UCRL - 71365
PREPRINT

Lawrence Radiation Laboratory

UNIVERSITY OF CALIFORNIA

LIVERMORE

A BRIEF DISCUSSION OF GORDO

Gary B. Anderson

September 1968

This paper was prepared for presentation at
The Association for Computing Machinery
23rd National Conference, Las Vegas, August 27, 1968

A BRIEF DISCUSSION OF GORDO*

Gary B. Anderson

September, 1968

INTRODUCTION

This paper is a copy of the oral presentation of Design of a Time-Sharing system Allowing Interactive Graphics as delivered at the Association for Computing Machinery 23rd National Conference. For a more complete discussion, the reader is referred to the original paper[1].

GORDO is the name of the Lawrence Radiation Laboratories time-shared graphics project. It is named GORDO as a tribute to Gus Arrolias fine cartoon strip. It was from the antics of one of the GORDO characters with a computer controlled cartoon generator that we derived much of our early stimulus.

Since there is insufficient time for even a partial description of the entire system, I will discuss briefly its purpose, the hardware involved, and a few of the unique design features of the system and their attributes.

Purpose & Philosophy

GORDO is to provide interactive graphic service to up to eight users. Four of these will be located near the primary computer and the other four will be remotely located 2000 ft. away. Each terminal will consist of a CRT with keyboard, function buttons, and a light pen. Due to the nature of LRL and the programs and data to be processed by GORDO, the system is required

[1]Anderson G. B., Bertran K.R., Conn R.W., Malmquist K.O., Milstein & Tokubo S., Design of a Time-Sharing System Allowing Interactive Graphics, Proceedings of 23rd ACM National Conference, ACM Publication P-68, Brandon/Systems Press Inc., Princeton, N. J., 1968.

7096-9 [illegible] 7021994

*Work performed under the auspices of the U.S. Atomic Energy Commission.

to provide high rates of interaction for a small number of users and a high degree of security in handling user files and processes. In addition, we wished the executive to be relatively small and simple; therefore, whenever a choice existed between making a service part of the executive or part of a process, we chose the latter.

HARDWARE

The hardware portion of GORDO is composed of the following: (figure 1)

Main Computing Unit

A. An SDS Sigma 7 cup with:

1. Two extra high speed register blocks (16 registers).
2. Memory Mapping and access protection on a pagewise basis (512 word page).
3. Some additional but not necessary equipment
   - i. Power fail-safe
   - ii. Memory protect
   - iii. Real time clocks
   - iv. Floating point hardware

B. Two Interleaved Core Modules

16 K, 32 bits and parity, three port.

1. Port 1 is connected to the Σ7 CPU.
2. Port 2 is connected to a selector I/O processor which drives one set of graphics display equipment through a device subcontroller.
3. Port 3 is connected to a multiplexor I/O processor onto which are connected a number of I/O devices including a fixed head disc with the following attributes:
   - i. Size - 3/4 million words approximately 1365 five-hundred twelve word pages

