=0), the next instruction executed is the one following the GD statement.

C Register: The C register is not affected by this statement.

Example:

```
        BALRTE  GD      OVRFLW,127
                :
                :
        ADDR1   EQU     143265
                :
                :
                GD      3,*ADDR1    <-- 3 = ZERO designator
                :                       *ADDR1 = high-order
                :                                bits: 143
                GD      1,ADDR1     <-- 1 = SIGN designator
                :                       ADDR1 = low-order
                :                               bits: 265
```

Purpose: To test a designator bit in the C register to determine whether the program is to branch to another address within the same memory page. The branch is made if the bit is not set.

$L \rightarrow R_5 \overline{C_N}$

Format: Machine Code:

Assembly Language: label GND N,L

label = optional label

N = number or name of C-register designator bit to be tested.

Bit	Name
0	LINK
1	SIGN
2	POS
3	ZERO
4	OVRFLW
5	PARITY
6	EXT1
7	EXT2

L = literal value or label equated to literal value: placed in P_1 if N is not set. An asterisk preceding the label indicates that the high-order eight bits of the literal are to be used; a label without an asterisk indicates the low-order bits.

Note: If the specified designator bit is not set (=0), the literal value replaces the contents of the lower eight bits of the P register, and the next instruction executed is taken from this revised address.

If the specified designator bit is set (=1), the next instruction executed is the one following the GND statement.

C Register: The C register is not affected by this statement.

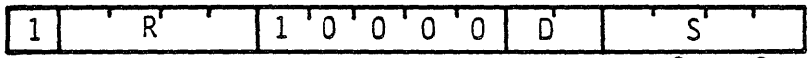
Example:

```
LAB1    GND     5,245
        :
VAL     EQU     132015
        :
        GND     SIGN,*VAL
```

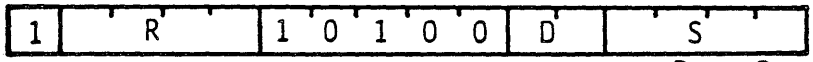
*VAL = high-order bits: 132

Purpose: To move the contents of the specified R or S register to the location specified by D. $S_N \rightarrow D$ $R_N \rightarrow D$

Format: Machine Code:



$S_N \rightarrow D$



$R_N \rightarrow D$

Assembly Language: label M S_N,R_N,D $S_N \rightarrow D$

label M R_N,S_N,D $R_N \rightarrow D$

Label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

Note:

1. Operand 1 (S_N or R_N) is moved to D.
2. Operand 2 (S_N or R_N) is required if:
   - D = RS, or
   - D is not the same type of register (S or R) as operand 1.
3. Operand 2 (S_N or R_N) is optional if:
   - D = P, or
   - D is the same type of register (S or R) as operand 1.
4. Operand 2, when used, must not be the same type (S or R) as operand 1.

C Register: If the destination (D) is not the C register, the designator bits in the C register are set as follows:

- LINK: unchanged.
- SIGN: always set.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: always cleared.

- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is Ø if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.
- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Example:

Valid Statements:

```
NAM1    M     S3,R2,R  <-------------  S3 → R2
        .
        .
        M     R5,,P    <-------------  R5 → P_U      RØ → P_L
        .
        .
        M     R7,S3,R  <-------------  R7 → R7
                                       S3 is ignored
```

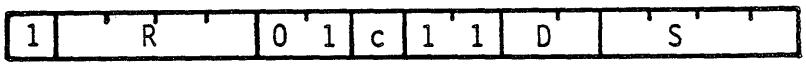
Invalid Statements:

```
        M     S1,S5,S  <--------------+  Second operand is
        .                             |  same type as first.
        .                             |
        M     R3,,S    <--------------+  Second operand is
        .                             |  required.
        .                             |
        M     S4,,RS   <--------------+
        .
        .
```

MC Statement (Move Ones Complement)

Purpose: To take the ones complement of the specified R register and to move the result to the location specified by D. $R_N \rightarrow D$ InhC

Format: Machine Code:



Assembly Language: label = optional label

R_N = number of R register

S_N = number of S register

D = destination: R, S, RS, or P

SC = optional: If used, C register not affected by this operation

Notes:

1. S_N is required if D = S or RS.
2. S_N is optional if D = P or R.
3. The MC statement can move values only from an R register; other move statements can move values from either an R or S register.

