GENERAL SPECIFICATION FOR THE B8500 INFORMATION PROCESSING SYSTEM

1761-2045

1 October 1969

Burroughs Corporation

Defense, Space and Special Systems Group

PAOLI, PA.

Burroughs Corporation

LARGE COMPUTER SYSTEMS ORGANIZATION
PAOLI, PA.
PRODUCT SPECIFICATION

1761-2045

B8500 GENERAL SPECIFICATION

REVISIONS

REV LTR	REVISION ISSUE DATE	PAGES REVISED ADDED DELETED OR CHANGE OF CLASSIFICATION	PREPARED BY	APPROVED BY
A	9/19/68	Original Issue	[illegible] 9-19-68 [illegible] 9-19-68	[illegible] 9/19/68
AB	6/9/69	This issue of specification 1761-2045 supersedes the original issue of this specification dated September 1968 and incorporates changes and additions authorized under ECN 2222.	[illegible] [illegible] 6-1-69	[illegible] 6/6/69 Released by: [illegible]
AC	10/1/69	This issue of specification 1761-2045 supersedes revision AB of this specification dated 9 June 1969 and incorporates changes and additions authorized under ECN 2337. Many minor corrections and clarifications have been made. The major changes are as follows: Section 2: Revision of the procedure display. Indirect reference entries in resource stack replaced by read-write inhibit control.	[illegible] 9-26-69 JPV 9-26-69	[illegible] 9/26/69 Released by: [illegible]

1761-2045

B8500 GENERAL SPECIFICATION

REVISIONS

REV LTR	REVISION ISSUE DATE	PAGES REVISED ADDED DELETED OR CHANGE OF CLASSIFICATION	PREPARED BY	APPROVED BY
AC (Cont)		Addition of base-of-name-stack relative names and program control relative names. Revision of list structure to include reset and sequence operations. Section 4: IMP functions are redefined. Section 5: Addition of chained requests. Section 6: Addition of chained requests.		

GENERAL SPECIFICATION

FOR

THE B8500 INFORMATION PROCESSING SYSTEM

1761-2045

1 October 1969

Burroughs Corporation

Defense, Space and Special Systems Group

PAOLI, PA.

TABLE OF CONTENTS

PREFACE

THIS SPECIFICATION GIVES THE READER AN OVERALL VIEW OF THE B8500 SYSTEM. THE SPECIFICATION EXPRESSES DESIGN PHILOSOPHY IN CONCERT WITH IMPLEMENTATION DETAIL. IT CONTAINS NO HARDWARE SPECIFICATIONS BUT DISCUSSES THE DESIGN CONCEPT OF EACH MAJOR SUBSYSTEM AND THEN LISTS REFERENCES TO THE APPROPRIATE HARDWARE SPECIFICATION. THUS, BEFORE HE EXAMINES IMPLEMENTATION DETAILS, THE READER IS ABLE TO SEE FUNDAMENTAL IDEAS USED.

THIS DOCUMENT IS INTENDED FOR THOSE WHO MUST IMPLEMENT THE SOFTWARE FOR THE SYSTEM, BUT IT MAY SERVE AS A BASIS FOR THE PRODUCTION OF MATERIAL FOR MARKETING AND FOR USERS INFORMATION.

A NEW TERMINOLOGY HAS BEEN INTRODUCED TO EMPHASIZE THE NEW RELATIONSHIPS BETWEEN CERTAIN PROCESSING COMPONENTS. THE NEW TERMINOLOGY OFTEN CONVEYS MEANINGS WHICH CAN BE MISSED IF NOT CAREFULLY OBSERVED.

THE B8500 DESIGN AROSE FROM A GROWING AWARENESS OF THE DEFICIENCIES OF CONVENTIONAL MACHINE DESIGN AND FROM THE KNOWLEDGE OF THE REQUIREMENTS OF ADVANCED APPLICATIONS. USERS NOW REQUIRE VERY COMPLEX DATA STRUCTURES (QUEUES, STACKS, ETC.) AND FORMATS AS WELL AS SOPHISTICATED PROGRAM STRUCTURES. THEY REQUIRE THE HANDLING OF LARGE RELATED DATA SETS AND MANY FORMS OF REMOTE MULTIACCESS DEVICES. THEY ALSO, OF COURSE, REQUIRE ALL THE CAPABILITIES OF CONVENTIONAL SYSTEMS.

THE DESIGN EFFORT ENCOMPASSED AN EVALUATION OF THE PRINCIPLES ON WHICH COMPUTERS ARE BASED AND LED TO THE CONCEPTS OUTLINED IN SECTION 1. TYPICAL OF THE ENDEAVOR WAS THE EXAMINATION OF THE USE OF A LATTICE (WORD SIZE OR CHARACTER SIZE) AS THE FUNDAMENTAL UNIT OF STORAGE. THIS INVESTIGATION LED TO THE IDEA OF FREE FORMAT, VARIABLE-LENGTH FIELDS, AND THEIR IMPLEMENTATION THROUGH THE ASSOCIATED UNIT.

SECTION 1

DESIGN PHILOSOPHY

THE DESIGN PHILOSOPHY OF THE B8500 SYSTEM CAN BE SIMPLY EXPRESSED AS AN EXTENSION OF THE DESIGN PHILOSOPHY OF THE B500 SERIES. FOR MANY YEARS, THE BURROUGHS DESIGN TEAMS HAVE USED THE CONCEPT OF INTEGRATING HARDWARE AND SOFTWARE AS A DESIGN PRINCIPLE.

THIS SECTION PRESENTS SOME OF THE DETAILS OF THE DESIGN PHILOSOPHY BY DISCUSSING DESIGN OBJECTIVES, CONCEPTS DEVELOPED, AND THE CHARACTERISTICS OF THE MACHINE. BY INTENTION, SOME CONCEPTS ARE REPEATED IN VARIOUS SECTIONS TO CONVEY THESE IDEAS IN SEVERAL CONTEXTS.

DESIGN OBJECTIVES

THE DESIGN OBJECTIVES FOR THE B8500 ARE:

A. TO PROVIDE A TRULY GENERAL PURPOSE PRODUCT.

B. TO EXPLOIT THE IDEAS EMBODIED IN THE B500 SERIES.

C. TO PROVIDE HIGH SYSTEM PERFORMANCE.

D. TO PROVIDE COMPATIBILITY WITH THE PRODUCT LINE.

E. TO PROVIDE NEW CAPABILITIES.

F. TO PROVIDE LONG PRODUCT LIFE.

G. TO PROVIDE AN EXPANDABLE SYSTEM.

A TRULY GENERAL PURPOSE PRODUCT

THE B8500 SYSTEM WAS DESIGNED TO MEET THE REQUIREMENTS OF A BROAD MARKET WITH SUCH DIVERSE APPLICATIONS AS TIME SHARING (AMONG THOUSANDS OF TERMINALS), SCIENTIFIC PROBLEM SOLVING, AND DATA PROCESSING.

THE MACHINE MUST HAVE A HIGH DEGREE OF SENSITIVITY TO EACH PROBLEM TO MEET THESE REQUIREMENTS. THUS, THE MACHINE HANDLES COMPLEX DATA STRUCTURES WHICH MAY BE BOTH NESTED AND COMPOSED OF VARIABLE-LENGTH ELEMENTS.

THE MACHINE IS ALSO AMENABLE TO THE SOPHISTICATED PROGRAM STRUCTURES DICTATED BY CURRENT AND FUTURE HIGHER LEVEL LANGUAGES AND THE REQUIREMENTS OF ADVANCED PROBLEMS.

TYPICAL APPLICATIONS WHICH ARE PROVIDED FOR INCLUDE INVENTORY CONTROL, PAYROLL, AUTOMATED DESIGN, TEACHING, SIMULATION, BANKING, SCIENTIFIC DATA COLLECTION AND REDUCTION, NUMERICAL CONTROL OF MACHINES, AND SUPPORT OF NEW DEVICES SUCH AS GRAPHIC TERMINALS.

EXPLOITING IDEAS EMBODIED IN THE B500 SERIES

THE IDEAS EMBODIED IN THE B5500 HAVE BEEN PROVEN SUCCESSFUL, SINCE THAT MACHINE OUTPERFORMS ALL OTHER DESIGNS WITH THE SAME COMPONENT SPEED AND SIZE. THE B6500 DESIGN EXTENDED THOSE IDEAS. THE B8500 DESIGN PHILOSOPHY INCLUDES A FURTHER EXTENSION OF THOSE IDEAS.

THE B5500 PROVED THE VALIDITY OF USING ONLY HIGHER LEVEL LANGUAGES FOR BOTH USERS AND SYSTEMS PROGRAMMERS. THE SOFTWARE-ORIENTED HARDWARE WAS AN ASSET IN WRITING EFFICIENT COMPILERS WITH OPTIMUM OBJECT CODE.

THE USE OF THE STACK TO ALLOW RECURSIVE PROGRAMMING, RAPID EXPRESSION EVALUATION, STANDARD PARAMETER TECHNIQUES, SOPHISTICATED LOCAL STORAGE ALLOCATIONS, AND RAPID PROCEDURE ENTRY WITH STATE SAVING IN THE HARDWARE LED TO EFFICIENT MACHINE USAGE. THIS ALLOWED THE DYNAMIC HISTORY OF A PROCESS TO BE RECORDED IN THE STACK. BY HAVING CONTROL IN THE STACK AND SPECIAL CONTROL WORDS (DESCRIPTORS), THE CAPABILITY OF PLACING PROCEDURES IN THE ACCESS PATH TO DATA ALLOWED FOR MORE SOPHISTICATED STRUCTURES. ALL THESE ADVANCED B5500 IDEAS HAVE BEEN EXAMINED, GENERALIZED, AND APPLIED TO THE B8500 DEVELOPMENT.

HIGH SYSTEM PERFORMANCE

PERFORMANCE MEASUREMENT IS ELUSIVE. THIS DESIGN PROVIDES BALANCED USE OF A MAXIMUM PORTION OF THE HARDWARE FOR A MAJORITY OF THE TIME. THE DISTINCTION MADE HERE IS TO OPTIMIZE OVERALL SYSTEM THROUGHPUT

AND AVOID LOCAL OPTIMIZATION OF A SPECIFIC SECTION SUCH AS AN ARITHMETIC UNIT OR AN I/O CHANNEL. THIS REQUIRES THE DESIGNERS TO PROVIDE HARDWARE AND SOFTWARE THAT BALANCE THE SYSTEM AND OPTIMIZE THROUGHPUT.

CONCEPTS WHICH ARE NEW WITH THE B8500 AND AFFECT THIS BALANCE INCLUDE THE MAIN MEMORY EXTENSION, THE ABSENCE OF ABSOLUTE ADDRESSES IN THE PROCESS SPACE, AND THE USE OF VARIABLE-LENGTH FIELDS. THESE IDEAS GIVE PROPORTIONAL WEIGHT TO EDITING, INPUT/ OUTPUT, AND OPERATING SYSTEMS FUNCTIONS AS WELL AS TO SIMPLE ARITHMETIC PROCEDURES.

PRODUCT LINE COMPATIBILITY

THE USERS VIEW OF THE B8500 WILL BE VERY MUCH LIKE THAT OF THE B5500 AND B6500. THE SOURCE LANGUAGES ACCEPTED BY THE B8500 COMPILERS ARE COMPATIBLE WITH THOSE FOR THE B6500. THE CONTROL AND OPERATOR MESSAGES FOR COMMUNICATION WITH THE B8500 INCLUDE THOSE USED ON THE B6500. B5500 USERS ARE ABLE TO MOVE TO THE B6500 OR B8500 BY THE USE OF PROGRAM FILTERS TO ACCOUNT FOR THE CHANGES FROM B5500 SYNTAX.

THE B8500 PROVIDES FACILITIES BEYOND ANY OF THOSE OFFERED ON ANY OTHER BURROUGHS SYSTEM. THEY ARE AVAILABLE TO THE USER BY EXTENSIONS IN THE HIGHER-LEVEL LANGUAGES.

NEW CAPABILITIES

AN ESSENTIAL GOAL OF THIS DESIGN IS TO PROVIDE NEW CAPABILITIES IN DATA PROCESSING THAT WERE NOT PREVIOUSLY PRACTICAL. THIS GOAL IS A DIRECT RECOGNITION OF THE FACT THAT DATA IS NOT WELL-SUITED TO THE RIGOROUS REPRESENTATION REQUIRED BY WORD OR CHARACTER ORGANIZED STORAGE. THE ACTUAL NATURE OF DATA DEMANDS VARIABLE SIZE REPRESENTATIONS, AND ADVANCES IN TECHNOLOGY PERMIT EXPLOITATION OF THIS CONCEPT.

THE CONCEPT THAT DATA HAS VARIABLE AND UNIQUE CHARACTERISTICS, RATHER THAN BEING OF A DEFINITE FIXED NATURE, REQUIRES A SET OF FLEXIBLE PRIMITIVE DATA STRUCTURES. THE FORM OF EACH STRUCTURE WILL DESCRIBE THE DATA CONTAINED WITHIN THE STRUCTURE. SOME OF THE MORE COMMON STRUCTURES ARE FIELDS, VECTORS, STACKS, AND QUEUES.

THESE COMMON STRUCTURES MAY BE COMBINED INTO MORE COMPLEX STRUCTURES. THIS PERMITS INCREASED FLEXIBILITY AND SPACE OPTIMIZATION. THESE FLEXIBLE PRIMITIVE DATA STRUCTURES WILL PROVIDE AN EFFICIENT METHOD OF SPACE MANIPULATION.

TIME SHARING AND MULTIPROGRAMMING AMONG A LARGE NUMBER OF PROGRAMS IMPLY A NEW CAPABILITY TO PROVIDE RAPID SWAPPING AND PROTECTION. THESE CAPABILITIES ARE PROVIDED IN THE PROCESS STRUCTURE OF THE B8500.

ONE OF THE MOST IMPORTANT CAPABILITIES IS AN EXTENSION OF THE B5500 ACCIDENTAL ENTRY IDEA. IT IS ESSENTIALLY THE IDEA OF AN EVALUATING MACHINE. SUCH A MACHINE CAN, AT ANY POINT IN THE ACCESS PATH TO DATA, ALLOW PROCEDURES TO BE INTRODUCED. THIS IDEA PRODUCES THE ABILITY TO HAVE INSTRUCTIONS OR DESCRIPTIONS OF ANY FORMAT AND COMPLEXITY. PROGRAMS AND DATA MAY THUS BE STRUCTURED IN ANY WAY.

LONG PRODUCT LIFE

BURROUGHS PRODUCTS ARE DESIGNED WITH THE USER IN MIND. FOR THIS REASON, CONSIDERATION IS GIVEN TO THE LONG LIFE OF THE PRODUCT. LONG LIFE APPEARS IN COMPATIBILITY FROM B3500 TO B5500 TO B6500 TO B8500... WHILE THIS COMPATIBILITY IS NOT COMPLETE, IT GOES A LONG WAY IN KEEPING PROGRAMS IN SERVICE.

A SECOND FORM OF LONG LIFE IS IN THE IDEA OF MODULARITY, WHICH ALLOWS THE INTRODUCTION OF MODERNIZED MODULES IN AN OLDER SYSTEM WITHOUT REPLACING THE ENTIRE SYSTEM. SYSTEMS MAY ALSO BE EXPANDED WITHOUT REPROGRAMMING.

THE ADVANCED DESIGN ALLOWS INTRODUCTION OF MODERN COMPLEX APPLICATIONS ON EXISTING EQUIPMENT, AS DEMONSTRATED BY THE B5500. THE B5500 IS STILL EXPANDING ITS APPLICATIONS AFTER MANY YEARS OF SERVICE.

EXPANDABLE SYSTEM

SYSTEM EXPANDABILITY IS BASED ON THE CONCEPT OF MODULARITY. MODULARITY PERMITS THE USER TO START WITH A SMALL SYSTEM AND TO EXPAND WITH NO PROGRAM CHANGES UP TO THE COMPLETE SYSTEM SIZE. IT ALSO PERMITS SUBSTITUTION OF NEW COMPONENTS FOR OLD COMPONENTS, AND THEREBY ALLOWS GRADUAL UPGRADING OF THE SYSTEM.

MODULARITY PERMITS, AS A NATURAL BY-PRODUCT, THE CAPABILITY TO OPERATE WITH A REDUCED CONFIGURATION, WHICH, IN TURN, PERMITS GRACEFUL DEGRADATION OF THE SYSTEM IF COMPONENTS BECOME UNAVAILABLE.

B8500 SYSTEM CONCEPTS

GENERAL

THERE ARE SEVERAL GENERAL IDEAS, APPLICABLE TO SYSTEMS DESIGN, WHICH ARE ADHERED TO IN THE B8500.

A. SIMPLE SOLUTIONS ARE CLEANER AND THUS MORE LIKELY TO BE SUCCESSFUL. THE POWERFUL CONCEPT OF RECURSION ALLOWS THE IMPLEMENTATION OF COMPLEX STRUCTURES FROM THE RECURSIVE USE OF SIMPLE STRUCTURES. RECURSION APPLIES BOTH IN HARDWARE AND SOFTWARE.

B. PROCESSES, HIERARCHIES OF ASYNCHRONOUS COOPERATING ROUTINES, HAVE BEEN USED IN COMMUNICATIONS AND SYSTEM ARCHITECTURE. PROCESSES APPEAR OFTEN IN SOFTWARE, SUCH AS IN SIMULATIONS, COMPILERS, AND OPERATING SYSTEMS.

C. ALL LANGUAGES AND ALL DATA REPRESENTATIONS HAVE CERTAIN COMMON ATTRIBUTES. THESE ATTRIBUTES MAY BE ISOLATED FROM THEIR PARTICULAR REPRESENTATION AND USED AS A BASE FOR A GENERAL MACHINE DESIGN. THESE BASIC ATTRIBUTES FORM THE BASIS FOR PRIMITIVE OPERATORS WHICH HAVE WIDE APPLICABILITY. THE BASIC ATTRIBUTES FORM A THEORETICAL BASE FOR THE STRUCTURE CONCEPT.

D. ALL COMPUTER PROCESSES ARE COMPOSED SOLELY OF EITHER NAMES OR OBJECTS. NAMES ARE THE MEANS BY WHICH OBJECTS ARE REFERENCED. THE EVALUATION CONCEPT GENERALIZES THE DERIVATION OF NAMES AND ALLOWS FOR ARBITRARY CONTROLS TO BE EFFECTIVE ALONG AN ACCESS PATH.

E. THE CONCEPT OF A STORAGE HIERARCHY IS THAT THE MOST ACTIVE SEGMENTS ARE KEPT IN THE HIGHEST-SPEED STORAGE AND THE LEAST ACTIVE ARE KEPT IN THE LOWEST-SPEED STORAGE. SEGMENTS MAY BE TRANSFERRED WITHIN THE STORAGE HIERARCHY AT THE SAME TIME THAT OTHER SEGMENTS ARE BEING EXECUTED. IN THE B8500, THREE LEVELS OF STORAGE ARE RECOGNIZED. SEGMENTS CONTAINED IN LEVEL-3 STORAGE (THE FAMILY OF PERIPHERALS) MAY, AT ANY INSTANT, ALSO BE CONTAINED IN LEVEL-1 OR LEVEL-2 STORAGE.

PROGRAMMING CONCEPTS

TWO CONCEPTS ARISE SOLELY FROM A KNOWLEDGE OF THE SCOPE OF APPLICATIONS PRACTICED BY COMPUTER USERS. THESE INVOLVE THE PROCESS CONCEPT AND THE DATA STRUCTURE CONCEPT. THESE APPLICATION-ORIENTED IDEAS HAVE BEEN KEPT IN MIND THROUGHOUT THE DESIGN EFFORT.

A. PROCESS STRUCTURE: THE PROCESS CONCEPT IS DISCUSSED FULLY IN SECTION 4, BUT SOME OF THE IDEAS ARE SUMMARIZED HERE.

PROCESS STRUCTURE MAKES POSSIBLE THE PROTECTION NEEDED IN MULTIPROCESSING SYSTEMS. PRACTICAL SWAPPING IS ACHIEVED BY DEFERRING ADDRESS BINDING. THE PROCESS CONCEPT ALSO ALLOWS STRUCTURING THE PROGRAMS INTO ASYNCHRONOUS, COOPERATING, INDEPENDENTLY-COMPILED SUBPROCESS. THEREFORE, MORE GENERAL PROGRAM RELATIONSHIPS ARE OBTAINED.

THE PROCESS TECHNIQUE ALSO IS DESIGNED TO ENABLE A GENERALIZED TREATMENT OF RESOURCE MANAGEMENT. IT GIVES A SIMPLE, GENERAL WAY TO DELEGATE AUTHORITY TO SUBPROCESSES WITHOUT VIOLATING THE PROTECTION OF OTHER PROCESSES.

B. DATA STRUCTURES: THE TREATMENT OF DATA HAS CONTINUOUSLY BEEN A PROBLEM IN THE PAST DUE TO THE INABILITY TO REPRESENT DATA PROPERLY IN THE STORAGE MEDIA PROVIDED. THE B8500 ALLOWS COMPLETE SEGMENTATION OF ANY SIZE (TO THE BIT), AND SUCH SEGMENTS MAY BE NESTED OR FRAGMENTED TO ANY DEPTH. THE DATA STRUCTURE CAN BE DYNAMIC, PERMITTING DEPTH OF DATA BOTH IN THE SENSE OF SPACE AND IN THE SENSE OF TIME WHERE THE ACCESS IS PROGRAM DEPENDENT.

THE VARIABLE SIZE FIELDS ALSO PERMIT SAVINGS IN UTILIZATION OF STORAGE BECAUSE OF THE COMPACTION OF DATA TO ONLY THE NECESSARY SPACE.

THE DATA SENSITIVITY BUILT INTO THE MACHINE IS ALSO ENHANCED BY THE GENERALIZATION OF THE DESCRIPTION CONCEPT. RECOGNIZING THAT DESCRIPTIONS ARE ACTUALLY SPECIAL CASES OF PROGRAM, THE DESIGNERS OF THE B8500 VIEW ALL DESCRIPTIONS AS PROGRAM, WHOSE EVALUATION PRODUCES THE DESIRED ITEM.

HARDWARE DEFINITION OF FORMAT HAS BEEN RELAXED. THE USER MAY DEFINE THE SIZE AND FORMATS OF THE OPERANDS WITH A GREAT DEAL OF FLEXIBILITY. THE FORMATS OF ARITHMETIC OPERANDS ARE DEFINED BY THE DESCRIPTIONS. THE USER MAY THEN OPERATE ON OPERANDS IN A STANDARD WAY TO HANDLE INPUT OPERANDS OF SEVERAL FORMATS WITHOUT PROGRAMMATIC CONVERSION TO HIS STANDARD FORMATS.

THE CONCEPTS INVOLVING THE GENERALIZATION OF THE FORMATS AND SIZES OF OPERANDS IS IN CONCERT WITH THE CONCEPTS OF RECURSIVELY DEFINED VARIABLE-SIZED FIELDS. THIS FREEING OF STORAGE CONVENTIONS FROM WORD OR CHARACTER SIZE RESTRICTIONS IS ONE OF THE SIGNIFICANT ADVANTAGES OFFERED BY THE B8500 OVER ITS CONTEMPORARIES.

B8500 CHARACTERISTICS

TO CLOSE THE DISCUSSION OF THE DESIGN PHILOSOPHY, A SUMMARY OF THE CHARACTERISTICS OF THE B8500 IS OFFERED TO EMPHASIZE USER-ORIENTED FEATURES OF THE SYSTEM.

A. FREE FIELD STORAGE IS ORGANIZED TO FREE THE USER FROM WORD SIZE OR FIELD SIZE RESTRICTIONS. THIS FREEDOM ALLOWS FIELDS OF ANY SIZE WHICH CAN BE NESTED ARBITRARILY DEEP. OPERANDS MAY BE OF ANY SIZE AND FORMAT. A CHOICE OF ARITHMETIC REPRESENTATION IS ALSO GIVEN.

B. PROGRAM LANGUAGES ARE BETTER ACCOMMODATED BECAUSE OF THE SELECTION OF A SET OF PRIMITIVES SUITED TO ALL LANGUAGES. THIS IS PARTICULARLY ADVANTAGEOUS FOR COMPLICATED CONSTRUCTS SUCH AS PROCEDURES AND COROUTINES. THE ISOLATION OF COMMON PRIMITIVES HAS ALSO BEEN EXTENDED TO COMMON DATA TYPES, THEREBY PROVIDING FLEXIBLE STRUCTURES FOR DATA STORAGE.

C. THE MACHINE PRIMITIVES UTILIZE PUSHDOWN STACKS FOR ADDRESS PREPARATION AND FOR EXPRESSION EVALUATION.

D. THE I/O ACCOMMODATES MANY PERIPHERALS AND REMOTE ACCESS LINES AS WELL AS A LARGE CENTRAL DISK SYSTEM. MANY CHANNELS PER I/O MODULE AND MULTIPLE I/O MODULES ALLOW PARALLEL SIMULTANEOUS ACCESS TO HUNDREDS OF DEVICES. THE I/O MODULE IS TAILORED TO FACILITATE MULTIPLE PROCESSES AND TO REDUCE OPERATING SYSTEM ATTENTION TO I/O-COMPLETE INTERRUPTS. THE JOBS ARE QUEUED, AND MAY BE LINKED SO THAT MANY I/O TRANSFERS CAN TAKE PLACE BEFORE ANY OPERATING SYSTEM ATTENTION IS REQUIRED.

E. THE MEMORY EXTENSION SYSTEM IS DESIGNED TO FACILITATE EFFICIENT INTER-LEVEL INFORMATION TRANSFERS REQUIRED TO HANDLE TIME SHARING AND MULTIPROCESSING. LEVEL-1 SPACE MANAGEMENT AND LEVEL-2 SPACE MANAGEMENT ARE HANDLED BY IDENTICAL ALGORITHMS. THIS GIVES THE USER A NATURAL MEANS OF PLACING INFORMATION IN BACKUP STORAGE OR MAIN STORAGE.

SECTION 2

INTERPRETER

KERNEL

THE B8500 INTERPRETER AUTOMATICALLY ENTERS, REMOVES, AND LOCATES INFORMATION IN DATA AND PROGRAM STRUCTURES. THE STRUCTURES CHOSEN FOR AUTOMATIC MANAGEMENT ARE SIMPLE STRUCTURES THAT APPEAR IN MOST PROGRAMMING LANGUAGES. THE AUTOMATIC STRUCTURE HANDLING MECHANISM IMPLIES A CERTAIN HARDWARE ARCHITECTURE REFERRED TO AS THE "KERNEL" OF THE MACHINE.

EACH STRUCTURE IS DESCRIBED BY A DESCRIPTOR WHICH DEFINES THE ATTRIBUTES OF THE STRUCTURE. THESE ATTRIBUTES INCLUDE ACCESSING PATH ATTRIBUTES, REPRESENTATION ATTRIBUTES, AND STRUCTURE TYPE ATTRIBUTES. STRUCTURE TYPE ATTRIBUTES ARE EXECUTED IN ATTRIBUTE STRUCTURE STACKS (AS) TO YIELD A REFERENCE TO THE INFORMATION WITHIN THE SPECIFIED STRUCTURE. THE REPRESENTATION ATTRIBUTES DETERMINE HOW THE INFORMATION IS TO BE INTERPRETED; THE ACCESSING PATH ATTRIBUTES DEFINE ACCESS PERMISSION TO THE INFORMATION.

THE KERNEL STARTS WITH A DESCRIPTOR. THE PART OF THE DESCRIPTOR BEING EXECUTED IS POINTED TO BY A DESCRIPTOR PROGRAM CONTROL REGISTER (DPCR). THE DESCRIPTOR MAY CALL ANOTHER DESCRIPTOR; THUS, NESTINGS OF THE DPCR OCCUR IN THE DPCR STACK. THE DIFFERENT ATTRIBUTE FIELDS OF THE DESCRIPTOR ARE EXTRACTED AND PLACED IN THE ATTRIBUTE STACKS FOR FURTHER EVALUATION.

THE ACCESS PERMISSION AND REPRESENTATION ATTRIBUTES ARE COLLECTED IN THE ATTRIBUTE COLLECTION STACK (ACS) FOR FUTURE REFERENCE. THE STRUCTURE TYPE IS STORED IN THE DESCRIPTOR STRUCTURE TYPE REGISTER. THE STRUCTURE TYPE WILL DEFINE THE ALGORITHM USED TO ENTER, LOCATE OR REMOVE INFORMATION IN THE STRUCTURE. THE DATA FIELDS OF THE DESCRIPTOR WHICH DEFINE THE DYNAMIC STATUS OF STRUCTURES ARE REFERRED TO AS THE STATE WORD (SW).

THE SW DEFINES THE CONTAINER OF THE STRUCTURE. THIS CONTAINER INFORMATION IS PLACED IN THE ATTRIBUTE CONTAINER START ADDRESS STACK (AC) AND THE ATTRIBUTE CONTAINER LENGTH STACK (LC). THE SW ALSO CONTAINS INFORMATION WHICH WILL BE USED TO DEFINE THE INFORMATION ELEMENT DESIRED IN THE STRUCTURE. THIS DATA IS PLACED

IN THE ATTRIBUTE ELEMENT START ADDRESS STACK (AE) AND THE ATTRIBUTE ELEMENT LENGTH STACK (LE).

THESE STACKS (AC, LC, AE, AND LE) MAKE UP THE ATTRIBUTE STACK (SEE FIGURE 2-1) AND ARE UTILIZED BY THE DESCRIPTOR EVALUATION ARITHMETIC UNIT. THE DESCRIPTOR EVALUATION ARITHMETIC UNIT PERFORMS THE NECESSARY ARITHMETIC IN THE STRUCTURE ALGORITHMS. AFTER A REFERENCE IS CALCULATED, THE DESCRIPTOR MAY BE UPDATED. THIS IS DONE BY RETURNING THE PROPER ATTRIBUTE STACK TO THE FIELD IN THE DESCRIPTOR POINTED TO BY THE DPCR.

THE ATTRIBUTE STACKS ARE FIXED IN SIZE. THE NUMBER OF ENTRIES IN EACH STACK IS DETERMINED BY THE MAXIMUM NUMBER OF ENTRIES REQUIRED BY THE ALGORITHM USING THE STACK. THE WIDTH OF EACH STACK IS LIMITED BY THE MAXIMUM LEVEL-2 STORAGE SIZE. THE DPCR STACK DEPTH DETERMINES THE DEPTH OF NESTING OF STRUCTURES.

THIS KERNEL IS THE BASIC PART OF THE B8500 INTERPRETER, AND THE REST OF THE MACHINE SHOULD BE CONSIDERED WITH THE KERNEL AS A REFERENCE.

STRUCTURES

A STRUCTURE IS AN ORDERING OF A SET OF ELEMENTS WHICH DEFINE THE ACCESSING OF AN ELEMENT IN THE SET. AN ELEMENT IS AN ACCESSIBLE, CONTIGUOUS SET OF BITS. ACCESSING IS THE TRANSFORMATION OF A NAME TO A PHYSICAL LOCATION. A SIMPLE ELEMENT IS AN ELEMENT WITH NO SUBSTRUCTURE.

SEVERAL STRUCTURES ARE HANDLED DIRECTLY BY KERNEL HARDWARE. THESE STRUCTURES INCLUDE FIXED-LENGTH AND VARIABLE-LENGTH FIELDS, LISTS AND VECTORS WITH FIXED-LENGTH ELEMENTS, FIELDS WITH PARAMETRIC POSITION AND LENGTH, AND COMPOSITE STRUCTURES.

ELEMENTS OF STRUCTURES ARE NOT RESTRICTED TO BEING SIMPLE ELEMENTS. EVERY STRUCTURE MAY HAVE ELEMENTS OF ANY STRUCTURE, INCLUDING THE SAME STRUCTURE. BY INCLUDING STRUCTURED ELEMENTS, INTERESTING NESTED STRUCTURES CAN BE DEFINED, SUCH AS ARRAYS (VECTORS OF VECTORS) AND RECORDS (FIELDS OF FIELDS OF FIELDS ...).

IT IS POSSIBLE TO GIVE A SET OF INFORMATION TWO OR MORE DIFFERENT INDEPENDENT STRUCTURES. SUCH STRUCTURES ARE CALLED COSPATIAL.

ALL MANIPULATIONS UPON AN INFORMATION STRUCTURE ARE PERFORMED WITH ONE OR MORE STRUCTURE DESCRIPTIONS.

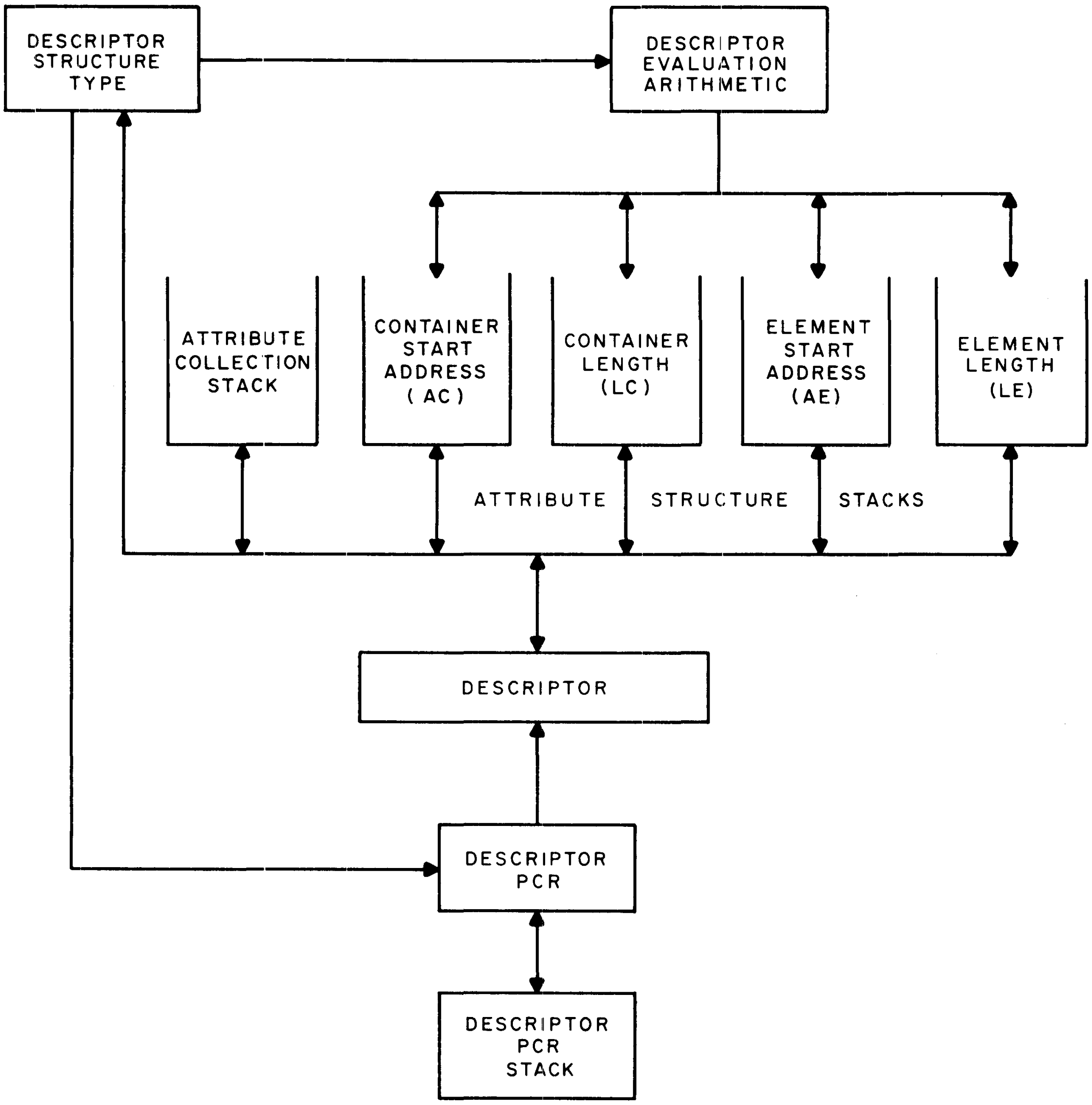

DESCRIPTOR STRUCTURE TYPE
DESCRIPTOR EVALUATION ARITHMETIC
ATTRIBUTE COLLECTION STACK
CONTAINER START ADDRESS (AC)
CONTAINER LENGTH (LC)
ELEMENT START ADDRESS (AE)
ELEMENT LENGTH (LE)
ATTRIBUTE STRUCTURE STACKS
DESCRIPTOR
DESCRIPTOR PCR
DESCRIPTOR PCR STACK

FIGURE 2-1. KERNEL

DESCRIPTION SYNTAX

BRACKETS ARE USED TO INDICATE SELF IDENTIFYING EXPRESSIONS WHICH ARE NOT DETAILED FURTHER IN ANY SYNTAX.

```
DESCRIPTION ::= ACCESS ATTRIBUTES, STRUCTURE EXPRESSION,
                INTERPRETER ATTRIBUTES

ACCESS ATTRIBUTES::= READ FAULT INDICATOR, WRITE FAULT
                     INDICATOR

READ FAULT INDICATOR ::= <NO READ FAULT>/
                         <READ FAULT>, FAULT PROCEDURE NAME

WRITE FAULT INDICATOR ::= <NO WRITE FAULT>/
                          <WRITE FAULT>, FAULT PROCEDURE NAME

FAULT PROCEDURE NAME ::= NAME

INTERPRETER ATTRIBUTES ::= <DESCRIPTOR>/
                           <PROGRAM, LEXIC LINK, PARAMETER
                           BIT, FUNCTION>/
                           <DATA>, FORMAT SELECTOR /
                           <LOCK>, FORMAT SELECTOR

FORMAT SELECTOR ::= <INDEX USED TO SELECT FORMAT FOR:
                    ARITHMETIC, LOGICAL, OR CHARACTER>/
                    <NULL FORMAT>

STRUCTURE EXPRESSION ::= <ADDRESS-FIELD-LENGTH>, CONTAINER,
                         EXPRESSION, <END>

CONTAINER ::= <ALLOCATE>, CONTAINER INSTRUCTION

CONTAINER-INSTRUCTION ::= <SEGMENT (NUMBER)>/ <CALL (NAME)>

EXPRESSION ::= INSTRUCTION /
               INSTRUCTION, EXPRESSION

INSTRUCTION ::= <FIELD (A,L)> /
                <VARIABLE FIELD> /
                <VECTOR (A,L)>/
                <LIST (A,P,Q)>/
                <QUEUE (AI,AO,L)> /
                <VARIABLE QUEUE (AI,AO,L)> /
                <STACK (A,L)> /
                <PUSHDOWN (A,L)> /
                <STACK-VECTOR (A,L)> /
                <CALL(NAME)>
```

NAMES

NAMES IN THE B8500 SYSTEM ARE SHORTHAND DESCRIPTIONS USED TO ACCESS ELEMENTS FROM STRUCTURES WHICH ARE KNOWN TO THE HARDWARE. NAMES ARE FOUND EMBEDDED IN PROGRAM AND DESCRIPTION. THE B8500 RECOGNIZES SIX (6) KINDS OF NAMES. THE SYNTAX FOR NAME IS:

```
NAME ::= <COROUTINE RELATIVE>, STACK NUMBER, LL, D/
         <DISPLAY RELATIVE>, LL,D/
         <SLICE RELATIVE>, D/
         <COROUTINE N-BASE RELATIVE>, STACK NUMBER, D/
         <N-BASE RELATIVE>, D/
         <PROGRAM RELATIVE>, DP

STACK NUMBER ::= <INDEX INTO THE COROUTINE STACK>

LL ::= <INDEX INTO THE DISPLAY VECTOR>

D ::= <DISPLACEMENT FROM BASE>

DP ::= <A DISPLACEMENT FROM BASE OF PROGRAM SEGMENT>
```

A <COROUTINE RELATIVE> NAME ALLOWS ACCESSING OF AN ELEMENT FROM AN N-STACK BY:

A. SELECT DISPLAY ← COROUTINE DISPLAY[STACK NUMBER].

B. SELECT SLICE ← DISPLAY[LL].

C. SELECT ELEMENT ← SLICE[D].

A <DISPLAY RELATIVE> NAME ALLOWS ACCESSING OF AN ELEMENT FROM THE CURRENT N-STACK BY:

A. SELECT SLICE ← CURRENT DISPLAY [LL]

B. SELECT ELEMENT ← SLICE [D]

A <SLICE RELATIVE> NAME ALLOWS ACCESSING OF AN ELEMENT FROM THE CURRENT SLICE CONTAINED IN THE CURRENT N-STACK BY:

A. SELECT ELEMENT ← CURRENT SLICE [D].

A <COROUTINE N-BASE> NAME ALLOWS ACCESSING OF AN ELEMENT FROM AN N-STACK WITHOUT USING THE DISPLAY MECHANISM. THE ELEMENT IS ACCESSED BY:

A. SELECT N-STACK ← COROUTINE DISPLAY [STACK NUMBER]

B. SELECT ELEMENT ← N-STACK [D]

A <N-BASE RELATIVE> NAME ALLOWS ACCESSING OF AN ELEMENT FROM THE CURRENT N-STACK BY

A. SELECT ELEMENT ← CURRENT N-STACK [D]

A <PROGRAM RELATIVE> NAME ALLOWS ACCESSING OF AN ELEMENT FROM THE CURRENT PROGRAM STRING BY:

A. SELECT ELEMENT ← CURRENT PCR [DP]

THE RESOLUTION OF DP IS THE SYLLABLE. THE LENGTH OF AN ELEMENT IS THE SAME AS FOR THE N-STACK.

