

Burroughs
B 1700
SYSTEMS

MICRO
IMPLEMENTATION
LANGUAGE
(MIL)

REFERENCE MANUAL

\$4.00

Burroughs

B 1700 SYSTEMS

MICRO IMPLEMENTATION

LANGUAGE

(MIL)

REFERENCE MANUAL

NOTICE

THE MATERIAL IN THIS DOCUMENT IS NOT TO BE REPRODUCED, COPIED OR UTILIZED FOR FURTHER PUBLICATION. ADDITIONAL COPIES SHOULD BE OBTAINED FROM BURROUGHS CORPORATION UNDER THE TERMS OF THE APPROPRIATE PROGRAM PRODUCTS LICENSE.



Burroughs Corporation

Detroit, Michigan 48232

\$4.00

COPYRIGHT © 1973 BURROUGHS CORPORATION

**The information contained herein is subject to change
without notice. Revisions may be issued to advise of
such changes and/or additions.**

**Correspondence regarding this document should be forwarded using the Remarks Form at
the back of the manual, or may be addressed directly to Systems Documentation, Technical
Information Organization, TIC-Central, Burroughs Corporation, Burroughs Place, Detroit,
Michigan 48232.**

TABLE OF CONTENTS

| SECTION | TITLE | PAGE |
|---------|---|------|
| | INTRODUCTION. | xi |
| 1 | MICRO PROGRAMMING CONCEPTS. | 1-1 |
| | General | 1-1 |
| | Micro Instructions. | 1-1 |
| | Defined Field Concepts. | 1-2 |
| | Interpretation of the Virtual Language. | 1-2 |
| 2 | MICRO IMPLEMENTATION LANGUAGE | 2-1 |
| | Assembly Coding Form. | 2-1 |
| | Conditions. | 2-1 |
| | Labels. | 2-3 |
| | Literals. | 2-4 |
| | Strings | 2-5 |
| 3 | REGISTERS | 3-1 |
| | General | 3-1 |
| | Alphabetical Listing of Registers | 3-1 |
| | Active Registers. | 3-5 |
| | X-Y Registers | 3-6 |
| | Field (F) Register. | 3-6 |
| | Local (L) Register. | 3-6 |
| | Transform (T) Register. | 3-6 |
| | Micro Instruction (M) Register. | 3-7 |
| | Base (BR) and Limit (LR) Registers. | 3-7 |
| | Address (A) Register. | 3-7 |
| | A Stack | 3-7 |
| | Top of Control Memory (TOPM) Register | 3-8 |
| | Memory Base Register (MBR). | 3-8 |
| | Control (C) Register. | 3-8 |

TABLE OF CONTENTS (cont)

| SECTION | TITLE | PAGE |
|----------|---|------|
| 3 (cont) | Combinatorial Logic or Function Box | 3-9 |
| | Result Registers | 3-9 |
| | XORY Result Register | 3-9 |
| | XANY Result Register | 3-9 |
| | XEOY Result Register | 3-10 |
| | CMPX Result Register | 3-10 |
| | CMPY Result Register | 3-10 |
| | MSKX Result Register | 3-10 |
| | MSKY Result Register | 3-10 |
| | SUM Result Register. | 3-10 |
| | Difference Result Register | 3-11 |
| | Scratchpad | 3-11 |
| | Scratchpad Words - 24 Bits Each. | 3-11 |
| | Double Scratchpad Words - 48 Bits Each. | 3-12 |
| | Constant Registers | 3-12 |
| | Maximum Main Memory Register | 3-12 |
| | Maximum Control Memory Register. | 3-12 |
| | Null Register. | 3-12 |
| | Input/Output Registers | 3-12 |
| | Console Cassette Tape Input (U) Register | 3-12 |
| | Command Register | 3-13 |
| | Data Register. | 3-13 |
| | Condition Registers. | 3-13 |
| | Binary Conditions (BICN) Register | 3-14 |
| | Least Significant Unit of Y. | 3-14 |
| | Carry Control Register | 3-14 |
| | Carry Difference Register. | 3-14 |
| | Carry Level (CYL) Register | 3-14 |

TABLE OF CONTENTS (cont)

| SECTION | TITLE | PAGE |
|----------|--|------|
| 3 (cont) | XY Conditions (XYCN) Register | 3-14 |
| | Most Significant Unit of X. | 3-14 |
| | XY States (XYST) Register | 3-15 |
| | Least Significant Unit of X | 3-15 |
| | Any-Interrupt | 3-15 |
| | Console Interrupt | 3-16 |
| | Main Memory Read Parity Error Interrupt | 3-16 |
| | Main Memory Address Out of Bounds Override | 3-16 |
| | Read Address Out of Bounds Interrupt | 3-16 |
| | Write/Swap Address Out of Bounds Interrupt | 3-17 |
| | Field Length Conditions (FLCN) Register . | 3-17 |
| | Interrupt Conditions (INCN) Register . . | 3-17 |
| | No Device | 3-17 |
| | Hi-Priority | 3-17 |
| | Interrupt | 3-17 |
| | Lockout | 3-17 |
| | Register Designations and Areas of Application. | 3-17 |
| | Micro Instruction Controls. | 3-18 |
| | S-Memory Controls | 3-18 |
| | Interrupt Controls. | 3-18 |
| | Parallel Width Controls | 3-18 |
| | Organization of Fields and Subfields | 3-18 |
| 4 | MICRO OPERATORS | 4-1 |
| | Notations and Conventions Used in the Syntax | 4-1 |

TABLE OF CONTENTS (cont)

| SECTION | TITLE | PAGE |
|----------|---|------|
| 4 (cont) | Key Words | 4-1 |
| | Lower Case Words. | 4-1 |
| | Braces. | 4-1 |
| | Brackets. | 4-1 |
| | Concurrent Execution of Micro Operators | 4-2 |
| | ADJUST. | 4-3 |
| | ADD SCRATCHPAD. | 4-4 |
| | AND | 4-5 |
| | BIAS. | 4-7 |
| | CALL. | 4-9 |
| | CARRY | 4-10 |
| | CASSETTE. | 4-11 |
| | CLEAR | 4-12 |
| | COMPLEMENT. | 4-13 |
| | COUNT | 4-15 |
| | DEC | 4-16 |
| | DEFINE. | 4-17 |
| | DEFINE-VALUE. | 4-18 |
| | DISPATCH. | 4-19 |
| | EOR | 4-21 |
| | EXIT. | 4-22 |
| | EXTRACT | 4-23 |
| | GO TO | 4-25 |
| | HALT. | 4-26 |
| | IF. | 4-27 |
| | INC | 4-33 |
| | JUMP. | 4-34 |
| | LIT | 4-35 |
| | LOAD. | 4-36 |
| | LOAD-MSMA | 4-37 |
| | LOAD-SMEM | 4-39 |
| | MACROS. | 4-40 |
| | MICRO | 4-42 |
| | MOVE | 4-43 |

- TABLE OF CONTENTS (cont)

| SECTION | TITLE | PAGE |
|------------|---------------------------------------|------|
| 4 (cont) | NOP | 4-46 |
| | NORMALIZE | 4-47 |
| | OR. | 4-48 |
| | OVERLAY | 4-49 |
| | READ. | 4-50 |
| | RESET | 4-51 |
| | ROTATE OR SHIFT T | 4-52 |
| | ROTATE OR SHIFT X, Y and XY | 4-54 |
| | SEGMENT | 4-55 |
| | SET | 4-56 |
| | SKIP. | 4-58 |
| | STORE | 4-60 |
| | SUBTRACT SCRATCHPAD | 4-61 |
| | SWAP. | 4-62 |
| | TABLE | 4-63 |
| | WRITE | 4-64 |
| | WRITE-STRING. | 4-65 |
| | XCH | 4-66 |
| 5 | PROGRAMMING TECHNIQUES. | 5-1 |
| | Virtual Language Definitions. | 5-1 |
| | Writing Rules | 5-1 |
| APPENDIX A | MIL COMPILER OPERATION | A-1 |
| | Compiler Control Card | A-1 |
| | The Module Option \$ Card. | A-3 |
| APPENDIX B | B 1710 HARDWARE TABLES. | B-1 |
| | B 1710 Register Addressing | B-1 |
| | Condition Registers | B-1 |
| APPENDIX C | B 1720 HARDWARE TABLES. | C-1 |
| | B 1720 Register Addressing | C-1 |
| | Condition Registers | C-1 |

TABLE OF CONTENTS (cont)

| SECTION | TITLE | PAGE |
|------------|---|------|
| APPENDIX D | MICRO INSTRUCTIONS | D-1 |
| APPENDIX E | I/O RESULT DESCRIPTORS | E-1 |
| APPENDIX F | MIL ERROR AND WARNING MESSAGES | F-1 |
| | MIL Error Messages | F-1 |
| | MIL Warning Messages | F-2 |
| APPENDIX G | B 1700 HARDWARE INSTRUCTION FORMATS | G-1 |
| | BIAS | G-1 |
| | BIT TEST RELATIVE BRANCH FALSE | G-2 |
| | BIT TEST RELATIVE BRANCH TRUE | G-3 |
| | BRANCH RELATIVE | G-4 |
| | CALL | G-5 |
| | CASSETTE CONTROL | G-6 |
| | CLEAR REGISTERS | G-7 |
| | COUNT FA/FL | G-8 |
| | DISPATCH | G-9 |
| | EXTRACT FROM REGISTER T | G-11 |
| | HALT | G-12 |
| | LOAD F FROM DOUBLEPAD WORD | G-13 |
| | MANIPULATE 4-BIT | G-14 |
| | MOVE 8-BIT LITERAL | G-15 |
| | MOVE 24-BIT LITERAL | G-16 |
| | NO OPERATION | G-17 |
| | NORMALIZE X | G-18 |
| | OVERLAY CONTROL MEMORY | G-19 |
| | READ/WRITE MEMORY | G-20 |
| | REGISTER MOVE | G-21 |
| | SCRATCHPAD MOVE | G-22 |
| | SCRATCHPAD RELATE | G-23 |
| | SET CYF | G-24 |
| | SHIFT/ROTATE REGISTER T LEFT | G-25 |

TABLE OF CONTENTS (cont)

| SECTION | TITLE | PAGE |
|----------------------|--|------|
| APPENDIX G (cont) | SHIFT/ROTATE REGISTERS X AND Y LEFT/RIGHT | G-26 |
| | SHIFT/ROTATE REGISTER X OR Y LEFT/RIGHT | G-27 |
| | SKIP WHEN | G-28 |
| | STORE F INTO DOUBLEPAD WORD | G-29 |
| | SWAP DOUBLEPAD WORD WITH REGISTER F | G-30 |
| | SWAP MEMORY | G-31 |
| APPENDIX H | MICRO PROCESSOR TIMING TABLES | H-1 |
| | B 1710 Micro Instruction Timing | H-1 |
| | B 1720 Micro Instruction Timing | H-3 |
| APPENDIX I | RESERVED WORDS AND SYMBOLS | I-1 |
| APPENDIX J | GLOSSARY | J-1 |
| INDEX | | one |

LIST OF ILLUSTRATIONS

| FIGURE | TITLE | PAGE |
|--------|----------------------------|------|
| B-1 | B 1710 Processor | B-3 |

LIST OF TABLES

| TABLE | TITLE | PAGE |
|-------|---|------|
| 4-1 | AND Truth Table | 4-5 |
| 4-2 | EOR Truth Table | 4-21 |
| B-1 | B 1710 Register Addressing | B-1 |
| B-2 | Condition Registers | B-1 |
| C-1 | B 1720 Register Addressing | C-1 |
| C-2 | Condition Registers | C-1 |
| H-1 | B 1710 Micro Instruction Timing | H-1 |
| H-2 | B 1720 Micro Instruction Timing | H-3 |

INTRODUCTION

The Burroughs Micro Implementation Language (MIL) is a symbolic coding technique that makes available all the capabilities of the B 1700 Processor. MIL assumes interpretive or indirect processing of information contained in main memory.

The machine language output from the MIL compiler is ready for execution directly upon the hardware. The user, however, must be prepared to programmatically control the total environment including bootstrap loading, interrupt servicing, and potential machine malfunctioning (i.e., parity error detection).

To use the MIL compiler properly and efficiently, the programmer must have an extensive knowledge of the available registers and their capabilities. This manual describes the registers, the syntax, and the semantics of the MIL language and may be used to write programs without prior knowledge of the system. A description of the Input/Output subsystem and the I/O descriptors as well as more detailed information about the registers will, however, be found in the B 1700 Systems Reference Manual (form 1057155).

SECTION 1
MICRO PROGRAMMING CONCEPTS

GENERAL.

Micro programming has been defined as "a means for programming a computer hardware architecture".(*1) The micro programmer is concerned with machine registers which were formerly the domain of the hardware systems designer. Strings of micro instructions manipulate those internal registers to present an outward appearance of system hardware which is more functional for problem programming. In most machines now in the market place, read only memories (ROM's) contain the micro programs which convert the unique internal environment of several different processors into a standard assembly language. Once created, the micro programs are unalterable and contain many compromises in their attempt to do everything within limited space and a single problem.

The Burroughs B 1700 system makes use of the latest technology to implement a writable control memory and has several micro programs, each optimized for the functions it will perform. The virtual system architectures chosen have been those of the standard (COBOL and FORTRAN), problem-oriented, compiler languages. Other micro programmers may choose architectures and create languages optimized for other purposes.

MICRO INSTRUCTIONS.

A micro instruction is the smallest programmable operation within the system. Each micro instruction is fetched from control memory and decoded in the (micro) register to be directly executed by the hardware.

1 The definition is by Professor M. V. Wilkes, Cambridge University. Computer Inaugural Conference 1951, pp 16-21.

DEFINED FIELD CONCEPTS.

A defined field concept allows the addressing of data in main memory to the individual bit and to be from one to 65,535 bits in length. There are no visible boundaries or "best" container size for any information contained in main memory. Virtual machine instruction strings (formerly called machine object code) and their data may thus be densely packed into meaningful fields, saving considerable memory space. The programming problem of packing and unpacking data fields across hardware container boundaries is completely resolved, saving much programming effort and processor time. The micro program fetches groups of bits in meaningful field sizes from anywhere in main memory as needed.

Special hardware, called a Field Isolation unit, has been implemented to achieve bit addressability and variable length fields and to automatically increment addresses, allowing for rapid iteration.

INTERPRETATION OF THE VIRTUAL LANGUAGE.

The traditional approach to supporting a higher-level language is to translate the source statements as written by the programmer into another language either directly recognized by the hardware, e.g., machine object code, or easily translatable into the machine object code, e.g., an assembly language. An alternate technique is the interpretive execution for each source statement with a logically equivalent routine in some lower-level language. A micro programmed system offers the opportunity to combine the best of both methods. The source statements in the higher-level language are transferred into a virtual system code by a compilation process. This system code, also called S-code or S-language, very closely resembles the original source language. Micro instruction routines then interpretively execute each virtual language statement. The results are: (1) a faster compilation, (2) a system architecture, as expressed in the set of micro routines, which is optimized to the source language,

(3) a reduction in the processor time required to perform the logical equivalent of each source statement, and (4) a reduction in the memory space required to encode each source language operation.

A set of micro programmed routines is called an interpreter and effectively creates a virtual system architecture for the source language being executed. That is, when the COBOL interpreter is executing, the system is effectively a COBOL machine. When the FORTRAN interpreter is executing, the system is a FORTRAN machine, and so on for any other S-language defined.

SECTION 2
MICRO IMPLEMENTATION LANGUAGE

ASSEMBLY CODING FORM.

The source program cards have the following format:

| <u>Card Columns</u> | <u>Usage</u> |
|---------------------|--|
| 1-5 | Reserved for label declarations which, if used, must begin somewhere within this field. |
| 1-72 | An asterisk (*) anywhere within this field indicates that everything to the right is a comment. |
| 6-72 | Instructions may appear anywhere within this field. At least one blank must be used between words except in those cases where a special character, e.g., a parenthesis or a relational operator, is required, in which case blanks are optional. |

Example:

```
EXTRACT 7 BITS FROM T(11) TO Y  
SKIP WHEN X = Y
```

73-80 This field is reserved for sequence numbers.

NOTE

Only one MIL source language
instruction is allowed per card.

CONDITIONS.

Whenever the phrase "condition-1" appears in an instruction syntax, it is to be replaced by any condition whose truth or falsity can be determined by testing one or more bits in one of the condition

registers. Therefore, any of the following may be used in the IF and SKIP statements:

NOTE

The "/" is to be read as the word "OR."

X = /</>/≤/≥/≠/EQL/LSS/GRT/LEQ/GEQ/NEQ Y

X = /≠/EQL/NEQ 0

Y = /</>/≤/≥/≠/EQL/LSS/GRT/LEQ/GEQ/NEQ X

Y = /≠/EQL/NEQ 0

ANY-INTERRUPT

FL = /</>/≤/≥/≠/EQL/LSS/GTR/LEQ/GEQ/NEQ/SFL

FL = /≠/EQL/NEQ 0 (zero)

CYL (CARRY OUT LEVEL)

CYD (BORROW OUT LEVEL)

LSUY (LEAST SIGNIFICANT UNIT OF Y)

MSBX (MOST SIGNIFICANT BIT OF X)

LSUX (LEAST SIGNIFICANT UNIT OF X)

LOCKOUT

INTERRUPT

HI-PRIORITY

NO-DEVICE

Any combination of conditions that is contained in one condition register can be tested using AND/OR logic if all bits can be tested for TRUE (ON) or FALSE (OFF). For example, the following are all valid conditions:

CYL AND LSUY

CYL OR CYD

Examples:

If CYL and LSU TRUE then go to END

If CYL or CYD FALSE then go to BEGIN

Labels are declared as the first item on a card and must begin somewhere in columns 1 through 5. Any number of labels may be declared for the same address, but one card is required for each label declaration.

Acceptable label characters are alpha A through Z, numeric 0 through 9, and the special character dash (-). A dash is not acceptable as the first character of a label.

Any part of a label beyond the twenty-fifth character is considered documentation only. A label may appear on a separate card immediately preceding the source statement that it references.

There are two types of labels which are acceptable to the compiler, unique labels and point labels. Unique labels are not reusable and therefore must be made unique within the 25 characters. Point labels are reusable labels which are usually, but not necessarily, used for short distance branching. Point labels are declared with a first character of "." and are referenced either + or - from the location of the present instruction.

The following is an example of point label usage:

```
.ABC  READ TO X INC FA
      SKIP WHEN X NEQ Y
      GO TO -ABC
      GO TO +ABC
      . . .
      . . .
      . . .
      .ABC  MOVE SUM TO L
```

The -ABC refers to the READ instruction and the +ABC refers to the MOVE instruction.

LITERALS

Whenever the phrase "literal" appears in the text of this manual, it is to be replaced by a decimal integer or a hexadecimal number whose first character is an identifying H, a binary number whose first character is an identifying B, or a STRING.

Characters which are valid for use as a literal are dependent upon the type of literal.

| <u>Type</u> | <u>Valid Characters</u> |
|------------------|---|
| Decimal | Numeric 0 thru 9 |
| Hexadecimal | Numeric 0 thru 9 and ALPHA A, B, C, D, E and F, representing 10, 11, 12, 13, 14 and 15 respectively |
| Binary | Numeric 0 and 1 |
| Character-string | Any valid EBCDIC character |
| Bit-string | @ (1) 11001100 @ |

Literals used in the MIL syntax have a maximum range of decimal 0 thru 16777215, which is equal to hexadecimal H0 thru HFFFFFF. A binary literal may not contain more than 25 characters; therefore, B0 thru B11111111111111111111111111111111 is the maximum range of a binary literal. Literals may also be character-strings if the characters are surrounded by quote symbols.

Leading zeros are not required for literals unless an actual value of zero is required; then, either 0, H0, or B0 must be used.

A character may not be embedded between the identifying H or B of a hexadecimal or binary literal and the number string that follows. Special characters, such as commas, may not be embedded in any decimal, hexadecimal or binary literal. It is possible, however, to encode any EBCDIC character within a character string.

Wherever a literal appears in this manual, a string of appropriate type and length may be substituted.

Character strings must be surrounded by quote symbols (") and represent EBCDIC characters. Imbedded percent signs (%) and quote symbols (") are not allowed. The percent sign, like the asterisk, designates the rest of the card as a comment.

Bit strings must be surrounded by "AT" symbols (@) and must contain a bit grouping length indicator. This indicator must immediately follow the first @ sign and be enclosed in parentheses. The possible bit groupings are 1, 2, 3, and 4. If the length indicator is omitted, 4 is assumed.

| <u>Type</u> | <u>Start-Stop Symbol</u> | <u>Length of Each Unit</u> | <u>Example</u> |
|-------------|------------------------------|--------------------------------|------------------|
| Character | " | 8 bits | "THIS IS ONE" |
| Hex | @ | 4 bits | @E3C8D1E2@ |
| Octal | @(3) | 3 bits | @(3)423765@ |
| Quartal | @(2) | 2 bits | @(2)123132113@ |
| Binary | @(1) | 1 bit | @(1)10011100110@ |

SECTION 3
REGISTERS

GENERAL.

This section contains a brief description of the registers within the processor. The registers are divided into logical groups as follows:

- Active
- Result
- Scratchpad
- Constant
- Input/Output
- Condition

A Register Address table and Condition Address table may be found for each of the processors along with processor flow diagrams and micro instruction decoding charts in the appendices.

This section is intended only as an overview of the registers within the processor. It is assumed that the reader is familiar with the contents of the B 1700 Systems Reference Manual(form 1057155).

An alphabetic listing of all registers follows.

ALPHABETICAL LISTING OF REGISTERS.

| <u>Name</u> | <u>Length in Bits</u> | <u>Note</u> |
|-------------|---------------------------|--|
| A | 20 | Control memory micro instruction address |
| BICN | 4 | Boolean Conditions |
| BR | 24 | Base register or low address S-memory protection |
| C | 24 | Control; not addressable as a unit |
| CA | 4 | Subfield of C |

| <u>Name</u> | <u>Length in Bits</u> | <u>Note</u> |
|----------------|---------------------------|--|
| CB | 4 | Subfield of C |
| CC | 4 | Subfield of C; interrupts and flags |
| CD | 4 | Subfield of C; interrupts and flags |
| CMND | 24 | I/O command register on B 1710/B 1720 series |
| CMPX | 24 | Result: Complement of X; masked by CPL |
| Control Memory | 16-bit words | Location of micro instructions |
| CP | 8 | Control parallel subfield of C |
| CPL | 5 | Control parallel length; subfield of CP |
| CPU | 2 | Control parallel unit; subfield of CP destination only |
| CYD | 1 | Carry Difference or Carry of Borrow |
| CYF | 1 | Carry flip-flop; subfield of CP |
| CYL | 1 | Carry latch or carry of sum |
| DATA | 24 | I/O Data register on B 1710/B 1720 series |
| DIFF | 24 | Result of $X - (Y + CYF)$; masked by CPL |
| F | 48 | Field in S-memory; FA and FB concatenated |
| FA | 24 | Field address in S-memory |
| FB | 24 | S-memory Field unit (FU), Field type (FT), and Field length (FL) |
| FL | 16 | Field length in S-memory |
| FT | 4 | Subfield of FB |
| FLC | 4 | Subfield of FL |
| FLD | 4 | Subfield of FL |

| <u>Name</u> | <u>Length in Bits</u> | <u>Note</u> |
|-------------|---------------------------|---|
| FLE | 4 | Subfield of FL |
| FLF | 4 | Subfield of FL |
| FLCN | 4 | Boolean Field length conditions |
| FU | 4 | S-memory unit size; subfield of FB |
| INCN | 4 | Boolean dispatch interrupt conditions B 1730 series |
| L | 24 | Local register also used in Dispatch and Overlay |
| LA | 4 | Subfield of L |
| LB | 4 | Subfield of L |
| LC | 4 | Subfield of L |
| LD | 4 | Subfield of L |
| LE | 4 | Subfield of L |
| LF | 4 | Subfield of L |
| LR | 24 | Limit register or high address S-memory protection |
| M | 16 | Current micro instruction register |
| MBR | 24 | Main memory micro instruction Base register; not on B 1710 and B 1720 series |
| MAXS | 24 | A constant; size in bits of available S-memory |
| MSKX | 24 | Result; mask of X; length by CPL |
| MSKY | 24 | Result; mask of Y; length by CPL |
| MSM | 16 | Only in TAPE mode - Control Memory addressed by the A register. B 1720 and B 1730 |

| <u>Name</u> | <u>Length in Bits</u> | <u>Note</u> |
|----------------|---------------------------|---|
| Main Memory | - | S-memory |
| Null | 24 | Always zero |
| SFL | 20 | Subfield of S zero B corresponding to FL in FB |
| S0 | 48 | Scratchpad double word |
| S1 | 48 | Scratchpad double word |
| . | . | Scratchpad double word |
| . | . | Scratchpad double word |
| . | . | Scratchpad double word |
| S15A-S15B | . | Subfields of S15 |
| S-memory | - | Main memory |
| SU | 4 | Subfield of SOB corresponding to FL in FB |
| SUM | 24 | Result (X + Y + CYF) length by CPL |
| T | 24 | Transform - will rotate, shift or extract bits |
| TAS | 24 | Top of A register-stack |
| TA | 4 | Subfield of T |
| TB | 4 | Subfield of T |
| TC | 4 | Subfield of T |
| TD | 4 | Subfield of T |
| TE | 4 | Subfield of T |
| TF | 4 | Subfield of T |
| TOPM | 4 | Top of control memory not on B 1710 series |

| <u>Name</u> | <u>Length in Bits</u> | <u>Note</u> |
|-------------|---------------------------|--|
| U | 16 | Cassette input only |
| X | 24 | Input to function box (Result registers) |
| XANY | 24 | Result; X and Y; length by CPL |
| XORY | 24 | Result; X or Y; length by CPL |
| XY | 48 | X and Y concatenated |
| XYCN | 4 | Boolean XY conditions |
| XYST | 4 | Boolean XY states |
| Y | 24 | Input to function box (Result registers) |

NOTE

The most-significant (left-most) bit in any register is identified in the MIL syntax as bit 0 (zero), the next most-significant as bit 1, etc. This is particularly advantageous in a bit-addressable machine since, for software purposes, it is often desirable to think of a register as being an extension of main memory. It should be noted that this convention is at variance with the hardware bit numbering convention where, generally, all bits are numbered right to left, 0 through 23. This difference has particular significance when any bit data is to be ORed into the M register at run time.

