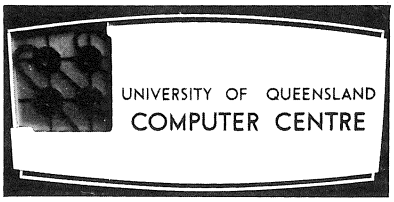


COMPUTER CENTRE

BULLETIN

Volume 2, Number 10.
6th October, 1969.

Editor:
H. L. Smythe.

THIS EDITION

This issue of the Bulletin contains two major articles that should be of interest to the reader, one discussing, in general terms, the position of the law and computers, and the other describing the addition of hardware to the PDP 10 to improve further the efficiency of multi-programming. The Bulletin continues to inform readers of FORTRAN errors on the PDP 10, and a Letter to the Editor is also published.

LETTERS TO THE EDITOR

Recent events have shown that man is capable of travelling to the moon and back in less than two weeks. In the same amount of time, it is possible for any of us, using more conventional means of transport, to travel around the world, even stopping off for occasional sight seeing.

In far less than two weeks, as recent history shows us, a war can commence and be won - or lost, according to one's particular outlook. Yet recently, some data I submitted to the Computer Centre were still awaiting preparation after two weeks. The only apparent movement was the transposition of my name from the end of a list to a position third from the top. The data were eventually prepared two weeks and five days after their submission and subsequently analysed by the GE 225 in four minutes.

It seems to me that this situation is ludicrous and unnecessary. If the university is able to spend a large sum on a modern computer, surely it should be possible to provide those ancillary services necessary to ensure its efficient use. If one has to wait for over two weeks to have one's data prepared, efficient use of the computer is not being made.

Surely it is possible to have faster data preparation than this. Many of us have either seen or heard of centres where a delay of more than a day for data preparation is the exception rather than the rule.

I have no immediate ambition to travel to the moon, to have a round-the-world trip, or participate in a war, however little time is required. But together, I am sure, with many others at the University of Queensland, I would like to receive my prepared data cards before I have had time to forget what they are all about. *Malcolm A. Colston (Department of Education), 5 Glencarron Street, Kenmore, Q. 4069.*

[Ed. Note:- Mr. Colston's letter raises a query about the type and quality of service that the Computer Centre provides for key-punching.

The Centre is essentially a scientific *data-processing* centre, not a *data-preparation* centre. The data preparation service we offer is only a token service. In fact, departments with significant punching requests have installed keypunches for their own data preparation.

Our records show that the average turnaround time for punching is approximately three days. There are, however, periods when the delay builds up because of sickness or changes of staff. This is unfortunate, but, to guarantee a maximum turnaround time of only one or two days, key-punching staff would have to be increased considerably in order to cope with the peak demands rather than the average. This would mean a significant increase in charges for punching, and would, no doubt, inspire numerous *Letters to the Editor!*

Incidentally, Mr. Colston's implication that the University purchased the computer and finances the ancillary services, is incorrect. The computer was purchased with funds from a variety of sources, the University contributing about one-third. In particular, the University does not contribute to the Centre's data-preparation section. In fact, the cost of providing the key-punch service is barely covered by the charges that are made.]

