We are currently using a disk for secondary storage, which is grossly inadequate for heavy swapping, but consider that our swapping problems can be virtually eliminated by the use of Extended Core Storage, which has a transfer rate of 600,000,000 bits per second.

The remaining problem is thus that of software, namely the possibility of completing it before the machine is obsolete. Our experience in this area may be of some interest. The peripheral program constituting the SHARER monitor comprises about 4000 12-bit instructions (perhaps equivalent to 1000 instructions of a conventional large machine). The central program is written in FORTRAN and amounts to about 2000 statements. Thirty-five utility routines, largely written in FORTRAN, have also been provided. About six man years of work have gone into the system thus far, expended by a group averaging four persons over an 18-month period.

We were fairly careful to maintain high standards of documentation during our programming effort. An initial design document specifying all principal system interfaces, table formats, and algorithms was written before any programming was started. This "reference manual" was kept in the form of a deck of cards. Modifications to the system were incorporated into this document as soon as convenient after decisions were made. The effort of documentation has more than paid for itself in reducing confusion in the coordination and debugging of the system programming.

Acknowledgments. The OCTOPUS operating system [7] for the 6600 provided a number of the ideas inherent in SHARER, as did the work of project MAC [8]. The work reported in this paper was performed under the auspices of the US Atomic Energy Commission, Contract AT(30-1)-1480.

RECEIVED FEBRUARY, 1967; REVISED JUNE, 1967

REFERENCES

1. DENNIS, J. B., AND VAN HORN, E. C. Programming semantics for multi-programmed computations. *Comm. ACM 9*, 3 (Mar. 1966), 143.
2. CORBATO, F. J., AND VYSSOTSKY, V. A. Introduction and overview of the multics system. Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Vol. 1, p. 185.
3. GLASER, E. L., COULEUR, J. F., AND OLIVER, G. A. System design of a computer for time-sharing applications. Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Vol. 1, p. 197.
4. VYSSOTSKY, V. A., CORBATO, F. J., AND GRAHAM, R. M. Structure of the multics supervisor. Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Vol. 1, p. 203.
5. DALEY, R. C., AND NEUMANN, P. G. A general-purpose file system for secondary storage. Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Vol. 1, p. 213.
6. OSSANA, J. F., MIKUS, L. E., AND DUNTEN, S. D. Communications and input/output switching in a multiplex computing system. Proc. AFIPS 1965 Fall Joint Comput. Conf., Vol. 27, Vol. 1, p. 231.
7. OCTOPUS/6600 Users Manual. Lawrence Radiation Lab., Livermore, Calif.
8. CORBATO, F. J. ET AL. *The Compatible Time-Sharing System: A Programmer's Guide.* MIT Press, Cambridge, Mass., 1963.

Algorithms

J. G. HERRIOT, Editor

ALGORITHM 312
ABSOLUTE VALUE AND SQUARE ROOT OF A COMPLEX NUMBER, [A2]

PAUL FRIEDLAND (Recd. 13 Feb. 1967 and 16 June 1967)
Burroughs Corporation, Pasadena, California

real procedure *cabs* (x,y);
  value x, y; **real** x, y;
comment This procedure returns the absolute value of the complex number $x + iy$. The procedure provides for the possible overflow on $x^2 + y^2$ in $|x + iy| = \sqrt{x^2 + y^2}$;
begin
  $x := abs\ (x)$; $y := abs\ (y)$;
  cabs := **if** $x = 0$ **then** y **else if** $y = 0$ **then** x **else**
      if $x > y$ **then** $x \times sqrt\ (1+(y/x) \uparrow 2)$
        else $y \times sqrt\ (1+(x/y) \uparrow 2)$
end *cabs*;
procedure *csqrt* (x,y,a,b);
  value x, y; **real** x, y, a, b;
comment This procedure computes a and b where $a + ib = \sqrt{x + iy}$. For $x = y = 0$ we have that $a = b = 0$ so we will assume that x and y are not both zero.
  Solving simultaneously for a and then b ...

$$(1) \qquad a = \pm \sqrt{\frac{x \pm |x + iy|}{2}}, \quad b = y/(2a)$$

and for b and then a...

$$(2) \qquad b = \pm \sqrt{\frac{-x \pm |x + iy|}{2}}, \quad a = y/(2b)$$

To keep the radical real, we will always use the positive sign with $|x + iy|$ and use equation (1) with the sign of "a" taken positive for $x \geq 0$ and (2) when $x < 0$, with the sign of "b" taken positive for $y \geq 0$ and negative for $y < 0$;
begin
  if $x = 0 \land y = 0$ **then** $a := b := 0$ **else**
  begin
    $a := sqrt\ ((abs\ (x) + cabs\ (x, y)) \times 0.5)$;
    if $x \geq 0$ **then** $b := y/(a + a)$ **else**
    begin
      $b :=$ **if** $y < 0$ **then** $-a$ **else** a;
      $a := y/(b + b)$
    end
  end
end *csqrt*

ALGORITHM 313
MULTI-DIMENSIONAL PARTITION GENERATOR [A1]

