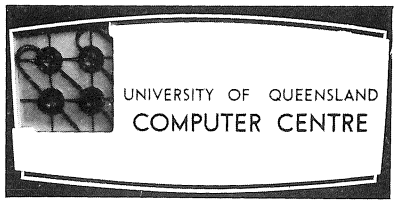

COMPUTER CENTRE BULLETIN

Volume 3, Number 11.
2nd November, 1970.

Editor:
S. J. Barry.

OPERATIONS SCHEDULE BOARD

The schedule board for PDP-10 operations has been in use for three weeks now. It is located in the clients' room and is intended to notify users of running times. While the Centre will attempt to maintain this schedule as much as possible, it cannot guarantee the proposed availability of the system. Times specified can only be regarded as approximate.

It is anticipated that there will be an increased requirement for System Development testing time over the next few weeks. This is for preliminary testing of remote terminal developments. Users will be advised of changes to the normal schedule via the schedule board.

CHRISTMAS SHUT DOWN

The Computer Centre will shut down for the Christmas period at midnight on Wednesday 23 December 1970. There will be no processing of work on the 24 December to allow the staff to finalise end-of-year operations and accounting. As usual, the Centre will be closed over the Christmas - New Year period and will re-open for business at 9 a.m. on Monday 4 January 1971.

FORTRAN COMPILER ERROR

The FORTRAN compiler does not create the correct code when both variable array dimensioning and an Arithmetic Statement Function are used in a subroutine, as illustrated in the following example:

```
SUBROUTINE TEST (A,N,B)
DIMENSION A(N,N)
FN (X,Y) = X+Y
.
.
.
```

The compiler at present does not create the necessary jump instruction to skip the execution of the Arithmetic Statement Function on initial entry to the subroutine. The only solution at present is to reorganise the subroutine so that an Arithmetic Statement Function is not used. This fault is cured in a subsequent version of the compiler which will be released when it has been adequately tested.

LIBRARY ACCESSIONS

This is a list of those books accepted by the University of Queensland Libraries during August 1970 which may be of interest to readers of the Bulletin.

DOLBY, J. L. *Computerized library catalogs* 1969 (025.3 DOL Main)

GRAY, Max *Documentation standards* 1969 (029.7 GRA Engin.)

SALTON, Gerald *Automatic information organization and retrieval* 1968 (029.7 SAL Engin.)

RODGERS, William H. *Think; a biography of the Watsons and IBM* 1969 (338.7616518 WAT 3 ROD Main)

WHALAN, Douglas J. *The law and computers* 1970 (340 WHA Main)

CONFERENCE ON APPLICATIONS OF ELECTRONIC DATA PROCESSING : STATE AND LOCAL GOVERNMENT. University of Georgia, 1965
EDP systems in public management 1968 (350.0018 CON Main)

BELLMAN, Richard Ernest *Methods of nonlinear analysis* 1970 (517.38 BEL Maths)

BELLMAN, Richard Ernest *Algorithms, graphs, and computers* (519.92 BEL Maths)

CONFERENCE ON THE NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS, Dundee, 1969
Conference on the numerical solutions of differential equations 1969 (517.38 CON Maths)

SEMINAR ON THE ROLE OF THE COMPUTER IN THE SECONDARY SCHOOL, Adelaide, 1969
Proceedings of the week-end seminar 1969 (510.78340712 SEM Main)

SMITH, Barry Whitmore, comp. *Australian Computer Society membership survey, 1968* 1969 (Qto 510.7834 SMI Engin.)

KLEINMUNTZ, Benjamin — *Clinical information processing by computer* 1969 (610.28 KLE Main)

WHITEHEAD, T. P. — *Automation and data processing in pathology* 1969 (616.07 WHI Clin.)

AUSTRALIAN COMPUTER CONFERENCE, 4th Adelaide, S.A., 1969 — *Proceedings* 1969 (Qto 612.38195 AUS Elect. Engin.)

BELL, Leonard W. — *Digital concepts* 1968 (621.3815 BEL Elect. Engin.)

HOESCHELE, David F. — *Analog-to-digital, digital-to-analog conversion techniques* 1968 (621.38195 HOE Engin.)

HOVANESSIAN, Shahen A. — *Digital computer methods in engineering* 1969 (620.0018 HOV Engin.)

LOSSIEVSKII, V. L. — *Automation of continuous production processes* 1964 (629.8 LOS Elect. Engin.)

MARTIN, James Thomas — *Teleprocessing network organizations* 1970 (621.38953 MAR Engin.)

SALOMAA, Arto — *Theory of automata* 1969 (629.89 SAL Main)

VAN VALKENBURG, Mac Elwyn — *Network analysis* 1964 (621.31921 VAN Elect. Engin.)

ARNOLD, Robert R. — *Modern data processing* 1969 (651.8 ARN Engin.)

BIRKLE, John — *Computer applications in management* 1969 (658.0018 BIR Engin.)

Business Management and Computers 1970 (658.00184 BUS Main)

CALDERBANK, Valerie Joyce — *A course on programming in FORTRAN IV* 1969 (651.8 CAL Chem.)