FORTRAN IV ERRORS

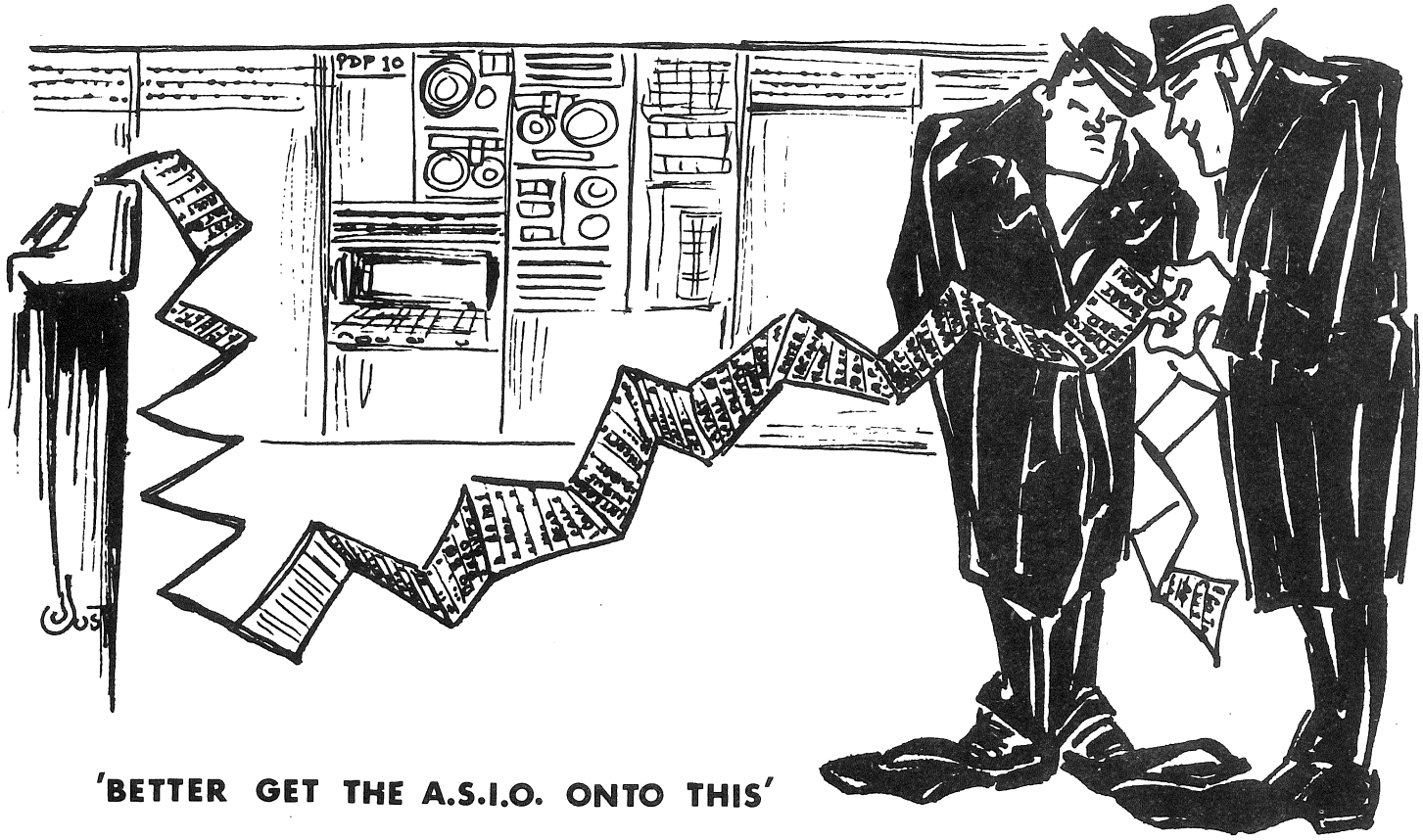
1. It has been discovered that certain types of integer expressions do not compile correctly on the PDP 10.

For example:

```
K = L*(N-1)+(M*(N-1))/2
```

At first sight, this would seem to be a fundamental failing of the compiler. However, a number of circumstances must exist before the error occurs.

(a) There must be some common expression in the statement. In the above example, this is (N-1).



'BETTER GET THE A.S.I.O. ONTO THIS'

(b) This common expression must be further operated on by other variables, different for each occurrence of the common expression, e.g. L*(N-1) and M*(N-1)

(c) The last of these composite expressions must be further operated on by an integer divide, e.g. (M*(N-1))/2

Incorrect code is generated by the compiler. If any one of the above conditions does not exist, the correct code will be produced. It is emphasized that the problem occurs only with integer expressions.

2. Variable dimensioning of arrays is permissible within subroutines and is a convenient means of creating a generalized routine for, say, matrix manipulation.

Where the type of an array is *not* implicitly defined, the appropriate type statement must appear before the dimensioning statement for correct results, although documentation would lead one to believe that the order of specification statements is not significant.

