Number: 1, 1973 July

Contents:

Introduction

From the editor and secretary

Technical contributions

D.L. Covill, Disk Checkerboarding

B.J.M. Morselt, Application-software for the B6700

D.L. Covill, Problems of the working-set 3HERRIFF

Notes

Mailing list
JUB6700, number 1, 1973 June

Editor
Harry P.J.M. Roumen
Eindhoven University of Technology
Computing Centre
P.O. Box 513
Eindhoven, Netherlands

Secretary
Tiny J.F.M. Fransen
Eindhoven University of Technology
Computing Centre
P.O. Box 513
Eindhoven, Netherlands

JUB6700 is an informal irregularly appearing publication for the users of the Burroughs 6700, for the time being edited at the Computing Centre of Eindhoven University of Technology.

The contents of JUB6700 will mainly consist of technical papers; contributions to JUB6700 may be sent to the editor. Where possible the papers should be typed and camera-ready.

Without charge three copies are sent to the "manager" of each B6700-installation and also to some Burroughs-representatives.

Matters pertaining the mailing list should be sent to the secretary.
Introduction.

Ever since our University ordered a B6700 I have been surprised about the scarce contacts that exist amongst B6700-users with respect to the technical aspects of hardware and software. I know that CUBE and ABCU (and shortly BCUA) do much for the furtherance of our communication, but there should be written communication - especially with respect to technical aspects - , so that one can give comments on proposals or ideas after reflection and consultation. Furthermore the B6700-user can benefit more from a world-wide contact. Discussions that took place during the CUBE-meeting in April 1973 with other B6700-users resulted in our promise to start editing a journal for B6700-users. Here you will find the first issue of JUB6700.

The purpose of this journal is primarily to set-up an exchange of knowledge and ideas about the hardware and the software of the B6700 for the users. Technical papers to be presented at B6700-subgroup-meetings could be published in advance.

The choice of our computing centre as editor is accidental; after some time another user might be willing to act as the editor and I still hope that in due time Burroughs will do it.

I don't know whether JUB6700 will be a viable project, but I see the need for it and I wish ourselves good luck with it.

Ben J.M. Morselt
Eindhoven University of Technology
Computing Centre
From the editor.

As already mentioned in the introduction, JUB6700 provides the B6700-users with the possibility to take note of the ideas that other users have and it gives to every user an opportunity to present his own ideas; concurrently JUB6700 will serve as a medium for circulating technical knowledge amongst all B6700-users. These goals will only be achieved if all B6700-users would look for technical papers that could be published and if they would comment on the published contributions. Several B6700-users have already some technical papers available that could be published and could be useful for the other users. In due time I shall ask some users for technical contributions and I hope that they will react in a positive way to my request. Besides technical papers, contributions of a different kind (like news and comments) are welcome. Permanent categories of JUB will be: Technical Contributions, Comments from the readers and Notes. I will try to have JUB6700 appear at least four times each year and I hope that every reader of JUB6700 will be as enthusiastic as I am.

From the secretary.

It is sure that the mailing list on page 21 and following will be incomplete. Therefore I request the assistance of all readers to help me complete the list as soon as possible by sending corrections and additions. All B6700-installations will receive (for the time being without charge) three copies of JUB6700. In special cases more copies may be obtained on demand.
DISK CHECKERBOARDING
by D.L. Covill
(San Diego: Computer Center; La Jolla, California 92037-USA).

1. DEFINITIONS.

Available Disk is the total number of segments of disk storage available for assignment. It is possible, however, that a disk request may be for a number of segments that is not available in one contiguous area. In this case the requesting task hangs on a "NO USER DISK". If the requesting task is the MCP or other essential system software a system hang results.

Since UCSD defines OLAYROW size as 700 segments (the same as DECK and BD rows), it is critically important that the system have available disk in areas of at least 700 segments.

Usable Disk is therefore defined as the total number of segments available in areas large enough to be used by the system software, 700 segments or more at UCSD.

Checkerboarding is defined as the difference between Available Disk and Usable Disk; that is, the amount of Available Disk which is in areas too small to be Usable by the system.

Some checkerboarding will obviously occur due to the natural process of multiple tasks asynchronously getting and releasing areas. Additionally, and more important, any process which gets areas larger than those it releases will in itself cause checkerboarding.

2. PROBLEM.

UCSD regularly experiences disk checkerboarding at rates of 50,000 segments per day. This seems in excess of any reasonable result of random activity, so some investigation has been performed. Two problems have been found.
a. The MCP procedure ENTERUSERFILE does not request a disk area of the size needed for the header it is passed. Instead it gets a "work area" of 655 segments, uses it as necessary, and returns the rest. This logic must inevitably cause checkerboarding at a rate proportional to the rate of file creation, because the removal of a file header will not, in general, release an area usable by the process which allocates a new one. Furthermore, the peculiar size of 655 segments virtually guarantees that the disk will be "sawed out" of a 700 segment BD or OLAY row.

b. The GETUSERDISK routine uses a two-stage search, each stage of which attempts to fit the smallest feasible area. The problem here is that, after a suitable elapsed time, available areas tend to become distributed randomly across the fine tables, and the size of the largest available area in the fine tables has no necessary relation to the size of any other area in that fine table (except, of course, that it's bigger). Allocation is therefore effectively from a randomly selected fine table as long as its largest area is usable. This tends to saw up large available areas more than is desirable.

