A Universal Dump Program for Minicomputer Software Debugging

C. C. Klimasauskas
Communications Systems Research Section

Low-cost minicomputers in wide variety are finding application in control and monitoring tasks ranging from laboratory testing to network operation. One significant problem which arises from this circumstance is that a significantly larger minicomputer system than is needed to perform the primary tasks must be acquired to do convenient software development. Consequently, work has been underway for some time to facilitate software development for the minimal configuration minicomputer using the Medium-Scale Xerox Data Systems Sigma 5. This article describes a general-purpose memory display program which runs on the Sigma 5 to dump memory images of minicomputer software to a printer or other man-readable device. The dump is formatted, as specified by control-card options, into machine-language instructions, character strings, or virtually whatever word/byte/field format is meaningful to the current problem.

I. Introduction

Low-cost minicomputers in wide variety are finding application in control and monitoring tasks ranging from laboratory testing to network operation. One significant problem which arises from this circumstance is that a significantly larger minicomputer system than is needed to perform the primary tasks must be acquired to do convenient software development. Consequently, work has been underway for some time to facilitate software development for the minimal configuration minicomputer using the Medium-Scale Xerox Data Systems Sigma 5. A general introduction to this activity may be found in Ref. 1. We expect ultimately to provide direct-coupled support where the minicomputer software is generated on the larger machine and is then loaded into the minicomputer and exercised via intercomputer communication links. One debugging tool which is useful, even if tedious to use, is a memory dump or display. This article describes a general-purpose memory display program which runs on the Sigma 5 to dump images of minicomputer software to a printer or other man-readable device. The dump is formatted, as specified by control-card options, into machine-language instructions, character strings, or virtually whatever word/byte/field format is meaningful to the current problem.
II. An Overview of the Processor

The universal dump program (UDP) looks at each record of the core dump (CD) as a single bit string. The user partitions this bit string into smaller units called words. Each word is partitioned further into fields (contiguous bit strings). Associated with each field is a field number and with that number a field conversion mode. Each word is indexed by an address which is specified dynamically by telling the UDP where to find the first address of the word or statically through specification. The address may be incremented by one or the number of bytes per word as specified through the addressing mode. The user has the option of indicating the conversion mode for the address, and the word, in addition to each field.

The first step in this process is description of the target machine. This phase generates tables which describe word size, byte size, field characteristics, conversion modes and all other data required to generate the dump. This phase produces syntax error messages and warnings. The following sections give a detailed description of the syntax of target machine description and listing format specification.

The next phase analyzes the tables for consistency and interpolates where possible to produce a complete set of consistent tables to drive the actual dump generator. This phase may find fatal specification errors which will prevent further execution. It also produces warning messages for nonfatal errors. A following section contains a complete description of all error messages.

Phase three is the actual routine for producing the dump listing. This program acts as a set of table-driven subroutines under the control of a single control routine. Figure 1 describes these program steps diagrammatically. The program is implemented in Xerox Data Systems META-SYMBOL on the Sigma 5 computer to be run under the Batch Processing Monitor. On the right-hand side of Fig. 1 is the overlay structure of the program. Each of the files to produce this resides on magnetic tape in the standard Xerox Data Systems 7-track labeled-tape format. The source file for each overlay is followed by its corresponding binary file (in parentheses).

III. Conventions for Syntax Definition

This section presents some general comments which describe the meaning of the constructions used in defining syntax.

(1) The bracket construction:
[ ] indicates that those items contained within the brackets are optional
[ ... ] indicates indefinite continuation of the optional construction in which this is imbedded.

(2) The brace construction:
\[
\begin{align*}
\text{option 1} \\
\text{option 2} \\
\end{align*}
\]
\[
\begin{align*}
\text{indicates that exactly one of the options indicated must be chosen.} \\
\end{align*}
\]

(3) Upper case literals and symbols:
These are key words and delimiters which must occur exactly as shown.