THE RESOLUTION OF D IS TO THE ELEMENT OF THE N-STACK.

DESCRIPTION EVALUATION -- EVALUATE OPERATION

ACCESSING STRUCTURED INFORMATION INVOLVES EVALUATION OF DESCRIPTIONS BY USE OF THE EVALUATE OPERATION PERFORMED BY THE KERNEL. THE PRODUCT OF DESCRIPTION EVALUATION IS A REFERENCE, CALLED THE TERMINAL DESCRIPTOR, ON THE ATTRIBUTE STACK. THE PARTICULAR ELEMENT REFERENCED IN THE STRUCTURE DEPENDS UPON THE MODE OF EVALUATION OF THE DESCRIPTION AND THE PARAMETERS SUPPLIED. THE EVALUATION MODES ENTER, REMOVE, AND CONSTRUCT MAY BE APPLIED TO ALL STRUCTURES. FOR LIST STRUCTURES TWO ADDITIONAL MODES, RESET AND SEQUENCE, ARE AVAILABLE. EVALUATION BEGINS WITH THE EXECUTION OF AN EVALUATE OPERATION WHICH USES AN EMPTY TERMINAL DESCRIPTOR AND A DESCRIPTOR TO BE SCANNED BY THE KERNEL.

ACCESS ATTRIBUTES

THE EVALUATE OPERATOR PERFORMS THESE STEPS. EACH STRUCTURE MAY DEFINE A FAULT PROCEDURE DETERMINED DURING EVALUATION BY THE RULE THAT IF THE FAULT PROCEDURE NAME IS DEFINED IN THE DESCRIPTOR BEING SCANNED THE NAME IN THE SCANNED DESCRIPTOR IS MOVED TO THE TERMINAL DESCRIPTOR. THE FAULT INDICATORS ARE ACCORDINGLY ACCUMULATED INTO THE TERMINAL DESCRIPTOR.

STRUCTURE EXPRESSION

THE STRUCTURE EXPRESSION CONSISTS OF AN ALLOCATE BIT FOLLOWED BY A

SEQUENCE OF STRUCTURE INSTRUCTIONS. IF THE ALLOCATE BIT IS FALSE AN IMMEDIATE ALLOCATE FAULT OCCURS. OTHERWISE, THE STRUCTURE EXPRESSION INSTRUCTIONS ARE EXECUTED FROM LEFT TO RIGHT. EACH INSTRUCTION CONSISTS OF AN OPERATION AND A STRUCTURE STATE.

THE STRUCTURE STATE CONTAINS ADDRESS AND LENGTH FIELDS. THE LENGTH OF THE FIELDS IN THE STRUCTURE STATE IS SPECIFIED BY THE ADDRESS-FIELD-LENGTH OF THE STRUCTURE EXPRESSION. THE FIRST INSTRUCTION OF A STRUCTURE EXPRESSION MUST DEFINE A SEGMENT NUMBER. THIS MAY BE DEFINED EITHER EXPLICITLY WITH A SEGMENT INSTRUCTION OR WITH A CALL INSTRUCTION OF ANOTHER STRUCTURE WHICH DEFINES THE SEGMENT NUMBER. THE SEGMENT NUMBER IS INSERTED IN A SEGMENT INSTRUCTION OF THE TERMINAL DESCRIPTOR.

STRUCTURES IN WHICH ALLOCATION MAY OCCUR ARE GOVERNED BY MODE-DEPENDENT INSTRUCTIONS. ACCESSES TO MODE-DEPENDENT STRUCTURES IN REMOVE OR ENTER MODE WILL CHANGE THE STRUCTURE STATE FOR ALLOCATION OR DEALLOCATION OF AN ELEMENT RESPECTIVELY.

ACCESSES TO ANY STRUCTURE IN CONSTRUCT MODE HAVE NO EFFECT ON STRUCTURE STATE. IN THE CASE OF MODE-INDEPENDENT STRUCTURES, ENTER AND REMOVE MODES ARE EQUIVALENT TO CONSTRUCT MODE. IN STRUCTURES WITH MORE THAN ONE MODE-DEPENDENT INSTRUCTION, MODE HAS EFFECT ONLY ON THE FIRST MODE-DEPENDENT INSTRUCTION; THAT IS, IF A STRUCTURE HAS SUBSTRUCTURES IN WHICH ALLOCATION MAY OCCUR, ALLOCATION CAN OCCUR ONLY IN THE FIRST ALLOCATABLE STRUCTURE.

EACH INSTRUCTION AFTER THE INITIAL ONE IN A STRUCTURE EXPRESSION OPERATES ON AN AC AND LC TO DEFINE A PROPER SUBSTRUCTURE WITHIN THE CONTAINER BY A SPECIFIC RULE. THE OUT-OF-BOUNDS FAULT OCCURS IF THE SUBFIELD IS NOT WHOLLY CONTAINED BY THE CONTAINER. UNLESS OTHERWISE SPECIFIED, PARAMETERS REQUIRED BY CERTAIN INSTRUCTIONS ARE FOUND ON THE VALUE STACK (VSS), WHICH IS DESCRIBED LATER.

STRUCTURE EXPRESSION OPERATORS

A DESCRIPTION OF EACH INSTRUCTION FOLLOWS (DETAILS CAN BE FOUND IN APPENDIX A):

THE FIELD INSTRUCTION DEFINES A SUBFIELD WITHIN THE CONTAINER BY SPECIFICATION OF THE ADDRESS, A, OF THE INITIAL BIT AND THE LENGTH, L, OF THE SUBFIELD. THE VARIABLE-FIELD INSTRUCTION SIMILARLY DEFINES A SUBFIELD BUT THE ADDRESS AND LENGTH ARE PARAMETRIC.

THE VECTOR INSTRUCTION DEFINES A FIELD WHICH IS A SINGLE ELEMENT OF A SET OF CONTIGUOUS, EQUAL-SIZE ELEMENTS BY MEANS OF A PARAMETRIC SUBSCRIPT, THE ADDRESS, A, OF THE FIRST ELEMENT, AND THE ELEMENTAL LENGTH, L.

THE STACK INSTRUCTION AND PUSHDOWN INSTRUCTION ARE USED FOR LAST-

IN-FIRST-OUT CONTIGUOUS STRUCTURES WITH FIXED-SIZE AND VARIABLE-SIZE ELEMENTS, RESPECTIVELY. EITHER INSTRUCTION DEFINES:

A. IN CONSTRUCT MODE, A REFERENCE TO THE TOP ELEMENT OF THE STRUCTURE WITH ADDRESS COUPLE (A,L).

B. IN REMOVE MODE, A REFERENCE TO AN ELEMENT DEFINED BY (A-L,L).

C. IN ENTER MODE, A REFERENCE TO AN ELEMENT DEFINED BY (A+L,L).

ELEMENT LENGTH IS PARAMETRIC FOR ACCESSES INTO PUSHDOWN STRUCTURES IN ENTER MODE. ACCESS TO STACK AND PUSHDOWN STRUCTURES CAUSE FAULTS ON OVERFLOW OR UNDERFLOW CONDITIONS.

THE QUEUE AND VARIABLE-LENGTH QUEUE INSTRUCTIONS ARE USED FOR FIRST-IN-FIRST-OUT CONTIGUOUS STRUCTURES WITH FIXED-SIZE AND VARIABLE-SIZED ELEMENTS, RESPECTIVELY. ACCESS TO THE FIRST OR LAST ELEMENT OF EITHER STRUCTURE IS MADE USING REMOVE MODE OR ENTER MODE, RESPECTIVELY. ELEMENT LENGTH IS PARAMETRIC FOR ACCESSES INTO VARIABLE-LENGTH QUEUE STRUCTURES IN ENTER MODE. ACCESSES TO BOTH TYPES OF QUEUE STRUCTURES CAUSE FAULTS ON QUEUE FULL OR EMPTY CONDITIONS.

THE LIST INSTRUCTION DEFINES REFERENCES TO ELEMENTS OF A LINKED LIST OF EQUAL-SIZE ELEMENTS. ALLOCATION AND DEALLOCATION ARE ACHIEVED BY USE OF A FREE LIST WHICH RESIDES WITH THE LIST STRUCTURE IN A CONTAINING FIELD. AN ACCESS TO THE LIST STRUCTURE IN CONSTRUCT MODE RESULTS IN A REFERENCE TO THE LIST ELEMENT. A HEADER POINTER, A, CURRENT ELEMENT POINTER, P, AND A POINTER, Q, TO THE ELEMENT WHICH PRECEDES THE CURRENT ELEMENT ARE PART OF THE LIST STRUCTURE STATE.

AN ENTER MODE ACCESS CAUSES AN ELEMENT, DESCRIBED BY A REFERENCE ON NSS, TO BE LINKED INTO THE LIST BETWEEN THE P AND Q POINTERS. THE P POINTER IS SET SO AS TO ADDRESS THIS NEW ELEMENT. SIMILARLY, A REMOVE MODE ACCESS DELINKS THE ELEMENT BETWEEN THE P AND Q POINTERS AND P IS ADVANCED TO THE SUCCESSOR TO THE REMOVED ELEMENT. THE P AND Q POINTERS MAY ALSO BE CONTROLLED BY THE LIST SEQUENCE AND RESET OPERATIONS. SEQUENCE ADVANCES P AND Q TOWARD THE TAIL OF THE LIST. RESET SETS P BACK TO THE HEAD POSITION, A. FAULTS OCCUR WHEN ATTEMPTING TO REMOVE OR SEQUENCE ON A NULL LIST OR A LIST IN WHICH P DOES NOT ADDRESS A LIST ELEMENT.

THE COMPOSITE STRUCTURES, PUSHDOWN-STACK AND PUSHDOWN-PUSHDOWN, ARE STRUCTURES SIMILAR TO CERTAIN HARDWARE STRUCTURES OF THE CENTRAL PROCESSOR AND DEAL WITH A COARSE STRUCTURE OVER A FINE STRUCTURE. THE PURPOSE OF PRESENTING THESE COMPOSITE STRUCTURES HERE IS TO MAKE FORMAL THEIR USE IN THE HARDWARE. THUS, THEY ARE NOT STRUCTURE INSTRUCTIONS USABLE IN DESCRIPTIONS BY SOFTWARE, BUT CERTAIN OPERATIONS CAN BE BETTER UNDERSTOOD BY FAMILIARITY WITH THESE STRUCTURES.

COMPOSITE STRUCTURES HAVE PROPERTIES SIMILAR TO THOSE OF COSPATIAL STRUCTURES IN THAT BOTH CONSIST OF MULTIPLE STRUCTURES OVER A SINGLE CONTAINER FIELD. HOWEVER, COMPOSITE STRUCTURE ACCESSES HAVE ADDITIONAL PROVISIONS FOR UPDATING THE STRUCTURE STATE OF ALL OF THE STRUCTURES IN THE COMPOSITE STRUCTURE. COMPOSITE STRUCTURES ARE ALSO ACCESSIBLE BY THEIR COARSE, FINE OR COMPOSITE NAMES. FOR EXAMPLE, WITH A SINGLE ACCESS TO A PUSHDOWN-STACK, A CHANGE IN THE STATES OF BOTH THE PUSHDOWN AND THE STACK, OR JUST THE PUSHDOWN, OR JUST THE STACK, CAN BE EFFECTED BY SELECTION OF THE APPROPRIATE NAME WITHIN THE STRUCTURE. IN PARTICULAR, A REMOVE MODE ACCESS WITH A PUSHDOWN-STACK INSTRUCTION RESULTS IN A SINGLE OPERATION EQUIVALENT TO A REMOVE OF THE PUSHDOWN ELEMENT AND A NUMBER OF REMOVES TO THE STACK STRUCTURE EQUAL TO THE NUMBER OF FINE ELEMENTS CONTAINED BY THE PUSHDOWN ELEMENT.

THE PUSHDOWN-PUSHDOWN INSTRUCTION PROVIDES ACTIONS SIMILAR TO THE PUSHDOWN-STACK BUT WITH THE FINE STRUCTURE BEING A PUSHDOWN INSTEAD OF A STACK.

THE STACK-VECTOR INSTRUCTION PROVIDES INDEXING OPERATIONS INTO A VECTOR OF STACK ELEMENTS. VECTOR ACCESSES IN A STACK-VECTOR STRUCTURE ARE BOUNDED BY THE STACK. THE STACK ELEMENT AND VECTOR ELEMENT ARE OF EQUAL SIZE, IN CONTRAST TO THE FINE-COARSE SIZES OF THE OTHER COMPOSITES.

THE CALL INSTRUCTION EVALUATES A DESCRIPTION BY SPECIFYING THE NAME OF THE DESCRIPTION TO BE EVALUATED. A RETURN OCCURS WHEN EVALUATION OF THE CALLED DESCRIPTION IS COMPLETE.

AT THE COMPLETION OF THE STRUCTURE EXPRESSION, THE INTERPRETER ATTRIBUTES ARE COPIED INTO THE TERMINAL DESCRIPTION. THIS COMPLETES THE EVALUATE OPERATION.

INTERPRETER ATTRIBUTES

THE INTERPRETER ATTRIBUTES ARE EXAMINED. IF THE INTERPRETER ATTRIBUTE IS "DESCRIPTION", EVALUATION THEN CONTINUES WITH AN EVALUATE OPERATION ON THE DESCRIPTION LOCATED BY THE TERMINAL DESCRIPTION. IF THE INTERPRETER ATTRIBUTE IS NOT DESCRIPTION THEN THE TOP OF THE ATTRIBUTE STACK IS MOVED TO NSS AND EVALUATION IS COMPLETE WITH THIS STEP.

TERMINAL DESCRIPTION APPLICATIONS

FURTHER USE OF THE REFERENCE DEVELOPED BY THE EVALUATION MECHANISM MAY BE MADE BY THE OPERATIONS VALUE, NAME, STORE, OR EXECUTE, WHICH ARE SPECIFIED IN THE PARAGRAPH IN THIS SECTION ENTITLED "DESCRIPTION OF OPERATORS".

IT SHOULD BE NOTED THAT THE LOCK INTERPRETER ATTRIBUTE IS TREATED AS A DATA INTERPRETER ATTRIBUTE WITH THE DEFINED DATA FIELD CONTAINING A ONE-BIT FLAG IN THE MOST SIGNIFICANT (OR ONLY) BIT POSITION OF THE FIELD.

PROCESSOR STRUCTURE DESCRIPTION

FUNCTIONAL DESCRIPTION

THE FOLLOWING IS A FUNCTIONAL DESCRIPTION OF THE STRUCTURE OF THE B8500 PROCESSOR MODULE AND USES ONLY THOSE STRUCTURES WHICH THE KERNEL CAN EVALUATE. THIS PERMITS THE PROCESSOR STRUCTURE TO BE DEFINED AS A STRUCTURE RESIDING IN LEVEL-1 STORAGE. THIS, IN ESSENCE, GUARANTEES THAT THE AMOUNT OF LOCAL BUFFERING USED IN THE PROCESSOR WILL NOT INFLUENCE THE FUNCTIONAL OPERATION OF THE MACHINE.

LOCAL BUFFERING MUST BE APPLIED AS ASSOCIATIVE MEMORY OVER THE LEVEL-1 STRUCTURE. IN THIS WAY THE BUFFERING IS ONLY A PARAMETER WHICH MAY BE VARIED TO CONTROL PROCESSOR SPEED. AS AN EXAMPLE, A SERIAL PROCESSOR WILL HAVE NO BUFFERING WHEREAS THE B8500 WILL MAKE EXTENSIVE USE OF LOCAL BUFFERING. IT CAN BE SEEN THAT BY USING A SINGLE FUNCTIONAL DESCRIPTION IT IS POSSIBLE TO DESCRIBE A SERIES OF MACHINES DIFFERING ONLY IN SPEED.

THE BASIC PROCESSOR STRUCTURES ARE:

A. THE RESOURCE CONTROL STRUCTURE.

B. THE PROCEDURE CONTROL STRUCTURE.

C. THE COROUTINE CONTROL STRUCTURE.

D. THE PROGRAM CONTROL STRUCTURE.

THESE STRUCTURES PROVIDE ALL THE MECHANISMS REQUIRED TO MANAGE:

A. "N" LEVELS OF STORAGE.

B. ALLOCATION OF PROCESSORS.

C. THE INTERNAL CONTROL OF COROUTINE AND PROCEDURE ENTRY AND RETURN

IT IS NECESSARY THAT HARDWARE BE PROVIDED TO PROTECT THE SYSTEM AGAINST A FAULTY PROCESS. THIS PROTECTION IS ACHIEVED BY USING THE RESOURCE STRUCTURE. THE RESOURCE STACK (RSS) PREVENTS ACCESS TO RESOURCES WHICH HAVE NOT BEEN GIVEN EXPLICITLY TO A PROCESS.

DECLARATION SYNTAX

A FAIRLY SIMPLE SYNTAX IS USED TO DECLARE THE STRUCTURE OF THE PROCESSOR.

```
<DECLARATION> ::= DECLARE <DECL-LIST>;

<DECL-LIST> ::= <DECL> /
                <DECL-LIST> , <DECL>

<DECL> ::= <STRUCTURE-DECL> <TYPICAL-ELEMENT>/
           <DECL> AND <STRUCTURE-DECL>
           <TYPICAL ELEMENT>

<STRUCTURE-DECL> ::= <TYPE> <NUMBER-OF-ELEMENTS> <NAME>/
                     COMPOSITE <NUMBER-OF-ELEMENTS>
                     <NAME> OF

<TYPE> ::= <IDENTIFIER>

<NAME> ::= <EMPTY> /
           NAMED <IDENTIFIER>

<NUMBER-OF-ELEMENTS> ::= <EMPTY>
                         OF <IDENTIFIER> ELEMENTS /
                         TEMPLATE

<TYPICAL-ELEMENT> ::= <SIZE> <FORMAT>

<SIZE> ::= WITH ELEMENT SIZE <IDENTIFIER> /
           VARIABLE /
           <EMPTY>

<FORMAT> ::= USING FORMAT <KIND> /
             <EMPTY>

<KIND> ::= BINARY SIGNED INTEGER /
           BINARY UNSIGNED INTEGER /
           DESCRIPTION /
           LOGICAL
```

```
<IDENTIFIER> ::= LETTER /
                 <IDENTIFIER> LETTER /
                 <IDENTIFIER> DIGIT

<EMPTY> ::= (DEVOID OF CONTENT)
```

THE RESOURCE CONTROL STRUCTURE.

THE B8500 SYSTEM IS VIEWED AS A SET OF RESOURCES AVAILABLE TO A NUMBER OF COMPETING PROCESSES. THE MANAGEMENT AND ALLOCATION OF THESE RESOURCES IS DISTRIBUTED OVER A SET OF CONTROL PROCESSES. EACH CONTROL PROCESS MANAGES SOME SUBSET OF PROCESSES. THE DISTRIBUTION OF RESOURCES TO PROCESSES WHICH ARE CREATED AND CONTROLLED BY A PARTICULAR PROCESS IS THROUGH THE RESOURCE CONTROL STRUCTURE (R).

ONE AND ONLY ONE R EXISTS FOR EACH PROCESSOR IN THE SYSTEM. AS THE PROCESSOR MOVES FROM PROCESS SPACE TO PROCESS SPACE, THE STRUCTURE KEEPS A HISTORY OF THE RESOURCES BEING PASSED. AS A PROCESS IS CALLED, THE SUBSET OF THE RESOURCES THE CALLER WISHES TO PASS ARE PLACED IN THE RESOURCE STRUCTURE FOR USE BY THE CALLED PROCESS. THE CALLED PROCESS MAY USE THESE RESOURCES BUT MAY NOT CHANGE THEM. WHEN A PROCESS RETURNS TO THE PROCESS THAT ACTIVATED IT, THE RESOURCES THAT HAD BEEN ALLOCATED FOR THAT PROCESS ARE REMOVED FROM THE RESOURCE STRUCTURE.

A NUMBER OF DIFFERENT RESOURCES ARE DESCRIBED BY ENTRIES IN THE R. TYPICAL ENTRIES ARE:

A. DESCRIPTIONS OF SEGMENT CONTAINERS IN LEVEL-1.

B. DESCRIPTIONS OF SEGMENT CONTAINERS IN LEVEL-2.

C. DESCRIPTIONS OF DEVICES IN LEVEL-3.

D. DESCRIPTION OF THE PROCESSOR TIME.

E. DESCRIPTION OF THE FAULT MASKS.

F. DESCRIPTION OF THE FAULT AND INTERUPT REGISTERS.

THE RESOURCE STRUCTURE PROVIDES PROTECTION AGAINST THE ILLEGAL USE OF RESOURCES BY A PROCESS AND THE CHANGING OF RESOURCES WHICH DO NOT BELONG TO A GIVEN PROCESS. THIS IS ACCOMPLISHED BY HAVING THE R OUTSIDE OF THE ADDRESSING SPACE OF ALL PROCESSES EXCEPT IMP. THE HARDWARE CAN ACCESS THE RESOURCE STRUCTURE FOR FINAL COMBINE OPERATIONS.

RESOURCE STRUCTURE

BRACKETS ARE USED IN THIS SYNTAX AND OTHERS IN THE PROCESSOR STRUCTURE SECTION TO INDICATE DEFINITIONS TO FOLLOW.

RESOURCE STRUCTURE DECLARATION:

```
DECLARE  COMPOSITE NAMED <R> OF PUSHDOWN NAMED <RS>
              AND STACK OF <N11> ELEMENTS
              NAMED <RSS> WITH ELEMENT SIZE <N2> USING
              FORMAT DESCRIPTION,

         COMPOSITE OF STACK NAMED <RL>
         WITH <N12> ELEMENTS AND VECTOR NAMED
         <RR> WITH ELEMENT SIZE <N2> USING
         FORMAT DESCRIPTION,

         VECTOR TEMPLATE NAMED <TR> WITH ELEMENT
              SIZE <N2> USING FORMAT DESCRIPTION,

         FIELD NAMED <IMP MODE> WITH SIZE
              1 USING FORMAT LOGICAL;
```

IDENTIFIER DEFINITION:

R	IS	THE RESOURCE STRUCTURE.
RS	IS	THE RESOURCE SLICE.
RSS	IS	THE TOP RESOURCE STACK CONTAINER.
RR	IS	THE RESOURCE STACK DISPLAY.
RL	IS	THE CURRENT RESOURCE LEVEL.
TR	IS	A VECTOR TEMPLATE USED TO CONSTRUCT A VECTOR DESCRIPTION WHICH IS PLACED IN RL.
IMP MODE	IS	A FLAG USED TO INDICATE THAT THE PROCESSOR IS IN THE INTERPETER MANAGEMENT PROCESS.

THE IDENTIFIER N1,N2,N3... STANDS FOR A PARAMETER TO BE DEFINED BY IMPLEMENTATION.

THE PROCEDURE CONTROL STRUCTURE

A NUMBER OF HIGHER-LEVEL LANGUAGES CAN MAKE EFFECTIVE USE OF A STRUCTURE FOR CONTROLLING THE ALLOCATION OF LEVEL-1 STORAGE FOR PASSING PARAMETERS TO PROCEDURES AND FUNCTIONS AND FOR ALLOCATING STORAGE FOR LOCAL VARIABLES USED WITHIN PROCEDURES, FUNCTIONS, AND BLOCKS. THE PROCEDURE CONTROL STRUCTURE PROVIDES THIS MECHANISM.

THE STRUCTURE CONSISTS OF A STACK FOR STORING DESCRIPTIONS OF THE DATA STRUCTURES USED BY A PROGRAM, AND OF A DISPLAY FOR CONTROLLING THE PARTICULAR DESCRIPTIONS WHICH ARE CURRENTLY VISIBLE TO THE PROGRAM BY LEXIC LEVEL NAMES. DESCRIPTIONS MAY ALSO BE ADDRESSED BY N-BASE RELATIVE NAMES. IN ADDITION, A VARIABLE-WIDTH STACK (PUSH-DOWN) IS PROVIDED FOR STORING VALUES OF VARIOUS TYPES AND FORMATS.

THE VSS IS STRONGLY CONNECTED TO THE NSS IN THAT ALL ENTRIES IN THE VSS ARE DESCRIBED IN THE NSS. THE VSS AND NSS ARE ALSO USED AS THE ARITHMETIC EXPRESSION EVALUATION STACK; THAT IS, THE TOPS OF THESE STACKS ARE USED BY THE AU FOR SOURCES OF OPERANDS AND AS A DESTINATION FOR RESULTS.

WHENEVER PARAMETERS OR LOCALS ARE TO BE ENTERED IN THE PROCEDURE STRUCTURE (PROCEDURE OR FUNCTION CALL AND BLOCK ENTRY), A SLICE DESCRIPTION MUST BE CREATED AND ENTERED INTO THE DISPLAY. THE SLICE CONSISTS OF THE DESCRIPTIONS THAT CURRENTLY EXIST IN THE ARITHMETIC EXPRESSION STACK PORTION OF THE NAME STACK. A VECTOR TEMPLATE IS BOUND TO A FIELD DESCRIPTION OF THE SPACE WHICH ENCOMPASSES THE PORTION OF THE NSS TO BE SLICED. THE RESULTING VECTOR DESCRIPTION IS THEN PLACED INTO THE DISPLAY.

PROCEDURE STRUCTURE DECLARATION:

```
DECLARE  COMPOSITE NAMED <N> OF PUSHDOWN
              NAMED <NS> AND STACK OF <N13> ELEMENTS
              NAMED <NSS> WITH ELEMENT SIZE <N2> USING
              FORMAT DESCRIPTION,

         COMPOSITE NAMED <V> OF PUSHDOWN
         NAMED <VS> AND PUSHDOWN OF <N14> ELEMENTS
         NAMED <VSS> WITH ELEMENT SIZE VARIABLE
         USING FORMAT DATA OR LOCK,

         VECTOR OF <N3> ELEMENTS NAMED <D> WITH
         ELEMENT SIZE <N5> USING FORMAT DESCRIPTION,

         VECTOR TEMPLATE NAMED <TN> WITH
         ELEMENT SIZE <N2> USING FORMAT DESCRIPTION;
```

IDENTIFIER DEFINITION;

N	IS	THE NAME STRUCTURE.
NS	IS	THE SLICE STRUCTURE.
NSS	IS	THE NAME STACK.
V	IS	THE VALUE STRUCTURE.
VS	IS	THE VALUE SLICE STRUCTURE.
VSS	IS	THE VALUE STACK.
D	IS	THE PROCEDURE DISPLAY.
TN	IS	A VECTOR TEMPLATE USED TO CONSTRUCT DESCRIPTIONS FOR THE DISPLAY D.

THE IDENTIFIER N1,N2,N3... STANDS FOR A PARAMETER TO BE DEFINED BY IMPLEMENTATION.

THE COROUTINE CONTROL STRUCTURE

PROCESSES REPRESENT ROUTINES WHICH EXIST CONCURRENTLY AND MAY BE EXECUTED ASYNCHRONOUSLY. COROUTINES REPRESENT ROUTINES WHICH EXIST CONCURRENTLY BUT ARE EXECUTED SYNCHRONOUSLY. A PROCESS MAY BE SWITCHED BETWEEN SEVERAL LINES OF CONTROL WITHIN A PROCESS BY USE OF THE COROUTINE STRUCTURE. EACH CONCURRENT ROUTINE IS DEFINED BY A PROCEDURE CONTROL STRUCTURE (NSS, VSS, AND D) AND A PROGRAM CONTROL STRUCTURE (P) WHICH IS DEFINED, BY NAME, IN THE NSS OF THE ROOT ROUTINE.

THE COROUTINE CONTROL STRUCTURE IS A STACK WHICH CONTAINS REFERENCES TO THE DISPLAY AND NAME OF EACH OF THE ROUTINES WHICH HAVE ACTIVATED A COROUTINE. THE TOP OF THE STACK IS THE ROOT OF THE CURRENT SET OF CONCURRENT ROUTINES WHICH ARE ACTIVE.

THE STATE OF THE ROUTINE CURRENTLY BEING EXECUTED IS HELD IN THE COROUTINE CONTROL FIELD.

COROUTINE STRUCTURE DECLARATION:

```
DECLARE  COMPOSITE OF STACK NAMED <C> WITH <N6> ELEMENTS
         AND VECTOR NAMED <CD> WITH ELEMENT
         FIELD NAMED <CPD> WITH SIZE <N2> USING FORMAT
              DESCRIPTION
         AND FIELD NAMED <CLL> WITH SIZE <N7> USING FORMAT
               BINARY
         UNSIGNED INTEGER AND FIELD NAMED <CDP> WITH SIZE
         <N8> USING FORMAT BINARY UNSIGNED INTEGER,

         FIELD NAMED <CCF> WITH
              FIELD NAMED <ND> WITH SIZE <N2> USING FORMAT
              DESCRIPTION
         AND FIELD NAMED <VD> WITH SIZE <N2> USING FORMAT
              DESCRIPTION
         AND FIELD NAMED <DD> WITH SIZE <N2> USING FORMAT
              DESCRIPTION
         AND FIELD NAMED <PD> WITH SIZE <N2> USING FORMAT
              DESCRIPTION;
```

IDENTIFIER DEFINITION:

C	IS	THE COROUTINE STRUCTURE.
CD	IS	THE COROUTINE DISPLAY.
CPD	IS	REFERENCE TO THE PROCEDURE DISPLAY OF THE ROOT ROUTINE.
CDP	IS	DISPLACEMENT FROM LL SPECIFIED IN CLL.
CLL	IS	THE LEXIC LEVEL WHERE THE DESCRIPTION OF THE CURRENTLY ACTIVE COROUTINE IS FOUND.
CCF	IS	THE CURRENTLY ACTIVE ROUTINES STATE.
ND	IS	DESCRIPTION OF THE N STRUCTURE.
VD	IS	DESCRIPTION OF THE V STRUCTURE.
DD	IS	DESCRIPTION OF THE D STRUCTURE.
PD	IS	DESCRIPTION OF THE P STRUCTURE.

THE IDENTIFIER N1,N2,N3... STANDS FOR A PARAMETER TO BE DEFINED BY IMPLEMENTATION.

THE PROGRAM CONTROL STRUCTURE

THE PROGRAM CONTROL STRUCTURE IS USED TO RETAIN THE HISTORY OF SUBROUTINE, PROCEDURE, FUNCTIONS, AND LOOP CALLS. WHENEVER A CALL IS MADE ON A NEW ROUTINE, THE DESCRIPTION OF THE ROUTINE IS PUSHED INTO P. UPON RETURN, P IS POPPED, AND THE DESCRIPTION OF THE ROUTINE BEING EXITED IS REMOVED. A BRANCH REPLACES THE CURRENT TOP DESCRIPTION IN THE STACK.

THE SIDE EFFECTS ON THE PROCEDURE STRUCTURE ARE CONTROLLED BY THE PARAMETER AND FUNCTION FLAGS CONTAINED IN THE ROUTINES DESCRIPTION.

PROGRAM STRUCTURE DECLARATION:

```
DECLARE  STACK OF <N9> ELEMENTS NAMED <P> WITH ELEMENT
         FIELD NAMED <CONTAINER-ADDRESS> WITH SIZE <N10>
         USING FORMAT BINARY SIGNED INTEGER AND
         FIELD NAMED <CONTAINER-LENGTH> WITH SIZE <N10>
         USING FORMAT BINARY SIGNED INTEGER AND
         FIELD NAMED <SEQUENCER> WITH SIZE <N11> USING
         FORMAT BINARY SIGNED INTEGER AND
         FIELD NAMED <LENGTH> WITH SIZE <N12> USING
         FORMAT BINARY SIGNED INTEGER AND
         FIELD NAMED <PARMS> WITH SIZE 1 USING
         FORMAT LOGICAL
         FIELD NAMED <FUNCT> WITH SIZE 1 USING
         FORMAT LOGICAL
         FIELD NAMED <CURRENT LL> WITH SIZE <N7> USING
         FORMAT BINARY INTEGER;
```

IDENTIFIER DEFINITION:

P	IS	THE PROGRAM CONTROL STRUCTURE.
CONTAINER-ADDRESS	IS	ADDRESS OF THE CONTAINER HOLDING THE PROGRAM STRING.
CONTAINER-LENGTH	IS	LENGTH OF THE CONTAINER HOLDING THE PROGRAM STRING.
SEQUENCER	IS	THE POINTER WHICH INDICATES THE CURRENT SYLLABLE BEING EXECUTED.
LENGTH	IS	THE SYLLABLE SIZE.

CURRENT LL	IS	THE LEXIC LEVEL WHICH IS CURRENTLY BEING EXECUTED.
PARMS	IS	A FLAG INDICATING WHETHER THE PROCEDURE HAS PARAMETERS OR NOT.
FUNCT	IS	A FLAG INDICATING WHETHER THE PROCEDURE RETURNS A VALUE.

THE IDENTIFIER N1,N2,N3,.. STANDS FOR A PARAMETER TO BE DEFINED BY IMPLEMENTATION.

DESCRIPTION OF OPERATORS

FOR CLARITY THE OPERATOR IS GIVEN FIRST AND ANY PARAMETERS THE OPERATOR MAY REQUIRE ARE GIVEN FOLLOWING THE OPERATOR.

FOR EXAMPLE:

CONSTRUCT (NAME)

A. CONSTRUCT IS AN OPERATOR.

B. (NAME) IS A PARAMETER OF THE OPERATOR CONSTRUCT.

CONSTRUCT (NAME)

CONSTRUCT FETCHES AND EVALUATES THE NAMED DESCRIPTION. EVALUATION IN CONSTRUCT MODE BUILDS A REFERENCE IN THE ATTRIBUTE STACK. THE EVALUATION SEQUENCE DEPENDS ON THE STRUCTURE OPERATORS GIVEN IN APPENDIX A. THE REFERENCE CONSTRUCTED BY THE EVALUATION SEQUENCE IN THE ATTRIBUTE STACK IS MOVED TO NSS.

ENTER (NAME)

ENTER FETCHES AND EVALUATES THE NAMED DESCRIPTION. EVALUATION IN ENTER MODE ALLOCATES A FIELD IN THE NAMED STRUCTURE, AND BUILDS A REFERENCE TO THAT FIELD. VARIABLE-LENGTH ELEMENT STRUCTURES REQUIRE THE LENGTH PARAMETER FOR THE FIELD TO BE ALLOCATED TO BE IN VSS PRIOR TO THE ENTER OPERATION. THE REFERENCE IN THE ATTRIBUTE STACK IS MOVED TO NSS.

REMOVE (NAME)

REMOVE FETCHES AND EVALUATES THE NAMED DESCRIPTION. EVALUATION IN REMOVE MODE DEALLOCATES A FIELD IN THE NAMED STRUCTURE, AND BUILDS A REFERENCE TO THAT DEALLOCATED FIELD. VARIABLE LENGTH STRUCTURES CONTAIN THE ELEMENT LENGTH INFORMATION NEEDED FOR DEALLOCATION. THE REFERENCE IN THE ATTRIBUTE STACK IS MOVED TO NSS.

NAME

NAME IS USED TO PRODUCE A REFERENCE TO A FIELD. THE FUNCTION OF THE NAME OPERATOR DEPENDS UPON THE INTERPRETER ATTRIBUTES OF THE DESCRIPTION IN NSS.

IF THE INTERPRETER ATTRIBUTE IS DATA OR LOCK, NO OPERATION TAKES PLACE.

IF THE INTERPRETER ATTRIBUTE IS PROGRAM, A FUNCTION CALL OCCURS. THE RESULT OF THE FUNCTION IS LEFT ON NSS AND A NAME OPERATION IS RE-EXECUTED. (SEE THE EXECUTE OPERATOR FOR FUNCTION CALL.)

IF THE INTERPRETER ATTRIBUTE IS DESCRIPTOR, THE ACCESS ATTRIBUTES ARE EXAMINED TO SEE IF THE DESCRIPTOR CAN BE FETCHED. IF IT CAN, THE NEW DESCRIPTOR IS EVALUATED; OTHERWISE, AN ACCESS FAULT IS GENERATED. A NAME OPERATION IS THEN PERFORMED ON THE RESULT OF THE EVALUATION.