(A simple patch was tested to correct this by selecting the fine table with the lowest address which could handle the request. It worked fine except that the rate of checkerboarding was much worse, at which point we found problem (a)!).

We conclude that problem (b) is important but cannot be corrected until the much more important problem (a) is fixed.

3. RECOMMENDATIONS.

3.1. Fix ENTERUSERFILE.

It is most important to correct the operation of ENTERUSERFILE. It should make as many calls on GETUSERDISK as it needs to, for the exact sizes it needs. This is complicated enough, however, that we need some advice at least before attempting it.
(Note that the concatenating of headers and directories, which economizing on GETUSERDISK calls at enter time, guarantees the longest possible directory search time in addition to maximum checkerboarding).

3.2. Change GETUSERDISK Strategy.
A one-card patch to GETUSERDISK at 44596000 will filter small areas to low addresses and large areas to high addresses, thus reducing the random tendency to checkerboard, with no increase in execution time. As noted before, it must not be done until ENTERUSERFILE is corrected.

3.3. Redesign the Table Sizes.
Our disk tables have about 150 coarse entries, each referring to a 30 word fine table. The search thus looks at 180 entries to get a best-of-30 fit. A simple reversal of size would result in a best-of-150 fit for the 180 entries looked at, with little or no effect on core or IO requirements. It is almost certain that a lot better can be done with a little more thought; the dope vectors are 405 words long and are largely unused!

3.4. Make System Area Sizes Modular.
We have tried to use 700 segments as a system module at UCSD to minimize the effect of checkerboarding. But we are constantly being defeated by new system programs and files which use randomly selected area sizes. Particular offenders are the compilers, which generate default area sizes as a function of record size. A system module should be defined, system files designed on that module, and the compilers should generate default area sizes as a function of that module. The module should, if possible, be modifiable by the installation, but that's not nearly so important as that there be one.

Disk Checkerboarding

Available Disk (K Segs):

```
| 447 | 401 | 373 | 396 | 377 | 381 | 338 | 422 | 430 | 440 |
```

Available Areas ≥ 700 Segs

```
| 442 | 383 | 345 | 357 | 330 | 326 | 281 | 367 | 376 | 387 |
```

Available - (Avail. ≥ 700) = "CHECKERBOARDING"

Cold Start at 0800

Time of Day

Thursday April 20, 1973

End of operations
APPLICATION-SOFTWARE FOR THE B6700
by Ben J.M. Morselt
(Eindhoven University of Technology, Computing Centre).

1. INTRODUCTION.
Our University (Eindhoven, Netherlands) has a B6700 since January, 1973 and is in general happy with this machine. The only aspect in which Burroughs has disappointed us, is the lack of application-software available for the engineer. We do not reproach Burroughs for this, because we were aware of it beforehand. On the other hand we - and also some other users, I think - have to fill this gap; and if Burroughs wants to penetrate in universities, it is also important for Burroughs. In which way should this problem be solved? In my opinion the best solution would be the construction of a structured library of procedures by cooperation of a number of B6700-users under coordination of Burroughs.

2. WHY NO STANDARD SOFTWARE-PACKAGES?
The more or less standard software-packages that are nowadays available have the advantage that they are ready-to-use for a small class of problems. A major disadvantage is the "black-box"-principle. Some people view this also as an advantage, but (a) it makes intellectual control for the user impossible and (b) it is practically impossible to modify the programs (and our experience proves that after one or two years the user wants modifications).
Furthermore, the number of conventional software-packages is increasing very fast, whereas in these different programs many parts of process-descriptions are equivalent (we found in 10 out of 12 programs of a well-known library the Runge-Kutta-procedure; even in "Numerals" of Burroughs there are several repetitions of the same text).
3. In Eindhoven we reject software-packages as a final solution for the problems of the user. On the other side it is for the user impossible to write himself all the programs he needs (too much time consuming, too difficult, too much multiplicated activities). Dijkstra [1] says about programming:

"I want to view the main program as executed by its own, dedicated machine, equipped with the adequate instruction repertoire operating on the adequate variables and sequenced under control of its own instruction counter, in order that my main program would solve my problem if I had such a machine. I want to view it that way, because it stresses the fact that the correctness of the main program can be discussed and established regardless of the availability of this (probably still virtual) machine: I don't need to have it, I only need to have its specifications as far as relevant for the proper execution of the main program under consideration.

For me, the conception of this virtual machine is an embodiment of my powers of abstraction, not unlike the way in which I can understand a program written in a so-called higher level language, without knowing how all kinds of operations (such as multiplication and subscription) are implemented and without knowing such irrelevant details as the number system used in the hardware that is eventually responsible for the program execution.

In actual practice, of course, this ideal machine will turn out not to exist, so our next task - structurally similar to the original one - is to program the simulation of the "upper" machine. In programming this simulation we have to decide upon data structures to provide for the state space of the upper machine; furthermore we have to make a bunch of algorithms, each of them providing an implementation of an instruction assumed for the order code of the upper machine.