(4) Lower case phrases and letters:
These are representative of other constructions which must be supplied according to elsewhere defined syntax. For instance, the phrase “usgc” occurs many times in the description of the syntax. This phrase represents a numeric constant and has the syntax described at the beginning of Section IV.

(5) Blanks or spaces:
Unless specifically noted, blanks are not allowed between characters. Three exceptions:
(a) \( \bang \) signifies that exactly one blank is mandatory.
(b) \( \wedge \) signifies that one or more blanks are optional.
(c) The blank occurs as part of a string as defined for the syntax of the particular statement. For example, see “constant literal field” under PRTDEF in Section IV.

IV. The Syntax of Machine Description and Listing Format Specification

Before describing the syntax, some general comments about constants, control cards, and comment cards are presented.

There are two classes of constants: (1) explicit or specified and (2) implicit. The explicit or specified constant will be denoted by “usgc” and the implicit constant by “usgc.” The syntax of each of these is given below:

\[
\begin{align*}
\text{usgc:} & \quad \begin{align*}
\text{D} & \quad \text{decimal digit [decimal digit [...]]} \\
\text{O} & \quad \text{octal digit [octal digit [...]]} \\
\text{X} & \quad \text{hexadecimal digit [hexadecimal digit [...]]}
\end{align*}
\end{align*}
\]

\[
\begin{align*}
\text{[blank] decimal digit [decimal digit [...]]}
\end{align*}
\]
Control cards must always have an asterisk in column one. Blanks are not allowed in control cards except where specifically noted. If the definition of a control card requires more than one physical card, continuation cards must not have an asterisk in column one. All other cards will be printed exactly as punched and may be used for documentation purposes.

As we consider in detail the syntax of each statement, we will first define its purpose and make general comments, then give the actual syntax and describe the various options which are possible. As we proceed, it may be helpful to study the decks and resulting dumps shown in Figs. 2 and 3.

**A. WORD Specification (Target Machine Description)**

This statement simply describes the length in bits of each word of the target machine. Since these are partitions of a continuous bit string, they are referred to as p-words or “partition words.” (When each p-word is printed, first the address of the word is printed, then the contents, then the alphanumeric representation of the contents, and finally the user-defined fields). The syntax is as follows:

\[
\text{*WORD} = \text{usgc}
\]

where usgc is the number of bits per target machine word.

**B. BYTE Specification (Target Machine Description)**

This card is optional and is used for the purpose of computing the target machine address of the current p-word. This card must be used if byte addressing is specified. (See ADDRESS specification.) The syntax is as follows:

\[
\text{*BYTE} = \text{usgc}
\]

where usgc is the number of bits per target machine byte.

**C. ADDRESS Specification (Target Machine Descriptor and Listing Format Specification)**

This card defines the addressing mode of the target machine for the purpose of generating a pseudoaddress for the current p-word on the dump listing. Two modes are available: Byte and Word (default) addressing. Byte addressing increments the listing location counter by the number of bytes per word for each successive instruction. The conversion mode for the address can be optionally specified. The syntax is as follows:

\[
\text{*ADDRESS} = \begin{cases} \text{BYTE} \\ \text{WORD} \end{cases} \begin{cases} \text{DEC} \\ \text{OCT} \end{cases}
\]

**D. CHAR and SPECIAL Specifications (Target Machine Descriptors)**

The CHAR card defines the actual width of a character and the number of right-justified significant bits needed to decode the character. For instance, ASCII characters are eight bits wide; however, the right seven bits actually contain all the character information. Three widely used character codes have been incorporated into the program. Provision is made for user-defined character conversion codes. The syntax of the CHAR specification is as follows:

\[
\text{*CHAR} = \begin{cases} \text{ANASCII} \\ \text{ASCII} \\ \text{EBCDIC} \end{cases} \begin{cases} \text{[(usgc\textsubscript{1},usgc\textsubscript{2})]} \end{cases}
\]

where

usgc\textsubscript{1} = the width of a character in bits.

usgc\textsubscript{2} = the number of right-justified significant bits per character.