VALUE

VALUE CAUSES A VALUE TO BE ENTERED INTO V. THE DESCRIPTION INTERPRETER ATTRIBUTES ARE EVALUATED TO DETERMINE SUBSEQUENT ACTION AS FOLLOWS:

IF THE INTERPRETER ATTRIBUTE IS DATA, THEN THE ACCESS ATTRIBUTES ARE EXAMINED TO SEE IF THE DATA CAN BE FETCHED. IF IT CAN BE FETCHED, THE REFERENCE GENERATED IS USED AS A PARAMETER TO MAKE AN ENTRY INTO V, CREATING AN ELEMENT VSS. THE FIELD DESCRIBED BY THE REFERENCE IS THEN MOVED TO VSS. THIS REFERENCE IS ALSO USED TO CREATE AN ENTRY IN NSS WHICH WILL DESCRIBE THE ELEMENT IN VSS. IF THE DATA CANNOT BE FETCHED, AN ACCESS FAULT IS GENERATED.

IF THE INTERPRETER ATTRIBUTE IS LOCK, THE ACTION IS SIMILAR TO DATA EXCEPT THAT A READ-WITH-LOCK CYCLE IS ENVOKED IN LEVEL-1 STORAGE, THAT IS, THE FIELD IS FETCHED FROM LEVEL-1 AND THE MOST SIGNIFICANT BIT SET TO ONE IN LEVEL-1 WITHOUT ANY INTERVENING STORAGE CYCLES.

IF THE INTERPRETER ATTRIBUTE IS PROGRAM, A FUNCTION CALL OCCURS. THE RESULT OF THE FUNCTION IS LEFT ON NSS AND A VALUE OPERATION IS RE-EXECUTED. (SEE THE EXECUTE OPERATOR FOR FUNCTION CALL.)

IF THE INTERPRETER ATTRIBUTE IS DESCRIPTION, THE ACCESS ATTRIBUTES ARE EXAMINED TO SEE IF THE DESCRIPTOR CAN BE FETCHED. IF IT CAN, THE DESCRIPTOR IS EVALUATED; OTHERWISE, AN ACCESS FAULT IS GENERATED. A VALUE OPERATION IS THEN PERFORMED ON THE RESULT OF THE EVALUATION.

STORE

THE STORE OPERATION IS UTILIZED TO MOVE A VALUE FROM VSS TO STORAGE OR STORAGE TO STORAGE. THE DESCRIPTION OF THE SOURCE (WHICH MAY BE VSS) AND THE DESTINATION FIELDS ARE ON NSS.

IF THE INTERPRETER ATTRIBUTES OF THE SOURCE AND DESTINATION DESCRIPTORS ARE NOT IDENTICAL AN ILLEGAL OPERATION FAULT OCCURS. THE ACCESS ATTRIBUTES ARE EXAMINED. IF READ FAULT IS SET IN THE SOURCE DESCRIPTOR OR WRITE FAULT IS SET IN THE DESTINATION DESCRIPTOR AN ACCESS FAULT OCCURS. IF NO FAULT OCCURS AND THE INTERPRETER ATTRIBUTES ARE DATA OR LOCK THE FORMAT AND LENGTH ATTRIBUTES OF THE SOURCE DESCRIPTOR AND DESTINATION DESCRIPTOR ARE COMPARED. IF NECESSARY, A TRANSFORMATION OPERATOR IS CALLED. SUBSEQUENT ACTION IS IDENTICAL FOR ALL CASES, NAMELY, THE SOURCE FIELD IS MOVED TO THE DESTINATION FIELD.

EXECUTE

THE EXECUTE OPERATOR IS USED TO CAUSE SUBROUTINE ENTRY. THE FUNCTION OF THE EXECUTE OPERATOR DEPENDS UPON THE INTERPRETER ATTRIBUTES OF THE DESCRIPTION IN NSS.

IF THE INTERPRETER ATTRIBUTE IS DATA OR LOCK, NO OPERATION TAKES PLACE.

IF THE INTERPRETER ATTRIBUTE IS PROGRAM A FUNCTION CALL OR A PROCEDURE CALL OCCURS. THE REFERENCE ON NSS IS MOVED TO P. IF THE PARAMETER BIT IS SET, A DISPLAY CONTROL WORD (DCW) IS PLACED ON NSS, A SLICE IS MADE IN N, THE CURRENT LEXIC LEVEL IS SET TO THE LEXIC LEVEL OF THE SUBROUTINE AND THE DISPLAY UPDATED.

THE DCW, INSERTED INTO NSS DURING FUNCTION CALL OR SUBROUTINE CALL, CONTAINS A FIELD FOR THE LEXIC LEVEL OF THE SLICE (SL), A LENGTH OF SLICE FIELD (L) AND A LINK TO THE DCW FOR THE SLICE IN WHICH THE SUBROUTINE OR FUNCTION WAS DECLARED. THIS LINK IS CARRIED IN THE LEXIC LINK PORTION OF THE PROGRAM DESCRIPTOR.

DISPLAY UPDATE CONSISTS OF SEQUENCING THROUGH A DCW LIST AND TESTING AGAINST THE DISPLAY ELEMENTS. IF THE DISPLAY ELEMENT, D [SL], DESCRIBES THE SLICE OF THE CURRENT DCW OR SL OF THE CURRENT DCW IS ZERO THE DISPLAY UPDATE IS COMPLETE; OTHERWISE, A VECTOR DESCRIPTOR FOR THE CURRENT SLICE IS STORED INTO THE CURRENT DISPLAY ELEMENT, D [SL]. THE UPDATE CONTINUES BY SEQUENCING TO THE NEXT DCW AND REPEATING THE ABOVE TEST UNTIL THE UPDATE IS COMPLETED.

IF THE INTERPRETER ATTRIBUTE IS DESCRIPTOR, THE ACCESS ATTRIBUTES ARE EXAMINED TO SEE IF THE DESCRIPTOR CAN BE FETCHED. IF IT CAN, THE NEW DESCRIPTOR IS EVALUATED; OTHERWISE, AN ACCESS FAULT IS GENERATED. AN EXECUTE OPERATION IS THEN PERFORMED ON THE RESULT OF THE EVALUATION.

LOAD

LOAD OPERATES ON THE DESCRIPTOR IN NSS. THE INTERPRETER ATTRIBUTES ARE EVALUATED IN ORDER TO DETERMINE SUBSEQUENT ACTION. IF THE INTERPRETER ATTRIBUTES ARE PROGRAM OR DATA, THE DESCRIPTOR IS LEFT IN NSS. IF THE INTERPRETER ATTRIBUTE IS A DESCRIPTOR, A REFERENCE IS BUILT BY EXECUTING THE DESCRIPTOR. THE CONTENTS OF THE FIELD SPECIFIED BY THE REFERENCE REPLACES THE GIVEN DESCRIPTION.

DESCRIBE (NAME)

DESCRIBE GENERATES A DESCRIPTION OF A DESCRIPTION. ACCESS ATTRIBUTES ARE SET TO THE CLEARED STATE. THE INTERPRETER ATTRIBUTES WILL BE SET TO DESCRIPTION. THE STRUCTURE EXPRESSION GENERATED WILL BE A FIELD WITH THE AE AND LE OF THE NAME PASSED TO THE DESCRIBE OPERATOR AS A PARAMETER FOLLOWED BY AN END. THE GENERATED DESCRIPTION IS LEFT IN NSS.

ALLOCATE (NAME)

ALLOCATE USES AN UNALLOCATED STRUCTURE IN NSS AND THE NAME OF AN ALLOCATED STRUCTURE. THE UNALLOCATED STRUCTURE IN NSS IS EVALUATED TO SEE IF THE FIRST STRUCTURE EXPRESSION IS A CALL OR A SEGMENT NUMBER, FIELD. IF IT IS NEITHER, AN ILLEGAL OPERATION FAULT IS SET. IF IT IS A CALL, THE NAME OF THE ALLOCATED STRUCTURE IS PASSED TO THE CALL. IF IT IS A SEGMENT-NUMBER FIELD, THE NAMED DESCRIPTOR OF THE ALLOCATED STRUCTURE IS EXECUTED IN ENTER MODE.

THE RESULTING REFERENCE IS PASSED TO THE FIELD EXPRESSION OF THE UNALLOCATED STRUCTURE. THE DESCRIPTION OF THE NEWLY ALLOCATED STRUCTURE IS LEFT ON NSS.

BIND (NAME)

THE BIND OPERATOR IS GIVEN THE NAME OF AN ALLOCATED STRUCTURE AND AN UNALLOCATED STRUCTURE IN NSS. THE DESCRIPTION OF THE UNALLOCATED STRUCTURE IS EVALUATED TO SEE IF THE FIRST STRUCTURE EXPRESSION IS A CALL OR A SEGMENT NUMBER, FIELD. IF IT IS NEITHER AN ILLEGAL OPERATION FAULT IS SET. IF IT IS A CALL THE NAME OF THE ALLOCATED STRUCTURE IS PASSED TO THE CALL. IF IT IS A SEGMENT-NUMBER-FIELD, THE NAMED DESCRIPTOR OF THE ALLOCATED STRUCTURE IS EXECUTED IN CONSTRUCT MODE.

THE RESULTING REFERENCE IS PASSED TO THE FIELD EXPRESSION OF THE UNALLOCATED STRUCTURE. THE DESCRIPTION OF THE NEWLY ALLOCATED STRUCTURE IS LEFT ON NSS.

SHORTEN (NAME)

THE SHORTEN OPERATOR FETCHES THE NAMED DESCRIPTOR. SHORTEN EXECUTES THE FIRST TERM OF THE STRUCTURE EXPRESSION THAT IS NOT SEGMENT NUMBER OR FIELD, AND REPLACES THAT TERM WITH THE DERIVED FIELD EXPRESSION. CONSECUTIVE FIELD EXPRESSIONS ARE COMBINED. THE OPERATOR YIELDS A NEW DESCRIPTOR IN NSS.

FINAL COMBINE

FINAL COMBINE USES THE SEGMENT NUMBER AS A DISPLACEMENT INTO RL TO FIND THE ABSOLUTE CONTAINER OF THE STRUCTURE. THIS ABSOLUTE CONTAINER IS USED TO CALCULATE THE ABSOLUTE ADDRESS OF THE STRUCTURE. THIS OPERATOR MAY BE EXPLICITLY CALLED ONLY IN PRIVILEGED MODE. IN NORMAL MODE, THIS OPERATOR IS CALLED IMPLICITLY EVERY TIME A LEVEL-1 OR LEVEL-2 FETCH OR STORE IS MADE.

SET ATTRIBUTES

THE ACCESS AND INTERPRETER ATTRIBUTES IN THE DESCRIPTION IN NSS ARE REPLACED BY THE VALUE FROM VSS.

STACK OPERATORS

DUPLICATE

THE DUPLICATE OPERATOR DUPLICATES THE CONTENTS OF NSS. IF THE CONTENTS OF NSS CONTAIN A REFERENCE TO VSS, THE CONTENTS OF VSS ARE ALSO DUPLICATED.

DELETE

THE DELETE OPERATOR DELETES THE CONTENTS OF NSS. IF THE CONTENTS OF NSS CONTAIN A REFERENCE TO VSS, THE CONTENTS OF VSS ARE ALSO DELETED.

EXCHANGE

THE EXCHANGE OPERATOR EXCHANGES THE TOP TWO ENTRIES OF NSS. IF BOTH ENTRIES OF NSS CONTAIN REFERENCES TO VSS, THE TOP TWO ENTRIES OF VSS ARE ALSO EXCHANGED.

VSS TO NSS

THIS OPERATOR MOVES THE CONTENTS OF VSS TO NSS. THE VSS ENTRY IS REMOVED FROM VSS AND ENTERED INTO NSS.

NSS TO VSS

THE CONTENTS OF NSS ARE ENTERED INTO VSS AND A REFERENCE TO THE NEW VSS ENTRY REPLACES THE ORIGINAL IN NSS. THE NEW ENTRY IS CLASSIFIED AS A LOGICAL FIELD.

PROGRAM CONTROL OPERATORS

LOOP (NAME)

LOOP IS USED TO BEGIN A PROGRAM LOOP. THE OPERATOR ASSUMES A CONTROL VARIABLE EXISTS IN VSS PRIOR TO EXECUTION OF THE LOOP OPERATOR. THE OPERATOR CAUSES A SUBROUTINE CALL ON THE NAMED SUBPROGRAM. THE TOP OF P IS DUPLICATED PRIOR TO THE ACTUAL EXECUTION OF THE SUBPROGRAM.

LOOP TEST

LOOP TEST TESTS THE VALUE OF THE VARIABLE IN VSS. IF IT IS LESS THAN ZERO, P, VSS, AND NSS ARE DELETED CAUSING A RETURN TO THE OPERATOR FOLLOWING THE LOOP OPERATOR. IF THE VALUE IS GREATER THAN OR EQUAL TO ZERO, P IS DUPLICATED AND EXECUTION RESUMED FROM THE START OF THE LOOP ROUTINE.

BRANCH -- BRANCH CONDITIONAL

BRANCH CAUSES A CHANGE IN PROGRAM CONTROL.

IF THE DESCRIPTION AT THE TOP OF N DESCRIBES DATA, THE DATA IS USED AS A PARAMETER TO SEQUENCE THE CURRENT PROGRAM VECTOR TO YIELD A NEW P SETTING. THE DATA MUST BE BINARY SIGNED INTEGER. NORMAL BOUNDS CHECKING IS PERFORMED.

IF THE DESCRIPTION AT THE TOP OF N DESCRIBES PROGRAM, THAT DESCRIPTION REPLACES THE CURRENT PROGRAM CONTAINER DESCRIPTION.

IF THE DESCRIPTION AT THE TOP OF N DESCRIBES ANOTHER DESCRIPTION, THIS NEW DESCRIPTION IS EVALUATED.

WHEN BRANCH CONDITIONAL IS USED, THE BOOLEAN AT THE TOP OF V IS TESTED. IF IT IS FALSE, THE ABOVE ACTION IS PERFORMED AFTER THE BOOLEAN AND ITS REFERENCE HAVE BEEN DELETED FROM V AND N. IF THE VALUE IS TRUE, EXECUTION PROCEEDS NORMALLY.

HALT

PROVIDES A MEANS OF PROGRAMMATICALLY STOPPING A PROCESSOR PRIOR TO THE NEXT OPERATOR FETCH.

THIS OPERATOR IS CONTROLLED BY A SWITCH. IF THE SWITCH IS IN THE "HALT" POSITION THE OPERATOR IS EXECUTED. IF THE SWITCH IS IN THE "CONDITIONAL HALT" POSITION THE CONTENTS OF P ARE COMPARED AGAINST THE CONTENTS OF THE CONDITIONAL HALT REGISTER. IF THE COMPARISON IS EQUAL, THE OPERATOR IS EXECUTED; OTHERWISE, IT IS TREATED AS A NO-OP.

IF THE SWITCH IS IN "NORMAL" POSITION THE OPERATOR IS TREATED AS A NO-OP.

NO-OP

NO-OP CAUSES P TO BE UPDATED AND THE NEXT OPERATOR TO BE EXECUTED.

PROCEDURE CONTROL OPERATORS

SLICE

SLICE ENTERS A DISPLAY CONTROL WORD (DCW) INTO NSS AND PERFORMS AN ENTER (N). A BIND OPERATION, WHICH USES THE CREATED REFERENCE AS THE ALLOCATED STRUCTURE AND A VECTOR TEMPLATE (TN) AS THE UNALLOCATED STRUCTURE, IS PERFORMED. THE CURRENT LL IS INCREASED BY ONE. THE NEW LL IS USED AS A PARAMETER TO PERFORM AN ENTER (D). THE NEW VECTOR DESCRIPTION IS STORED INTO D. THE OPERATOR THEN PERFORMS AN ENTER (V) AND DELETES THE REFERENCE CREATED.

UNSLICE

UNSLICE USES THE CURRENT LL TO PERFORM A REMOVE(D), AND THE CURRENT LL IS DECREASED BY ONE.

A REMOVE (N) IS PERFORMED AND THE CREATED REFERENCE IS DELETED. A REMOVE (V) IS PERFORMED AND THE CREATED REFERENCE IS DELETED.

PROCEDURE RETURN

PROCEDURE RETURN INVOKES A RETURN FROM A PROCEDURE OR FUNCTION.

THE REFERENCE PRODUCED BY A REMOVE (N) IS DELETED. THE REFERENCE PRODUCED BY A REMOVE (V) IS DELETED. IF THE PARAMETER ATTRIBUTE IS TRUE, THE ENTRY IN D INDICATED BY THE CURRENT LL IS REMOVED AND THE REFERENCE PRODUCED IS DELETED.

IF THE FUNCTION ATTRIBUTE IS TRUE, THE REMOVE D OPERATION IS REPEATED, AND AN UNSLICE IS PERFORMED ON N AND V. USING THE DCW IN NSS AS THE FIRST ELEMENT OF A DCW LIST, A DISPLAY UPDATE IS PERFORMED.

THE REFERENCE CREATED BY A REMOVE ON P IS DELETED. THE NEXT OPERATOR WILL BE FETCHED FROM THE PROGRAM DESCRIBED BY THE NEW P ENTRY.

COROUTINE OPERATORS

COROUTINES ARE ESTABLISHED BY ALLOCATING D, N, V, AND P. THE DESCRIPTIONS OF THESE STRUCTURES ARE RETAINED IN THE CURRENTLY ACTIVE N. THE CODE FOR THE COROUTINE BEING ESTABLISHED IS REFERENCED BY A DESCRIPTOR THAT IS ENTERED INTO THE P OF THE COROUTINE.

COROUTINE ACTIVATE (NAME)

THE COROUTINE ACTIVATE OPERATOR FINDS A REFERENCE TO THE D OF THE COROUTINE TO BE ACTIVATED ON THE TOP OF N. THE OPERATOR ENTERS THE FOLLOWING RECORD INTO THE CD STACK:

A. REFERENCE TO THE CURRENT D.

B. THE NAME PARAMETER OF THIS OPERATOR.

THE PROGRAM ENVIRONMENT IS CHANGED TO REFLECT THE COROUTINE BEING ACTIVATED. ALLOCATION AND PRESENCE CHECKS ARE ALSO ACCOMPLISHED.

COROUTINE CALL (NAME)

THE COROUTINE CALL OPERATOR TRANSFERS PROGRAM ENVIRONMENT FROM ONE COROUTINE TO ANOTHER COROUTINE IN THE FAMILY OF COROUTINES AT THIS LEVEL. THE TRANSFER OF PROGRAM ENVIRONMENT IS ACCOMPLISHED BY SAVING THE REFERENCES TO THE CURRENT COROUTINE D, N, V, AND P IN

THE LOCATIONS SPECIFIED BY THE REFERENCES IN THE PARENT P. THE ENTERING COROUTINES PROGRAM ENVIRONMENT IS FOUND AT THE LOCATION INDICATED BY THE NAMED PARAMETER. THIS PROGRAM ENVIRONMENT IS NOW ACTIVATED WITH THE C-ELEMENT BEING REPLACED BY THE NEW COROUTINE FIELDS. ALLOCATION AND PRESENCE CHECKS ARE MADE AT ACTIVATE TIME.

COROUTINE END

COROUTINE END REMOVES THE TOP ENTRY IN THE C STACK. THE NEW TOP IS USED TO RESET THE D, N, V, AND P OF THE PARENT TO WHICH RETURN IS BEING MADE.

PROCESS CALL OPERATORS

PROCESS PARAMETER (NAME)

PROCESS PARAMETER FETCHES THE NAMED DESCRIPTION AND BUILDS A REFERENCE BY EXECUTING THE NAMED DESCRIPTION. AN ABSOLUTE REFERENCE IS CALCULATED BY FINAL COMBINE. THIS ABSOLUTE REFERENCE IS ENTERED INTO RSS.

PROCESS CALL

PROCESS CALL CREATES A SLICE IN R, MAPS A VECTOR ON THE REFERENCE, AND ENTERS THE REFERENCE TO THAT SLICE IN RL.

PROCESS END

PROCESS END REMOVES THE CURRENT SLICE IN R AND THE REFERENCE TO THAT SLICE IN RL. THIS OPERATOR IS PRIVILEGED

MISCELLANEOUS OPERATORS

LITERAL

THE LITERAL OPERATOR PROVIDES A MECHANISM FOR INCLUDING LITERAL OPERANDS IN A PROGRAM STRING. OPERAND LENGTHS MAY BE 4, 8, 16, AND 32 BITS. THE OPERAND MAY BE PLACED IN EITHER NSS OR VSS DEPENDING

UPON THE TRANSFER TYPE SPECIFIED IN THE LITERAL OPERATOR. IF THE VALUE IS PLACED IN VSS THEN THE REFERENCE CONSTRUCTED IN NSS WILL HAVE THE FOLLOWING PROPERTIES:

A. ALL ACCESS ATTRIBUTES OFF.

B. STRUCTURE EXPRESSION SET TO FIELD WITH APPROPRIATE LENGTH.

C. INTERPRETER ATTRIBUTE IS SET TO DATA. THE FORMAT SELECTOR OF THE DATA IS SET FROM THE PREFIX OF THE LITERAL IN THE PROGRAM STRING.

SET INTERRUPT

SET INTERRUPT CAUSES AN INTERRUPT TO BE SET IN THE DEVICE INDICATED BY THE PARAMETER IN VSS (THE CHANNEL NUMBER OF THE UNIT). THE POSITION WITHIN THE INTERRUPT REGISTER IN THE SIGNALED DEVICE IS DETERMINED BY THE CHANNEL ASSIGNED TO THE DEVICE EXECUTING THE INTERRUPT OPERATOR.

TEST INTERRUPT (NAME)

THIS OPERATOR TESTS THE INTERRUPT REGISTER. IF THERE WAS AN INTERRUPT SET, THE BIT IS CLEARED, THE RELATIVE BIT NUMBER IS ENTERED INTO VSS, AND A REFERENCE TO IT IS ENTERED INTO VSS. THE NAME PASSED AS A PARAMETER WITH THIS OPERATOR IS TNE NAME OF THE INTERRUPT-HANDLING SUBROUTINE. THIS SUBROUTINE IS NOW ENTERED. IF THERE WERE NO INTERRUPTS SET, THE NEXT OPERATOR IS EXECUTED. THIS OPERATOR MAY BE EXECUTED IN IMP MODE ONLY.

GET V SPACE

THIS OPERATOR IS USED TO ALLOCATE SPACE IN V AND TO CREATE A REFERENCE IN N. THE OPERATOR USES THE DESCRIPTION OF AN UNALLOCATED FIELD (NO ADDRESS BUT WITH LENGTH AND FORMAT) IN NSS AND PERFORMS AN ENTER IN V OF THAT LENGTH. THE COMPLETED REFERENCE IS LEFT IN NSS.

IMP CALL

IMP CALL IS A MEANS BY WHICH A PROCESS EXECUTES PROCESSOR CONTROL FUNCTIONS. IMP MAY BE ENTERED PROGRAMMATICALLY BY IMP CALL OR AUTOMATICALLY BY FAULT. THE PROGRAMMATIC IMP CALL SETS THE PROGRAM

FAULT BIT. THE PROCESS ENVIRONMENT CURRENTLY HELD IN THE PROCESSOR IS SAVED IN THE PROCESS SPACE OF THE PROCESS DOING THE CALL. THEN THE PROCESSOR IS LOADED WITH IMP STATE FROM THE IMP PROCESS SPACE. AN IMP MODE FLIP-FLOP IS SET. THE NUMBER OF THE FAULT OR INTERRUPT BIT THAT CAUSED THE IMP CALL IS ENTERED IN VSS.

IMP RETURN

IMP RETURN ESTABLISHES A PROCESS ENVIRONMENT BY LOADING THE APPROPRIATE PROCESS STATE INTO THE PROCESSOR. IMP RETURN IS THE REVERSE OPERATION OF IMP CALL. THE IMP MODE FLIP-FLOP IS RESET. THIS OPERATOR IS PRIVILEGED.

STUFF ENVIRONMENT

A POINTER TO THE CURRENT DCW ON NS IS STORED IN THE LEXIC LINK PORTION OF THE PROGRAM DESCRIPTOR ON NSS. IF NSS DOES NOT CONTAIN A PROGRAM DESCRIPTOR NO ACTION TAKES PLACE.

SEQUENCE (NAME)

THE CURRENT ELEMENT POINTER, P, AND PRIOR ELEMENT POINTER, Q, IN THE STATE WORD OF THE FIRST LIST STRUCTURE INSTRUCTION ARE ADVANCED TOWARD THE TAIL OF THE LIST BY SETTING Q TO P AND P TO THE LINK OF THE ELEMENT POINTED TO BY P. IF P IS UNDEFINED A SEQUENCE FAULT OCCURS.

RESET (NAME)

IN THE STATE WORD OF THE FIRST LIST INSTRUCTION OF THE NAMED DESCRIPTION, THE CURRENT ELEMENT POINTER, P, IS SET TO THE HEAD POINTER, A, AND THE PRIOR ELEMENT POINTER, Q, IS SET TO UNDEFINED.

ARITHMETIC OPERATORS

THESE OPERATORS ARE DEFINED IN DETAIL IN SECTION 3 (AU):

TRANSFORM

ADD

SUBTRACT

MULTIPLY

MULTIPLY ADD

DIVIDE FOR INTEGER

DIVIDE FOR REMAINDER

DIVIDE FOR REAL

AND

OR

NOT

XOR

REDUCTION AND

REDUCTION OR

REDUCTION XOR

CHARACTER COMPARE -- EQL

CHARACTER COMPARE -- NEQ

CHARACTER COMPARE -- GEQ

CHARACTER COMPARE -- LEQ

CHARACTER COMPARE -- GTR

CHARACTER COMPARE -- LSS

TEST -- EQL

TEST -- NEQ

TEST -- GEQ

TEST -- LEQ

TEST -- GTR

TEST -- LSS

CHARACTER MOVE

ABSOLUTE

NEGATE

SCALE

COMPARE -- EQL

COMPARE -- NEQ

COMPARE -- GEQ

COMPARE -- LEQ

COMPARE -- GTR

COMPARE -- LSS

SECTION 3

ARITHMETIC UNIT

INTRODUCTION

THE ARITHMETIC UNIT (AU) PROVIDES THE CPU WITH THE CAPABILITY OF PERFORMING ARITHMETIC, CHARACTER, AND LOGICAL OPERATIONS ON DATA WITH THESE VARIABLE CHARACTERISTICS:

A. OPERAND LENGTHS:

1. THE AU ACCEPTS OPERANDS WITH LENGTHS FROM 0 TO 256 BITS.

2. ALL AU OPERATIONS WITH MULTIPLE INPUTS ACCEPT OPERANDS WITH DIFFERENT LENGTHS.

B. OPERAND FORMATS:

1. SEVERAL NUMERICAL FORMAT ASPECTS ARE VARIABLE. THEY HAVE BEEN CHOSEN TO MAXIMIZE EFFICIENCY IN OPERATING ON FREQUENTLY USED NUMBER REPRESENTATIONS.

2. AU OPERATIONS WITH MULTIPLE INPUTS ACCEPT OPERANDS WITH DIFFERENT FORMATS, WITH A LIMITED NUMBER OF EXCEPTIONS.

C. OPERATION VARIANTS:

1. ONE VARIANT BIT DETERMINES HOW THE AU SELECTS THE OUTPUT LENGTH AND FORMAT.

2. TWO VARIANT BITS DETERMINE THE MODE OF TRUNCATION EMPLOYED BY THE AU DURING OPERATION.

3. ONE VARIANT BIT DETERMINES WHEN THE AU NORMALIZES OPERANDS.

FUNCTION

THE AU OPERATES ON OPERANDS STORED IN THE TOP SPACES OF THE VSS. THE LENGTHS AND FORMATS OF THESE OPERANDS ARE SPECIFIED BY THE DESCRIPTIONS IN THE CORRESPONDING SPACES OF THE NSS. THE OPERATORS ARE SUPPLIED BY THE INTERPRETER, AND THE OPERATION VARIANTS ARE DETERMINED BY A FOUR-BIT OPERATION VARIANT REGISTER.

THE POTENTIALLY COMPLEX ASPECTS OF DATA REPRESENTATION, INCLUDING VARIABLE LENGTHS, VARIABLE FORMATS, EXTREME LENGTH, INTERLEAVED NUMERIC AND NONNUMERIC DATA (IN NUMERIC CHARACTER STRINGS), ETC., REQUIRE THE AU TO BE CAPABLE OF EXTENSIVE PROCESSING OF DATA AND DATA DESCRIPTIONS. THIS PROCESSING IS DONE IN AU REGISTERS. THUS, NO EXTRA PROCESSING CAPABILITY IS REQUIRED IN THE VALUE STACK AND NAME STACK EXCEPT TO STORE RESULTS.

NUMBER REPRESENTATION BACKGROUND

THERE IS A SIGNIFICANT REQUIREMENT FOR INTEGER ARITHMETIC. A DISTINCT INTEGER FORMAT IS RECOGNIZED (AS OPPOSED TO REPRESENTING INTEGERS IN A FLOATING-POINT FORMAT) IN ORDER TO SAVE STORAGE SPACE AND IMPROVE OPERATION EXECUTION TIME. THIS INTEGER FORMAT REQUIRES THAT THERE BE NO EXPONENT AND THAT THE RADIX POINT BE AT THE RIGHT OF THE DATA.

THE INTEGER FORMAT IS EXTENDED TO A FIXED-POINT FORMAT WITH THE POSITION OF THE POINT SPECIFIED BY THE FORMAT. THIS IS DONE BY A LENGTH VALUE CALLED THE BIAS, WHICH CAN BE DESCRIBED AS A CONSTANT EXPONENT. WHEN THE BIAS EQUALS ZERO, THEN THE FIXED-POINT NUMBER IS AN INTEGER. THE PURPOSE OF A FIXED-POINT FORMAT IS TO ALLOW A REAL NUMBER TO BE REPRESENTED WITHOUT FLOATING THE NUMBER, AND ITS USE IS EQUIVALENT TO ATTACHING AN EXPONENT FIELD TO THE NUMBER. THE LATTER METHOD IS INEFFICIENT FOR SOME APPLICATIONS.

SOME REAL NUMBERS MUST BE FLOATED, AND FOR THESE A FLOATING-POINT FORMAT EXISTS. THE RADIX POINT IS AT THE LEFT OF THE MANTISSA, AND THUS THE EFFICIENCY OF FLOATING-POINT OPERATIONS IS IMPROVED.

BECAUSE FLOATING-POINT NUMBERS INCLUDE TWO VARIABLES, THE EXPONENT AND THE MANTISSA, THERE ARE ALWAYS SEVERAL POSSIBLE REPRESENTATIONS FOR THE SAME NUMBER. IN ADDITION, THE SIGNIFICANCE OF THE MANTISSA CAN BE QUESTIONED AND MAY BE DIFFICULT TO DEFINE. THUS, THE AU FOLLOWS THESE PRINCIPLES IN OPERATING ON FLOATING-POINT NUMBERS:

A. ALL NUMBERS WILL BE TREATED AS IF THEY ARE EXACT.

B. WITHIN THE LIMIT OF A PREDETERMINED OUTPUT LENGTH, ALL POSSIBLE SIGNIFICANCE IS SAVED.

C. WHENEVER TRUNCATION OCCURS, THE OPERATION VARIANT RULES ARE FOLLOWED.

BINARY AND DECIMAL NUMBERS ARE BOTH IN COMMON USAGE, BUT OTHER ARITHMETIC UNITS GENERALLY HANDLE ONLY ONE TYPE. THE B8500 AU ACCEPTS EITHER BINARY NUMBERS OR DECIMAL NUMBERS (IN PACKED 4-BIT BCD FORMAT) AND APPLIES THE SAME SET OF OPERATORS AND VARIANTS TO BOTH TYPES.

FIXED-POINT NUMBERS MAY BE REPRESENTED IN 8-BIT ASCII OR EBCDIC CHARACTER FORMAT BECAUSE THESE ARE SIGNIFICANT CASES. FIXED-POINT NUMBERS REPRESENTED IN SUCH AN UNPACKED FORMAT GIVE THE SAME RESULTS AS IF THEY ARE REPRESENTED IN PACKED DECIMAL FORMAT, UNLESS THE EXTRA LENGTH OF THE UNPACKED FORMAT BECOMES A FACTOR. FLOATING-POINT NUMBERS MAY NOT BE REPRESENTED IN UNPACKED FORMAT.

NUMBER FORMAT VARIANTS

NUMBER TYPE (THREE BITS)

A. BINARY.

B. DECIMAL (PACKED BCD).

C. ASCII.

D. EBCDIC.

E. OCTAL.

SIGN TYPE (ONE BIT)

A. UNSIGNED.

B. SIGNED.

EXPONENT TYPE (ONE BIT)

A. FIXED POINT.

B. FLOATING POINT (ILLEGAL IF ASCII OR EBCDIC).

EXPONENT LENGTH (FIVE BITS)

MAXIMUM LENGTH OF BINARY EXPONENTS IS 30; MAXIMUM LENGTH OF DECIMAL EXPONENTS IS 24.

BIAS (NINE BITS)

OPERATION VARIANTS

OUTPUT LENGTH MODE (ONE BIT)

A. FIXED LENGTH: THE OUTPUT LENGTH IS EQUAL TO THE LENGTH OF THE INPUT WHOSE FORMAT IS SELECTED TO BE THE OUTPUT FORMAT.

B. VARIABLE LENGTH: THE AU DERIVES THE NATURAL LENGTH OF THE OUTPUT, WHICH IS USUALLY THE MINIMUM REQUIRED TO SAVE THE COMPLETE RESULT OF THE OPERATION.

TRUNCATION MODE (TWO BITS)

A. NOT ROUNDED, MAGNITUDE TRUNCATION: TRUNCATE TOWARDS ZERO.

B. NOT ROUNDED, ALGEBRAIC TRUNCATION: TRUNCATE POSITIVE NUMBERS TOWARD ZERO, NEGATIVE NUMBERS AWAY FROM ZERO.

C. ROUNDED, MAGNITUDE TRUNCATION: ROUND AWAY FROM ZERO.

D. ROUNDED, ALGEBRAIC TRUNCATION: ROUND POSITIVE NUMBERS AWAY FROM ZERO, ROUND NEGATIVE NUMBERS TOWARD ZERO.

NORMALIZATION MODE (ONE BIT)

A. UNCONDITIONAL NORMALIZATION: ALL FLOATING POINT RESULTS ARE NORMALIZED, INDEPENDENT OF THE INPUT(S) AND THE OPERATOR.

B. CONDITIONAL NORMALIZATION: ALL POSSIBLE SIGNIFICANCE IS SAVED, BUT NUMBERS ARE NOT NORMALIZED IF THERE IS NO SIGNIFICANCE TO BE SAVED. THE RESULT OF MULTIPLY IS NORMALIZED UNLESS BOTH INPUTS HAD ZERO EXPONENTS AND THE OUTPUT CAN HAVE A ZERO EXPONENT WITHOUT CAUSING A MANTISSA OVERFLOW.

OPERATORS

ARITHMETIC OPERATORS

A. INPUT FORMAT: NUMERICAL OPERANDS WITH VALID FORMATS AND WITH LENGTHS NOT EXCEEDING 256 BITS. DIFFERENCES IN FORMAT AT THE START OF A DYADIC OPERATION MAY CAUSE THE AU TO TRANSFORM AN INPUT BEFORE THE ACTUAL OPERATION. SUCH AN IMPLIED TRANSFORMATION OPERATES IN A MANNER SIMILAR TO AN EXPLICIT TRANSFORM OPERATOR. IF THE NUMBER BEING TRANSFORMED IS NOT AN INTEGER AND THE TRANSFORMATION IS BETWEEN BINARY AND DECIMAL, THEN A FAULT RESULTS AND THE INPUTS ARE SAVED WITHOUT CHANGE.

B. NUMERICAL COMPARE: B OPPOSED TO A HAS THE SAME RESULT AS B MINUS A OPPOSED TO 0, AND THE POSSIBLITIES OF FAULTING ARE THE SAME AS IN SUBTRACTION. THE RESULT IS A SINGLE-BIT, LOGICAL OPERAND WHICH SIGNIFIES A TRUE TEST IF IT IS ONE. THE REFERENCE TO THE RESULT ESTABLISHES ITS LENGTH AND FORMAT. THE FORMAT IS DETERMINED BY THE ARITHMETIC UNIT BY CONVENTION, THAT IS, THE AU ASSUMES THAT FORMAT #1 IS ALWAYS LOGICAL. IF SOME OTHER FORMAT IS SPECIFIED ON FORMAT #1 THE RESULTS ARE UNDEFINED.

1. GREATER THAN: $B > A$.

2. GREATER THAN OR EQUAL: $B \geq A$.

3. EQUAL: $B = A$.

4. LESS THAN OR EQUAL: $B \leq A$.

5. LESS THAN: $B < A$.

6. UNEQUAL: $B \neq A$.

C. TRANSFORM: THE REPRESENTATION OF A IS CHANGED TO FIT THE FORMAT AND LENGTH SPECIFIED BY THE OUTPUT DESCRIPTION GIVEN IN THE NSS. ANY COMBINATION OF A LEGAL INPUT NUMERICAL DESCRIPTION AND A LEGAL OUTPUT NUMERICAL DESCRIPTION IS A LEGAL INPUT TO TRANSFORM, UNLESS THE TRANSFORMATION INCLUDES A CHANGE BETWEEN BINARY AND DECIMAL AND THE INPUT IS NOT AN INTEGER. IN THE LATTER CASE, AN ILLEGAL TRANSFORMATION FAULT OCCURS.

D. OTHER ARITHMETIC OPERATORS: THE RESULT IS A NUMBER WITH THE SAME FORMAT AS ONE OF THE INPUTS. IF THE INPUTS HAVE DIFFERENT FORMATS, THE MORE GENERAL IS CHOSEN AS THE OUTPUT FORMAT. THE OUTPUT LENGTH IS DETERMINED BY THE AU IN AGREEMENT WITH THE OUTPUT LENGTH MODE BIT. IF A FAULT OTHER THAN ILLEGAL FORMAT OR ILLEGAL TRANSFORMATION OCCURS, THEN THE INPUTS ARE LOST AND AN UNDEFINED OUTPUT IS DEVELOPED.

1. ADD: B + A.

2. SUBTRACT: B - A.

3. MULTIPLY: B × A.

4. DIVIDE FOR INTEGER QUOTIENT: B/A.

5. DIVIDE FOR REMAINDER: MOD B/A.

6. DIVIDE FOR REAL QUOTIENT: B/A.

7. NEGATE: - A .

9. INTEGERIZE (ZERO THE EXPONENT AND ADJUST THE MANTISSA ACCORDINGLY).

CHARACTER OPERATORS

A. INPUT FORMAT: EIGHT-BIT ASCII OR EBCDIC CHARACTER STRINGS, WITH NO LIMIT AS TO LENGTH.

B. EXPANSION: IF AN INPUT IS EXTENDED, IT IS DONE BY ADDING THE APPROPRIATE BLANK CHARACTERS AT THE RIGHT END OF THE STRING.