Finally, the "lower" machine may have a set of private variables, introduced for its own benefit and completely outside the realm and scope of the upper machine."
But this bunch of programs is written for a machine that in all probability will not exist, so our next job will be to simulate it in terms of programs for a next-lower machine, etc. until finally we have a program that can be executed by our hardware. If we succeed in building up our program along the lines just given, we have arranged our program in layers. Each program layer is to be understood all by itself, under the assumption of a suitable machine to execute it, while the function of each layer is to simulate the machine that is assumed to be available on the level immediately above it".

For the B6700 this idea can be realized by a procedural extension of the languages (see Dennis [2] and Goos [3]) in the following manner:

On the base of the statements and procedures of Burroughs Extended Algol we will construct a structured library of procedures (routines and functions); each procedure is described in terms of underlying procedures, whereby the specifications and the names of the procedures must be chosen with much care (who has standardized the name "sin"?).

4. WHY A STRUCTURED LIBRARY OF PROCEDURES?

The user is able to read the program text, which will be relatively short as it is expressed in underlying procedures, and therefore can be controlled. The user will so receive the possibility to build his own virtual machines.

The computing centre may have the following considerations:
- possibility to prove the correctness;
- demonstration to the user of structured programs;
- permanent results of programming-activities and therefore more motivation for the programmers;
- an easy continuous way of extension of the library with new process descriptions;
- software-packages in the present form will be superfluous (one should avoid the "black-box"-principle); they can be described in terms of highest-level-procedures;
- the handicap of Burroughs in the field of application-software is resolved in a manner corresponding to the Burroughs-design-philosophy;
- it is not inconceivable that in the future the hardware will be extended with "hard procedures".

A disadvantage will be some inefficiency in the usage of computer-time.

5. REALIZATION.
Eindhoven has started the library mentioned above with a number of numerical and stringhandling procedures.
Naturally other B6700-users will have also a number of procedures and programs.
In my opinion it would be of benefit to most of the (scientific) B6700-users if these activities could be coordinated, and in my opinion Burroughs should be the coordinator.


PROBLEMS OF THE WORKING-SET SHERRIFF
(WITH SOLUTIONS)
by D.L. Covill
(San Diego: Computer Center; La Jolla, California 92037-USA).

UCSD has been an enthusiastic user of the Working-set provisions in the II.3 MCP, as well as an earlier version we installed in II.2. It definitely improves system throughput, and there is never any question as to whether or not it will be used.

Our observations, however, indicated that the MCP procedure WSSHERRIFF suffered from three problems; it never actually realized the overlay goal specified, it suspended too many jobs and restarted them in the wrong order, and measurement revealed that it accounted for over 60% of the MCP's CPU time!

This memo analyzes these problems and describes the solutions we have implemented. The solutions are simple, and at UCSD have resulted in improved system throughput plus a 50% reduction in MCP CPU time. Since the II.4 WSSHERRIFF is effectively identical to the original II.3 release, this analysis remains current and valid.

(Please note that this memo follows the MCP's spelling of Sherriff rather than the dictionary's).

1. MEETING THE OVERLAY GOAL.
   Our measurements showed that, with a system OLAYGOAL of 100% per minute, the actual Sherriff performance was about 25-30%! The key lies in the basic operating strategy of a three-second time-tunnel between cycles of the Sherriff. The computations are based on a time interval of three seconds, thus containing the assumption that Sherriff operations is virtually instantaneous. Measurements showed an average of 7 seconds elapsed, with individual cycles as long as 45 seconds under certain conditions!
The solution was to modify the Sherriff's basic operating strategy to insure that it starts a cycle every three seconds.

a. The clock time is recorded when starting a computation cycle.

b. When the cycle is complete, the time tunnel is entered for only the remainder of the three-second period.

c. If three seconds elapse before the cycle is complete, overlaying stops and Sherriff returns to start a new cycle. (Overlay goals need to be recomputed by now anyway). Time in the time tunnel will be 0.

With this patch, we reduced our System Overlay goal from 100 to 50 and actually achieved the 50%.

2. SUSPENDING AND RESUMING TASKS.

There are two separate problems here.

2.1. Suspending too many tasks.

Our analysis showed that tasks are suspended because of some traumatic (relative to core usage) event elsewhere in the system — a "spike" in core requirements. The priority order is observed when suspending; it just suspends too many. The reason is that when a job is suspended its OLAYGOAL remains the same, so the Sherriff takes the usual percentage of its core. This is almost never enough to recover AVAILMIN, so the next cycle the Sherriff suspends someone else. We frequently got into situations where Interactive tasks were suspended.

The solution is to (effectively) change the OLAYGOAL for a suspended task so that the Sherriff will take all its core. This greatly improves the chance of reaching AVAILMIN in the current cycle and thus not suspending a second job. This patch has effectively reduced the level of suspending during core spikes. (NOTE: It still doesn't do as much as it could because we don't take the code space on the same basis, due to possible reentrancy. See paragraph 3.).
2.2. Resuming Tasks.

The first problem is that tasks are not resumed soon enough.
The computation requires enough available core (in excess of AVAILMIN) to hold the entire working set of the task. But the entire working set hasn't been overlaid - the save core certainly hasn't! So it is sufficient to have enough core for the difference between COREINUSE and the working set.