If SPCL is specified, the actual binary-to-character conversion table must be specified. This is done through the SPECIAL card. The syntax is as follows:

\[
\text{*SPECIAL} = \text{usgc}
\]

\[
\begin{array}{c}
\square \text{character string}_1 \\
\square \text{character string}_2 \\
\ldots \\
\square \text{character string}_n
\end{array}
\]

where

usgc = the number of characters per character string. Because of the physical limitations of a card, this is at most 79.
character string, = the jth contiguous string of characters in the binary-to-alphanumeric conversion vector. The jth character\(^1\) in the character conversion vector should be (a) a period, if no valid extended alphanumeric character corresponds to the target machine's internal binary representation of j or if the character is not printable, or (b) the actual character if the target machine's internal binary representation of j corresponds to that character. (For example: in EBCDIC, for j = 253, the internal binary representation of j is 1111 1101 and does not correspond to a valid character, so the 253rd character in the conversion vector is a period. For j = 240, the internal binary representation of j is 1111 0000. This corresponds to the character "@"; hence, the 240th character in the conversion vector is @.)

E. ROTATE Specification (Target Machine Preprocessing Descriptor)

In certain modes of operation, the halfwords may be swapped due to the method of transmission. This occurs, for instance, in the I/O processes of the PDP-11. It may also be desirable to rotate the contents of each word before breaking it into fields and decoding it. The ROTATE specification causes each p-word to be rotated right the number of bits specified. This occurs before any other procession on a record is done. The syntax is as follows:

*ROTATE:usge

where

usge is the number of bits to right-rotate each p-word before processing.

F. INSDEF Specification Card (Listing Format Specification)

INSDEF partitions the word into contiguous fields of bits. Each field has a corresponding field number which associates it with a print field (see PRTDEF). The syntax is as follows:

*INSDEF: \( f_1 f_2 f_3 \ldots f_r \)

where

\( f_i \) is the ith contiguous string of bits corresponding to a print field; \( f_i \) itself is a contiguous string of digits (all the same) in the form dddd...d, where d takes on values 0-9. When \( d = 0 \), the field is ignored. When \( d \neq 0 \), the field corresponds to the dth print field and will be connected for printing according to the attributes of the dth print field. If \( k(f_i) \) represents the number of bits in field

\[
i \sum_{i=1}^{n} k(f_i)
\]

must equal the number of bits per p-word. Hence \( f_1 f_2 \ldots f_r \) partitions the p-word into bit fields.

* is an optional string of blanks.

G. PRTDEF Specification Card (Listing Format Specification)

PRTDEF specifies the type of conversion to perform when printing the bit fields of a p-word. It allows for inclusion of unchanging literals, conversion mode specifies, and zero width field delimiters. The syntax is as follows:

*PRTDEF: \( f_1 f_2 \ldots f_r \)

where

\( f_i \) is a field of one of three types:

1. An unchanging literal field. This field is always printed exactly as it appears on the PRTDEF card, with the exceptions noted under the SKIP and NULL attributes described later. This field may not contain any of the following characters: M,X,O,D,B,C,L,:

2. A conversion mode field. The jth conversion mode field is the dth print field \( (j = d) \) described under INSDEF. This field describes the mode of conversion (X-hexadecimal, O-octal, D-decimal, B-binary, C-character, M-mnemonic, L-literal) and the number of print positions allocated to printing the field. (Note: the rightmost characters of the field are always printed. Truncation always occurs to the left.) The general format of the field is \( S_1 S_2 \ldots S_n \) where \( S_1 = S_2 = \ldots = S_n = S \), and \( S \)

\( 1 \)The index j starts at zero and goes through \( 2^{**} \) (number of significant bits/character) – 1 in steps of one.
defines the conversion mode as above; \( n \) is the width of the field in characters. For all but literal and mnemonic conversion, the maximum width is 12. For literals and mnemonics, the maximum is 4. There may be only one mnemonic field.

(3) A zero-width field delimiter. This is simply the character ‘:’. It serves to delimit two fields with the same conversion mode.