C. CHARACTER COMPARE: TWO ASCII OR TWO EBCDIC CHARACTER STRINGS ARE COMPARED, THE SHORTER BEING EXPANDED TO EQUAL THE LONGER. THE COMPARISON TREATS THE STRINGS AS UNSIGNED INTEGERS. THE RESULT IS A ONE BIT LOGICAL OPERAND. SEE NUMERIC COMPARE FOR RULES ABOUT REFERENCE TO RESULT.

1. GREATER THAN: B > A.

2. GREATER THAN OR EQUAL: B ≥ A.

3. EQUAL: B = A.

4. LESS THAN OR EQUAL: B ≤ A.

5. LESS THAN: B < A.

6. UNEQUAL: B ≠ A.

D. CHARACTER MOVE: THE AU EXPANDS OR TRUNCATES (ON THE RIGHT) THE CHARACTER STRING TO FIT ITS DESTINATION SPACE IN LEVEL-1 STORAGE. THE EXPANSION CHARACTER USED IS THE ASCII OR EBCDIC BLANK.

A. FORMAT: INPUT(S) AND OUTPUT MUST BE LOGICAL OPERANDS, WITH LENGTHS NOT EXCEEDING 256 BITS.

B. REDUCTION LOGICAL OPERATORS: THE FOLLOWING LOGICAL OPERATIONS COMBINE ALL THE BITS OF THE INPUT INTO A SINGLE-BIT RESULT.

1. REDUCTION AND.

2. REDUCTION INCLUSIVE OR.

3. REDUCTION EXCLUSIVE OR.

C. OTHER LOGICAL OPERATORS: IF THERE ARE TWO INPUTS OF UNEQUAL LENGTH, THE SHORTER INPUT IS ALIGNED ON THE LEFT WITH THE LONGER, AND IS REPEATED UNTIL EQUAL IN LENGTH TO THE LONGER INPUT. THE RESULT LENGTH IS THAT OF THE LONGER INPUT. THE FOLLOWING ARE LOGICAL OPERATORS:

1. NOT.

2. AND.

3. INCLUSIVE OR.

4. EXCLUSIVE OR.

NUMBER REPRESENTATION CONSTANTS

UNIVERSAL

A, ALL NUMBERS ARE TRUE MAGNITUDES, SIGNED OR UNSIGNED.

B. OPERAND SIGN POSITION:

1. BINARY, OCTAL, OR PACKED DECIMAL: MOST SIGNIFICANT DIGIT POSITION OF OPERAND.

2, ASCII OR EBCDIC: HIGH-ORDER FOUR BITS OF THE MOST SIGNIFICANT CHARACTER.

C. SIGN CODE:

1. BINARY:

A, 0 = POSITIVE.

B. 1 = NEGATIVE.

2. DECIMAL:

A, 1100 = POSITIVE.

B. 1101 = NEGATIVE.

3. ASCII:

A. 1011 = POSITIVE.

B. 1101 = NEGATIVE.

4. EBCDIC

A, 1100 = POSITIVE.

B. 1101 = NEGATIVE.

5. UNDEFINED INPUT SIGN CODES: IF NOT THE DEFINED NEGATIVE CODE, THEN THE SIGN IS DECODED AS POSITIVE.

FLOATING-POINT

A. RADIX POINT IS AT THE LEFT END OF THE MANTISSA

B. EXPONENT.

1. EXPONENT IS AN INTEGER LOCATED BETWEEN THE OPERAND SIGN AND THE MANTISSA.

2. EXPONENT IS SIGNED IF AND ONLY IF THE FORMAT SIGN TYPE INDICATES "SIGNED".

C. BIAS.

1. BIAS IS A SIGNED-MAGNITUDE INTEGER TO BE ADDED TO THE OPERAND EXPONENT TO DERIVE THE ACTUAL EXPONENT.

2. BIAS MAGNITUDE HAS THE SAME FORMAT AND BASE AS THE EXPONENT.

3. BIAS SIGN IS A SINGLE BIT: 0 = POSITIVE AND 1 = NEGATIVE

D. BINARY FLOATING-POINT.

1. EXPONENT AND MANTISSA ARE BINARY.

2. RADIX IS TWO.

E. OCTAL FLOATING-POINT.

1. EXPONENT AND MANTISSA RRE BINARY.

2. RADIX IS EIGHT.

F. DECIMAL FLOATING-POINT.

1. EXPONENT AND MANTISSA ARE DECIMAL.

2. RADIX IS TEN.

G. MANTISSA IS ALWAYS SIGNED.

FIXED-POINT

A. RADIX POINT IS AT THE RIGHT END OF THE MANTISSA.

B. BIAS IS A SIGNED-MAGNITUDE BINARY-LENGTH FORMAT, THE VALUE OF WHICH IS EQUAL TO THE NUMBER OF NUMERIC BITS OVER WHICH THE RADIX POINT IS SHIFTED (NEGATIVE : SHIFT LEFT; POSITIVE : SHIFT RIGHT).

CHARACTERS

A. NUMERIC CODE FOR HIGH-ORDER FOUR BITS.

1. ASCII : 0101.

2. EBCDIC : 1111.

B. BLANK CHARACTER.

1. ASCII : 0010 0000.

2. EBCDIC : 0100 0000.

STANDARD CHARACTER REPRESENTATION

STANDARD REPRESENTATION OF CHARACTER DATA IN THE SYSTEM INCLUDES 4-BIT PACKED INTEGERS AND 8-BIT UNPACKED ASCII OR EBCDIC CHARACTERS. IN THE PACKED FORMAT DATA IS INTERPRETED IN UNITS OF FOUR BITS.

THE INTERNAL CODE IN 4-BIT FORMAT AS INTERPRETED BY THE ARITHMETIC UNIT IS:

CODE	VALUE
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	UNDEFINED
1011	UNDEFINED
1100	UNDEFINED
1101	UNDEFINED
1110	UNDEFINED
1111	UNDEFINED

IN 8-BIT FORM, DATA IS INTERPRETED IN UNITS OF EIGHT BITS. STANDARD 8-BIT CODE INTERPRETATIONS INCLUDE EBCDIC AND AN 8-BIT EXTENSION OF ASCII. WHEN A SIGN IS SPECIFIED, IT IS INTERPRETED AS THE HIGH-ORDER FOUR BITS OF THE MOST SIGNIFICANT CHARACTER.

TRANSFORMATIONS INVOLVING NUMERIC CHARACTER DATA

IN ALL CASES INVOLVING TRANSFORMATIONS OF CHARACTER FIELDS, ROUNDING ARITHMETIC MAY BE IMPOSED. THIS ROUNDING IS APPLIED TO THE INPUT FIELD PRIOR TO THE ACTUAL COMPLETION OF THE DATA MOVE, BUT THE EFFECTS OF THE ROUNDING ARITHMETIC ARE CONSIDERED IN DETERMINING OVERFLOW.

IN ALL CASES WHERE SCALE (BIAS) IS SPECIFIED FOR THE RESPECTIVE FIELDS IN A TRANSFORMATION, THE INPUT SCALE IS ADJUSTED TO THE RESULT SCALE WHILE THE INDICATED TRANSFORMATION IS BEING PERFORMED.

WHEN 4-BIT TO 8-BIT TRANSFORMATION IS INDICATED, THE MOST SIGNIFICANT FOUR BITS OF THE RECEIVING FIELD ARE SET TO THE CODE WHICH INDICATES THE NUMERIC SUBSET OF THE SELECTED 8-BIT CODE. THIS CODE IS 1111 IN THE CASE OF EBCDIC AND 0101 IN THE CASE OF ASCII. WHEN SIGNED DATA IS INDICATED FOR THE RESULT, THE HIGH-ORDER FOUR BITS OF THE MOST SIGNIFICANT CHARACTER IN THE FIELD ARE SET TO THE APPROPRIATE CODE AS DESCRIBED ABOVE.

LEADING OR TRAILING ZERO CHARACTERS ARE SUPPLIED TO FILL THE RESULT FIELD AFTER THE APPROPRIATE SCALE ADJUSTMENT, IF ANY, HAS BEEN ACCOMPLISHED. IF NONZERO LEADING CHARACTERS WOULD BE TRUNCATED AS A RESULT OF THE TRANSFORMATION, THE OVERFLOW FAULT IS SET.

WHEN 8-BIT TO 4-BIT TRANSFORMATION IS INDICATED AND THE RECEIVING FIELD SPECIFIES SIGNED DATA, THE APPROPRIATE SIGN CODE IS GENERATED AS DESCRIBED ABOVE. NORMAL SCALE AND/OR ROUNDING ARITHMETIC IS APPLIED.

WHEN 4-BIT TO 4-BIT TRANSFORMATION IS INDICATED, THE FIELDS ARE TREATED AS SIGNED OR UNSIGNED INTEGER FIELDS. IF THE INPUT FIELD IS SHORTER THAN THE RESULT FIELD AFTER SCALE ADJUSTMENT, LEADING ZERO CHARACTERS ARE SUPPLIED TO MAKE THE SIZE OF THE INPUT CONFORM TO THE RESULT SIZE. IF THE INPUT FIELD IS LONGER THAN THE RESULT FIELD AFTER SCALE ADJUSTMENTS AND NONZERO INTEGER VALUE CHARACTERS WOULD BE TRUNCATED, THE OVERFLOW FAULT IS SET.

SECTION 4

PROCESSES

PROCESS HIERARCHY

IN THIS SECTION THE TERM "PROCESS" IS USED TO DESCRIBE THE ASSOCIATION OF A PROCESSOR AND AN ADDRESS SPACE. THE ADDRESS SPACE IS THE SET OF ALL STORAGE THAT IS ACCESSIBLE BY THE PROCESS. A PROCESS IS AN EXECUTION OF SOME PROGRAM CODE BY A PROCESSOR. THE WORK SPACE OF THE PROCESS INCLUDES ALL INFORMATION PRIVATE TO THIS EXECUTION. IN THIS SENSE A PROCESS IS A DYNAMIC ENTITY. THE OBJECT OF THE PROCESS IS THE CHANGE OF STATE OCCURRING WITHIN ITS ADDRESS SPACE.

THE MOST ELEMENTARY PROCESS IS A SERIAL EXECUTION OF OPERATORS BY A SINGLE PROCESSOR. A PROCESS HIERARCHY IS ESTABLISHED BY PARTITIONING A PROCESS INTO SUBPROCESSES. A SUBPROCESS IS A PROCESS WHOSE ADDRESS SPACE IS PROPERLY INCLUDED IN THE ADDRESS SPACE OF ITS IMMEDIATE PARENT PROCESS. MULTIPROCESSING AND PARALLELISM OF EXECUTIONS ARE ATTAINED IN THE B8500 BY SUBDIVIDING A PROCESS INTO SUBPROCESSES. THE PARENT OF EVERY PROCESS IN THE SYSTEM IS CALLED THE GLOBAL PROCESS. ITS WORK SPACE INCLUDES ALL AVAILABLE STORAGE SPACE IN THE SYSTEM, AND IT IS THE ANCESTOR OF ALL PROCESSES.

BOOTSTRAPPING THE SYSTEM IS ACCOMPLISHED BY INITIALIZING THE ADDRESS SPACE OF THE GLOBAL PROCESS AND ESTABLISHING EXECUTION OF THE PROCESS BY ASSOCIATING A GIVEN PROCESSOR WITH THIS SPACE (STARTING EXECUTION AT SOME PREDETERMINED POINT).

IN THE FOLLOWING DISCUSSION, THE TERM "PROCESS" IS USED TO DENOTE AN ELEMENTARY PROCESS. THE TERM "COMPOUND PROCESS" OR "PROCESS SUBTREE" IS USED TO DENOTE A SET OF SUBPROCESSES OR THE TREE STRUCTURE OF PROCESSES.

WORK SPACE

THE ADDRESS SPACE OF A PROCESS MAY EXTEND OVER THREE LEVELS OF STORAGE: LEVEL-1, LEVEL-2, AND LEVEL-3. REFERENCES TO THE WORK SPACE OF A PROCESS ARE RELATIVE TO THE DEFINITIONS OF THE PHYSICALLY ALLOCATED WORK SPACE AND ARE BOUND ON ACCESS BY THE FINAL COMBINE OPERATION. EACH PROCESSOR, WHEN ASSOCIATED WITH SUBPROCESS WORK SPACES, STACKS THE DEFINITION OF THE NEW WORK SPACES IN ITS PROCESSOR RESOURCE STRUCTURE.

THE INDIVIDUAL PROCESS RESOURCE STRUCTURE PERMITS EACH PROCESSOR TO KEEP TRACK OF ITS PATH INTO THE TREE OF PROCESSES FROM THE GLOBAL PROCESS SPACE TO THE PROCESS SPACE WITH WHICH IT IS CURRENTLY ASSOCIATED.

PROCESS RESOURCE STRUCTURE

THE PROCESS RESOURCE STRUCTURE IS THE MECHANISM USED TO PASS ALL RESOURCES BETWEEN PROCESSES OF THE PROCESS TREE. THE RESOURCE STRUCTURE IS AN INTEGRAL PART OF THE RESOURCE PROTECTION SCHEME. THERE IS A RESOURCE STRUCTURE FOR EACH PROCESSOR IN THE SYSTEM. AS THE PROCESSOR MOVES LEAFWARD IN THE PROCESS TREE, ALLOCATED RESOURCES ARE STACKED IN THE PROCESSORS RESOURCE STRUCTURE. AS THE PROCESSOR MOVES ROOTWARD RESOURCES ARE REMOVED FROM THE STRUCTURE. IN THIS WAY THE RESOURCE STRUCTURE(S) CONTAIN ALL DYNAMICALLY ALLOCATED RESOURCES ALONG THE PATH FOLLOWED BY THE PROCESSOR(S) THROUGH THE PROCESS TREE. THE RESOURCE STRUCTURES ARE TOTALLY ALLOCATED OUTSIDE OF THE PROCESS TREE AND THEREFORE ARE NOT ACCESSABLE BY THE PROCESSES. IMP IS THE ONLY PROCESS WHICH MAY DIRECTLY ACCESS ENTRIES IN THE RESOURCE STRUCTURE.

THE RESOURCE STRUCTURE CONSISTS OF A SUBSTRUCTURE CALLED R WHICH CONSISTS OF A PUSHDOWN MAPPED ONTO A STACK. THE ENTRIES IN THE STACK DESCRIBE THE INDIVIDUAL RESOURCES WHICH HAVE BEEN ALLOCATED. THE ENTRIES IN THE PUSHDOWN REPRESENT THE PARTITIONING OF THESE RESOURCES BY PROCESS. A GIVEN PROCESS MAY PASS RESOURCES TO A SUBPROCESS BY EXECUTING THE PROCESS PARAMETER OPERATOR. THESE RESOURCES ARE PLACED ON TOP OF THE RESOURCE STACK (RSS). WHEN THE SUBPROCESS IS ENTERED (PROCESS CALL) THE COLLECTION OF RESOURCES ON TOP OF R ARE BOUND TOGETHER BY MAKING AN ENTRY IN THE PUSHDOWN. A REFERENCE TO THE BOUND RESOURCES IS PLACED IN A DISPLAY STRUCTURE CALLED RR. THIS REFERENCE IS A VECTOR DESCRIPTION WHICH ALLOWS ACCESSING OF THE BOUND SEGMENT CONTAINER DESCRIPTIONS BY INDEX (SEG. CONTAINER NUMBER).

ALL RESOURCES PLACED IN THE RESOURCE STRUCTURE ARE FINAL COMBINED PRIOR TO ENTRY INTO THE STRUCTURE, AND THEREFORE, REPRESENT ABSOLUTE RESOURCES. THE SELECTION OF THE PROPER RESOURCE CONTAINER FROM THE RESOURCE STRUCTURE IS ACCOMPLISHED BY USING THE SEGMENT NUMBER WHICH IS PART OF EVERY TERMINAL DESCRIPTION.

IT SHOULD BE NOTED THAT THE ONLY RESOURCES ACCESSABLE BY A PROCESS ARE THOSE WHICH ARE CURRENTLY ON TOP OF THE RESOURCE STRUCTURE PUSHDOWN, I.E., IN RS.

A TYPICAL ENTRY IN THE RESOURCE STRUCTURE CONSISTS OF:

A. STORAGE LEVEL.

B. READ FAULT.

C. WRITE FAULT.

D. SUBSTORAGE LEVEL.

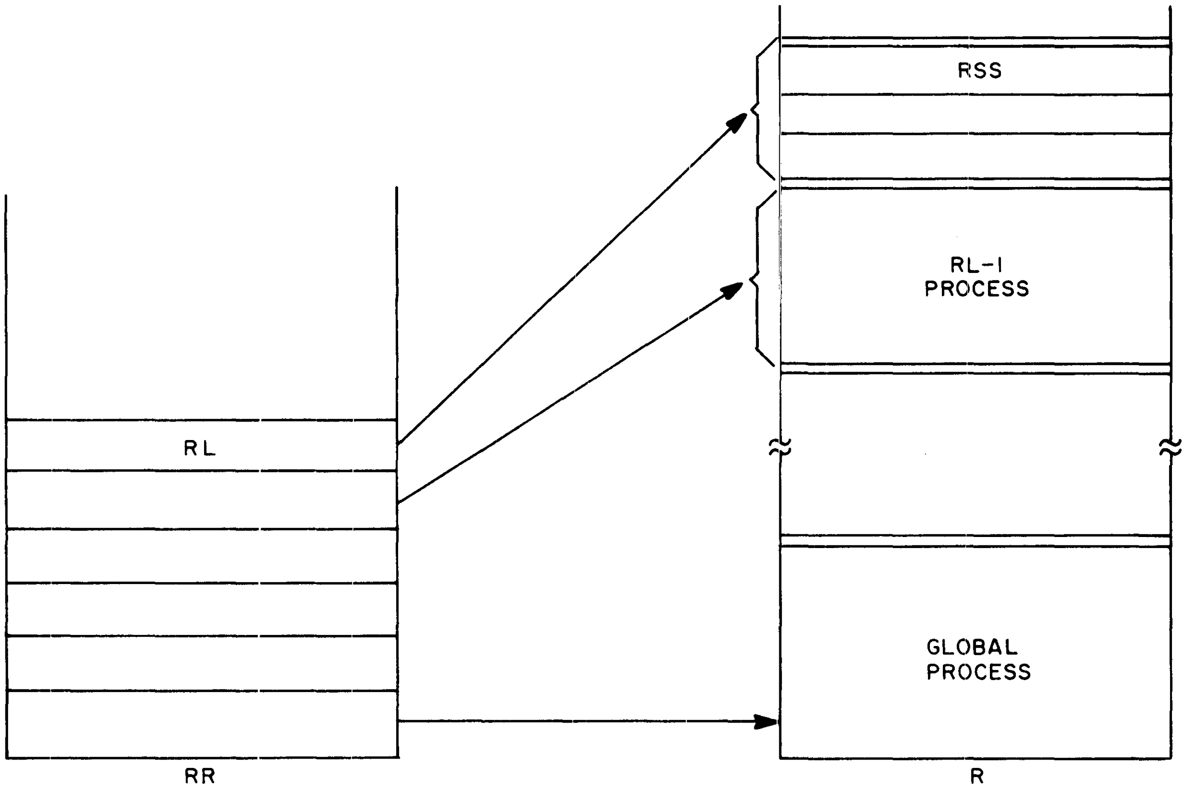
E. ABSOLUTE FIELD DESCRIPTION.

STORAGE LEVEL AND SUBSTORAGE LEVEL DEFINE THE KIND OF RESOURCE.

READ FAULT ALLOWS A PARENT TO GAIN CONTROL IF THE SUBPROCESS ATTEMPTS TO READ THE ASSOCIATED RESOURCE.

WRITE FAULT ALLOWS A PARENT TO GAIN CONTROL IF THE PROCESS ATTEMPTS TO WRITE INTO THE ASSOCIATED RESOURCE.

THE FIELD DESCRIPTION DEFINES THE RESOURCE.



R - RESOURCE STACK
RR - RESOURCE STACK DISPLAY
RL - CURRENT RESOURCE SLICE REFERENCE
RSS - TOP RESOURCE STACK SEGMENT CONTAINER

FIGURE 4-1, PROCESS RESOURCE STRUCTURE

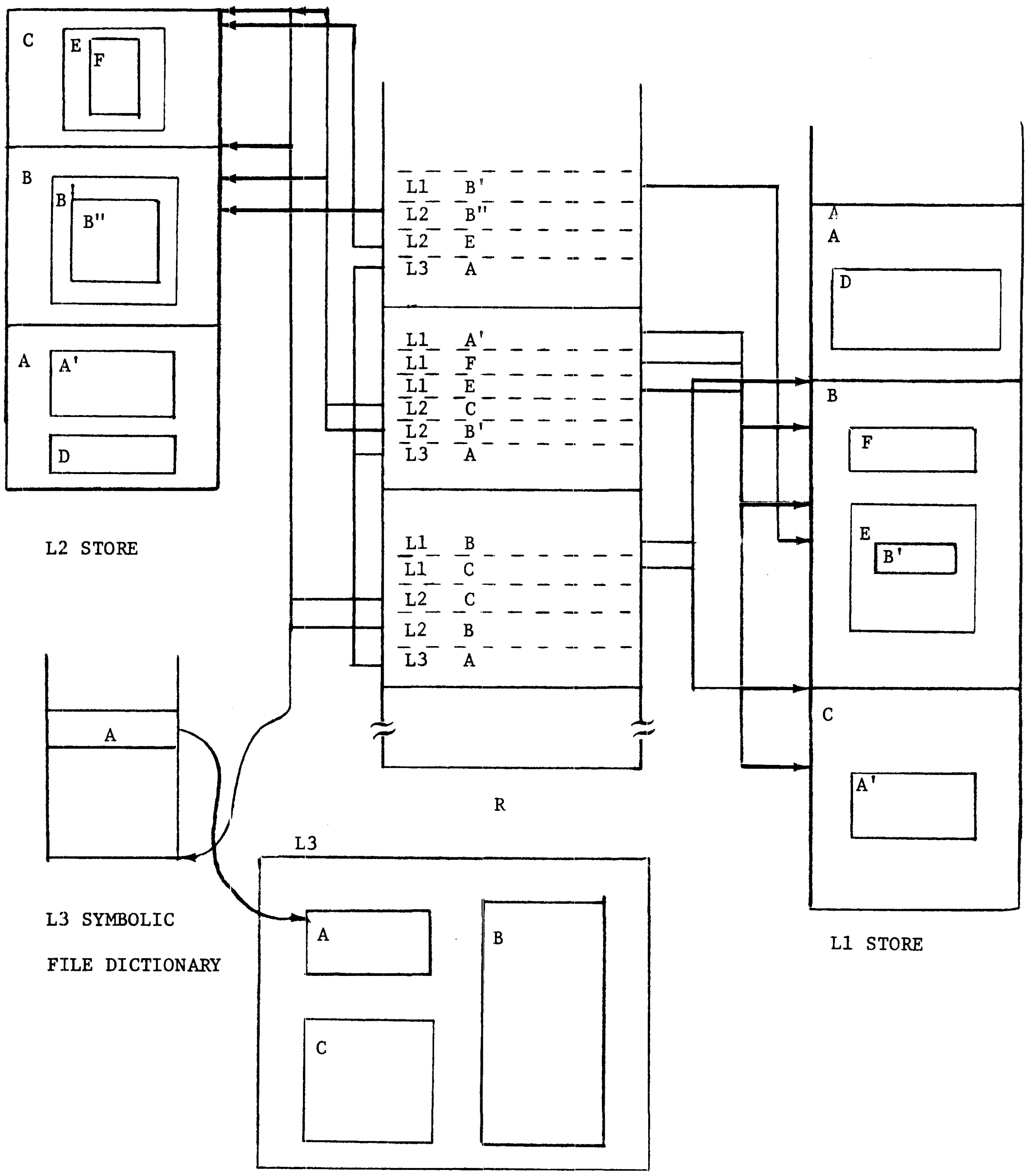


FIGURE 4-2, PROCESS SEGMENTATION

PROTECTION

PROTECTION OF A PROCESS AGAINST A FAULTY SUBPROCESS MAY BE OBTAINED BY THE ESTABLISHMENT OF PROTECTION AT THE TIME THE PROCESS IS CREATED. THE PROTECTION IS ACCOMPLISHED WITH THE IMPOSITION BY THE PARENT PROCESS OF AN ADEQUATE RESTRICTION ON THE RESOURCE DESCRIPTIONS OF THE SUBPROCESS. THIS RESTRICTION, IN FACT, IS THE STIPULATION THAT THE ADDRESS SPACE OF THE SUBPROCESS IS LIMITED TO THE WORK SPACES AND RESOURCES GIVEN TO THE SUBPROCESS. SHOULD A PROCESS ATTEMPT TO MAKE AN ACCESS OUTSIDE ITS RESOURCES, A FAULT WILL RESULT. PROTECTION IS ACHIEVED THROUGH THE USE OF THE FINAL COMBINE OPERATION.

IN LEVEL-1 AND LEVEL-2, FINAL COMBINE OPERATES AS DESCRIBED IN THE SECTION 2 OPERATOR DESCRIPTIONS. IN LEVEL-3 THE FINAL COMBINE OPERATION IS IMPLEMENTED USING LEVEL-3 DEFINITIONS IN THE RESOURCE STRUCTURE. A WORK SPACE IN LEVEL-3 IS A LIST OF INTEGERS USED AS INDEXES INTO THE PARENT RESOURCE LIST. FINAL COMBINE USES THE RESOURCE STRUCTURE TO FIND THE ABSOLUTE REFERENCE IN THE LIST OF THE GLOBAL PROCESS WORK SPACE. THE RESOLUTION OF LEVEL-3 IS TO THE DEVICES.

FAULTS AND INTERRUPTS

A DISTINCTION IS MADE BETWEEN PROCESS-DEPENDENT AND PROCESS-INDEPENDENT INTERRUPTS. THE FORMER ARE CALLED FAULTS, AND THE LATTER ARE CALLED INTERRUPTS. FAULTS ARE DETECTED BY THE PROCESSOR AS PART OF THE CURRENT EXECUTION. AMONG THE FAULTS DETECTED ARE:

A. ARITHMETIC-UNIT-DETECTED FAULTS.

1. OVERFLOW.
2. UNDERFLOW.
3. DIVIDE BY ZERO.
4. ILLEGAL FORMAT.
5. ILLEGAL OPERATION.
6. ARITHMETIC BUFFER OVERFLOW.

B. INTERPRETER-DETECTED FAULTS.

1. LEVEL-1, LEVEL-2, AND LEVEL-3 FINAL-COMBINE OUT-OF-BOUNDS (RESOURCE SPACE).

2. LEVEL-1, LEVEL-2, AND LEVEL-3 FINAL-COMBINE ACCESS PERMISSION FAULT.

3. DESCRIPTION EXECUTION RESULTING IN A CONTAINER SPACE OUT-OF-BOUNDS.

4. ILLEGAL OPERATOR.

5. OPERATOR IMP CALL SET.

6. TIMER RUN OUT.

C. HARDWARE MALFUNCTIONS.

1. PARITY ERROR ON LEVEL-1 ACCESS.

2. NO ACCESS TO LEVEL-1.

FAULTS

THESE FAULTS EXIST:

A. INTERNAL: THE PARENT PROCESS ASSUMES THAT ITS SUBPROCESS WILL HANDLE THE FAULT.

B. EXTERNAL: THE PARENT PROCESS WANTS TO RECEIVE CONTROL ON THE OCCURRENCE OF THIS FAULT IN THE COMPOUND PROCESS.

C. LOCALLY EXTERNAL: THE PARENT PROCESS WANTS TO RECEIVE CONTROL ON THE OCCURRENCE OF THIS FAULT IN THE SUBPROCESS (AS AN ELEMENTARY PROCESS).

BECAUSE THEY ARE PROCESS-DEPENDENT, FAULTS INTERNAL TO A PROCESS MAY BE PASSED BY THAT PROCESS TO A SUBPROCESS AS INTERNAL FAULTS. A MASK DEFINES WHICH FAULTS ARE INTERNAL FAULTS. AN INTERNAL FAULT IS HANDLED BY AN ACCIDENTAL ENTRY TO THE HANDLING PROCEDURE. THE PROCESSOR REMAINS ASSOCIATED WITH ITS CURRENT WORK SPACE, AND THE STACKS OF THE PROCESS ARE USED.

NON-INTERNAL FAULTS, CALLED EXTERNAL FAULTS, ARE RECOGNIZED BY THE INTERPRETER HARDWARE WHICH PERFORMS AN IMP CALL OPERATOR. THIS MECHANISM ALLOWS A PROCESS TO BE SIGNALED AND ACTIVATED ON OCCURRENCE OF A SUBPROCESS FAULT.

EXTERNAL FAULTS MAY IMPLY SIGNALING THE IMMEDIATE PARENT (LOCALLY EXTERNAL) OR ANY ANCESTOR PROCESS WHICH OWNS THE FAULT AS AN INTERNAL FAULT. SIGNALING THE OCCURRENCE OF AN EXTERNAL FAULT IS IMPLEMENTED BY ENCODING THE FAULT NUMBER IN THE VALUE STACK OF IMP.

THE PROCESSOR THEN LEAVES THE PROCESS SPACE OF THE FAULTING PROCESS AND REINITIATES ITSELF BY IMP.

THE RULE FOR PROPER NESTING OF FAULTS IS THAT EXTERNAL FAULTS TO A PARENT PROCESS MUST REMAIN EXTERNAL TO A SUBPROCESS AND LOCALLY EXTERNAL FAULTS MAY BECOME EXTERNAL OR REMAIN LOCALLY EXTERNAL TO THE SUBPROCESS. INTERNAL PARENTAL FAULTS MAY BECOME EXTERNAL, LOCALLY EXTERNAL, OR INTERNAL TO THE SUBPROCESS. THIS NESTING IS INSURED BY THE PROCESS CALL OPERATOR.

EXAMPLES OF FAULT HANDLING CAPABILITY ARE:

A. A DIVIDE BY ZERO FAULT MAY BE HANDLED INTERNALLY.

B. A TIMER RUN OUT MAY BE EXTERNAL OR LOCALLY EXTERNAL.

C. A MEMORY PARITY ERROR WILL BE EXTERNAL FOR ALL PROCESSES BEYOND THE GLOBAL PROCESS.

INTERRUPTS

INTERRUPTS ARE PROCESS INDEPENDENT IN THAT THEIR OCCURRENCE IS NOT CONTROLLED BY THE CURRENT EXECUTION OF A PROCESSOR. INTERPROCESSOR COMMUNICATION AND I/O COMPLETION (LEVEL-3 MOVES) ARE EXAMPLES OF INTERRUPTS. A MATRIX OF 16 × 16 HALF-DUPLEX BUSES ALLOWS ANY PROCESSOR, I/O MODULE, OR MEMORY EXTENSION CONTROLLER (MEC) TO INTERRUPT ANY OTHER MODULE.

THE SENSING OF AN INTERRUPT BY THE PROCESSOR FORCES THE EXECUTION OF AN IMP CALL. THEREFORE, INTERRUPTS ARE INVISIBLE TO THE PROCESSES. ONLY IMP IN EACH PROCESSOR CAN TREAT INTERRUPTS OR SEND INTERRUPTS TO OTHER MODULES.

EACH MODULE MAY COMPLEMENT THE INTERRUPT BY A MESSAGE LEFT IN A LEVEL-1 INTERCOMMUNICATION ARRAY COMMON TO ALL IMPS. THROUGH THE USE OF THIS INTERRUPT FACILITY AND THE LEVEL-1 MAILBOX, PROCESSORS CAN COMMUNICATE WHEN PERFORMING IMP FUNCTIONS. USING THE SAME MECHANISM, AN I/O MODULE OR MEC MAY SEND INTERRUPTS TO ANY INTERPRETER.

TYPES OF PROCESSES

IN THE HIERARCHY, THERE ARE TWO TYPES OF PROCESSES CALLED LEAVES AND NODES. LEAVES ARE PROGRAMS THAT ARE EXECUTED IN THE CONVENTIONAL SENSE; USER PROGRAMS IN CURRENT MULTIPROCESSING SYSTEMS ARE GOOD EXAMPLES OF LEAVES.

THE FUNCTIONS A LEAF PROCESS PERFORMS ARE:

A. RELEASES ITS PROCESSOR.

B. REQUESTS A SERVICE OF ITS PARENT PROCESS AND WAITS UNTIL THE REQUEST IS ACKNOWLEDGED BY THE PARENT.

C. INITIATES ASYNCHRONOUS MOVES TO AND FROM LEVEL-2 AND LEVEL-3 (RELEASING THE PROCESSOR AND WAITING FOR COMPLETION).

D. EVOLVES INTO A NODE PROCESS BY SPAWNING SUBPROCESSES AND ASSUMING THEIR MANAGEMENT AND CONTROL.

A NODE PROCESS HAS FOR ITS ONLY TASK THE CONTROL OF ITS SUBPROCESSES.

BECAUSE A NODE MUST ALWAYS BE IN A POSITION TO RECEIVE A PROCESSOR BOTH FROM SUBPROCESSES AND FROM ITS OWN PARENT PROCESS, A NODE CAN NEVER WAIT ON AN EVENT; IN OTHER WORDS, IT MUST ALWAYS BE POSSIBLE TO ACTIVATE A NODE BY PASSING A PROCESSOR TO IT. THE FOREGOING STATEMENT IS OF PRIME IMPORTANCE IN THE DESIGN OF A HIERARCHICAL SYSTEM. IT IS ESPECIALLY IMPORTANT TO THE OPERATING SYSTEM DESIGN.

FOR EXAMPLE, A NODE CAN SERVICE A REQUEST FOR MORE RESOURCES FROM A SUBPROCESS OR SOME PROCESSING FROM ITS PARENT. IT MAY FIND THAT A MESSAGE MUST BE DISPATCHED TO ANOTHER SUBPROCESS OR THAT IT MUST PASS THE MESSAGE TO ITS PARENT OR ANY COMBINATION OF THE ABOVE.

DISPATCHING AND SCHEDULING ARE THE MEANS BY WHICH PROCESSES ARE CONTROLLED. THESE TERMS ARE DEFINED AS FOLLOWS:

A. DISPATCHING: THE ACTION INVOLVED IN RECOGNIZING REQUESTS FROM LEAVES OR RECOGNIZING OTHER MESSAGES FROM SONS AND EITHER TRANSMITTING THE MESSAGES (LEAFWARD OR ROOTWARD) OR SERVICING CERTAIN MESSAGES LOCALLY.

B. SCHEDULING: THE OPERATION OF SELECTING AND ACTIVATING A SUBPROCESS OR RETURNING THE PROCESSOR TO THE PARENT PROCESS.

THE TWO STAGES (DISPATCHING AND SCHEDULING) ARE WELL SEPARATED (SEE FIGURE 4-3). WHEN A NODE IS ACTIVATED, ALL MESSAGES IN THE PROCESS QUEUE ARE PROCESSED.

THE NORMAL WAY TO ENTER THE SCHEDULER IS ON A QUEUE-EMPTY CONDITION.

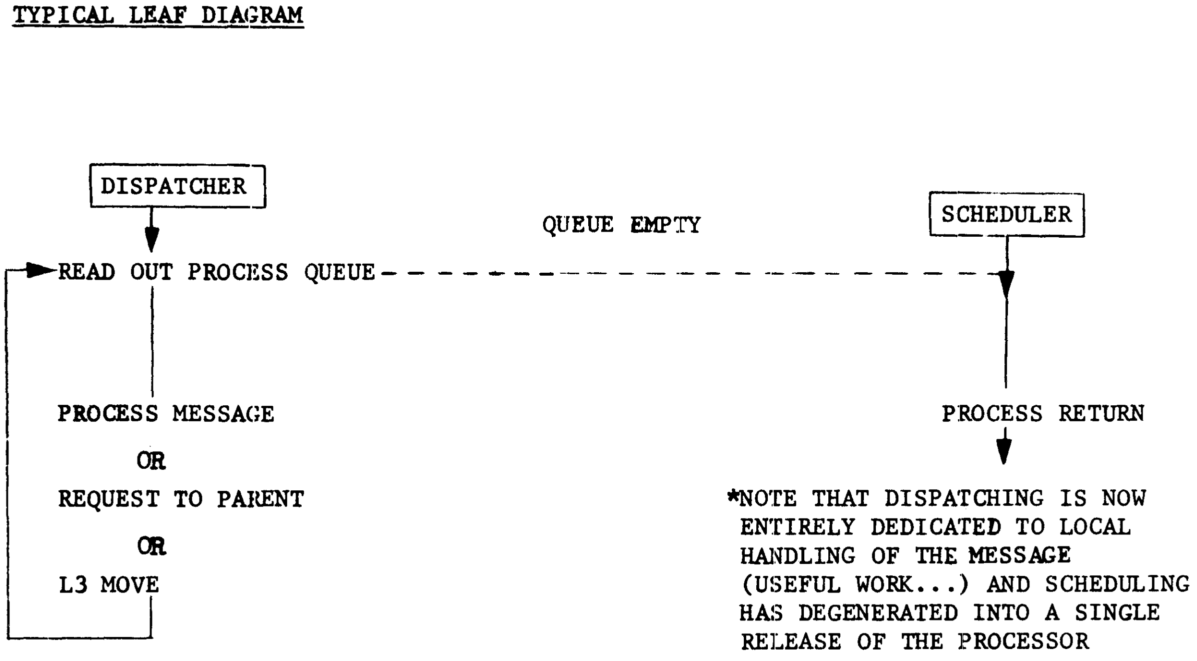
NODES MAY EXERCISE DIFFERENT STRATEGIES TO CHOOSE THE MOST APPROPRIATE BRANCH TO WHICH TO SEND A PROCESSOR. THE INFORMATION MAINTAINED AND RELIABLY PROPAGATED ALL ALONG THE PROCESSOR PATH IN THE PROCESS SPACE TREE CONSISTS OF THE ACTUAL NEEDS OF THE PROCESSORS, THE CURRENT NUMBER OF PROCESSORS, AND THE CURRENT NUMBER OF ACTIVE I/O OPERATIONS ASSIGNED TO THE SUBTREE.

IMP DESIGN PHILOSOPHY

IMP IS THE NAME OF AN INTERPRETER AVAILABLE TO EVERY PROCESS IN THE PROCESS TREE STRUCTURE; IT IS THOUGHT OF AS A SET OF HARDWARE OPERATORS. IMP IS PRIVILEGED, NONINTERRUPTIBLE, AND RELIABLE.

IMP PROVIDES THE MINIMUM AMOUNT OF MACHINERY REQUIRED TO CIRCULATE THE PROCESSORS IN THE PROCESS WORK SPACE HIERARCHY. THIS CIRCULATION OF PROCESSORS MUST BE POSSIBLE IN ANY PROGRAMMING ENVIRONMENT AND ESPECIALLY MUST ALLOW FOR GRADUAL PROTECTION IN THE HIERARCHY OF PROCESSES. A PROCESSOR MUST, AMONG OTHER THINGS, ALWAYS BE CAPABLE OF RETURNING ROOTWARD TO A LEVEL IN THE TREE WHERE THE SYSTEM IS STILL RELIABLE (RETURN TO PROGRAMS FREE OF ERRORS).

