DECUS
PROGRAM LIBRARY

DECUS NO.	8-32
TITLE	A Program to Relocate and Pack Programs in Binary Format
AUTHOR	J. W. Bowman
COMPANY	Atomic Energy of Canada Ltd.
DATE	November 1965
SOURCE LANGUAGE	PAL III

TABLE OF CONTENTS

APPENDICES

A PROGRAM TO RELOCATE AND PACK PROGRAMS IN BINARY FORMAT

DECUS Program Library Write-up DECUS No. 5/8-32a

SYNOPSIS

A relocation package has been written to provide a means to shuffle machine language programs around in memory to make the most efficient use of computer store. This report describes this relocation program, the format required to make it possible and the operation of the program itself. The program is now in use and has proved to be a most effective tool, both in assembling a combination of existing programs, ard in amending and fault-finding new programs.

1.0 INTRODUCTION

Implicit in the concept of using a small computer to perform the basic logic for a general purpose nuclear particle spectrometer is the need to provide a library of program material. The computer memory can store only a given amount of this material at one time. To load the required programs into the computer it is necessary to map out the available space in memory and change the starting address of each program to fit these memory locations. This can be done by changing the symbolic tapes and reassembling them with "PAL" (the assembler for the PDP-5/8 computer). This is tedious, and in keeping with the general purpose nature of the system a faster method was evolved for receiving the programs and packing them into memory.

This report describes a relocation program that allows immediate transfer of a program in binary format to locations selected automatically by this program or to locations described by the operator from the keyboard. The main obstacle to relocation of programs for the PDP-5/8 computer is "PAGE ADDRESSING." The memory is divided into 32 pages of 128 registers each. Direct addressing is only allowed within the limits of a page. To relocate a program onto two pages would require a program that set up indirect addressing where required, to allow the relocated program to communicate between pages. This type of program would be very complicated and would require a large area of memory. By keeping the relocation program small a larger area is available for the actual relocated programs. The restriction that is put on this program then is this: "Do not attempt to relocate a program written to occupy one page on to two pages of memory."

Before proceeding any further, a brief explanation of the term "Subroutine" may be in order. A subroutine is a program that becomes accessible from various areas of memory through a "JMS" instruction. The subroutine will exit and return to the JMS register plus one. In this manner, a subroutine can be used repeatedly from anywhere in memory. A non-subroutine on the other hand will exit to a fixed location only.

The system adopted for our library of kicksorter programs is subroutine oriented. The programs for each function such as plot or printout are a collection of subroutines, each subroutine being not greater than one page in length. These subroutines are modular in nature and are placed into memory with the relocation program. The relocation program, in the automatic mode of operation will search memory, starting at page one, for enough consecutive registers on one page to contain the subroutine. The subroutine is then altered by the relocation program to operate in the new registers and relocated to those registers.

In order to use a subroutine whose starting address is not fixed it is necessary to store the new starting address. Page zero is used for this purpose. An explanation is given under the heading

"Description." The relocation program will store the new starting address in a specific location on page zero. This subroutine will always be referenced indirectly through its zero page address.

A punch program has been included in this package to retrieve the relocated program.

A special format is required to make relocation in the above described maner possible. This format is quite simple and easily adopted to existing programs.

2.0 DESCRIPTION

Before explaining in detail the description of the program and how it operates a brief outline of the organization of the system might be helpful.

All binary tapes for our kicksorter functions are punched on one long library tape. An experimenter, upon determining which subroutines are required for his particular experiment will read this tape with the relocation program and accept or reject the subroutines as they occur. The detailed use of each subroutine will be listed in a catalogue for this purpose.

2.1 Relocation Format

A special format must be followed when writing programs to be successfully relocated with this program.

2.1.1 Zero Page

Zero page plays an important role in the organization of this system. Since the location of subroutines are not fixed it is necessary to know how to get to these subroutines and how to extract information from them. Zero page is used for this purpose. A zero page location is allotted each subroutine. The relocation program will insert the new starting address into the allotted zero page register, providing the zero page register is mentioned on the tape in the following manner.

```
Example,    *33          /Page zero location
            3000
           *3000         /Current page starting address
PRT,        0
            CLA CLL
            ETC.
```

Notice that this program can be read into memory with a binary loader and the starting address (3000) will appear in zero page register 33. Tapes written in this format are completely compatible with the binary loader.

A program is sucessfully entered by jumping (or JMS) indirect to the zero page register.

A map of zero page is required to keep track of allotted zero page registers. A suggested division is outlined below.