H. HEAD Specification Card (Listing Format Specification)

This card allows the user to put a heading at the top of each column of each page of listing. The header appears above the user-defined print fields. The first character after the colon appears immediately above the first character following the colon on the PRTDEF card (see PRTDEF). The syntax is as follows:

\[ *\text{HEAD}: \text{literal string} \]

where ‘literal string’ is the user-defined header.

I. PARAM Specification Card (Listing Format Specification)

The PARAM card allows the user to modify the conversion mode of a print field when it is undesirable to print it. This card also is used to enter the literal table if the literal conversion mode was specified on the PRTDEF card. The syntax is as follows:

\[
*\text{PARAM}: \text{usgc}_1 = \begin{cases} 
\text{SKIP} & \left( \begin{cases} \text{null} \\
\text{null} 
\end{cases} \right) \\
\text{null} & \left( \begin{cases} \text{null} \\
\text{null} 
\end{cases} \right)
\end{cases} \text{usgc}_2
\]

where:

usgc\(_1\) is the print field to be modified.

*SKIP* indicates that, following the current conversion mode field, blanks should be inserted to the end of the line and no further conversions made. This will occur if the SKIP-NULL condition is satisfied (see below); i.e., if the SKIP-NULL condition is satisfied, skip to the end of the line.

*NULL* indicates that the current conversion mode field and the preceding unchanging literal field should be blanked out. This will occur if the SKIP-NULL condition is satisfied.

SKIP-NULL condition: The relation operators in conjunction with usgc\(_2\) form the SKIP-NULL condition. There may be a linear combination of at most two relational operators. If the relation thus defined between the contents of the current field and usgc\(_2\) holds, the SKIP-NULL condition is ‘-0’. For the mnemonic conversion mode, the SKIP-NULL condition is implicitly satisfied if the operation code has not been assigned a mnemonic. In this case, an implicit NULL is performed, regardless of specification.

\[ \text{lit}_0, \text{lit}_1, \ldots, \text{lit}_n \] is the literal conversion vector for the current field if the literal conversion mode is specified. There are \( n = 2^r - 1 \) literals in this vector, where \( r \) is the bit width of the print field. Commas are used to delimit the literals. The \( i \)th literal corresponds to the internal binary representation of \( i \) (\( i \) takes on values 0 through \( 2^r - 1 \)).

J. OPS Specification (Listing Format Specification)

This card makes a one-to-one correspondence between machine operation codes and mnemonics. It builds the table for the mnemonic conversion field specified on the PRTDEF card. The syntax is as follows:

\[
*\text{OPS} = \begin{cases} 
\text{HEX} \\
\text{DEC} \\
\text{OCT}
\end{cases}
\]

\[ \text{ussc}_1, \text{literal string}_1 \]

\[ \text{ussc}_2, \text{literal string}_2 \]

\[ \vdots \]

\[ \text{ussc}_n, \text{literal string}_n \]
where

ussc₁ is the opcode for some instruction, ussc₂ is decoded as if it were in the mode specified by the ‘OPS=’ card.

literal string, is the mnemonic corresponding to opcode ussc₁.

This card must follow the PRTDEF card.

K. BIN Specification (Listing Format Specification)

This card defines the mode of conversion for printing the contents of the entire p-word. (When each p-word is printed, first the address of the word, then its contents, in the mode specified by the BIN card, then the alphanumeric equivalent of the contents, and finally the user-defined fields are printed.) The syntax is as follows:

\[ *\text{BIN} = \begin{cases} \text{HEX} \\ \text{DEC} \\ \text{OCT} \end{cases} \]