ON THE OTHER HAND, IMP DOES NOT MAKE ANY ASSUMPTION REGARDING THE ACTUAL RESOURCE ALLOCATION STRATEGIES USED AT EACH NODE. THIS IS CONSISTENT WITH THE GENERAL PHILOSOPHY OF A PARTITIONED SYSTEM IN WHICH VARIOUS UNIQUE ALGORITHMS ARE USED TO DECENTRALIZE THE RESPONSIBILITIES OF RESOURCE ALLOCATION (AMONG THESE, THE RESPONSIBILITIES FOR SPACE ALLOCATION AND PROCESSOR SCHEDULING).


TYPICAL LEAF DIAGRAM
DISPATCHER
READ OUT PROCESS QUEUE
QUEUE EMPTY
SCHEDULER
PROCESS MESSAGE
OR
REQUEST TO PARENT
OR
L3 MOVE
PROCESS RETURN
*NOTE THAT DISPATCHING IS NOW ENTIRELY DEDICATED TO LOCAL HANDLING OF THE MESSAGE (USEFUL WORK...) AND SCHEDULING HAS DEGENERATED INTO A SINGLE RELEASE OF THE PROCESSOR

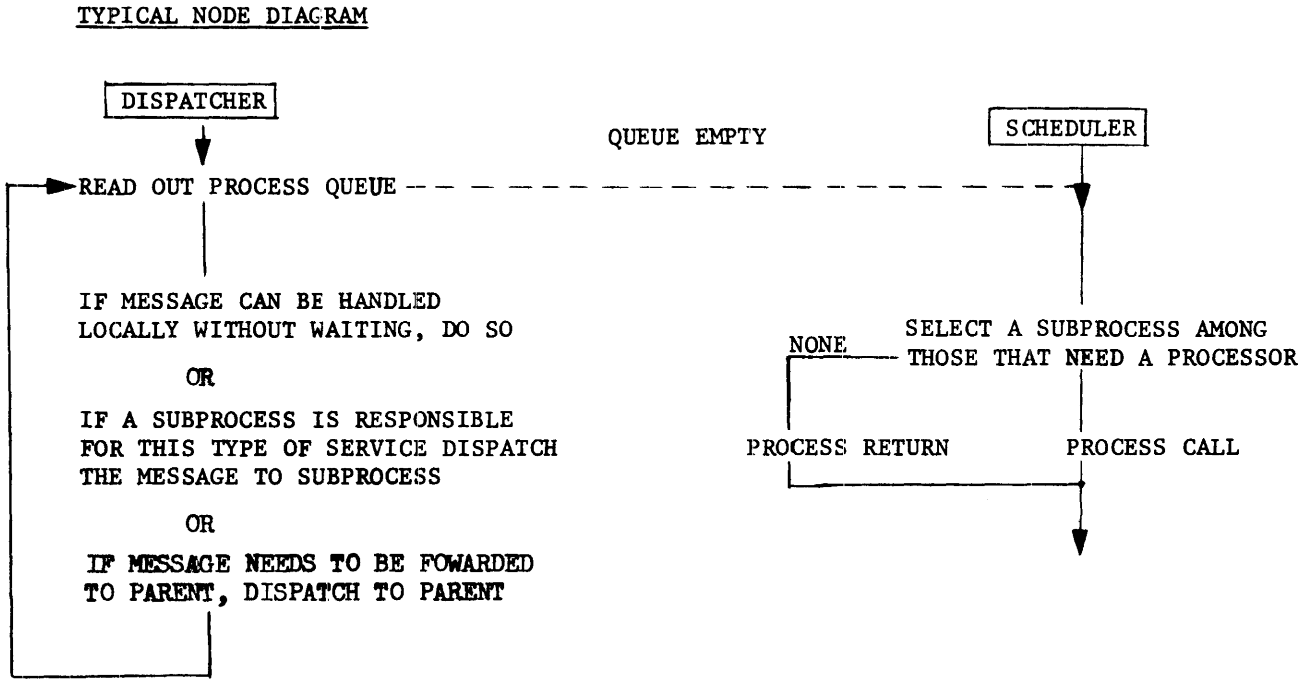

TYPICAL NODE DIAGRAM
DISPATCHER
READ OUT PROCESS QUEUE
QUEUE EMPTY
SCHEDULER
IF MESSAGE CAN BE HANDLED LOCALLY WITHOUT WAITING, DO SO
OR
IF A SUBPROCESS IS RESPONSIBLE FOR THIS TYPE OF SERVICE DISPATCH THE MESSAGE TO SUBPROCESS
OR
IF MESSAGE NEEDS TO BE FOWARDED TO PARENT, DISPATCH TO PARENT
NONE
SELECT A SUBPROCESS AMONG THOSE THAT NEED A PROCESSOR
PROCESS RETURN
PROCESS CALL

FIGURE 4-3, PROCESSING LEAF AND NODE FUNCTIONS

DATA BASE OF IMP

THE DATA BASE OF IMP IS REPRESENTED BY THE FOLLOWING CATEGORIES: (1) PROCESS CONTROL INFORMATION, (2) PROCESS V STRUCTURE, (3) IMP-S D, N, V AND P STRUCTURES, (4) RESOURCE STRUCTURES OF EACH PROCESSOR, AND (5) DATA PECULIAR TO IMP.

PROCESS CONTROL INFORMATION

A. PROCESS QUEUE: EACH PROCESS WORK SPACE CONTAINS A FIRST-IN, FIRST-OUT QUEUE CALLED PROCESS QUEUE, WHICH RESIDES AT A PREDETERMINED LOCATION.

ENTRIES INTO THE QUEUE ARE INITIATED BY THE PROCESSES OR BY IMP. PROCESS INITIATED ENTRIES ARE MADE EITHER BY A SUBPROCESS THROUGH AN IMP CALL OR DIRECTLY BY A PARENT. THE ENTRIES ARE REMOVED BY THE PROCESS ITSELF. IN ORDER TO AVOID PROCESSOR CONFLICTS, THE QUEUE MUST BE LOCKED BEFORE EACH SERVICE OF THE QUEUE IS MADE. HOWEVER, IMP WILL NOT WAIT ON A PROCESS-INITIATED ACCESS TO A LOCKED QUEUE; I.E., ON ENCOUNTERING AN ALREADY LOCKED QUEUE, IMP WILL RETURN TO THE CALLER WITHOUT WAITING FOR THE RELEASE OF THAT LOCK BIT.

A TYPICAL ENTRY IN THIS QUEUE CONTAINS AN IDENTIFICATION DEPENDING ON THE FUNCTION PERFORMED BY IMP. FOR EXAMPLE, IN THE CASE OF EXTERNAL FAULTS, THE FAULT-ID IS QUEUED IN THE ANCESTOR PROCESS FOR WHICH THIS FAULT IS INTERNAL. (SEE DETAILED EXPLANATIONS OF THE IMP FUNCTIONS BELOW.)

B. READY FIELD: INDICATES WHICH SUBPROCESSES WORK SPACES ARE READY TO RECEIVE A PROCESSOR. THESE SUBPROCESSES CAN BEGIN EXECUTION AS SOON AS A PROCESSOR IS DELIVERED.

C. MONOPROCESSOR BIT: ALLOWS ONE PROCESSOR IN THE SUBTREE WHOSE ROOT HAS A MONOPROCESSOR BIT SET; I.E., A PROCESS CALL TO A SUBPROCESS WHOSE MONOPROCESSOR BIT IS ALREADY SET IS INEFFECTIVE.

D. ACTIVE BIT: INDICATES THAT THE PROCESS WORK SPACE AT THIS POINT IN THE TREE CONTAINS A PROCESSOR AND IS THEREFORE AN ACTIVE PROCESS.

E. PROCESS NAME FIELD: IS AN INDEX USED TO LOCATE THE PROCESS AMONG ITS DIRECT BROTHERS--PARTICULARLY THE READY BIT OF A PROCESS IS LOCATED IN THE READY FIELD BY THIS INDEX.

F. I/O COUNT: INDICATES THE NUMBER OF I/O DEVICES ACTIVE WITHIN THE PROCESS BRANCH. THIS COUNT AT ANY PROCESS LEVEL IN THE HIERARCHY INDICATES THE NUMBER OF I/O OPERATIONS IN THE PROCESS SPACE.

PROCESS V STRUCTURE

THE PROCESS V STRUCTURE WILL BE USED WHEN A COMMUNICATION IS REQUIRED FROM IMP TO A PROCESS.

D, N, V, AND P STRUCTURES OF IMP

THERE MUST BE A SET OF THESE STRUCTURES FOR EVERY PROCESSOR IN THE SYSTEM TO BE USED WHEN THE PROCESSOR IS IN PRIVILEGED MODE.

RESOURCE STRUCTURES OF EACH PROCESSOR

THE RESOURCE STRUCTURES OF EACH PROCESSOR ARE USED TO DESCRIBE THE CURRENT ADDRESSING SPACE OF THE PROCESSOR.

DATA PECULIAR TO IMP

THE DATA PECULIAR TO IMP IS AS FOLLOWS:

A. A MAILBOX WHICH IS USED FOR PROCESSOR TO PROCESSOR COMMUNICATIONS (IOM OR CPU).

B. A PRECEDENCE MATRIX WHICH DEFINES THE RELATIVE POSITION OF THE PROCESSORS IN THE TREE. (PREORDER).

IMP FUNCTIONS

ACTIVATION AND DEACTIVATION OF IMP TAKE PLACE AS FOLLOWS. IMP IS ENTERED IN TWO DIFFERENT WAYS.

A. IMP IS CALLED FROM A PROCESS.

IN ORDER TO PERFORM A FUNCTION WHICH CAN ONLY BE DONE IN PRIVILEGED MODE (NONINTERRUPTABLE AND NONFINAL COMBINED) A PROCESS MAY CALL IMP. THE PARAMETER OF THE CALL IS PLACED ON THE TOP OF THE V STRUCTURE BEFORE THE IMP CALL.

B. AUTOMATIC IMP CALL.

THE OCCURRENCE OF AN EXTERNAL FAULT OR ANY INTERRUPT CAUSES AN AUTOMATIC IMP CALL. THE STATUS IS FOUND BY IMP ON THE TOP OF ITS V STRUCTURE.

WITH THE AUTOMATIC OR SOFT CALL OF IMP, THE STATE OF THE PROCESS IS STORED IN A SPECIAL AREA OF THE PROCESS-S SPACE (COROUTINE CONTROL FIELD) AND THE ENVIRONMENT OF IMP IS LOADED INTO THE PROCESSOR. UPON IMP RETURN THE REVERSE PROCEDURE IS PERFORMED.

THE IMP FUNCTIONS BASICALLY AID IN THE PERFORMANCE OF THREE FUNCTIONS: SCHEDULING, DISPATCHING, AND AUTOMATIC FAULT AND INTERRUPT DETECTION. THESE FUNCTIONS REQUIRE SOME PARAMETERS AND AN IMP CALL. THE IMP FUNCTIONS ARE:

A. SCHEDULING FUNCTIONS.

1. PROCESS CALL.

2. PROCESS RETURN.

3. PREEMPT.

B. DISPATCHING FUNCTION - QUEUE-IN TO PARENT.

C. REQUEST AN ASYNCHRONOUS MOVE (LEVEL-2 OR LEVEL-3).

D. AUTOMATIC FUNCTIONS.

1. EXTERNAL FAULT DISPATCHING.

2. INTERPROCESSOR COMMUNICATION.

3. I/O COMPLETE.

SCHEDULING FUNCTIONS

THE SCHEDULING FUNCTIONS ARE THE MOST COMPLEX IMP FUNCTIONS.

PROCESS CALL FUNCTION

PROCESS CALL ESTABLISHES A NEW PROCESS IN THE HIERARCHY BY THE ADDITION OF A NEW SET OF RESOURCE DEFINITIONS TO THE PROCESS RESOURCE STRUCTURE AND THE ACTIVATION OF THE NEW PROCESS. PARAMETERS FOR PROCESS CALL INCLUDE SEGMENT SPACE DEFINITION, TIMER SETTING, FAULT MASKS, AND LEVEL-3 RESOURCES. THESE PARAMETERS HAVE BEEN ENTERED INTO THE PROCESS RESOURCE STRUCTURE WITH THE PROCESS PARAMETER OPERATORS. THE CALLER CAN PROVIDE A MONOPROCESSOR PARAMETER REQUIRING THE NUMBER OF PROCESSORS WORKING IN THE SUBPROCESS-S SPACE TO BE LIMITED TO ONE.

THE PROCESS CALL FUNCTION MUST PERFORM THE FOLLOWING ACTIONS IN ORDER TO START A SUBPROCESS:

A. THE PROCEDENCE MATRIX IS TESTED AND LOCKED.

B. IF THE SUBPROCESS IS ACTIVE OR THE MONOPROCESSOR BIT IS SET, A STATUS INDICATING THAT FACT IS LEFT AT THE TOP OF THE CALLING PROCESS-S V STRUCTURE, THE MAILBOX IS UNLOCKED AND AN IMP RETURN IS EXECUTED.

C. TWO CHECKS ARE PERFORMED: FIRST THE TIME GIVEN TO THE SUBPROCESS IS COMBINED IN ORDER TO INSURE A PROPER NESTING WITHIN THE CALLER-S REMAINING PROCESSING TIME, SECONDLY IMP INSURES THAT THE CALLER-S PROCESS INFORMATION IS NOT PASSED WITH WRITE PERMISSION TO THE SUBPROCESS. IF ANY OF THESE CONDITIONS FAIL - THE PROCESSOR PERFORMS AN IMP RETURN IN THE SAME WAY AS IN B.

D. THE READY BIT OF THE SUBPROCESS IS RESET.

E. IF THE CALLING PROCESS NEEDS MORE COMPUTATION TIME, THAT IS, THE QUEUE IS NOT EMPTY OR ONE OF THE SUBPROCESSES IS READY, THIS FACT IS COMMUNICATED ROOTWARD VIA THE READY BIT.

F. THE ACTIVE BIT OF THE CALLER IS RESET.

G. THE PRECEDENCE MATRIX IS UPDATED.

H. THE ACTIVE BIT OF THE SUBPROCESS IS SET.

I. THE HARDWARE PROCESS CALL IS EXECUTED SLICING THE RESOURCE STACK AND UPDATING THE RESOURCE DISPLAY.

J. THE PRECEDENCE MATRIX IS UNLOCKED.

PROCESS RETURN FUNCTION

THE PROCESS RETURN FUNCTION RETURNS A PROCESSOR TO A PARENT PROCESS.

A. THE PRECEDENCE MATRIX IS FIRST TESTED AND LOCKED.

B. THE MONOPROCESSOR BIT AND THE ACTIVE BIT OF THE CALLER ARE RESET.

C. THE PRECEDENCE MATRIX IS UPDATED.

D. IF THE PARENT PROCESS IS ACTIVE, REFER TO STEP K ELSE

E. THE ACTIVE BIT OF THE PARENT IS SET.

F. THE READY BIT OF THE PARENT IS RESET AND IF THERE IS NO NEED FOR A PROCESSOR ALONG THE PATH ROOTWARD, THE READY BITS DOWN THE TREE ARE RESET.

G. THE TIME COUNTER OF THE PROCESSOR FOR THE PARENT IS COMBINED WITH THE REMAINING TIME OF THE CALLER.

H. THE HARDWARE PROCESS RETURN IS EXECUTED.

I. THE PRECEDENCE MATRIX IS UNLOCKED.

J. IMP RETURN IS EXECUTED.

K. IF THE QUEUE OF THE PARENT IS NOT LOCKED OUT A MESSAGE IS LEFT IN IT.

L. A HARDWARE PROCESS RETURN IS EXECUTED.

M. A PROCESS RETURN IS TRIED AT THAT NEW LEVEL, I.E., REFER TO C FOR THE FOLLOWING.

WHEN A PROCESS RETURN IS EXECUTED BY A DIRECT SUBPROCESS OF THE GLOBAL, AND IF AT THAT TIME THE GLOBAL PROCESS IS ACTIVE, THE PROCESSOR GOES TO ITS RESPECTIVE DOGHOUSE. IT WILL REMAIN HERE UNTIL GLOBAL BECOMES INACTIVE, AT WHICH TIME IT WILL ENTER THE GLOBAL PROCESS.

PREEMPT FUNCTION

THE PREEMPT FUNCTION IS USED BY A PROCESS TO FORCE ALL THE INTERPRETERS WORKING IN THE SPACE OF ONE OF ITS SON PROCESSES TO RETURN AT A LEVEL BELOW IT.

IMP DETERMINES THE PROCESSORS WHICH ARE IN THE SUBPROCESS SPACE WITH THE PRECEDENCE MATRIX AND INTERRUPTS THEM LEAVING A MESSAGE FOR EACH IN THEIR MAILBOX. UPON INTERRUPTION EACH PROCESSOR LOOKS AT THE MAILBOX AND "PROCESS RETURNS" AT LEAST TO THE REQUESTED LEVEL.

DISPATCHING FUNCTION -- QUEUE-IN TO PARENT

THE DISPATCHING FUNCTION PASSES INFORMATION FROM A SUBPROCESS TO ITS PARENT BY MEANS OF A PROCESS QUEUE ASSOCIATED WITH EACH PROCESS SPACE. EACH PROCESS MAY READ ITS OWN QUEUE WITHOUT REQUESTING IMP. IT IS ALSO ABLE TO QUEUE A MESSAGE IN THE QUEUE OF ONE OF ITS SUBPROCESSES.

THE IMP QUEUE IN FUNCTION QUEUES A MESSAGE IN THE PARENT-S QUEUE AND OPTIONALLY EXECUTES A PROCESS RETURN. THE FOLLOWING STEPS ARE EXECUTED:

A. THE QUEUE LOCK BIT IS TESTED. IF SET, AN IMP RETURN IS EXECUTED, AND A STATUS IS LEFT AT THE TOP OF THE CALLER-S V STRUCTURE. IF IT WAS RESET, IMP SETS IT AND THE FOLLOWING OCCURS.

B. THE DESCRIPTOR OF THE PARENT-S QUEUE IS EXECUTED IN ENTER MODE AND THE MESSAGE IS LEFT IN THAT NEW ENTRY - IF THAT ENTRY WAS THE FIRST ONE AND THE PARENT WAS INACTIVE IT IS MARKED READY - UPON OCCURRENCE OF AN OUT OF BOUNDS FAULT AN IMP RETURN IS EXECUTED WITH A STATUS LEFT AT THE TOP OF THE CALLER-S V STRUCTURE.

C. IF PROCESS RETURN IS INDICATED IN THE REQUEST, IT WILL BE PERFORMED AS SPECIFIED PREVIOUSLY.

D. IMP RETURN IS EXECUTED.

REQUEST AN ASYNCHRONOUS MOVE (L2 OR L3)

ANY PROCESS CAN REQUEST IMP TO INITIATE AN I/O OPERATION THROUGH THE IMP CALL. BEFORE INITIATING THE OPERATION IMP WILL INSURE THAT THE DEVICE IN QUESTION IS WITHIN THE ENVIRONMENT OF THE REQUESTOR. IF IT IS NOT, AN OUT OF BOUNDS FAULT CONDITION WILL EXIST.

AUTOMATIC FUNCTIONS

IF AN EXTERNAL FAULT, LOCAL EXTERNAL FAULT, OR AN INTERRUPT IS DETECTED DURING THE RUNNING OF A PROCESS, AN AUTOMATIC ENTRY INTO IMP OCCURS.

WHEN A LOCAL EXTERNAL FAULT OCCURS, AN ENTRY WILL BE MADE IN THE REQUEST QUEUE OF THE PARENT PROCESS SPACE TO SERVICE THIS FAULT. IMP WILL THEN PERFORM A PROCESS RETURN.

WHEN AN EXTERNAL FAULT OCCURS, IMP MUST GO ROOTWARD LOOKING FOR THE PROCESS WHICH OWNS THAT FAULT, THE FAULTY PROCESS IS SET ACTIVE AND A MESSAGE IS LEFT ALONG THE LINE IN EACH NODE-S QUEUE UNTIL THE OWNER OF THE FAULT IS REACHED. AN ULTIMATE MESSAGE WITH A STATUS OF THE FAULT IS ENTERED INTO ITS QUEUE. IF THE NODE IS ACTIVE A PROCESS RETURN IS EXECUTED. OTHERWISE, THAT NODE WILL BE ACTIVATED BY AN IMP RETURN.

ALL INTERRUPTS ARE HANDLED BY IMP. THIS INCLUDES PROCESSOR INTERCOMMUNICATION INTERRUPTS AND I/O COMPLETES.

AN I/O OR MEC COMPLETE INTERRUPT WILL BORROW A PROCESSOR IN ORDER TO INSERT AN I/O COMPLETE MESSAGE INTO THE SPECIFIED PROCESS QUEUE. IF THE PROCESS IS INACTIVE AND ITS QUEUE IS EMPTY IT WILL PROPAGATE THE NEED FOR A PROCESSOR DOWN THE LINE VIA THE READY BITS.

SECTION 5

STORAGE

INTRODUCTION

THE STORAGE FACILITIES OF THE B8500 SYSTEM ARE ORGANIZED INTO THREE LEVELS. THESE LEVELS ARE DIFFERENTIATED ACCORDING TO SPEED, COST, AND SIZE AND ARE STRUCTURED HIERARCHICALLY FROM THE LEVEL PERMITTING FASTEST ACCESS TO THE ONE WITH SLOWEST ACCESS. IN THE ORDER OF ACCESS SPEED, THE LEVELS OF STORAGE ARE AS FOLLOWS: LEVEL-1 STORAGE (MAIN STORAGE), LEVEL-2 STORAGE (MEMORY EXTENSION DEVICES), AND LEVEL-3 STORAGE (AN ARRAY OF PERIPHERAL EQUIPMENT). THE HIERARCHY IS TREATED AS A DISCRETE SET OF CONTINUUMS.

LEVEL-1 STORAGE

LEVEL-1 IS A FIELD-DEFINED, PHASED STORAGE CONSISTING OF FIELD ISOLATION UNITS (FIU) AND MEMORY STORAGE UNITS (MSU). AN FIU CONTROLS EITHER ONE OR TWO MSU(S), EACH OF WHICH CONTAINS 8192 256-BIT WORDS (SEE FIGURE 5-1). LEVEL-1 MAY EXPAND FROM ONE FIU TO A MAXIMUM OF 16 FIU(S). STORAGE CAPACITY IS EXTENDED BY ADDING ONE MSU AT A TIME.

LEVEL-1 STORAGE IS CONSIDERED LOGICALLY TO BE A CONTINUUM OF BITS. FIELDS MAY BE DEFINED BY DECLARING THE LOCATION OF THE STARTING BIT AND THE NUMBER OF BITS COMPOSING THE FIELD. THE FIELD ISOLATION UNITS (FIU) OF THE STORAGE SYSTEM PROVIDE THE CAPABILITY OF EXTRACTING OR INSERTING FIELDS OF INFORMATION INDEPENDENT OF THE STRUCTURE OF THE PHYSICAL STORAGE DEVICES.

PHASING IS A TECHNIQUE THAT ALLOWS THE FIELD ISOLATION UNIT TO INITIATE MEMORY STORAGE UNIT CYCLES IN PARALLEL WITH DATA TRANSFER. IN THIS WAY THE DATA TRANSFER RATE IS MAXIMIZED.

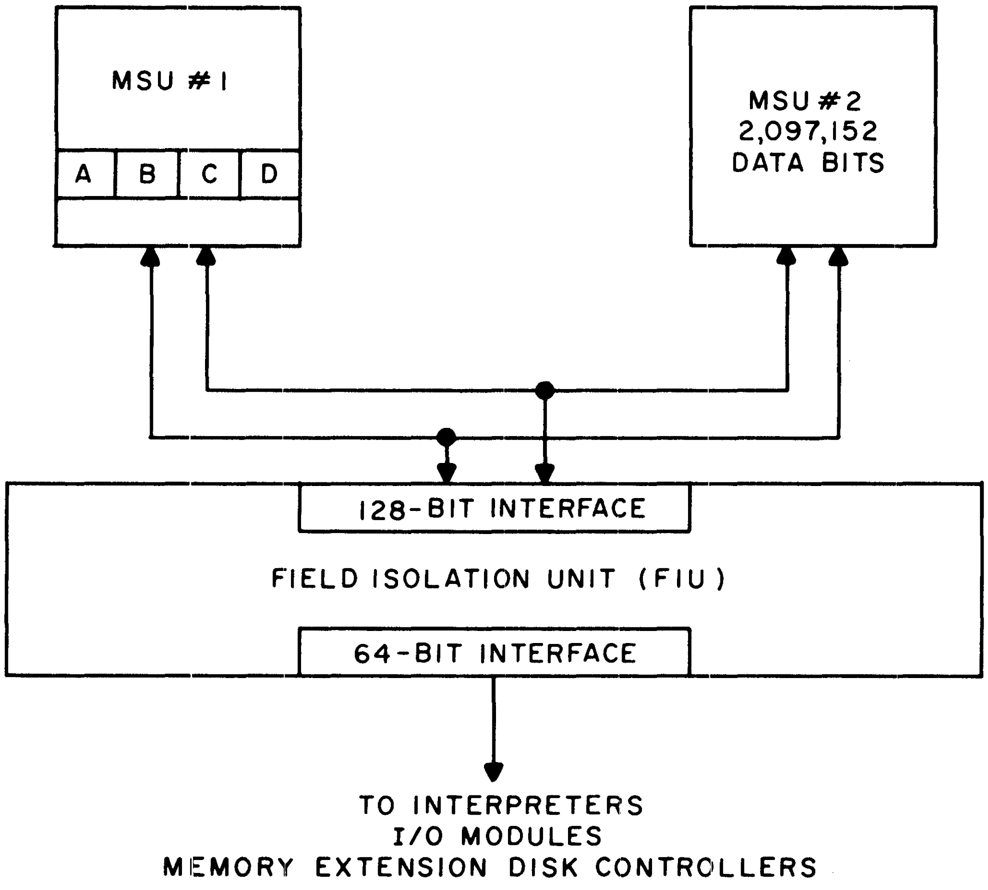


FIGURE 5-1. LEVEL-1 STORAGE

ERROR DETECTION

WHEN A GROUP OF BITS IS MOVED FROM THE MSU TO THE FIU, THESE BITS ARE CHECKED. PARITY IS CHECKED ON EACH 64-BIT GROUP RECEIVED BY THE FIU FROM THE ACCESSING UNIT AND IS ESTABLISHED ON EACH GROUP OF 64 BITS TRANSMITTED BY THE FIU. WHEN A DATA ERROR IS DETECTED BY THE FIU, THE CONTROL WORD IN FORCE IS PLACED IN THE MEMORY FAIL REGISTER WITH AN IDENTIFICATION OF THE ERROR, AND A FAULT INDICATOR IS PASSED TO THE UNIT THAT REQUESTED THE DATA MOVEMENT.

ERROR INDICATIONS ARE PLACED IN THE FAIL REGISTER TO INDICATE THE TYPE AND PLACEMENT OF THE ERROR. THE ERRORS ARE:

A. CONTROL WORD PARITY ERROR.

B. ILLEGAL OPERATION CODE.

C. CONTROL WORD RECEIVED BY THE WRONG FIU.

D. DATA ERROR ON READ OPERATION.

E. DATA ERROR ON WRITE OPERATION.

F. FIU BOUNDARY ERROR.

G. REQUESTING DEVICE TOOK TOO LONG TO SEND DATA.

ERROR DETECTION IN LEVEL-1 STORAGE USES A TECHNIQUE THAT ALLOWS DETECTION OF ALL ERRORS INVOLVING TWO BITS, AND ALLOWS THE HARDWARE OF THE FIU TO CORRECT ALL SINGLE-BIT ERRORS. CHECK BITS ARE AT THE END OF EACH 64-BIT DATA FIELD FOR PARITY CHECKING AND ERROR CORRECTIONS. THIS PERMITS THE SYSTEMS PROGRAMMER TO MORE READILY ACCESS AND MANIPULATE DATA WHICH IS KNOWN TO HAVE BEEN CORRECTED. THE PRESENCE OF A 64-BIT WORD ALLOWS A GREATER NUMBER OF BITS TO BE CHECKED WITH ONLY A SMALL INCREASE IN LOGIC.

AT THE CONCLUSION OF AN ERROR CORRECTION A SIGNAL IS PROVIDED TO THE REQUESTOR TO INDICATE THAT A CORRECTION HAS BEEN MADE. SIMPLE PARITY IS USED BY THE FIU ON ALL INTERMODULE DATA TRANSMISSIONS.

FIU OPERATIONS

OPERATIONS THAT MAY BE PERFORMED IN CONJUNCTION WITH LEVEL-1 STORAGE ARE:

A. FETCH: ALLOWS AN EXTERNAL UNIT TO MOVE DATA FROM LEVEL-1.

B. STORE: ALLOWS AN EXTERNAL UNIT TO MOVE INFORMATION TO LEVEL-1.

C. FETCH MEMORY FAIL REGISTER: ALLOWS AN EXTERNAL UNIT TO OBTAIN THE MEMORY FAIL REGISTER FROM A SPECIFIC FIU. THE FAIL REGISTER IS CLEARED FOLLOWING THE FETCH.

D. FETCH AND LOCK: ALLOWS AN EXTERNAL UNIT TO FETCH DATA AND SET THE MOST SIGNIFICANT BIT TO INDICATE THAT THE FIELD HAS BEEN LOCKED.

LEVEL-2 STORAGE

LEVEL-2 STORAGE IS FIELD-DEFINED STORAGE OPERATING AS A LOGICAL LEVEL BETWEEN LEVEL-1 AND LEVEL-3 IN THE STORAGE HIERARCHY. IT IS USED TO MAINTAIN PROGRAM AND DATA SEGMENTS WHICH ARE NOT BEING CHANGED BY A PROCESSING ELEMENT OF THE SYSTEM AT A GIVEN MOMENT.

WITH THE EXCEPTION OF ACCESS AND TRANSFER SPEED, LEVEL-2 STORAGE IS TREATED AND MANAGED THE SAME AS LEVEL-1. BECAUSE LEVEL-2 STORAGE HAS CHARACTERISTICS SIMILAR TO THOSE OF LEVEL-1 STORAGE, THE SAME MANAGEMENT ALGORITHMS MAY BE APPLIED TO THE RESPECTIVE BIT CONTINUUMS.

LEVEL-2 STORAGE IS COMPRISED OF A RANGE OF DEVICES, INCLUDING HEAD-PER-TRACK DISK AND BULK CORE. A MEMORY EXTENSION CONTROLLER (MEC) PROVIDES INTERFACES BETWEEN LEVEL-1 STORAGE AND THE LEVEL-2 MEDIA, AND ALSO PROVIDES THE FIELD-ISOLATION FACILITIES. THE MEC HAS THE CAPABILITY TO SERVICE MULTIPLE REQUESTS THRU MULTIPLE ASYNCHRONOUS DATA PATHS. THE MEMORY EXTENSION CONTROLLER UTILIZES THE SAME QUEUE AND ELEMENT FORMATS AND QUEUEING DISCIPLINES AS THE I/O MODULE (IOM). THIS PERMITS THE IOM AND THE MEC TO ENTER REQUESTS INTO EACH OTHERS QUEUES IN ORDER TO ACHIEVE TRANSFERS OF FIELDS BETWEEN LEVEL-1, LEVEL-2, AND LEVEL-3 STORAGE WITHOUT PROCESSOR INTERVENTION.

SINCE ONLY A SMALL PART OF THE TOTAL ACTIVE PROCESS SPACE IS ACTUALLY BEING OPERATED UPON BY THE PROCESSING ELEMENTS OF THE SYSTEM AT ANY GIVEN MOMENT, A RELATIVELY SMALL SPACE IS REQUIRED IN DIRECT-ACCESS STORAGE (LEVEL-1 AND REGISTERS). THUS, THE BULK OF THE PROCESS SPACE MAY BE MAINTAINED IN LEVEL-2 STORAGE WHILE IT IS NOT BEING CHANGED. BY HAVING A HIERARCHY OF STORAGE WITH IDENTICAL MANAGEMENT, "FIX-UP" IS ELIMINATED WHEN DATA IS MOVED FROM LEVEL TO LEVEL, THUS ACHIEVING EFFICIENT UTILIZATION OF THE STORAGE HIERARCHY. THE INTEGRATION OF LEVEL-2 STORAGE INTO THE SYSTEM PROVIDES FOR EFFICIENT UTILIZATION OF SYSTEM STORAGE RESOURCES. IT PROVIDES FOR A NATURAL FLOW OF INFORMATION FROM FILE STORAGE TO REGISTERS, WITH INTERMEDIATE BUFFERING IN A RANGE OF DEVICES, THUS YIELDING A LOW COST-PER-BIT PER PROCESSING CYCLE.

MEMORY EXTENSION CONTROLLER

THE MEC EXERCISES CONTROL OVER THESE LEVEL-1 AND LEVEL-2 OPERATIONS:

A. THE SELECTION OF REQUESTS FROM THE LEVEL-2 DEVICE QUEUE.

B. THE INITIATION OF REQUESTS ON THE APPROPRIATE LEVEL-2 DEVICE.

C. THE MOVEMENT OF INFORMATION BETWEEN LEVEL-1 AND LEVEL-2 STORAGE.

D. ACTING ON REQUESTS TO CONTROL THE MOVEMENT OF DATA BETWEEN LEVEL-3 AND LEVEL-2 VIA LEVEL-1 STORAGE.

E. THE REPORTING OF STATUS BACK TO THE REQUESTING PROCESS.

THE MEC ACTS AS A PROCESSOR WORKING ON THE PROCESS SPACES OF THE PROCESS HIERARCHY. ALL ACCESSES TO LEVEL-1 AND LEVEL-2 STORAGE ARE PROTECTED BY FINAL COMBINE OPERATIONS (PERFORMED PRIOR TO THE INITIATION OF THE REQUEST).

TO ALLOW THE MEC(S) TO PROPERLY SELECT PATHS AND SERVICE THE LEVEL-2 DEVICE QUEUE, A MAP OF THE LEVEL-2 SUBSYSTEM CONFIGURATION IS CONSTRUCTED. THIS MAP ALLOWS EACH MEC TO BE AWARE OF THE DEVICES IT CAN SERVICE,THE PATHS AVAILABLE FOR THAT SERVICE, AND THE LOCATION OF THE LEVEL-2 QUEUE THAT REQUIRES SERVICE. THE MEC PATH SELECTION AND QUEUE SERVICE OPERATES FROM THE LEVEL-2 SUBSYSTEM MAP. THE BASIC LEVEL-2 MAP IS LOADED AT INITIALIZE TIME AND CAN BE MODIFIED DYNAMICALLY BY A GLOBAL PROCESS. THE LEVEL-2 MAP RESIDES IN LEVEL-1 STORAGE, AND MAY BE ACCESSED BY MEC(S), IOM(S), IMP, AND GLOBAL PROCESSES.

REQUESTS FOR LEVEL-2 ACTIVITY ARE ENTERED INTO THE DEVICE QUEUE VIA THE DEVICE VECTOR; THE SERVICING OF REQUESTS IS DONE BY MEC(S) USING THE LEVEL-2 MAP. ALL QUEUEING IS AT THE DEVICE LEVEL. THE PATH SELECTION AND QUEUE SERVICE PROVIDES MAXIMUM FLOW OF TRAFFIC BY USING ALL AVAILABLE ACCESS PATHS.

DEVICE QUEUE ELEMENT

A DEVICE IS CONSIDERED TO BE A PERIPHERAL DEVICE WHICH IS CONNECTED TO A CHANNEL OF THE MEC. THERE IS ONE DEVICE QUEUE ELEMENT (DQE), AND THEREFORE ONE QUEUE, FOR EACH ADDRESS CONTINUUM. THE CONTENTS OF EACH DEVICE QUEUE ELEMENT ARE DESCRIBED IN THE IOM DISCUSSION IN SECTION 6.

EACH MEC HAS ITS OWN COMPLETION QUEUE. THE LOCATION OF THE HEAD OF THE QUEUE IS RETAINED BY THE MEC. THE HEAD CONTAINS A COUNT OF

ELEMENTS, AND POINTERS TO THE FIRST AND LAST ENTRIES OF THE QUEUE. THE ELEMENTS OF THE COMPLETION QUEUES ARE IDENTICAL WITH THE ELEMENTS OF THE DEVICE QUEUES EXCEPT THAT THE COMPLETION STATUS IS PLACED IN THE REQUEST STATUS FIELD. EACH LEVEL-2 ELEMENT OF THE DEVICE VECTOR CONTAINS BOTH A POINTER TO THE CORRESPONDING DEVICE ELEMENT IN THE LEVEL-2 MAP, AND THE IDENTIFYING NUMBER OF THE MEC WHICH CAN SERVICE THAT DEVICE.

REQUESTING A LEVEL-2 OPERATION

WHEN A PROCESS DESIRES A MOVE BETWEEN LEVEL-1 AND LEVEL-2 STORAGE, A REQUEST IS PREPARED WHICH CONTAINS A DESCRIPTION OF THE LEVEL-1 AREA, AN OPERATION CODE, A DESCRIPTION OF A LEVEL-2 ADDRESS, AND A BIT TO INDICATE WHETHER THE PROCESS REQUIRES NOTIFICATION OF COMPLETION OF THE LEVEL-2 OPERATION.

THE CONSTRUCTION OF THE REQUEST INVOLVES AN ALLOCATION OF STORAGE FOR THE REQUEST, PLACEMENT OF THE ABSOLUTE LEVEL-1 ADDRESS IN THE REQUEST, AND THE CREATION OF A PROCESS IDENTIFICATION TO BE PLACED IN THE REQUEST. THE COMPLETED REQUEST IS THEN LINKED INTO THE END OF THE DEVICE QUEUE. THE MEC, WHICH HAS PATHS TO THE DEVICE AND IS IDENTIFIED IN THE DEVICE VECTOR, IS NOTIFIED VIA THE INTERRUPT BUS.

EACH MEC HAS AN INTERNAL REGISTER THAT HOLDS THE NUMBER OF JOBS REMAINING TO BE INITIATED BY AN MEC. THE COUNT OF THE REGISTER IS INCREASED EACH TIME THE SOFTWARE SENDS AN INTERRUPT FOR JOB INITIATION AND IS DECREASED EACH TIME AN MEC ACCEPTS A JOB. EACH MEC COMMUNICATES WITH THE OTHER MEC WHEN A JOB IS ACCEPTED. THEREFORE, THE NUMBER OF JOBS REMAINING IN THE COMMON JOB QUEUE IS MAINTAINED BY EACH MEC.

SINCE THE MAPS, QUEUES, AND DEVICE QUEUE ELEMENTS FOR THE MEC AND IOM HAVE IDENTICAL FORMATS, REQUESTS FOR A SEQUENCE OF OPERATIONS INVOLVING BOTH LEVEL-2 AND LEVEL-3 MAY BE LINKED UNDER A REQUEST. THIS CAPABILITY PERMITS A SERIES OF INTERLEVEL AND INTRALEVEL MOVES TO BE PERFORMED WITHOUT CPU INTERVENTION.