C Register: If the destination (D) is not the C register and the SC operand is omitted, the designator bits in the C register are set as follows:

- LINK: unchanged.
- SIGN: unchanged.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only if the initial contents of the R register are positive.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.

- If 9 bits have odd number of 1's, PARITY bit is cleared.

- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

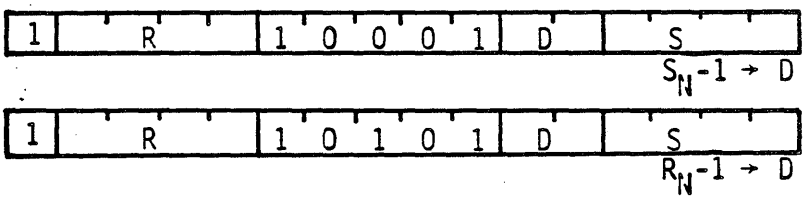
Example:

```
COMPLM    MC    R6,S6,RS
          :
          :
          MC    R6,,R,SC
          :
          :
```

Purpose: To move the contents of the specified S or R register minus a binary 1 to the location specified by D. $S_N - 1 \rightarrow D$

$R_N - 1 \rightarrow D$

Format: Machine Code:



Assembly Language:

label MD S_N,R_N,D $S_N - 1 \rightarrow D$

label MD R_N,S_N,D $R_N - 1 \rightarrow D$

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: S, R, RS, or P

Notes:

1. Operand 1 (S_N or R_N) is moved to D.
2. Operand 2 is required if:
   - D = RS, or
   - D is not the same type of register (S or R) as operand 1.
3. Operand 2 is optional if:
   - D = P, or
   - D is the same type of register (S or R) as operand 1.
4. Operand 2, when used, must not be the same type (S or R) as operand 1.

C Register: If the destination (D) is not the C register, the designator bits in the C register are set as follows:

- LINK: set only if this instruction generated a carry-out from the adder.
- SIGN: always cleared.

- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only if the initial contents of the register name in operand 1 are negative and the sign of the result is positive.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
    - If 9 bits have even number of 1's, PARITY bit is set.
    - If 9 bits have odd number of 1's, PARITY bit is cleared.
- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Example:

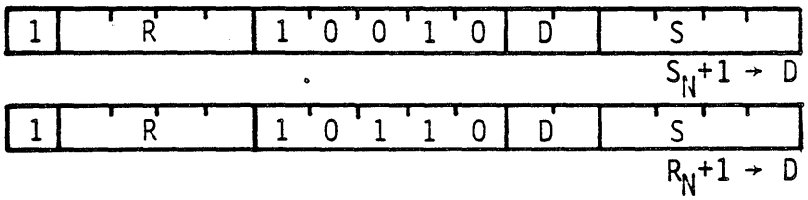
```
SEC     MD      S1,R1,R
        .
        .
        .
REG1    EQU     S3
REG2    EQU     R3
        .
        .
        .
        MD      REG1,,S
        MD      REG2,,R
        .
        .
        .
```

MI Statement (Move and Increment Register)

Purpose: To move the contents of the specified S or R register plus a binary 1 to the location specified by D. $S_N+1 \rightarrow D$

$R_N+1 \rightarrow D$

Format: Machine Code:



Assembly Language:

label	MI	S_N,R_N,D	$S_N+1 \rightarrow D$
label	MI	R_N,S_N,D	$R_N+1 \rightarrow D$

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: S, R, RS, or P

Notes:

1. Operand 1 (S_N or R_N) is moved to D.
2. Operand 2 is required if:
   - D = RS, or
   - D is not the same type of register (S or R) as operand 1.
3. Operand 2 is optional if:
   - D = P, or
   - D is the same type of register (S or R) as operand 1.
4. Operand 2, when used, must not be the same type (S or R) as operand 1.