Thus:

```
SUBROUTINE SR(A,N1,N2)
COMPLEX A
DIMENSION A(N1,N2)
.
.
```

or

```
SUBROUTINE SR(A,N1,N2)
COMPLEX A(N1,N2)
.
.
```

are correct, *but*

```
SUBROUTINE SR(A,N1,N2)
DIMENSION A(N1,N2)
COMPLEX A
.
.
```

will lead to incorrect results.

3. GE 225 FORTRAN IV ERROR

The form of expression that may be used as a subscript, is limited to a few simple expression types and does not include the use of a subscripted integer variable. Thus, an expression of the form:

```
DIMENSION A(12),II(15),JJ(15),X(15)
....
A(II(J)) = A(II(J)) + X(JJ(J))
....
```

will not produce correct code although no compilation error is generated. While the compiler may, in fact, produce the correct code in some similar cases, it cannot be relied upon to do this, and thus should not be used.

Any suspected errors should be reported to the Administrative Officer (Mr. John Jauncey, extension 8471).

PDP 10 MEMORY PROTECTION AND RELOCATION

E. J. Sokoll

In a previous article, *Multiprocessing Concepts* by I. Oliver (Vol. 2, No. 3, 3rd March, 1969), reference was made to the concept of *multiprogramming* and the necessity for special hardware to provide this method of operation. This feature is available on the PDP 10 and allows a preset number of programs to be executed, even though they may not all exist in core at the same time. This is achieved by swapping programs between core memory and the fixed head disk (which has the characteristics of a drum) as dictated by the core space requirements of the program to be executed. Thus, in the process of execution, a program may be swapped in and out of core many times. Under these conditions, it is desirable that a program be capable of execution regardless of where it is stored in memory. This capability is provided by *hardware relocation*. Since a number of programs, including the executive for controlling the operation of the whole system, may exist in memory at the one time, provision must be made to ensure that these programs do not destroy one another, and this is provided by *hardware protection*.

Before describing the operation of the relocation and protection hardware, it is necessary to consider the events leading to the execution of a program. As a result of compilation or assembly of a source program, a relocatable

object program is produced, where the instruction addresses are relative to zero. By means of a relocatable or linking loader, a number of relocatable programs and subprograms may be combined, or *linked* into a composite program ready for execution. The linking of program A, consisting of subprograms A_1, A_2 and A_3 of length k, m and n, respectively, is illustrated in Figure 1.