ACTIVE REGISTERS.

The following is a description of the active registers.

X-Y REGISTERS.

The X and Y registers (both of which are 24 bits wide) are used as inputs into the 24-bit function box. All functions are performed under control of the C (Control) register, which regulates the length of the operation, class of arithmetics, and least-significant carry input. The X and Y registers are capable of being shifted or rotated and may receive or transmit data to or from main memory.

FIELD (F) REGISTER.

The F register is divided into FA and FB, each sub-register being 24 bits wide. The FA (Field Address) portion is used to address main memory. FB is divided into FU (Field Unit), four bits indicating arithmetic unit size; FT (Field Type), a general-purpose 4-bit field; and FL (Field Length), consisting of 16 bits used to indicate the length of fields in main memory. FL is subdivided into FLC, FLD, FLE and FLF, each four bits in length.

LOCAL (L) REGISTER.

The L register is 24 bits wide and is subdivided into LA, LB, LC, LD, LE and LF, each four bits in length. L and its subdivisions are generally used to temporarily hold the contents of other processor registers. It is also used as a source and destination for main memory access and has implicit use in the DISPATCH and OVERLAY micro instructions.

TRANSFORM (T) REGISTER.

The T register is a 24-bit transformation register used extensively for interpretation of virtual-language operators. It is subdivided into TA, TB, TC, TD, TE and TF, each four bits in length. T has strong shift and extract logics associated with it and is the principal formatting register of the processor. This register also has the capability of receiving or transmitting data from and to main memory.

MICRO INSTRUCTION (M) REGISTER.

The M register is a 16-bit register which holds the micro operator for decoding and subsequent execution by the hardware.

BASE (BR) AND LIMIT (LR) REGISTERS.

The BR and LR registers are each 2⁴ bits wide and are used to hold the main memory base and limit addresses for the currently active main memory process. The hardware(*2) uses these registers to determine if addresses in the Field Address (FA) register are within the base/limit boundaries.

ADDRESS (A) REGISTER.

The A register is the micro program address register. It is an 18-bit register which contains the bit address of the next micro instruction. Values in the A-register are always mod 16; i.e., the low-order four bits are always zero. It is capable of addressing 16,384 micro instructions located in either control memory or main memory or both. The A register is automatically incremented to the next micro instruction before the current micro instruction is executed. It is also capable of having any value from 0 to 4095 added to or subtracted from it to facilitate micro code branching.

NOTE

The displacement values in micro instructions will be multiplied by 16 before being added to the A register. Wraparound may occur and is permitted.

A STACK.

The A stack is a 32-element-deep, 2⁴-bit-wide, push-down memory, i.e., a last-in-first-out (LIFO) storage structure. The A stack is used to nest micro routine linkages and allows highly shared routines, thus

2 This is not available on the B 1710 series.

reducing control memory requirements. Although the A stack was intended for micro code addresses, it has been made 24 bits wide to allow for any operand storage.

NOTE

The B 1710 A stack has only 16 storage elements.

TOP OF CONTROL MEMORY (TOPM) REGISTER. (*3)

The TOPM register is four bits wide and is used to determine which memory (control or main) contains the next micro instruction. If the A register is equal to or greater than (512 X TOPM), the next micro instruction will be fetched from main memory rather than control memory. The TOPM register is addressable as a source or as a destination.

MEMORY BASE REGISTER (MBR). (*3)

The MBR register is used with the A register to obtain the main memory address of the next micro instruction. The A register is added to the 24 bits of MBR to yield a 24-bit main memory address. The MBR register is addressable as both a source and as a destination.

CONTROL (C) REGISTER.

The C register is a 24-bit control register for the micro processor. It contains the 24-bit function box controls and carry input plus some of the processor interrupts and flags. It is subdivided into CA, CB, CC, CD, each of four bits, and CP, of eight bits. CA and CB may be used as general-purpose registers. CC and CD represent processor interrupts and flags. CP contains function box controls: CYF (0 bit of CP), CPU (1 and 2 bits of CP), and CPL (3,4,5,6, and 7 bits of CP). CYF (Carry Flip Flop) notifies the function box that a previous unit carry must be added to its summary results. CPU (Control Parallel

3 This is not available on the B 1710 series.

Unit) notifies the function box of the type of unit contained in X and Y: 00 = binary, 01 = 4-bit decimal, and 11 = 8 bit decimal. CPL (Control Parallel Length) specifies the width, in bits, of the function box and READ/WRITE micro instructions.

COMBINATORIAL LOGIC OR FUNCTION BOX.

The combinatorial logic, often called the function box, produces the Result registers for the processor. Inputs are the X register, the Y register and the Carry flip-flop (CYF). The inputs are combined under control of the Control Parallel Unit (CPU) register and the Control Parallel Length (CPL) register. The Result registers are available to the immediately following micro instruction after a change to one of the input registers.

RESULT REGISTERS.

The Result registers are outputs from the 2⁴-bit function box. Their contents are produced immediately and automatically from the inputs to the function box (X, Y and CYF) and cannot be changed except by changing inputs or by changing CPU (Control Parallel Unit) or CPL (Control Parallel Length). If the value of CPL is less than 2⁴, then the 2⁴ minus CPL most-significant bits of all result registers will be zero. These registers are source registers only and therefore cannot be used as the destination register in a MOVE or in any other instruction.

XORY RESULT REGISTER.

This register ORs the contents of the X register inclusively with the contents of the Y register. This is a bit by bit operation, with corresponding pairs of bits treated independently.

XANY RESULT REGISTER.

This register ANDs the X register with the Y register. This is the logical product of the X register and the Y register. Corresponding pairs of bits are treated independently.

XEOY RESULT REGISTER.

This register ORs the contents of the X register exclusively with the contents of the Y register.

CMPX RESULT REGISTER.

This register is the 1's complement of the X register.

CMFY RESULT REGISTER.

This register is the 1's complement of the Y register.

MSKX RESULT REGISTER.

MASKED X is the low-order bits of the X register. The value of the CPL determines the number of bits placed in MSKX. All other high-order bits are zero. IF CPL is equal to 2^4 , then MSKX is identical to the X register.

MSKY RESULT REGISTER.

MASKED Y is the low-order bits of the Y register. The value of CPL determines the number of bits placed in MSKY. All other high-order bits are zero. If CPL is equal to 2^4 , MSKY is identical to the Y register.

SUM RESULT REGISTER.

SUM is the decimal or binary value (determined by CPU) of the X register plus the Y register plus the CYF register. Corresponding pairs of bits are grouped by CPU control; and groupings may be binary, decimal four bit or eight bit. If the sum of (X+Y+CYF) is larger than the size specified by CPL, then the CYL (carry latch) will be TRUE (ONE). The zone bits are ORed together for 8-bit decimal.

NOTE

The 8-bit decimal SUM is not available on the B 1710 series.

DIFFERENCE RESULT REGISTER.

This register stores the amount resulting from the subtraction of the sum of the contents of the Y and CYF registers from the contents of the X register. The contents of the CPU register determine whether the subtraction is decimal or binary. Corresponding pairs of bits are grouped by CPU. If the difference is negative, $X - (Y + CYF) < 0$, then DIFF result will be in 2's complement form or 10's complement form depending upon the mode, either binary or decimal respectively; and CYD (Carry Difference) will be TRUE (ONE). The zone bits are ORed together for 8-bit decimal.

NOTES

1. The 8-bit decimal difference is not available on the B 1710 series.
2. The CYD register is not conditioned by CPL; it is always based on a 24-bit comparison. The programmer, therefore, must know what is in the high-order positions of the X register and the Y register if CPL is less than 24.

SCRATCHPAD.

The processor makes use of the scratchpad for temporary storage of active registers. The scratchpad may be addressed as sixteen, 48-bit double words or thirty-two, 24-bit words.

SCRATCHPAD WORDS - 24 BITS EACH.

| | | | |
|-----|-----|------|------|
| SOA | S4A | S8A | S12A |
| SOB | S4B | S8B | S12B |
| S1A | S5A | S9A | S13A |
| S1B | S5B | S9B | S13B |
| S2A | S6A | S10A | S14A |

| | | | |
|-----|-----|------|------|
| S2B | S6B | S10B | S14B |
| S3A | S7A | S11A | S15A |
| S3B | S7B | S11B | S15B |

DOUBLE SCRATCHPAD WORDS - 48 BITS EACH.

(S# = S#A and S#B CONCATENATE, WHERE # = 0 THROUGH 15)

| | | | |
|----|----|-----|-----|
| S0 | S4 | S8 | S12 |
| S1 | S5 | S9 | S13 |
| S2 | S6 | S10 | S14 |
| S3 | S7 | S11 | S15 |

CONSTANT REGISTERS.

The following is a description of the constant registers.

MAXIMUM MAIN MEMORY REGISTER.

The MAXS register is set by the field engineer and contains the value of the maximum installed number of main memory bits. It is addressable as a source only. Main memory addresses begin at zero.

MAXIMUM CONTROL MEMORY REGISTER.

The MAXM register is set by a field engineer and contains the value of the maximum installed number of control memory words, each word comprising 16 bits. It is addressable as a source only. On the B 1710, it will always contain zero.

NULL REGISTER.

The Null register is a source-only, 24-bit, addressable field of zeros.

INPUT/OUTPUT REGISTERS.

The following is a description of the input/output registers.

CONSOLE CASSETTE TAPE INPUT (U) REGISTER.

The U register accumulates the data read from the tape cassette on the console control panel. It is addressable as a source in the RUN mode with the MOVE REGISTER micro instruction and in the TAPE mode with the

MOVE 24-BIT LITERAL micro instruction. (Refer to the LOAD-MSMA micro instruction.) It is not addressable as a destination.

COMMAND REGISTER. (*4)

The CMND register is used to transfer commands to the I/O controllers. It is 24 bits wide and is addressable as a destination only.

DATA REGISTER. (*4)

The Data register is used to transfer data to and from the I/O controllers and their peripherals. It is 24 bits wide and is addressable as a source or as a destination.

NOTE

The Data register may require attention whenever the Interrupt Condition register is SET.

CONDITION REGISTERS.

There are five condition registers:

Binary Conditions (BICN)

Field Length Conditions (FLCN)

Interrupt Conditions (INCN)

X AND/OR Y Register(s) Conditions (XYCN)

X AND/OR Y Register(s) States Conditions (XYST)

Each condition register consists of four bits. The bits are identified from left to right and are assigned the position numbers 0 thru 3, with 0 being the most-significant bit.

All condition registers are source registers only. They may be moved to another register or tested, using the IF and SKIP instructions, for their current contents. They may not be the destination register of any instruction.

⁴ This is active on the B 1710/B 1720 series only.

| <u>Bit</u> | <u>BICN</u> | <u>XYCN</u> | <u>XYST</u> | <u>FLCN</u> | <u>INCN</u> |
|------------|-------------|-------------|---------------|-------------|-------------|
| 0 | LSUY | MSBX | LSUX | FL=SFL | NO-DEVICE |
| 1 | CYF | X=Y | ANY-INTERRUPT | FL>SFL | HI-PRIORITY |
| 2 | CYD | X<Y | X≠0 | FL<SFL | INTERRUPT |
| 3 | CYL | X>Y | X≠0 | FL≠0 | LOCKOUT |

BINARY CONDITIONS (BICN) REGISTER.

LSUY is TRUE if the least-significant unit of the Y register is: (1) 1 and the Control Parallel Unit (CPU) register specifies binary (CPU = 0) or (2) 9 and the Control Parallel Unit (CPU) register specifies other than binary (CPU ≠ 0).

The CYF register is the carry in the Control Parallel register. The CYF register may be manipulated as part of the CP register and by the CARRY instruction.

The Carry Difference (CYD) register is TRUE if $X - (CYF + Y) < 0$. This condition is not affected by CPL, i.e., a 2⁴-bit compare is always made.

The Carry Level (CYL) register is TRUE if $(X+Y+CYF)$, limited by the Control Parallel Length (CPL) register, overflows.

XY CONDITIONS (XYCN) REGISTER.

MSBX is TRUE if the most-significant bit of the X register, as determined by the Control Parallel Length (CPL) register, is a 1.

NOTE

The comparisons of the X register to the Y register are not affected by CPL; they are always 2⁴-bit compares.

XY STATES (XYST) REGISTER.

LSUX is TRUE if the least-significant unit of the X register is: (1) 1 and the Control Parallel Unit (CPU) register specifies binary (CPU = 0) or (2) 9 and the Control Parallel Unit (CPU) register specifies other than binary (CPU \neq 0). The comparisons of the X register or the Y register to zero are not affected by CPL; all 24 bits of the X register and/or the Y register are used in the comparisons.

ANY-INTERRUPT.

This bit is TRUE if any of the following conditions in registers CC, CD, or INCN (B 1730 only) are TRUE:

| <u>Event</u> | <u>Register</u> (Bit Position) |
|---|--------------------------------|
| Missing device | INCN(0) |
| Port interrupt | INCN(2) |
| Timer interrupt | CC(1) |
| I/O service request interrupt | CC(2) |
| Console interrupt | CC(3) |
| Main memory READ parity error interrupt | CD(0) |
| Memory WRITE/SWAP address out of bounds interrupt | CD(3) |

The CC and CD registers are both 4-bit source and destination registers within the C register. The bits in each are numbered 0 through 3, with bit 0 being the most significant. They have been assigned the following uses and meanings:

| | |
|-------|-------------------------------|
| CC(0) | State light |
| CC(1) | Timer interrupt |
| CC(2) | I/O service request interrupt |
| CC(3) | Console interrupt |

CD(0) Main memory parity error
CD(1) Main memory WRITE/SWAP error override
CD(2) Main memory READ error
CD(3) Main memory WRITE/SWAP error

All bits in the CC and CD portions of the C register, once SET, remain SET even though the conditions that caused them to be SET may no longer exist. Therefore, if it is desired to clear any of these bits to ZERO; this must be done explicitly. CD(1), CD(2), and CD(3) of the C register are always ZERO in the B but still may be addressed and tested.

CONSOLE INTERRUPT - (CC(3)).

This bit is SET when the INTERRUPT toggle switch on the console control panel is turned ON. It remains SET as long as the switch is ON.

MAIN MEMORY READ PARITY ERROR INTERRUPT - (CD(0)).

This bit is SET when a main memory parity error is detected during a READ or a READ portion of a SWAP operation or when an attempt is made to access non-existent main memory.

MAIN MEMORY ADDRESS OUT OF BOUNDS OVERRIDE - (In CD(1)). (*5)

This bit is tested if the Field Address (FA) register setting is less than the Base register (BR) setting or greater than the Limit register (LR) setting; then WRITE or SWAP operations will be inhibited unless this bit is SET (ONE). The state of this bit does not affect the setting of CD(2) or CD(3).

READ ADDRESS OUT OF BOUNDS INTERRUPT - (In CD(2)(*5)). (*5)

This bit is SET when a READ operation is attempted and the Field Address (FA) register setting is either less than the Base register

5 This is not available on the B 1710 series.

(BR) setting or greater than the Limit register (LR) setting. The READ operation is not inhibited.

WRITE/SWAP ADDRESS OUT OF BOUNDS INTERRUPT - (CD(3)). (*6)

This bit is SET when a WRITE or SWAP operation is attempted and the Field Address (FA) register setting is either less than the Base register (BR) setting or greater than the Limit register (LR) setting.

FIELD LENGTH CONDITIONS (FLCN) REGISTER.

All conditions are based upon comparisons between the 16 bits of the FL register and either zero or the corresponding low-order 16 bits (SFL) of the first word in the scratchpad (SOB).

INTERRUPT CONDITIONS (INCN) REGISTER. (*6)

NO-DEVICE is TRUE if an interrupt message is present in the Dispatch buffer for a port or channel which does not have a device attached to it. This condition is normally cleared by the processor with a DISPATCH READ AND CLEAR instruction.

HI-PRIORITY is TRUE if there is a high-priority message present in the Dispatch buffer.

INTERRUPT is TRUE if there is a message present in the Dispatch buffer for the processor. This condition is normally cleared by a DISPATCH READ AND CLEAR instruction.

LOCKOUT is TRUE if the interrupt system is locked (marked as "in use").

REGISTER DESIGNATIONS AND AREAS OF APPLICATION.

The following is a list, arranged by areas of application, of registers and their associated designations.

⁶ This is not available on the B 1710 series.

MICRO INSTRUCTION CONTROLS.

A (Micro Instruction Address)
M (Current Micro Instruction)
TAS (Top of Address Stack)
TOPM(*7) (Top of M Memory)
MBR(*7) (Micro Instruction Base register)

S-MEMORY CONTROLS.

BR (Base register)
LR (Limit register)
FA, FL (Field Address, Field Length)
CP (Control Parallel)

INTERRUPT CONTROLS.

CC
CD
INCN(*7)

PARALLEL WIDTH CONTROLS.

C
CP
CPL
CPU

ORGANIZATION OF FIELDS AND SUBFIELDS.

| <u>Field</u> | <u>Subfields</u> |
|--------------|------------------|
| C | CA CB CC CD CP |
| CP | CYF CPU CPL |

7 This is not available on the B 1710/B 1720 series.

| <u>Field</u> | <u>Subfields</u> |
|--------------|-------------------|
| F | FA FB |
| FB | FU FT FL |
| FL | FLC FLD FLE FLF |
| L | LA LB LC LD LE LF |
| T | TA TB TC TD TE TF |

NOTE

C does not exist as a composite, only as subfields.

SECTION 4
MICRO OPERATORS

NOTATIONS AND CONVENTIONS USED IN THE SYNTAX.

The following details the notations and conventions used in the syntax.

KEY WORDS.

All capital words in the syntax of a micro instruction are required. They must appear in the positions as given. An omission or misplacement will result in a syntax error.

Example:

$$\text{BIAS BY} \left\{ \begin{array}{l} \text{UNIT} \\ \text{register-1} \end{array} \right\} [\text{AND register-2}] [\text{TEST}]$$

The key words are BIAS, BY, UNIT, AND, and TEST.

LOWER CASE WORDS.

Lower case words represent generic terms which must be supplied in that format position by the programmer. In the example above, the words "register-1" and "register-2" are generic; and register names must be supplied.

BRACES.

When words or phrases are enclosed in braces {}, a choice of one of the enclosed words or phrases must be made. In the example above, a choice between the word "UNIT" or "register-1" must be made.

BRACKETS.

When a word or phrase is enclosed in brackets [], the word or phrase is an optional feature of the instruction. The programmer may include the feature by coding according to the syntax enclosed in the brackets or he may omit the feature completely.

CONCURRENT EXECUTION OF MICRO OPERATORS. (*8)

Concurrent or overlapping execution of micro operators can be achieved if a READ, WRITE, or SWAP instruction is followed by any of the following instructions:

BIAS
COUNT
GO TO
JUMP
LOAD
NOP
READ
STORE
SWAP
WRITE
XCH

NOTE

Consecutive READ, WRITE, or SWAP operations will save at least one clock cycle per operation.

8 This is not available on the B 1710 series.

FORMAT 1:

| | |
|--------------------|---------|
| ADJUST LOCATION TO | literal |
|--------------------|---------|

FORMAT 2:

| | | |
|-----------------------------|--|---------|
| ADJUST LOCATION TO LOCATION | $\left. \begin{matrix} \text{PLUS} \\ + \end{matrix} \right\}$ | literal |
|-----------------------------|--|---------|

DESCRIPTION.

This pseudo-operator is used to adjust the location of the compiler counter. The value of the location counter specifies the location (control memory address) into which the next generated micro instruction is to be placed.

FORMAT 1.

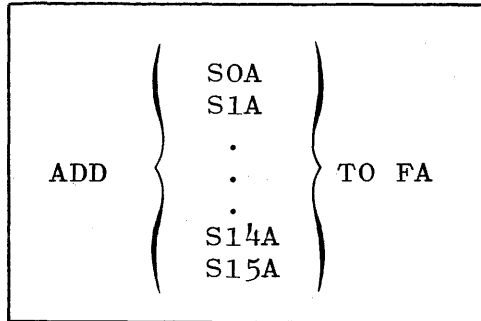
This option sets the location counter to the value of the literal. Code is not generated for the omitted control memory addresses.

FORMAT 2.

This option causes the location counter to be incremented by the literal and NOP commands to be generated for the omitted control memory addresses.

ADD
SCRATCHPAD

FORMAT.



DESCRIPTION.

This instruction adds the left half of any scratchpad word (SnA) to the Field Address (FA) register. The result is placed in FA, and the contents of the scratchpad remain unchanged.

FORMAT.

| |
|---|
| AND register-1 WITH { literal register-2 } |
|---|

DESCRIPTION.

This instruction is used to logically AND the contents of register-1 with the bit configuration of literal or the contents of register-2 and place the result in register-1.

Register-1 may be any 4-bit register (in column 0 or 1 of the Register Addressing table) except a condition register or the CPU register. (Refer to the appendix.)

Register 2 may be any 4-bit register (in column 0 or 1 of the Register Addressing table) except the CPU register. The contents of this register will not be changed.

Literal has a range from 0 to 15.

Table 4-1
AND Truth Table

| Register-1 | AND | Literal Register-2 | Yields | Register-1 |
|------------|-----|-----------------------|--------|------------|
| 0 | AND | 0 | Yields | 0 |
| 0 | AND | 1 | Yields | 0 |
| 1 | AND | 0 | Yields | 0 |
| 1 | AND | 1 | Yields | 1 |

AND
cont

Example:

AND TB WITH 3

| | TA | TB | TC | TD | TE | TF | |
|---|------|------|------|------|------|------|-----------------|
| T | 0000 | 1010 | 1111 | 0011 | 0001 | 0010 | before (0AF312) |
| | -- | 0011 | -- | -- | -- | -- | literal (3) |
| T | 0000 | 0010 | 1111 | 0011 | 0001 | 0010 | after (02F312) |

FORMAT.

| | | | |
|---------|------------------------|------------------|--------|
| BIAS BY | { UNIT register-1 } | [AND register-2] | [TEST] |
|---------|------------------------|------------------|--------|

DESCRIPTION.

This instruction sets the Control Parallel Length (CPL) register and the Control Parallel Unit (CPU) register to values calculated from the given operands.

If only register-1 is used, it must be either register F or register S.

NOTE

All references to register S refer to the SFL or SFU registers in the second half of the first scratchpad word, i.e., the SFL part of the SOB register.

If register-1 and register-2 are used, they may not be the F register and the Control Parallel (CP) register; i.e., one of them must be the F register.

Only the F, S, or CP registers may be used. Identical instructions are produced if the positions of any two registers are interchanged within the instruction.

The Control Parallel Unit (CPU) register will be set to 1, 3, or 0 depending on whether the Field Unit (FU) register is set to 4, 8, or some other value less than 16. This is done for all variations of BIAS except BIAS BY S, which sets the CPU register from SFU rather than from the FU register.

BIAS BY ... sets the Control Parallel Length (CPL) register equal to 2⁴ or to the value in the specified register if it is less than 2⁴. BIAS BY UNIT sets the CPL register equal to the FU register (4 for 4-bit decimal, 8 for 8-bit decimal, or any other value less than 16 for binary).