C Register: If the destination (D) is not the C register, the designator bits in the C register are set as follows:

- LINK: set only if this instruction generated a carry-out from the adder.
- SIGN: always cleared.

- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only if the initial contents of the register name in operand 1 are positive and the sign of the result is positive.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.
- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Example:

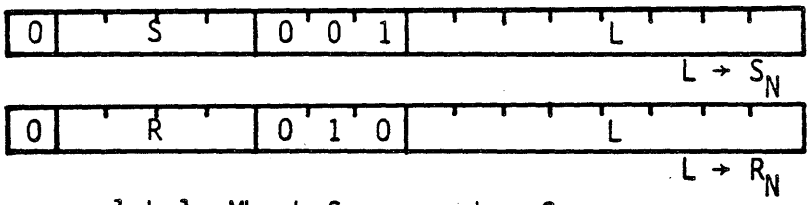
```
SEC1    MI      S1,R1,R
        .
        .
REG1    EQU     S3
REG2    EQU     R3
        .
        .
        MI      REG1,,S
        MI      REG2,,R
        .
        .
        .
```

MIL Statement (Move Literal to Register)

Purpose: To move a literal value to the specified S or R register.

$L \rightarrow S_N$ $L \rightarrow R_N$

Format: Machine Code:



Assembly Language:

label MIL L,S_N $L \rightarrow S_N$

label MIL L,R_N $L \rightarrow R_N$

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: S, R, RS, or P

L = literal value or label equated to literal value. An asterisk preceding the label indicates that the high-order bits of the literal are to be used; a label without an asterisk indicates the low-order bits.

C Register: The C register is not affected by this statement.

Example:

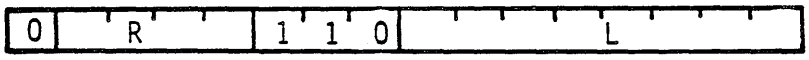
```
MOV1    MIL     377,R5
RES1    EQU     342173
          .
          .
          .
        MIL     *RES1,S3
        MIL     RES1,R1
          .
          .
          .
```

NDL Statement (Logical AND Literal to R register)

Purpose: To logically AND a literal value and the contents of the specified R register, and to store the result in the specified R register.

$L \cap R_N \rightarrow R_N$

Format: Machine Code:



Assembly Language: label NDL L,R_N

label = optional label

L = literal value or label equated to literal value. An asterisk preceding the label indicates that the high-order eight bits of the literal are to be used; a label without an asterisk indicates the low-order bits.

R_N = number of the R register

C Register: The C register is not affected by this statement.

AND Operation:

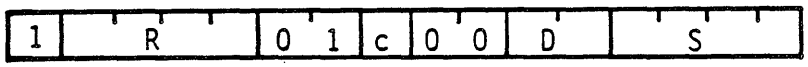
```
  1 1 0 0
  1 0 1 0
  1 0 0 0
```

Example:

```
LABL    EQU    152311
        :
        :
        NDL    LABL,R7
        NDL    *LABL,R1
        :
        :
```

Purpose: To perform an inclusive OR on the contents of the specified S and R registers, and to store the result in the location specified by D. $S_N \cup R_N \rightarrow D$

Format: Machine Code:



Assembly Language: label OR S_N,R_N,D,SC

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

SC = optional: If used, C register not affected by this operation.

C Register: If the destination (D) is not the C register and the SC operand is omitted, the designator bits in the C register are set as follows:

- LINK: unchanged.
- SIGN: unchanged.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only if the arithmetic sum of the contents of the specified S and R registers would cause an overflow.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.

- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Inclusive OR Operation:

```
  1 1 0 0
  1 0 1 0
  1 1 1 0
```

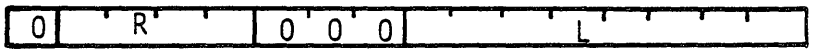
Example:

```
NAM   OR    S2,R2,R
      :
      :
      OR    S4,R4,RS,SC
      :
      :
```

ORL Statement (Logical OR Literal to R Register)

Purpose: To perform an inclusive OR of a literal value and contents of the specified R register, and to store the result in the specified R register. $L \cup R_N \rightarrow R_N$

Format: Machine Code:



Assembly Language: label ORL L,R_N

label = optional label

L = literal value or label equated to literal value. An asterisk preceding the label indicates that the high-order eight bits of the literal are to be used; a label without an asterisk indicates the low-order bits.