The second problem is that the algorithm resumes the highest-priority task for which there is enough core. With multiple priorities, this frequently results in resuming a lower-priority, long-running job which effectively kills the chances of resuming a higher-priority job that needs more core (like a compiler). The solution to this one is to resume only the highest-priority task; if there isn't enough core wait for some more to build up.

We submitted these fixes as FTR 113-0539 on January 24, 1973. To date it has not been acted upon.

3. EXCESSIVE CPU TIME.

We were forced to install an instrumentation patch to WSSHERRIFF, with an Installation Intrinsic to extract the results, in order to gather data on why the CPU time was so large. Some of the results:

a. With a heavy, but not overloaded, mix, total MCP CPU time averaged 25-30 seconds per minute elapsed. Sherriff averaged 60% of this total!

b. Sherriff CPU time approximated 30 ms. per area overlaid/forgotten!

c. Stack searches, originally thought to be the problem, are only about 3 ms. each.

d. Buzzing MEMLOCK accounted for 10-12% of the Sherriff's CPU time. (This is due to interference with GETSPACE, which has to handle all activity in the Swapspace, including overlaying).
e. The areas actually forgotten were about 60% code areas.
f. The average size of all areas forgotten is 150-175 words.
g. Data areas overlaid divide almost evenly between 1-29 words, 30-99 words, and 100 words or more.

Thus, CPU time is clearly excessive, but the cause is not clear.

3.1. Problem Analysis.

The problem, it turns out, results from Sherriff's approach to management of code (D1) space. Sherriff computes overlay requirements for D2 stacks only.

Each D2 has its corresponding D1 included in the computation, but the goal is recorded in the D2. (Presumably the reasons are the possibility of re-entrancy and fact that different D2 task stacks could have different OLAYGOALS while using the same D1).

There are two unfortunate results:

a. When a code segment is found in core by the Sherriff, it has to check all the task stacks in the system for potential owners and do the bookkeeping for all of them if the area is forgotten. Since over half of all areas are code, this is the cause of the CPU time!

b. Since we can't depending on having the D2 for a Swap job, swap jobs are exempted. The result of this is that the code segments for a swap job are normally exempt from management by the Sherriff! This reduces the Sherriff's ability to maintain an AVAILMIN pool, thus increasing the number of tasks suspended, and increases core costs for swap jobs, because once-used code is never forgotten.

It also exacerbates the CPU time problem because it guarantees that a large number of the expensive D2 stack checks will be fruitless.

3.2. Solution.

We have implemented a solution as follows:

a. Overlay goals computed for D2 stacks do not include the D1.

b. D1 stacks have their own goal, computed using the system's OLAYGOAL. (Any difference between multiple D2's is immaterial, because the effect of the original mechanism was always to use the larger).
c. When a code segment is encountered during overlaying, its D1 stack is checked and decremented directly, without reference to any D2's.

3.3. Results.
The results exceeded our expectations.

a. MCP total CPU time is reduced by 50%.
b. Sherriff CPU is now 6-7 ms. per area overlaid/forgotten.
c. The Sherriff is now active only about 10% of elapsed time.
d. The system OLAYGOAL is met within 1 or 2%. There is a significantly larger pool of available memory.

Most important, we have regained the ability to maintain batch throughput during periods of heavy interactive load. Figure 1* is an example analysis during current daytime operations.

4. COMMENTS.

4.1. Figure 2* is a listing of the patches to WSSHERRIFF which effect all the solutions described herein. They are believed valid for II.4, but have not yet been tested in that version.

4.2. A little more should be done on the suspending of tasks. If the task suspended is something like a compiler, we can get more core from the code than from the data, and cheaper, but the way I fixed the D1 problem we don't get it. A solution would be, when we find a D2 that's suspended, to check the D1. If the usage count is 1, then set the goal to take all of its core too. The problem is to do this in a way that doesn't get tangled up in its own bookkeeping.

4.3. We're not at all convinced that having different OLAYGOALS for different tasks accomplishes anything. Theoretically, maybe it does. In actuality, the number of words to be overlaid by the Sherriff in one pass (1/20 of OLAYGOAL % of the non-save core) is small relative to the average segment size of 150-175, so what you actually get varies widely anyway. It would be simpler just to use one system goal.
If multiple goals are justified, Dr. Bowles' studies indicate that it might be desirable to have higher goals for the MCP and MCS's. In any event, there is no demonstrable linear relationship between priority and proper overlay rate.

4.4. We also have a patch which exempts areas of less than 10 words from action by the Sherriff. A definite major improvement can be shown on a program (admittedly pathological, but real) which does a matrix multiply on an array dimensioned 370 x 3 x 3! We are still arguing whether the patch is better for the general case or not, since some part of what Sherriff doesn't do has to be done later by GETSPACE.