LEVEL-3 STORAGE

LEVEL-3 STORAGE IS IMPLEMENTED WITH A WIDE RANGE OF PERIPHERAL DEVICE, SUCH AS MAGNETIC TAPES, FILE DISKS, PUNCHED CARD EQUIPMENT, PRINTERS, AND DATA COMMUNICATIONS EQUIPMENT. ACCESS TO LEVEL-3 STORAGE IS MADE THROUGH THE INPUT-OUTPUT MODULE (SEE SECTION 6).

RELATED HARDWARE SPECIFICATIONS

DISK QUEUER	CP 1760-1220
FIELD ISOLATION UNIT	CP 1720-5576
MEMORY EXTENSION CONTROLLER	CP 1720-5584
MEMORY STORAGE UNIT	CP 1762-3182
PARALLEL DISK CONTROLLER	CP 1760-4454

SECTION 6

INPUT/OUTPUT SUBSYSTEM

INTRODUCTION

THE INPUT/OUTPUT SUBSYSTEM DESCRIBED IN THIS SPECIFICATION IS DESIGNED TO PROVIDE A HIGH DEGREE OF:

PARALLELISM BETWEEN INTERPRETERS AND I/O MODULES (IOM) BY RELEASING THE INTERPRETER AT THE EARLIEST POSSIBLE MOMENT FROM THE I/O OPERATION.

THROUGHPUT BY DELAYING THE BINDING OF PATHS THROUGH THE I/O SUBSYSTEM UNTIL INITIATE TIME, THUS MAXIMIZING CHANNEL ACTIVITY.

THE IOM ACTS AS A PROCESSOR WORKING ON THE DEFINED PROCESS SPACES OF THE PROCESS HIERARCHY. ALL ACCESSES TO LEVEL-1 AND LEVEL-3 STORAGE ARE PROTECTED BY FINAL COMBINE OPERATIONS (PERFORMED BY IMP PRIOR TO THE INITIATION OF THE IOM ON THE REQUEST).

THE INPUT/OUTPUT MODULE EXERCISES CONTROL OVER:

A. SELECTION OF I/O REQUESTS FROM DEVICE QUEUES.

B. SELECTION OF THE OPTIMUM PATH TO THE DEVICE.

C. INITIATION OF REQUESTS ON THE APPROPRIATE DEVICE.

D. MOVEMENT OF DATA BETWEEN LEVEL-1 AND LEVEL-3.

E. QUEUEING REQUESTS TO OTHER DEVICE QUEUES OR MEC QUEUES.

CONCEPTUAL VIEW OF THE I/O SUBSYSTEM

THE I/O SUBSYSTEM IS MADE UP OF I/O MODULES, PERIPHERAL DEVICE CONTROLLERS, EXCHANGES, AND PERIPHERAL DEVICES. THE SUBSYSTEM MAY BE INTERCONNECTED INTO THE CONFIGURATION MOST SUITED TO THE USERS REQUIREMENTS. FIGURE 6-1 ILLUSTRATES AT LEAST ONE EXAMPLE OF EACH

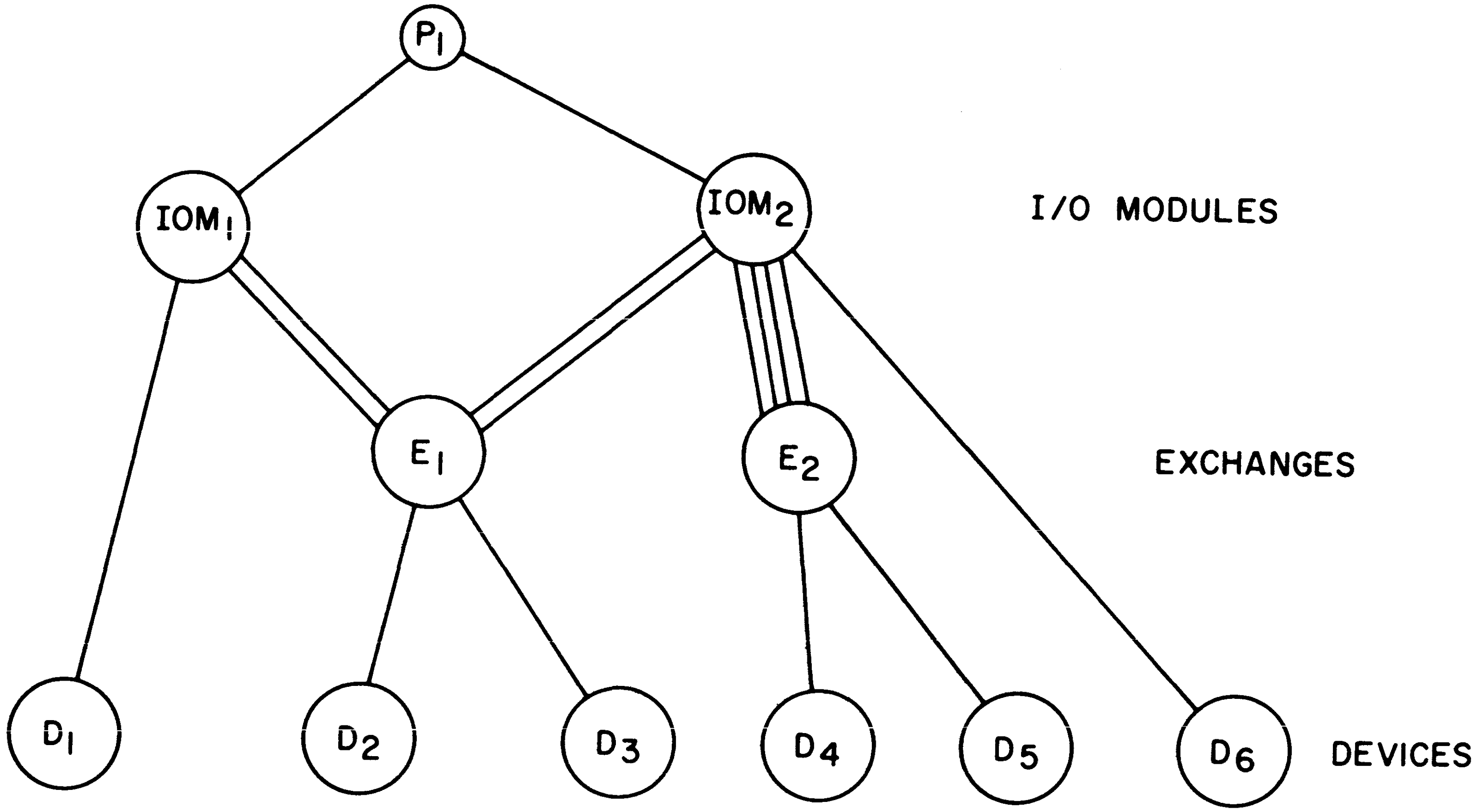


FIGURE 6-1. GENERAL I/O SUBSYSTEM CONFIGURATION

POSSIBLE TYPE OF CONNECTION THAT CAN BE MADE BETWEEN MODULES OF THE I/O SYSTEM. IT CAN BE SEEN THAT I/O MODULES CAN BE CONNECTED TO EXCHANGES, SHARED EXCHANGES, OR DIRECTLY TO THE DEVICES. THE DEVICE CONTROLLERS ARE NOT ILLUSTRATED IN THIS DIAGRAM BECAUSE THEY DO NOT INFLUENCE THE DESIGN CONCEPTS OF THE I/O SUBSYSTEM AND MAY BE THOUGHT OF AS IOM EXTENSIONS. A SHARED EXCHANGE IS AN EXCHANGE WHICH IS SHARED BY MORE THAN ONE IOM.

THROUGHPUT CAN BE MAXIMIZED IF THE BINDING OF A COMPLETE PATH BETWEEN PROCESS AND DEVICE IS DELAYED UNTIL THE DEVICE IS READY TO ACCEPT THE I/O JOB. AS AN EXAMPLE, IF DEVICE D2 IS TO BE INITIATED, THE PATH REQUIRED TO CONNECT THE INITIATING PROCESS P1 WITH D2 INVOLVES A CHOICE BETWEEN TWO I/O MODULES AND THEN BETWEEN TWO CHANNELS WITHIN A GIVEN IOM TO THE EXCHANGE E1. IF THE PATH HAD BEEN PRESELECTED, THAT IS SELECTED PRIOR TO THE TIME THE DEVICE IS INITIATED, A SITUATION CAN DEVELOP IN WHICH THE DEVICE IS FREE BUT THE PRE-SELECTED PATH TO THE DEVICE IS NOT, THUS DELAYING EXECUTION. THIS SITUATION IS FAR LESS LIKELY TO OCCUR IF MORE THAN ONE PATH EXISTS WHICH CAN BE CONNECTED BETWEEN THE PROCESS AND DEVICE.

I/O SUBSYSTEM MAP (IOSM)

TO ALLOW THE I/O MODULES TO PROPERLY SELECT PATHS TO THE DEVICES AND SERVICE I/O REQUESTS, A MAP OF THE I/O SUBSYSTEM CONFIGURATION IS CONSTRUCTED AT INITIALIZE TIME. I/O MODULES ARE AWARE OF THE DEVICES THEY CAN SERVICE AND THE PATHS AVAILABLE TO THE DEVICE FOR THAT SERVICE. THE I/O SUBSYSTEM MAP (IOSM) ALSO INDICATES DEVICE QUEUES WHICH HOLD SERVICE REQUESTS FOR ASSOCIATED DEVICES. THE IOSM RESIDES IN LEVEL-1 STORAGE, AND MAY BE ACCESSED BY IOMS, MECS, IMP AND THE GLOBAL PROCESS.

THE IOSM (SEE FIGURE 6-2) HAS:

A. A DEVICE VECTOR CONTAINING DEVICE VECTOR ELEMENTS.

B. DEVICE ELEMENTS.

C. INDIRECT EXCHANGE ELEMENTS.

D. AN EXCHANGE ELEMENT.

E. DEVICE QUEUES WHICH HOLD THE DEVICE QUEUE ELEMENTS.

F. COMPLETION QUEUES WHICH HOLD THE STATUS QUEUE ELEMENTS (SQE).

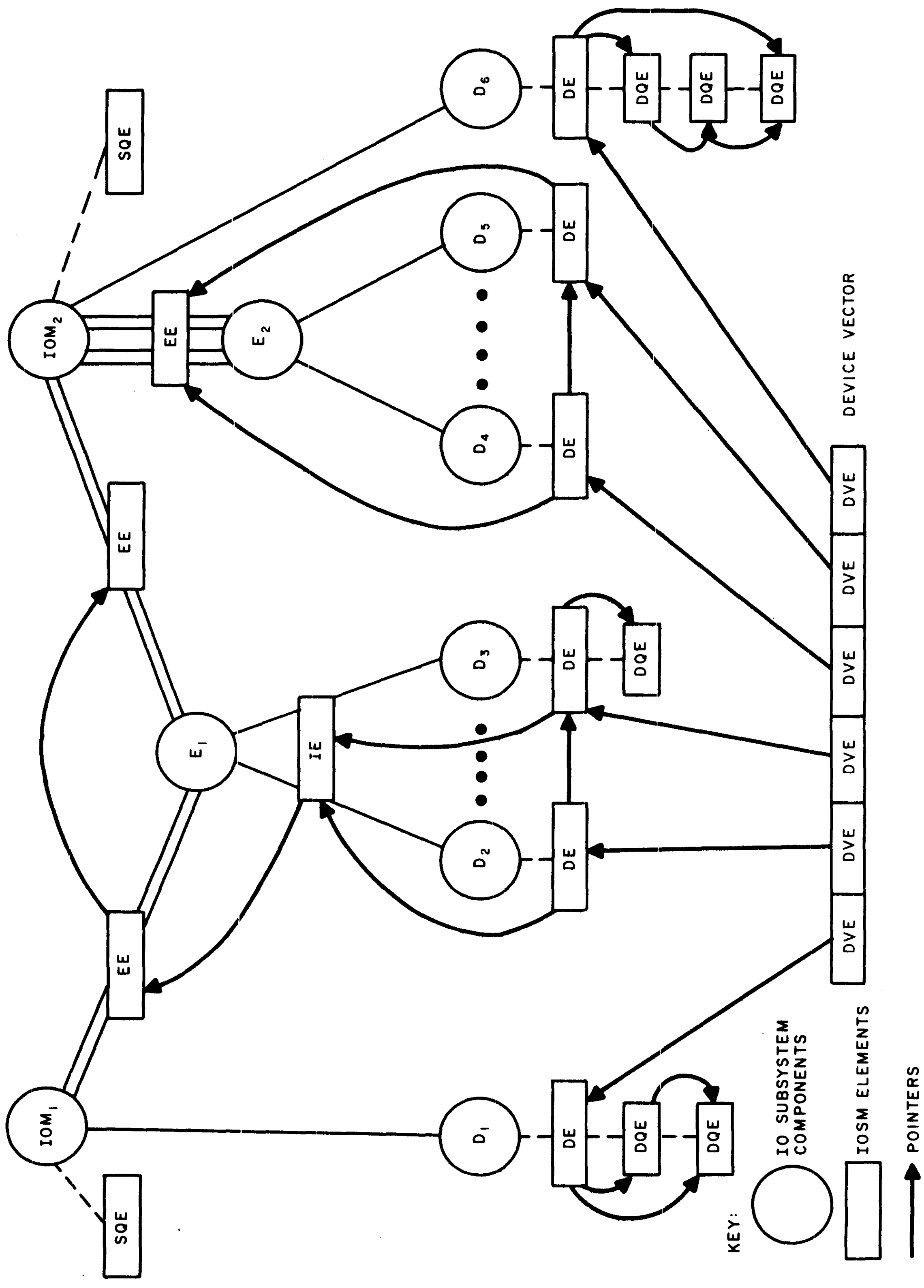


FIGURE 6-2. TYPICAL I/O SUBSYSTEM MAP CONFIGURATION

STATUS QUEUE ELEMENTS

THERE IS A STATUS QUEUE FOR EACH IOM. THE STATUS QUEUES CONTAIN STATUS QUEUE ELEMENTS (SQE) WHICH CONTAIN THE STATUS OF A COMPLETED OR TERMINATED I/O REQUEST. THE STATUS QUEUE ELEMENTS ARE IDENTICAL TO THE DEVICE QUEUE ELEMENTS WITH THESE EXCEPTIONS:

A. A RESULT DESCRIPTOR MAY BE LOADED INTO SPECIFIED BIT POSITIONS.

B. CERTAIN BIT POSITIONS MAY BE SET IF THE TERMINATION WAS DUE TO DETECTION OF AN IMP OR IOM ERROR CONDITION.

THE DEVICE STATUS QUEUES ENABLE THE INITIATING PROCESS TO EXAMINE THE TERMINATING STATUS OF ITS I/O REQUEST.

DEVICE VECTOR ELEMENTS (DVE)

THE DEVICE VECTOR IS AN ARRAY CONTAINING DEVICE VECTOR ELEMENTS (DVE). THERE IS ONE DVE FOR EACH DEVICE. THE DVE CONTAINS A POINTER TO ITS ASSOCIATED DEVICE ELEMENT, AND A LIST OF I/O MODULES THAT CAN SERVICE THE DEVICE ASSOCIATED WITH THE DVE. THE DEVICE VECTOR ELEMENTS ARE USED SOLELY BY IMP TO FORMAT AND LIST THE I/O REQUESTS IN THE PROPER DEVICE QUEUE.

DEVICE ELEMENT (DE)

A DEVICE ELEMENT (DE) IS ALSO ASSOCIATED WITH EACH DEVICE. THE DE CONTAINS THE STATUS OF THE DEVICE, AND POINTS TO THE FIRST AND LAST DEVICE QUEUE ELEMENT IN THE DEVICE QUEUE. THE VARIOUS FIELDS ARE:

LOCK (L) BIT	INDICATES WHEN PRESENT, THAT THE DE OR ITS QUEUED DEVICE QUEUE ELEMENTS ARE BEING OPERATED ON BY EITHER A IOM OR IMP, AND PREVENTS MORE THAN ONE OF THESE OPERATORS FROM TRYING TO ACCESS AND OPERATE ON THE DE OR DQE AT THE SAME TIME.
ELEMENT TYPE (ET) FIELD	IDENTIFIES THE ELEMENT AS BEING A DE (CODE 001).
BUSY (B) BIT	INDICATES THE STATUS OF THE ASSOCIATED DEVICE.

I/O ERROR (IOE) BIT	INDICATES THAT THE ASSOCIATED I/O CHANNEL HAS BEEN TERMINATED AS A RESULT OF AN ERROR CONDITION BEING DETECTED. ALSO CAUSES THE CONTENTS OF THE DEVICE QUEUE TO BE TRANSFERRED INTO THE STATUS QUEUE.
DE RING LINK (RL) FIELD	IF APPLICABLE, THIS FIELD WILL CONTAIN A POINTER TO A DE FOR A DEVICE WHICH IS CONNECTED ON THE SAME EXCHANGE. (I.E. THE DE FOR DEVICE D2 IN FIGURE 6-2 WOULD CONTAIN A POINTER TO THE DE FOR DEVICE D3.)
REQUEST COUNT (RC) FIELD	MAINTAINS A RECORD OF THE NUMBER OF I/O REQUESTS (DEVICE QUEUE ELEMENTS) QUEUED FOR ITS ASSOCIATED DEVICE.
END RING (ER) BIT	IF APPLICABLE, INDICATES THAT ITS ASSOCIATED DEVICE IS THE LAST DEVICE ON AN EXCHANGE.
IOM CHANNEL NO. FIELD (ICN)	IDENTIFIES THE ONE OF 64 CHANNELS RESERVED TO SERVICE THE ASSOCIATED DEVICE. IF THE DEVICE IS CONNECTED TO AN EXCHANGE, THIS FIELD IS IGNORED UNLESS THE USE INDICATED CHANNEL IS SET.
USE INDICATED CHANNEL BIT (UIC)	USED IN CONJUNCTION WITH THE ICN WHEN THE DEVICE IS CONNECTED TO AN EXCHANGE. THE BIT IS USED TO INDICATE A SPECIFIC CHANNEL WHEN IT IS DESIRABLE TO OVERRIDE THE IOM PATH SELECTION.
EXCHANGE BIT (EB)	IDENTIFIES THE ASSOCIATED DEVICE AS BEING CONNECTED TO AN EXCHANGE (EB=1) OR A NON-EXCHANGE DEVICE (EB=0).
BYTE TYPE (BT) BIT	SPECIFIES THE BYTE SIZE AS BEING EITHER SIX (BT=1) OR EIGHT (BT=0) BITS FOR STANDARD PERIPHERALS.
TRANSLATE (TR) FIELD	SPECIFIES THE CODE TRANSLATION REQUIREMENTS OF THE IOM FOR THE ASSOCIATED DEVICE.
DEVICE QUEUE HEAD LINK (DQHL) FIELD	IDENTIFIES THE LOCATION OF THE FIRST IO REQUEST (DQE) IN THE DEVICE QUEUE.

QUEUER DEVICE ELEMENT (QDE) BIT	IDENTIFIES THE DE AS A QUEUER DE.
BYTE SIZE (BS) FIELD	SPECIFIES THE AMOUNT OF DATA TO BE TRANSFERRED PER TRANSFER TIME, IN BYTES.
NO. OF SERVICE ATTEMPTS (NS) BITS	MAINTAINS A RECORD OF THE NUMBER OF IOM(S) THAT HAVE ATTEMPTED TO START A JOB ON THE EXCHANGE ASSOCIATED WITH THE DEVICE.
DEVICE QUEUE TAIL LINK (DQTL) FIELD	IDENTIFIES THE LOCATION OF THE LAST IO REQUEST (DQE) IN THE DEVICE QUEUE.
IO ERROR RETRY COUNT (IOER) FIELD	SPECIFIES THE NUMBER OF TIMES AN IO REQUEST SHOULD BE RETRIED IF AN ERROR CONDITION IS PRESENT. (USED FOR AUTOMATIC RETRY.)
HEAD-WARD LINK (HWL) FIELD	IF APPLICABLE, IDENTIFIES THE LEVEL-1 ADDRESS OF THE EXCHANGE ELEMENT OR INDIRECT ELEMENT ASSOCIATED WITH THE DEVICE.
EXCEPTION MASK (EM) FIELD	SPECIFIES CONDITIONS WHICH ARE TO BE PROCESSED AS EXCEPTIONS.

INDIRECT EXCHANGE ELEMENT (IE)

AN INDIRECT ELEMENT (IE) IS REQUIRED FOR EACH SHARED EXCHANGE IN THE I/O SUBSYSTEM. INDIRECT ELEMENTS CONTAIN I/O REQUESTS, INCLUDING ASSOCIATED EXCHANGES IN THE I/O PATH, WHICH HAVE NOT YET BEEN STARTED. THE IE ALSO CONTAINS THE NUMBER OF I/O MODULES LINKED TO THE EXCHANGE. WHEN AN IOM IS SERVICING A REQUEST FOR A DEVICE CONNECTED WITH A SHARED EXCHANGE, THE REQUEST IS DIRECTED TO THE IE BY THE DE. THE IE IN TURN DIRECTS IT TO THE EXCHANGE ELEMENT ASSOCIATED WITH THE LOWEST NUMBERED IOM LINKED TO THE SHARED EXCHANGE. THE VARIOUS FIELDS ARE:

LOCK (L) BIT	INDICATES WHEN PRESENT, THAT THE IE IS BEING OPERATED ON BY AN IOM, AND PREVENTS ANOTHER IOM FROM TRYING TO OPERATE ON IT AT THE SAME TIME.
ELEMENT TYPE (ET) FIELD	IDENTIFIES THE ELEMENT AS AN IE (CODE 011).

NO. OF IOM(S) FIELD	SPECIFIES THE NUMBER OF IOM(S) WHICH SHARE THE EXCHANGE.
HEAD-WARD LINK (HWL) FIELD	IDENTIFIES THE LEVEL-1 ADDRESS OF THE EE ASSOCIATED WITH THE LOWEST NUMBERED IOM CONNECTED TO THE EXCHANGE, TO ENABLE THE "EXCHANGE RING WALK" ROUTINE TO BE EXERCISED WHEN REQUIRED.
RING WALK COUNT (RWC) FIELD	MAINTAINS A RECORD OF THE NUMBER OF JOBS QUEUED FOR THIS EXCHANGE WHICH MUST BE STARTED WITH THE "RING WALK" ROUTINE WHEN AN EXCHANGE CHANNEL BECOMES AVAILABLE.
DE RING LINK (DE-RL) FIELD	IDENTIFIES THE LEVEL-1 ADDRESS OF THE DE OF THE LOWEST NUMBERED DEVICE CONNECTED TO THE EXCHANGE, TO ENABLE THE "DEVICE RING WALK" ROUTINE TO BE EXERCISED WHEN REQUIRED.

EXCHANGE ELEMENT (EE)

AN EXCHANGE ELEMENT (EE) IS ASSOCIATED WITH EACH IOM-TO-EXCHANGE MODULE INTERFACE IN THE I/O SUBSYSTEM. EXCHANGE ELEMENTS CONTAIN THE EXCHANGE CHANNELS WHICH ARE CONNECTED TO A SPECIFIC IOM. IF THE EXCHANGE IS SHARED BY MORE THAN ONE IOM, THE EE ASSOCIATED WITH IOM-EXCHANGE POINTS TO THE NEXT-HIGHER NUMBERED IOM-TO-EXCHANGE EE ASSOCIATED WITH THE SAME EXCHANGE. WHEN AN IOM IS SERVICING AN I/O REQUEST FOR A DEVICE CONNECTED TO A NON-SHARED EXCHANGE, IT IS DIRECTED TO THE EE BY THE DE FOR THAT DEVICE. IT WILL DETERMINE IF AN EXCHANGE CHANNEL IS AVAILABLE TO SERVICE THAT REQUEST. WHEN AN IOM IS SERVICING AN I/O REQUEST FOR A DEVICE CONNECTED TO A SHARED EXCHANGE, IT IS DIRECTED TO THE EE BY EITHER THE IE FOR THAT EXCHANGE OR ANOTHER EE. THE CORRECT EE HAS BEEN FOUND WHEN THE NUMBER IN THE IOM NUMBER FIELD IS THE SEARCHING IOM NUMBER. THE VARIOUS FIELDS ARE:

LOCK (L) BIT	INDICATES WHEN PRESENT, THAT THE EE IS BEING OPERATED ON BY AN IOM, AND PREVENTS ANOTHER IOM FROM TRYING TO OPERATE ON IT AT THE SAME TIME.
ELEMENT TYPE (ET) FIELD	IDENTIFIES THE ELEMENT AS BEING AN EE (CODE 100).

EE RING LINK-DE RING LINK FIELD	IF THE EE IS ASSOCIATED WITH A SHARED (INDIRECT) EXCHANGE, THIS FIELD WILL POINT TO ANOTHER EE WHICH IS ASSOCIATED WITH THE SAME EXCHANGE, TO ENABLE THE "EXCHANGE RING LINK" ROUTINE TO BE EXERCISED BY AN IOM.
	
	IF THE ASSOCIATION IS WITH A NON-SHARED EXCHANGE, THIS FIELD WILL POINT TO THE DE OF THE FIRST DEVICE CONNECTED WITH THE EXCHANGE, TO ENABLE THE "DEVICE RING WALK" ROUTINE TO BE EXERCISED.

RING WALK COUNT (RWC) FIELD	MAINTAINS A RECORD OF THE NUMBER OF JOBS QUEUED FOR THIS EXCHANGE WHICH MUST BE STARTED BY A RING WALK WHEN A CHANNEL IN THE EXCHANGE BECOMES AVAILABLE.
CHANNEL COUNT (CC) FIELD	SPECIFIES THE NUMBER OF CHANNELS RESERVED TO SERVICE THE EXCHANGE.
CHANNEL NO. FIELD	IDENTIFIES THE LOWEST NUMBERED (BASE) I/O CHANNEL ASSOCIATED WITH THE EXCHANGE.
BUSY FIELD (BF)	INDICATES THE STATUS OF EACH CHANNEL CONNECTED WITH THE EXCHANGE.
IOM NO. FIELD	IDENTIFIES THE IOM ASSOCIATED WITH THE EXCHANGE ELEMENT.

DEVICE QUEUE ELEMENT (DQE)

EACH DEVICE HAS ITS OWN DEVICE QUEUE WHICH HOLDS THE I/O REQUESTS, KNOWN AS DEVICE QUEUE ELEMENTS (DQE). THESE REQUESTS ARE SERVICED IN A FIRST IN-FIRST OUT MANNER.

THE DEVICE QUEUE ELEMENTS ARE CONSTRUCTED BY IMP WHEN A PROCESS REQUIRES THAT AN ASYNCHRONOUS TRANSFER BE MADE BETWEEN LEVEL-1 AND LEVEL-3. THE DQE DEFINES THE I/O OPERATION REQUESTED BY THE PROCESS. WHEN IMP LISTS THE DQE IN THE DEVICE QUEUE IT ALSO NOTIFIES ALL I/O MODULES CAPABLE OF SERVICING THE DEVICE TO WHICH THE DEVICE QUEUE BELONGS THAT AN I/O REQUEST IS WAITING TO BE SERVICED.

THERE ARE TWO FORMS OF DEVICE QUEUE ELEMENTS: A GENERAL DQE AND A A DISK DEVICE QUEUE ELEMENT (DDQE). THE DQE IS REQUIRED TO REQUEST AN I/O OPERATION OF ALL PERIPHERAL DEVICES IN THE I/O SUBSYSTEM, EXCLUDING DISK FILE SYSTEMS. THE DDQE IS REQUIRED TO REQUEST AN I/O OPERATION OF A DISK FILE SYSTEM. WHEN REQUESTING THE SERVICES OF A QUEUER-CONTROLLED DISK FILE SYSTEM, THE QUEUER AND THE DISK FILE ARE BOTH CONSIDERED DEVICES. THEREFORE, BOTH THE QUEUER AND THE DISK FILE REQUIRE A DDQE. THE VARIOUS FIELDS ARE:

ELEMENT TYPE (ET) FIELD	IDENTIFIES THE ELEMENT AS BEING A DQE (CODE 010).
INTERRUPT BIT (INB)	SPECIFIES WHETHER OR NOT THE ORIGINATING PROCESS WANTS TO BE NOTIFIED UPON COMPLETION OF THE IO OPERATION.
EXECUTED (EXE) BIT	THIS BIT POSITION IS UNUSED IN THE DQE. HOWEVER, WHEN PRESENT IN THE SQE, INDICATES THAT THE DQE WAS EXECUTED.
LINK TO NEXT REQUEST (H) FIELD	IDENTIFIES THE LEVEL-1 ADDRESS OF THE FOLLOWING DQE QUEUED FOR THIS DEVICE.
BYTE LENGTH FIELD (BL)	DEFINES THE LENGTH OF THE DATA TO BE TRANSFERRED IN BYTES.
OP CODE FIELD	DEFINES THE I/O OPERATION REQUIRED OF THE PERIPHERAL DEVICE.
INSTRUCTION CODE FIELD	DEFINES THE I/O OPERATION REQUIRED OF THE IOM AS FOLLOWS:

CODE 000 - INITIATE I/O

CODE 001 - INTERROGATE PERIPHERAL STATUS -- NOTIFY WHEN NOT READY. (NOT VALID FOR QUEUER CONTROLLED DISK FILE SYSTEMS AND EXCHANGE DEVICES.)

CODE 010 - READ RESULT DESCRIPTOR.

CODE 011 - INTERROGATE PERIPHERAL STATUS -- NOTIFY WHEN READY.

	CODES 100 THRU 111 - HAVE DIFFERENT INTERPRETATIONS FOR THE VARIOUS PERIPHERAL DEVICES IN THE I/O SUBSYSTEM AND, THEREFORE, ARE DEFINED IN THE DATA SERVICE UNIT PORTION OF THIS SPECIFICATION.
NO. OF RETRIES (NR) FIELD	MAINTAINS A RECORD OF THE NUMBER OF TIMES THE DQE WAS INITIATED BEFORE BEING SUCCESSFULLY EXECUTED. (USED FOR AUTOMATIC RETRY OPERATION.)
IMP ERROR (IMPE) BIT	THIS BIT POSITION IS UNUSED IN THE DQE. HOWEVER, WHEN PRESENT IN THE SQE, INDICATES THAT ONE OF THESE ERROR CONDITIONS WAS DETECTED DURING THE INITIATION OF THE DQE AND THE DQE THEREFORE TERMINATED PREMATURELY. A. THE IOE BIT IN THE ASSOCIATED DE WAS SET AT THE TIME THE IOM WAS DIRECTED TO INITIATE THE DQE. B. THE RC FIELD OF THE ASSOCIATED DE WAS EQUAL TO ZERO AT THE TIME THE DQE WAS BEING INITIATED. C. THE BUSY (B) BIT OF THE ASSOCIATED DE WAS RESET WHEN THE IOM ATTEMPTED TO REINITIATE THE DEVICE AFTER A SUCCESSFUL (NORMAL) TERMINATION OF A SUBSEQUENT DQE.
RESULT DESCRIPTOR (RD) FIELD	THIS FIELD IS RESERVED FOR USE IN THE SQE. THE RESULT DESCRIPTOR GENERATED FOR THE OPERATION SPECIFIED IN THE DQE SHALL BE PLACED IN THIS FIELD JUST PRIOR TO THE DQE BEING TRANSFERRED INTO THE STATUS QUEUE.
LEVEL-1 ADDRESS MEMORY BOUNDS (L1MB)	THIS FIELD IS NORMALLY USED TO IDENTIFY THE LEVEL-1 ADDRESS AT WHICH THE DATA TRANSFER IS TO START. HOWEVER, WHEN USED IN CONJUNCTION WITH "LOAD DCP BOUNDS REGISTER" (UNIQUE DSU COMMAND) INSTRUCTION CODE, THIS FIELD IDENTIFIES THE

	MEMORY BOUNDS TO BE LOADED INTO THE MEMORY BOUNDS REGISTER OF THE DCP.
	WHEN THIS FIELD IS NOT USED TO CONTAIN A LEVEL-1 ADDRESS, THE LENGTH IN BYTES, OP CODE, NUMBER OF RETRIES, AND RESULT DESCRIPTOR FIELDS WILL BE INTERPRETED BY THE IOM AS CONTAINING NO VALID DATA.
CHANNEL USED (CU) FIELD	THIS FIELD IS RESERVED FOR USE IN THE SQE TO IDENTIFY THE CHANNEL USED BY THE IOM TO SERVICE THE SPECIFIED DEVICE.
DEVICE VARIANT (DV) FIELD	DEFINES THE OPERATING PARAMETERS OF THE ASSOCIATED PERIPHERAL DEVICE.
LINK BIT (LB)	SIGNIFIES A LINK TO ANOTHER DEVICE AFTER A NORMAL TERMINATION.
LINK ADDRESS (LA) FIELD	THIS FIELD IS USED BY THE IOM TO QUEUE REQUESTS TO ANOTHER DEVICE OR TO THE MEC. THE ADDRESS POINTS TO A DQE OR A DDQE WHICH IS LINKED TO ITS ASSOCIATED DE.
HOME ADDRESS (HA) FIELD	THIS FIELD IS USED IN CONJUNCTION WITH LA TO QUEUE REQUESTS TO OTHER DEVICES.
IOM HOME ADDRESS (IOM-HA) FIELD	IOM NUMBER ASSOCIATED WITH HA.

THE VARIOUS DDQE FIELDS ARE:

ELEMENT TYPE (ET) FIELDS	IDENTIFIES THE ELEMENT AS BEING A DDQE (CODE 101.)
INTERRUPT BITS (INB)	SPECIFIES WHETHER OR NOT THE ORIGINATING PROCESS WANTS TO BE NOTIFIED UPON COMPLETION OF THE I/O OPERATION.
EXECUTED (EXE) BITS	(SAME AS SPECIFIED FOR DQE.)
HEAD-WARD LINK (HWL/H) FIELD	IDENTIFIES THE LOCATION OF THE DE ASSOCIATED WITH THE REQUESTED DISK AND IS ALSO USED WHEN THE DDQE IS LINKED TO THE DE.

BYTE LENGTH (BL) FIELD	SPECIFIES THE TOTAL LENGTH OF THE DATA TO BE TRANSFERRED TO OR FROM THE DISK FILE SYSTEM, IN BYTES,
OP CODE FIELD	DEFINES THE I/O OPERATION REQUIRED OF THE PERIPHERAL DEVICE.
INSTRUCTION (INS) CODE FIELD	(SAME AS SPECIFIED FOR DQE.)
NUMBER OF RETRIES (NR) FIELD	MAINTAINS A RECORD OF THE NUMBER OF TIMES THE DDQE WAS INITIATED BEFORE BEING SUCCESSFULLY EXECUTED (USED FOR AUTOMATIC RETRY OPERATION.)
IMP ERROR (IMPE) BIT	(SAME AS SPECIFIED FOR DQE)
QUEUER INSTRUCTION CODE (QIC)	THE INSTRUCTION CODE USED WHEN THE QDE BIT IS SET IN THE ASSOCIATED DE.
QUEUER OPERATION CODE (QOC)	THE OPERATION CODE USED WHEN THE QDE BIT IS SET IN THE ASSOCIATED DE.
RESULT DESCRIPTOR (RD) FIELD	(SAME AS SPECIFIED FOR DQE)
LEVEL-1 ADDRESS FIELD	IDENTIFIES THE LEVEL-1 ADDRESS AT WHICH THE DATA TRANSFER IS TO START.
CHANNEL USED (CU) FIELDS	(SAME AS SPECIFIED FOR GENERAL DQE)
DISK ADDRESS FIELD (DA)	IDENTIFIES THE STARTING DISK ADDRESS INVOLVED IN THE DATA TRANSFER.
BIT LENGTH (BL) FIELD	USED IN CONJUNCTION WITH BL TO ALLOW DISK TRANSFER TO BE SPECIFIED TO THE BIT.

TYPICAL I/O INITIATION USING IOSM

A TYPICAL I/O OPERATION STARTS WHEN A PROCESS BEING PERFORMED BY A CPM REQUIRES THAT AN ASYNCHRONOUS TRANSFER OF DATA BE MADE BETWEEN LEVEL-1 AND LEVEL-3 STORAGE. WHEN THIS OCCURS, THE PROCESS PREPARES A REQUEST WHICH CONTAINS:

A. A TERMINAL DESCRIPTION OF THE LEVEL-1 STORAGE AREA INVOLVED.

B. AN OPERATION CODE WHICH DEFINES THE OPERATIONS REQUIRED OF THE PERIPHERAL DEVICE.

C. A TERMINAL DESCRIPTION OF A LEVEL-3 DEVICE.

D. AN I/O COMPLETE CODE TO INDICATE WHETHER OR NOT THE REQUESTING PROCESS WANTS TO BE NOTIFIED WHEN THE I/O OPERATION IS COMPLETE.

E. AN I/O INSTRUCTION CODE WHICH DEFINES THE OPERATIONS REQUIRED OF THE IOM.

THIS REQUEST IS PASSED TO IMP VIA A PROGRAMMATIC IMP CALL. IMP THEN PERFORMS FINAL COMBINE OPERATIONS ON THE TERMINAL DESCRIPTIONS PRESENT IN THE REQUEST. IF A FINAL COMBINE ERROR OCCURS, IMP WILL FAULT THE REQUESTING PROCESS. IF NO FINAL COMBINE ERROR CONDITIONS EXIST, IMP WILL USE THE ABSOLUTE DESCRIPTION OF THE LEVEL-3 DEVICE (DEVICE VECTOR INDEX) TO SELECT, FROM THE DEVICE VECTOR, THE POINTER TO THE APPROPRIATE DE, WHICH IT THEN READS AND LOCKS. IMP THEN CHECKS THE DE TO ENSURE THAT THE I/O ERROR (IOER) BIT IS OFF, AND THAT THE BUSY BIT IS NOT SET CONCURRENT WITH THE REQUEST COUNT FIELD BEING EQUAL TO ZERO. IF ANY OF THESE ERROR CONDITIONS EXIST, IMP WILL FAULT THE REQUESTING PROCESS. IF NO ERROR CONDITIONS EXIST, IMP WILL CONSTRUCT A DQE WHICH WILL IDENTIFY THE I/O OPERATION REQUIRED. THE CONSTRUCTION OF THE DQE BY IMP INVOLVES THE FOLLOWING OPERATIONS:

A. ALLOCATING STORAGE SPACE FOR THE REQUEST.

B. DETERMINING THE BYTE COUNT TO BE PLACED IN THE REQUEST. THIS IS ACCOMPLISHED BY READING THE BYTE SIZE OF THE DEVICE FROM THE DE, AND USING IT TO INTEGER DIVIDE THE LENGTH FIELD OF THE LEVEL-1 STORAGE DESCRIPTION SPECIFIED BY THE REQUESTING PROCESS.