CLULEY, John Charles — *Electronic computers* 1967 (651.8 CLU Chem.)

GEAR, Charles William — *Computer organization and programming* 1969 (651.8 GEA Engin.)

GREENWOOD, William T. — *Decision theory and information systems* 1969 (658.4 GRE Main)

GRISWOLD, R. E. — *The SNOBOL 4 programming language* 1968 (651.8 GRI Engin.)

ILIFFE, J. K. — *Basic machine principles* 1968 (651.8 ILI Engin.)

KIVIAT, Philip J. *The Simscript II programming language* 1968 (651.8 KIV Engin.)

KRAUSS, Leonard I. *Administering and controlling the company data processing function* 1969 (Qto 658.0018 KRA Engin.)

LEV, Baruch *Accounting and information theory* 1969 (657 LEV Main)

MCKINSEY and COMPANY *Unlocking the computer's profit potential* 1968 (651.8 MACK Main)

PHILLIPS, George McArtney *Computers* 1969 (651.8 PHI Engin.)

PRITSKER, A. Alan B. *Simulation with GASP II* 1969 (651.8 PRI Engin.)

SYMPOSIUM ON CRITICAL FACTORS IN DATA MANAGEMENT, University of California, Los Angeles, 1968 *Critical factors in data managements* 1969 (658.0018 SYM Engin.)

FEATURES OF ATLAS - HARDWARE AND SUPERVISOR

M. J. McLean

This is the second half of Mr McLean's seminar on the Atlas computer. Last month's Bulletin carried a description of the hardware features; this section outlines the structure of the Supervisor.

The One-Level-Store consists of the sixty-four 512-word pages of core store and a hundred and ninety-two 512-word sectors on the drum. This gives a total effective store which can hold 256 blocks or 128K. The organization of the One-Level-Store is centred on certain directories which are maintained by the system.

The Block Directory contains one entry for each page and sector (ie. one entry for each block in the One-Level-Store). The directory is used to answer the question 'Where is block x of program p?'. Each entry indicates:

(i) The number of the page or sector which is represented

(ii) If the page is full or empty

(iii) The number of the block occupying it if it is full.

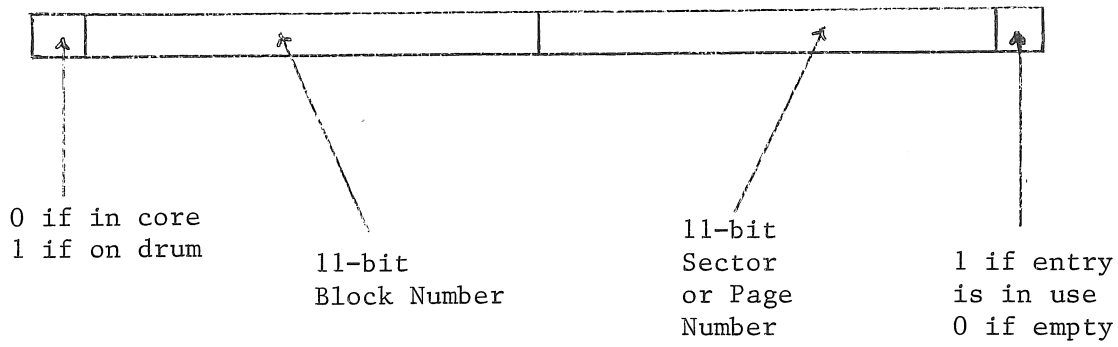

Figure 5. Format of Block Directory Entry

The Block Directory is divided in areas, one for each program which is currently active. The number of entries in each area is the maximum number of blocks which that program is allowed to use (this number is specified by the User in the Job Control Information). The Block Directory is accessed via the Program Store Directory. This has 8 entries; one for each of the seven User Programs which can be active simultaneously, and one for the Supervisor. Each entry specifies the start of the corresponding Block Directory area relative to the start of that directory and the size of the area (Fig. 6).