* For technical reasons the original computer listings could not be published; a copy of the listings is given. - Ed.
UCSD CORE MANAGEMENT STATISTICS

LAST HALT-LOAD WAS AT 06:54:40 (348.8 MINUTES)
19 TASKS IN THE MIX  CORE = 105744  SAVE 94779  OLAY 45237 AVL
WORKING-SET GOAL: 50%

STATISTICS AT 12:43:28 FOR 348.8 MINUTES

CURRENT DISK IO RATES:

    PBIT = 11.0  SHERIFF = 1.1  GETSPC = 0.0  TOTAL = 12.0
AVGFRAGE CORF = 84K SAVE 86K OLAY 74K AVAIL
MCP TIMES (SECONDS/MINUTE):

    MCP TOTAL:  CPU = 7.7  IO = 20.7
SHERRIFF:  CPU = 1.8 (23% OF MCP)  IO = 2.3 (11% OF MCP)

WORKING-SET PERFORMANCE:

    SHERRIFF ACTIVE/ELAPSED = 1908/20710 SECS (9%)
    ACTUAL OLAY = 14723714 WORDS IN 89860 AREAS = 42212 WDS/MIN = 49%
    AREAS FORGOTTEN: 258/MINUTE (72% CODE, 28% DATA)
    CPU TIME/AREA FORGOTTEN: 6.9 MILLISECONDS

STATISTICS AT 12:44:30 FOR 1.0 MINUTES

CURRENT DISK IO RATES:

    PBIT = 5.7  SHERIFF = 1.2  GETSPC = 0.0  TOTAL = 7.0
AVGFRAGE CORF = 103K SAVE 96K OLAY 48K AVAIL
MCP TIMES (SECONDS/MINUTE):

    MCP TOTAL:  CPU = 7.1  IO = 16.8
SHERRIFF:  CPU = 1.7 (25% OF MCP)  IO = 2.1 (13% OF MCP)

WORKING-SET PERFORMANCE:

    SHERRIFF ACTIVE/ELAPSED = 6/62 SECS (10%)
    ACTUAL OLAY = 46896 WORDS IN 342 AREAS = 45383 WDS/MIN = 47%
    AREAS FORGOTTEN: 331/MINUTE (80% CODE, 20% DATA)
    CPU TIME/AREAS FORGOTTEN: 5.2 MILLISECONDS
PATCH NUMBER 22

$# MCP PTR.539 — REVISED CRITERIA FOR REINITIATING AN STED STACK

$ SET VOIDT SEQ 33063000 + 1000

IF (L := TSK [PRIORITY]) > I
OR (L = I AND (X := TSK [TIMESTARTED]) > K) THEN BEGIN % BETTER
  I := L; K := X;
  IF (AVAILCORE - MEMMIN) > (STACK [J, COREINTEGRAL]/
    (ISK [IOTIME] + TSK [PROCESSTIME])) % WS - COREINUSE
  OR F THEN SN := J % OK. HES IT
ELSE SN := -1; % THE TOP DOG OR NOBODY
END;

$ POP VOIDT SEQ
PATCH NUMBER 81

$# MCP UCSD 280 — WSSHERRIFF IMPROVEMENTS

POLAYRATE = (IF K := STK[OLAYCNTL].OLAYFREQ = 0 THEN OLAYGOAL
ELSE K) #

REAL STARTCYCLE := -CLOCK, ACTIVETIME;
ACTIVETIME := (STARTCYCLE + CLOCK) / ONESECOND;
IF CYCLETIME > ACTIVETIME THEN TIMETUNNEL(CYCLETIME -ACTIVETIME);
STARTCYCLE := -CLOCK;

ZONE

FOR SN := FIRSTUSERSTACK STEP 1 UNTIL MAXSTACKS DO BEGIN
IF (K := STACKINFO[SN].STACKKINDF) = TASKSTACK
OR K = SEGDICTIONSTACK THEN

IF LOCATION(SN) IS INCOREV AND STACKSTATE(SN) GEQ ALIVE THEN

$ SET VOID

STK := STACK[SN,*];
I := STK[COREINUSE] - STK[SAVECOREINUSE];
IF NOT BOOLEAN (STK[USAGE].WSEDF) THEN

% IF STED. TAKE ALL OF IT

I := INTEGER (I : POLAYRATE : FUDGEFACTOR); POLAYRATE := I;
OLAYAMOUNT := OLAYAMOUNT + I;

$ POP VOID

END SN LOOP;

IF ((STARTCYCLE+CLOCK) / ONESECOND) GEQ CYCLETIME THEN GO GETOUT

% BETTER GO AROUND AGAIN AND RECOMPUTE

$ SET VOID

% VOID

$ POP VOID

$ SET VOID

% VOID

$ POP VOID

$ POP VOID
Note: A postponed symposium.

By rumours some persons are already aware of a planned symposium for scientific users of B6700-systems, but it seems wise to give a short report of the facts.

During a visit of Dr. R.S. Barton in Eindhoven the idea came up to have a meeting of scientific users of B6700-systems, in order to collect ideas about improving the use of the B6700-system, especially with respect to application-software (and without the intention to create another usergroup).

As possible discussions points were named:
1. Structured programming concepts
2. Proposals for a structured program library
3. Signal processing applications
4. Optimization (Algol and Fortran)
5. Other languages
6. Hierarchical MCP
7. B1700 architecture

Eindhoven was proposed as the meeting-place; May 1973 as the date. In order to form an impression whether the users would and could attend such a meeting, I wrote to 17 University computer centers with a B6700.

