DECUS

PROGRAM LIBRARY

DECUS NO.	8-47
TITLE	ALBIN, A PDP-8 LOADER FOR RELOCATABLE BINARY PROGRAMS
AUTHOR	J. L. Visschers, P. U. ten Kate and M. A. A. Sonnemans
COMPANY	Instituut Voar Kernphysisch Onderzoek (IKO) Amsterdam, The Netherlands
DATE	1966
SOURCE LANGUAGE	

ALBIN, A PDP-8 LOADER FOR RELOCATABLE BINARY PROGRAMS

DECUS Program Writeup | DECUS No. 8-47

Abstract

As part of the current design of a PDP-8 real-time monitoring system, a simple method has been obtained to construct relocatable binary formatted programs, using the PAL III Assembler. Allocation of these programs can be varied in units of one memory page (128_{10} registers). While loading an ALBIN program the actual absolute addresses of indicated program elements (e. g. the keypoints of subroutines) are noted down in fixed program-specified locations on page 0. In order to make a DEC symbolic program suitable for translation into its relocatable binary equivalent, minor changes are required; which, however, do not influence the length of the program. Due to its similarity to the standard DEC Binary Loader, the ALBIN loader is also able to read-in normal DEC binary tapes. The loader is presented here in its simplest form, although the loading method, in a slightly advanced manner, will be used for automatic "piling while loading" of arbitrary sequences of more or less independent programs. In the form to be described, ALBIN requires 122_{10} locations, including the RIM loader. Piling up in core memory of ALBIN programs stored on conventional or DECtape can be achieved using the same method with minor modifications.

Introduction

When a selected set of programs have to be executed in one run, this normally means full assembling of the set stacked together in the desired combination although a binary equivalent of each program is present. This inconvenience becomes a serious limitation in our application, the design of a parallel multiprocessing system. We, therefore, constructed a loading method which enables the loading of an arbitrary sequence of binary program tapes.

This implies that a binary program tape has to be loaded from a relocatable starting address. The problem with relocation is that the binary form of some program elements depends on the starting address of the program. If we, however, restrict ourselves to relocations in units of one memory page, and if we assume that the page 0 parts of relocatable programs are not to be relocated, the only changeable binary program elements are absolute (12 bits) addresses, referring to a relocatable program point.

Description of ALBIN - Format

For a binary loader, it is rather difficult to differentiate between 12-bit addresses and instructions unless the loader receives additional information from the assembler.

A DEC binary format tape consists of origin settings and memory words. A relocatable binary tape should not contain any origin setting, since this information changes with every loading. Instead, one may insert the origin information via the switch register. In doing so, the origin setting symbol can be used for "key setting," which then implies that "*ADRES" means: set the address of the next instruction in ADRES.

In this way we can handle the absolute (12 bit) addresses. If the register REG should contain an absolute address referring to a relocatable program point AA, we place just above AA the expression "*REG". While loading, the ALBIN loader then places the right value of AA in REG, for example:

```
                ______           ______
                ______           ______

       REG, AA                   *REG
                ______       AA, ______
```

The condition must be fulfilled that REG is an address in the same relocatable program as *REG, or that REG is fixed (c.q. page 0) address. Therefore, it is impossible to have references between independently relocatable programs over a key address which belongs to one of them. References between independently relocatable programs, therefore, should occur over fixed key addresses at page 0.

Applications

One of the essential advantages of a relocatable loading system is the possibility of automatic piling of programs. In that case the above mentioned switch register setting is replaced by a register in the loader itself which, after each loading, points to the next free page.

Applying a "chaining" principle, one can also load and "remove" less frequently used routines while running (segmenting). This is particularly useful if magnetic tape input is available.

This work is preparatory for a software project performed under the direction of R. van Dantzig.

Directions of Use

1. Add to the permanent symbol table of PAL III the symbols:

   INPUT = 7545
   ALBIN = 0

2. The program to be loaded must exist of one or more blocks separated by leader trailer.

   A block can be either a BIN or an ALBIN block. At the beginning of each block, the loader is automatically set into BIN mode. It can be brought into ALBIN mode by:

   *INPUT
   ALBIN

The loader remains in ALBIN mode until the next block end ($).

