FORMAT OF BINARY PROGRAM INPUT TO DDT

L. P. Deutsch

Roger House

University of California, Berkeley

Document No. 30.40.30

Issued January 26, 1967

Contract SD-185

Office of Secretary of Defense

Advanced Research Projects Agency

Washington 25, D.C.

TABLE OF CONTENTS

1.0 Introduction

This paper precisely describes the format of binary programs which can be loaded by DDT. The immediate motivation for this description is the current programming of a new assembler, NARP, which will differ only slightly from ARPAS as far as source language and object language are concerned. However, looking further into the future, this description will enable any user to generate DDT-loadable output from any assembler, compiler, etc., that he may care to construct.

2.0 General Loading Scheme

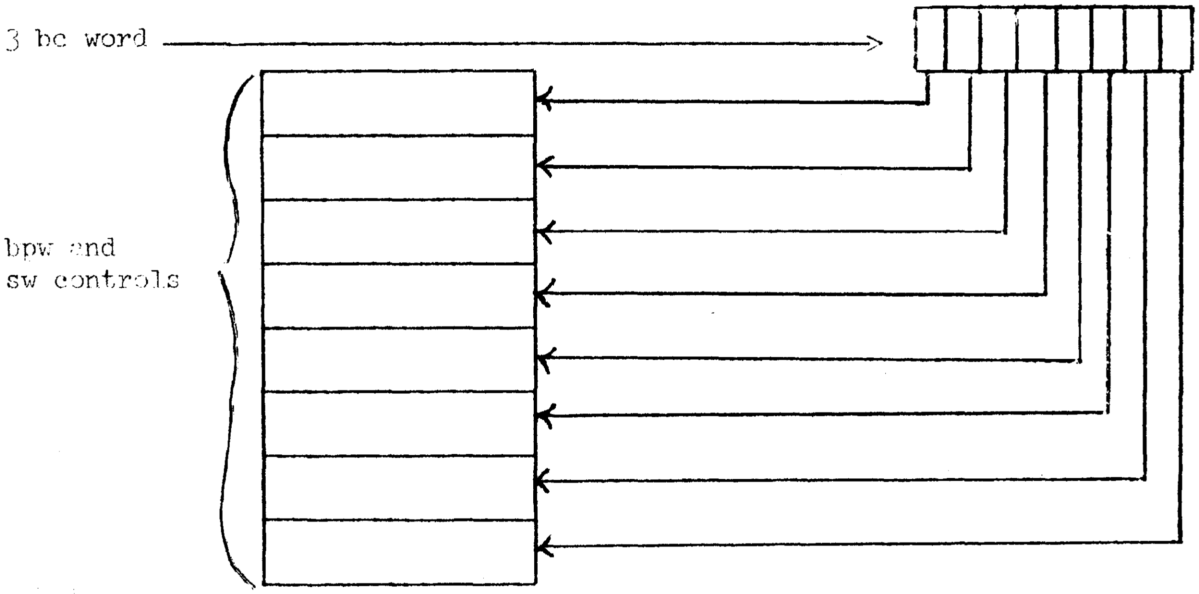
The binary program loaded by DDT consists of binary program words (bpw) which are placed at the address indicated by the location counter (lc) and control words which indicate a special action to be performed by DDT. A bpw is absolute, meaning it is to be placed in core as it is, or relocatable, meaning some multiple of the base address (ba) should be added to the bpw before placing it in core. Two types of relocation are distinguished: normal, meaning only the ba is added, and special (srel), meaning ba*rfactor is added, where rfactor (relocation factor) is a positive or negative integer different from 0 or 1. In the case of ARPAS output, a bpw may also be altered in other ways before being placed in core, namely if it is a literal reference or external symbol reference.

References to undefined symbols are handled somewhat differently, depending on whether the undefined symbol table (also called the external symbol usage table) occurs at the beginning of the binary program (as in ARPAS output) or at the end (as in NARP output). In the former case, DDT itself must build chains, linking all references to a given symbol together. When the symbol is defined this chain is followed and all links are replaced by the value of the symbol (a process referred to as "fixing up" or "patching up"). In the latter case, NARP has already constructed the chains, and DDT need only link up the chains in the various packages.

3.0 Format of Binary Programs

3.1 General

The basic unit of binary program is the binary program block (bp block), a sequence of nine words. The first word of each block contains eight 3-bit codes (3bc) indicating how the eight remaining words of the block should be treated. The last eight words are either bpw or single-word controls (sw controls), the latter of which are not loaded into core like bpw but direct DDT to take some special action.



In certain cases the normal sequence of output is interrupted so that a bp block of less than nine words is output, followed by a variable-length block (vl block) headed by a multiple-word control (mw control). These cases are discussed below.