Of the 15 answers received, 13 agreed with the proposal but the time and/or the place appeared to be not suitable.

On the moment we are looking for a new date and place. Suggestions are welcome.

Ben J.M. Morselt
Eindhoven University of Technology
Computing Centre
(postbox 513, Eindhoven, Netherlands).
Note about B6700-MCP-documentation.

During the CUBE-meeting in April 1973 David M. Perlman made a proposal to the present users to cooperate with respect to the documentation of the B6700-MCP.

David himself will coordinate this work.

I hope that in the next issue he will explain in more details the purpose and working method.

Everyone, who is interested may write to:

    David M. Perlman; Computer Center,
    University of California at San Diego,
    La Jolla, California 92037, USA.
Mailing list of JUB6700, June 1973

Institutes

ABCU, Chairman of c/o J.R.F. van Wingerden
Bankgirocentrale, postbox 9120
Amsterdam, Netherlands

ABCU, Secretary of c/o R.R. Fawcett
CMG Ltd, Lennig House Masons Avenue
Croydon, Surrey, England

American Hospital Supply c/o Harvey L. Cohen
2020, Ridge Avenue
Evansion, Illinois 60205, USA

Bankgirocentrale B.V. c/o J.R.F. van Wingerden
Postbox 9120
Amsterdam, Netherlands

Battelle-Institute e.V. c/o M. Büchner
Am Römerhof 35, Postfach 900 160
D-6000 Frankfurt/M. 90, Germany

Boise Cascade E & C Group, Central Computer Facility c/o Joel Koppelman
2 Rector Street
New York, New York 10006, USA

C.F. Braun c/o Sat Chhabra
1000 S. Freemont
Alhambra, California 91802, USA

C&O/B&O Railroads c/o William C. Prinn
2 North Charles Street
Baltimore, Maryland 21201, USA
California State Dept. of Social Welfare  c/o Hugh K. Haines  
744 P Street 
Sacramento, California 95814, USA

Canadian Industries Limited  c/o Ron Horrocks  
Cil House, Box 10  
Montreal 101, Quebec, Canada

Chase Manhattan Bank, N.A.  c/o Jordan L. Bier  
One New York Plaza  
New York, New York 10015, USA

Cimass, Unam  c/o Jaime Jimenez  
Ciudad Universitaria  
Mexico 20, D.F., Mexico

Clevite Engine Parts Div. of Gould Inc.  c/o D.R. Saunders  
1700 St. Clair Avenue  
Cleveland, Ohio 44110, USA

Colonial Mutual Life Assurance Society  c/o P.H. Murton  
330 Collins Street  
Melbourne, Australia

Colonial Pipeline CO.  c/o Lenroy Lowe  
3390 Peachtree RD, N.W.  
Atlanta, Georgia 30326, USA

CUBE, Chairman of  c/o William G. Ferrara  
Federal Reserve Bank of New York, 99 John Street  
New York, New York 06905, USA

CUBE, Secretariat of  c/o Thomas S. Grier  
Burroughs Place  
Detroit, Michigan 48232, USA
Danish Atomic Energy Comm., Computer Group  c/o Leif Hansson
Risø
DK 4000 Roskilde, Denmark

Data Resources  c/o John K. Ahlstrom
29 Hartwell Avenue
Lexington, Massachusetts 02173, USA

DLSC - DDS  c/o Robert J. Knez
Federal Center BLDG, 50 N. Washington St.
Battle Creek, Michigan 49016, USA

Eindhoven University of Technology Computing Centre  c/o Ben J.M. Morselt
Postbox 513
Eindhoven, Netherlands

Federal Reserve Bank of New York  c/o William Ferrara
33 Liberty Street
New York, New York 10045, USA

First Bank System  c/o John E. Jonas
Manager Computer Audit Group, E-901 First National Bank BLDG
St. Paul, Minnesota 55101, USA

First Computer Corporation  c/o E.S. Eichhorn
800 E. First National Bank BLDG
St. Paul, Minnesota 55101, USA

First National Bank of Memphis  c/o Lary Stoltenberg
P.O. Box 62
Memphis, Tennessee 38101, USA

First Pennsylvania Bank  c/o W. Ronald Block
3020 Market Street, Sixth Floor
Philadelphia, Pennsylvania 19104, USA
Florida Dept. of HWY, Safety & Motor Vehicles  c/o Obie McDaniel
Neil Kirkman BLDG
Tallahassee, Florida 32304, USA

Ford Motor Company  c/o William A. Adams
Room 1563, American Road
Dearborn, Michigan 48121, USA

General Mills  c/o Dean Skjonsby
9200 Wayzata BLVD
Minneapolis, Minnesota 55426, USA

Gleason Works  c/o Ronald J. Pedrone
1000 University Avenue
Rochester, New York 14603, USA

Harris Trust & Savings Bank  c/o Clif H. Merry
311 West Monroe Street
Chicago, Illinois 60690, USA

I.N.P.S.  c/o Jose Neves
Au Almirante Barroso 5Y 11 Floor
7000 RIB GB Brazil

Illinois Secretary of State  c/o Gene Brimm
Information Services Division, Centennial Building, Room 508
Springfield, Illinois 62706, USA