0-7	INTERRUPT routine instructions
10-17	AUTO-INDEX Channels
20-157	STARTING ADDRESSES of subroutines
160-177	TEMPORARY DATA that is required repeatedly by more than one subroutine.

Note that the last 16 registers are referred to as temporary data holding registers. When a subroutine accumulates information required by one or more other subroutines it is convenient to store it in these registers. When the information is no longer required that particular register may be used to hold other similar data. Zero page is conserved by sharing registers in this manner.

Programs that are not subroutines may be relocated in the same manner as described above. The program must follow the same format rules, but the zero page register is deleted.

Subroutines that are never used together can be allocated the same zero page location. For example, the linear display employing a 24-bit word and a similar display program using an 18-bit word cannot be used together, therefore, they may use the same zero page register.

The basic rule to remember is, not to refer to fixed addresses in programs on other pages because the location of a program is not fixed. Always get into the program through its zero page register.

2.1.2 Marker Tags

The actual program to be relocated consists of the following materials.

(a) Memory reference instructions

(b) Micro instructions

(c) IOT instructions

(d) Fixed constants

(e) Registers containing the addresses of registers.

All micro, IOT, and memory reference instructions that refer to zero page are relocated unaltered. All other memory reference instructions are adjusted to suit their new locations. The fixed constants listed at the end of the page may look like memory reference instructions, but must not be altered. A marker is inserted in the program to inform the relocation program that the list of fixed constants begins here. The marker is 6670 and is used as follows:

Example,

```
HLT                                  /End of memory buffer inst.
STORE,      6670                     /marker
COM1,       7214                     /Beginning of tags
ETC.
```

Note that the marker is tagged "STORE." In the actual program "STORE" is a location that will normally contain zero. By using "STORE" in this manner the marker (6670) does not occupy an additional address and does not increase the length of the program.

It is necessary to store the markers to make the programs compatible with the existing binary loader, and also to assemble programs in this format with the "PAL" assembler.

Registers containing the addresses of other registers are dealt with in a special manner. For this reason they are put at the end of the program and are preceded by the marker (6670) in the following manner.

Example,

```
STORE,      6670
HOME,       0132                     /Fixed constants
            4214
            4456
LOCATE,     6770                     /Marker No. 2
ADR,        HOME                     /Contains address "HOME"
            $                        /On current page.
```

Tags containing memory reference instructions are dealt with in the same manner as the memory reference instructions in the program proper, therefore these tags are put ahead of the marker 6670.

Example,

```
HLT,
MOD1,       JMP OVER
MOD2,       TAD MORE
STORE,      6670                     /Beginning of
ETC.                                 /Fixed Constants.
```

The following is an example of how all three types of tags are used:

Example,

```
                    HLT              /End of memory buffer
MOD1,               JMP OVER         /Instructions
MOD2,               TAD MORE
STORE,              6670             /Marker No. 1
COM1,               7315
COM2,               7317
HOME,               0132
                    4215
                    3317
                    2111
LOCATE,             6770             /Marker No. 2
ADR,                HOME
                    $
```

3.0 ASSOCIATED PROGRAMS

Three programs have been included in this package to aid in relocating and retrieving subroutines.

3.1 Clear Memory

A clear memory routine has been included to zero registers from 200_8 to 6377_8. This area in our system is for programs. Since the relocation program recognizes unused registers only if they contain zero, a means of zeroing these registers was necessary. This area can be readily changed by altering the program. The starting address of this program is 6671_8. The program is 12 registers in length.

3.2 Display Memory

A programmed display routine has been included to display the entire memory. Empty registers will appear on the base line while occupied addresses appear as 4096 counts. The starting address is 6527 and its length is 15 registers.

3.3 Binary Punch

Often it is necessary to retrieve the relocated programs on paper tape. This punch routine will punch only those areas of memory containing program material. This program senses and deletes the gaps in memory that contain nothing. A great deal of time is saved because start and end addresses of each program need not be entered as with other binary punch programs.

The tape produced is complete with checksum. The starting address of this program is 6400_8 and is 90 registers in length.

The relocation program and associated programs are themselves relocatable. If the registers 6400_8 to 7400_8 are not convenient the entire relocation program or any part of it can be relocated to another part of memory. It must be remembered that in so doing the starting addresses mentioned above will change accordingly.

4.0 OPERATION

Steps followed by the operator in putting this program into operation are as follows:

(a) Load the relocation program into the computer with binary loader. The starting address of the binary loader is 7777_8.

The relocation program and associated programs occupy memory from 6400_8 to 7400_8.