3.2 3-Bit Codes

The actions specified by the 3bc are discussed below:

0	absolute:	Load bpw just as it is
*1	ext 14:	Bits 10-23 of bpw indicate an external symbol. Replace bits 10-23 by the value of the external symbol, and if the external symbol is undefined, replace its value by lc.

* Appears only in ARPAS output.

2 rel14: Add ba to bpw (mod 2^{14})

3 srel: Add ba*rfactor to bpw (mod 2^{24})

4 control: The word is not a bpw but a control word. It if is a sw control then take the specified action (see below) and continue. If it is a mw control then abandon the current bp block, process the following vl block as specified by the control word, and expect a new 3bc word following the vl block.

*5 ext24: Same as ext14 above, but replace all 24 bits of bpw instead of just bits 10-23.

6 rel24: Same as rel14 but addition in mod 2^{24}.

*7 lit ref: Add ba to bpw (mod 2^{14}). Used in place of code 2 for instructions which refer to literals.

3.3 Control Words

3.3.1 Single-Word Controls

The following controls consist of one word only; they occur in bp blocks mixed in with bpw. (The shaded areas are not used; they are all zeroes.)

0 - alter lc

0	▨	a
0-2	3 9	10 23

lc←lc+a; comment: all changes of lc are mod 2^{14};

1 - pop link

1	b	▨	a
0-2	3 8	9	10 23

lc←lc+a; b+64← BRU lc;

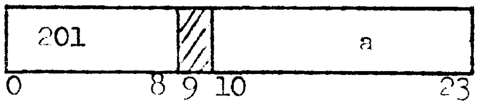
200 - end prog

200	▨
0 8	9 23

terminate loading;

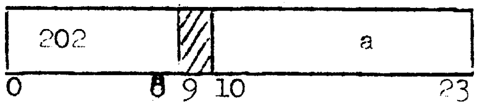
*Appears only in ARPAS output.

201 - lit org



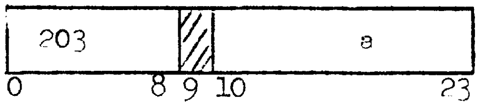
lit org ← a+ba;

202 - spec rfactor



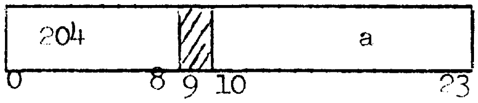
extend bit 10 (i.e., leftmost bit of a) to bits 0-9 and place the result in rfactor.

*203 - fixup lc



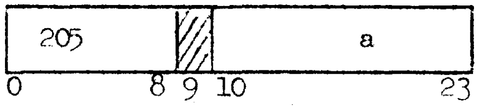
replace every link of the chain starting at a+ba by lc; comment: a chain ends when a link points to itself.

*204 - fix 14



lc ← lc-1; replace every link in the chain starting at a+ba by (lc); comment: replacement is done mod 2^{14};

*205 - fix 24



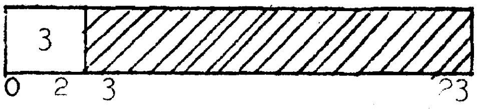
same as 204 but replacement is mod 2^{24};

3.3.2 Multiple-Word Controls

The following controls consist of a single word indicating the type of vl block which follows the word. All vl blocks end with a word of all ones. Following the all-ones word is a new 3bc word. (The shaded areas are not used; they are all zeroes.)

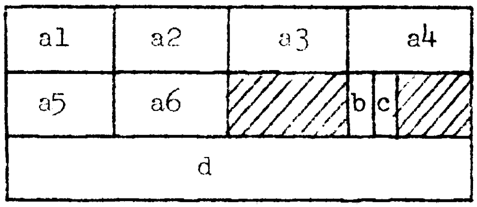
* Appears only in NARP output.

3 - opcode definition:



each 3-word entry of the following vl block is an opcode definition; there are two formats, depending on whether the opcode is output by NARP or ARPAS:

NARP:



a1 - a6: character of the opcode (left-justified with blank fill on the right)