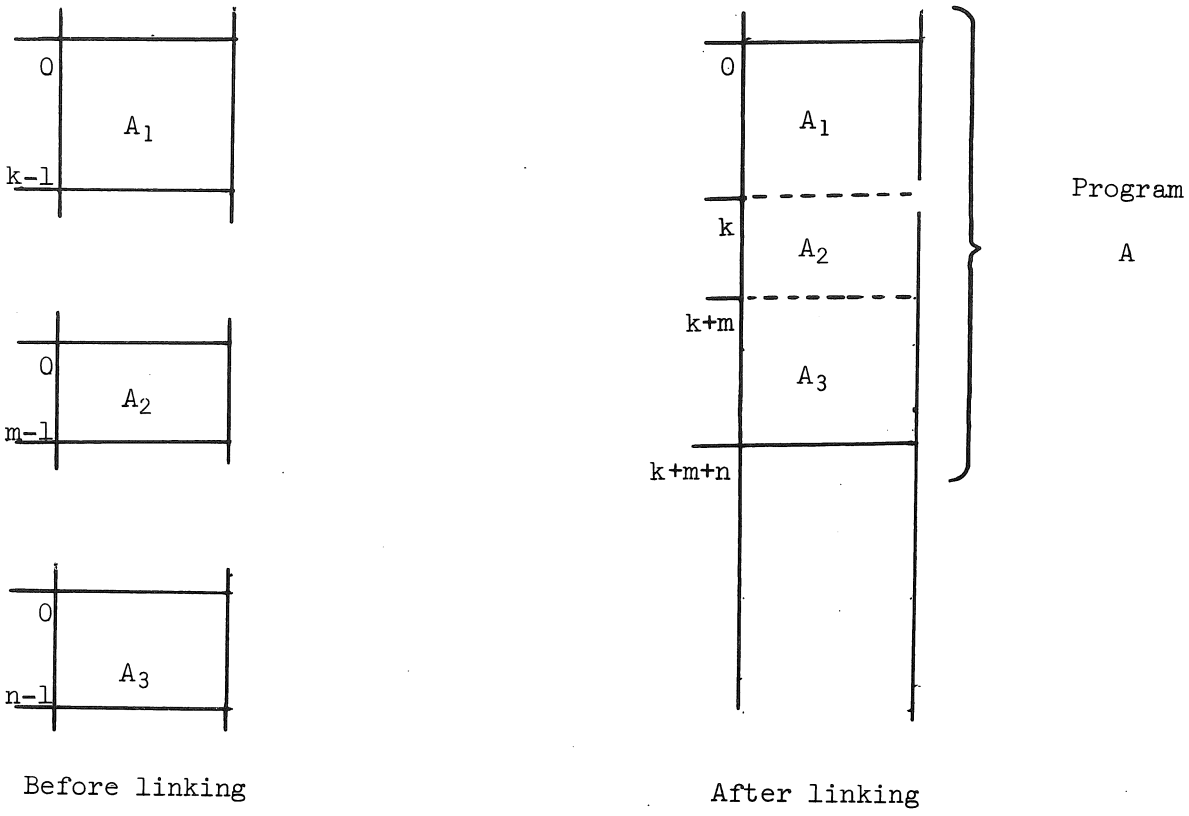


Figure 1. Linking of Program A

The addresses of programs A_1, A_2 and A_3 were each originally relative to zero. After operation of the relocatable loader, addresses of programs A_2 and A_3 have undergone *software relocation* by an amount k, and k + m respectively, and now all addresses in the complete program A are relative to zero. So far then, the individual subprograms and components of a complete program have been linked together, relocation of addresses has taken place, and the program is ready for execution. However, the program will not be stored from physical location zero since the resident executive occupies the first 16K of memory. The executive will determine what physical space the program will occupy. To ensure that the program

will be executed correctly in the physical addressing space, relocation of addresses must be performed. This is achieved by *hardware relocation* during execution of the program.

To simplify the design and implementation of the protection and relocation hardware, core memory is divided in blocks of 1K words. To ensure that each program is loaded at the start of a block, each program is allocated an integral number of 1K blocks. The program A illustrated in Figure 1 occupies k + m + n locations, and may require P_1 + 1 blocks of core memory, designated 0 to P_1. Clearly, a portion of the last block, P_1, may not be utilized.

When a program is placed in core memory for execution, it must be stored beginning at a 1K boundary. As shown in Figure 2, the program A occupies P_1 + 1 blocks, starting at block R_1. However, the addresses in program A are still