If the TEST option is used, the above actions are performed and the next micro instruction is skipped if CPL has not been set to zero. The test option may be used with all variations of BIAS.

Example:

BIAS BY F This instruction sets the Control Parallel Length (CPL) register to 24 or to the value in the Field Length (FL) register if it is less than 24. It also sets the Control Parallel Unit (CPU) register equal to the unit in the FU register.

BIAS BY F AND CP This instruction sets the CPL register to 24, to the value in the FL register, or to the value in the CPL register, whichever is the smallest. It also sets the CPU register to the unit in the FU register.

BIAS BY UNIT This instruction sets the CPL register equal to the length of the unit of the type specified by the FU register. It also sets the CPU register equal to one unit of the type specified in the FU register, i.e., 4-bit decimal, 8-bit decimal, or binary.

NOTE

The B 1710 does not permit the 8-bit decimal setting.

FORMAT.

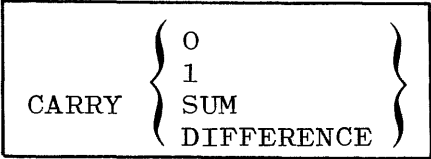
| |
|------------|
| CALL label |
|------------|

DESCRIPTION.

This instruction stores the address of the next micro instruction in the A stack, then branches to the label specified. The location specified by the label may be a maximum of 4095 micro instructions away from the call instruction.

CARRY

FORMAT.



DESCRIPTION.

This instruction sets the Carry (CYF) register to either 0 or 1.

CARRY 0 or CARRY 1 sets the CYF register to 0 or 1 respectively.

CARRY SUM sets the CYF register to the value of the CYL register.

CARRY DIFFERENCE sets the CYF register to the value of the Carry Difference (CYD) register, i.e., to 0 if the X register is greater than the Y register or if the X register equals the Y register and the CYF register equals 0 or to 1 if the X register is less than the Y register or if the X register equals the Y register and the CYF register equals 1.

The CYD register, unlike the CYL register is not conditioned by the CPL register. That is, all 24 bits of the X and the Y registers are compared when setting the CYD register. The programmer should, therefore, know what is in the high-order position of the X and Y registers when using the CYD register if the CPL register is set to less than 24.

FORMAT.

| | | | | |
|----------|---|-------|----------------|---|
| CASSETTE | { | START | | } |
| | | STOP | [WHEN X NEQ Y] | |

DESCRIPTION.

This instruction is used to cause the system cassette tape to start or stop a READ operation at the next inter-record gap. The stop may be unconditional or dependent on the inequality of the X and Y registers.

The information read from the cassette is loaded into the U register and remains there for a maximum of two clock cycles before the U register is cleared.

NOTE

The data on the cassette tape is duplicated every eight bits to insure its validity. The cassette will discriminate against parity incorrect data and, if necessary, use the duplicate eight bits. If both copies are in error the load will be aborted. If the STEP-RUN-TAPE switch is in the TAPE position (refer to LOAD-MSMA) and the START button is pushed, the successive 2-byte increments will be moved from the U register. If the instruction being executed is a 24-bit literal MOVE TO MSMA, then the next 16 bits (2 bytes) that appear in the U register are loaded into control memory at the address indicated by the A register, after which the A register is incremented by 1.

CLEAR

FORMAT.

CLEAR register-1 [register-2...register-n]

DESCRIPTION.

This instruction sets the specified registers to zero.

Any register in column 0, 1, or 2 of the Register Addressing table except condition registers and the register of any 24-bit scratchpad words may be cleared. One clock cycle is used to clear each register or scratchpad word.

A MOVE NULL TO REGISTER-N will be generated for each register specified on the B 1710 series.

FORMAT.

COMPLEMENT register (literal-1)
 [AND register (literal-2) [AND register (literal-n)]]

DESCRIPTION.

This instruction is used to complement (switch the state of) the bit in the register specified by literal-1. By using the options, more than one bit in any one register may be complemented with the same instruction if all bits are in the same 4-bit register.

The register may be any 4-bit (column 0 or 1 in the Register Addressing table) register except a condition register and the CPU register. It may be the FL, FB, L or the T register. All bits must then be in the same 4-bit subfield.

All the literals have a range from 0 to 3 for a 4-bit register, from 0 to 15 for the FL register, and from 0 to 23 for the FB, L, and T registers.

NOTE

Parentheses are required around each literal.

Example:

COMPLEMENT LD(0) AND L(13)

| | LA | LB | LC | LD | LE | LF | |
|---|------|------|------|------|------|-------|-----------------|
| L | 0001 | 0010 | 0011 | 1000 | 0101 | 0110 | before (123856) |
| L | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | after (123456) |
| | 0 | 3 4 | 7 8 | 11 | 15 | 16 17 | 18 23 |

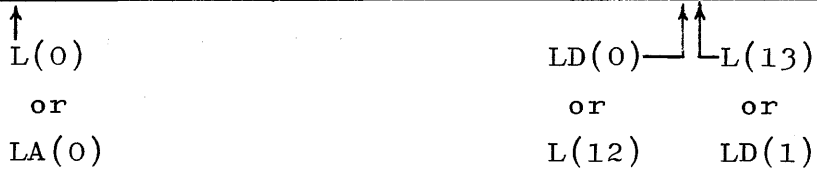
↑ ↙
 LD(0) L(13)

It should be noted that most registers may be addressed in either of two ways.

COMPLEMENT
cont

L Register

| LA | LB | LC | LD | LE | LF |
|---------|-----|------|---------|-------|-------|
| 0 3 | 0 3 | 0 3 | 0 3 | 0 3 | 0 3 |
| 0 ... 3 | 4 7 | 8 11 | 12...15 | 16 19 | 20 23 |



FORMAT.

| |
|--|
| COUNT {FA} {UP} [AND {FL} {DOWN}] [BY {CPL {FL} {DOWN} {FA} {UP}] [BY {literal}]] |
|--|

DESCRIPTION.

This instruction increments or decrements the designated registers by the value of literal or the contents of the Control Parallel Length register (CPL). If the value of literal is 0, the value contained in the CPL register is used.

COUNT FA UP AND FL UP is an invalid instruction.

If the FA register is counted down, it may pass through 0 (i.e., if FA=0 and is counted down by 1, it will be set to hex FFFF). If the FL register is counted down, it will not become less than 0.

If either the FA or the FL register overflows, wraparound to or through 0 will occur; e.g., if either is equal to the maximum value it can contain and is counted up by 1, it becomes equal to 0.

Literal is 5 bits long and has a maximum decimal value of 24.

Example:

COUNT FA UP AND FL DOWN BY 10

| | | | | | | | |
|----|------|------|------|------|------|------|-----------------|
| FA | 0000 | 1001 | 1010 | 0111 | 1111 | 1011 | before (09A7FB) |
| | -- | -- | -- | -- | -- | 1010 | literal + A |
| FA | 0000 | 1001 | 1010 | 1000 | 0000 | 0101 | after (09A805) |
| | | FL | 0000 | 0000 | 0000 | 1000 | before (0008) |
| | | | -- | -- | -- | 1010 | literal - A |
| | | FL | 0000 | 0000 | 0000 | 0000 | after (0000) |

FA is counted up by decimal 10 (hexadecimal A), while FL is counted down by 8 to its minimum value.

DEC

FORMAT.

DEC register-1 BY { literal
register-2 } [TEST]

DESCRIPTION.

This instruction is used to decrement register-1 by the literal or the contents of register-2 and replace the contents of register-1 with the result.

Register-1 may be any 4-bit register (in column 0 or 1 of the Register Addressing table) except the CPU register. The contents of register-2 will not be changed by this instruction.

The literal has a decimal range from 0 to 15.

If the TEST option is used and register-1 underflows (is decremented beyond 0, the smallest value it can contain) as a result of this instruction, the next micro instruction will be skipped (not executed). If no underflow occurs with the TEST option or if the TEST option is not used, the next micro instruction will be executed.

NOTE

All 4-bit registers count modulo 16; e.g., if a register contains a value of 0 and is decremented by 2, it underflows to a value of 14.

FORMAT.

```
DEFINE identifier = string #
```

DESCRIPTION.

This declaration is used to assign a name (identifier) to a string of characters. Any later reference to the identifier is replaced by the string.

The identifier may be made up of alpha (A thru Z) or numeric (0 thru 9) characters in any combination, except that no reserved word (WRITE, MOVE, etc.) in the MIL syntax may be used as an identifier. The special character, dash (-), is acceptable; however, it may not be used as the first character. A length restriction of 25 characters is imposed; i.e., everything after that is considered to be documentation only.

String may be a scratchpad name (24- or 48-bit), a register name, a literal, a part of one instruction, an entire instruction, part of which may have been previously DEFINED, or it may be empty (blank). It must be terminated by a pound sign (#) and may neither begin with a pound sign nor contain any embedded pound signs.

The entire DEFINE declaration must be contained on one card, and all DEFINES must be declared prior to any executable instructions.

Nested DEFINES are allowed up to 13 levels.

Examples:

```

DEFINE SOURCE-POINTER = S3#           * LOAD F FROM SOURCE-POINTER
DEFINE OP-REG = L#                     * CLEAR OP-REG
DEFINE TEST-OP = H800000#             * MOVE TEST-OP TO OP-REG
DEFINE HINT = CC(3)#                  * RESET HINT
DEFINE IGNORE-HALT = RESET HINT#      * IGNORE-HALT

```


DEFINE-VALUE

FORMAT.

```
DEFINE[-VALUE] identifier = literal-1 [ {+} literal-2 ]
```

DESCRIPTION.

This instruction is used to assign the value of the arithmetic result of the literals to the identifier. Any occurrence of the identifier in the program will be replaced by its assigned value.

The DEFINE-VALUE instruction will create up to a 24-bit literal. Values less than zero will be in 2's complement notation and be 24 bits long.

Previously defined identifiers may be used as literals.

The literal may be a hex value, a binary value, or a character used as two hex values. Any literal without a prefixed type indicator (H, B, or ") is considered a decimal literal.

| <u>Examples</u> | <u>Comments</u> |
|-----------------------------|---------------------|
| DEFINE-VALUE AA = H50 | Value is hex 000050 |
| DEFINE B = AA + 1 | Value is hex 000051 |
| DEFINE C = AA - 3 | Value is hex 00004D |
| DEFINE-VALUE F03 = B0010 +4 | Value is hex 000006 |
| DEFINE LOX = F03 - AA | Value is hex FFFFB6 |

| |
|------------------|
| DISPATCH cont |
|------------------|

The DISPATCH READ option is used to transfer both a 24-bit message from the Dispatch buffer to the L register and the source port and channel numbers to the seven least-significant bits of the T register.

NOTE

T(23), when found set after a DISPATCH READ and when the source port is an I/O multiplexor, indicates that a main memory parity error was encountered during the fetch of an I/O descriptor address or an I/O descriptor, or during a RESULT SWAP operation. In this case, the message transferred to the L register will be the address +24 of the parity error.

The DISPATCH READ AND CLEAR option does everything a DISPATCH READ will do and additionally clears the Interrupt Condition (INCN) register. That is, it will RESET all INCN bits to ZERO.

The contents of the L and T registers are unchanged after a DISPATCH WRITE operation. Only the least-significant seven bits of the T register are involved in any dispatch operation.

If the SKIP WHEN UNLOCKED option is used with any variant other than a DISPATCH LOCK, the next micro instruction is skipped.

FORMAT.

EOR register-1 WITH { literal
register-2 }

DESCRIPTION.

This instruction is used to logically EXCLUSIVE OR the bits in register-1 with the value of literal or the contents of register-2 and place the result in register-1.

Register-1 may be any 4-bit register in column 0 or 1 of the Register Address chart except the CPU register. The contents of register-2 will not be changed.

Literal has a decimal range from 0 to 15.

Table 4-2
EOR Truth Table

| register-1 | { literal register-2 } | | | register-1 |
|------------|---------------------------|---|--------|------------|
| 0 | EOR | 0 | yields | 0 |
| 0 | EOR | 1 | yields | 1 |
| 1 | EOR | 0 | yields | 1 |
| 1 | EOR | 1 | yields | 0 |

Example:

| EOR TB WITH 3 | | | | | | |
|---------------|------|------|------|------|------|------|
| | TA | TB | TC | TD | TE | TF |
| T | 0000 | 0101 | 1111 | 0011 | 0001 | 0010 |
| | -- | 0011 | -- | -- | -- | -- |
| T | 0000 | 0110 | 1111 | 0011 | 0001 | 0010 |

before (05F312)

EOR (030000)

after (06F312)

EXIT

FORMAT.

EXIT

DESCRIPTION.

This instruction allows the return of proper program control to the calling routine by causing the compiler to generate a MOVE TAS TO A operation.

The top of the A stack (TAS) will be moved to the Address (A) register, which is used by the hardware logic as the address of the next micro instruction to be fetched. The stack is decremented automatically by the hardware after the move.

A MOVE TAS TO A may be used instead of EXIT with the same result.

FORMAT.

EXTRACT literal BITS FROM T (literal-2) [TO register]

DESCRIPTION.

This instruction isolates the specified bits from the T register and moves them to a receiving register.

Any number of bits from 0 to 2⁴ may be moved. These may be taken from any bit position of the T register. Literal-1, the number of bits to be extracted, may be from 0 to 24. If zero, the CPL indicates the number of bits extracted. Literal-2 indicates the left-most bit to be moved and may be from 0 to 23.

The selected bits are right-justified in the receiving field. Left zeros are inserted if the number of bits moved is less than the length of the receiving register.

The designated register may be the T, X, Y, or L register. The T register remains unchanged unless it is specified as the destination register.

If a register is not specified, the T register is assumed.

Spaces before and after the parentheses are optional; however, parentheses are required.

EXTRACT 0 BITS FROM T(23) TO register may be specified and the programmer must OR into the M register the number of bits to be extracted. The extracted bits are the right-most within the T register.

| |
|-----------------|
| EXTRACT cont |
|-----------------|

Example:

EXTRACT 4 BITS FROM T(20) TO L

| | TA | TB | TC | TD | TE | TF | |
|---|------|------|------|------|------|------|-----------------|
| T | 0000 | 0001 | 0011 | 1000 | 1110 | 0100 | before (0138E4) |

T(20)

| | LA | LB | LC | LD | LE | LF | |
|---|------|------|------|------|------|------|-----------------|
| L | 1001 | 1110 | 0011 | 1001 | 1111 | 1100 | before (1E39FC) |
| L | 0000 | 0000 | 0000 | 0000 | 0000 | 0100 | after (000004) |

Register T remains unchanged while the four extracted bits from the T register are placed in the L register. The bits are right-justified, and leading zeros are added.

NOTE

Caution must be exercised when ORing into the M register before an extract. The machine hardware instruction requires the right-bit pointer for the extraction field, not the left. The hardware also indexes the T register from 1 to 24, left to right, not 0 to 23. The assembler performs this conversion.

GO TO

FORMAT.

GO TO label

DESCRIPTION.

Label must be associated with a run time address that has a displacement from the GO TO instruction of less than 4096 micro instructions.

HALT

FORMAT.

HALT

DESCRIPTION.

This instruction causes the processor to come to an orderly halt. The settings of the console switches determine the register displayed.

Pressing the START pushbutton on the system console will cause the processor to again begin executing micro instructions. If the STEP/RUN switch is in the STEP position, only one micro instruction is executed.

FORMAT 1.

```
IF {register (literal)} {TRUE} THEN {CALL} label  
   {condition}          {FALSE}  {GO TO}
```

FORMAT 2.

```
IF {register (literal)} {TRUE} THEN  
   {condition}          {FALSE}  
  
BEGIN  
    any instructions (one per card)  
  
END [ELSE  
  
BEGIN  
    any instructions (one per card)  
  
END]
```

FORMAT 3.

```
IF module-option {TRUE} THEN INCLUDE  
                 {FALSE}  
  
BEGIN  
    any instructions (one per card)  
  
END [ELSE  
  
BEGIN  
    any instructions (one per card)  
  
END]
```

DESCRIPTION.

FORMAT 1.

This format is used to test a bit (or bits) for a TRUE (ONE) or FALSE (ZERO), then to either CALL or GO TO a label if the test condition is met. By using logic, more than one bit can be tested at the same time; but only if all bits are in the same 4-bit register.

The register may be any 4-bit register (column 0 or 1 in the Register Addressing table) except the CPU register; or it may be FL, FB, L or T.

The literal points to the bit position which is to be tested and has a range from 0 to 3 for a 4-bit register, from 0 to 15 for FL, and from 0 to 23 for FB, L and T. Parentheses are required around the literal.

The condition may be any condition which is available from condition registers. (Refer to CONDITIONS in section 2.)

If neither TRUE nor FALSE (ON nor OFF) is specified, TRUE is assumed.

Example:

IF TD(2) TRUE THEN GO TO LABEL7

| Register TD | Branch |
|-------------|--------|
| 0101 | NO |
| 1101 | NO |
| 0111 | YES |
| 0011 | YES |

In cases three and four, the branch to LABEL7 would be taken since bit position two of TD is ON. Note that TD(2) could have been referred to as T(14).

FORMAT 2.

This format is the same as format 1 except that BEGIN/END pairs replace the CALL and GO TO options.

If the test condition is met, all statements between the first BEGIN/END pair are performed; and a branch is taken around any second (optional) BEGIN/END pair. If the test condition is not met, a branch around the first BEGIN/END pair is taken.

The first BEGIN statement must immediately follow the IF statement, except that comment or blank cards may be used between them. The second BEGIN/END pair is optional; however, if used, the first BEGIN card must be paired with an END ELSE card. Again, no cards other than comment or blank cards may be used between the END ELSE and following BEGIN cards. There are no restrictions on how many types of statements may be used between any BEGIN/END pair.

FORMAT 3.

This format should be used for conditional inclusion of code, depending on the setting of a user-defined, module-option toggle. This module-option toggle is declared and SET or RESET via a module option \$ card. (Refer to the appendix.)

More than one module-option toggle can be tested with the same IF statement by using AND/OR logic. If NOT is used in front of any module-option toggle, that module-option toggle is checked for the RESET state. If both TRUE and FALSE are omitted, TRUE is assumed.

The BEGIN/END pairs are explained in format 2.

In the following examples the statements on the left will generate codes equivalent to those generated by the statements on the right.

| <u>Statement</u> | <u>Equivalent</u> |
|------------------------------|---|
| IF X = Y THEN GO TO +A | IF X EQL Y GO TO +A |
| IF TB(1) OR TB(3) THEN EXIT | SKIP WHEN TB ANY B0101 FALSE EXIT |
| IF LF(2) THEN MOVE X TO Y | IF LF(2) FALSE THEN GO TO +C MOVE X TO Y |
| SET TA(1) | .C OR TA WITH B0100 |

| <u>Statement</u> | <u>Equivalent</u> |
|--|--|
| IF FU(1) FALSE THEN COMPLEMENT T(10) ELSE RESET FL(5) SET L(6) AND L(7) | IF FU(1) GO TO +D EOR TC WITH B0010 TO TO +E .D AND FLD WITH B1011 .E OR LB WITH B0011 |
| IF FLF(3) FALSE THEN BEGIN RESET FB(1) AND FB(3) CLEAR S14A END XCH S14 F S14 | IF FLF(3) GO TO +A AND FU WITH B1010 MOVE NULL TO S14A .A |
| IF LA(0) THEN BEGIN MOVE TAS TO T END ELSE MOVE FA TO T MOVE TE TO LF | IF LA(0) FALSE GO TO +B MOVE TAS TO T GO TO +C .B MOVE FA TO T .C MOVE TE TO LF |
| IF TD(3) THEN MOVE L TO X ELSE BEGIN MOVE T TO X MOVE SUM TO X END MOVE SUM TO FA | IF TD(3) FALSE GO TO +D MOVE L TO X TO TO +E .D MOVE T TO X MOVE SUM TO X .E MOVE SUM TO FA |
| IF LA = 14 THEN BEGIN MOVE 512 TO X END COMPLEMENT FU(0) AND FU(2) | SKIP WHEN LA ≠ B1110 GO TO +A MOVE 512 TO X .A EOR FU WITH B1010 |

Following are examples of conditional inclusion of code:

```

$ SET DEBUG, RESET TRACE
$ SET TRACE, RESET B1700
  
```

After processing these \$ cards, the module options will be set TRUE or FALSE as follows:

```

DEBUG = TRUE
TRACE = TRUE
B1700 = FALSE

IF DEBUG THEN INCLUDE          CALL DEBUG ROUTINE
    CALL DEBUG-ROUTINE
IF TRACE THEN INCLUDE          CALL SAVE-REGISTERS
BEGIN                          CALL TRACE-ROUTINE
    CALL SAVE-REGISTERS
    CALL TRACE-ROUTINE
END

IF DEBUG AND NOT B 1700 INCLUDE MOVE T TO X
BEGIN
    MOVE T TO X
END ELSE
BEGIN
    MOVE L TO X
    MOVE T TO SOA
END

IF NOT TRACE OR B 1700 INCLUDE CALL TRACE-ROUTINE
BEGIN                          MOVE T TO X
    MOVE L TO X
    MOVE T TO S1A
END ELSE
BEGIN
    CALL TRACE-ROUTINE
    MOVE T TO X
END
  
```

Any of the preceding examples may be nested within any of the above BEGIN/END pairs up to a maximum of 31 levels. That is, at any given time during a compilation there may be at most 31 begins that have not been paired with their respective ends.

NOTES

1. A conditional inclusion-block may not be used to include or exclude a BEGIN statement card when the associated END statement card is not part of the block.
2. Logical operators are valid on the registers immediately following the IF, with the following restrictions:
 - a) All registers logically related must be within the same 4-bit group, e.g., IF T(0) and T(3) are valid while IF T(2) and T(4) are not.
 - b) Only two register elements may be logically related, e.g., IF T(2) or T(0) is valid while IF T(2) and T(1) and T(0) are not
 - c) NOT logic may be applied anywhere, e.g., IF NOT(L(3) or NOT L(0)).

FORMAT.

```
INC register-1 {literal  
                register-2} [TEST]
```

DESCRIPTION.

This instruction is used to increment register-1 by the value of literal or the contents of register-2 and to place the result in register-1.

Register-1 may be any 4-bit register in column 0 or 1 of the Register Addressing table except a condition register and the CPU register.

Register-1 may be any 4-bit register in column 0 or 1 of the Register Addressing table except the CPU register. The contents of register-2 are not changed.

Literal has a decimal range from 0 to 15.

If the TEST option is used and register-1 overflows (is incremented beyond 15, the largest value it can contain), the next micro instruction will be skipped (not executed). If overflow does not occur with the TEST option, or if the TEST option is not used, the next micro instruction will be executed.

NOTE

All 4-bit registers count modulo 16; e.g., if a register contains a value of 15 and is incremented by 2, it overflows to a value of 1.

JUMP

FORMAT.

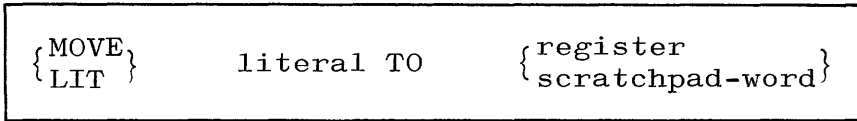
| | | | |
|------|---|----------|---|
| JUMP | { | FORWARD | } |
| | | TO label | |
| | | BACKWARD | } |

DESCRIPTION.

This instruction transfers control to the designated location. The address of label is limited to a maximum relative displacement of plus or minus 4095 micro instructions.

The FORWARD or BACKWARD options are used to cause the compiler to generate a JUMP instruction with a displacement of zero and a direction sign of plus or minus. This is to facilitate ORing the actual displacement into the M register prior to the execution of a JUMP instruction.

FORMAT.



DESCRIPTION.

This instruction is used to move a literal to any register except a condition register or the M register; it may also be used to move a literal to any 24-bit scratchpad word.

The literal may be a decimal integer from 0 to 16777215, a hexadecimal number from H0 to HFFFFFF, or a binary number from B0 to B111111111111111111111111. It may also be a character string up to three characters in length. Leading zeros are not required unless the actual value of the literal is zero. The value of the literal should not be greater than the maximum value that the destination register can contain; if less, left zero fill occurs.

Literal moves to a 24-bit scratchpad word will generate MOVE literal TO TAS followed by MOVE TAS TO scratchpad-word.

Examples:

MOVE 12 TO L

| | LA | LB | LC | LD | LE | LF | |
|---|------|------|------|------|------|------|-----------------|
| L | 0011 | 0000 | 1001 | 1010 | 0001 | 0011 | before (309A13) |
| | -- | -- | -- | -- | -- | -- | LIT (C) |
| L | 0000 | 0000 | 0000 | 0000 | 0000 | 1100 | after (00000C) |

LOAD(*10)

FORMAT.