Inmont Corporation  c/o Jerri Clemens
1133 Avenue of the Americas
New York, New York 10036, USA

Japan Information Processing Service Co., LTD.  c/o Mr. Shoda Masayuki
Edobashi Building, 1-15-5 Edobashi Nihonbashi
Chuo-Ku Tokyo, Japan
Joseph Frisch Consulting c/o Joseph Frisch
24 Revere Road
Port Washington, New York 11050, USA

Laboratory of Road Data Processing, Ministry of Transport c/o Mr. Karl Anton Andersen
Stationsalleen 42
2730 Herley, Denmark

Marathon Oil Company c/o Henri Berce
P.O. Box 269
Littleton, Colorado 80122, USA

Maryland National Bank c/o Richard G. Schneider
Systems and Programming Dept. NO. 153
Baltimore, Maryland 21203, USA

Massey University, Computer Centre c/o G. Tate
Palmerston North, New Zealand

Michigan Bell Telephone Company c/o Henry Hartson
Operations, 23500 Northwestern HWY. C-185
Southfield, Michigan 48075, USA

Michigan Sec. of State Office c/o John G. Lucia
Data Processing Center
Lansing, Michigan 48918, USA

Midland Bank Limited c/o D.A. Whittworth
Computer Services Dept.,
27/32 Poultry London E.C.2, England

Mitsubishi Oil Co. Limited c/o Mr. Shiro Kasuga
Operations Research & Computer Dept., 1 Shiba Kotohira - CHO
Minato - KU Tokyo, Japan
Philco-Ford Corp.  c/o Joseph R. Phelan
515 Pennsylvania Ave.
Fort Washington, Pennsylvania 19034, USA

Philco-Ford Corp.  c/o Roger N. Bauer
C & Tidga Streets
Philadelphia, PA 19134, USA

Police National Computer Unit  c/o E. Quinney
Romney House Marsham St.
London SWIP 3DY, England

Queen's University  c/o George Taylor
Computing Center
Kingston, Ontario, Canada

Spillers LTD  c/o M.A. Cannon
7-15 Lansdowne Road
Croydon, Surrey CR9 2JG, England

Super Valu Stores  c/o Robert J. Lee
101 Jefferson Avenue S.
Hopkins, Minnesota 55343, USA

Teleprocessing Industries  c/o John Alton
82 McKee Drive
Mahwah, New Jersey 07430, USA

U.S. Treasury Dept.  c/o Robert D. Brown
Oasia-Research Division, Room 512F Main Treasury
Washington, D.C. 20220, USA

United Technology Center, Computing Center  c/o Thomas E. Saunders
Sunnyvale, California 94088, USA
Union Dime Service Corp.  c/o Anthuny Rizzo
1065 Avenue of Americas
New York, New York 10036, USA

Universidade do Rio Grande do Sul  c/o...
Avenida João Pessoa, Porto Alegre
Rio Grande do Sul, Brazil

Universidad Nacional Autónoma de México  c/o...
Ciudad Universitaria
Villa Obregón, Mexico City, Mexico

University of Auckland, Computer Centre  c/o J.C.B. White
Princess Street
Auckland C.1, New Zealand

University of California at Davis, Computer Centre  c/o James Burdette
Davis, California 95616, USA

University of California at San Diego, Computer Center  c/o Dan Covill
P.O. Box 109
La Jolla, California 92037, USA

University of Canterbury, Computer Centre  c/o B.A.M. Moon
Christchurch 1, New Zealand

University of Delaware  c/o Dr. Kevin Jones
Computing Center
Newark, Delaware 19711, USA

University of Denver  c/o W.H. Eichelberger
University Park
Denver, Colorado 80210, USA
University of Helsinki Computing Centre  c/o Lars Backström
Töölönkatu 11
SF-00100 Helsinki 10, Finland

University of Illinois at Urbana Champaign, Center for Advanced Computation  c/o J. Meir
120 Advanced Computation Building
Urbana, Illinois 61801, USA

University of Karlsruhe, Computing Center  c/o Werner Zorn
2, Neuer Zirkel
75 Karlsruhe, W-Germany

University of Otago  c/o...
Dunedin, New Zealand

USAF Academy  c/o Carl Edwards
Colorado 80840, USA

Victoria University of Wellington, department of information science  c/o T. Stallknecht
Postbox 196
Wellington, New Zealand

P.O. Box 1591
Waco, Texas 76703, USA

Westinghouse Electric Corp.  c/o George Sanchez
Hendy Avenue 72-4
Sunnyvale, California 94088, USA

Young & Rubicam  c/o David Guest
285 Madison Avenue
New York, New York 10017, USA
Mailing list of Burroughs-representatives

* Mr. R.W. Macdonald, President,
Burroughs Corporation, Burroughs Place, P.O. Box 418,
Detroit, Michigan 48232 U.S.A.

Mr. R.R. Johnson, Vice President, Engineering,
Burroughs Corporation, Burroughs Place, P.O. Box 418,
Detroit, Michigan 48232, U.S.A.