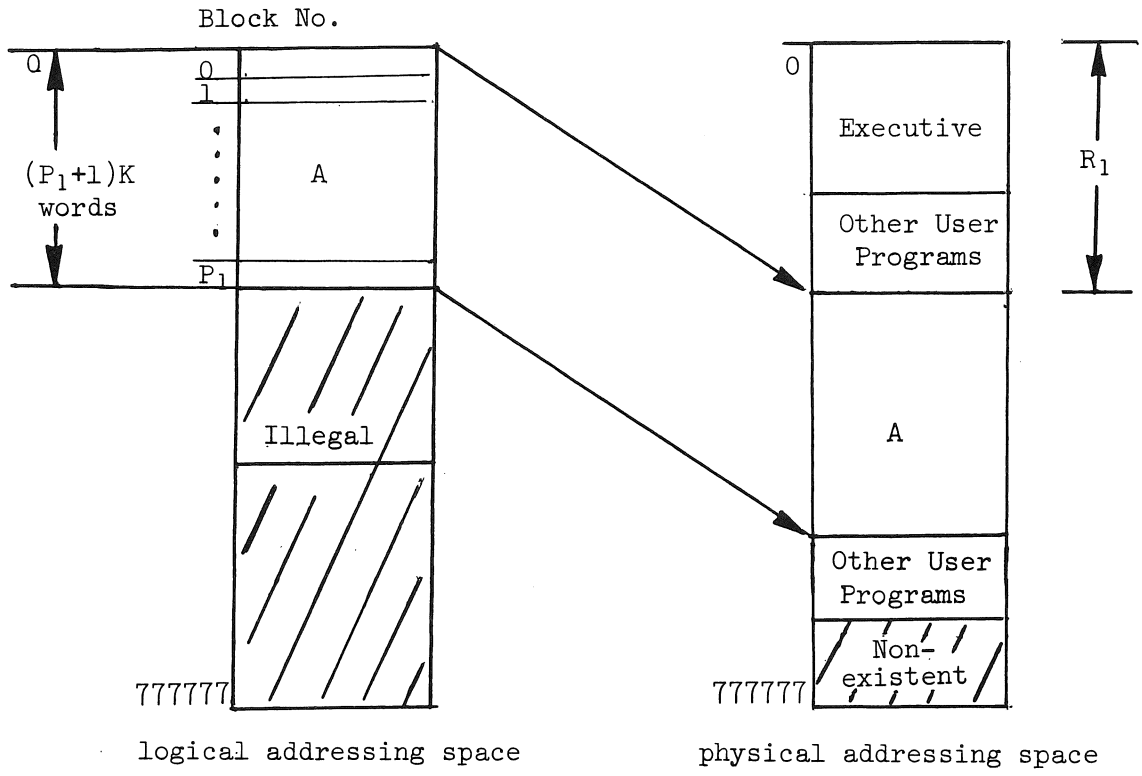


Figure 2. Logical and Physical Addressing

relative to zero prior to execution. The values of protection, P_1, and relocation R_1, are placed in the protection and relocation registers respectively by the executive. When execution begins, a memory reference is first checked to ensure that it is legal. The number of the block of memory being addressed is compared with the contents of the protection register, and if the block number exceeds P_1, execution of program A is interrupted and control is transferred to another program. In this case, the executive would indicate that an illegal memory reference had occurred. If the protection check is satisfactory, then relocation is performed to transform the relative program address to an absolute physical memory address. This is achieved by adding the contents of the relocation register, R_1, to the relative program address. This is not done on all addresses. Index registers and accumulators reference one of the first sixteen locations in memory, and since these are integrated circuit registers with a very low access time, relocation is not performed to allow all programs to use these high performance registers.

Program A may be swapped out of core and later swapped back in to continue execution. If this occurs, it is not necessary for A to occupy the same physical area that it occupied previously, nor is it likely or desirable. Before execution of A is continued, the executive reloads the relocation register with the appropriate relocation constant. This constant is sometimes referred to as the *base address*. A program may also dynamically alter its memory requirements, in which case the executive will modify the contents of the protection register to allow more or less memory to be used by a program.

In a static situation, during which swapping is not in operation, a number of programs may exist in core memory. A program in the executive, known as the *scheduler*, determines which program should be run next. When this is determined, the executive will set the appropriate values in the protection and relocation register, and pass control to the selected program. After a short period of time, control may be passed to another program which requires use of the processor, but with values for protection and relocation which apply to this particular program.

So far, the operation of a simple form of relocation and protection which was incorporated in the PDP 10 has been described. Recently, additional hardware has been installed in the PDP 10 to improve further the efficiency of multiprogramming. This additional hardware includes a second set of protection and relocation registers, and the advantages of this feature are now discussed.

It is possible that a number of jobs may require a FORTRAN compilation. In the original PDP 10 system, a copy of FORTRAN compiler was loaded for each job which required it, requiring 11K of memory for each copy of the compiler. With more than two such jobs, swapping was necessary to service other programs requesting access to the processor. As the number of programs requiring execution increases, then the processor is involved in a larger amount of swapping, and so processor efficiency decreases. It also results in a poor response time of terminals in a time-sharing system.

A program may be designed so that it is capable of being re-entered by a number of different programs. With this capability, a *re-entrant* program may be shared by any task in the system which requires it. For frequently-used programs such as the FORTRAN compiler, this capability is very desirable, since only *one* copy of the compiler is required in core.

In general, a re-entrant program consists of a *pure* part and an *impure* part. The pure part is that part of a program which is never modified during execution, and would normally include most of the instructions and constants, while the impure part may be modified during execution, and would include areas for buffers, tables and results.