R_N = number of the R register

C Register: The C register is not affected by this statement.

Inclusive OR Operation:

```
  1 1 0 0
  1 0 1 0
  -------
  1 1 1 0
```

Example:

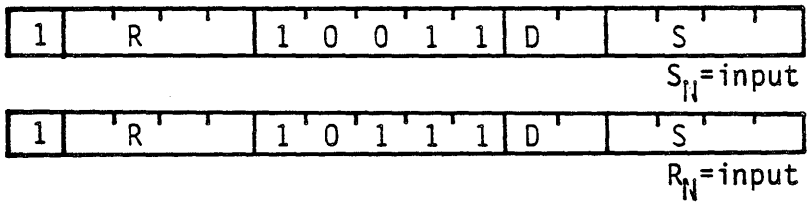
```
LIT     EQU     013225
        :
NAM     ORL     *LIT,R4
HIJ     ORL     LIT,R1
        :
```

PC Statement (Propagate Carry)

Purpose: To propagate the carry from a previous operation (as stored in the LINK and SIGN designators) into the location specified by D.

$S_N \pm LINK \rightarrow D$ $R_N \pm LINK \rightarrow D$

Format: Machine Code:



Assembly Language:

label PC S_N,R_N,D S_N = input

label PC R_N,S_N,D R_N= input

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

Notes:

1. Operand 1 (S_N or R_N) specifies the input register.
2. The PC statement permits the sign and carry out from a previous operation to be carried forward (to the left) in a multibyte increment or decrement operation. The operation performed is:

   (operand 1) - 1 + LINK → D (if SIGN bit = Ø)

   or

   (operand 1) + LINK → D (if SIGN bit = 1)
3. Operand 2 (S_N or R_N) is required if:
   - D = RS, or
   - D is not the same type of register (S or R) as operand 1.
4. Operand 2 (S_N or R_N) is optional if:
   - D = P, or
   - D is the same type of register (S or R) as operand 1.

5. Operand 2, when used, must not be the same type (S or R) as operand 1.

C Register: If the destination (D) is not the C register, the designator bits in the C register are set as follows:

- LINK: set only if this instruction generated a carry-out from the adder.
- SIGN: unchanged.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only under the two following conditions:
  1. The SIGN designator was set, operand 1 is positive, and the result is negative.
  2. The SIGN designator was not set, operand 1 is negative, and the result is positive.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.
- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

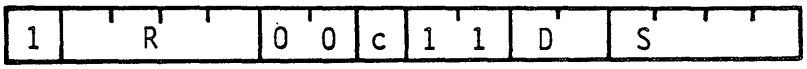
Example:

```
RGS1    PC    S1,R1,R
        .
        .
        .
        PC    R5,,R
        .
        .
        .
```

SBC Statement (Binary Subtract Plus Carry)

Purpose: To subtract (in binary) the contents of the specified R register from the value of the specified S register, to add to this result the carry from a previous operation, and to store the final result in the location specified by D. $S_N - R_N + LINK \rightarrow D$

Format: Machine Code:



Assembly Language: label SBC S_N,R_N,D,SC

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

SC = optional: If used, C register not affected by this operation.

C Register: If the destination (D) is not the C register and the SC operand is omitted, the designator bits in the C register are set as follows:

- LINK: set only if this instruction generated a carry-out from the adder.
- SIGN: set only if the initial contents of the R register are negative.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only if the signs of the initial contents of the S and R registers are alike, and the signs of the result and the initial contents of the R register are not alike.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):

- If 9 bits have even number of 1's, PARITY bit is set.
- If 9 bits have odd number of 1's, PARITY bit is cleared.

- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

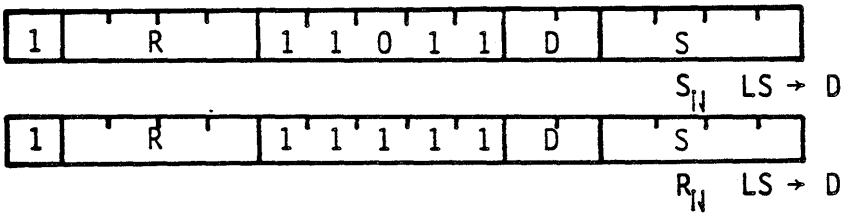
Example:

```
SAV1    EQU     S3
        .
        .
        .
        SBC     SAV1,R2,R,SC
        .
        .
        .
NUM     SBC     S4,R2,RS
        .
        .
        .
```

SL Statement (Shift Left With Zero Fill)

Purpose: To shift the contents of the specified S or R register one bit position to the left and to store the result in the location specified by D. The vacated position on the right is set to binary zero. S_{iN} LS → D Fill Ø's R_{iN} LS → D Fill Øs

Format: Machine Code:



Assembly Language:

label SL S_{iN},R_{iN},D — S_{iN} LS → D

label SL R_{iN},S_{iN},D — R_{iN} LS → D

label = optional label

S_{iN} = number of S register

R_{iN} = number of R register

D = destination: R, S, RS, or P

Notes:

1. Operand 1 is left shifted to D.
2. Operand 2 is required if:
   - D = RS, or
   - D is not the same type of register (S or R) as operand 1.
3. Operand 2 is optional if:
   - D = P, or
   - D is the same type of register (S or R) as operand 1.
4. Operand 2, when used, must not be the same type (S or R) as operand 1.

C Register: If the destination (D) is not the C register, the designator bits in the C register are set as follows:

- LINK: unchanged.
- SIGN: set only if the leftmost bit of the initial value of the shifted register is Ø.

- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: always cleared.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.
- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

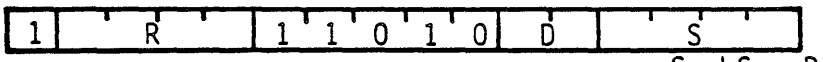
Example:

```
NAM     SL      S1,R1,R
        .
        .
        .
        SL      R1,,R
        .
        .
        .
        SL      S1,,P
        .
        .
        .
```

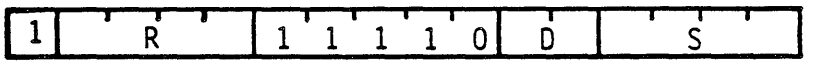
SLS Statement (Shift Left with $\overline{SIGN}$ Fill)

Purpose: To shift the contents of the specified S or R register one bit position to the left and to store the result in the location specified by D. The vacated position on the right is filled with the complement of the SIGN bit (SIGN=Ø,Fill=1; SIGN=1, Fill=Ø). S_N LS → D Fill $\overline{SIGN}$ R_N LS → D Fill $\overline{SIGN}$

Format: Machine Code:



S_N LS → D



R_N LS → D

Assembly Language:

label SLS S_N,R_N,D S_N LS → D

label SLS R_N,S_N,D R_N LS → D

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

Notes:

1. Operand 1 is left shifted to D.
2. Operand 2 is required if:
   - D = RS, or
   - D is not the same type of register (S or R) as operand 1.
3. Operand 2 is optional if:
   - D = P, or
   - D is the same type of register (S or R) as operand 1.
4. Operand 2, when used, must not be the same type (S or R) as operand 1.

C Register: If the destination (D) is not the C register, the designator bits in the C register are set as follows:

- LINK: unchanged.