C. TRANSFERRING THE OPERATION AND INSTRUCTION CODES FROM THE PROCESS REQUEST TO THE REQUEST BEING CONSTRUCTED.

D. PLACING THE ABSOLUTE LEVEL-1 STARTING ADDRESS OF THE LEVEL-1 STORAGE AREA INVOLVED IN THE REQUEST.

E. GENERATING AND PLACING THE PROCESS IDENTIFICATION IN THE REQUEST.

IMP THEN LINKS THE DQE INTO THE BOTTOM OF THE APPROPRIATE DEVICE QUEUE, AS POINTED TO IN THE DEVICE QUEUE TAIL LINK FIELD IN THE APPROPRIATE DE. THE DEVICE QUEUE TAIL LINK FIELD IS THEN UPDATED, AND THE REQUEST COUNT (RC) FIELD IS INCREMENTED. THE DE IS THEN WRITTEN BACK INTO MEMORY, UNLOCKING THE ELEMENT.

IF THE REQUEST COUNT FIELD OF THE DE WAS ORIGINALLY ZERO, IMPLYING THERE WERE NO JOBS ALREADY QUEUED FOR THE REQUESTED DEVICE, THE DE ADDRESS IS SENT TO THE "HOME DEVICE" FIELD OF ALL THE I/O MODULES IDENTIFIED IN THE DEVICE VECTOR AS HAVING PATHS TO THE DEVICE, AND THESE I/O MODULES ARE NOTIFIED OF THIS ACTION VIA AN INTERRUPT BUS. IF THE REQUEST COUNT OF THE DE WAS NOT EQUAL TO ZERO, THE DQE WOULD JUST BE LINKED INTO THE BOTTOM OF THE DEVICE QUEUE, AND NOTIFICATION OF THE I/O MODULES WOULD NOT BE NECESSARY.

THE PROCESSOR IS THEN FINISHED WITH THE IO OPERATION AND CAN DO AN IMP RETURN. IF THE ELEMENTS IMP ATTEMPTED TO READ FROM THE IOSM WERE LOCKED, IMP MUST WAIT TO QUEUE THE DQE UNTIL THEY ARE UNLOCKED.

FUNCTIONAL IOM DIVISION

THE IOM IS FUNCTIONALLY DIVIDED INTO:

A. TRANSLATOR UNIT (TU.)

B. MEMORY INTERFACE UNIT (MIU).

C. DATA SERVICE UNIT (DSU).

TRANSLATOR UNIT

THE TU IS A SPECIALIZED PROCESSOR. THE TU IS THE IOM MECHANISM WHICH SERVICES THE I/O REQUESTS, BY ADDRESSING THE I/O SUBSYSTEM MAP TO DETERMINE THE LOCATION OF THE DQE HOLDING THE SERVICE REQUEST AND DETERMINE THE PATHS AVAILABLE TO DEVICE BEING SERVICED. THE TRANSLATOR RESPONDS TO:

A. NEW REQUEST INITIATIONS.

B. DATA SERVICE COMPLETE CONDITIONS.

C. IMP AND IOM ERROR CONDITIONS.

D. DISK QUEUER INITIATED I/O OPERATIONS AS AN EXPLICIT MODE OF OPERATION.

MEMORY INTERFACE UNIT

THE MIU WILL PERFORM ALL LEVEL-1 TO LEVEL-3 TRANSFERS. THE MIU WILL HANDLE THESE TRANSFERS AS FIELD-ORIENTED OPERATIONS ON A PREASSIGNED BASIS OF PRIORITY. THE PRIORITY IS BASED ON THE FUNCTIONAL UNIT MAKING THE REQUEST. (SEE SPECIFICATION CP 1760-0008 FOR DETAILS).

DATA SERVICE UNIT

THE DSU IS COMPOSED OF:

A. A DATA COMMUNICATION PROCESSOR INTERFACE UNIT, WHICH INTERFACES WITH DATA COMMUNICATION DEVICES.

B. A MULTIPLE WORD DEVICE INTERFACE UNIT WHICH INTERFACES WITH EXTREMELY FAST DEVICES, AS DEFINED IN SPECIFICATIONS CP 1760-1220, CP 1720-4454, AND CP 1904-3223.

C. A PERIPHERAL CONTROLLER INTERFACE UNIT WHICH INTERFACES WITH ALL OTHER DEVICES.

ALL THREE UNITS WILL WORK ASYNCHRONOUSLY WITH THEIR DEVICE CONTROLLERS (SEE SPECIFICATION CP 1760-0008 FOR DETAILS).

RELATED SPECIFICATIONS

MEMORY MODULE	CP1720-5576
INTERPRETER	CP1720-5592
I/O MODULE	CP1760-0008
B8500 GENERAL SPECIFICATION	1761-2045

RELATED HARDWARE SPECIFICATIONS

A. CARD READERS

B 9111	800 CPM	7 1657
B 9112	1400 CPM	7 1657

B. CARD PUNCH

B 9213	300 CPM	1188 2271

C. PRINTERS

B 9242	815 LPM	78 411
B 9243	1040 LPM	78 411

D. PAPER TAPE READER

B 9120	500-1000 CPS	1187 9376

E. PAPER TAPE PUNCH

B 9220	100 CPS	1187 9384

F. MAGNETIC TAPE UNITS

B 9381	36 KB CLUSTER, 4-STA. (9-CH.800 BPI)	1630 0824
B 9382	72 KB CLUSTER, 4-STA. (9-CH. 1600 BPI)	1630 0824
B 9383	9-25-36 KC CL.,4-STA. (7-CH.200 556 800 BPI)	1630 0824
B 9391	18-50-72 KC M.T. UNIT (7-CH. 200 556 800 BPI)	
B 9392	72 KB M.T. UNIT (9-CH. 800 BPI)	1129 1218
B 9393	144 KB M.T. UNIT (9-CH. 1600 BPI)	
B 9394	24-66-96 KC M.T. UNIT (7-CH. 200 556 800 BPI)	1129 1218

G. DISK FILES (SYSTEMS MEMORY AND MODULAR RANDOM STORAGE)

B 9372-1	10 MILLION BYTE STORAGE 20 MS	1630 0626
B 9372-7	20 MILLION BYTE STORAGE 23 MS	1122 5265

H. DISK FILE (DATA MEMORY BANKS)

B 9375-0	100 MILLION BYTE STORAGE 23 MS	1122 5265
B 9375-2	100 MILLION BYTE STORAGE 40 MS	1122 5265
B 9375-3	100 MILLION BYTE STORAGE 60 MS	

I. DISK FILE ELECTRONIC UNITS

B 9371-1	DFEU FOR B 9372-1	1198 0778
B 9371-2	DFEU FOR B 9372-7	1122 5273

J. DATA COMMUNICATION DEVICES

B 9350	TYPEWRITER INQUIRY STATION	1187 9756

K. DATA COMMUNICATION CONTROL

B 6350	DATA COMMUNICATION PROCESSOR	1141 8225
B 7350	DATA COMMUNICATION PROCESSOR	1141 8225

L. MULTIPLE TAPE LISTER

B 9244	SIX TAPE DRUM LISTER	1631 0096

M. MICR SORTER READER

B 9130	1149 5454
B 9131	1149 5454
B 9132	1149 5454

N. REMOTE COMPUTERS

TC 500	
B 500	1631 0963
B 2500/3500	1121 0606
B 5500	1631 0963
B 6500/7500	1128 7505

O. DISPLAY DEVICES

B 9351	INPUT AND DISPLAY	1700 2890
B 9352	INPUT AND DISPLAY	1718 1777

P. I/O CONTROL

B 8512	INPUT/OUTPUT MODULE	CP1760 0008

SECTION 7

OPERATORS CONSOLE

INTRODUCTION

THE OPERATORS CONSOLE PROVIDES THE OPERATOR WITH FACILITIES FOR COMMUNICATING WITH THE B8500 SYSTEM. THE OPERATORS CONSOLE CONSISTS OF A SUPERVISORY DISPLAY AND AN OPERATORS CONTROL PANEL.

SUPERVISORY DISPLAY

THE SUPERVISORY DISPLAY INCLUDES A 7-INCH BY 8.5-INCH CATHODE-RAY TUBE MONITOR SCREEN DISPLAYING UP TO 24 LINES OF 40 CHARACTERS EACH. THE DISPLAY PROVIDES THE CAPABILITY OF DISPLAYING 69 DIFFERENT CHARACTERS PLUS A CURSOR WHICH IS USED TO INDICATE THE ACTIVE INPUT POSITION. INPUT FROM THE KEYBOARD AS WELL AS OUTPUT FROM THE SYSTEM IS DISPLAYED. THE KEYBOARD INCLUDES A CONVENTIONAL TYPEWRITER KEY ARRANGEMENT PLUS SPECIAL CHARACTERS AND CONTROL KEYS. KEYS ARE ELECTRICALLY INTERLOCKED TO PREVENT ACTIVATION OF MORE THAN ONE KEY AT A TIME.

OPERATORS CONTROL PANEL

THE OPERATORS CONTROL PANEL CONTAINS SWITCHES AND INDICATORS NEEDED FOR OPERATING THE SYSTEM BUT DOES NOT INCLUDE SWITCHES AND INDICATORS USED IN SYSTEM MAINTENANCE. THE CONTROL PANEL ALSO HAS A CONVENIENT WORK SURFACE.

CONTROL PANEL SWITCHES

THE CONTROL PANEL CONTAINS THE FOLLOWING:

LOAD SWITCH

THE LOAD SWITCH IS USED TO START OPERATION OF THE SYSTEM. THE LOAD SWITCH CLEARS ALL CONTROLLERS, LOADS DATA FROM A DESIGNATED CARD READER OR DISK (DEPENDING UPON THE SETTING OF THE LOAD SELECT SWITCH) INTO ADDRESS 0 OF THE DESIGNATED LEVEL-1 MEMORY MODULE, AND INITIATES THE CPM DESIGNATED BY THE DESIGNATE SWITCH.

LOAD SELECT SWITCH

THE LOAD SELECT SWITCH HAS TWO SETTINGS WHICH, UPON ACTIVATION OF THE LOAD SWITCH, PROVIDE ONE OF THE FOLLOWING FUNCTIONS:

A. CARD: THE "CARD" SETTING IS USED TO READ A BOOTSTRAP PROGRAM INTO THE SYSTEM FROM A CARD READER.

B. DISK: THE "DISK" SETTING OF THE SELECT SWITCH IS USED TO READ IN A BOOTSTRAP PROGRAM FROM DISK.

DESIGNATE SWITCH

THE DESIGNATE SWITCH IS USED TO SPECIFY THE NUMBER OF THE CPM WHICH WILL BE AFFECTED BY ACTIVATION OF THE HALT, RESTART, OR LOAD SWITCH.

HALT SWITCH

THE HALT SWITCH CAUSES THE CPM INDICATED BY THE DESIGNATED SWITCH TO EITHER STOP AT COMPLETION OF THE CURRENT OPERATION, TREAT HALTS ON NO-OP OPERATORS, OR TO STOP WHEN THE VALUE OF THE ADDRESS REGISTER IS EQUAL TO THE VALUE OF THE STOP-ADDRESS REGISTER.

RESTART SWITCH

THIS SWITCH CAUSES THE CPM INDICATED BY THE DESIGNATED SWITCH TO BE RESTARTED.

POWER-ON SWITCH

THE POWER-ON SWITCH CAUSES POWER TO BE SUPPLIED TO THE SYSTEM. PERIPHERAL DEVICES HAVE THEIR OWN POWER CONTROL SWITCHES AND ARE NOT AFFECTED BY THIS SWITCH.

POWER-OFF SWITCH

THIS SWITCH CAUSES POWER TO BE REMOVED FROM ALL THE SYSTEM UNITS EXCEPT PERIPHERAL DEVICES.

RELATED HARDWARE SPECIFICATIONS

B9352	INPUT AND DISPLAY TERMINAL	1718 1777

SECTION 8

DIAGNOSTICS AND CONFIDENCE TESTING

INTRODUCTION

DIAGNOSTIC AND CONFIDENCE TESTING FOR THE B8500 IS DIVIDED INTO OFF-LINE STATIC TESTING AND ON-LINE DYNAMIC TESTING. THE OFF-LINE TEST FACILITY IS THE MAINTENANCE DIAGNOSTIC PROCESSOR (MDP) AND THE ON- LINE TEST FACILITY IS THE AUTOMATIC DIAGNOSTIC SUBSYSTEM (ADS).

MAINTENANCE DIAGNOSTIC PROCESSOR (MDP)

THE MAINTENANCE DIAGNOSTIC PROCESSOR (MDP) IS ACTIVATED WHEN THE PRESENCE OF A RECURRING HARDWARE ERROR BECOMES APPARENT. THE MDP PERMITS AN EXHAUSTIVE TEST OF ALL PROCESSORS (CPU, IOM, MEC, ETC.) AND ALL PRIMITIVE FUNCTIONS WITHIN THEM.

SHOULD THIS HARDWARE ERROR BE VALIDATED BY THE ADS, IT THEN BECOMES POSSIBLE FOR THE SYSTEM TO DETECT AND FOR THE MDP TO RESOLVE AN ERROR CONDITION WITHOUT STOPPING THE ENTIRE SYSTEM.

AUTOMATIC DIAGNOSTIC SUBSYSTEM (ADS)

THE ADS ESTABLISHES AND MAINTAINS SYSTEM PERFORMANCE. IF AN ERROR EXISTS, THE ADS WILL INITIATE THE NECESSARY LEVELS OF TESTING TO ISOLATE A REASONABLE LOGIC STRING WHILE THE DEGRADED SYSTEM CONTINUES WITH USEFUL WORK.

THE ADS WILL BE SCHEDULED BY OUTSIDE INTERVENTION TO PERFORM PERIODIC SYSTEM TESTING. THE ADS WILL BE SUPPLIED ALL REQUIRED RESOURCES, INCLUDING A TIME FRAME, TO PERFORM A GIVEN TEST FUNCTION. FOR EXAMPLE, THE ADS WILL BE ACTIVATED AND PASSED THE REQUIRED RESOURCES TO TEST A DISK FILE STORAGE UNIT WHICH HAS BEEN INDICATING ERRORS, OR THE OPERATOR WILL REQUEST THE ADS TO EXECUTE A SYSTEM CONFIDENCE CHECK.

THE ADS OPERATES IN DETECTION AND ISOLATION MODES. IN DETECTION MODE, SYSTEM CONFIDENCE IS MAINTAINED BY THE EXECUTION OF CONFIDENCE ROUTINE SEGMENTS THAT ARE STRUCTURED TO THE COMPLEXITY OF USER PROGRAMS. IF AN ERROR IS DETECTED, THE ADS WILL ENTER THE ISOLATION MODE WHERE A WELL-DEFINED SEQUENCE, DETERMINED BY THE CONFIDENCE SUBFUNCTION THAT DETECTS THE ERROR, WILL BE EXECUTED.

THE ADS CONTAINS A SYSTEMS CONFIDENCE FUNCTION (SCF), WHICH IS IMPLEMENTED BY INVOLVING AS MUCH OF THE OPERATING SYSTEM CAPABILITIES AS POSSIBLE. THE SCF CONSISTS OF DATA PROCESSING, COMPUTATION, AND LIST PROCESSING TEST SEGMENTS. FOR EXAMPLE, A SEGMENT IN THE DATA PROCESSING SUBFUNCTION MIGHT WELL BE A SORT PROBLEM THE RESULTS OF WHICH ARE CHECKED AGAINST KNOWN SORTED DATA.

SECTION 9

GLOSSARY

ACCESSING	TRANSFORMING A LOGICAL NAME TO A PHYSICAL LOCATION.
ADDRESS SPACE	THE SET OF STORAGE AND DEVICES ACCESSIBLE BY THE PROCESS.
ALLOCATE	BIND A STRUCTURE TO A MORE-GLOBAL STRUCTURE.
ATTRIBUTE	A CHARACTERISTIC OF AN OBJECT THAT CAN BE INTERPRETED BY THE SYSTEM.
BINDING	THE ASSOCIATION OF A SPACE TO AN OBJECT IDENTIFIED IN A PROCESS.
DEALLOCATE	TO REMOVE A STRUCTURE FROM A MORE-GLOBAL STRUCTURE. THE SPACE IS RETURNED TO UNASSIGNED SPACE.
DESCRIPTION	SEE STRUCTURE.
DESCRIPTOR	SEE DESCRIPTION.

DISPATCHING	THE ACTION INVOLVED IN RECOGNIZING MESSAGES FROM SUBNODES AND EITHER TRANSMITTING THE MESSAGES OR SERVICING THEM.
ELEMENT	AN ACCESSIBLE CONTIGUOUS SET OF BITS.
CONTAINER ELEMENT	A CONTIGUOUS SET OF BITS WHICH BOUND A STRUCTURE.
SIMPLE ELEMENT	AN ELEMENT WHICH, FOR A GIVEN STRUCTURE, HAS NO SUBSTRUCTURE; THAT IS, THE ELEMENT CANNOT BE PARTITIONED INTO A SET OF ELEMENTS.
FAULT	AN OCCURRENCE SUCH AS DIVIDE BY ZERO, MEMORY PARITY ERROR, ETC.
EXTERNAL FAULT	A FAULT IN WHICH AN ANCESTOR PROCESS IS TO RECEIVE CONTROL ON OCCURRENCE IN THE COMPOUND PROCESS.
INTERNAL FAULT	A FAULT WHICH THE PARENT PROCESS DELEGATES TO A SUBPROCESS.
LOCALLY EXTERNAL FAULT	A FAULT IN WHICH THE PARENT PROCESS WANTS TO RECEIVE CONTROL ON THE OCCURRENCE IN THE SUBPROCESS.
FIELD	AN ELEMENT WHICH IS TREATED AS AN ENTITY; THAT IS, THE ELEMENT IS TREATED AS IF IT HAS NO SUBSTRUCTURE.
FIELD ISOLATION UNIT	A MODULE WHICH SERVES AS THE INTERFACE BETWEEN THE CPM AND MEMORY STORAGE UNITS.

FREE SPACE	THAT SPACE WITHIN A CONTAINER ELEMENT THAT IS NOT ALLOCATED. FREE SPACE IS STRUCTURED.
INTERPRETER	NAME PROCESSING ELEMENT IN THE CPM.
INTERRUPT	A MEANS FOR MODULES OF THE SYSTEM TO COMMUNICATE WITH EACH OTHER.
LEAFWARD	A TERM USED TO INDICATE THE DIRECTION FROM A PROCESS TOWARD THE SUBPROCESS IN A PROCESS TREE. (OPPOSITE OF ROOTWARD).
LENGTH	THE NUMBER OF BITS IN A FIELD.
LEVEL-1	MAIN MEMORY STORAGE.
LEVEL-2	LEVEL-1 BACKUP STORAGE.
LEVEL-3	PERIPHERAL UNITS.
LINKED LIST	A CONTAINER ELEMENT PARTITIONED INTO A SET OF ALLOCATED ELEMENTS AND A FREE SPACE. EACH ALLOCATED ELEMENT (OR LIST ELEMENT) IS PARTITIONED INTO A LINK ELEMENT AND AN INFORMATION ELEMENT. EACH LINK ELEMENT IS A SET OF REFERENCES WHICH DEFINE THE PATH TO THE LOGICALLY ADJACENT LIST ELEMENT.
LOCATION	THE FIRST BIT.

MEMORY STORAGE UNIT	THE LEVEL-1 STORAGE MODULES. THE MEMORY STORAGE UNITS AND FIELD ISOLATION UNITS TOGETHER MAKE UP LEVEL-1 STORAGE.
MULTIPROCESSING	THE SIMULTANEOUS OR INTERLEAVED EXECUTION OF TWO OR MORE PROGRAMS OR SEQUENCES OF INSTRUCTIONS BY A COMPUTER NETWORK.
MULTIPROGRAMMING	THE INTERLEAVED EXECUTION OF TWO OR MORE PROGRAMS BY A COMPUTER.
NAME	STACK NUMBER, LEXIC LEVEL, DISPLACEMENT, OR LEXIC LEVEL, DISPLACEMENT, OR DISPLACEMENT
NESTED	TO BE INCLUDED WITHIN ANOTHER, AS, FOR EXAMPLE, A SUBPROCESS WORK SPACE IS CONTAINED WITHIN THE WORK SPACE OF A PARENT PROCESS.
NODE INDEX VECTOR	THE PATH FOLLOWED BY THE INTERPRETER TO REACH ITS CURRENT PROCESS IN THE PROCESS HIERARCHY.
OBJECT	ANY STRUCTURE WHICH IS ACCESSED AS A SIMPLE ELEMENT.
PREEMPT FUNCTION	AN IMP SCHEDULING FUNCTION THAT ALLOWS A PROCESS TO REMOVE ALL PROCESSORS FROM A COMPOUND SUBPROCESS.

PROCESS	EXECUTION OF PROGRAM CODE BY A PROCESSOR.
ANCESTOR PROCESS	A PROCESS THAT IS A PREDECESSOR OF A PROCESS.
COMPOUND PROCESS	AN ORDERED SET OF PROCESSES OR THE TREE STRUCTURE OF NESTED PROCESSES (ALSO CALLED PROCESS SUBTREE).
ELEMENTARY PROCESS	A LEAF PROCESS.
GLOBAL PROCESS	THE PARENT PROCESS OF EVERY OTHER PROCESS IN THE SYSTEM.
LEAF PROCESS	A PROCESS WHICH HAS NO SUB-PROCESS. A LEAF PROCESS BECOMES A NODE PROCESS BY CREATING SUBPROCESSES.
NESTED PROCESS	A PROCESS WHICH HAS ITS WORK SPACE CONTAINED WITHIN THE WORK SPACE OF ANOTHER PROCESS.
NODE PROCESS	A PROCESS WHICH MANAGES SUB-PROCESSES.
PARENT PROCESS	THE PROCESS THAT CREATED THE PRESENT PROCESS.
PROCESS QUEUE	A QUEUE OF MESSAGES ESTABLISHED FOR EACH PROCESS.
SON PROCESS	A PROCESS CREATED BY A PARENT PROCESS.
PROCESS CALL	AN IMP SCHEDULING FUNCTION THAT ESTABLISHES A NEW PROCESS IN THE HIERARCHY BY ADDING A NEW PROCESS DEFINITION TO THE RESOURCE STACK AND ACTIVATING THE NEW PROCESS.
PROCESS RETURN	AN IMP SCHEDULING FUNCTION THAT ALLOWS FOR THE RETURN OF A PROCESSOR TO A PARENT PROCESS.

PROCESS SUBTREE	A SET OF SUBPROCESSES OR THE TREE STRUCTURE OF NESTED PROCESSES.
PROCESSOR	A PROCESSING ELEMENT WORKING ON A PROCESS SPACE.
PUSHDOWN	A STACK OF VARIABLE-LENGTH ELEMENTS.
QUEUE	A CONTAINER ELEMENT WHICH IS PARTITIONED INTO ALLOCATED AND FREE SPACE. THE ALLOCATION RULE IS "FIRST-IN, FIRST-OUT."
READ	THE OPERATION WHICH OBTAINS A COPY OF A SPECIFIED ELEMENT IN A STRUCTURE.
REFERENCE	TERMINAL DESCRIPTION OF A FIELD.
ROOTWARD	A TERM USED TO INDICATE THE DIRECTION TOWARD THE ANCESTOR PROCESSES IN THE PROCESS TREE. (OPPOSITE OF LEAFWARD).
SCHEDULING	THE OPERATION OF SELECTING AND ACTIVATING A SUBPROCESS OR RETURNING TO A PARENT PROCESS.
SEGMENT	A PORTION OF INFORMATION PERTAINING TO A PROCESS.
SEGMENT CONTAINER	ALLOCATED STORAGE WHICH HOLDS ONE OR MORE SEGMENTS.
SEGMENTATION	A METHOD OF RESTRUCTURING OF INFORMATION FROM ONE

	CONTAINER TO MULTIPLE CONTAINERS.
SET	A COLLECTION OF ELEMENTS.
STACK	A STORAGE AREA SUCH THAT ITEMS ENTERED INTO IT CAN BE REMOVED ONLY IN A FIRST-IN, LAST-OUT ORDER AND CONTAINING A SET OF CONTIGUOUS, EQUAL-LENGTH ELEMENTS.
STRUCTURE	AN ORDERING OF A SET OF ELEMENTS WHICH DEFINES THE MANNER IN WHICH AN ELEMENT IN THE SET IS ACCESSED.
COMPLEX STRUCTURE	A NESTED SET OF STRUCTURES IN WHICH THE INNERMOST STRUCTURE IS A SIMPLE ELEMENT, FOR EXAMPLE, A STACK OF STACKS, A STACK OF VECTORS, A VECTOR OF STACKS, ETC.
COMPOSITE STRUCTURE	A PHYSICAL LOCATION WHICH MAY BE ACCESSED BY DIFFERENT STRUCTURE DISCIPLINES, THAT IS, A MAPPING OF DIFFERENT STRUCTURES ONTO THE SAME PHYSICAL LOCATION.
SEQUENTIAL STRUCTURE	A STRUCTURE IN WHICH, GIVEN AN ELEMENT, EI, IN THE STRUCTURE, THE NEXT ELEMENT EI+1, IN SUCCESSION IN THE STRUCTURE MAY BE FOUND IN A UNIQUE MANNER.
SUBPROCESS	A PROCESS WHOSE ADDRESS SPACE IS PROPERLY NESTED IN THE ADDRESS SPACE OF ITS IMMEDIATE PARENT PROCESS.
TEMPLATE	A DESCRIPTION WHICH HAS NOT BEEN BOUND TO AN ALLOCATED STRUCTURE. THE STRUCTURE

	EXPRESSION CONTAINS A CALL OPERATOR WITH A MEANINGLESS "LL, D" ADDRESS.
TERMINAL DESCRIPTION	A SEGMENT CONTAINER RELATIVE DESCRIPTION.
WORK SPACE	A SPACE WHICH INCLUDES ALL INFORMATION PRIVATE TO THE EXECUTION OF A PROCESS.
VECTOR	A SEQUENCE OF EQUAL-LENGTH, CONTIGUOUS ELEMENTS IN WHICH EACH ELEMENT IN THE VECTOR IS ACCESSED BY AN INDEX VALUE. FOR EXAMPLE, THE INDEX VALUE MULTIPLIED BY THE LENGTH OF THE ELEMENT IS ADDED TO THE LOCATION FIELD OF THE CONTAINER OF THE VECTOR TO GIVE THE LOCATION FIELD OF THE ELEMENT DESIRED).
WRITE	THE OPERATION WHICH REPLACES THE CONTENTS OF A SPECIFIED ELEMENT IN A STRUCTURE WITH A NEW ELEMENT OF THE SAME SIZE.

APPENDIX A

TYPICAL STRUCTURE MANIPULATIONS

THE MICRO-SEQUENCES ILLUSTRATED IN THIS APPENDIX ARE MEANT TO BE FUNCTIONAL REPRESENTATIONS ONLY. FOR THE ACTUAL HARDWARE IMPLEMENTATION, REFER TO CP 1720-5592 (DETAILED DESIGN SPECIFICATION FOR B8502 CENTRAL PROCESSOR MODULE).

APPENDIX A PRESENTS INSTRUCTION MICRO-SEQUENCES AND ILLUSTRATIONS OF THE ATTRIBUTE STACK (AS) AND STATE WORD (SW) BEFORE, DURING, AND AFTER TYPICAL STRUCTURE MANIPULATIONS.

STANDARD MICRO-OPERATORS USED IN THE INSTRUCTION SEQUENCES ARE DEFINED BY THE STEPS WHICH FOLLOW THE NAME OF THE MICRO-OPERATOR (SEE TABLE A-1).

WITHIN AN INSTRUCTION SEQUENCE, AN AS0 ON THE RECEIVING SIDE OF AN OPERATION INDICATES A WRITE-OVER; AN AS0 OR AS1, ETC. ON THE SENDING SIDE INDICATES DEPTH IN THE STACK (AS0 INDICATES THE TOP OF THE STACK).

THE ATTRIBUTE STACK IS SHOWN ON THE TOP OF EACH ILLUSTRATION. THE NUMBERS AT THE LEFT OF THE ATTRIBUTE STACK CORRESPOND TO THE NUMBERS PRECEDING EACH STEP OF THE INSTRUCTION SEQUENCE AND INDICATE AN ENTRY IN THE STACK. THE NUMBERS TO THE RIGHT OF THE ATTRIBUTE STACK INDICATE A STACK DELETION. THESE DELETIONS ARE ALSO INDICATED BY A CROSSHATCH ON THE ILLUSTRATION. NOTICE THAT NOT ALL STACK ENTRIES ARE DELETED DURING AN OPERATION.

THE STATE WORD IS SHOWN AT THE BOTTOM OF EACH ILLUSTRATION. WHEN AN INSTRUCTION SEQUENCE DOES NOT CHANGE THE STATE WORD, A STATE WORD ILLUSTRATION IS NOT PROVIDED.

WHEN A CONSTRUCT EXAMPLE IS NOT ACCOMPANIED BY AN ILLUSTRATION THE EXAMPLE IS IDENTICAL WITH THE REMOVE EXAMPLE, EXCEPT THAT THERE ARE NO CHANGES IN THE STATE WORD.

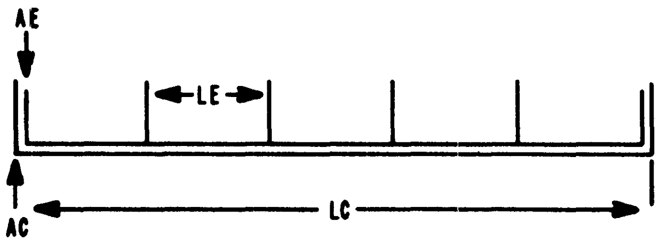
TABLE A-1. STANDARD MICRO-OPERATORS

```
<BRANCH> ::= <DPCRO, AC, LC  ← ASO, AC, LC> <DEL AS, AC, LC>

<CALL>  ::=  <DPCR, AC, LC ← ASO, AC, LC> <DEL AS, AC, LC>

<COMBINE> ::=  <ASO, AC ← ASO, AC + ASO, AE>
               <ASO, LC ← ASO, LE> <DEL AS, AE, LE> <BOUNDS CHECK>

<CSP (M)> ::=  <CSPR ← M> <ASO, AE ← ASO, AE + LENGTH (M)>
               <ASO, LE ← ASO, LE + LENGTH (M)>

<DEL A> ::=  <AS,A ← EMPTY>

<DUP A> ::=  <AS,A ← ASO,A>

<EOQ, "L"> ::=  <DUP AS, AC, LC,LE> <ASO, AE ← "L">
                <DUP AS,AE> <COMBINE> <+ SEQUENCE> <BRANCH>

<EXCH AS,F> ::=  <AS,F ← AS1,F> <AS2,F ← EMPTY>

<FETCH TO (X)> ::= <X ← CONTENTS OF LEVEL-1 STORAGE AT AS,AC, LC>

< + MOD SEQUENCE> ::=  <ASO, AE ← (ASO, AE + ASO, LE)
                       MOD ASO, LC> <SW, AE ← ASO,AE>

<MOVE (P)> ::=  <CONTAINER (ASO, AC, LC) ← CONTENTS (P)>
                <DEL AS, AC, LC>

<RET> ::=  <DPCRO,AC,LC ← EMPTY>
```

TABLE A-1. STANDARD MICRO-OPERATORS (CONTINUED)

```
<RTS> ::=  <AS.AC,LC ← DPCRO.AC,LC> <DPCRO.AC,LC ← EMPTY>

< + SEQUENCE> ::=  <ASO, AE ← ASO, AE + ASO, LE>
                   <SW, AE ← ASO, AE>

<+SEQUENCE AS,AE> ::= <AS,AE ← AS,AE + AS,LE>

< + SEQUENCE, SW> ::=  <SW, AE ← ASO, AE + ASO, LE>

< - SEQUENCE> ::= <ASO, AE ← ASO, AE - ASO, LE>
               <SW,AE ← ASO,AE>

<-SEQUENCE AS,AE> ::= <ASO,AE ← ASO,AE - ASO,LE>

<SELECT> ::= <ASO, AE ← ASO, AE + (ASO, LE × AS1, LE)>
             <DEL ASO, LE>

<SSW> ::=  <RETURN SW TO MEMORY>

<STORE FROM (X)> ::= <LEVEL-1 STORAGE AT (AS.AC, LC) ← (X)>

<R> ::=  <END OF INSTRUCTION STRING>
```

VECTORS



CONSTRUCT

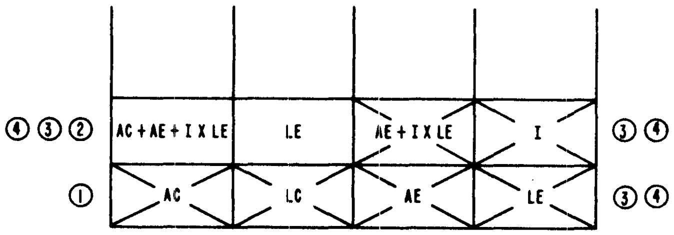
1. AS.AC,LC,AE,LE ← SW.AC,LC,AE,LE

2. AS.LE ← P.I

3. SELFCT

4. COMRINE

5. R



ATTRIBUTE STACK
NO STATE WORD CHANGE

FIELD

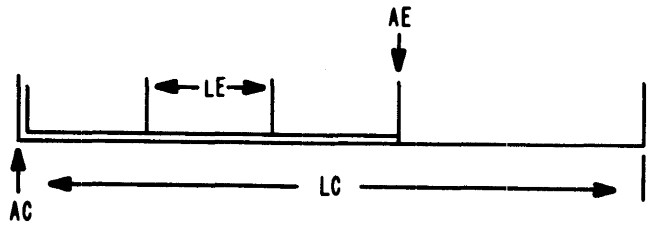
CONSTRUCT

1. AS.AC,LC,AE,LE + SW.AC,LC,AE,LE

2. COMBINE

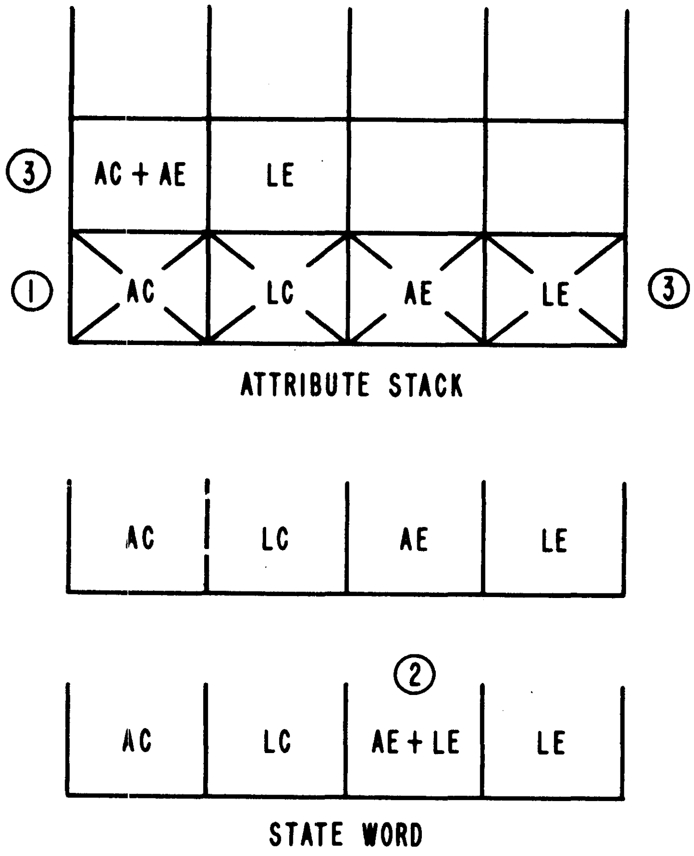
3. R

STACKS



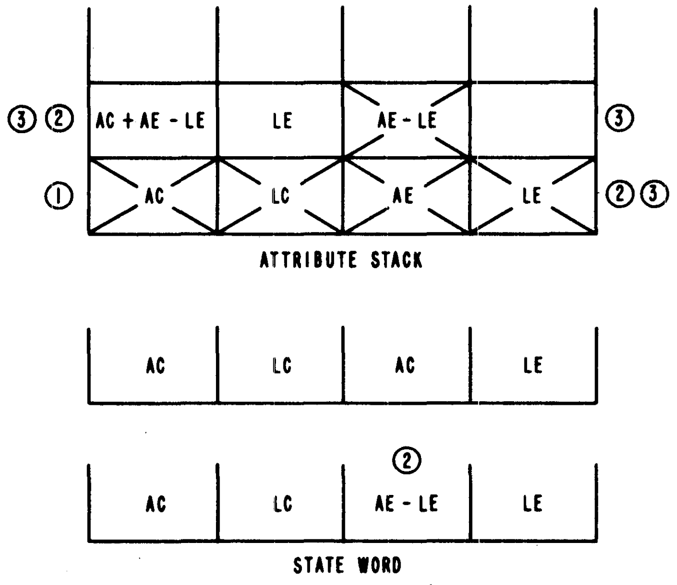
ENTER

1. AS.AC,LC,AE,LE ← SW.AC,LC,AE,LE
2. + SEQUENCE. SW
3. COMBINE
4. SSW
5. R



REMOVE

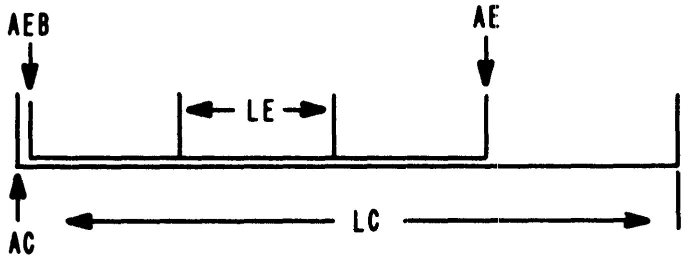
1. AS.AC,LC,AE,LE ← SW.AC,LC,AE,LE
2. - SEQUENCE
3. COMBINE
4. SSW
5. R