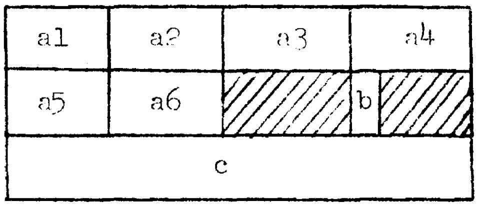
b=1 : indicates opcode def is in NARP format

c=0 : operand optional

=1 : opcode does not take an operand

d : opcode value

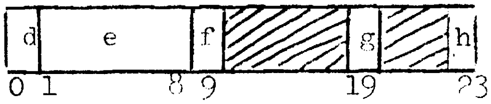
ARPAS:



a1 - a6: characters of the opcode

b=0 : indicates ARPAS format

c : information for computing opcode; there are two formats:



d=1: ARPAS class 1

e : opcode value

f : sign bit of opcode

g=0: operand optional

=1: operand required

h=0: 14-bit operand

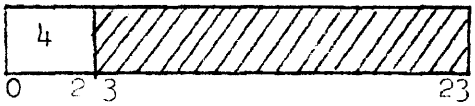
=1: 9-bit operand

d	e

d=0: ARPAS class 2 (no operand)

e : opcode value

4 - external symbol definition



each 3-word entry of the following vl block is an external symbol definition; there are two possible formats, the second of which only occurs in NARP output:

normal

a1	a2	a3	a4
a5	a6	b ▨	c ▨
d			

a1 - a6: characters of the symbol

b=0 : value of symbol is absolute

=1 : value is to be relocated (mod 2^{24})

c=0 : indicates normal relocation

d : value of the symbol

special relocation

a1	a2	a3	a4
a5	a6	b ▨	c ▨
e			
d			

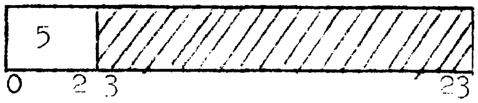
a1 - a6: characters of the symbol

b=0,c=1: indicates special relocation

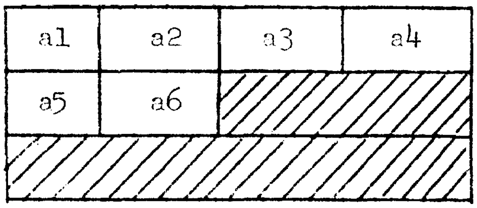
e : special relocation factor

d : value of the symbol; it should be relocated **mod** 2^{24} using the special relocation factor.

5 - ident

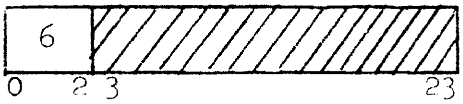


each 3-word entry is an identification symbol:



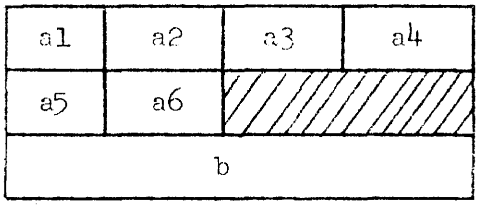
a1 - a6: characters of the symbol

6 - undefined symbol table



each entry is a 3-word undefined symbol or a variable length Polish string representing an undefined expression:

symbol:

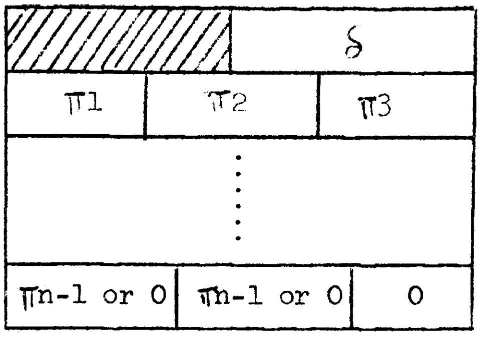


a1 - a6: characters of the symbol

b ≠ -1 : NARP: relative address of the start of a chain

ARPAS: not relevant

= -1 : indicates that symbol only appeared in expressions expression:



ς : patchup chain

π1 - πn: a Polish string composed of the following elements:

000xxxxx Operator. Presently there are the following:

xxxxx=0	end of string
1	@ (NOT)
2	unary -
3	+, keep bits 0-9 unchanged
4	+
5	-
6	*
7	/
10	& (AND)
11	! (OR)
12	% (EOR)
13	↑
21	<
22	>
23	#
24	=
25	<=
26	>=

01xxxxxx Small constant. Value is xxxxxx-40b, e.g.

142=>2 and 134=>3.

001srrnn; (01xxxxxx) nn+1 of these; (xxxxxxxx) see <u>rr</u> below.