Mr. B.L. Rouse, Vice President and Group Executive,
International Group, Burroughs Corporation, Burroughs Place,
P.O. Box 418, Detroit, Michigan 48232, U.S.A.

Mr. J.A. McCullough, Vice President and Group Executive,
Business Machines Group, Burroughs Corporation, Burroughs Place, P.O. Box 418, Detroit, Michigan 48232, U.S.A.

Mr. E. Gary Clark, Vice President and Group Executive,
Defense, Space & Special Systems Group Operating Units,
Burroughs Corporation, Paoli Area Manufacturing and Engineering, Paoli, Pennsylvania 19301, U.S.A.

Mr. L(loyd) Turner, General Manager, Large Computer Systems Division Operating Units, Burroughs Corporation, Large Systems Plant - Proctor, 14724 E. Proctor Avenue, City of Industry, California 91749, U.S.A.

Mr. R.N. Thompson, General Manager Large Computer Systems Division Operating Units, Burroughs Corporation, Large Systems Plant - Tredyffrin, Paoli, Pennsylvania 19301, U.S.A.

Mr. D(on) M. Lyle, Manager, Programming, Large Computer Systems Division Operating Units, Burroughs Corporation, Large Systems Plant - Proctor, 14724 E. Proctor Avenue, City of Industry, California 91749, U.S.A.

* will receive 1 copy.
Mr. F.B. MacKenzie, Manager, Programming Activity, Large Computer Systems Division Operating Units, Burroughs Corporation, Large Systems Plant - Tredyffrin, Paoli, Pennsylvania 19301.

Dr. R.S. Barton,
P.O. Box 844, Avalon, California 90704, U.S.A.

Mr. R.D. Merrell, Director of Engineering - Programming,
Burroughs Corporation, Burroughs Place, P.O. Box 418,
Detroit, Michigan 48232, U.S.A.

* Mr. C.R. Stark, President and Managing Director,
Burroughs International S.A., 18 Rue St. Pierre,
1700 Fribourg, Switzerland.

* Mr. E.R. Carter, General Manager, Latin American Area Headquarters,
Burroughs Corporation, Burroughs Place, P.O. Box 418, Detroit,
Michigan 48232, U.S.A.

* Mr. I.R. Rector, Managing Director,
Burroughs Machines Ltd., Heathrow House, Bath Road, Hounslow,
Cranford, Middlesex, England.

* Mr. J. Roy Henry, General Manager, Pacific & Canadian Area Headquart.
Burroughs Corporation, Burroughs Place, P.O. Box 418, Detroit,
Michigan 48232, U.S.A.

* Mr. R.W. Akers, Regional Manager - North Region,
Burroughs International S.A., 18 Rue St. Pierre, 1700 Fribourg,
Switzerland.

* Mr. J. Dumas, Regional Manager, South Region,
Burroughs International S.A., 18 Rue St. Pierre, 1700 Fribourg,
Switzerland.

* Mr. P.F. Smidt, President,
Burroughs Business Machines Ltd., 801 York Mills Road, Don Mills Ontario, Canada.
Mr. G. Prache, Group III Product Promotion Manager, 
Societe Anonyme Burroughs, 230-242 Avenue Laurent Cely, 
Boite Postale 50, 92231 Gennevilliers (Hauts de Seine), France.

Mr. J.J. Holton, Manager E.D.P. Products, 
Burroughs Corporation, International Group, Burroughs Place, 
P.O. Box 418, Detroit, Michigan 48232, U.S.A.

Mr. R.S. MacDermott, Manager Group III Products, 
Burroughs Corporation, International Group, Burroughs Place, 
P.O. Box 418, Detroit, Michigan 48232, U.S.A.

Mr. E.J. Nugent, Promotion Manager, Large Systems, 
Burroughs Corporation, International Group, Burroughs Place, 
P.O. Box 418, Detroit, Michigan 48232, U.S.A.

Mr. H.F. Hayde, Manager, Group III Products & Systems Support, 
Burroughs Corporation, International Group, Burroughs Place, 
P.O. Box 418, Detroit, Michigan 48232, U.S.A.

Mr. R.I. Meyer, Manager Group III Field Systems Support, 
Latin American Area, Burroughs Inter-America Ltd. Burroughs Place, 
P.O. Box 556A, Detroit, Michigan 48232, U.S.A.

Mr. J.A. Fischer, Promotion Manager, Group III Products, 
Pacific & Canadian Area, Burroughs Corporation, Burroughs Place, 
P.O. Box 418, Detroit, Michigan 48232, U.S.A.

Mr. S. Ferguson, Project Leader, 
Burroughs Limited, P.O. Box 30354, Lower Hutt, New Zealand.

* Mr. R.F. Werling, Product Manager, Series L. Products & Systems 
Director, E.D.P. Systems, Burroughs Corporation, Burroughs Place, 
P.O. Box 418, Detroit, Michigan 48232, U.S.A.

Mr. J.G. Cleary, Chief Software Engineering U.K. & Europe, 
Burroughs Machines Ltd., 512 Purley Way, Croydon, CRO-4NZ, 

Mr. R.W. Scholma, Manager Product Promotion, Group III, Burroughs B.V., 
Prof. E.M. Meijerslaan 2, P.O. Box 48, Amstelveen, Holland.