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
306-312.

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:

\[ r^\text{th} \text{ position in sequence} = (1 + \frac{1}{2}r^2 - \frac{1}{2}r) \mod M \]

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>
<thead>
<tr>
<th>Table size (positions)</th>
<th>Period of search</th>
<th>Capacity for overflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,196</td>
<td>63</td>
<td>336</td>
</tr>
<tr>
<td>1,197</td>
<td>49</td>
<td>160</td>
</tr>
<tr>
<td>1,198</td>
<td>301</td>
<td>600</td>
</tr>
<tr>
<td>1,199</td>
<td>65</td>
<td>330</td>
</tr>
<tr>
<td>1,200</td>
<td>53</td>
<td>352</td>
</tr>
<tr>
<td>1,201</td>
<td>601</td>
<td>601</td>
</tr>
<tr>
<td>1,202</td>
<td>302</td>
<td>602</td>
</tr>
<tr>
<td>1,203</td>
<td>203</td>
<td>402</td>
</tr>
</tbody>
</table>

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