(b) The relocation program in the automatic mode assumes that registers not being used contain 0000_8. For this reason it is important to zero all registers that will be used to contain program material. The

"Clear Memory" routine previously described is used for this purpose. The starting address of this routine is 6671_8.

(c) Set the binary tape of the program to be relocated into the reader. Load the starting address of the relocation program (SA 7200_8) on the switch register and start. The binary tape will be read into a buffer page in memory

The zero page address is printed out on the teleprinter along with the total number of registers required for that program in octal. If a zero page address is not specified on the binary tape, the relocation program will print 0000 for this address.

At this point the relocation program will make a decision based on the contents of the switch register.

(d) Switch register bits "0" and "1" are interpreted as follows.

Bit 0 = 0, Bit 1 = 0

Whenever bit 1 is set to zero the present program is rejected. The relocation program returns to the reader to read the next program into the buffer page. If bit 1 is still a zero that subroutine will also be rejected, and so on.

Bit 0 = 0, Bit 1 = 1

This combination of the switch register is an instruction to read the present subroutine into memory using the automatic mode of operation. In this mode the program will search memory starting at register 200_8 for enough consecutive empty registers on one page to contain the program to be relocated. In this manner the various sized programs are packed together, the smaller programs filling up the gaps at the ends of each page.

When the search finds suitable accommodations for the program, the new starting address is printed on the teleprinter in octal.

Upon relocation this program will return to the reader to accept or reject the next program.

Bit 0 = 1, Bit 1 = 0

Since Bit 1 = 0 the present program is rejected. Operation is the same as Bit 0 = 0, Bit 1 = 0.

Bit 0 = 1, Bit 1 = 1

The relocation program will wait for the operator to decide whether to accept or reject the present program. If a number of programs are to be selected from a library tape of subroutines this mode of operation is desired. The relocation program will wait for the operator to set bit 0 or bit 1 to a zero to accept or reject the program, as outlined above.

This mode of operation will also allow the operator to select a starting address from the keyboard. If specific registers are required, into which this program will be relocated, the starting address can be entered through the keyboard in octal. Be sure the starting address used does not permit the relocated program to write over the end of a page. For example, if the number of registers required as described in (c) of this section, is 170_8 and a start address of 3620 were chosen, the end address would be 4010$_8$. Since the end of the program is on another page, the program was not correctly relocated and will not operate correctly.

Upon successful relocation of the program in either mode of operation the end address is printed out and the program returns to the reader to receive the next program. A selection of programs from a library tape is shown in Table 1. Notice that programs selected have a start and end address whereas the rejected programs do not.

If the program is not read properly and the checksum is wrong the program will halt. If this happens reload the binary tape of the subroutine to be relocated and restart the relocation program again.

Occasionally it is desired to use a fixed program, a program that is not written in relocatable format. This can be done quite easily by loading the fixed program with the binary loader and then proceeding at (c) of this section. The relocatable programs will fit themselves around the existing program.

5.0 SUMMARY

Satifactory results have been obtained from this program for our use. Our library of subroutines is now easily accessible and it takes a relatively short time to pack programs into memory for a specific combination of functions for spectrum analysis. Subroutines can now be compared to plug in modules for ease of accessibility.

Although this program was conceived to increase the apparent sorting power of a PDP-5/8 as a nuclear particle spectrometer there is no reason why it cannot be used to increase the storing capability of this computer in other fields.