- SIGN: set only if the leftmost bit of the initial value of shifted register is Ø.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: always cleared.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.
- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

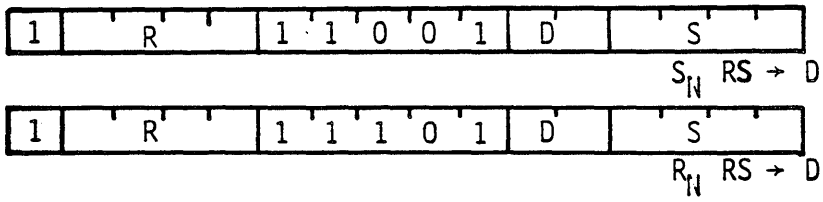
Example:

```
START   SLS     S1,R1,R
        .
        .
        .
        SLS     R1,,R
        .
        .
        .
        SLS     S1,,S
        .
        .
        .
```

SRF Statement (Shift Right With 1 Fill)

Purpose: To shift the contents of the specified S or R register one bit position to the right and to store the result in the location specified by D. The vacated position on the left is set to binary 1. S_N RS → D Fill 1's R_N RS → D Fill 1's

Format: Machine Code:



Assembly Language:

label SRF S_N,R_N,D S_N RS → D

label SRF R_N,S_N,D R_N RS → D

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

Notes:

1. Operand 1 is right shifted to D.
2. Operand 2 is required if:
   - D = RS, or
   - D is not the same type of register (S or R) as operand 1.
3. Operand 2 is optional if:
   - D = P, or
   - D is the same type of register (S or R) as operand 1.
4. Operand 2, when used, must not be the same type (S or R) as operand 1.

C Register: If the destination (D) is not the C register, the designator bits in the C register are set as follows:

- LINK: unchanged.
- SIGN: set only if the rightmost bit of the initial value of the shifted register is Ø.

- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: always cleared.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.
- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Example:

```
TEST    EQU     S3
        .
        .
NAM     SRF     R1,,R
        .
        .
        SRF     TEST,,S
        .
        .
```

SRS Statement (Shift Right With $\overline{SIGN}$ Fill)

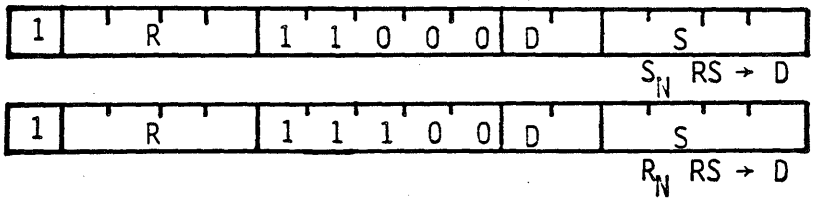
Purpose:

To shift the contents of the specified S or R register one bit position to the right and to store the result in the location specified by D. The vacated position on the left is filled with the complement of the SIGN bit (SIGN=Ø,Fill=1; SIGN=1,Fill=Ø).

S_N RS → D Fill $\overline{SIGN}$ R_N RS → D Fill $\overline{SIGN}$

Format:

Machine Code:



Assembly Language:

label SRS S_N,R_N,D S_N RS → D

label SRS R_N,S_N,D R_N RS → D

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

Notes:

1. Operand 1 is right shifted to D.
2. Operand 2 is required if:
   - D = RS, or
   - D is not the same type of register (S or R) as operand 1.
3. Operand 2 is optional if:
   - D = P, or
   - D is the same type or register (S or R) as operand 1.
4. Operand 2, when used, must not be the same type (S or R) as operand 1.

C Register:

If the destination (D) is not the C register, the designator bits in the C register are set as follows:

- LINK: unchanged.

- SIGN: set only if the rightmost bit of the initial value of the shifted register is Ø.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: always cleared.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.
- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Example:

```
HERE    SRS     S1,R1,R
        .
        .
        .
        SRS     R1,,R
        .
        .
        .
        SRS     S1,R2,RS
        .
        .
        .
```

SUB Statement (Binary Subtract)

Purpose: To subtract (in binary) the contents of the specified R register from the value of the specified S register, and to store the result in the location specified by D. $S_N - R_N \rightarrow D$

Format: Machine Code:

Assembly Language: Label SUB S_N,R_N,D,SC

label = optional label

S_N = number of S register

R_N = number of R register

D = destination: R, S, RS, or P

SC = optional: If used, C register not affected by this operation.