L. START and NEWLOC Specification (Listing Format Specification)

The purpose of START is to establish the initial value of the location counter for the address of the first p-word of the first record. If no NEWLOC card is present, the location counter is incremented for following records as if they were contiguous with the first. (This assumes that the core dump consists of more than one physical record.) This value may be explicitly stated or the program may be told where to find it in the first record. The syntax to do this is as follows:

\[ *\text{START} = \begin{cases} \text{ussc₁} \\ \text{REC}[+\text{ussc₂}]:f₁f₂f₃[\{±\text{ussc₁}\}] \end{cases} \]

where

ussc₁ is the initial value of the location counter for listing purposes. REC indicates that the initial location will be found within the record. ussc₂ indicates the offset of the word in the record which contains the initial value of the location counter. (The word accessed is ussc₂+1.) f₁f₂f₃ forms a mask for extracting the location from the word. f₁ and f₂ are strings of zeros indicating bits to ignore. Either may be null. f₃ is the mask for extracting the bits which represent the initial location. It consists of a string of Ns. The total length of the three strings must be the number of bits per p-word. ussc₃ is an offset to be added to the address extracted, before using it as the initial value of the listing location counter.

NEWLOC performs exactly the same function as START and has the same syntax. However, NEWLOC applies to the second and each succeeding record. If the record mode is specified, it is assumed that each record is treated exactly the same as if it were the first record and START where specified with these parameters. If an absolute starting location is given under NEWLOC, this applies to the second record only, and each succeeding record is treated as if it were contiguous to the preceding record and the location counter is incremented appropriately. The syntax of NEWLOC is as follows:

\[ *\text{NEWLOC} = \begin{cases} \text{ussc₁} \\ \text{REC}[+\text{ussc₂}]:f₁f₂f₃[\{±\text{ussc₃}\}] \end{cases} \]

where each symbol has the same meaning as for START.

Table 1 gives an incidence matrix which shows those cards that are necessary and any cards which they must precede. Those items in the left column must either precede, follow, or are optional with respect to the items across the top. This table should be helpful to the user for setting up a deck. If a card is optional, except in the presence of another, this is also noted. For instance, BYTE must precede the ADDRESS specification if used.

M. END Specification

The END card delimits the description deck and calls in the consistency analyzer. This must be the last card in the description deck. The syntax is as follows:

\[ *\text{END} \]

This concludes the discussion of the syntax of target machine description and listing format specification. Actual examples of each of the statements are shown in Figs. 2 and 3.

V. Program Limitations and Diagnostics

The Universal Dump Program is aimed at minicomputers, and this is reflected in some design limitations. Even so, it is very flexible, and should provide no insurmountable limitations to the user. Table 2 describes its characteristics and limitations.
The initial phases of the program produce a considerable quantity of diagnostics designed specifically to aid the user in quickly identifying and correcting control-deck errors. Because of this, and the philosophy that a program should be executed if at all possible, most diagnostics do not prevent continued execution of the program. In the initial syntax analysis section, diagnostics do not prevent continued execution, although a FAILURE will terminate processing of that statement. All consistency checks will be made on the tables, even though fatal errors in consistency may be discovered. Fatal errors prevent execution of the listing phase and termination will occur. Nonfatal errors (WARNINGS) have no effect on continued execution and are merely to call the user’s attention to unusual conditions. Table 3 lists possible errors, their probable cause, and fatality.

VI. Two Examples of Use

In this section we consider two examples to illustrate use of the program. The first decodes a dump of part of the monitor program for the XDS 930 computer, which is a second-generation machine that uses word addressing. The word is 24 bits, character code BCD, and usual listing mode OCTAL. The first record is a bootstrap which loads into location 2. Each following record is in absolute loadable binary form, with load location information in the second word. The instruction format is shown in Fig. 4. For a further description of the 930, see the XDS 930 Computer Reference Manual. Figure 2 illustrates how this information was used to generate a machine description deck. Figure 2 also shows the resulting dump.