LOAD F FROM double-scratchpad-word

DESCRIPTION.

This instruction moves a pair of scratchpad words (48 bits total) to the Field (F) register.

Addresses for double-scratchpad-words are:

| | | | |
|----|----|-----|-----|
| S0 | S4 | S8 | S12 |
| S1 | S5 | S9 | S13 |
| S2 | S6 | S10 | S14 |
| S3 | S7 | S11 | S15 |

10 The compiler will generate two MOVE instructions for B 1710 systems.

FORMAT.

| | |
|-----------|-----------|
| LOAD-MSMA | { START } |
| | { STOP } |

DESCRIPTION.

This pseudo-instruction is used to cause the compiler to either start or stop prefacing all emitted micro code with the first 16 bits of a MOVE 24 BIT LITERAL TO MSMA instruction.

The above action is required when a micro program is to be loaded into control memory from a cassette tape while the system is in the TAPE mode. The action of the hardware while in this mode is as follows:

.READLOOP

```

READ 16 BITS FROM THE CASSETTE TO THE U REGISTER
MOVE U TO M
IF M = FIRST HALF OF 24-BIT LITERAL MOVE, THEN READ 16 BITS
    FROM THE CASSETTE TO U
EXECUTE THE MICRO OPERATOR IN M
    (IF M=H9D00=MOVE 24-BIT LITERAL TO THE CONTROL MEMORY
    WORD ADDRESSED BY THE A REGISTER; THEN U, WHICH NOW
    CONTAINS THE ACTUAL MICRO INSTRUCTION, IS MOVED TO
    CONTROL MEMORY ADDRESSED BY THE A REGISTER AND A IS
    INCREMENTED BY 1)
IF M=CASSETTE STOP THEN
    STOP CASSETTE AND HALT PROCESSOR
ELSE
    JUMP TO -READLOOP

```

| |
|-------------------|
| LOAD-MSMA cont |
|-------------------|

No statement between a LOAD-MSMA START and its corresponding LOAD-MSMA STOP may be a reference to any label which has not been declared prior to the LOAD-MSMA STOP.

As an example, the following source code could be used to enable a micro program to be loaded from a cassette into control memory, beginning at control memory address zero:

```
MOVE 0 TO A
SEGMENT ANYNAME AT 0
LOAD-MSMA START
      :
(MAIN MICRO-PROGRAM)
      :
LOAD-MSMA STOP
MOVE 0 TO A
CASSETTE STOP
```

FORMAT.

| | |
|-----------|-----------------------|
| LOAD-SMEM | { START } { STOP } |
|-----------|-----------------------|

DESCRIPTION.

This pseudo-instruction is used to cause the compiler to either start or stop appending each micro instruction with the following two micro instructions:

```
MOVE 24 BIT LITERAL TO X
WRITE 16 BITS FROM X INC FA
```

The above two micro instructions are required when a micro program is to be loaded into main memory from a cassette tape while the system is in the TAPE mode.

Example:

```
MOVE 4096 TO FA          *START ADDRESS
LOAD-SMEM START
.
.
.
MICRO PROGRAM
.
.
.
LOAD-SMEM STOP
CASSETTE STOP
```

NOTE

The FA must start at a mod 32 value.

FORMAT.

```
MACRO macro-identifier (FP-1, FP-2,..., FP-N) =  
    statement-1  
    statement-2  
    .  
    .  
    .  
    .  
    statement-n#
```

DESCRIPTION.

This declaration is used to assign a name (macro-identifier) to a series of statements and to declare any formal parameters which may be used in the macro definition. Any later reference to macro-identifier is replaced by the actual parameters in the reference.

A macro-identifier may be made up of alpha (A through Z) or numeric (0 through 9) characters in any combination; however, no reserved word (WRITE, MOVE, etc.) in the MIL syntax may be used as a macro-identifier. The special character, dash (-), is acceptable but may not be used as the first character. Anything after the twenty-fifth character in the macro-identifier is considered to be documentation only.

Formal parameters (FP-N) are optional; and, if used, they must be enclosed in parentheses. Each has the format macro-identifier-n, where n is originally 1 and is incremented by 1 for each formal parameter. If more than one formal parameter is used in the declaration, they must be separated by commas.

The actual parameters used in the reference to a macro must be single identifiers. They must not contain embedded blanks or special characters.

The one exception is that an actual parameter could be a DEFINE identifier and therefore could contain an embedded dash; however, the DEFINE identifier itself would then have to define a valid, actual parameter. For example, X, 3, H801F, and T0 are valid, actual parameters; but 3 T0 X and (are not. Actual parameters may not be omitted; and, as with formal parameters, they must be enclosed in parentheses and separated by commas.

The macro declaration must be contained on one line and must be terminated with an equal sign (=).

The macro definition must then follow with one statement per line, and the last statement must be terminated by a pound sign (#). For this reason, a MACRO must not itself contain a pound sign. A MACRO may reference another MACRO or a DEFINE which has been previously declared but must not be recursive.

All MACROS must be declared prior to any executable instructions.

Example:

The declaration

```

MARCO WRITEM(WRITE1, WRITE2, WRITE3)=
    XCH WRITE1 F WRITE1
    WRITE 24 BITS FROM WRITE2 WRITE3 FA AND DEC FL
    XCH WRITE1 F WRITE1#
  
```

when referenced as

```
WRITEM(SO,X,INC)
```

results in the reference being replaced by the following in-line code:

```

XCH SO F SO
WRITE 24 BITS FROM X INC FA AND DEC FL
XCH SO F SO
  
```


MICRO

FORMAT.

MICRO {string
 literal}

DESCRIPTION.

This instruction is used to place a 16-bit constant in line. It is the responsibility of the programmer to provide any protection that may be needed to prevent a MICRO from executing.

The literal has a decimal range from 0 to 65535 (hexadecimal H0 thru HFFFF).

MOVE

FORMAT 1.

MOVE {register-1
scratchpad-word-1} TO {register-2
scratchpad-word-2}

FORMAT 2.

MOVE ADDRESS (label) TO register

FORMAT 3.

MOVE segment-count TO register

FORMAT 4.

{MOVE
LIT} literal TO {register
scratchpad-word}

FORMAT 1.

Register-1, the source register, may be any register except the CPU register or control memory (MSMA). If it is the M register, either the MOVE micro itself is moved to register-2 or scratchpad-word-2 is cleared to zeros.

Register-2, the destination register, may be any register except a condition register or a result register. If it is the M register, the source register or scratchpad-word is bit ORed into the next micro instruction to be executed.

Both scratchpad-word-1 and scratchpad-word-2 may be any 24-bit scratchpad words. Note that a MOVE scratchpad-word-1 TO scratchpad-word-2 will generate a MOVE scratchpad-word-1 TO TAS followed by a MOVE TAS TO scratchpad-word-2.

The source field is unchanged by any move unless MOVE TAS TO... is used where the stack is automatically decremented after the move.

FORMAT 2.

This format is used to move the compiler-generated control memory address of the specified label to the specified register. The register may be any column-2 register on the Register Addressing table except the M register. Parentheses are required around the label, which may be either a unique label, a point label, or the name of a segment. (Refer to the pseudo operator SEGMENT.) If it is a point label, it must be referenced either + or -, e.g., MOVE ADDRESS (+PL) TO TAS.

FORMAT 3.

If the SEGMENT pseudo-instruction is used in the program, this format can be used to move, at run time, an 8-bit literal count of the number of times the SEGMENT statement appears in the program previous to the occurrence of this instruction.

The register may be any column-2 register on the Register Addressing table except the M register.

FORMAT 4.

Refer to the LIT instruction for an explanation of this format. Up to eight bits may be moved to CP.

All moves of unequal lengths will justify right with left zero fill or truncation.

Example:

MOVE X TO S7B

| | | | | | | | |
|-----|------|------|------|------|------|------|-----------------|
| X | 0011 | 1001 | 1111 | 1110 | 1101 | 0000 | in Hex (39FED0) |
| S7B | 0011 | 0011 | 0011 | 1110 | 0000 | 0101 | before (333E05) |
| S7B | 0011 | 1001 | 1111 | 1110 | 1101 | 0000 | after (39FED0) |

The contents of register X have been moved to the scratchpad-word S7B; X remains unchanged.

NOP

FORMAT.

NOP

DESCRIPTION.

This is the NO OPERATION instruction. It will do nothing except use one clock cycle and take up one word of control memory.

NORMALIZE

FORMAT.

NORMALIZE

DESCRIPTION.

This instruction shifts the contents of the X register left while counting the FL register down until either the most-significant bit of the X register (determined by the CPL register) equals 1 or until the FL register equals 0.

OR

FORMAT.

OR register-1 WITH { literal
register-2 }

DESCRIPTION.

This instruction is used to logically OR register-1 with the value of the literal or the contents of register-2 and place the result in register-1.

Register-1 may be any 4-bit register in column 0 or 1 of the Register Addressing table except a condition register or the CPU register.

Register-2 may be any 4-bit register in column 0 or 1 of the Register Addressing table except the CPU register. The contents of this register are not changed by this instruction.

The literal has a decimal range of 0 through 15.

| register-1 | OR | { literal register-2 } | yields | register-1 |
|------------|----|---------------------------|--------|------------|
| 0 | OR | 0 | yields | 0 |
| 1 | OR | 0 | yields | 1 |
| 0 | OR | 1 | yields | 1 |
| 1 | OR | 1 | yields | 1 |

Example: OR TB WITH 3

| | TA | TB | TC | TD | TE | TF | |
|---|------|------|------|------|------|------|-----------------|
| T | 0000 | 0101 | 1111 | 0011 | 0001 | 0010 | before (05F312) |
| | -- | 0011 | -- | -- | -- | -- | literal |
| T | 0000 | 0111 | 1111 | 0011 | 0001 | 0010 | after (07F312) |

FORMAT.

OVERLAY

DESCRIPTION.

This instruction overlays control memory from main memory. Previous to initiating an overlay the L register must contain the first control memory overlay address, the FA register must contain the beginning main memory address, and the FL register must contain the length in bits to be overlaid. Overlay will continue until the FL register equals 0 or the A register is out of bounds. If A goes out of bounds, the FA register and the FL register contain the address of the next micro instruction in main memory and the length in bits of unfetched data respectively.

```
MOVE A TO TAS
MOVE L TO A
READ 16 BITS TO L INC FA AND DEC FL
MOVE L TO CONTROL MEMORY ADDRESSED BY A
INC A
TEST FL=0 OR A OUT OF BOUNDS
NO LOOP TO READ EVENT
YES END INSTRUCTION
```

1 Not available on the B 1710 systems.

READ

FORMAT.

READ literal BITS [REVERSE] TO $\left\{ \begin{array}{c} T \\ X \\ Y \\ L \end{array} \right\}$ [{INC} {FA}] [AND {DEC} {FL}]
[{DEC} {FL}]

DESCRIPTION.

This instruction initiates a main memory read cycle to the specified register (T,X,Y,L). The Field Address (FA) register must have been previously set to the appropriate main memory address.

The literal contains the number (of bits) to read and the number to INC or DEC. It is optional and, if used, must be an integer from 0 to 2⁴. If the literal is omitted or if it equals zero, the contents of the CPL register indicate the number of bits to be read.

If the number of bits to be read is zero, the destination register will be set to all zeros; otherwise, the selected bits are right-justified in the destination register. Left zeros are inserted if the number of bits read is less than the length of the destination register.

If the REVERSE option is used, the contents of the FA register point at the last bit + 1 of the field to be read from memory. The destination register still receives the contents of this field as though it had been read in a forward direction.

For example, READ 2⁴ BITS TO T, with FA = zero, and READ 2⁴ BITS REVERSE TO T, with FA = 2⁴, will both result in the T register being set to the same value; i.e., T(0) corresponds to address zero, and T(23) corresponds to address 23.

Incrementing or decrementing occurs following a READ and is optional. For details refer to the COUNT verb.

FORMAT.

RESET register (literal-1)
 [[AND register (literal-2)] AND register (literal-n)]

DESCRIPTION.

This instruction is used to RESET (set to ZERO) the bit in the register which is specified by literal-1. More than one bit in any one register can be RESET with the same instruction if all bits are in the same 4-bit register.

The register may be any 4-bit register in column 0 to 1 of the Register Address chart except a condition register or the CPU register; it may be the FL, the FB, the L or the T register. If more than one bit is to be RESET in the FL, the FB, the L or the T registers, all bits must be in the same 4-bit subfield.

All literals have a range from 0 to 3 for a 4-bit register, from 0 to 15 for the FL register and from 0 to 23 for the FB, the L and the T registers. Parentheses are required around all literals.

Example:

RESET T(0) AND TA(3)

| | TA | TB | TC | TD | TE | TF | |
|---|------|------|------|------|------|------|-----------------|
| T | 1111 | 1010 | 1100 | 1110 | 1001 | 1001 | before (FACE99) |
| T | 0110 | 1010 | 1100 | 1110 | 1001 | 1001 | after (6ACE99) |

↑ T(0)
↑ TA(3)

ROTATE OR SHIFT T

FORMAT 1.

{SHIFT
ROTATE} T LEFT BY {literal BITS}
CPL [TO register]

FORMAT 2.

ROTATE T RIGHT BY literal BITS [TO register]

FORMAT 3.

SHIFT T RIGHT BY literal BITS [TO {
X
Y
T
L}]

DESCRIPTION.

This instruction is used to rotate or shift the contents of the T register and place the result either in the T register or in some other register. If the result is not placed in the T register, T remains unchanged.

FORMAT 1.

The literal has a decimal range from 0 to 23. When the literal equals zero or when CPL is used, a shift or rotation by the value of the CPL register will occur. If CPL is greater than 24, 24 is used.

If the TO register option is used, the rotated or shifted result will be placed in the register and the T register will remain unchanged; otherwise, the result is placed in the T register. The register may be any column 0, 1 or 2 register in the Register Addressing table

except a condition register. If it is the M register, the result of the ROTATE or SHIFT operation is BIT-ORed into the M register and modifies the next micro instruction.

FORMAT 2.

Because the hardware can only rotate the T register to the left, the compiler converts this format to the proper left rotate to accomplish the same result as the rotate right. The T0 register option is the same as in format 1 above.

FORMAT 3.

Because the hardware can only shift the T register left, the compiler will generate an EXTRACT to accomplish the same result. Therefore, the T register may be shifted right only to the X, the Y, the T or the L register. If the T0... option is not used, the result is placed in the T register; otherwise, the T register is unchanged.

The literal has a decimal range from 0 to 23.

NOTE

It is recommended that the EXTRACT instruction itself be used, rather than this format.

ROTATE OR SHIFT
X, Y AND XY

FORMAT.

| | | | | |
|---------------------|------------------|-------------------|----|--------------------------|
| { SHIFT ROTATE } | { X Y XY } | { LEFT RIGHT } | BY | { UNIT literal BITS } |
|---------------------|------------------|-------------------|----|--------------------------|

DESCRIPTION.

This instruction rotates or shifts the X, the Y, or the XY register (X concatenate Y) a specified number of bits to the right or left. Zero fill will occur with the SHIFT instruction.

When UNIT is used, the operand is rotated or shifted by 1, 4, or 8 bits according to the value of the CPU register (type of units contained in the X register and the Y register as determined by the CPU register, either binary, 4-bit, or 8-bit decimal). For the XY register, the literal value may be decimal 0 through 47. For the X register and the Y register, the literal value may be 0 through 23.

The 8-bit decimal is not applicable on the B 1710.

NOTE

The literal has a maximum value of 1 on the B 1710 systems with the concatenated XY register.

FORMAT.

```

SEGMENT {NEWSEGMENT} AT {literal
        {label-1}      {ADDRESS(label-2)}

```

DESCRIPTION.

This pseudo operator is used when a segment of micro code is to be physically placed at the end of a mainline interpreter or emulator at compile time; however, at run time it is overlaid into the mainline routine in control memory at the location specified by the literal or ADDRESS(label-2).

Label-1, if used, must be a unique label; i.e., it must not be a point label. The address associated with label-1 is the physical location of the first micro instruction following the SEGMENT statement.

The NEWSEGMENT option may be used in lieu of label-1 if it is not necessary to name the segment, e.g., a test routine which is to be executed from the cassette.

The literal or ADDRESS(label-2) must specify an address which is less than the address associated with label-1. Label-2, if used, must have been declared previous to the occurrence of the segment statement and must be enclosed in parentheses.

All micro code appearing after a SEGMENT statement is assumed to be in that segment until another SEGMENT statement is found.

There is no limit to the number of SEGMENT statements that may appear in one program.

Labels outside a segment may be referenced from within the segment, and vice versa, with the correct run time linkage addresses being generated.

SET

FORMAT 1.

SET register TO literal

FORMAT 2.

SET register (literal-1)
[AND register (literal-2) [AND register (literal-n)]]

DESCRIPTION.

FORMAT 1.

This statement is used to SET the register to the value of literal-1. The register may be any 4-bit register in column 0 or 1 of the Register Addressing table except a condition register.

Literal-1 has a decimal range from 0 to 15, except when the register is the CPU register, in which case literal-1 has a range from 0 to 3.

FORMAT 2.

This instruction is used to SET the bit specified by literal-1 (bit equal to ONE) into the register. More than one bit in any one register can be SET with the same instruction if all the bits are in the same 4-bit register.

The register may be any 4-bit register in column 0 or 1 of the Register Addressing table except a condition register or the CPU register. It may also be the FL, the FB, the L or the T registers; and, if so, all the bits must be in the same 4-bit subfield.

| |
|-------------|
| SET cont |
|-------------|

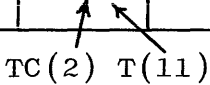
All literals have a decimal range from 0 to 3 for a 4-bit register or subfield, from 0 to 15 for the FL register, and from 0 to 23 for the FB, the L, and the T registers. Parentheses are required around all literals.

Example of format 1:

| | | SET TA TO 3 | | | | | |
|---|--|-------------|------|------|------|------|----------------------|
| | | TA | TB | TC | TD | TE | TF |
| T | | 1111 | 0100 | 0101 | 0110 | 0111 | 1000 before (F45678) |
| T | | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 after (345678) |

Example of format 2:

| | | SET TC(2) AND T(11) | | | | | |
|---|--|---------------------|------|------|------|------|----------------------|
| | | TA | TB | TC | TD | TE | TF |
| T | | 0001 | 0010 | 0000 | 0100 | 0101 | 0110 before (120456) |
| T | | 1001 | 0010 | 0011 | 0100 | 0101 | 0110 after (123456) |



 TC(2) T(11)

SKIP

FORMAT 1.

SKIP WHEN register $\left\{ \begin{array}{l} \text{ALL} \\ \text{ANY} \\ \text{EQL} \end{array} \right\}$ [CLEAR] literal [FALSE]

FORMAT 2.

SKIP WHEN condition [FALSE]

DESCRIPTION.

This statement causes one micro instruction to be skipped if the designated condition is satisfied.

FORMAT 1.

The literal contains a 4-bit mask. It may be comprised of decimal, binary, or hexadecimal entries.

ALL is considered to be TRUE only if all the bits in the register corresponding to one bits in the mask are TRUE. That is, only the designated bit positions are tested to see if they contain ones. ANY is TRUE if at least one bit in the register corresponding to a one bit in the mask is TRUE. EQL is TRUE if all the register bits equal the corresponding bits in the mask. That is, the register must be exactly like the mask.

The CLEAR option is used only with ALL and causes the masked bits of the register to be set to zeros after testing the ALL condition. All bits are not cleared, only the bits tested. If ALL CLEAR is used, the clearing action always occurs whether the SKIP is taken or not. If ALL is used with a mask of 0000, the result is always FALSE.

The FALSE option causes a skip when the whole condition is FALSE.

FORMAT 2.

The condition may be any condition available from the condition registers.

The register may be declared as being:

| | | | | |
|-----|----|----|----|------|
| FU | TA | LA | CA | BICN |
| FT | TB | LB | CB | FLCN |
| FLC | TC | LC | CC | INCN |
| FLD | TD | LD | CD | XYCN |
| FLE | TE | LE | | XYST |
| FOF | TF | LF | | |

PROGRAMMING NOTE

The use of the IF...THEN...ELSE instruction is recommended rather than the SKIP instruction. The SKIP is limited to one, 4-bit grouping mask in one register and may only skip one micro instruction. The IF statement will generate a SKIP WHEN hardware micro instruction whenever possible and is also capable of skipping blocks of micro instructions or testing any combination of bits in many registers.

STORE(*12)

FORMAT.

STORE F INTO double-scratchpad-word

DESCRIPTION.

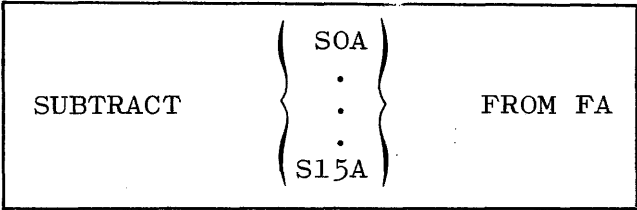
This instruction MOVEs the F register (48 bits) into a double-scratchpad-word. The F register is unchanged.

Double-scratchpad-words are:

| | | | |
|----|----|-----|-----|
| S0 | S4 | S8 | S12 |
| S1 | S5 | S9 | S13 |
| S2 | S6 | S10 | S14 |
| S3 | S7 | S11 | S15 |

12 The compiler generates two MOVE instructions on the B 1710 systems.

FORMAT.



DESCRIPTION.

This instruction is used to subtract the left half scratchpad-word from the Field Address (FA) register. The result is placed in the FA register, and the contents of the scratchpad-word remain unchanged.

SWAP(*13)

FORMAT.

SWAP literal BITS [REVERSE] WITH register

DESCRIPTION.

This instruction swaps the specified number of bits between main memory and the register. The register may be the T, the X, the Y, or the L registers.

The Field Address (FA) register must have been previously set to the proper main memory address.

The literal must be an integer from 0 to 24. It determines the number of bits to swap between main memory and the register. If the value of the literal is zero, the contents of the CPL register are used. If the CPL register is also zero, the register is cleared to all zeros.

If less than 24 bits are swapped, the leading bits of the register is zero. For an explanation of the Reverse option, refer to the READ and WRITE instructions.

NOTE

Incrementing or decrementing of the FA or the FL registers is not allowed with the SWAP instruction.

13 Not available on the B 1710 series.

FORMAT.

```

TABLE label
BEGIN
character-string
character-string
.
.
.
.
END

```

DESCRIPTION.

TABLE is used to create in-line character-strings. Only one character-string is allowed per line, and only character-strings are allowed between the BEGIN and the END statements.

The BEGIN/END pair must surround all strings in the TABLE. The characters are grouped two per address, i.e., 16 bits.

The label must be unique and its use references the first two characters in the table.

Example.

| | |
|-----------|--------------------|
| TABLE REF | The code generated |
| BEGIN | will be: |
| AB | C1C2 |
| ABC | C1C2 |
| D | C3C4 |
| 45 | F4F5 |
| END | |

| | |
|-------------------------|--|
| MOVE ADDRESS (REF) TO Y | The address of the table REF will be loaded into the Y register. |
|-------------------------|--|

WRITE

FORMAT.

WRITE literal BITS [REVERSE] FROM $\begin{Bmatrix} T \\ X \\ Y \\ L \end{Bmatrix}$ [{INC} {FA}] [AND {DEC} {FL}]

DESCRIPTION.

This instruction initiates a main memory write cycle from the specified register (the T, the X, the Y, or the L registers). The Field Address (FA) register must have been previously set to the appropriate main memory address.

The literal, the number (of bits) to write and the number (value) to INC or DEC, is optional and must be an integer from 0 to 24. If the literal is omitted or equals zero, the Control Parallel Length (CPL) indicates the number of bits to write and the number to INC or DEC. If the CPL register also equals zero, no operation (NOP) takes place. WRITE CPL BITS is not permitted.

If less than 24 bits are written, the data in the register is truncated from the left. The contents of the register are unchanged by this instruction.

If the REVERSE option is used, the FA register points to the last bit +1 of the main memory field involved. The contents of the source register are placed in main memory as though they had been written in a forward direction. For example, WRITE 24 bits from T, with the FA register = zero, and WRITE 24 bits REVERSE from T, with the FA register = 24, will both result in the memory field being set to the same value; i.e., T(0) here corresponds to main memory address zero, and T(23) corresponds to main memory address 23.

Incrementing or decrementing occurs following the READ and is optional.