C Register: If the destination (D) is not the C register and the SC operand is omitted, the designator bits in the C register are set as follows:

- LINK: set only if the operation generates a carry-out from the adder.
- SIGN: set only if the initial contents of the R register are negative.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only if the signs of the initial contents of the S and R registers are not alike, and the signs of the result and the initial contents of the R register are alike.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):

- If 9 bits have even number of 1's, PARITY bit is set.
- If 9 bits have odd number of 1's, PARITY bit is cleared.

- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

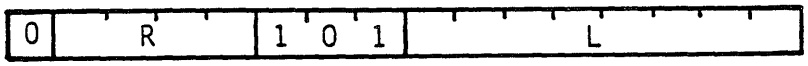
Example:

```
LBL1    EQU    S5
LBL2    EQU    S6
LBL3    EQU    R2
        .
        .
        .
        SUB    S4,LBL3,R5
        .
        .
        .
ARIT    SUB    S6,R3,RS,SC
        .
        .
        .
```

TL Statement (Test Literal and Set C Register)

Purpose: To perform an exclusive OR of a literal value and the contents of the specified R register, and to use the result to set the C register. $L \oplus R_N$

Format: Machine Code:



Assembly Language: label TL L,R_N

label = optional label

L = literal value or label equated to literal value. An asterisk preceding the label indicates that the high-order eight bits of the literal are to be used; a label without an asterisk indicates the low-order bits.

R_N = number of the R register

Note: The contents of the R register are unchanged by this statement.

C Register: The designator bits in the C register are set as follows:

- LINK: unchanged.
- SIGN: unchanged.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: unchanged.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.

- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Exclusive OR Operation:

```
  1 1 0 0
  1 0 1 0
  0 1 1 0
```

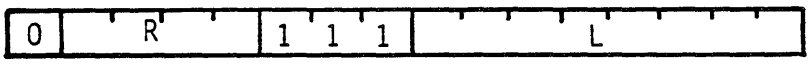
Example:

```
VALUE   EQU     102304
        .
        .
        .
        TL      *VALUE,R7
        .
        .
        .
        TL      277,R3
        .
        .
        .
```

TM Statement (Test Mask and set C Register)

Purpose: To perform a logical AND on a literal value and the contents of the specified R register, and to use the result to set the C register. $L \frown R_N$

Format: Machine Code:



Assembly Language: label TM L,R_N

label = optional label

L = literal value or label equated to literal value. An asterisk preceding the label indicates that the high-order eight bits of the literal are to be used; a label without an asterisk indicates the low-order bits.

R_N = number of the R register

Note: The contents of the R register are unchanged by this statement.

C Register: The designator bits in the C register are set as follows:

- LINK: unchanged.
- SIGN: unchanged.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: always cleared.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.

- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

AND Operation:

```
  1 1 0 0
  1 0 1 0
  1 0 0 0
```

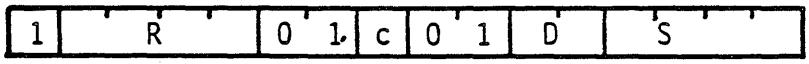
Example:

```
LIT       EQU    000020
          .
          .
          .
CHKBIT    TM     004,R5
          .
          .
          .
          TM     LIT,R1
          .
          .
          .
```

XOR Statement (Exclusive-OR)

Purpose: To perform an exclusive OR on the contents of the specified S and R registers, and to store the result in the location specified by D. $S_N \oplus R_N \rightarrow D$

Format: Machine Code:



Assembly Language: label XOR S_N,R_N,D,SC

label = optional label

S_N = number of the S register

R_N = number of the R register

D = destination: R, S, RS, or P

SC = optional: If used, C register not affected by this operation.

Register: If the destination (D) is not the C register and the SC operand is omitted, the designator bits in the C register are set as follows:

- LINK: unchanged.
- SIGN: unchanged.
- POS: set only if bit 7 of the result is Ø.
- ZERO: set only if the result is Ø.
- OVRFLW: set only if the initial contents of the S register are negative, and the initial contents of the R register are positive.
- PARITY: The nine bits (8 data and 1 parity) from the specified R register are checked for parity (the parity bit is 0 if it is not implemented on an addressed register, such as R0, R1, R2, R3, R4, and R5):
  - If 9 bits have even number of 1's, PARITY bit is set.
  - If 9 bits have odd number of 1's, PARITY bit is cleared.