In the case of the PDP 10 FORTRAN compiler, the pure part occupies 9K, while the minimum size of the impure part is 2K. With 32K of memory available, a maximum of 11 FORTRAN compilations could be performed without the need for swapping. If swapping is necessary, only the impure part is swapped out onto disk, with a resultant saving in the swapping time, and more economic utilization of disk storage.

For flexibility, it is desirable that the two parts of a re-entrant program occupy non-contiguous areas of memory, and, to ensure execution of either part and access from one part to the other, two sets of protection and relocation registers are used. To ensure that the pure part is not altered, the hardware can be enabled to prevent information being written into this part.

In this environment, as control is passed from one job to another, the executive now must load both sets of registers with the appropriate values.

Figure 3 illustrates the transformation from logical addresses to physical addresses with the two sets of registers. P_1 and R_1 are the values of protection and relocation for the impure part, while P_2 and R_2 are the values of protection and relocation for the pure part.

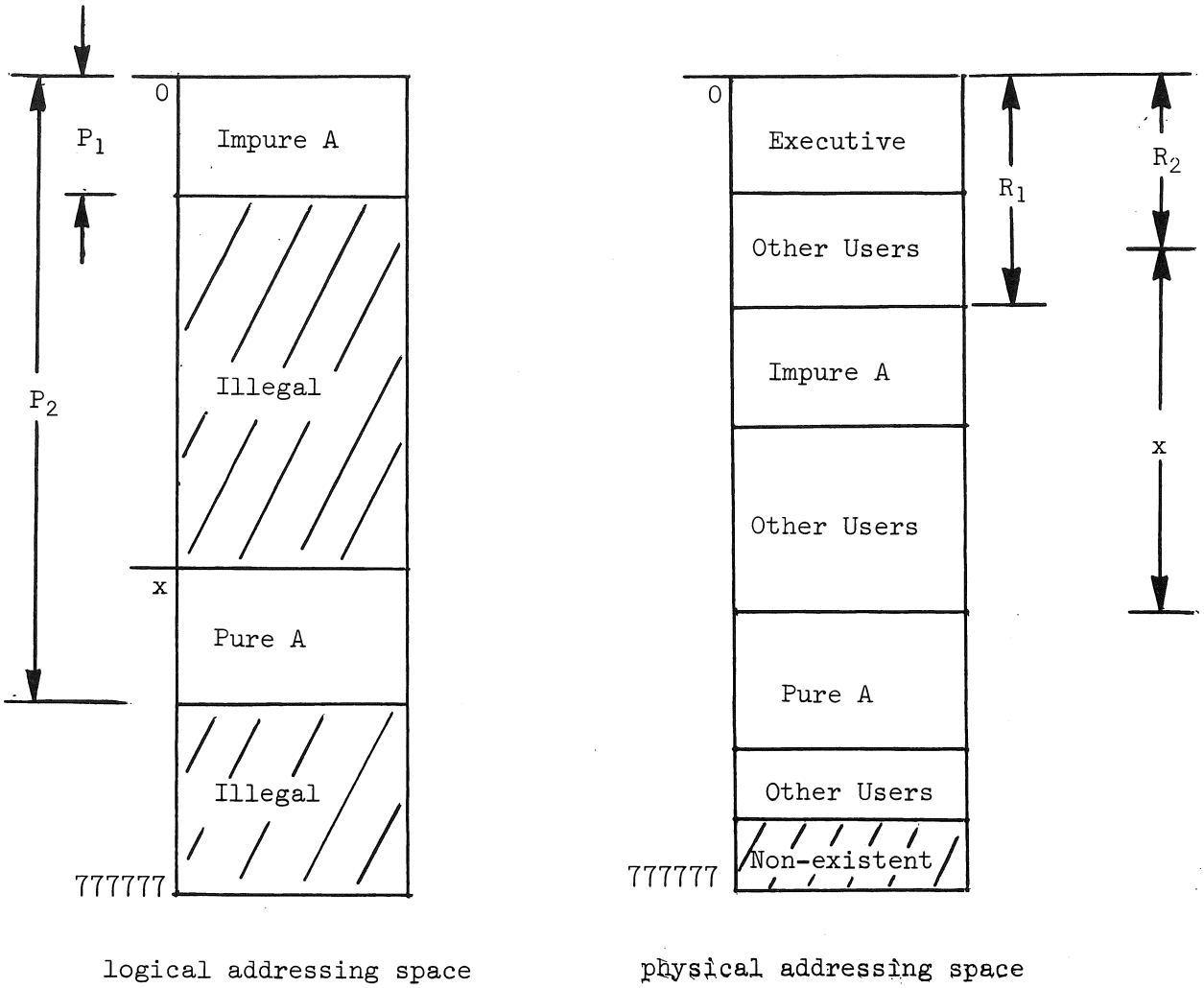


Figure 3. Relation between Logical and Physical addressing space with dual protection and relocation

There is no doubt that the re-entrant hardware will improve the efficiency of the multiprogramming capability of the PDP 10. The executive and system programs such as the FORTRAN compiler have been modified to make use of the re-entrant hardware. Externally, there will be no noticeable difference in batch jobs. When time sharing is provided, however, it should mean that a larger number of users will be serviced without any reduction in response time.

THE LAW AND COMPUTERS

This article is adapted from the Inaugural Lecture of Douglas J. Whalan, LL.B.(N.Z.), LL.M.(N.Z.), Ph.D.(Otago), Professor of Law in the University of Queensland. The University of Queensland Press is to publish the lecture in its Inaugural Lecture Series.

Professor Whalan was born in New Zealand in 1929, qualifying as a solicitor (1950) and barrister (1951) of the Supreme Court of New Zealand. In 1958, he was appointed Senior Lecturer in Law at the University of Auckland, and gained the first Doctorate of Philosophy in Law to be awarded by a New Zealand University. While on study leave in London in 1963-1964, Professor Whalan furthered his research into the registration of titles to land. From 1964-1967 he was successively Visiting Fellow, Fellow and Senior Fellow in the Research School of Social Sciences at the Australian National University, and, in 1967, he was appointed Professor of Law at the University of Queensland.