3. The expression *ADRES means in BIN mode: set ADRES in the location counter (origin setting) but in ALBIN mode: set the contents of the location counter in ADRES ("key-setting")
(The location counter contains the address where the next instruction will be placed.)
Due to the fact that the PAL III Assembler does not differentiate between BIN and ALBIN mode, the expression *ADRES in an ALBIN block always has to be followed by a restoring expression, which restores the location counter of the assembler, for example:

```
        ______
        ______
AA,     ______
        *ADRES
        *AA + 1

        ______
        ______
```

The loader interprets this as a "key-setting" in AA + 1 but this is irrelevant since AA + 1 is filled with the next program instruction.

4. Each ALBIN mode block is translated beginning with an address at page 1. This means that the first restoring expression has to be: . 200 + x, x is the relative page address where the program has to be placed. This expression must occur before the first instruction of the block. It may, however, be preceded by one or more "key-settings":

```
*INPUT
ALBIN
*SUBI
*200

SUB, 0


______
______
```

This program can be loaded from the beginning of each page and its actual place then has been noted down in the (page 0) address SUBI.

5. Reading a binary tape, the ALBIN loader will halt at the beginning of each ALBIN mode block. The actual starting address of the block then has to be set via the switch register. Loading will follow by pressing the continue button.

The starting address must agree with paragraph 4 which implies that the least significant 7 bits of the switch register must contain x. In most cases, x can be 0. The blocks then always start at the beginning of a memory page.