The second example is from a considerably different machine, the PDP-11, manufactured by Digital Equipment Corporation. The PDP-11 is a medium-speed, third-generation minicomputer. It has 16-bit words, which are byte-addressable, and uses the ASCII character set. The usual listing mode is octal. The example here is a dump of an I/O routine in the standard PDP-11 absolute dump format. This format swaps the low- and high-order bytes in each word, making it necessary to reverse them before analyzing a record. This was done by means of the ROTATE command. Address information is contained in the third word of each record. The PDP-11 uses the first four bits of each word as an op-code, and for the op-codes '00' and '10', other bits in the word are used as modifiers. This condition is indicated by using the mnemonics SPCL and SPLB, respectively. The PDP-11 instruction format is shown in Fig. 5. For a further description of the PDP-11 instructions, see the PDP-11/20 Processor Handbook (Digital Equipment Corp.). Figure 3 shows the actual specification deck and dump listing for the I/O routine.

Reference

Table 1. The order of hierarchy of specification cards

O—may optionally precede or follow
P—must precede the statement on the top of the matrix
F—must follow the statement on the top of the matrix
blank—optional card

<table>
<thead>
<tr>
<th></th>
<th>WORD</th>
<th>BYTE</th>
<th>ADDRESS</th>
<th>CHAR</th>
<th>SPECIAL</th>
<th>ROTATE</th>
<th>INSDEF</th>
<th>PRTDEF</th>
<th>HEAD</th>
<th>PARAM</th>
<th>OPS</th>
<th>BIN</th>
<th>START</th>
<th>NEWLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORD</td>
<td>-</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BYTE</td>
<td>-</td>
<td>P^a</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADDRESS</td>
<td></td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>SPECIAL</td>
<td>F^b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROTATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INSDEF</td>
<td>F</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>O</td>
<td>P</td>
<td>P</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>PRTDEF</td>
<td>F</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>-</td>
<td>O</td>
<td>P</td>
<td>P</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>HEAD</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARAM</td>
<td>F</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>F</td>
<td>F</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>OPS</td>
<td>F</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>F</td>
<td>F</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>BIN</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>START</td>
<td>F</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td>O</td>
<td></td>
</tr>
<tr>
<td>NEWLOC</td>
<td></td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

^a BYTE must be specified before address if byte addressing mode is used.
^b SPECIAL must be used if and only if SPCL is specified for character type.