Professor Whalan is a member of a Federal Attorney-General's Working Group on Reform in the Australian Capital Territory. His publications include a book on the law of trusts and trustees, and numerous articles on the fields of wills, equity, registration of titles to land, and computers and the law.

In his Inaugural Lecture, Professor Whalan pointed out that the issue of the law and computers was of vital importance to the development of law in society. He strongly deprecated the fact that, in its attitude towards computers, Australian society was, like Gaul, divided into three parts - one part fears, one sneers, and the other cheers. Regrettably, said Professor Whalan, the law barely hears! This is one of the principal themes of the lecture; either lawyers quickly wake up or they will be run over by the rapidly-revolving wheels of the computer revolution. Professor Whalan also discussed the legal problems arising from the application of computer technology to fields such as tort, evidence, individual freedom and privacy. The conclusion he reaches is that

radical legislative changes are urgently needed to meet the demands of the computer age. The lawyer must think for the future, be flexible and ready to adapt; above all, he must recognize that scientific developments can be used to assist him, not to enslave him.

Professor Whalan described briefly four main ways of using the computer in legal practice and analysis. Firstly, the computer is an invaluable tool which can rapidly and efficiently perform the more mundane functions of costing, accounting, and general clerical and office management work. However, the problem of *confidentiality* arises as material will be processed outside the regulating bounds of the law office by an independent person who is not bound by legal ethics. Secondly, the computer provides convenient facilities for storage as a memory bank that never forgets. This power to retrieve information can be used to produce legal documents composed of stored data, and, at a more advanced level, it may be used to assist lawyers in analysing and comparing matter. Thirdly, statistical material can be processed with different variables used, and predictions may be made to assess future trends. England, Professor Whalan pointed out, was using data collected on accident cases and awards in this way. Fourthly, and perhaps the most complex of all, court judgments and statutes could be stored, indexed and analysed to determine prevailing judicial attitudes and plan future legislation. As this process depends very much on the particular interpretation that the person doing the processing gives to certain phrases, the number of combinations and resulting complexity are greatly increased.