6. Sample conversion of a symbolic BIN mode program block in ALBIN mode.

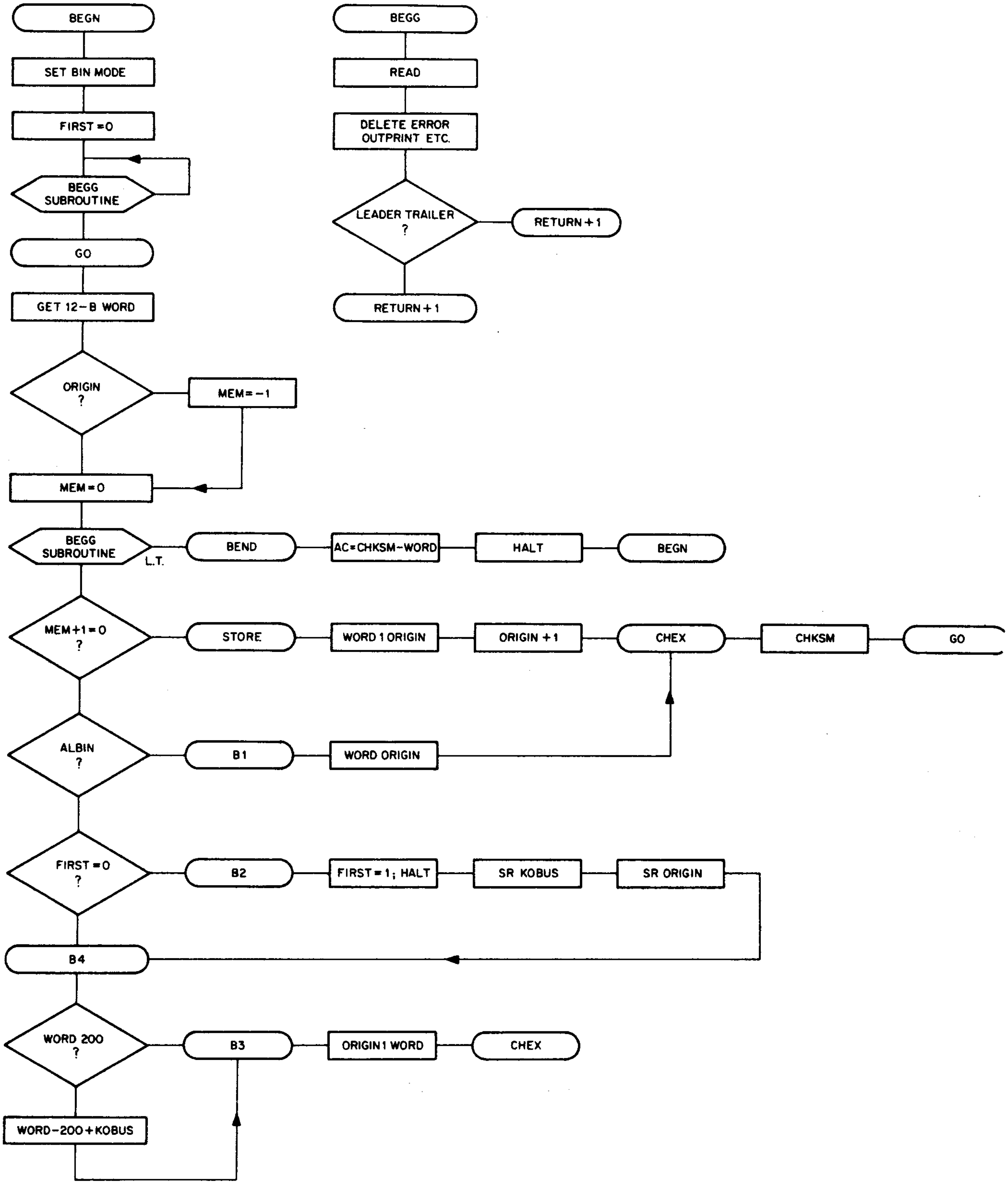
```
/BIN                                 /ALBIN
*KEY                                 *INPUT
SUB1, SUBI1                          ALBIN
SUB2, SUBI2                          *SUB1
                                     *2ØØ
*BEG
SUBI1, Ø                             SUBI1, Ø
       --                                   --
       --                                   --
       JMP I SUBI1                   BB, JMP I SUBI1
                                     *SUB2
                                     *BB+1
SUBI2, Ø                             SUBI1, Ø
       --                                   --
       --                                   --
       TAD I  AAI                           TAD I AAI
       --                                   --
AAI,   AA                            AAI,   Ø
       --                                   --
______  ______  ______  ______  ______  ______    Page Transition

       --                                   --
       --                            CC,    --
                                     *AAI
                                     *CC+1
AA,    NUMBER                        AA,    NUMBER
       --                                   --
$                                    $
```

```
                  *7400
7400    6032    BEGRIM, KCC                         /RIM-LOADER
7401    4216            JMS READIN
7402    7106            CLL RTL
7403    7006            RTL
7404    7510            SPA
7405    5201            JMP BEGRIM+1
7406    7006            RTL
7407    6031            KSF
7410    5207            JMP .-1
7411    6034            KRS
7412    7420            SNL
7413    3753            DCA I KOBUS
7414    3353            DCA KOBUS
7415    5600            JMP I BEGRIM
7416    0000    READIN, 0                           /READ-ROUTINE
7417    6031            KSF
7420    5217            JMP .-1
7421    6036            KRB
7422    3346            DCA CHAR
7423    1346            TAD CHAR
7424    5616            JMP I READIN
7425    7240    BEGN,   CLA CMA                     /(AL)BIN-LOADER
7426    3344            DCA INDIC                   /SET BIN-MODE
7427    3345            DCA CHKSM
7430    3352            DCA FIRST
7431    4320            JMS BEGG
7432    5231            JMP .-1
7433    1346    GO,     TAD CHAR
7434    3347            DCA CKT
7435    1346            TAD CHAR
7436    7106            CLL RTL
7437    7006            RTL
7440    7006            RTL
7441    3350            DCA WORD
7442    4216            JMS READIN
7443    1350            TAD WORD
7444    3350            DCA WORD
7445    7430            SZL
7446    7240            CLA CMA
7447    3351            DCA MEM
7450    1346            TAD CHAR
7451    1347            TAD CKT
7452    3347            DCA CKT