FORMAT.

$$\text{WRITE-STRING string [REVERSE] FROM } \left\{ \begin{array}{c} \text{T} \\ \text{L} \\ \text{X} \\ \text{Y} \end{array} \right\} \left[\left\{ \begin{array}{c} \text{INC} \\ \text{DEC} \end{array} \right\} \left\{ \begin{array}{c} \text{FA} \\ \text{FL} \end{array} \right\} \right] \left[\text{AND } \left\{ \begin{array}{c} \text{INC} \\ \text{DEC} \end{array} \right\} \left\{ \begin{array}{c} \text{FA} \\ \text{FL} \end{array} \right\} \right]$$
DESCRIPTION.

This instruction generates the necessary in-line literals for a string with moves to the indicated register and the WRITE commands to write the string into main memory beginning at the address in the FA register.

The length of the string is limited to the remainder of the source card image. It may be any of the following data types.

| <u>Type</u> | <u>Start-Stop Symbol</u> | <u>Length of Each Unit</u> | <u>Example</u> |
|-------------|------------------------------|--------------------------------|----------------|
| Character | " | 8 bits | "APC128JKL" |
| Hex | @ | 4 bits | @124ADF@ |
| Octal | @(3) | 3 bits | @(3)123567@ |
| Quartal | @(2) | 2 bits | @(2)123321@ |
| Unary | @(1) | 1 bit | @(1)11001101@ |

XCH

FORMAT.

XCH double-scratchpad-word-1 F double-scratchpad-word-2

DESCRIPTION.

This instruction moves the F register to double-scratchpad-word-2; then double-scratchpad-word-1 is moved to the F register. The two words may be the same.

Double-scratchpad-words are as follows:

| | | | |
|----|----|-----|-----|
| S0 | S4 | S8 | S12 |
| S1 | S5 | S9 | S13 |
| S2 | S6 | S10 | S14 |
| S3 | S7 | S11 | S15 |

SECTION 5
PROGRAMMING TECHNIQUES

VIRTUAL-LANGUAGE DEFINITIONS.

A set of virtual-instructions for the virtual machine must first be defined as each being a unique string of bits. This definition may be chosen from any relevant criteria. For example, COBOL verbs may be encoded according to their frequency of usage, the higher frequency verbs being encoded in three bits with one escape code that specifies the next eight bits as an extended code string. Another approach might be to accept directly the source language as in a time-sharing, "line-at-a-time," interactive mode. After the S-instructions and their operand fields have been defined, any standard location or technique should be selected. For example, the base values of S-instructions and S-data items might be in S4A and S5A of the scratch-pad; or all routines are to be referenced with CALL and end with an EXIT instruction to facilitate subrouting. The micro programmer is now ready to begin creating the micro routines needed to perform each of the events in the S-language.

WRITING RULES.

The compiler accepts card images consisting of one symbolic micro instruction per card. Columns 1 through 5 contain the beginning of a label, and columns 6 through 72 contain the micro instructions. Both point labels and unique labels are allowed, with a limit of 25 characters and no imbedded blanks. A blank is the separator between the label and the beginning of the micro instructions. An asterisk (*) anywhere in columns 1 through 72 denotes the beginning of a comment. Columns 73 through 80 are used for sequence numbers.

Source code maintenance as well as other compiler options may be specified by the use of the \$ option cards. The default \$ options are \$ CARD and \$ LIST. An example of a card deck is:

Col 1

```
?  COMPILER PROG-NAME WITH MIL TO LIBRARY
?  DATA CARD
$  (OPTIONAL $ CARDS)
    .
    .
    SOURCE PROGRAM CARDS
    .
    .
    .
?  END
```

Updating a source file requires FILE EQUATE cards for the compiler files SOURCE and NEWTAPE.

In the following examples, S-language statements are explained first; and there is assumed to exist some basic driver routine which is in control at the beginning and the end of each S-instruction. This control routine performs the hardware functions of maintaining an Instruction register and fetching the next S-instruction.

Each example is written as if it were CALLED and returns control with the EXIT micro instruction.

Example 1:

Assume:

- a) The 3 bits 010 imply an S-instruction of ADD 6 decimal digits, Indirect address-1, Indirect address-2 and store the answer in Indirect address-2.
- b) Indirect addresses are displacements from the beginning of a table, and the actual base value of the table is the current setting of the Base register.
- c) The lengths of the Indirect addresses are 9 bits.

- d) All data is in 4-bit decimal form and is 6 decimal digits (24 bits) long.
- e) Overflow is to be ignored.

The instruction might appear as follows:

010 0000110010000101100 in Main Memory.

This bit string represents an S-instruction compiled from a source language statement such as the following:

ADD SUM TO ROLLTOTAL.

That portion of an interpreter which would perform the addition might appear as follows:

```

NEXTSOP  READ 3 BITS TO X INC FA    * GET OP-CODE
        MOVE X TO M                * PREPARE TO DECODE
        JUMP FORWARD               * GO TO DECODER
        GO TO ROUTINE-FETCH
        GO TO ROUTINE-STORE
        GO TO ROUTINE-ADD          * ADD ROUTINE
        .
        .
        .

ROUTINE-ADD                                * LABEL FIRST LINE
        MOVE BR TO S1A              * SET BASE FOR ADD
        READ 9 BITS TO T INC FA     * READ FIRST INDEX
        READ 9 BITS TO L INC FA     * READ SECOND INDEX
        MOVE L TO FA                * LOAD INDEX
        ADD S1A TO FA               * ADD ACTUAL BASE TO INDEX
        READ 24 BITS TO X           * GET DATA-2
        MOVE T TO FA                * LOAD INDEX
        ADD S1A TO FA               * ADD ACTUAL BASE TO INDEX
        READ 24 BITS TO Y           * GET DATA-1

```

| | |
|----------------------|-------------------------------|
| MOVE 00111000 TO CP | * CLEARS CARRY |
| | * SETS CPU AND CPL CORRECTLY |
| MOVE SUM TO T | * GET SUM READY TO WRITE |
| WRITE 24 BITS FROM T | * WRITE SUM |
| MOVE 24 TO CP | * MUST RESTORE CPU IN GENERAL |
| GO TO NEXTSOP | * GO TO NEXT X-INSTRUCTION |

Example 2:

Sample Source Language Statement

MOVE INVERTING FIELD 1 TO FIELD 2

The S-Language might be

| OP-CODE | DATA TYPE | ADDRESS-1 | LENGTH-1 | ADDRESS-2 | LENGTH-2 |
|------------|-------------------------------|------------------|----------|--------------|----------|
| BIT STRING | 0001 = BINARY | LEFT-MOST UNIT | IN BITS | RIGHT-MOST | IN BITS |
| | 0100 = 4-BIT DECIMAL (BCD) | ABSOLUTE ADDRESS | | UNIT ADDRESS | |
| | 1000 = 8-BIT DECIMAL (EBCDIC) | IN BITS | | IN BITS | |

NOTES

1. The size of each element in the S-language is 24 bits or less.
2. The left-most-address-1 points to the beginning of field-1 and the data will be accessed with READ FORWARD commands. The right-most-address-2 points to the end of field-2 and the data will be accessed with READ REVERSE commands.

Assume the following events have been performed in a manner similar to that used in example 1.

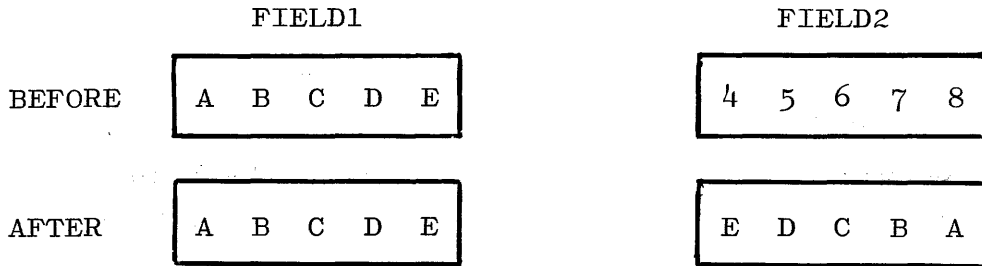
- a. The OP-CODE has been properly decoded and the correct routine has been entered.
- b. The address and length of Field 1 are in the F register.
- c. The address and length of Field 2 are in scratchpad-word SO.

The following code then performs the INVERTED-MOVE operation and properly pads if the receiving field (field 2) is longer than the sending field (field 1).

```
INMOV  BIAS BY UNIT          * SET CPU AND CPL
      READ 4 BITS REVERSE TO L * GET TYPE INDICATOR
TOP    IF FL = 0 THEN GO TO PAD * TEST LIMIT FIELD1
      IF SFL = 0 THEN GO TO ENDOP * END OF FIELD2 STOP
      READ TO X INC FA AND DEC FL * GET A UNIT OF DATA
      SCH SO F SO             * EXCHANGE SO AND F REGISTERS
      WRITE REVERSE FROM X DEC FA AND DEC FL
                                   * PUT A UNIT OF DATA
      XCH SO F SO             * EXCHANGE FOR GET
      GO TO TOP
PAD    LOAD F FROM SO        * GET ADDRESS INTO F REGISTER
      MOVE 0 TO X             * SET ZERO FOR PAD
      IF LF(4) THEN          * TEST FOR EBCDIC
        MOVE H40 TO X        * ADD PAD SPACES
.A    WRITE FROM X DEC FA DEC FL * WRITE A SPACE
      IF FL ≠ 0 GO TO -A     * TEST LIMIT
      GO TO ENDOP           * END OF OPERATION
```

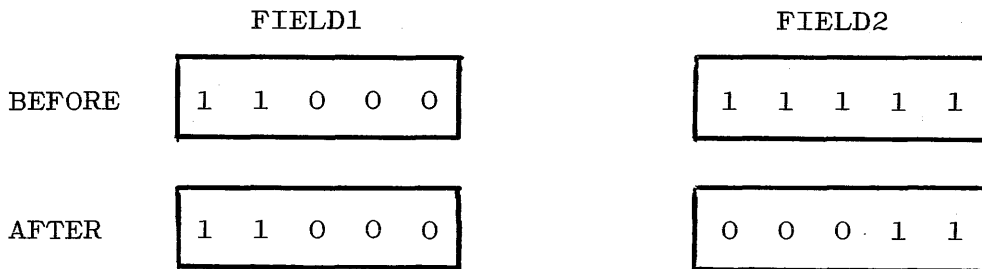
The resultant data movement in memory would be:

An Alpha Data String



OR

With Bit Strings



Notice that the same micro instruction sequence will work in either case.

Example 3:

Given: A list of data items.

Problem: Sort the data items into ascending sequence in place.
(Do a bubble sort.)

Assume: a. The S-operator has the following general format.

| S-OP | Type indicator | Left most address in bits | Length of list in bits |
|-----------------|---|---------------------------|------------------------|
| A BIT STRING | 4 = 4-bit decimal 8 = 8-bit alpha (EBCDIC) any other value from 0 to 15 | A BIT-STRING | A BIT-STRING |

- b. The S-Op has been decoded (see example 1) and the necessary routine has been entered.
- c. Scratchpad-word S5 contains the most-significant (left-most) address in S5A and the TYPE and length in S5B.

Then the following routine will perform the bubble sort.

```

BSORT  CLEAR L                                * CLEAR SWITCH
CYCLE  LOAD F FROM S5                         * FETCH BEGINNING ADDRESS
      IF FL = 0 THEN GO TO ENDOP             * DEGENERATE CASE TEST
      BIAS BY UNIT                            * SET CPL TO UNIT FOR READ/WRITE
.A     COUNT FA UP AND FL DOWN BY COP        * PLACE BETWEEN ITEMS
      IF FL = 0 GO TO ENDOP                  * LAST ITEM TEST
      READ REVERSE TO X                      * GET ITEM ON LEFT
      READ TO Y                              * GET ITEM ON RIGHT
      IF X=Y THEN GO TO -A                   * LEAVE ALONE
      WRITE REVERSE FROM Y                   * REPLACE RIGHT TO LEFT
      WRITE FROM X                           * REPLACE LEFT TO RIGHT
      MOVE HF TO LF                          * MARK NOT ALL SORTED SWITCH
      GO TO -A                               * GO GET NEXT

ENDOP  IF LF (0) TRUE GO TO EXITR            * EXIT ROUTINE
      CLEAR L                                * RESET SWITCH
      GO TO CYCLE                            * TRY WHOLE LIST AGAIN

EXITR  EXIT

```

APPENDIX A
MIL COMPILER OPERATION

COMPILER CONTROL CARD.

The purpose of the compiler control card is to allow the programmer to specify options to the compiler I/O files.

SYNTAX.

| |
|--|
| \$ [NO] option-1 [NO] option-2 ... |
|--|

SEMANTICS.

The \$ must appear in column 1 of the control card.

There must be at least one space between each item on the control card. The options may be in any order. Columns 73 through 80 of the \$-card are used for sequence numbers. Any number of \$-cards may be used and may appear anywhere in the source deck. The options specified will become active from that point on.

If the optional word NO appears before any option, that option is turned OFF. Once an option is ON or OFF, it remains in that state until altered by another control card.

The compiler is preset with the following options: LIST, SINGLE, HEX, CODE, CHECK.

CONTROL CARD OPTIONS.

- LIST(*14) List all compiled source input.
- SINGLE(*14) Single space all printed output.
- DOUBLE Double space all printed output.
- CONTROL List control cards.
- NEW Produce a new source file NEWSOURCE.
- CHECK(*14) Check for sequence errors.
- SUPPRESS Suppress all warning messages except sequence error messages.

14 Default is ON; all others are OFF.

B Generate micro instructions for the B.

MERGE The source of input is the file SOURCE which will have the file CARD merged into it.

PAGE Page eject if listing.

SEQ Causes updated source output and/or listing to be sequenced. If the item following SEQ is an integer, this integer is used as the resequence starting value; otherwise, 1000 is used. If the item following the above integer is another integer, the first character of which is a plus sign (+), this second integer is used as the resequence increment value; otherwise, 1000 is used.

PUNCH Causes object code to be punched on cards. The cards have the following format, with all fields, except the program identifier, in hexadecimal format (0 through 9 and A through F):

CARD COLUMNS

| | |
|---------------|---|
| 1 through 6, | 24-bit control memory address. |
| 8 and 9 | 8-bit count of the number of bits of data. |
| 11 through 70 | up to 240 bits of data, left-justified. |
| 72 through 80 | program identifier from PROGID statement, right-justified and for documentation only. |

VOID The VOID card will void those images from the secondary input file SOURCE which have sequence fields less than or equal to the terminating sequence field. If the terminating sequence field is missing, the only image voided is the first one with the same sequence field as the VOID card.

THE MODULE OPTION \$ CARD.

The module option \$ card is used to SET or RESET user-defined toggles to be used in conjunction with the IF statement for the conditional inclusion of source statements. It may be used anywhere within the source deck, and each module option \$ card affects only those user toggles which are referenced on that card. Before any toggle can be referenced by an IF statement, it must be declared (SET or RESET) on a module option \$ card.

Example:

```
$ SET SYSTEM1
$ RESET SYSTEM2, RESET SYSTEM3, RESET SYSTEM4
$ SET SW1, RESET SW2, RESET SW3, SET SW4, SET SW5
```

To compile a MIL source deck from the card reader and not save a copy of the source code on the disk, the following are applied:

```
? Compile prog-name WITH MIL TO LIBRARY
? DATA CARD
(any MODULE OPTIONS, i.e. $ SET)
.
.
.
(SOURCE PROGRAM IN MIL)
.
.
.
? END
```

To compile a MIL program and save a copy of the source code on the disk, the following are applied:

```
? COMPILE prog-name WITH MIL TO LIBRARY
? MIL FILE NEWSOURCE = user-new-source-file-name
? DATA CARD
(MODULE OPTIONS if used)
```

```

$ NEW
.
.
.
(SOURCE PROGRAM IN MIL)
.
.
.
? END

```

To compile a MIL program from a disk file and merge cards into the program, the disk file not being altered, that is, the cards not permanently altering the disk file, the following are applied:

```

? COMPILE prog-name WITH MIL TO LIBRARY
? MIL FILE SOURCE = user-old-source-file-name
? DATA CARD
(MODULE OPTIONS if used)
$ MERGE
.
.
.
(SOURCE PROGRAM IN MIL)
.
.
.
? END

```

To compile a MIL program from a source image disk file and merge cards into the program, the cards permanently altering the disk file, the following are applied:

```

? COMPILE prog-name WITH MIL TO LIBRARY
? MIL FILE SOURCE = user-old-source-file name
? MIL FILE NEWSOURCE = user-new-source-file-name

```

? DATA CARD
(MODULE OPTIONS if used)

\$ NEW MERGE

.

.

.

(SOURCE PROGRAM IN MIL)

.

.

.

? END

APPENDIX B
B 1710 HARDWARE TABLES

Table B-1
B 1710 Register Addressing

| | | Column Number | | | |
|---------------|----|---------------|------------|---------------|----------|
| | | 0 | 1 | 2 | 3 |
| Row Number | 0 | TA | FU | X | SUM |
| | 1 | TB | FT | Y | CMPX |
| | 2 | TC | FLC | T | CMPY |
| | 3 | TD | FLD | L | XANY |
| | 4 | TE | FLE | A | XEOY |
| | 5 | TF | FLF | M | MSKX |
| | 6 | CA | BICN | BR | MSKY |
| | 7 | CB | FLCN | LR | XORY |
| | 8 | LA | UNASSIGNED | FA | DIFF |
| | 9 | LB | RESERVED | FB | MAXS |
| | 10 | LC | RESERVED | FL | MAXM |
| | 11 | LD | RESERVED | TAS | U |
| | 12 | LE | XYCN | CP | RESERVED |
| | 13 | LF | XYST | RESERVED | CMND |
| | 14 | CC | RESERVED | CONSOLE READ | DATA |
| | 15 | CD | CPU | CONSOLE WRITE | NULL |

Table B-2
Condition Registers

| | | | | |
|------|---|-----------------|-----------------|-------------------|
| BICN | LSUY | CYF | CYD | CYL |
| XYCN | MSBX | X=Y | X<Y | X>Y |
| XYST | LSUX | INT | Y≠0 | X≠0 |
| FLCN | FL=SFL | FL>SFL | FL<SFL | FL≠0 |
| CC | STATE LIGHT | TIMER INTERRUPT | I/O SERVICE REQ | CONSOLE INTERRUPT |
| CD | MEMORY READ DATA PARITY ERROR INTERRUPT | RESERVED | RESERVED | RESERVED |
| BIT | 0 | 1 | 2 | 3 |

ADDRESS MATRIX NOTES.

- a. BICN, FLCN, XYST, and XYCN may not be addressed as destination registers.
- b. MAXM, CC(0), and CD(1,2, and 3) are addressable but will always contain a value of 0.
- c. CPU is a destination register only.
- d. NULL always contains a value of 0. Any register or scratchpad-word to which it is moved will be cleared to 0.

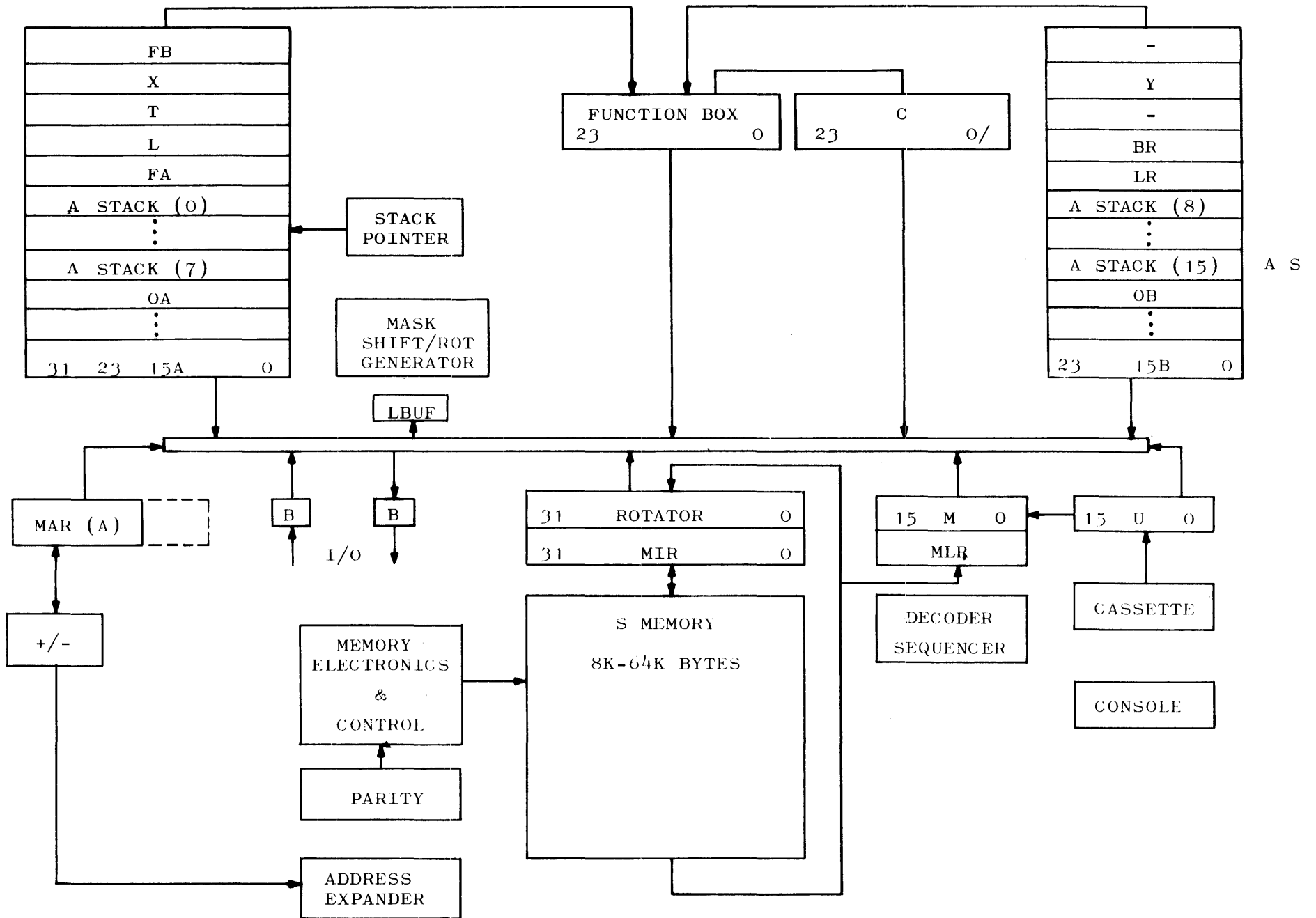


Figure B-1. B 1710 Processor

APPENDIX C
B 1720 HARDWARE TABLES

Table C-1
B 1720 Register Addressing

| | | Column Number | | | |
|---------------|----|---------------|----------|---------------|------|
| | | 0 | 1 | 2 | 3 |
| Row Number | 0 | TA | FU | X | SUM |
| | 1 | TB | FT | Y | CMPX |
| | 2 | TC | FLC | T | CMPY |
| | 3 | TD | FLD | L | XANY |
| | 4 | TE | FLE | A | XEOY |
| | 5 | TF | FLF | M | MSKX |
| | 6 | CA | BICN | BR | MSKY |
| | 7 | CB | FLCN | LR | XORY |
| | 8 | LA | TOPM | FA | DIFF |
| | 9 | LB | RESERVED | FB | MAXS |
| | 10 | LC | RESERVED | FL | MAXM |
| | 11 | LD | RESERVED | TAS | U |
| | 12 | LE | XYCN | CP | MBR |
| | 13 | LF | XYST | MSMA | DATA |
| | 14 | CC | INCN | CONSOLE READ | MCND |
| | 15 | CD | CPU | CONSOLE WRITE | NULL |

Table C-2
Condition Registers

| | | | | |
|------|---|--|--|--|
| BINC | LUSY | CYF | CYD | CYL |
| XYCN | MSBX | X=Y | X<Y | X>Y |
| XYST | LSUX | INT | Y≠0 | X≠0 |
| FLCN | FL=SFL | FL>SFL | FL<SFL | FL≠0 |
| INCN | RESERVED | RESERVED | RESERVED | RESERVED |
| CC | STATE LIGHT | TIMER INTERRUPT | I/O INTERRUPT | CONSOLE INTERRUPT |
| CD | MEMORY READ DATA PARITY ERROR INTERRUPT | MEMORY WRITE/SWAP ADDR OUT OF BOUNDS OVER-RIDE | MEMORY READ ADDR OUT OF BOUNDS INTERRUPT | MEMORY WRITE/SWAP ADDR OUT OF BOUNDS INTERRUPT |
| BIT | 0 | 1 | 2 | 3 |

ADDRESS MATRIX NOTES.