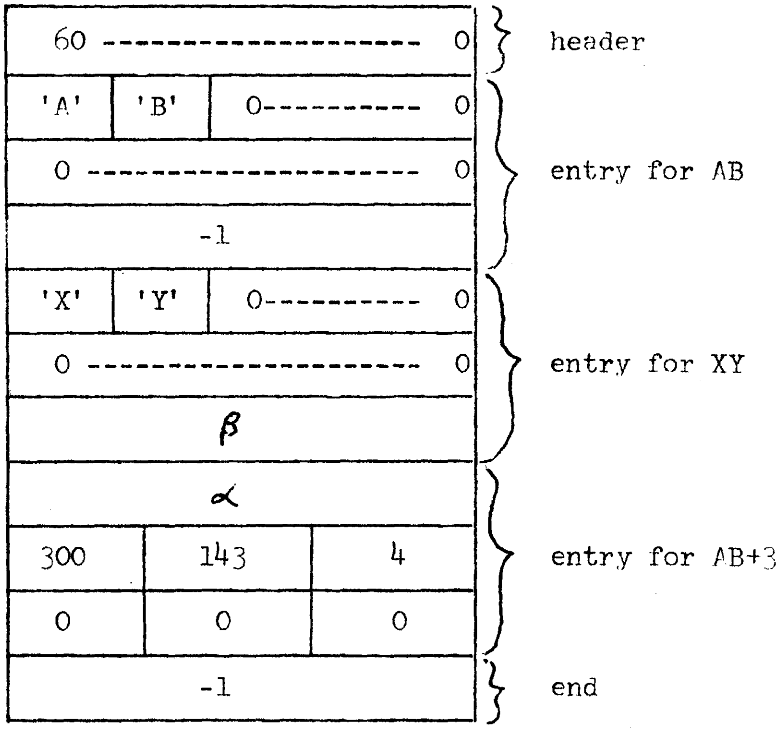
Large constant s=1 means take negative of value. nn+1 is the number of following data bytes. rr designates relocation:

	rr=00	absolute
	01	relocate+1
8-bit bytes	10	1 byte of special relocation, biased by 200b
	11	3 bytes of special location.

11xxxxxx Short symbol. See below.

10xxxxxx; 11xxxxxx Long symbol. xxxxxx or xxxxxxxxxxxx is an index within the usage table, with the first entry numbered 0. An example follows.

Suppose at location α there is the instruction LDA AB+3, where AB is not defined or used anywhere else in the program. Suppose also at location there is STA XY, which is the only use of the undefined symbol XY. Then the usage table might appear as follows:

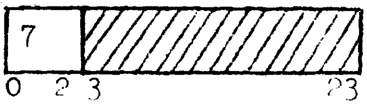


60 -------------------- 0			header
'A'	'B'	0---------- 0	entry for AB
0 -------------------- 0			
-1			
'X'	'Y'	0---------- 0	entry for XY
0 -------------------- 0			
β			
α			entry for AB+3
300	143	4	
0	0	0	
-1			end

300 = symbol 0 (AB)
143 = constant 3
4 = + operator

All fixups of external symbols are done mod 2^{24}.

7. local symbol definition



a block of the same format as for external symbol definitions follows, only each symbol is local instead of external.

3.4 Overall structure

The first input to DDT should be a 3bc word. After that the input depends on whether NARP or ARPAS output is being processed.

3.4.1 NARP

1. Body of the program
   - a. ident
   - b. alter lc
   - c. bpw
   - d. pop link
   - e. special rfactor
   - f. fixup lc
   - g. fixup 14
   - h. fixup 24

   in any order (b–h)

2. Literals
   - a. literal table origin
   - b. special rfactor
   - c. fixup lc
   - d. fixup 14

   in any order

3. Undefined symbol table

4. Definitions
   - a. opcodes
   - b. external symbols
   - c. local symbols

5. End of Program

3.4.2 ARPAS

1. Ident
2. Undefined symbol table
3. Literal table origin
4. Body of program
   - a. binary program words
   - b. pop links
   - c. external symbol definitions

   in any order
5. Local symbol definitions
6. End of program

APPENDIX I

List of Abbreviations

ba	base address
bp block	binary program block
bpw	binary program word
fixup	process of following a chain and replacing each link by some value
lc	location counter
mw control	multiple-word control
patchup	see fixup
rfactor	relocation factor
srel	special relocation
sw control	single-word control
vl block	variable length block
3bc	3-bit code

APPENDIX II

One-Page Summary:

Binary Program Input to DDT

3-bit codes

0	absolute	
1	external (mod 2^{14})	A
2	relocate (mod 2^{14})	
3	special relocation	
4	control word	
5	external (mod 2^{24})	A
6	relocate (mod 2^{24})	
7	literal reference	A

binary program blocks: 3-bit code word followed by binary program words and single-word controls

variable length blocks: multiple-word controls followed by a table

A: occurs only in ARPAS output

N: occurs only in NARP output

control words

0	alter location counter	
1	pop link	
200	end program	
201	literal table origin	
202	special relocation factor	
203	fixup with location counter	N
204	fixup (mod 2^{14})	N
205	fixup (mod 2^{14})	N
3	opcode definition	
4	external symbol definition	
5	ident	
6	undefined symbol table	
7	local symbol definition	