5. COMPIL as a compiler

A version of COMPIL can be used in conjunction with the FORTRAN compiler to produce a mixed code program: Subroutines compiled via COMPIL are compressed into sections of pseudo-instructions. Linking information is set up in such a way that whenever subroutines of interpretive codes are called, the run time system of COMPIL will take over control and interpret the subroutines. When COMPIL is used in this mode, all linking is done through Linker-11.

6. Conclusion

It is realised that COMPIL involves a substantial run time overhead (1·8K word) which means that the core run time trade-off offered by mixed code is exploited only when many or long subroutines are used.

The run time function definition feature is found to be quite useful for certain program packages since on the present PDP-11 system the process of compiling and linking new functions is time consuming and inconvenient.

References

DAKIN, R. J., and POOLE, P. C. (1973). A mixed-code approach, *The Computer Journal*, Vol. 16, No. 3, pp. 219-222.

DALEY, R. C., and DENNIS, J. B. (1968). Virtual Memory, Processes, and Sharing in MULTICS, *CACM*, Vol. 11, No. 5, May 1968, pp. 306-312.

DAWSON, J. L. (1973). Combining interpretive code with machine code, *The Computer Journal*, Vol. 16, No. 3, pp. 216-219.

Testing overflow algorithms for a table of variable size

W. B. Samson

Dundee College of Technology, Bell Street, Dundee DD1 1HG

Large scatter tables which vary in size with time present a problem from the point of view of overflows. This paper describes a program which simulates overflows and indicates which table sizes are to be avoided when the size of the table is to be altered.

(Received February 1975)

1. Introduction

Ecker (1974) has shown that for a given table size a quadratic hash or some related overflow method can be chosen to give a period of search equal to the table size. If the table size varies with time, as may well happen in the case of a scatter table, then it is not usually practicable to alter the overflow algorithm to suit. However, it will usually be possible to adjust the table size by a small amount to give a period of search which is adequate to contain any overflow that is likely to occur.

2. Overflow testing program

The program described below computes the period of search, which is defined as the number of entries which appear in an overflow sequence before any entry is encountered twice, and the capacity for overflow which we define as the number of positions encountered for the first time in an overflow sequence before an endless cycle is entered.

e.g. Table size = M = 10 positions

Overflow algorithm:

$$r\text{th position in sequence} = (1 + \tfrac{1}{2}r^2 - \tfrac{1}{2}r) \bmod M \quad (1)$$

r	*Position*	*Remarks*
1	1	First time encounter
2	2	First time encounter
3	4	First time encounter
4	7	First time encounter
5	1	Second time encounter ∴ Period = 4
6	6	First time encounter
7	2	Second time encounter
8	9	Last first time encounter ∴ Capacity = 6
9	7	Second time encounter
10	6	Second time encounter
11	6	Third time encounter

The period of search in this table is 4 and the capacity for overflow is 6.

A program was written to simulate overflow for various table sizes and overflow algorithms. The following table shows results for table sizes in the region of 1,200 positions with overflow algorithm (1).

Table size (positions)	*Period of search*	*Capacity for overflow*
1,196	63	336
1,197	49	160
1,198	301	600
1,199	65	330
1,200	53	352
1,201	601	601
1,202	302	602
1,203	203	402

In the case of a table size of this order it is clearly more sensible to choose 1,201 positions than 1,197 positions from the point of view of overflows.

3. Conclusions

When one is dealing with a variable table of large size, it is advisable to simulate overflows for table sizes in the region of the preferred table size to find the most favourable size for overflow considerations.

Acknowledgements

I am grateful to Dr. R. H. Davis and Mr. J. R. Lowe for helpful discussions leading to the work described here and to the staff of Dundee College of Technology Computer Centre for their help in preparing and running the program.

Reference

ECKER, A. (1974). The period of search for the quadratic and related hash methods, *The Computer Journal*, Vol. 17, No. 4, pp. 340-343.