- a. BICN, FLCN, XYST, and XYCN are addressable as source registers only.
- b. The TOPM, MBR, and the A registers are used to determine the memory (control or main) and location of the next micro instruction.
- c. MSMA is control memory and may be addressed only from the maintenance console and during tape mode.
- d. CPU is a destination register only.
- e. NULL always contains a value of 0. Any register or scratchpad-word to which it is moved will be cleared to 0.

APPENDIX D
MICRO INSTRUCTIONS

| MICROMNEMONICS | OP CODE (HEXADECIMAL) | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------------------------|--------------------------|----|----|----|----|---|----------------------|---|---|-------------------------------------|----------------------------------|---------------------------|---------------------------------|---------------------------------------|------------|-----------------|----------|
| | | | | | | SOURCE REG. ROW | | | | SOURCE REG. COL. | DESTINATION REG. COL. | | DESTINATION REG. ROW | | | | |
| REGISTER MOVE | 1 n n n | 0 | 0 | 0 | 1 | SOURCE REG. ROW | | | | SOURCE REG. COL. | DESTINATION REG. COL. | | DESTINATION REG. ROW | | | | |
| SCRATCHPAD MOVE | 2 n n n | 0 | 0 | 1 | 0 | SOURCE/DESTINATION REG. ROW | | | | SOURCE/DEST. REG. COL. | REG. TO S.P. TO REG. | S.P. A | S.P. B | SCRATCHPAD WORD ADDRESS | | | |
| FOUR-BIT MANIPULATE | 3 n n n | 0 | 0 | 1 | 1 | AFFECTED REGISTER ROW | | | | REG. COL. | MANIPULATE VARIANTS | | FOUR-BIT MANIPULATE LITERAL | | | | |
| BIT TEST REL BR ON FALSE | 4 n n n | 0 | 1 | 0 | 0 | SOURCE REG. (FOUR-BIT) ROW | | | | REG. COL. | TEST BIT NUMBER | + / - | | RELATIVE BRANCH DISPLACEMENT VALUE | | | |
| BIT TEST REL BR ON TRUE | 5 n n n | 0 | 1 | 0 | 1 | SOURCE REG. (FOUR-BIT) ROW | | | | REG. COL. | TEST BIT NUMBER | + / - | | RELATIVE BRANCH DISPLACEMENT VALUE | | | |
| SKIP WHEN | 6 n n n | 0 | 1 | 1 | 0 | SOURCE REG. (FOUR-BIT) ROW | | | | REG. COL. | SKIP TEST VARIANTS | | FOUR-BIT TEST MASK | | | | |
| READ/WRITE MEMORY | 7 n n n | 0 | 1 | 1 | 1 | R / W | COUNT FA/FL VARIANTS | | | DATA REG. (XYTL) | FOR / REV | MEMORY FIELD LENGTH | | | | | |
| MOVE EIGHT-BIT LITERAL | 8 n n n | 1 | 0 | 0 | 0 | DESTINATION REG. ROW COL 2 ASSUMED | | | | EIGHT-BIT LITERAL | | | | | | | |
| MOVE 24-BIT LITERAL | 9 n n n | 1 | 0 | 0 | 1 | DESTINATION REG. ROW COL 2 ASSUMED | | | | FIRST EIGHT BITS OF LITERAL | | | | | | | |
| SHIFT/ROTATE REGISTER | A n n n | 1 | 0 | 1 | 0 | DESTINATION REG. ROW | | | | DEST. REG. COL. | SFT. / ROT. | SHIFT/ROTATE COUNT (1-24) | | | | | |
| EXTRACT FROM REGISTER | B n n n | 1 | 0 | 1 | 1 | RIGHT BIT POINTER (1-24) FOR EXTRACTION FIELD OP. | | | | DEST. REG. CODE (XYTL) | WIDTH OF EXTRACTION FIELD (1-24) | | | | | | |
| BRANCH REL. FORWARD | C n n n | 1 | 1 | 0 | 0 | RELATIVE DISPLACEMENT MAGNITUDE | | | | | | | | | | | |
| BRANCH REL. REVERSE | D n n n | 1 | 1 | 0 | 1 | RELATIVE DISPLACEMENT MAGNITUDE | | | | | | | | | | | |
| CALL REL. FORWARD | E n n n | 1 | 1 | 1 | 0 | RELATIVE CALLED ADDRESS MAGNITUDE | | | | | | | | | | | |
| CALL REL. REVERSE | F n n n | 1 | 1 | 1 | 1 | RELATIVE CALLED ADDRESS MAGNITUDE | | | | | | | | | | | |
| SWAP MEMORY | 0 2 n n | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | DEST. GEN. PURPOSE REG. (XYTL) | FOR. / REV. | MEMORY FIELD LENGTH | | | | | |
| CLEAR REGISTERS * | 0 3 n n | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | L. REG. | T. REG. | Y. REG. | X. REG. | FA. REG. | FL. REG. | FU. REG. | CP. REG. |
| SHIFT/ROTATE X OR Y | 0 4 n n | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | SFT. / ROT. | LFT. / RT. | X / Y | | SHIFT/ROTATE COUNT (1-24) | | | |
| SHIFT/ROTATE X AND Y | 0 5 n n | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | SFT. / ROT. | LFT. / RT. | RES. | | SHIFT/ROTATE COUNT (1-24) * | | | |
| COUNT FA AND FL | 0 6 n n | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | COUNT VARIANTS | | LITERAL MAGNITUDE | | | | | |
| EXCHANGE DOUBLEPAD WORD | 0 7 n n | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | DESTINATION 48-BIT SCRATCHPAD ADDR. | | | | SOURCE 48-BIT SCRATCHPAD ADDR. | | | |
| SCRATCHPAD RELATE FA | 0 8 n n | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | RESERVED | + / - | | A(LEFT) SCRATCHPAD WORD ADDRESS | | | | |
| MONITOR | 0 9 n n | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | LITERAL OCCURRENCE IDENTIFIER | | | | | | | |
| CASSETTE CONTROL | 0 0 2 n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | CASSETTE MANIPULATE VARIANTS | | RES. | |
| BIAS | 0 0 3 n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | BIAS VARIANTS | | TEST / NOT TEST | |
| STORE F INTO DOUBLEPAD WORD * | 0 0 4 n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | DESTINATION SCRATCHPAD WORD (48 BITS) | | | |
| LOAD F FROM DOUBLEPAD WORD * | 0 0 5 n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | SOURCE SCRATCHPAD WORD (48 BITS) | | | |
| SET CYF | 0 0 6 n | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | CYF TO CYD | CYF TO CYL | CYF TO 1 | CYF TO 0 |
| HALT | 0 0 0 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| OVERLAY M-STRING * | 0 0 0 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| NORMALIZE X | 0 0 0 3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| NO OPERATION | 0 0 0 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

* NOT AVAILABLE ON B 1710 SYSTEMS

ONE BIT VARIANTS -

| <p style="text-align: center;">FOUR-BIT MANIPULATE (3nnn) VARIANTS</p> <table border="1"> <thead> <tr> <th><u>BITS 4-6</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>000</td><td>SET</td></tr> <tr><td>001</td><td>AND</td></tr> <tr><td>010</td><td>OR</td></tr> <tr><td>011</td><td>EOR</td></tr> <tr><td>100</td><td>INC</td></tr> <tr><td>101</td><td>INC/TEST</td></tr> <tr><td>110</td><td>DEC</td></tr> <tr><td>111</td><td>DEC/TEST</td></tr> </tbody> </table> | <u>BITS 4-6</u> | <u>CONDITIONS</u> | 000 | SET | 001 | AND | 010 | OR | 011 | EOR | 100 | INC | 101 | INC/TEST | 110 | DEC | 111 | DEC/TEST | <p style="text-align: center;">SKIP WHEN (6nnn) SKIP TEST VARIANTS</p> <table border="1"> <thead> <tr> <th><u>BITS 4-6</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>000</td><td>ANY. SKIP</td></tr> <tr><td>001</td><td>ALL. SKIP</td></tr> <tr><td>010</td><td>EQU. SKIP</td></tr> <tr><td>011</td><td>ALL CLR. SKIP</td></tr> <tr><td>100</td><td>NOT ANY. SKIP</td></tr> <tr><td>101</td><td>NOT ALL. SKIP</td></tr> <tr><td>110</td><td>NOT EQU. SKIP</td></tr> <tr><td>111</td><td>NOT ALL. CLR. SKIP</td></tr> </tbody> </table> | <u>BITS 4-6</u> | <u>CONDITIONS</u> | 000 | ANY. SKIP | 001 | ALL. SKIP | 010 | EQU. SKIP | 011 | ALL CLR. SKIP | 100 | NOT ANY. SKIP | 101 | NOT ALL. SKIP | 110 | NOT EQU. SKIP | 111 | NOT ALL. CLR. SKIP | <p style="text-align: center;">READ/WRITE MEMORY (7nnn) VARIANTS</p> <table border="1"> <thead> <tr> <th><u>BITS 6-7</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>00</td><td>X REG.</td></tr> <tr><td>01</td><td>Y REG.</td></tr> <tr><td>10</td><td>T REG.</td></tr> <tr><td>11</td><td>L REG.</td></tr> </tbody> </table> <table border="1"> <thead> <tr> <th><u>BITS 8-10</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>000</td><td>NOP</td></tr> <tr><td>001</td><td>FA↑</td></tr> <tr><td>010</td><td>FL↑</td></tr> <tr><td>011</td><td>FA↑ FL↓</td></tr> <tr><td>100</td><td>FA↓ FL↑</td></tr> <tr><td>101</td><td>FA↓</td></tr> <tr><td>110</td><td>FL↓</td></tr> <tr><td>111</td><td>FA↓ FL↓</td></tr> </tbody> </table> | <u>BITS 6-7</u> | <u>CONDITIONS</u> | 00 | X REG. | 01 | Y REG. | 10 | T REG. | 11 | L REG. | <u>BITS 8-10</u> | <u>CONDITIONS</u> | 000 | NOP | 001 | FA↑ | 010 | FL↑ | 011 | FA↑ FL↓ | 100 | FA↓ FL↑ | 101 | FA↓ | 110 | FL↓ | 111 | FA↓ FL↓ | | | | | | | | | | | | | | | | | | | | | |
|---|------------------------|-------------------|--------|--------|-----|--------|-----|--------|------|---------|---|-----------------|-------------------|----------|--------|-----|--------|----------|--|-----------------|-------------------|--|-----------------|-------------------|----------------|------------|---------------|-------------|-------------------|-------------|---------------|----------|---------------|------|---------------|-----|--------------------|---|-----------------|-------------------|-----|--------|-----|----------|------|-----------|------|-----------------|------------------|-------------------|-----|-----------------|-----|------|------|------------------------|------|---------|-------|---------|-----|------|-----|-------|------|---------|---|--|-------|------|----|------|--|-------|------|-------|------|--|-------|-------|------|------|--|-------|-----|------|------|
| <u>BITS 4-6</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000 | SET | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 001 | AND | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 010 | OR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 011 | EOR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 100 | INC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 101 | INC/TEST | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 110 | DEC | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 111 | DEC/TEST | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <u>BITS 4-6</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000 | ANY. SKIP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 001 | ALL. SKIP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 010 | EQU. SKIP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 011 | ALL CLR. SKIP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 100 | NOT ANY. SKIP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 101 | NOT ALL. SKIP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 110 | NOT EQU. SKIP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 111 | NOT ALL. CLR. SKIP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <u>BITS 6-7</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 | X REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01 | Y REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | T REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | L REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <u>BITS 8-10</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000 | NOP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 001 | FA↑ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 010 | FL↑ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 011 | FA↑ FL↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 100 | FA↓ FL↑ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 101 | FA↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 110 | FL↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 111 | FA↓ FL↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p style="text-align: center;">EXTRACT FROM T REGISTER (8nnn) VARIANTS</p> <table border="1"> <thead> <tr> <th><u>BITS 5-6</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>00</td><td>X REG.</td></tr> <tr><td>01</td><td>Y REG.</td></tr> <tr><td>10</td><td>T REG.</td></tr> <tr><td>11</td><td>L REG.</td></tr> </tbody> </table> | <u>BITS 5-6</u> | <u>CONDITIONS</u> | 00 | X REG. | 01 | Y REG. | 10 | T REG. | 11 | L REG. | <p style="text-align: center;">SWAP MEMORY (02nn) VARIANTS</p> <table border="1"> <thead> <tr> <th><u>BITS 6-7</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>00</td><td>X REG.</td></tr> <tr><td>01</td><td>Y REG.</td></tr> <tr><td>10</td><td>T REG.</td></tr> <tr><td>11</td><td>L REG.</td></tr> </tbody> </table> | <u>BITS 6-7</u> | <u>CONDITIONS</u> | 00 | X REG. | 01 | Y REG. | 10 | T REG. | 11 | L REG. | <p style="text-align: center;">CASSETTE CONTROL (002n) VARIANTS</p> <table border="1"> <thead> <tr> <th><u>BITS 3-1</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>000</td><td>START TAPE</td></tr> <tr><td>001</td><td>STOP ON GAP</td></tr> <tr><td>010</td><td>STOP ON X≠Y</td></tr> <tr><td>011-111</td><td>RESERVED</td></tr> </tbody> </table> | <u>BITS 3-1</u> | <u>CONDITIONS</u> | 000 | START TAPE | 001 | STOP ON GAP | 010 | STOP ON X≠Y | 011-111 | RESERVED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <u>BITS 5-6</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 | X REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01 | Y REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | T REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | L REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <u>BITS 6-7</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 00 | X REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 01 | Y REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | T REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | L REG. | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <u>BITS 3-1</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000 | START TAPE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 001 | STOP ON GAP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 010 | STOP ON X≠Y | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 011-111 | RESERVED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p style="text-align: center;">COUNT FA AND FL (06nn) VARIANTS</p> <table border="1"> <thead> <tr> <th><u>BITS 5-7</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>000</td><td>NOP</td></tr> <tr><td>001</td><td>FA↑</td></tr> <tr><td>010</td><td>FL↑</td></tr> <tr><td>011</td><td>FA↑ FL↓</td></tr> <tr><td>100</td><td>FA↓ FL↑</td></tr> <tr><td>101</td><td>FA↓</td></tr> <tr><td>110</td><td>FL↓</td></tr> <tr><td>111</td><td>FA↓ FL↓</td></tr> </tbody> </table> | <u>BITS 5-7</u> | <u>CONDITIONS</u> | 000 | NOP | 001 | FA↑ | 010 | FL↑ | 011 | FA↑ FL↓ | 100 | FA↓ FL↑ | 101 | FA↓ | 110 | FL↓ | 111 | FA↓ FL↓ | <p style="text-align: center;">DISPATCH (001n) VARIANTS</p> <table border="1"> <thead> <tr> <th><u>BITS 1-3</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>000</td><td>DISPATCH LOCK</td></tr> <tr><td>001</td><td>DISPATCH WRITE</td></tr> <tr><td>010</td><td>DISPATCH READ</td></tr> <tr><td>011</td><td>DISPATCH RD & CLR</td></tr> <tr><td>100</td><td>RESERVED</td></tr> <tr><td>101</td><td>RESERVED</td></tr> <tr><td>110</td><td>RESERVED</td></tr> <tr><td>111</td><td>RESERVED</td></tr> </tbody> </table> | <u>BITS 1-3</u> | <u>CONDITIONS</u> | 000 | DISPATCH LOCK | 001 | DISPATCH WRITE | 010 | DISPATCH READ | 011 | DISPATCH RD & CLR | 100 | RESERVED | 101 | RESERVED | 110 | RESERVED | 111 | RESERVED | <p style="text-align: center;">BIAS (003n) VARIANTS</p> <table border="1"> <thead> <tr> <th><u>BITS 3-1</u></th> <th><u>CONDITIONS</u></th> </tr> </thead> <tbody> <tr><td>000</td><td>FU</td></tr> <tr><td>001</td><td>24 OR FL</td></tr> <tr><td>010</td><td>24 OR SFL</td></tr> <tr><td>011</td><td>24 OR FL OR SFL</td></tr> <tr><td>100</td><td>NOP</td></tr> <tr><td>101</td><td>24 OR CPL OR FL</td></tr> <tr><td>110</td><td>NOP</td></tr> <tr><td>111</td><td>24 OR CPL OR FL OR SFL</td></tr> </tbody> </table> | <u>BITS 3-1</u> | <u>CONDITIONS</u> | 000 | FU | 001 | 24 OR FL | 010 | 24 OR SFL | 011 | 24 OR FL OR SFL | 100 | NOP | 101 | 24 OR CPL OR FL | 110 | NOP | 111 | 24 OR CPL OR FL OR SFL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <u>BITS 5-7</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000 | NOP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 001 | FA↑ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 010 | FL↑ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 011 | FA↑ FL↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 100 | FA↓ FL↑ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 101 | FA↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 110 | FL↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 111 | FA↓ FL↓ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <u>BITS 1-3</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000 | DISPATCH LOCK | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 001 | DISPATCH WRITE | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 010 | DISPATCH READ | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 011 | DISPATCH RD & CLR | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 100 | RESERVED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 101 | RESERVED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 110 | RESERVED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 111 | RESERVED | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <u>BITS 3-1</u> | <u>CONDITIONS</u> | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 000 | FU | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 001 | 24 OR FL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 010 | 24 OR SFL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 011 | 24 OR FL OR SFL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 100 | NOP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 101 | 24 OR CPL OR FL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 110 | NOP | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 111 | 24 OR CPL OR FL OR SFL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| <p style="text-align: center;">REGISTER COLUMN</p> <table border="1"> <thead> <tr> <th></th> <th>0</th> <th>1</th> <th>2</th> <th>3</th> </tr> </thead> <tbody> <tr><td>R</td><td>0 TA</td><td>FU</td><td>X</td><td>SUM</td></tr> <tr><td>E</td><td>1 TB</td><td>FT</td><td>Y</td><td>CMPX</td></tr> <tr><td>G</td><td>2 TC</td><td>FLC</td><td>T</td><td>CMPY</td></tr> <tr><td>I</td><td>3 TD</td><td>FLD</td><td>L</td><td>XANY</td></tr> <tr><td>S</td><td>4 TE</td><td>FLE</td><td>A(MAR)</td><td>XEOY</td></tr> <tr><td>T</td><td>5 TF</td><td>FLF</td><td>M</td><td>MSKX</td></tr> <tr><td>E</td><td>6 CA</td><td>BICN</td><td>BR</td><td>MSKY</td></tr> <tr><td>R</td><td>7 CB</td><td>FLCN</td><td>LR</td><td>XORY</td></tr> <tr><td>R</td><td>8 LA</td><td>TOPM*</td><td>FA</td><td>DIFF</td></tr> <tr><td>O</td><td>9 LB</td><td>RES.</td><td>FB</td><td>MAXS</td></tr> <tr><td>W</td><td>10 LC</td><td>RES.</td><td>FL</td><td>MAXM</td></tr> <tr><td></td><td>11 LD</td><td>RES.</td><td>TAS</td><td>U</td></tr> <tr><td></td><td>12 LE</td><td>XYCN</td><td>CP</td><td>MBR*</td></tr> <tr><td></td><td>13 LF</td><td>XYST</td><td>MSMA*</td><td>DATA</td></tr> <tr><td></td><td>14 CC</td><td>INCN*</td><td>READ</td><td>CMND</td></tr> <tr><td></td><td>15 CD</td><td>CPU</td><td>WRIT</td><td>NULL</td></tr> </tbody> </table> <p><u>CC REGISTER</u> 0 = CONSOLE INTR. 1 = I/O SERVICE REQ. 2 = CLOCK INTR (100 MS) 3 = STATE FLAG</p> <p><u>CD REGISTER</u> 0 = WRT/SWAP OUT OF BDS* 1 = READ OUT OF BDS* 2 = OUT OF BDS OVERRIDE* 3 = MEM. RD. PARITY ERR.</p> <p><u>INCN REGISTER*</u> 0 = PORT DISP. LOCKOUT 1 = PORT DISP. INTR. 2 = PORT PRIORITY INTR. 3 = MISSING CONTROLLER ON PORT OR CHANNEL</p> | | | | 0 | 1 | 2 | 3 | R | 0 TA | FU | X | SUM | E | 1 TB | FT | Y | CMPX | G | 2 TC | FLC | T | CMPY | I | 3 TD | FLD | L | XANY | S | 4 TE | FLE | A(MAR) | XEOY | T | 5 TF | FLF | M | MSKX | E | 6 CA | BICN | BR | MSKY | R | 7 CB | FLCN | LR | XORY | R | 8 LA | TOPM* | FA | DIFF | O | 9 LB | RES. | FB | MAXS | W | 10 LC | RES. | FL | MAXM | | 11 LD | RES. | TAS | U | | 12 LE | XYCN | CP | MBR* | | 13 LF | XYST | MSMA* | DATA | | 14 CC | INCN* | READ | CMND | | 15 CD | CPU | WRIT | NULL |
| | 0 | 1 | 2 | 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R | 0 TA | FU | X | SUM | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E | 1 TB | FT | Y | CMPX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| G | 2 TC | FLC | T | CMPY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| I | 3 TD | FLD | L | XANY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| S | 4 TE | FLE | A(MAR) | XEOY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| T | 5 TF | FLF | M | MSKX | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| E | 6 CA | BICN | BR | MSKY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R | 7 CB | FLCN | LR | XORY | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| R | 8 LA | TOPM* | FA | DIFF | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| O | 9 LB | RES. | FB | MAXS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| W | 10 LC | RES. | FL | MAXM | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 11 LD | RES. | TAS | U | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 12 LE | XYCN | CP | MBR* | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 13 LF | XYST | MSMA* | DATA | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 14 CC | INCN* | READ | CMND | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| | 15 CD | CPU | WRIT | NULL | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

* NOT AVAILABLE ON B 1710 SYSTEMS

| RD BITS UNIT | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | CONTROLLER * ID. (17-23) |
|-------------------|-------------------------|---------------------|----------------------------|--------------------|-------------------|-----------------------------|------------------|----------------------|-----------------------------|--------------------------|-----------------|-----------------|----------------------------|----|----|--|
| CARD READER | VALIDITY CHECK | MEMORY ACCESS ERROR | | READ CHECK | | | | | | | | | | | | CTL. 1 0101010 CTL. 2 0010100 |
| CARD PUNCH | PUNCH CHECK | MEMORY ACCESS ERROR | MEMORY PARITY ERROR | | | | | | | | | | | | | CTL. 0000100 |
| PAPER TAPE READER | TAPE PARITY ERROR | MEMORY ACCESS ERROR | | END OF TAPE | BEGINNING OF TAPE | | SHORT REC. RD. | UNIT REWINDING | | | | | | | | CTL. 10001110 CTL. 20001100 |
| LINE PRINTER | PRINT CHECK | | MEMORY PARITY ERROR | END OF PAGE | 7-9 = CHAR. SET* | | | 10-11 PRT. SPD.* | | PAPER * IN MOTION | MOTOR * ON TEST | NO. PRT. POS. | | | | CTL. 0010110 |
| CONSOLE PRINTER | KEYBOARD E. CANCEL | | ATTP. TO EXECUTE END ADDR. | | INPUT REQUEST | | | | | | | | | | | CTL. 0010110 |
| READER SORTER | UNENCODED DOCUMENT | MEMORY ACCESS ERROR | CANNOT READ | AMOUNT FIELD ERROR | ON-US FIELD ERROR | TRANSIT FIELD ERROR | DOUBLE DOCUMENTS | TOO LATE TO READ | JAM | MISSORT | BATCH TICKET | FLOW STOPPED | EMPTY HOPPER, FULL STACKER | | | CTL. 0010100 |
| MAGNETIC TAPE | TAPE PARITY ERROR. BUSY | MEMORY ACCESS ERROR | MEMORY PARITY ERROR | END OF TAPE | BEGINNING OF TAPE | WRITE LOCKOUT | END OF FILE | UNIT REWINDING | TIME-OUT (3 FT. BLANK TAPE) | CRC ERROR TRACK IN ERROR | | | | | | MTC1-0110010 MTC2-0110000 MTC3-0110100 MTC4-0110110 MTC5-0111000 |
| MFCU | INV. CH. COL. 1 | MEMORY ACCESS ERROR | MEMORY PARITY ERROR | READ CHECK | PUNCH CHECK | PRI. HOP. EMPTY | SEC. HOP. EMPTY | | | | | | | | | CTL. 0001000 |
| DISK CARTRIDGE | READ PARITY ERROR | MEMORY ACCESS ERROR | MEMORY PARITY ERROR | WRITE LOCKOUT | TRACK | OVERRUN | DENSITY | | SEEK | SEEK * STATUS | | SEEKING | | | | CTL. 0011100 |
| SINGLE LINE | PARITY ERROR | MEMORY ACCESS ERROR | MEMORY PARITY ERROR | TIME-OUT | BREAK DET. | END. CTL. CODE NOT RECEIVED | CHAINING TERM | LOSS OF CLR. TO SD. | CARRIER LOSS | | OFF HOOK | RINGING OR ENQ. | | | | SEE BELOW |
| DISK PACK | READ PARITY ERROR | MEMORY ACCESS ERROR | MEMORY PARITY ERROR | WRITE LOCKOUT | TRANS. P.E. | OVERRUN | TRK/DENS.* | ADDRESS PARITY ERROR | SEEK T.O. | SEEK * STATUS | CTL. NO. 0/1 | SEEKING | | | | CTL. 0011110 |

* TEST DESCRIPTOR ONLY

COMMON RESULT DESCRIPTOR BITS:
0 = I/O COMPLETE
1 = EXCEPTION CONDITION
2 = NOT READY

DATA COMMUNICATIONS CTL. ID. BITS 17-23
1000000 = ADAPTER NOT PRESENT
10nnnnn = LEASED OR DIRECT CONNECT
11nnnnn = SWITCHED LINE
nnnnn = 00010 - STANDARD LINE ADAPTER
01000 - TELETYPE ADAPTER
00100 - A.C.U. ADAPTER

I/O RESULT DESCRIPTORS

APPENDIX E

APPENDIX F
MIL ERROR AND WARNING MESSAGES

MIL ERROR MESSAGES.