Professor Whalan said that he was not being unrealistic in his desire to use computer techniques to revolutionize law and to rationalize law reform in Australasia. His conclusions, in fact, were derived from successful methods that were already operating in several U.S. jurisdictions, and would soon be instituted in England. While he admitted that the great cost of the computer was an inhibiting factor, he said that this was no suitable reason for postponing feasibility studies.

Professor Whalan then went on to discuss some of the areas in which problems were likely to arise with the application of computer technology to most fields of law. One major problem area was the question of the validity of *evidence*. As most data are supplied to a computer through human agency, the *hearsay* rule is held to apply unless all the data are sworn to in the Court by the particular individual

who initially supplied the data. This problem, said Professor Whalan, requires urgent legislation. Professor Whalan also mentioned the difficulties in probing and investigating evidence as he pointed out that identical output can result from the same program and data run on similar computers. With interesting far-sightedness, Professor Whalan suggested the use of the simulation processes to aid in the obtaining of evidence by setting up the conditions of the situation.

Professor Whalan then discussed the issue of *tort* or civil liability, an increasingly significant question as computers become generally accepted in all kinds of enterprise. In this context, Professor Whalan pointed out that it could become negligence *not* to use the superior facilities of a computer! Will it, he asked, be negligence to disregard the use of simulation models to test stresses and strains when building an aircraft or a motor car or a bridge? Will it be negligence for the doctor to ignore the availability of computer diagnostic programs? Problems are also raised as to precisely where liability lies should a mistake occur. These are only a few of the issues that will be reality for Australian law in the very near future.

While Professor Whalan indicated that there was cause for concern with numerous other fields such as insurance, copyright, patents, property and labour relations, he felt that the issues of *privacy* and *individual freedom* in particular required serious thought. He warned that, if no legislative action is taken, individual freedom will be blighted and our personal rights of privacy destroyed by indiscriminate access to computerized information. With an apt misquotation of Sir Winston Churchill, Professor Whalan said that never in the history of human endeavour has there been the opportunity for so much to be made available to so many by so few. He pointed out that not since the Domesday Book of William the Conqueror was such a vast amount of personal information available about the individual, ranging from his health record to his business and financial assets. This was accessible to both governmental and private authorities, to be used and misused. The computer simply collates quickly and efficiently all the scattered records existing in various places about each person. With grim accuracy, it can add and delete information, recall selected facts instantly, and, ominously, forget nothing. Professor Whalan pointed out that legislation must be enacted concerning data banks: it should not be

piecemeal, but rather an overall legislative provision dealing with the problem as a whole.

Professor Whalan also suggested that information supplied *unknowingly* through the use of secret or bugging devices be dealt with severely. However, information supplied *knowingly* had considerable complications. Should, for example, information given willingly for one purpose be used for another purpose? Should governmental and private agencies have access to the same memory dumps? How can a person know whether the information supplied about him is accurate or not? If inaccurate, can the person sue for damages, and whom (or what) does he sue? Broadly, said Professor Whalan, rules must be established to restrict access to records, be they accurate or inaccurate. Professor Whalan then put forth the interesting concept of *habeas notae* ("that you may have the writings") to control information banks. Thus, if a person knows or suspects that a record is maintained about him in a data bank or computer complex, he will be legally empowered to ask for a copy of its contents. Professor Whalan said that the idea of *habeas notae* could be extended to include records other than those which are maintained by the computer.

Towards the end of his lecture, Professor Whalan pointed out that today's legislation explosion must be tamed and rechannelled with the aid of computer technology. There was a great need to stem this legislative and interpretative explosion, the 39 miles of law books in the American Library of Congress being a typical example. Professor Whalan urged lawyers to take advantage of the computer age, to use the computer to lessen burdens of technical routine, and make the lawyer freer to think and create. He did not envisage the computer replacing the judge and jury, but said it was a useful tool to handle the ever-growing quantity of legal material. Law must be stable, he said, but it must not stand still. By harnessing computer facilities, the lawyer will be relieved of laborious detail; he will be free to analyse, interpret and plan for the future.