- EXT1 and EXT2: set only if a related external condition is present in the interfacing hardware.

Exclusive OR Operation:

```
  1 1 0 0
  1 0 1 0
  0 1 1 0
```

Example:

```
START   XOR    S2,R1,S
        .
        .
        .
        XOR    S3,R2,RS,SC
        .
        .
        .
```

SECTION IV

OPERATING PROCEDURE

OPERATING PROCEDURE

1. Load the Assembler program with the Software Loader (see Section I of Utilities Manual, Form No. M-2601) or the Tape Monitor System (see the TMS Manual, Form No. M-2606). Program name is "ASM601".

2. Set the sense switches according to Table 4-1. The valid input and output devices are:

   Source Input:

   - Tape (Logical Unit 1)
   - Card Reader
   - Reader/Punch

   Object Output:

   - Tape (Logical Unit 2)
   - Card Punch
   - Reader/Punch

   List Output:

   - Tape (Logical Unit 3)
   - Printer

Table 4-1. Sense Switch Settings

Switch	Meaning
A	Source Input: ON Tape (Logical Unit 1) OFF Card (set switch E as desired)
B	Object Output: ON Tape (Logical Unit 2) OFF Card (set switch D as desired)
C	List Output: ON Tape (Logical Unit 3) OFF Line Printer

Table 4-1. Sense Switch Settings (Continued)

Switch	Meaning
D	Card unit for Object Output (if B is ON, switch D is ignored): ON Reader/Punch OFF Card Punch
E	Card unit for Source Input (if A is ON, switch E is ignored): ON Card Reader OFF Reader/Punch
F	Not used.
G	Not used.
H	Object Output: ON Suppress output OFF Output to selected device.

3. Press RUN.

4. At completion of first pass, reload card input if used. If input is from tape unit, tape automatically rewinds.

5. The indicator lights are explained in Table 4-2. If an error occurs, two options are available:

   - Press RUN and ignore the error, or
   - Restart the job.

Table 4-2. Indicator Lights

Lights	Meaning
A	Processing second pass.
F	Error on List device.
G	Error on Object Output device.
H	Error on Source Input device.
ABCDEFGH	End of assembly.

Both the list and object output tapes contain double tape marks for denoting end of tape. Multi-file output tapes have a single tape mark between each file and two tape marks at the end of the last file.

READER'S COMMENT FORM

Software Reference Manual
601 Assembler, Form No. PM-2625-1073

Please restrict remarks to the publication itself, giving specific page and line references with your comments when appropriate. This form will be sent to the publication's author for appropriate action. All comments and suggestions become the property of MDS.

Requests for system assistance or publications should be directed to your MDS representative or to the MDS Branch Office serving your area.

ERRORS NOTED:

SUGGESTIONS FOR IMPROVEMENT:

How do you use this document?

- ☐ As an operator's Reference Manual
- ☐ As an introduction to the subject
- ☐ As an aid to instruction in a class
- ☐ As a student text book
- ☐ For advanced knowledge of subject

Do you wish a reply?

- ☐ Yes
- ☐ No

Your Name _____ Date _____

Occupation _____

Company _____

Address _____

Street City State Zip Code

Fold

Fold

Postage Will Be Paid by Addressee

No Postage Stamp Necessary If Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS MAIL PERMIT NO. 4118 ST. PAUL, MINN.

MOHAWK DATA SCIENCES CORP.

1256 TRAPP ROAD

ST. PAUL, MINNESOTA 55121

ATTN: SOFTWARE PUBLICATIONS

Fold

Fold

Cut Along Line

THE THROUGHPUT SPECIALISTS

MOHAWK DATA
SCIENCES CORP.

ORPORATE HEADQUARTERS · UTICA, NEW YORK 13503