UNRECOGNIZABLE OR INVALID IDENTIFIER
INSTRUCTION IS INCOMPLETE
MISSING IDENTIFIER IN DEFINE
MISSING = SIGN
MISSING IDENTIFIER IN MACRO
MACRO TOO LONG
INVALID MACRO PARAMETER
MISSING LEFT PARENTHESIS
MISSING RIGHT PARENTHESIS
INVALID MACRO SYNTAX
MISSING COMMA
TOO MANY PARAMETERS IN PARAMETRIC MACRO
REGULAR LABEL HAS BEEN PREVIOUSLY DEFINED
INVALID BEGINNING STATEMENT IDENTIFIER
PREVIOUSLY DEFINED OR RESERVED WORD
MISSING PARAMETER IN PARAMETRIC MACRO
UNRECOGNIZABLE REGISTER NAME
UNRECOGNIZABLE SINGLE PAD NAME
UNRECOGNIZABLE DOUBLE PAD NAME
MISSING OR IMPROPER NOISE WORD
INVALID LITERAL
INVALID SOURCE REGISTER
INVALID SINK REGISTER
LITERAL TOO LARGE
BRANCH ADDRESS OUT OF RANGE
NO SUCH LABEL EXISTS
INVALID LABEL IDENTIFIER
INVALID RELATIONAL
INVALID CONDITION
TABLE FULL ---- VERY FATAL

DEFINES NESTED TOO DEEP
INVALID OR MISSING BIAS VARIANT
DEFINE STARTS WITH OTHER THAN ALPHA
INVALID BIAS VARIANT COMBINATION
DIRECTION MISSING FROM COUNT STATEMENT
INVALID COUNT COMBINATION
SHIFT OR ROTATE DIRECTION MISSING
INVALID SHIFT OR ROTATE DIRECTION
SHIFT OR ROTATE AMOUNT TOO LARGE
TRAILING CODE ON END OF INSTRUCTION
MISSING BEGIN
MISSING END
CANNOT HAVE (ANDS) AND (ORS) IN ONE STMT
CANNOT MIX CONDITION TYPES
CANNOT TEST ON DIFFERENT 4-BIT REGISTERS
(NOT) IS NOT ALLOWED FOR THIS CONDITION
INVALID MULTIPLE CONDITION COMBINATION
(ELSE) IS NOT VALID IN THIS CONTEXT
LABEL ADDRESS IS UNKNOWN
INVALID INSTRUCTION SYNTAX
MISSING ADJUST DIRECTION
INVALID INSTRUCTION SYNTAX
MISSING ADJUST DIRECTION
INVALID SEGMENT NAME
CANNOT ACCESS DIFFERENT 4-BIT REGISTERS
M-REG IS ZERO BEFORE AND AFTER EXEC OF THIS INST
M-REG IS INVALID SINK FOR THIS INSTRUCTION
M-REG IS NOT ZERO FOR THIS MOVE
THERE IS NO ASSOCIATED LOAD-MSMA START
THERE EXIST UNRESOLVED ADDRESSES
TOO MANY ENDS
BADERRCODE

MIL WARNING MESSAGES.

THE FOLLOWING CARD IS OUT OF SEQUENCE

THE FOLLOWING TEST MAY BE IN ERROR DUE TO BIT NUMBERING CONVENTIONS
THE ABOVE CODE MAY BE INEFFICIENT DUE TO (SKIP WHEN) GENERATION
BAD MESSAGE CODE

APPENDIX G
B 1700 HARDWARE INSTRUCTION FORMATS

FORMAT.

| | | |
|------------------------------|-------------------------------|---|
| OP CODE 0000 0000 0011 | BIAS VARIANTS (V) 0...7 | CPL \neq 0 FLAG 0 - NO TEST 1 - TEST CPL RESULT |
| 15 | 4 3 | 1 0 |

This instruction sets CPU to the value 1, 3, or 0 respectively depending on whether the value of FU is 4, 8, or any other value, 0 through 15, except for V = 2. For V = 2, the value of the CPU is determined by SFU in lieu of FU. SFU is the first 4 bits of the scratchpad-word SOB. (On the B 1710, FU = 8 will set CPU = 0.)

The value of CPL is also set to the smallest of the values denoted in the following table.

| <u>V</u> | <u>VALUES</u> |
|----------|---|
| 0 | FU |
| 1 | 24 or FL |
| 2 | 24 or SFL |
| 3 | 24 or FL or SFL |
| 4 | NO-OPERATION |
| 5 | 24 AND CPL AND FL |
| 6 | NO-OPERATION |
| 7 | 24 AND CPL AND SFL AND FL (not defined on the B 1710) |

If the test flag equals 1 and the final value of CPL is not 0, the next micro instruction is skipped.

| |
|-----------------------------------|
| BIT TEST RELATIVE BRANCH FALSE |
|-----------------------------------|

FORMAT.

| OP CODE | REGISTER ROW # | REGISTER COLUMN # | REGISTER BIT # | DISPLACEMENT SIGN | DISPLACEMENT VALUE |
|------------|-------------------|----------------------|-------------------|--------------------------|-----------------------|
| 0100 | 0...15 | 0...1 | 0...3 | 0-POSITIVE 1-NEGATIVE | 0...15 |

15 12 11 8 7 6 5 4 3 0

This instruction tests the designated bit within the specified register and branches (relative to the next instruction) by the amount and direction of the signed displacement value if the bit is 0. If it is 1, a displacement value of 0 is assumed; and control passes to the next in-line M-instruction. A displacement value indicates the number of 16-bit words from the next in-line instruction. A negative sign indicates lower addresses in control memory (backward displacement). The maximum displacement is 15 micro instructions.

| |
|----------------------------------|
| BIT TEST RELATIVE BRANCH TRUE |
|----------------------------------|

FORMAT

| OP CODE | REGISTER ROW # | REGISTER COLUMN # | REGISTER BIT # | DISPLACEMENT SIGN | DISPLACEMENT VALUE |
|------------|-------------------|----------------------|-------------------|--------------------------|-----------------------|
| 0101 | 0...15 | 0...1 | 0...3 | 0-POSITIVE 1-NEGATIVE | 0...15 |

15 11 7 6 5 4 3 0

This instruction tests the designated bit within the specified register and branches (relative to the next instruction) by the amount and direction of the displacement value if the bit is 1. If the bit is 0, a displacement value of 0 is assumed; and control passes to the next in-line M instruction. A displacement value indicates the number of 16-bit words from the next in-line instruction. A negative sign indicates lower addresses in control memory (backward displacement). The maximum displacement is 15 micro instructions.

FORMAT.

| OP | SIGN | DISPLACEMENT VALUE |
|-------|-------|--------------------|
| CODE | 0=+ | 0...4095 |
| 111 | 1=- | |
| 15 13 | 12 11 | 0 |

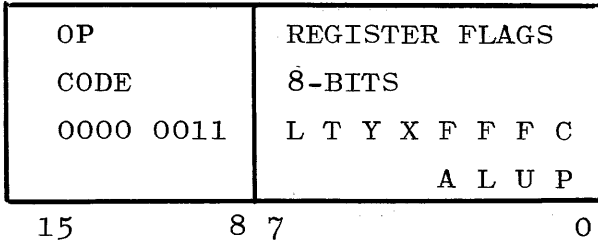
This instruction pushes the address of the next in-line micro instruction (already contained in A register) into the A stack and then fetches the next micro instructions from the locations starting at the number of micro instructions as given in the signed displacement value.

NOTE

EXIT, the opposite of CALL, is accomplished by employing the Move register instruction with the TAS as the source register and A as the destination register.

CLEAR REGISTERS

FORMAT.



This instruction clears all the specified registers to 0 if their respective flag bit is a 1. It is not available on the B 1710.

| |
|-------------|
| COUNT FA/FL |
|-------------|

FORMAT.

| | | |
|-------------------------|--------------------------------|-------------------|
| OP CODE 0000 0110 | COUNT VARIANTS (V) 0...7 | LITERAL 0...31 |
| 15 8 | 7 5 | 4 0 |

This instruction increments (decrements) binarily the designated register(s) by the value of the literal contained in the instruction or by the value of CPL if the value of the literal is 0.

Neither overflow nor underflow of FA is detected. The value of FA may go through its maximum value or its minimum value and wrap around. Overflow of FL is also not detected. The value of FL may go through its maximum value and wrap around. Underflow of FL is detected and will not wrap around; the value of 0 is left in FL.

Count variants are as follows:

- V = 000 No Count
- 001 Count FA Up
- 010 Count FL Up
- 011 Count FA Up and FL Down
- 100 Count FA Down and FL Up
- 101 Count FA Down
- 110 Count FL Down
- 111 Count FA Down and FL Down

FORMAT.

| | | |
|------------------------------|--|---|
| OP CODE 0000 0000 0001 | DISPATCH VARIANTS 000-LOCKOUT 001-WRITE 010-READ & CLEAR | SKIP FLAG 0-SKIP IF ALREADY LOCKED 1-SKIP IF NOT ALREADY LOCKED |
| 15 | 4 3 | 1 0 |

The DISPATCH operation is used to initiate I/O operations and to receive interrupt information from other ports.

Since the interrupt system is shared by all ports, the normal procedure for the processor in initiating a message, such as an I/O INITIATE, is to gain control of the interrupt system by attempting a LOCKOUT.

If the LOCKOUT is successful, the processor can proceed to generate the INITIATE I/O DISPATCH message and then perform the WRITE DISPATCH operation.

The skip variant allows skipping of the next 16-bit instruction based upon the success or failure of the LOCKOUT attempt.

The WRITE DISPATCH operation stores the contents of the L register in main memory location 0 to 23. L contains the absolute address of the beginning I/O descriptor. It also transfers the contents of the least-significant seven bits of the T register (designating the destination port # and destination channel (#) to the Port Interface Dispatch register).

15 The DISPATCH command requires a hardware I/O subsystem which is available on the B 1730 only.

DISPATCH
cont

The READ & CLEAR DISPATCH operation performs the reverse of the WRITE DISPATCH as well as clearing the INCN flip-flops in the interrupt system.

FORMAT.

| OP CODE | ROTATE BIT COUNT | DESTINATION REGISTER | EXTRACT BIT COUNT |
|------------|---------------------|--------------------------------------|----------------------|
| 1011 | 0...24 | 00 - X 01 - Y 10 - T 11 - L | 0...24 |

15 12 11
7 6
5 4
0

This instruction rotates the T register contents left by the ROTATE count, extracts the bits specified and moves the result to the destination register. If the extract bit count is less than 24, the data is right-justified with left (most-significant) zero bits supplied.

The contents of the T-register are unchanged unless it is also the destination register.

A rotate value of 24 is equal to 0 and is equivalent to a NO operation.

NOTE

The micro programming assembler uses the left-most bit to be extracted and calculates the rotate bit count to be used by the hardware circuits. The assembler addresses the bits within the T-register left-to-right as 0 through 23, while hardware addresses right to left as 0 through 23.

HALT

FORMAT.

OP

CODE

0000 0000 0000 0001

15

0

This instruction stops the execution of the micro instructions.

NOTE

Whatever register to which the register select switches point will be displayed.

FORMAT.

| | |
|----------------|--------------|
| OP | SCRATCHPAD |
| CODE | WORD ADDRESS |
| 0000 0000 0101 | 0...15 |
| 15 | 4 3 0 |

This instruction moves the contents of the A and B portions of the designated scratchpad-word to the FA and FB registers respectively.

The instruction is not available on the B 1710.

| |
|------------------|
| MANIPULATE 4-BIT |
|------------------|

FORMAT.

| OP CODE | REGISTER ROW # | REGISTER COLUMN # | MANIPULATE VARIANTS (V) | LITERAL |
|------------|-------------------|----------------------|----------------------------|---------|
| 0011 | 0...15 | 0...1 | 0...7 | 0...15 |
| 15 12 11 | 8 | 7 | 6 4 | 3 0 |

This instruction performs the operation specified by the variants on the designated register.

- V = 0 The register is set to the value of the literal.
- 1 The register is set to the logical AND of the register and literal.
 - 2 The register is set to the logical OR of the register and literal.
 - 3 The register is set to the logical EXCLUSIVE-OR of the register and literal.
 - 4 The register is set to the binary sum modulo 16 of the register and literal.
 - 5 The register is set to the binary difference modulo 16 of the register and literal.
 - 6 The register is set to the binary sum modulo 16 of the register and literal, and the next micro instruction is skipped if a carry is produced.
 - 7 The register is set to the binary difference modulo 16 of the register and literal, and the next micro instruction is skipped if a borrow is produced.

Exceptions:

BICN, XYCN, XYST, INCN, and CPU are excluded as source/destination registers.

| |
|--------------------|
| MOVE 8-BIT LITERAL |
|--------------------|

FORMAT.

| OP CODE | DESTINATION REGISTER | LITERAL |
|------------|-------------------------|---------|
| 1000 | ROW # 0...15 | 0...255 |

15 12 11 8 7 0

This instruction moves the 8-bit literal given in the instruction to the destination register. If the move is between registers of unequal lengths, the data is right-justified with left (most-significant) zero bits supplied.

Only registers X, Y, T, L, A, BR, LR, FA, FB, FL, TAS and CP can be specified. The register select number is assumed to be 2. (Refer to Table B-1.)

| |
|---------------------|
| MOVE 24-BIT LITERAL |
|---------------------|

FORMAT.

| | | |
|----------|-------------|----------------|
| OP | DESTINATION | 24-BIT LITERAL |
| CODE | REGISTER | 0...MAX |
| 1001 | ROW # | |
| | 0...15 | |
| 15 12 11 | 8 7 | 0 |

This instruction moves the 24-bit literal given in the double-length micro instruction to the destination register. If the move is between registers of unequal lengths, the literal is truncated from the left.

Only registers X, Y, T, L, A, BR, LR, FA, FB, FL, TAS and CP can be specified. The register select number is assumed to be 2.

The MSM register (not available on the B 1710) may be a destination only in the TAPE mode.

READ/WRITE MEMORY

FORMAT.

| OP | DIRECTION | COUNT | REGISTER # | FIELD | MEMORY |
|-------|---------------|----------|------------------|------------------------------|--------|
| CODE | 0 TO REGISTER | VARIANTS | 00 = Y | DIRECTION | FIELD |
| 0111 | 1 TO MEMORY | 0...7 | 01 = Y 10 = T | 0 - POSITIVE 1 - NEGATIVE | LENGTH |
| 15 12 | 11 | 10 | 8 7 | 6 | 5 4 0 |

This instruction moves the contents of the register (memory) to the memory (register). If the value of the memory field length is less than 2⁴, the data from memory is right-justified with left (most-significant) zero bits supplied while the data from the register is truncated from the left.

The contents of the source is unchanged.

Register FA contains the bit address of the memory field while the memory field direction sign and memory field length are given in the instruction.

If the value of the memory field length as given in the instruction is 0, the value in CPL is used.

Refer to the COUNT FA/FL instruction for a description of the count variants.

Positive field direction indicates ascending memory addresses. Writes of 26 bits will actually write 2⁴ bits and invert the parity bits of the four bytes accessed. Writes of 25 bits will actually write 2⁴ bits and correct any improper parity.

NOTE

A wrong parity detected during a read cycle will not be corrected during a write cycle unless specified. Lengths of 25 and 26 are for TEST/MAINTENANCE routines and length greater than 26 are reserved.

FORMAT.

| | | | | |
|-------|----------|----------|-------------|-------------|
| OP | SOURCE | SOURCE | DESTINATION | DESTINATION |
| CODE | REGISTER | REGISTER | REGISTER | REGISTER |
| 0001 | ROW # | COLUMN # | ROW # | COLUMN # |
| | 0 - 15 | 0...3 | 0...2 | 0...15 |
| 15 12 | 11 | 8 7 | 6 5 | 4 3 |
| | | | | 0 |

This instruction moves the contents of the source register to the destination register. If the move is between registers of unequal lengths, the data is right-justified with left (most-significant) zero bits supplied or the data is truncated from the left, whichever is appropriate.

The contents of the source register are unchanged unless it is also the destination register.

Exceptions:

- a. Control memory (MSM) and CPU are excluded as source registers.
- b. When the M register is used as a destination, the operation results in an INCLUSIVE OR function, which modifies the next micro instruction. It does not modify the instruction stored in control memory.
- c. BICN, FLCN, XYCN, XYST, INCN, and control memory (MSM) are excluded as destination registers as are those designated as result registers. (Refer to column 3 of table C-1.)
- d. U is excluded as a source register in the STEP and TAPE modes.
- e. U is included as a source register in moves to the Data, Command, MBR, and MSMA registers in the RUN mode.

SCRATCHPAD MOVE

FORMAT.

| | | | | | |
|-------|----------|----------|------------|-------------|------------|
| OP | REGISTER | REGISTER | DIRECTION | SCRATCHPAD | SCRATCHPAD |
| CODE | ROW # | COLUMN # | 0-TO | WORD | WORD |
| 0010 | 0...15 | 0...3 | SCRATCHPAD | 0-LEFT WORD | ADDRESS |
| | | | 1-FROM | 1-RIGHT | 0...15 |
| | | | SCRATCHPAD | WORD | |
| 15 12 | 11 8 | 7 6 | 5 | 4 | 3 0 |

This instruction moves the contents of the register (scratchpad) to the scratchpad (register). If the move is between fields of unequal lengths, the data is right-justified with left (most-significant) zero bits supplied or the data is truncated from the left, whichever is appropriate.

The contents of the source register are unchanged.

Exceptions:

- a. Control memory (MSM), U, and CPU are excluded as source registers.
- b. When the M register is used as a destination, the operation results in an INCLUSIVE OR with the next micro instruction. It does not modify the instruction stored in control memory.
- c. BICN, FLCN, XYCN, XYST, INCN, and control memory (MSM) are excluded as destination registers as are those designated in column 3 of table C-1.

FORMAT.

| | | | |
|-------------------------|---------------------|---------------------------|---|
| OP CODE 0000 1000 | RESERVED 000 | SIGN OF 0 = + 1 = - | LEFT HALF ADDRESS OF A SCRATCHPAD WORD 0 - 15 |
| 15 8 7 | 5 | 4 3 | 0 |

This instruction replaces the contents of the FA register with the SUM of FA and the left half of a scratchpad-word.

| |
|---------|
| SET CYF |
|---------|

FORMAT.

| | |
|----------------|--------------|
| OP | SET |
| CODE | VARIANTS (V) |
| 0000 0000 0110 | 0...15 |
| 15 | 4 3 0 |

This instruction sets the carry flip-flop as specified by the variants.

- V = 1 SET CYF TO 0
2 SET CYF TO 1
4 SET CYF TO CYL (carry total from sums)
8 SET CYF TO CYD (carry borrow from difference)

NOTES

1. CYL is generated under the control of the length in CPL.
2. CYF is an input to the arithmetic logic along with the X and Y registers. CYF is the left-most bit of the CP portion of the C register.

FORMAT.

| OP CODE | DESTINATION REGISTER | DESTINATION REGISTER | SHIFT/ROTATE | SHIFT/ROTATE BIT, COUNT |
|------------|-------------------------|-------------------------|-------------------------|----------------------------|
| 1010 | ROW # | COLUMN # | 0 - SHIFT 1 - ROTATE | 0...24 |
| 15 12 | 11 | 8 7 | 6 | 5 4 0 |

This instruction shifts (rotates) register T left by the number of bits specified and then moves the 24-bit result to the destination register. If the move is between registers of unequal lengths, the data is right-justified, with data truncated from the left.

The contents of the T register are unchanged unless it is also the destination register.

Zero fill on the right and truncation on the left occurs with the shift operation. ROTATE is an end-around shift with no truncation or fill.

If the value of the SHIFT/ROTATE COUNT as given in the instruction is zero, the value given in CPL is used.

Exceptions:

- a. When the M register is used as a destination register, the operation results in an inclusive OR with the next micro instruction. It does not modify the instruction stored in control memory.
- b. BICN, FLCN, XYCN, XYST, INCN, and control memory (MSM) are excluded as destination registers, as are the result registers. (Refer to column 3 of table C-1.)

| |
|--|
| SHIFT/ROTATE REGISTERS X AND Y LEFT/RIGHT |
|--|

FORMAT.

| | | | |
|-------------------------|--|---|--|
| OP CODE 0000 0101 | SHIFT/ROTATE VARIANT 0-SHIFT 1-ROTATE | SHIFT/ROTATE DIRECTION VARIANT 0-LEFT 1-RIGHT | SHIFT/ROTATE BIT COUNT 0...48 |
| 15 8 | 7 | 6 | 5 0 |

This instruction shifts (rotates) register (X-Y) left (right) by the number of bits specified. The register X is the left-most (most-significant) half of the concatenated 48-bit XY register. Only a count of one may be specified on the B 1710 for the concatenated XY register.

Zero fill on the right and truncation on the left occurs with the left shift. Zero fill on the left and truncation on the right occurs with the right shift.

If the value of the SHIFT/ROTATE COUNT as given in the instruction is zero, the operand is shifted/rotated by the amount determined by CPU as follows:

| <u>CPU</u> | <u>SHIFT/ROTATE COUNT</u> |
|------------|--------------------------------------|
| 00 | 1 bit |
| 01 | 4 bits |
| 10 | Undefined |
| 11 | 8 bits (not available on the B 1710) |

NOTE

The shift by CPU option is
not available on the B 1710.

| |
|--|
| SHIFT/ROTATE REGISTER X OR Y LEFT/RIGHT |
|--|

FORMAT.

| | | | | |
|-------------------------|--|--|--------------------------------------|--|
| OP CODE 0000 0100 | SHIFT/ROTATE VARIANT 0-SHIFT 1-ROTATE | SHIFT/ROTATE DIRECTION 0-LEFT 1-RIGHT | X/Y VARIANT 0-X REG 1-Y REG | SHIFT/ROTATE BIT COUNT 0...24 |
| 15 8 | 7 | 6 | 5 | 4 0 |

This instruction shifts (rotates) register X or Y left or right by the number of bits specified.

Zero fill on the right and truncation on the left occurs with the left shift. Zero fill on the left and truncation on the right occurs with the right shift.

If the value of the SHIFT/ROTATE COUNT as given in the instruction is zero, the operand is shifted (rotated) by the amount determined by CPU as follows:

| <u>CPU</u> | <u>SHIFT/ROTATE COUNT</u> |
|------------|--------------------------------------|
| 00 | 1 bit |
| 01 | 4 bits |
| 10 | Undefined |
| 11 | 8 bits (not available on the B 1710) |

NOTE

The shift by the CPU option is not available on the B 1710.

| |
|-----------|
| SKIP WHEN |
|-----------|

FORMAT.

| OP CODE | REGISTER ROW # | REGISTER COLUMN # | SKIP TEST VARIANTS (V) | MASK 0...15 |
|------------|-------------------|----------------------|---------------------------|----------------|
| 0110 | 0...15 | 0...1 | 0...7 | |
| 15 12 11 | 8 | 7 | 6 4 | 3 0 |

This instruction tests only the bits in the register that are referenced by the 1 bits in the mask and ignores all others. It then performs the actions specified below. Exception: If V = 2 or V = 6, it compares all bits for an equal condition.

V = 0 If any of the referenced bits are 1's, the next M instruction is skipped.