Table 2. Characteristics and limitations on the Universal Dump Program

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-word length (bits)</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>Binary, octal, decimal, hexadecimal, character field width (print positions)</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>Literal and mnemonic field width (print positions)</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>Location counter characters (print positions)</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Buffer size (bits) (current implementation)</td>
<td>$2 \times 10^5$</td>
<td>$5 \times 10^5$</td>
</tr>
<tr>
<td>Message</td>
<td>Meaning</td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------------------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>FAILURE: UNEXPECTED END OF LINE ENCOUNTERED</td>
<td>Syntax scan expected to see a character, but came to end of card first</td>
<td></td>
</tr>
<tr>
<td>WARNING: CARD WAS NOT COMPLETELY PROCESSED</td>
<td>Syntax scan ended and released card, yet nonblank characters still remain</td>
<td></td>
</tr>
<tr>
<td>WARNING: I/O ERROR-ABORT**</td>
<td>An irrecoverable I/O error has occurred and program has aborted (fatal)</td>
<td></td>
</tr>
<tr>
<td>WARNING: END CARD SUPPLIED—NO EXTRA CHARGE</td>
<td>End-of-file on card input was encountered before an *END card</td>
<td></td>
</tr>
<tr>
<td>FAILURE: ILLEGAL SIZE: MUST BE (1-32)</td>
<td>Attempt to redefine size of a p-word—ignored</td>
<td></td>
</tr>
<tr>
<td>FAILURE: CANNOT REDEFINE WORD SIZE—ERROR</td>
<td>Length of byte is zero, greater than the length of a p-word, or p-word length has not been defined</td>
<td></td>
</tr>
<tr>
<td>FAILURE: ILLEGAL BYTE LENGTH</td>
<td>Attempt to redefine byte length has been made</td>
<td></td>
</tr>
<tr>
<td>WARNING: WORD IS NOT BYTE MULTIPLE</td>
<td>The p-word is not an even byte multiple</td>
<td></td>
</tr>
<tr>
<td>FAILURE: BYTE LENGTH: UNDEFINED—IGNORED</td>
<td>The byte length must be defined before the ADDRESS card is used</td>
<td></td>
</tr>
<tr>
<td>FAILURE: ILLEGAL PARAMS: WORD OR BYTE EXPECTED</td>
<td>Self-explanatory diagnostic of ADDRESS card</td>
<td></td>
</tr>
<tr>
<td>FAILURE: ROTATE OPTION ALREADY IN EFFECT</td>
<td>Self-explanatory</td>
<td></td>
</tr>
<tr>
<td>FAILURE: ROTATE BITS ( &gt; = ) WORD LENGTH; OR WORD LENGTH = 0</td>
<td>Self-explanatory</td>
<td></td>
</tr>
<tr>
<td>FAILURE: ATTEMPT TO REDEFINE START/NEWLOC</td>
<td>Self-explanatory</td>
<td></td>
</tr>
<tr>
<td>FAILURE: ILLEGAL FIELD FMT ( () ) EXPECTED</td>
<td>Self-explanatory</td>
<td></td>
</tr>
<tr>
<td>WARNING: MASK AND WORD ARE UNEQUAL MASK ASSUMED RT JUSTIFIED</td>
<td>Self-explanatory in context of START/NEWLOC card</td>
<td></td>
</tr>
<tr>
<td>FAILURE: ATTEMPT TO REDEFINE INSTRUCTION</td>
<td>One or more INFDEF cards have already been processed</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE: MUST HAVE WORD LENGTH BEFORE INSDEF—YOU LOSE</td>
<td>WORD card must precede INSDEF statement</td>
</tr>
<tr>
<td>WARNING: DEFINED INS LENGTH AND ACTUAL INS LENGTH DIFFER</td>
<td>Self-explanatory in context of INSDEF statement</td>
</tr>
<tr>
<td>WARNING: ILLEGAL FIELD DEF CHARACTER—SCAN ENDED</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: ONLY ONE PRTDEF CARD ALLOWED—THIS IGNORED</td>
<td>One or more PRTDEF cards have been processed</td>
</tr>
<tr>
<td>FAILURE: UNKNOWN CHAR TYPE: ASCII, ASCII, BCD, EBCDIC, SPCL, EXPECTED</td>
<td>Self-explanatory in context of CHAR card</td>
</tr>
<tr>
<td>FAILURE: CANNOT REASSIGN CHARACTER TYPE</td>
<td>One or more CHAR cards have been processed</td>
</tr>
<tr>
<td>FAILURE: CHARACTERS ARE LONGER THAN ONE WORD</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: WORD IS NOT AN EVEN CHARACTER MULTIPLE</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: ILLEGAL CHARACTER, COMMA EXPECTED</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: MASK IS LARGER THAN CHARACTER</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: ILLEGAL CHAR, RT PAREN EXPECTED</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: NO SPECIAL CHARACTER TABLE NEEDED</td>
<td>SPCL was not an option on the CHAR card</td>
</tr>
<tr>
<td>FAILURE: MAX OF 79 CHAR/CARD EXCEEDED</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: NO CHAR NEEDED—WIDTH IS ZERO</td>
<td>SPCL card has occurred before CHAR card</td>
</tr>
<tr>
<td>FAILURE: UNEXPECTED END OF CARD</td>
<td>A control card was encountered before the character conversion vector was completed</td>
</tr>
</tbody>
</table>
### Table 3 (contd)

**Syntax errors (syntax analysis phase: nonfatal)**

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>FAILURE: ILLEGAL</td>
<td>Self-explanatory in context of BIN card</td>
</tr>
<tr>
<td>TYPE: HEX, DEC, OCT EXPECTED</td>
<td></td>
</tr>
<tr>
<td>FAILURE: UNIDENTIFIED</td>
<td>SKIP/NUL expected while scanning PARAM card—not found</td>
</tr>
<tr>
<td>PARAMETER JUST ENCLOSED</td>
<td></td>
</tr>
<tr>
<td>WARNING: MISSING PRTDEF FIELD—WIDTH OF 4 ASSUMED</td>
<td>PARAM card precedes the PRTDEF card</td>
</tr>
<tr>
<td>WARNING: IN incorrect NUMBER OF LITERALS</td>
<td>The number of literals on the PARAM card is less than needed to complete the binary-to-literal conversion vector</td>
</tr>
</tbody>
</table>