P. Bratley and J. K. S. McKay (Recd. 23 Aug. 1966, 15 Feb. 1967 and 14 Apr. 1967)
Dept. of Computer Science, University of Edinburgh

procedure *partition* (N, *dim*, *use*);
value N, *dim*; **integer** N, *dim*; **procedure** *use*;
comment A partition of N is an ordered sequence of positive integers, $n_1 \geq n_2 \geq n_3 \geq \cdots \geq n_k$, such that $\sum_{i=1}^{k} n_i = N$. Such a partition may be represented by a Ferrers-Sylvester graph of nodes with n_i nodes in the ith row, e.g.,

```
* * * * *
* * * *
* *
* *
```

represents 5, 4, 2, 2. This two-dimensional diagram may be generalized in a natural way to three, or more, dimensions. More formally, we regard a d-dimensional partition of n as a set S of n nodes, each defined by its non-negative integer coordinates such that

$(x_1, x_2, \cdots, x_d) \in S$ if and only if $(x_1', x_2', \cdots, x_d') \in S$

whenever

$$0 \leq x_i' \leq x_i \quad \text{for all} \quad i = 1, 2, \cdots, d.$$

This generalization reduces to the usual definition when $d = 2$. There is little literature on these generalized partitions. It is with a view to facilitating numerical studies that this algorithm is published.

After generation, each partition is presented to the procedure *use*, which should be supplied by the user for the purpose he requires. *use* has three formal parameters, the first being the name of a two-dimensional integer array, and the second and third being integers giving the size of this array. When the procedure is called by

use (*current, dim, N*)

then the coordinates of the nodes entering into the newly generated multi-dimensional partition will be found in *current* [1:*dim*,1:N]. The parameters of *use* should be called by value, or alternatively care should be taken that neither *dim, N,* nor the contents of the array *current* are disturbed.

References:

1. Gupta, H., Gwyther, C. E., and Miller, J. C. P. *Tables of Partitions.* Royal Society Mathematical Tables, Vol. 4, Cambridge Univ. Press, 1958.
2. MacMahon, P. A. *Combinatory Analysis*, Vol. 2, Cambridge Univ. Press, 1916.
3. Chaundy, T. W. Partition generating functions. *Quart. J. Math. 2* (1931), 234–240.
4. Atkin, A. O. L., Bratley, P., MacDonald, I. G., and McKay, J. K. S. Some computations for m-dimensional partitions. *Proc. Cambridge Phil. Soc.* (to appear);

```
begin
  integer i; integer array current [1:dim, 1:N],
    x[1:dim,0:(N−1)×dim];
  procedure part (n,q,r); value n, q, r; integer n, q, r;
  begin integer s, i, j, k, p, m, z;
    for p:= q step 1 until r − 1 do
    begin
      for i := 1 step 1 until dim do current [i,n] := x[i,p];
      if n = N then begin use (current,dim,N); go to L2 end;
      s := r;
      for i := 1 step 1 until dim do
      begin
        for j := 1 step 1 until dim do x[j,s] := x[j,p];
        x[i,s] := x[i,s] + 1;
        for j := 1 step 1 until dim do
        begin
          if x[j, s] = 0 then go to L3;
          for k := 1 step 1 until n do
          begin
            for m := 1 step 1 until dim do
            begin
              z := if j = m then 1 else 0;
              f ciurrent [m, k] ≠ x[m,s] − z then go to L4
            end;
            go to L3;
L4:
          end k;
          go to L5;
L3:
        end j;
        s := s + 1;
L5:
      end i;
      part (n+1,p+1,s);
L2: end p
  end part;
  for i := 1 step 1 until dim do x[i,0] := 0;   part (1,0,1)
end partition
```

REMARK ON CORRECTION TO CERTIFICATION OF ALGORITHM 279 [D1]
CHEBYSHEV QUADRATURE [F.R.A. Hopgood and C. Litherland, *Comm. ACM 9* (Apr. 1966), 270 and *10* (May 1967), 294]

Kenneth Hillstrom (Recd. 26 June 1967)
Applied Mathematics Division, Argonne National Laboratory, Argonne, Illinois

There are two corrections that should be appended to the certification of Algorithm 279.

Due to programming error, the integrand function routines for e^{-x^2} and $sin(x)+1$, used by the Chebyshev routine, incorrectly evaluated the functions at $x = 0$, thus delaying convergence.

The revised Chebyshev routine still converges more rapidly than the original scheme in the first two examples, but the advantage is much less pronounced than previously indicated.

The amended Table I should read as follows, with the numerical corrections italicized.

TABLE I

Integrand	*A*	*B*	*EPS*	*VI*	*Routine*	*VA*	*Number of function evaluations*
e^{-x^2}	0	4.3	10^{-6}	0.886226924	Havie	0.886226924	17
					Romberg	0.886226925	65
					Chebyshev	*0.8862269261*	*33*
					Chebyshev (Rev.)	0.8862269258	17
$sin(x)+1$	0	2π	10^{-6}	6.283185308	Havie	*6.283185307*	*3*
					Romberg	*6.283185307*	*3*
					Chebyshev	*6.2831853086*	*9*
					Chebyshev (Rev.)	6.2831853089	5