1 If all the referenced bits are 1's, the next M instruction is skipped.

3 This is the same as V = 1, but the referenced bits are also cleared to 0 without affecting the non-referenced bits.

4 If any of the referenced bits are 1's, the next M instruction is not skipped.

5 If all the referenced bits are 1's, the next instruction is not skipped.

6 If the register is equal to the mask, the next instruction is not skipped.

7 This is the same as V = 5, but the referenced bits are also cleared to 0 without affecting the non-referenced bits.

NOTES

1. If the mask equals 0000 the ANY result is FALSE. The skip is made for V = 0 and is not made for V = 4.
2. If the mask equals 0000, the ALL result is TRUE. The skip is made for V = 5 and V = 7 and is not made for V = 1 and V = 3.

Exceptions:

BICN, FLCN, XYCN, XYST and INCN are excluded as operand registers when V = 4 or when V = 7.

STORE F INTO
DOUBLEPAD WORD

FORMAT.

| | |
|----------------|--------------|
| OP | SCRATCHPAD |
| CODE | WORD ADDRESS |
| 0000 0000 0100 | 0...15 |
| 15 | 4 3 0 |

This instruction moves the contents of the FA and FB registers to the designated scratchpad-word. FA is transferred to the A half of the scratchpad-word, and FB (which contains FL, FT, and FU) is transferred to the B scratchpad-word.

This is not available on the B 1710.

| |
|--|
| SWAP DOUBLEPAD WORD WITH REGISTER F |
|--|

FORMAT.

| | | |
|-------------------------|---|---|
| OP CODE 0000 0111 | DESTINATION 48-BIT SCRATCHPAD WORD 0...15 | SOURCE 48-BIT SCRATCHPAD WORD 0...15 |
| 15 8 7 | 4 3 | 0 |

This instruction moves the contents of the 48-bit register to a hardware holding register. It also moves the contents of the source scratchpad-word to the F register and moves the contents of the hardware holding register to the destination register.

FORMAT.

| OP | REGISTER # | FIELD | MEMORY |
|-----------|------------|--------------|--------|
| CODE | 00 = X | DIRECTION | FIELD |
| 0000 0010 | 01 = Y | 0 - POSITIVE | LENGTH |
| | 10 = T | 1 - NEGATIVE | 0...24 |
| | 11 = L | | |
| 15 | 8 | 7 | 6 |
| | | | 5 |
| | | | 4 |
| | | | 0 |

This instruction swaps data from main memory with the data in the specified register. If the value of the memory field is less than 24, the data from memory is right-justified with left (most-significant) zero bits supplied. The data from the register is truncated from the left before entering memory.

Register FA contains the absolute binary address of the main memory field while the field direction sign and field length is given in the instruction.

If the value of the memory field length as given in the instruction is 0, the value given in CPL is used.

This is not available on the B 1710.

APPENDIX H
MICRO PROCESSOR TIMING TABLES

Table H-1
B 1710 Micro Instruction Timing

| MICRO INSTRUCTIONS | NUMBER OF CLOCKS | NOTES |
|---|------------------|-------|
| BIAS | 2 | |
| BIT TEST RELATIVE BRANCH FALSE | 2 | |
| BIT TEST RELATIVE BRANCH TRUE | 2 | |
| BRANCH RELATIVE | 4 | |
| CALL | 5 | 1 |
| CASSETTE CONTROL | 2 | |
| COUNT FA AND/OR FL REGISTERS | 4 | |
| EXTRACT FROM T REGISTER | 3 | |
| FOUR BIT MANIPULATE | 2 | |
| HALT | 2 | |
| MOVE EIGHT-BIT LITERAL | 2 | |
| MOVE TWENTY-FOUR-BIT LITERAL | 6 | |
| MOVE REGISTER TO REGISTER | 2 | 2 |
| NO OPERATION | 2 | |
| NORMALIZE X REGISTER | 6 | 3 |
| READ OR WRITE MAIN MEMORY | 8 | |
| SCRATCHPAD MOVE | 2 | 2 |
| SCRATCHPAD RELATE | 4 | |
| SET CYF REGISTER | 2 | |
| SHIFT OR ROTATE T REGISTER LEFT | 3 | |
| SHIFT OR ROTATE X OR Y REGISTER LEFT OR RIGHT | 3 | |
| SHIFT X AND Y REGISTERS LEFT OR RIGHT | 6 | 4 |
| SKIP WHEN | 2 | |
| SWAP F REGISTER WITH DOUBLE SCRATCHPAD WORD | 10 | |

NOTES

The basic clock of the B 1710 is 1 mega hertz

1. This includes the fetch of the called micro instruction.
2. For BCD result register moves, there are three clocks.
3. There are six clocks per bit plus one additional clock.
4. Only a value of one bit is allowed in the B 1710.

Table H-2

B 1720 Micro Instruction Timing

| MICRO INSTRUCTIONS | NUMBER OF CLOCKS | NOTES |
|---|------------------|-------|
| BIAS | 1 | |
| BIT TEST RELATIVE BRANCH FALSE | 1 | |
| BIT TEST RELATIVE BRANCH TRUE | 1 | |
| BRANCH RELATIVE | 1 | 1 |
| CALL | 2 | |
| CASSETTE CONTROL | 1 | |
| CLEAR REGISTERS | 1 | |
| COUNT FA AND/OR FL REGISTERS | 1 | |
| EXTRACT FROM T REGISTER | 1 | |
| FOUR-BIT MANIPULATE | 1 | |
| HALT | 1 | |
| LOAD F REGISTER FROM DOUBLE SCRATCHPAD WORD | 1 | |
| MOVE EIGHT-BIT LITERAL | 1 | |
| MOVE TWENTY-FOUR-BIT LITERAL | 2 | |
| MOVE REGISTER TO REGISTER | 1 | |
| NO OPERATION | 1 | |
| NORMALIZE X REGISTER | 1 | 2 |
| OVERLAY CONTROL MEMORY | 5 | 3 |
| READ OR WRITE MAIN MEMORY | 5/4 | 4 |
| SCRATCHPAD MOVE | 1 | |
| SCRATCHPAD RELATE | 1 | |
| SET CYF REGISTER | 1 | |
| SHIFT OR ROTATE T REGISTER LEFT | 1 | |
| SHIFT OR ROTATE X OR Y REGISTER LEFT OR RIGHT | 1 | 2 |
| SHIFT X AND Y REGISTERS LEFT OR RIGHT | 1 | 2 |
| SKIP WHEN | 1 | |
| STORE F REGISTER INTO DOUBLE SCRATCHPAD WORD | 1 | |
| SWAP F REGISTER WITH DOUBLE SCRATCHPAD WORD | 2 | |
| SWAP REGISTER WITH MAIN MEMORY | 4 | 5 |

NOTES

The basic clock of the B 1720 is 6 mega hertz.

1. If the relative address is not within control memory (therefore in main memory), there are two clocks.
2. There is one clock per bit.
3. There are five clocks per 16 bits (one micro instruction) plus five clocks.
4. READ is five clocks until the processor receives the data. WRITE is four clocks until the processor is released. Some instructions may be performed during the processor READ or WRITE command times if they immediately follow the READ or WRITE commands. This is called "concurrency." Consecutive READ or WRITE commands operate at MAIN MEMORY READ cycle speed (four clocks) or WRITE cycle speed (six clocks) respectively.
5. The data is presented to the processor and is released in one MAIN MEMORY READ cycle. Concurrent execution of certain micro instruction is performed if they immediately follow the SWAP command. The WRITE portion of the SWAP command is begun and performed in parallel to the READ portion, and main memory is not available for the duration of a WRITE cycle. For consecutive main memory commands, refer to note 4.

APPENDIX I
RESERVED WORDS AND SYMBOLS

| | | | | | |
|-----------|------------|-----------|---------|-------|-------|
| = | COUNT | JUMP | PORT | S8A | T |
| ≠ | CP | L | READ | S8B | TA |
| | CPL | LA | REVERSE | S9 | TAS |
| - | CPU | LABEL | RIGHT | S9A | TB |
| | CYD | LB | ROTATE | S9B | TC |
| - | CYF | LC | S | S10 | TD |
| (| CYL | LD | S0 | S10A | TE |
|) | DATA | LE | S0A | S10B | TEST |
| * | DEC | LEFT | S0B | S11 | TF |
| A | DIFF | LF | S1 | S11A | TO |
| ALL | DOWN | LIT | S1A | S11B | TOPM |
| ALL CLEAR | EOR | LOAD | S1B | S12 | TRUE |
| AND | EQL | LR | S2 | S12A | UNIT |
| ANY | EXIT | LSBX | S2A | S12B | UP |
| | DIFFERENCE | LSBY | S2B | S13 | WHEN |
| BIAS | EXTRACT | M | S3 | S13A | WITH |
| BICN | F | MAXM | S3A | S13B | WRITE |
| BITS | FA | MAXS | S3B | S14 | X |
| BR | FALSE | | S4 | S14A | XANY |
| BY | FB | | S4A | S14B | XCH |
| C | FL | | S4B | S15 | XEOY |
| CA | FLC | | S5 | S15A | XORY |
| CALL | FLD | MOVE | S5A | S15B | XY |
| CARRY | FLF | MSBX | S5B | SET | XYCN |
| CB | FT | MSKX | S6 | SFL | XYST |
| CC | FU | MSKY | S6A | SFU | Y |
| CD | GO | MSMA | S6B | SHIFT | U |
| CLEAR | IF | NORMALIZE | S7 | SKIP | |
| CMND | INC | NULL | S7A | STORE | |
| CMPX | INCN | OR | S7B | SUM | |
| CMPY | INTO | OVERLAY | S8 | SWAP | |

NOTE

There are no reserved words as far as labels are concerned; however, no word used in the MIL syntax may be used as a define or macro identifier or as a module-option toggle.

APPENDIX J

GLOSSARY

| | |
|------------------------------|--|
| Control memory | the high speed memory portion of main memory which contains the firmware. |
| Emulator | a virtual machine. |
| Firmware | a set of interpreters. |
| Horizontal micro programming | Each micro instruction is composed of bits which directly gate hardware. An extensive knowledge of the hardware gates and their trimming is required by the programmer. A high degree of parallelism is usually achieved similar to the conventional trimming and control circuits of a wired central processor. |
| Host machine | the micro programmed computer with a writable control memory. |
| Interpreter | a set of micro program routines which describe a computer or system architecture. |
| Micro instruction | a bit string directly executable by the host machine hardware. |
| Micro programming | "a means for programming a computer hardware architecture." (By WILKES, M.V. <u>The Best Way to Design an Automatic Calculating Machine</u> . Manchester University, Computer Inaugural Conference 1951) |

S-language an object code of the virtual machine.

S-machine a virtual machine.

S-memory that portion of the main memory in use by the virtual machine and containing the S-language, the data, and any other information required by the interpreter to operate the virtual machine.

Vertical micro programming Sets of registers are conceived and manipulated by sequences of micro instructions. Micro programs more closely resemble typical programming techniques. The usual form is interpretive with the virtual machine holding operators and operands. The micro instructions fetch and execute operators which produce the transformations upon the operands.

Virtual machine the effective computer architecture as seen by the user.

INDEX

Active Registers, 3-5
 Address, 3-7
 Base, 3-7
 Control, 3-8
 Field, 3-6
 Limit, 3-7
 Local, 3-6
 Memory Base, 3-8
 Micro Instruction, 3-7
 Top of Control Memory, 3-8
 Transform, 3-6
 X-Y, 3-6

Address Matrix Notes for B 1710, B-2
Address Matrix Notes for B 1720, C-2
Address Register, 3-7
ADD SCRATCHPAD, 4-4
ADJUST, 4-3
Alphabetical Listing of Registers, 3-1
AND, 4-5
Any-Interrupt, 3-15
Assembly Coding Form, 2-1
A Stack, 3-7

B 1710 Condition Registers, B-1
B 1710 Hardware Tables, B-1
B 1710 Micro Instruction Timing, H-1
B 1710 Processor, B-3
B 1710 Register Addressing, B-1
B 1720 Condition Registers, C-1
B 1720 Hardware Tables, C-1
B 1720 Micro Instruction Timing, H-3

B 1720 Register Addressing, C-1
Base Register, 3-7
BIAS, 4-7, G-1
Binary Conditions Register, 3-14
BIT TEST RELATIVE BRANCH FALSE, G-2
BIT TEST RELATIVE BRANCH TRUE, G-3
Braces Used in Syntax, 4-1
Brackets Used in Syntax, 4-1
BRANCH RELATIVE, G-4

CALL, 4-9, G-5
CARRY, 4-10
CASSETTE, 4-11
CASSETTE CONTROL, G-6
CLEAR, 4-12

INDEX (cont)

CLEAR REGISTERS, G-7
CMPX Result Register, 3-10
CMPY Result Register, 3-10
Combinatorial Logic, 3-9
Command Register, 3-13
Compiler Control Card, A-1
Compiler Operations, A-1
COMPLEMENT, 4-13
Concurrent Execution of Micro Operators, 4-2
Condition Registers, 3-13
 B 1710 Condition, B-1
 B 1720 Condition, C-1
 Binary Conditions, 3-14
 Field Length Conditions, 3-17
 Interrupt Conditions, 3-17
 XY Conditions, 3-14
 XY States, 3-15
Conditions, 2-1
Console Cassette Tape Input Register, 3-12
Console Interrupt, 3-16
Constant Registers, 3-12
 Maximum Control Memory, 3-12
 Maximum Main Memory, 3-12
Control Register, 3-8
Conventions Used in Syntax, 4-1
COUNT, 4-15
COUNT FA/FL, G-8

Data Register, 3-13
DEC, 4-16
DEFINE, 4-17
Defined Field Concepts, 1-2
DEFINE-VALUE, 4-18
Difference Result Register, 3-11
DISPATCH, 4-19, G-9

EOR, 4-21
Error Message, F-1
EXIT, 4-22
EXTRACT, 4-23
EXTRACT FROM REGISTER T, G-11

Field and Subfield Organization, 3-18
Field Length Conditions Register, 3-17
Field Registers, 3-6
Function Box, 3-9

Glossary, J-1
GO TO, 4-25

INDEX (cont)

HALT, 4-26, G-12
Hardware Instruction Formats, G-1
 BIAS, G-1
 BIT TEST RELATIVE BRANCH FALSE, G-2
 BIT TEST RELATIVE BRANCH TRUE, G-3
 BRANCH RELATIVE, G-4
 CALL, G-5
 CASSETTE CONTROL, G-6
 CLEAR REGISTERS, G-7
 COUNT FA/FL, G-8
 DISPATCH, G-9
 EXTRACT FROM REGISTER T, G-11
 HALT, G-12
 LOAD F FROM DOUBLEPAD WORD, G-13
 MANIPULATE 4-BIT, G-14
 MOVE 8-BIT LITERAL, G-15
 MOVE 24-BIT LITERAL, G-16
 NO OPERATION, G-17
 NORMALIZE X, G-18
 OVERLAY CONTROL MEMORY, G-19
 READ/WRITE MEMORY, G-20
 REGISTER MOVE, G-21
 SCRATCHPAD MOVE, G-22
 SCRATCHPAD RELATE, G-23
 SET CYF, G-24
 SHIFT/ROTATE REGISTER T LEFT, G 25
 SHIFT/ROTATE REGISTERS X AND Y LEFT/RIGHT, G-26
 SHIFT/ROTATE REGISTER X OR Y LEFT/RIGHT, G-27
 SKIP WHEN, G-28
 STORE F INTO DOUBLEPAD WORD, G-29
 SWAP DOUBLEPAD WORD WITH REGISTER F, G-30
 SWAP MEMORY, G-31
Hardware Tables for B 1710, B-1
Hardware Tables for B 1720, C-1

IF, 4-27
INC, 4-33
Input/Output Registers, 3-12
 Command, 3-13
 Console Cassette Tape Input, 3-12
 Data, 3-13
Input/Output Result Descriptors, E-1
Instruction Formats for B 1700 Hardware, G-1
Interpretation of the Virtual Language, 1-2
Interrupt Conditions Register, 3-17
Interrupts,
 Any, 3-15
 Console, 3-16

INDEX (cont)

- Main Memory Read Parity Error, 3-16
- READ Address Out of Bounds, 3-16
- WRITE/SWAP Address Out of Bounds, 3-17
- Introduction, xi

- JUMP, 4-34

- Key Words Used in Syntax, 4-1

- Limit Register, 3-7
- LIT, 4-35
- Literals, 2-4
- LOAD, 4-36
- LOAD F FROM DOUBLEPAD WORD, G-13
- LOAD MSMA, 4-37
- LOAD SMEM, 4-39
- Local Registers, 3-6
- Lower Case Words Used in Syntax, 4-1

- MACROS, 4-40
- Main Memory Address Out of Bounds Override, 3-16
- Main Memory Read Parity Error Interrupt, 3-16
- MANIPULATE 4-BIT, G-14
- Maximum Control Memory Register, 3-12
- Maximum Main Memory Register, 3-12
- Memory Base Register, 3-8
- MICRO, 4-42
- Micro Implementation Language, 2-1
 - Assembly Coding Form, 2-1
 - Conditions, 2-1
 - LABELS, 2-3
 - LITERALS, 2-4
 - STRINGS, 2-5
- Micro Implementation Language Compiler Operation, A-1
- Micro Implementation Language Error and Warning Messages, F-1
- Micro Instruction Register, 3-7
- Micro Instructions, 1-1, D-1
- Micro Instruction Timing for B 1710, H-1
- Micro Instruction Timing for B 1720, H-3
- Micro Operators, 4-1
 - ADJUST, 4-3
 - ADD SCRATCHPAD, 4-4
 - AND, 4-5
 - BIAS, 4-7
 - CALL, 4-9
 - CARRY, 4-10

INDEX (cont)

CASSETTE, 4-11
CLEAR, 4-12
COMPLEMENT, 4-13
COUNT, 4-15
DEC, 4-16
DEFINE, 4-17
DEFINE-VALUE, 4-18
DISPATCH, 4-19
EOR, 4-21
EXIT, 4-22
EXTRACT, 4-23
GO TO, 4-25
HALT, 4-26
IF, 4-27
INC, 4-33
JUMP, 4-34
LIT, 4-35
LOAD, 4-36
LOAD-MSMA, 4-37
LOAD-SMEM, 4-39
MACROS, 4-40
MICRO, 4-42
MOVE, 4-43
NOP, 4-46
NORMALIZE, 4-47
OR, 4-48
OVERLAY, 4-49
READ, 4-50
RESET, 4-51
ROTATE OR SHIFT T, 4-52
ROTATE OR SHIFT X, Y AND XY, 4-54
SEGMENT, 4-55
SET, 4-56
SKIP, 4-58
STORE, 4-60
SUBTRACT SCRATCHPAD, 4-61
SWAP, 4-62
TABLE, 4-63
WRITE, 4-64
WRITE-STRING, 4-65
XCH, 4-66

Micro Processor Timing Tables, H-1
Micro Programming Concepts, 1-1
 Defined Field Concepts, 1-2
 General, 1-1
 Interpretation of the Virtual Language, 1-2
Micro Instructions, 1-1

INDEX (cont)

- Module Option \$ Card, A-3
- MOVE, 4-43
- MOVE 8-BIT LITERAL, G-15
- MOVE 24-BIT LITERAL, G-16
- MSKX Result Register, 3-10
- MSKY Result Register, 3-10

- NO OPERATION, G-17
- NOP, 4-46
- NORMALIZE, 4-47
- NORMALIZE X, G-18
- Notations and Conventions Used in Syntax, 4-1
 - Braces, 4-1
 - Brackets, 4-1
 - Key Words, 4-1
 - Lower Case Words, 4-1
- Null Register, 3-12

- OR, 4-48
- Organization of Fields and Subfields, 3-18
- OVERLAY, 4-49
- OVERLAY CONTROL MEMORY, G-19
- Override,
 - Main Memory Address Out of Bounds, 3-16

- Programming Techniques, 5-1
 - Virtual-Language Definitions, 5-1
 - Writing Rules, 5-1

- READ, 4-50
- READ Address Out of Bounds Interrupt, 3-16
- READ/WRITE MEMORY, G-20
- Register Addressing for B 1710, B-1
- Register Addressing for B 1720, C-1
- Register Designations and Applications, 3-17
 - Interrupt Controls, 3-18
 - Micro Instruction Controls, 3-18
 - Parallel Width Controls, 3-18
 - S-Memory Controls, 3-18
- REGISTER MOVE, G-21
- Registers, 3-1
 - Active, 3-5
 - Address, 3-7
 - Alphabetical Listing of, 3-1
 - Base, 3-7
 - Binary Conditions, 3-14

INDEX (cont)

CMPX Result, 3-10
CMPY Result, 3-10
Command, 3-13
Condition, 3-13
Console Cassette Tape Input, 3-12
Constant, 3-12
Control, 3-8
Data, 3-13
Difference Result, 3-11
Field, 3-6
Field Length Conditions, 3-17
General, 3-1
Input/Output, 3-12
Interrupt Conditions, 3-17
Limit, 3-7
Local, 3-6
Maximum Control Memory, 3-12
Maximum Main Memory, 3-12
Memory Base, 3-8
Micro Instruction, 3-7
MSKX Result, 3-10
MSKY Result, 3-10
Null, 3-12
Result, 3-9
SUM Result, 3-10
Top of Control Memory, 3-8
Transform, 3-6
XANY Result, 3-9
XEOY Result, 3-10
XORY Result, 3-9
X-Y, 3-6
XY Conditions, 3-14
XY States, 3-15
Reserved Words and Symbols, I-1
RESET, 4-51
Result Descriptors for Input/Output, E-1
Result Registers, 3-9
 CMPX, 3-10
 CMPY, 3-10
 Difference, 3-11
 MSKX, 3-10
 MSKY, 3-10
 SUM, 3-10
 XANY, 3-9
 XEOY, 3-10
 XORY, 3-9

INDEX (cont)

ROTATE OR SHIFT T, 4-52
ROTATE OR SHIFT X, Y AND XY, 4-54

Scratchpad, 3-11
SCRATCHPAD MOVE, G-22
SCRATCHPAD RELATE, G-23
SEGMENT, 4-55
SET, 4-56
SET CYF, G-24
SHIFT/ROTATE REGISTER T LEFT, G-25
SHIFT/ROTATE REGISTERS X AND Y LEFT/RIGHT, G-26
SHIFT/ROTATE REGISTER X OR Y LEFT/RIGHT, G-27
SKIP, 4-58
SKIP WHEN, G-28
STORE, 4-60
STORE F INTO DOUBLEPAD WORD, G-29
Strings, 2-5
Subfield and Field Organization, 3-18
SUBTRACT SCRATCHPAD, 4-61
SUM Result Register, 3-10
SWAP, 4-62
SWAP DOUBLEPAD WORD WITH REGISTER F, G-30
SWAP MEMORY, G-31
Syntax Notations and Conventions, 4-1
 Braces, 4-1
 Brackets, 4-1
 Key Words, 4-1
 Lower Case Words, 4-1

TABLE, 4-63
Timing Tables for Micro Processor, H-1
Top of Control Memory Register, 3-8
Transform Registers, 3-6

Virtual-Language Definitions, 5-1

Warning Messages, F-2
WRITE, 4-64
WRITE-STRING, 4-65
WRITE/SWAP Address Out of Bounds Interrupt, 3-17
Writing Rules, 5-1

XANY Result Register, 3-9
XCH, 4-66
XEOY Result Register, 3-10
XORY Result Register, 3-9
XY Conditions Register, 3-14
X-Y Registers, 3-6
XY States Register, 3-15

BURROUGHS CORPORATION
DATA PROCESSING PUBLICATIONS
REMARKS FORM

TITLE: B 1700 SYSTEMS MICRO
IMPLEMENTATION LANGUAGE (MIL)
Reference Manual

FORM: 1072568
DATE: 12-73

CHECK TYPE OF SUGGESTION:

ADDITION

DELETION

REVISION

ERROR

GENERAL COMMENTS AND/OR SUGGESTIONS FOR IMPROVEMENT OF PUBLICATION:

FROM: NAME _____
TITLE _____
COMPANY _____
ADDRESS _____

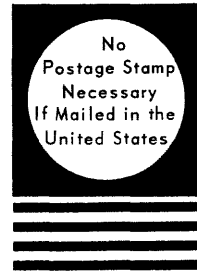
DATE _____

STAPLE

FOLD DOWN

SECOND

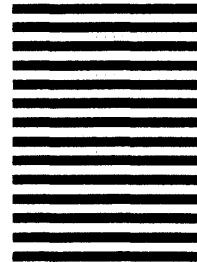
FOLD DOWN



BUSINESS REPLY MAIL
First Class Permit No. 817, Detroit, Mich. 48232

Burroughs Corporation
Burroughs Place
Detroit, Michigan 48232

attn: Systems Documentation
Technical Information Organization, TIC-Central



FOLD UP

FIRST

FOLD UP



*Wherever There's
Business There's*

Burroughs