**Syntax errors (table analysis phase: FAILURES are fatal)**

<table>
<thead>
<tr>
<th>Message</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WARNING: ZERO BYTES/WRD—ONE ASSUMED</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: ZERO WIDTH CHARACTERS</td>
<td>An irrecoverable error occurred while processing the CHAR card, or field width specifications are in error</td>
</tr>
<tr>
<td>FAILURE: IMPROPER CHARACTER FIELD WIDTH</td>
<td></td>
</tr>
<tr>
<td>WARNING: NO CHR OPT—EBCDIC ASSUMED</td>
<td>No CHAR card in description deck</td>
</tr>
<tr>
<td>FAILURE: SPECIAL CHARACTER TABLE IS NULL</td>
<td>SPCL was specified on the CHAR card, but no SPECIAL card was validly processed</td>
</tr>
<tr>
<td>WARNING: TOTAL PRINT FIELD &gt; 33 CHR—TRUNCATED</td>
<td>The number of characters required to print the location, contents, and user fields exceeds 33 characters</td>
</tr>
<tr>
<td>FAILURE: NO PRTDEF CARD PRESENT</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: NO INSDEF CARD PRESENT</td>
<td>Self-explanatory</td>
</tr>
<tr>
<td>FAILURE: LIT/MNE, BUT NO LIT ENTRY—FIELD; XX</td>
<td>L or M specification on PRTDEF, but no ‘LIT’ on PARAM card or OPS card respectively</td>
</tr>
</tbody>
</table>

XX is the field number (1-9).
Fig. 1. Dump Program diagram
Fig. 2. An application of the Universal Dump Program to the XDS 930 computer
Fig. 2 (contd)
**CONSISTENCY TEST FOR TABLES**

<table>
<thead>
<tr>
<th>LOC LLLLLL AA BP SRC DST</th>
<th>LOC LLLLLL AA BP SRC DST</th>
<th>LOC LLLLLL AA BP SRC DST</th>
<th>LOC LLLLLL AA BP SRC DST</th>
<th>LOC LLLLLL AA BP SRC DST</th>
<th>LOC LLLLLL AA BP SRC DST</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RECORD NUMBER 001004</strong></td>
<td><strong>RECORD NUMBER 001005</strong></td>
<td><strong>RECORD NUMBER 001006</strong></td>
<td><strong>RECORD NUMBER 001007</strong></td>
<td><strong>RECORD NUMBER 001008</strong></td>
<td><strong>RECORD NUMBER 001009</strong></td>
</tr>
</tbody>
</table>

*Fig. 3. An application of the Universal Dump Program to the PDP-11 computer*
THE 930 INSTRUCTION:

0-3  |  FUNCTION |
-----|-----------|
0    | UNUSED    |
1-2  | TAG FIELD: BIT 1 SET, INDEXING IS USED  
     | BIT 2 SET, OPERATOR IS PROGRAMMED |
3-8  | OPERATION CODE |
9    | INDIRECT ADDRESSING BIT |
10-23| ADDRESS OF OPERAND |

Fig. 4. Instruction format for the XDS 930 computer

THE PDP-11 INSTRUCTION:

0-4  |  FUNCTION |
-----|-----------|
0-3  | OPERATION CODE |
4-6  | ADDRESSING MODE FOR SOURCE OPERAND |
7-9  | SOURCE OPERAND REGISTER |
10-12| ADDRESSING MODE FOR DESTINATION OPERAND |
13-15| DESTINATION OPERAND REGISTER |

Fig. 5. Instruction format for the PDP-11 computer