APPENDIX I

```
/Relocation program Modes 1 and 2
*173
7200
*7200
START, 6046
JMS CRLF
DCA Z 170
DCA ONELES
DCA CKSM
REDY, JMS KBD
SNA
JMP REDY                /BLANK
AND LEADER
SZA CLA
JMP REDY
HERE, TAD STOR
JMS ASSEMB
SNL                     /ORIGIN SET?
JMP REDY
DCA STOR
TAD STOR
TAD CON1                /ZERO PAGE?
SNL CLA
JMP .+4
TAD STOR
DCA Z 170
JMP REDY
TAD STOR
AND MSK2
DCA CHG
TAD CHG
TAD CON3
DCA Z 14                /SA OF BUFFER
DCA Z 12                /STORE IN BUFFER.
JMS KBD
JMP UP+1
STAR, JMS KBD
AND LEADER
SZA CLA
JMP ON
TAD HOLD
DCA I Z 14
ISZ Z 12
UP, TAD STOR
AND ORIGIN
SZA CLA
JMP HERE
TAD STOR
JMS ASSEMB
DCA HOLD
JMP STAR
ON, TAD Z 170
JMS I Z 175             /PRINT ZERO PAGE LOCATION
```

```
TAD Z 12
JMS I Z 175          /PRINT TOTAL NUMBER OF LOCATIONS REQUIRED
TAD HOLD             /FIND NEW LOCATION
CMA IAC
TAD CKSM
SZA CLA              /CORRECT CHECKSUM?
HLT
DMS I Z 4            /FIND
DCA Z 16
IAC
DCA Z 171
RETURN, TAD Z 12
TAD Z 14
DCA Z 17
TAD Z 16
IAC
AND MSK2
DCA STOR
IAC
TAD Z 17
AND MSK2
CMA IAC
TAD STOR
JMP I Z 172          /DIFFERENCE NEW SA AND OLD SA
NOP
KBD, 0
6014
6011
JMP .-1
CLA
6012
DCA STOR
TAD STOR
JMP I KBD
ASSEMB, 0
DCA FIRST
TAD ONELES
TAD CKSM
DCA CKSM
TAD FIRST
RTL CLL
RTL CLL
RTL CLL
DCA TEMP
JMS KBD
TAD FIRST
DCA ONELES
TAD STOR
TAD TEMP
JMP I ASSEMB
CRLF, 0
CLA CLL
TAD CR
JMS PRT
TAD LF
JMS PRT
JMP I CRLF
PRT, 0
TSF
JMP .-1
TLS
CLA
JMP I PRT
CKSM, 6670
CR, 215
LF, 212
STOR, 0
LEADER, 200
ORIGIN, 100
CON1, 7600
MSK2, 177
CHG, 0
CON3, 7377
HOLD, 0
TEMP, 0
ONELES, 0
FIRST, 7777
$
```

```
/RELOCATION PROGRAM PART TWO
*172
7000
*7000
/ALTER INSTR
DCA TWOO
AGNN, TAD I Z 17
DCA Z 11
TAD Z 11
TAD SEN3
SNA CLA          /CODE 6670 ?
JMP TAGG
TAD Z 11
AND MSK6
TAD CON6
SNA              /TAG ?
JMP STRR
TAD CON6
SNA CLA          /IOT ?
JMP STRR
TAD Z 11
AND TWO
SNA CLA          /ZERO PAGE ?
JMP STRR
TAD Z 11
TAD TWOO
DCA Z 11
JMP STRR
NOP              /STORAGE ON NEW PAGE
STRR, CLA
TAD Z 11
DCA I Z 16
ISZ Z 12
JMP AGNN
JMP END
NOP
TAGG, CLA
TAD Z 11
DCA I Z 16
ISZ Z 12
SKP
JMP END
TAD I Z 17
DCA Z 11
TAD Z 11
TAD SEN4
SZA CLA
JMP TAGG+1
TAD Z 11
DCA I Z 16
ISZ Z 12
SOM, TAD I Z 17
DCA TEMP
TAD TEMP
AND MSK2
DCA TEMP
TAD Z 16
AND MSK5
TAD TEMP
TAD TWOO
DCA I Z 16
ISZ Z 12
JMP SOM
JMP END
END, CLA CLL
TAD Z 16
JMS I Z 175  /PRINT
TAD Z 170
SNA CLA
JMP I Z 173
TAD Z 171
DCA I Z 170
JMP I Z 173 /READY
NOP
TWOO, 6670
SEN3, 1110
MSK6, 7000
CON6, 1000
TWO, 200
SEN4, 1010
TEMP, 0
MSK2, 177
MSK5, 7600
$
```

```
/RELOCATION PROGRAM PART 3
*4
6600
*6600
FIND, 0
TAD Z 12
CMA IAC
DCA Z 12
DCA NUM
CLA OSR
CLL RAL
SMA CLA
JMP I Z 173    /REJECT
SZL
JMP MANUAL
TAD MSK2
DCA Z 16
OSC, TAD CON1
DCA TIMES
TAD Z 12
DCA TALY
ISZ TIMES