7453    4320            JMS BEGG
7454    5357            JMP BEND
7455    2351            ISZ MEM
7456    5302            JMP STORE
7457    1344            TAD INDIC
7460    7640            SZA CLA
7461    5306            JMP B1
7462    1352            TAD FIRST
7463    7650            SNA CLA
7464    5311            JMP B2
7465    1350    B4,     TAD WORD
```

```
7466  1273           TAD M200
7467  7510           SPA
7470  5273           JMP B3
7471  1353           TAD KOBUS
7472  3350           DCA WORD
7473  7600   B3,     M200,7600          /CLA
7474  1354           TAD ORIGIN
7475  3750           DCA I WORD
7476  1347   CHEX,   TAD CKT
7477  1345           TAD CHKSM
7500  3345           DCA CHKSM
7501  5233           JMP GO
7502  1350   STORE,  TAD WORD
7503  3754           DCA I ORIGIN
7504  2354           ISZ ORIGIN
7505  5276           JMP CHEX
7506  1350   B1,     TAD WORD
7507  3354           DCA ORIGIN
7510  5276           JMP CHEX
7511  2352   B2,     ISZ FIRST
7512  7402           HLT
7513  7604           CLA OSR
7514  3353           DCA KOBUS
7515  1353           TAD KOBUS
7516  3354           DCA ORIGIN
7517  5265           JMP B4
7520  0000   BEGG,   0
7521  3355           DCA SWITCH
7522  4216           JMS READIN
7523  1356           TAD M377
7524  7640           SZA CLA
7525  5331           JMP .+4
7526  2355           ISZ SWITCH
7527  7240           CLA CMA
7530  5321           JMP BEGG+1
7531  1355           TAD SWITCH
7532  7640           SZA CLA
7533  5322           JMP BEGG+2
7534  1346           TAD CHAR
7535  0341           AND MASK
7536  1273           TAD M200
7537  7450           SNA
7540  5723           JMP I BEGG
7541  7700   MASK,   SMA CLA
7542  5322           JMP BEGG+2
7543  2320           ISZ BEGG
7544  0000   INDIC,  0
7545  0000   CHKSM,  0
7546  0000   CHAR,   0
7547  0000   CKT,    0
7550  0000   WORD,   0
```

```
7551  0000  MEM,     0
7552  0000  FIRST,   0
7553  0000  KOBUS,   0
7554  0000  ORIGIN,  0
7555  0000  SWITCH,  0
7556  7401  M377,    -377
7557  1350  BEND,    TAD WORD
7560  7041           CMA IAC
7561  1345           TAD CHKSM
7562  7402           HLT
7563  5225           JMP BEGN
```

```
BEGG    7520
BEGN    7425
BEGRIM  7400
BEND    7557
B1      7506
B2      7511
B3      7473
B4      7465
CHAR    7546
CHEX    7476
CHKSM   7545
CKT     7547
FIRST   7552
GO      7433
INDIC   7544
KOBUS   7553
MASK    7541
MEM     7551
M200    7473
M377    7556
ORIGIN  7554
READIN  7416
STORE   7502
SWITCH  7555
WORD    7550
```


BEGN
SET BIN MODE
FIRST =0
BEGG SUBROUTINE
GO
GET 12-B WORD
ORIGIN ?
MEM=-1
MEM=0
BEGG SUBROUTINE
L.T.
BEND
AC=CHKSM-WORD
HALT
BEGN
MEM+1=0 ?
STORE
WORD 1 ORIGIN
ORIGIN +1
CHEX
CHKSM
GO
ALBIN ?
B1
WORD ORIGIN
FIRST =0 ?
B2
FIRST=1; HALT
SR KOBUS
SR ORIGIN
B4
WORD 200 ?
B3
ORIGIN 1 WORD
CHEX
WORD-200+KOBUS
BEGG
READ
DELETE ERROR OUTPRINT ETC.
LEADER TRAILER ?
RETURN+1
RETURN+1