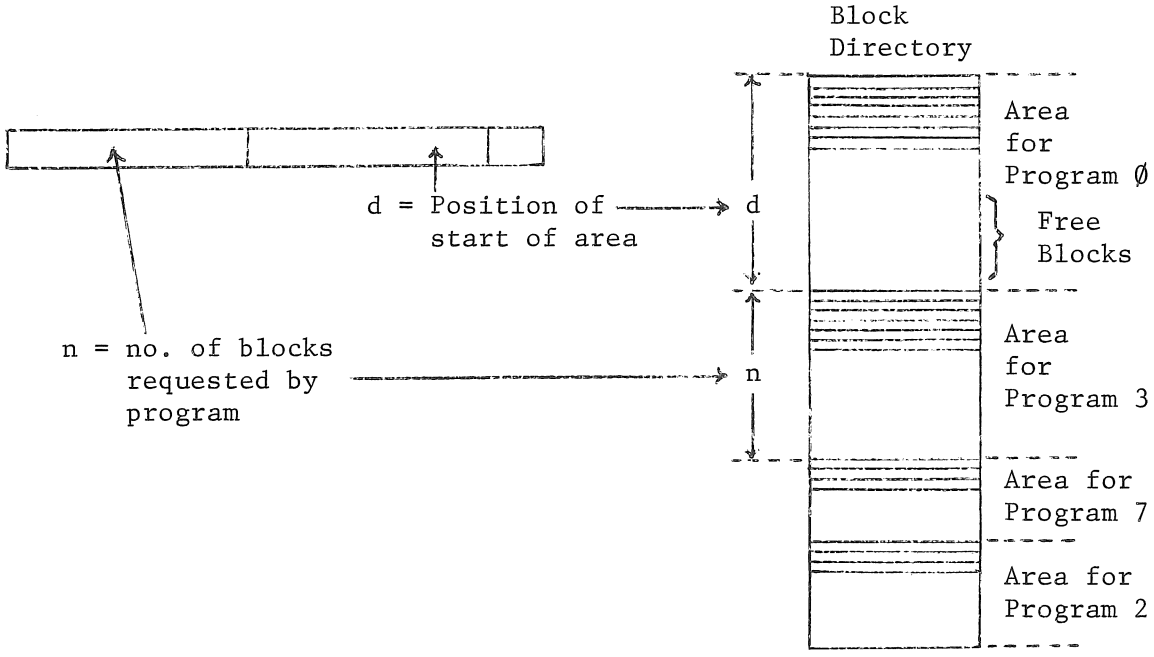

Figure 6. Entry of Program Store Directory

When a non-equivalence interrupt occurs the interrupt routine has a look to see which job is currently running. It then consults the Program Store Directory to find the correct area in the Block Directory, and then searches these entries looking for the missing block number. The search will reveal one of three possible situations.

(i) If the block is on the drum then the job is halted and a drum transfer is initiated. The system always maintains one empty page in core so that there is room to bring in a new block. When the drum transfer is finished the drum service routine initiates the routine which will set up the Page Address Registers and release the job.

(ii) If the block is in core, but locked out, it means that the block is currently in use as a buffer for Input/Output. In this case the job is simply halted. The tape or drum service routines will release the job when the transfer is complete.

(iii) If the block is not listed in the Block Directory then a search is made for an entry which is not in use. If such an entry is found then it is set up as the new block occupying the vacant page in core and the program is released. If no empty entry is found then the program is attempting to use more blocks than it originally requested. In this case the program is terminated as requiring excess blocks.

In order to maintain one empty page in core it is necessary to write blocks back to the drum when they are not being used. The choice of which block to write back is made by a Learning Program which works on statistics about the occupancy of each page. These statistics are gathered by the Instruction Counter Interrupt Routine. This routine is initiated by an interrupt every time 2048 user instructions have been executed. Associated with each page of core is a single bit flag called the Use Digit, which is automatically set whenever a page is accessed. The Instruction Counter Interrupt reads these flags, notes their status and clears them to zero. The drum Learning Program calculates for each block in core two parameters; t and T. t is the number of Instruction Counter Interrupts which have occurred since the block was last referenced. T is the number of Instruction Counter Interrupts for which the block was out of use last time it was selected for writing to the drum. These two parameters can be shown graphically (Fig. 7).

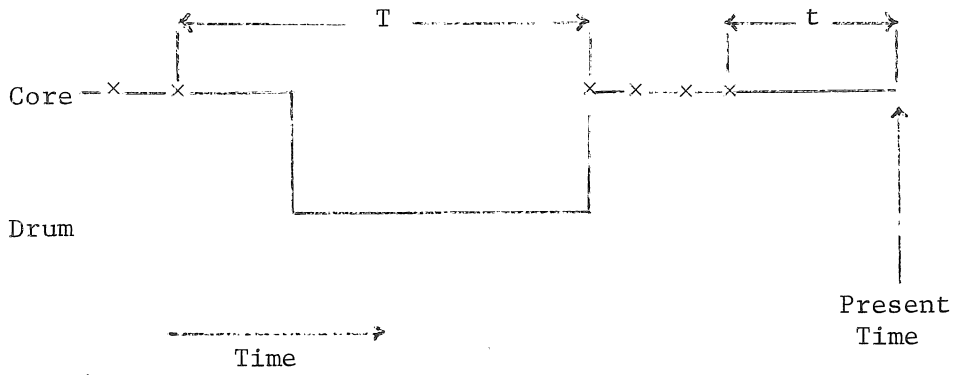

Figure 7. History of Usage of a Single Block

The parameter T is calculated as follows. When a block is written to the drum, the time on the Instruction Counter Clock, t, is recorded in a directory which is almost identical to the Block Directory and is called the Block Status Directory. When the block is read back T can be calculated.

The learning algorithm is largely empirical and therefore no attempt will be made to justify it. The algorithm is as follows.

1. If there is an empty page, do nothing.
2. Choose the page for which $t - T - 1$ has a maximum positive value. (ie. a page which has now not been referenced for longer than it was out of use last time it was written to the drum).
3. For all pages with $t \neq 0$ choose maximum value of $T - t$.
4. If $t = 0$ for all pages, choose maximum T.