SKP
JMP OSC
TAD I Z 16
SZA CLA
JMP .-7
ISZ TALY
JMP .-7
TAD Z 12
TAD Z 16
DCA Z 16
TAD Z 16
IAC
JMS I Z 175    /PRINT SA
TAD Z 16
JMP I FIND
NOP
MANUAL, NOP
KSF
JMP FIND+5
KRB
TLS
DCA STORE
TAD STORE
TAD SEN1       /COMMA
SNA CLA
JMP DWN
TAD STORE
AND MASK
DCA STORE
TAD NUM
RAL CLL
RTL CLL
TAD STORE
DCA NUM
JMP MANUAL+1
DWN, CLA CMA
TAD NUM
JMP I FIND
NOP
/CLEAR MEMORY
CLA
TAD FOUTH
DCA FORTH
TAD STRT
DCA Z 10
DCA I Z 10
ISZ FORTH
JMP .-2
HLT
NOP
TIMES, 6670
MSK2, 177
CON1, 7577
TALY, 0
HOLD, 7777
NUM, 7777
STORE, 0
SEN1, 7524
FOUTH, 0-6377
FORTH, 0
STRT, 177
MASK, 7
$
```

```
CONNN, 100
HOLD, 0
TIME, 7677
TIM, 7600
LDR, 200
FORT, 4000
FINI, 0-6377
$


/PRINT AN OCTAL NUMBER
*175
7140
*7140
OCTOUT, 0
DCA THIS
TAD SPACE
JMS PRT
TAD FOUR
DCA FOR
TAD THIS
RAL
RAL
RTL
DCA THIS
TAD THIS
AND MSKK
TAD CON
JMS PRT
TAD THIS
ISZ FOR
JMP .-11
CLA CLL
JMP I OCTOUT
PRT, 0
TSF
JMP .-1
TLS
CLA
JMP I PRT
THIS, 6670
SPACE, 240
FOUR, 7774
FOR, 0
MSKK, 7
CON, 260
$
```

```
/PUNCH ONLY INFO IN MEMORY
*6400
CLA
DCA SCORE
JMS TRAL
DCA Z 10
UP, TAD I Z 10
SNA
JMP .-2
DCA HOLD
TAD Z 10
CLL
TAD FINI
SZL CLA
JMP FINISH
TAD Z 10
RTR
RTR
RTR
AND MSK100          /0077
TAD CONNN           /0100 ORIGIN
JMS WRITE
TAD Z 10
AND MSK100          /0077
JMS WRITE
TAD HOLD
JMP TO
HALF, TAD Z 10
TAD FINI
SZL CLA             /REACHED LIMIT ?
JMP FINISH
TAD I Z 10
SNA
JMP STRZRO
CLIMB, DCA HOLD
TAD HOLD
TO, RTR
RTR
RTR
AND MSK100
JMS WRITE
TAD HOLD
AND MSK100
JMS WRITE
JMP HALF
FINISH, CLA CLL
TAD SCORE
NOP
DCA HOLD
TAD HOLD
RTR
RTR
RTR
AND MSK100
JMS WRITE
TAD HOLD
AND MSK100
JMS WRITE
JMS TRAL
HLT
NOP
WRITE, 0
DCA CKSM
TAD SCORE
TAD CKSM
DCA SCORE
TAD CKSM
6026
6021
JMP .-1
CLA CLL
JMP I WRITE
TRAL, 0
CLA CLL
TAD TIM
DCA TIME
TAD LDR
6026
6021
JMP .-1
CLA CLL
ISZ TIME
JMP .-6
JMP I TRAL
STRZRO, CLA
JMS WIRTE
JMS WRITE
TAD I Z 10
SNA
JMP UP
JMP CLIMB
NOP
DSPY, CLA CLL    /DISPLAY
6051
DCA Z 10
TAD I Z 10
SZA CLA
CMA
6067
ISZ FORT
JMP .-5
JMP DSPY
NOP
SCORE, 6670
CKSM, 0
MSK100, 77
```

APPENDIX II

TABLE I

ZP	REGS	SA	EA
0037	0047	0200	0246
0065	0066		
0022	0140	0400	0537
0031	0200	0600	0777
0054	0161	1000	1160
0034	0125	0247	0373
0026	0117		
0026	0124		
0067	0062		
0066	0013	0540	0552
0057	0173	1200	1372
0060	0200	1400	1577
0050	0156	1600	1755
0055	0041	2000	2040
0020	0023	0553	0575
0064	0065	2041	2125
0044	0067	2200	2266
0032	0040	2126	2165
0025	0150		
0021	0063	2267	2351
0062	0033	2400	2432
0000	0021	1756	1776
0035	0146	2600	2745
0053	0056	2433	2510
0023	0015	1161	1175
0027	0122	3000	3121
0030	0164	3200	3363
0033	0200	3400	3577
0040	0051	2511	2561
0041	0035	3122	3156
0042	0163	3600	3762
0045	0013	2352	2364
0043	0012	2166	2177
0047	0134	4000	4133
0051	0072		
0056	0144	4200	4343
0063	0052		