CONSTRUCT

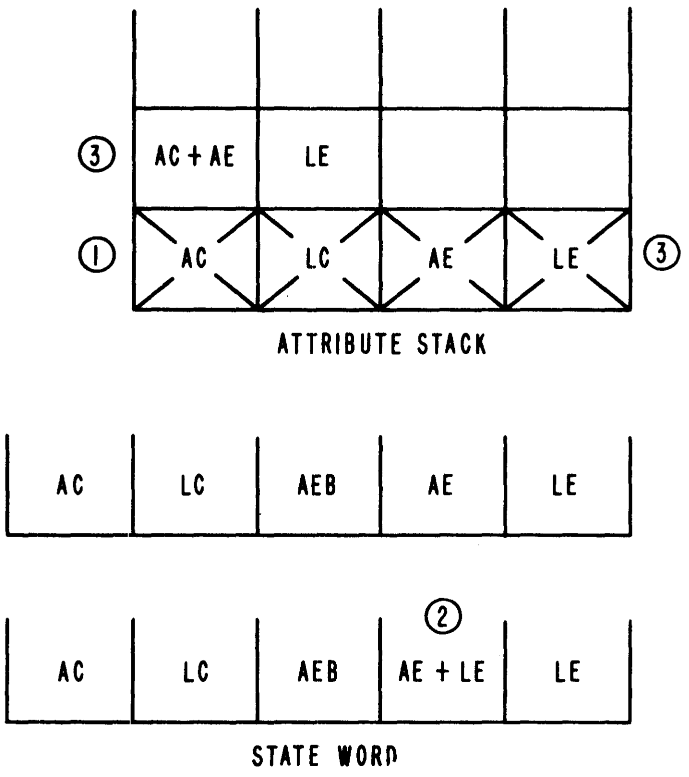
1. AS.AC,LC,AE,LE + SW.AC,LC,AE,LE
2. - SEQUENCE
3. COMBINE
4. R

STACK-VECTORS



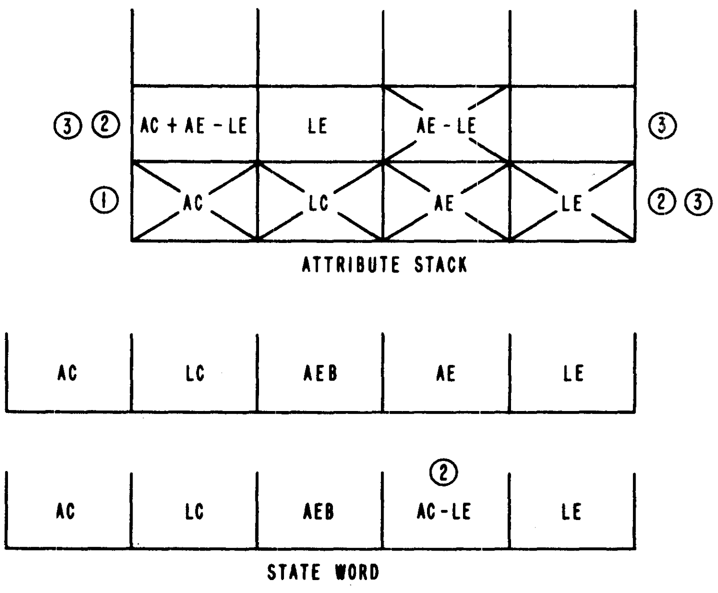
ENTER

1. AS.AC,LC,AE,LE ← SW.AC,LC,AE,LE
2. + SEQUENCE, SW
3. COMBINE
4. SSW
5. R



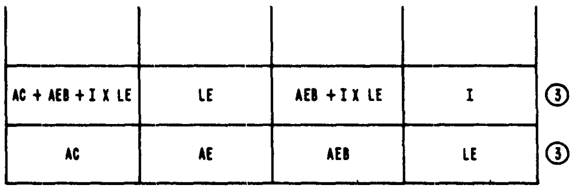
REMOVE

1. AS.AC,LC,AE,LE ← SW.AC,LC,AE,LE
2. - SEQUENCE
3. COMBINE
4. SSW
5. R

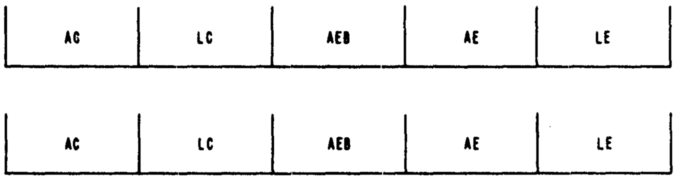


CONSTRUCT

1. AS.AC,LC,AE,LE ← SW.AC,LC,AEB,LE
2. AS. LE ← P.I
3. SELECT
4. COMBINE
5. R



ATTRIBUTE STACK

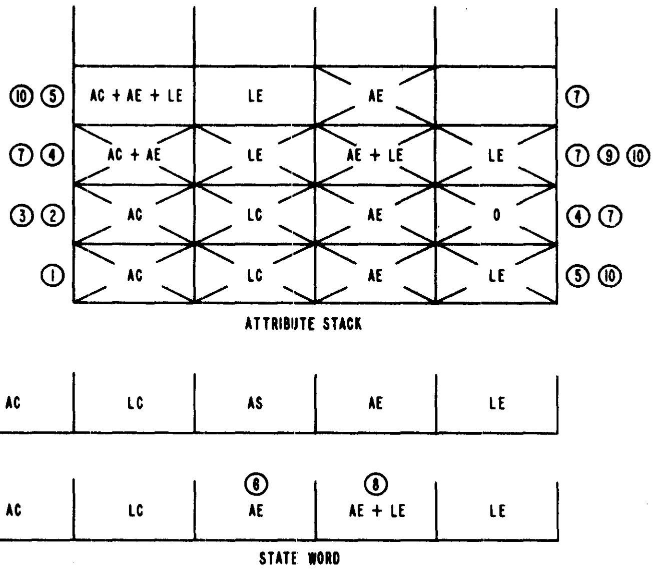


STATE WORD

PUSHDOWN STACK

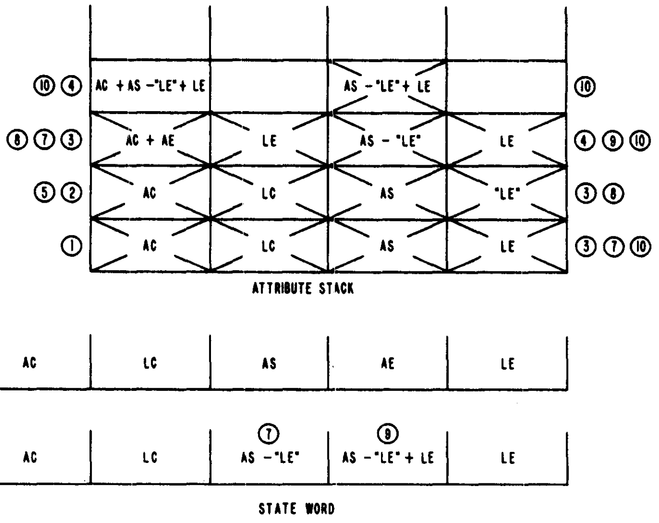
ENTER

```
1.  AS.AC,LC,AE,LE ← SW.AC,LC,AE,LE
2.  DUP AS,AC,LC,AE
3.  AS.LE ← 0
4.  CSP (AS.LE ← "LE",RET,NOP) <"LE" =
    ASO.AE - SW.AS> <PROG LENGTH = LE>
5.  EXCH AS,AE
6.  SW.AS ← AS.AE
7.  COMBINE
8.  SW.AE ← AS.AE
9.  MOVF (CSP)
10. COMBINE
11. SSW
12. R
```

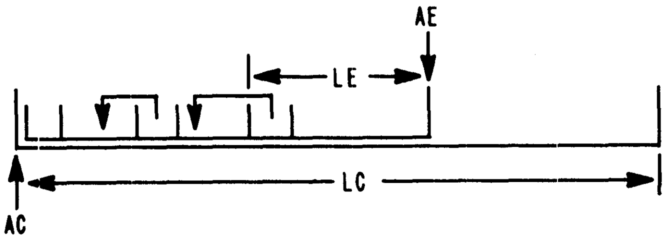


REMOVE

```
1.  AS.AC,LC,AE,LE ← SW.AC,LC,AS,LE
2.  DUP AS.AC,LC,AE
3.  COMBINE
4.  CALL
5.    AS.LE ← "LE"
6.    RET
7.  - SEQUENCE
8.  ASO.LE ← SW.LE
9.  + SEQUENCE
10. COMBINE
11. SSW
12. R
```

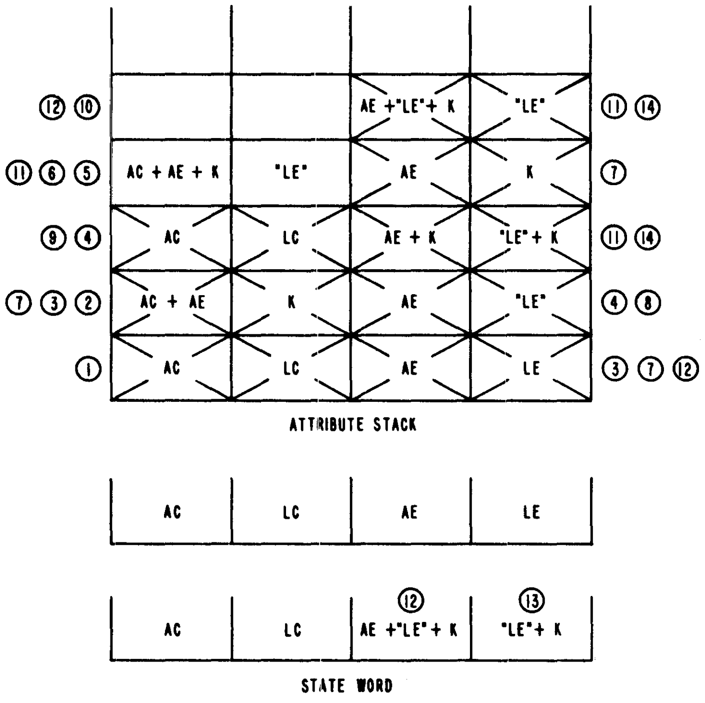


PUSHDOWN



ENTER

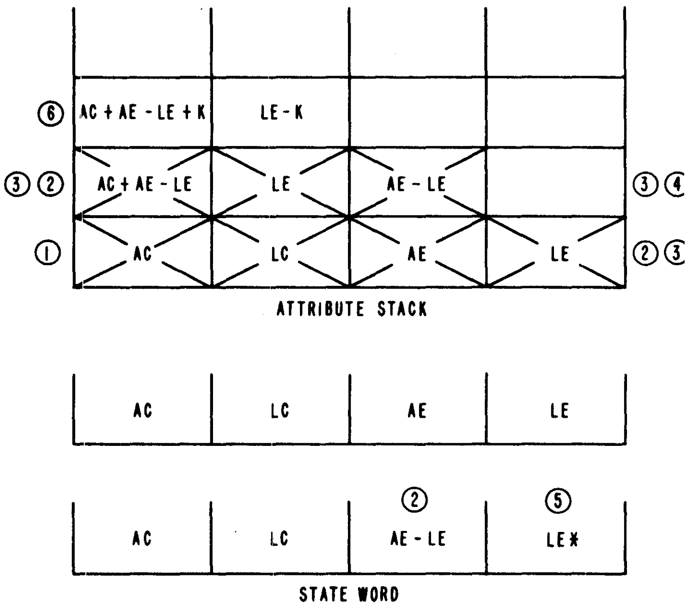
```
 1.  AS.AC,LC,AE,LE ← SW.AC,LC,AE,LE
 2.  DUP AS.AE
 3.  ASO.LE ← P."LE" <NEW ELEMENT LENGTH>
 4.  CSP  (SW.LE ← LE*, RTS) <PRESENT LE>
 5.  AS.LE ← K
 6.  AS.AE ← SW.AE
 7.  COMBINE
 8.  MOVE (CSP)
 9.  AS.AC,LC ← SW.AC,LC
10.  AS.LE ← P."LE"
11.  COMBINE
12.  ← SEQUENCE
13.  SW.LE ← AS.LE
14.  DEL AS.AE,LE
15.  SSW
16.  R
```



LE* EQUALS LE PRIME

REMOVE

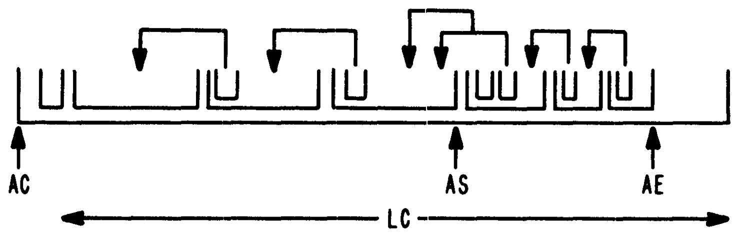
```
1.  AS.AC,LC,AE,LE ← SW.AC,LC,AE,LE
2.  - SEQUENCE
3.  COMRINE
4.  CALL
5.    SW.LE ← LE* <PREVIOUS LE>
6.    RTS
7.  SSW
8.  R
```



CONSTRUCT

```
1.  AS.AC,LC,AE,LE ← SW.AC,LC,AE,LE
2.  - SEQUENCE
3.  COMRINE
4.  CALL
5.    SW.LE ← LE* <PREVIOUS LE>
6.    RTS
7.  R
```

PUSHDOWN-PUSHDOWN



ENTER

```
1.   AS.AC,LC,AE,LE, ← SW.AC,LE,AE,0
2.   CSP(AS,"LE",RET)<LE = SW.AC - SW.AS>
3.   ASO.LE ← K
4.   SW.AS ← AS.AE
5.   COMBINE
6.   MOVE CSP
7.   CSP<AS.LE ← "LE", RET>
8.   AS.AC,LC,AE,LE ← SW.AC,LC,AE,0
9.   ASO.LE ← K
10.  ← SEQ
11.  ASO.LE ← K
12.  COMBINE
13.  MOVE CSR
14.  R
```

AC	LC	AS	AE	LE	LE

AC	LC	AE	AE + 2K	LS *	LE *

* EQUALS PRIME

STATE WORD

REMOVE

```
 1.  AS.AC,LC,AC,LE ← SW.AC,LC,AS,LS
 2.  DUP AS.LE
 3.  - SEQ
 4.  SW.AE ← AS.AE
 5.  COMBINE
 6.  CALL
       SW.LS ← RS <PREVIOUS LE>
       RTS
 7.  CALL
 8.    SW.LE ← <PREVIOUS LE>
       RTS
 9.  ASO.AE ← SW.AS
10.  COMBINE
11.  R
```

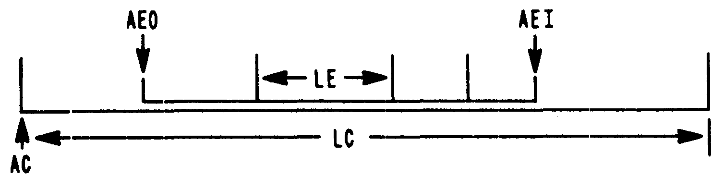
AC	LC	AS	AE	LS	LE

AC	LC	AS - (LS + K)	AS *	LS *	LE *

* EQUALS PRIME

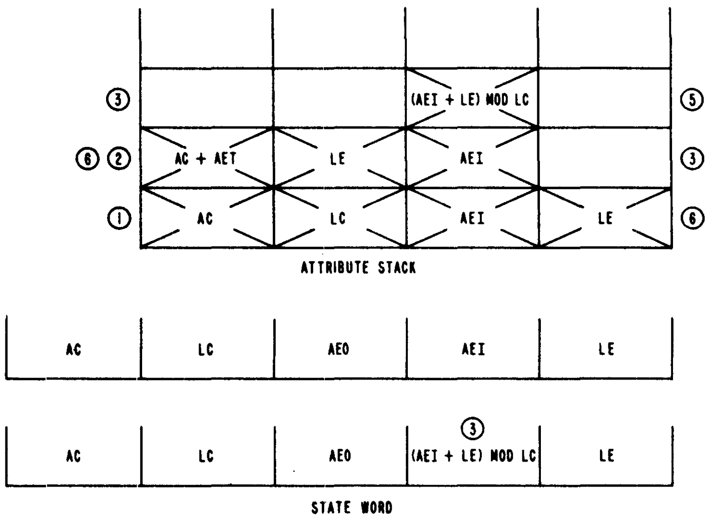
STATE WORD

QUEUES



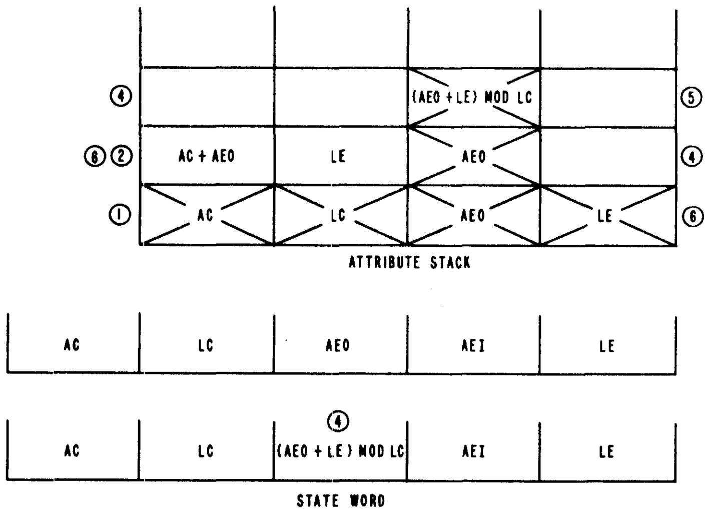
ENTER

1. AS.AC,LC,AE,LE ← SW.AC,LC,AEI,LE
2. DUP AS.AE
3. + MOD SEQUENCE
4. IF AS.AE = SW.AEO THEN FAULT (FULL)
5. DEL AS.AE
6. COMBINE
7. SSW
8. R



REMOVE

1. AS.AC,LC,AE,LE ← SW.AC,LC,AEO,LE
2. DUP AS.AE
3. IF AS.AE = SW.AEI THEN FAULT (EMPTY)
4. + MOD SEQUENCE
5. DEL AS.AE
6. COMBINE
7. SSW
8. R

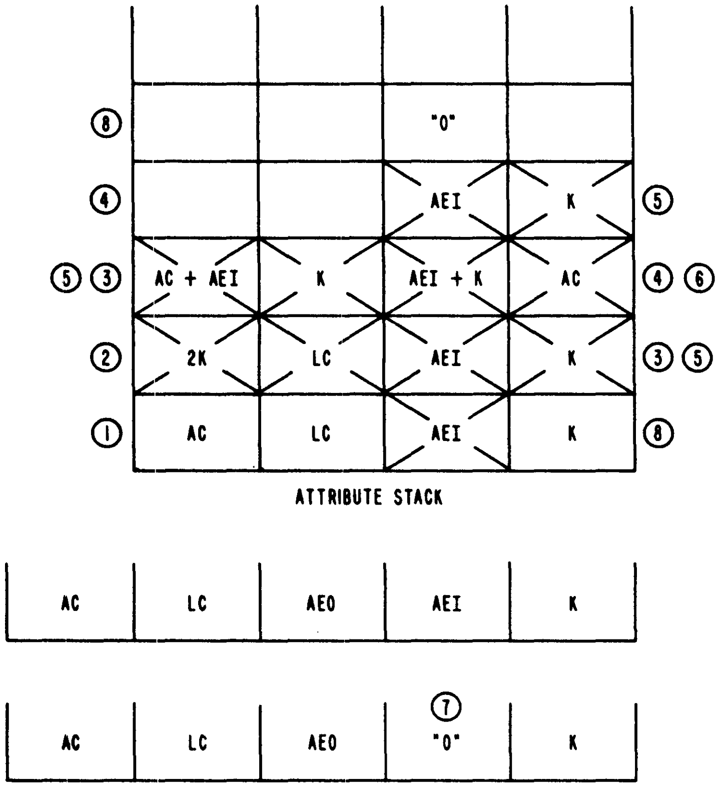


CONSTRUCT

```
1.   AS.AC,LC,AE,LE ← SW.AC,LC,AEO,LE

2.   DUP AS.AE

3.   IF AS.AE = SW.AEI THEN FAULT (EMPTY)

4.   ← MOD SEQUENCE

5.   DEL AS.AE

6.   COMBINE

7.   R
```

QUEUES WITH VARIABLE-LENGTH ELEMENTS

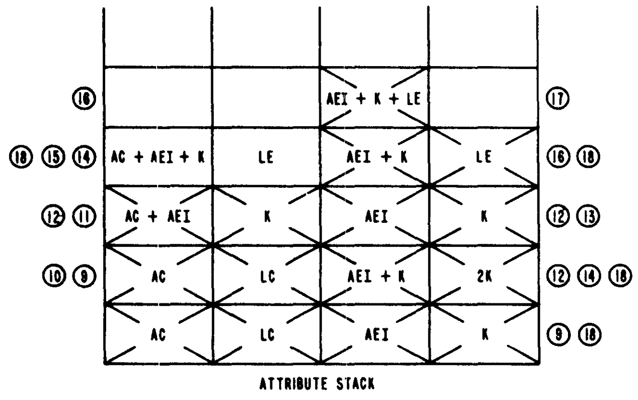
```
ENTER

1.  AS.AC,LC,AE,LE ← SW.AC,LC,AEI,K
    IF AEI ≥ AEO THEN BEGIN BEGIN
      IF AEI+K+P.LE+K > LC-"O" THEN
         BEGIN
           IF  K+P.LE  ≥  AEO-"O"  THEN
           FAULT (FULL)
           ELSE
             BEGIN
2.             DUP AS.AC,LC,AE,LE
3.             CSP (EOQ."O")
4.             ASO.AE,LE ← SW.AEI,K
5.             COMBINE
6.             MOVE (CSP)
7.             SW.AEI ← "O"
8.             ASO.AE ← "O"
             END;
         END;
      END
    ELSE
         IF AEI+K+P.LE ≥ AEO THEN FAULT
         (FULL);
```

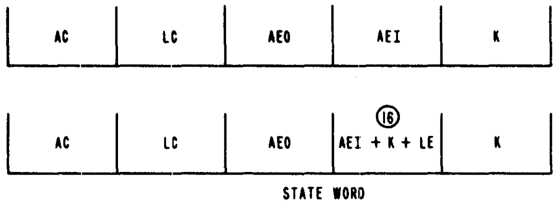


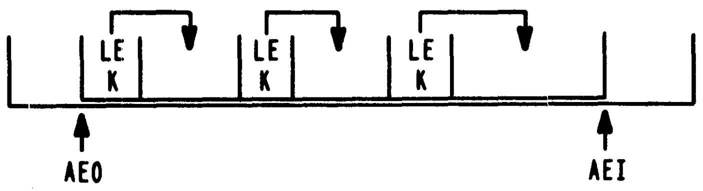
ATTRIBUTE STACK

STATE WORD

```
 9.   CSP (ASO.LE ← "LE",R)
10.   DUP AS.AC,LC
11.   AS.AE,LE ← SW.AEI,K
12.   COMBINE
13.   MOVE (CSP)
14.   ASO.LE ← P.LE
15.   DUP AS.AE
16.   ← SEQUENCE
17.   DELETE AS.AE
18.   COMBINE
19.   SSW
20.   R
```

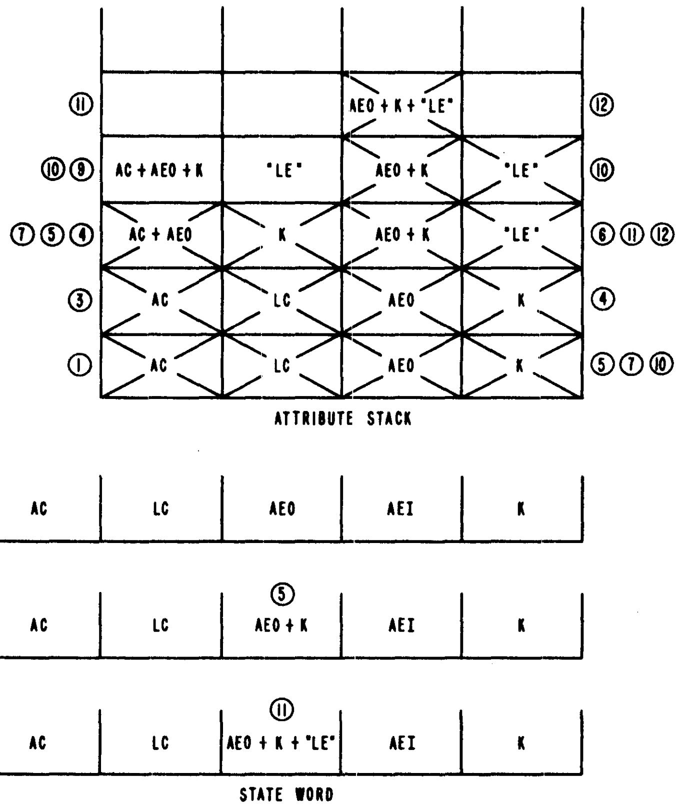


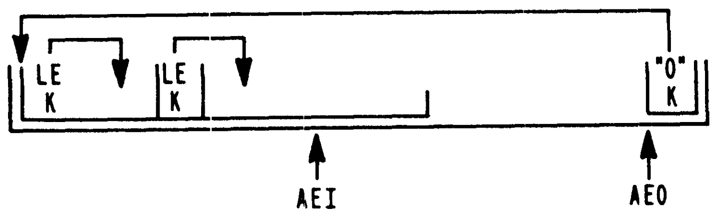




REMOVE

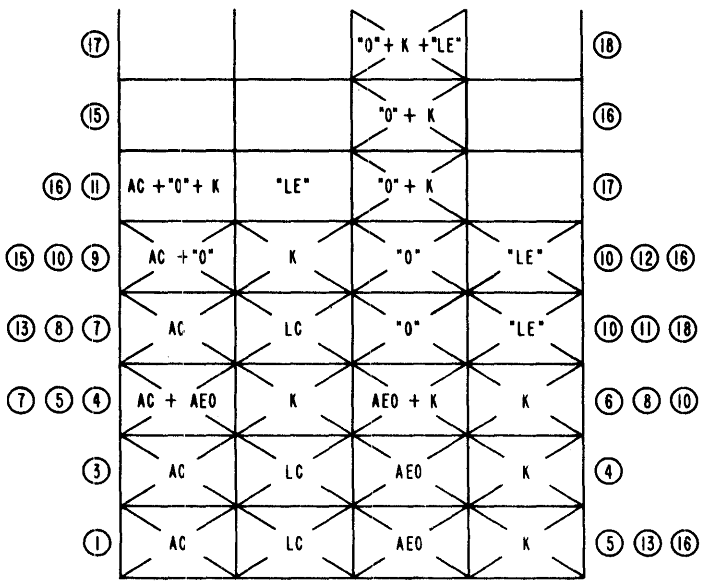
```
1.  AS.AC,LC,AE,LE ← SW.AC,LC,AEO,K
2.  TEST FOR EMPTY (AEO = AEI)
3.  DUP AS.AC,LC,AE,LE
4.  COMBINE
5.  + SEQUENCE
6.  CALL
7.    ASO.LE ← "LE"
8.    RET
9.  DUP AS.AE,LE
10. COMBINE
11. + SEQUENCE
12. DELETE AS.AE,LE
13. SSW
14. R
```



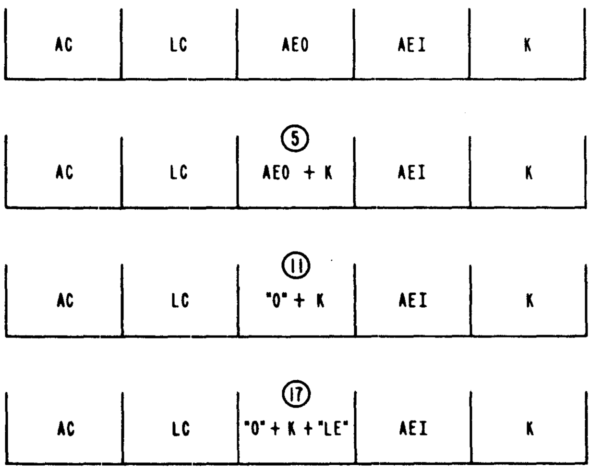


REMOVE ACROSS THE CONTAINER BOUNDARY

```
1.  AS.AC,LC,AE,LE ← SW.AC,LC,AEO,K
2.  TEST FOR EMPTY (AEO = AEI)
3.  DUP AS. AC,LC,AE,LE
4.  COMBINE
5.  + SEQUENCE
6.  CALL
7.    EOQ "O": DUP AS.AC,LC,LE
8.              ASO.AE ← "O"
9.              DUP AS.AE
10.             COMBINE
11.             + SEQUENCE
12.             BRANCH
13.      ASO.LE ← "LE"
14.      RET
15. DUP AS.AE,LE
16. COMBINE
17. + SEQUENCE
18. DELETE AS.AE,LE
19. SSW
20. R
```

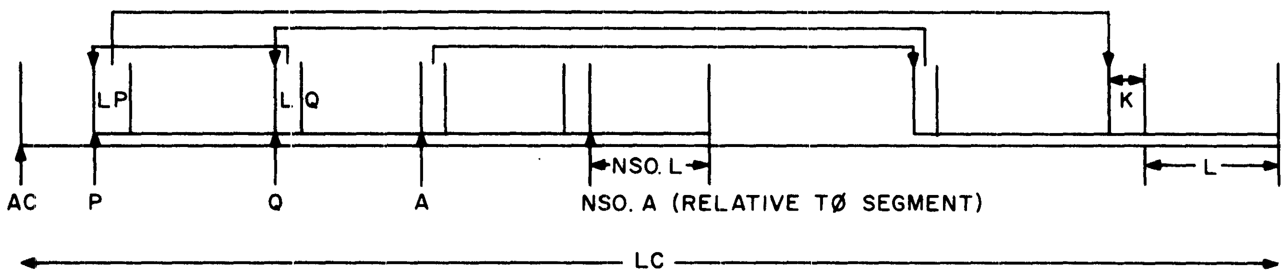


ATTRIBUTE STACK



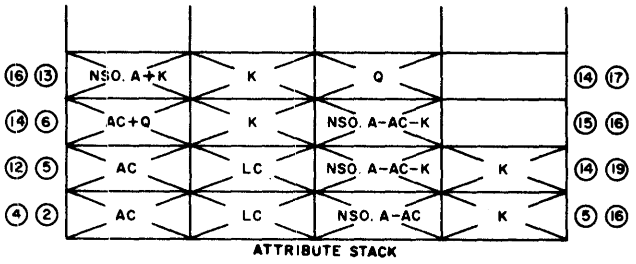
STATE WORD

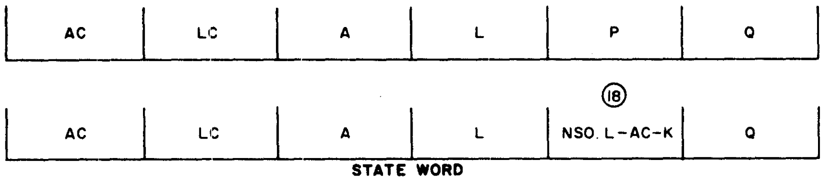
LINKED LISTS



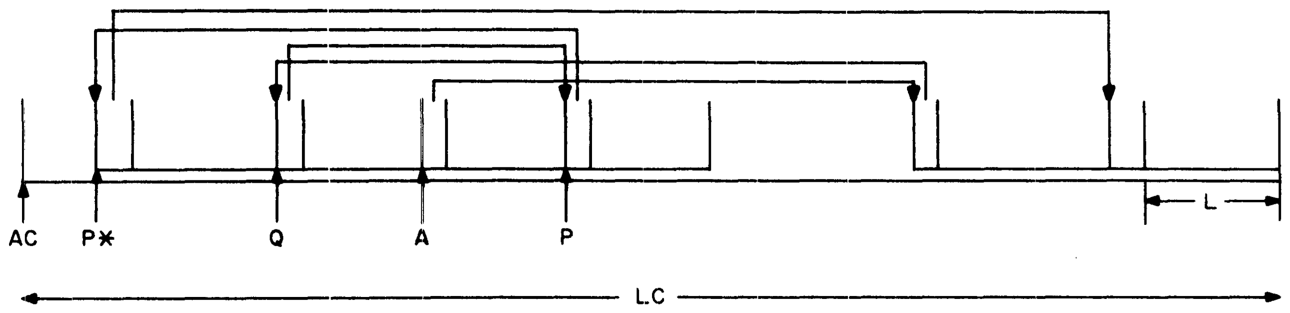
ENTER

```
1.  IF (NSO.SEG# ≠ AC.SEG#) OR
    (NSO.L ≠ SW.L) THEN FAULT (LIST)
2.  AS.AE ← NSO.A -AS.AC
3.  IF AS.AE < 0 OR SW.L > AS.LC
    THEN FAULT (LIST)
4.  AS.LE ← K
5.  -SEQUENCE AS.AE
6.  DUP AS.AE
7.  IF (SW.A = -1) OR (SW.Q = -1) THEN
      BEGIN
8.      COMBINE
9.      STORE FROM SW.A
10.     SW.A ← AS.AE
11.     SW.Q ← -1
      END
    ELSE
      BEGIN
```



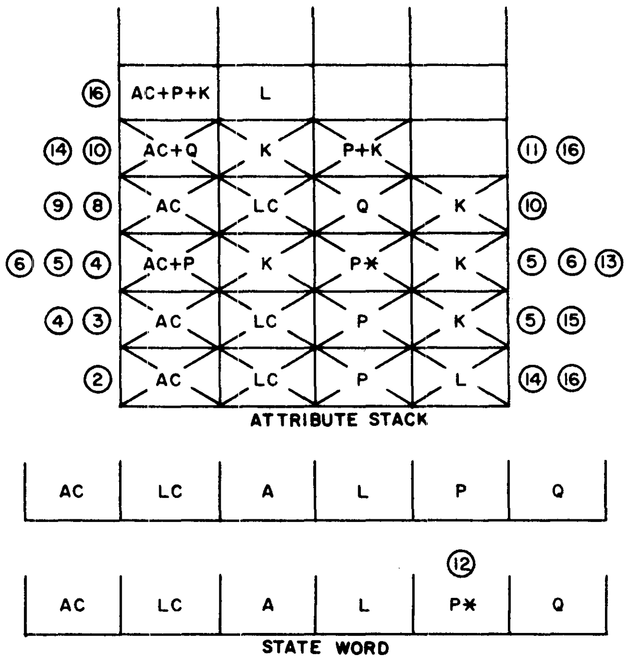


```
12.       DUP AS.AC, LC, LE
13.       AS.AE ← SW.Q
14.              COMBINE
15.              STORE FROM AS.AE
16.              COMBINE
17.              STORE FROM SW.P
        END
18.   SW.P ← AS.AE
19.   DEL AS.AE
20.   R
```



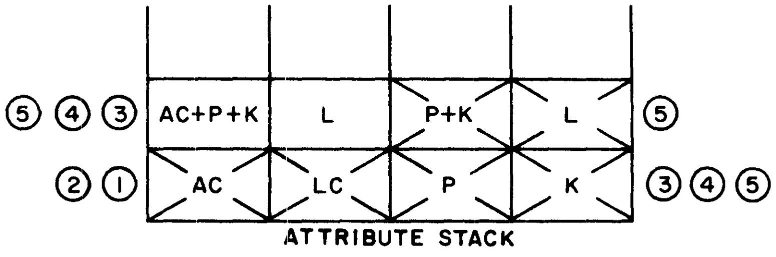
REMOVE

```
1.  IF (SW.A = -1) OR (SW.P = -1)
    THEN FAULT (LIST)
2.  AS.AE, LE ← SW.P, L
3.  AS.LE ← K
4.  DUP AS.AC, LC, AE, LE
5.  COMBINE
6.  FETCH TO AS.AE
7.  IF SW.P = SW.A THEN SW.A ← AS.AE
    ELSE
      BEGIN
8.      DUP AS.AC, LC, LE
9.      AS.AE ← SW.Q
10.     COMBINE
11.     STORE FROM AS.AE
      END
12. SW.P ← AS.AE
13. DEL AS.AE
14. + SEQUENCE AS.AE
15. DEL AS.LE
16. COMBINE
17. R
```



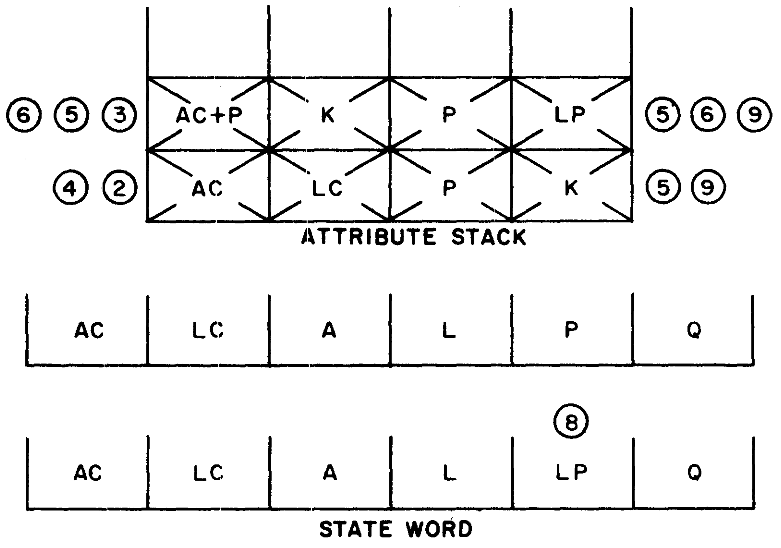
CONSTRUCT

```
1.  AS.AE ← SW.P
2.  AS.LE ← K
3.  ← SEQUENCE AS.AE
4.  ASO.LE ← SW.L
5.  COMBINE
6.  R
```



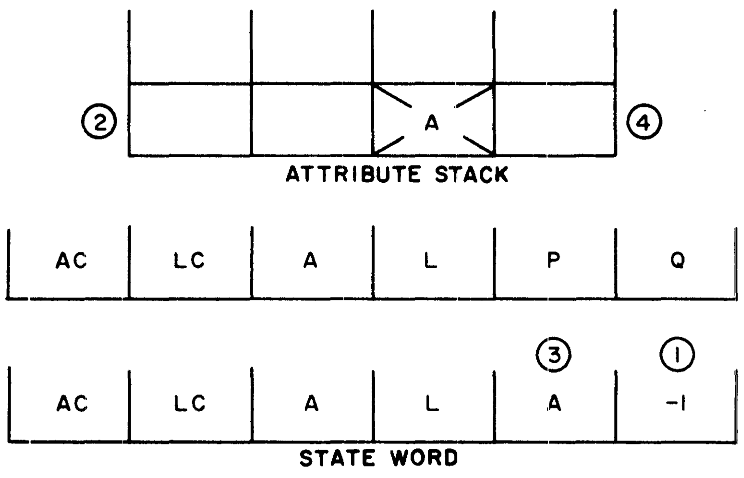
SEQUENCE

```
1.  IF (SW.A = -1) OR (SW.P = -1)
    THEN FAULT (LIST)
2.  AS.AE ← SW.P
3.  DUP AS.AE
4.  AS.LE ← K
5.  COMBINE
6.  FETCH TO AS.LE
7.  SW.Q ← AS.AE
8.  SW.P ← AS.LE
9.  DEL AS.AE, LE
10. R
```



RESET

```
1.  SW.Q ← -1
2.  AS.AE ← SW.A
3.  SW.P ← AS.AE
4.  DEL AS.AE
```



```
VARIABLE FIELDS
-------- ------

1.  AS.AE ← VSS <AE FIELD>

2.  REMOVE VSS

3.  AS.LE ← VSS <LE FIELD>

4.  REMOVE VSS
```