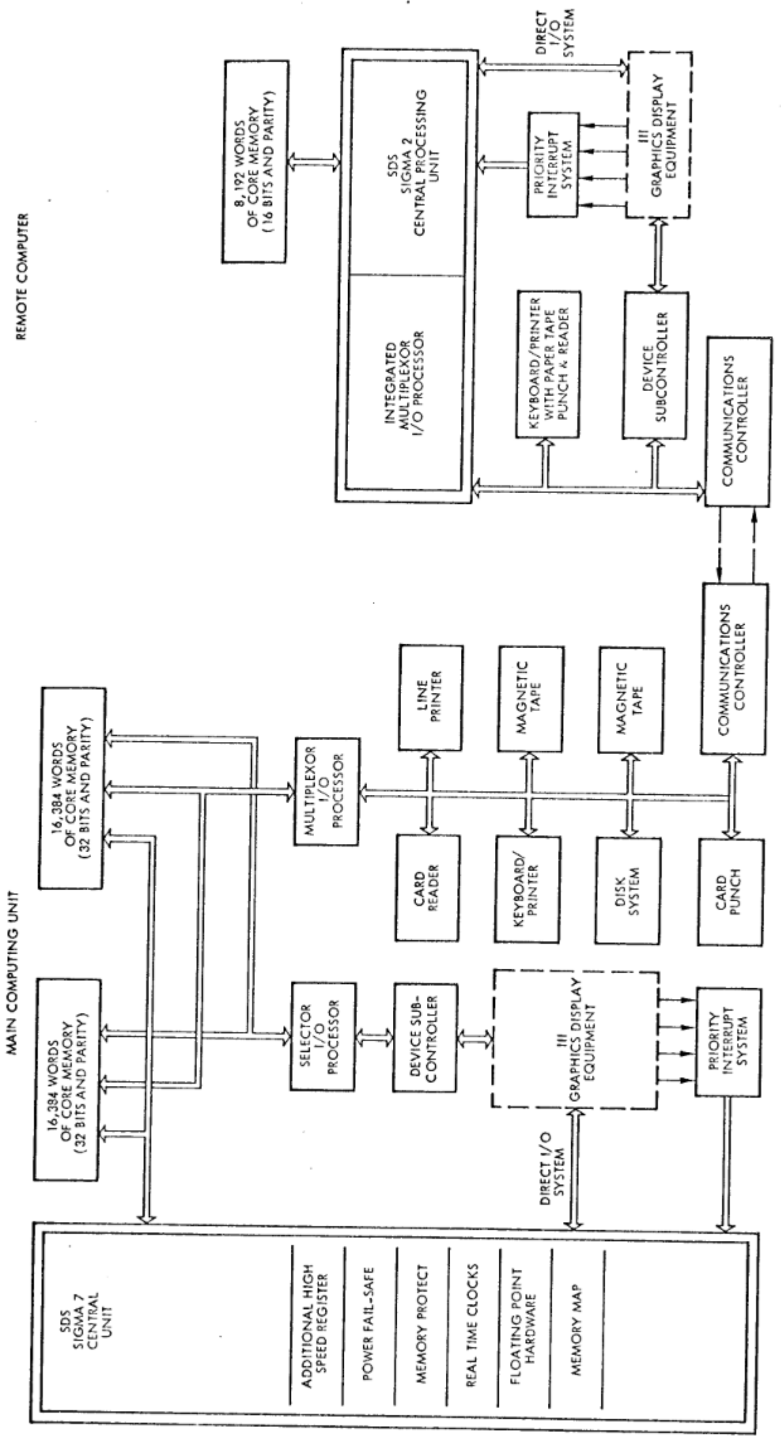

REMOTE COMPUTER
8,192 WORDS OF CORE MEMORY (16 BITS AND PARITY)
INTEGRATED MULTIPLEXOR I/O PROCESSOR
SDS SIGMA 2 CENTRAL PROCESSING UNIT
DIRECT I/O SYSTEM
PRIORITY INTERRUPT SYSTEM
III GRAPHICS DISPLAY EQUIPMENT
KEYBOARD/PRINTER WITH PAPER TAPE PUNCH & READER
DEVICE SUBCONTROLLER
COMMUNICATIONS CONTROLLER
MAIN COMPUTING UNIT
16,384 WORDS OF CORE MEMORY (32 BITS AND PARITY)
16,384 WORDS OF CORE MEMORY (32 BITS AND PARITY)
MULTIPLEXOR I/O PROCESSOR
LINE PRINTER
MAGNETIC TAPE
MAGNETIC TAPE
COMMUNICATIONS CONTROLLER
CARD READER
KEYBOARD/PRINTER
DISK SYSTEM
CARD PUNCH
SELECTOR I/O PROCESSOR
DEVICE SUB-CONTROLLER
III GRAPHICS DISPLAY EQUIPMENT
PRIORITY INTERRUPT SYSTEM
DIRECT I/O SYSTEM
SDS SIGMA 7 CENTRAL UNIT
ADDITIONAL HIGH SPEED REGISTER
POWER FAIL-SAFE
MEMORY PROTECT
REAL TIME CLOCKS
FLOATING POINT HARDWARE
MEMORY MAP

Figure 1

ii. Transfer rate - 42,500 words/sec approximately 12.7 ms/page

iii. Average access - 16.9 ms

C. An Information International Inc

display controller and from one to four display consoles which consist of:

1. CRT display

2. light pen

3. 128 character keyboard

4. function box - 16 buttons - 8 overlays

D. A Scientific Data Systems Inc

communciations channel connecting the Sigma 7 to a Sigma 2.

1. Transfer rate - 1 million bits/sec

Remote Computer

E. A Scientific Data Systems Sigma 2 with 8 k of 16 Bit Core Memory

F. A Second Display Controller with up to Four Display Consoles

Interesting Features

Initially we felt that GORDO would be a warmed-over version of the U.C. Berkeley GENIE System. But as the actual design progressed (with the help and consultation of the GENIE people) it moved further and further away from GENIE until it evolved into an entity of its own. During this evolution, some new approaches were born which I will now discuss.

User Processes & System Processes

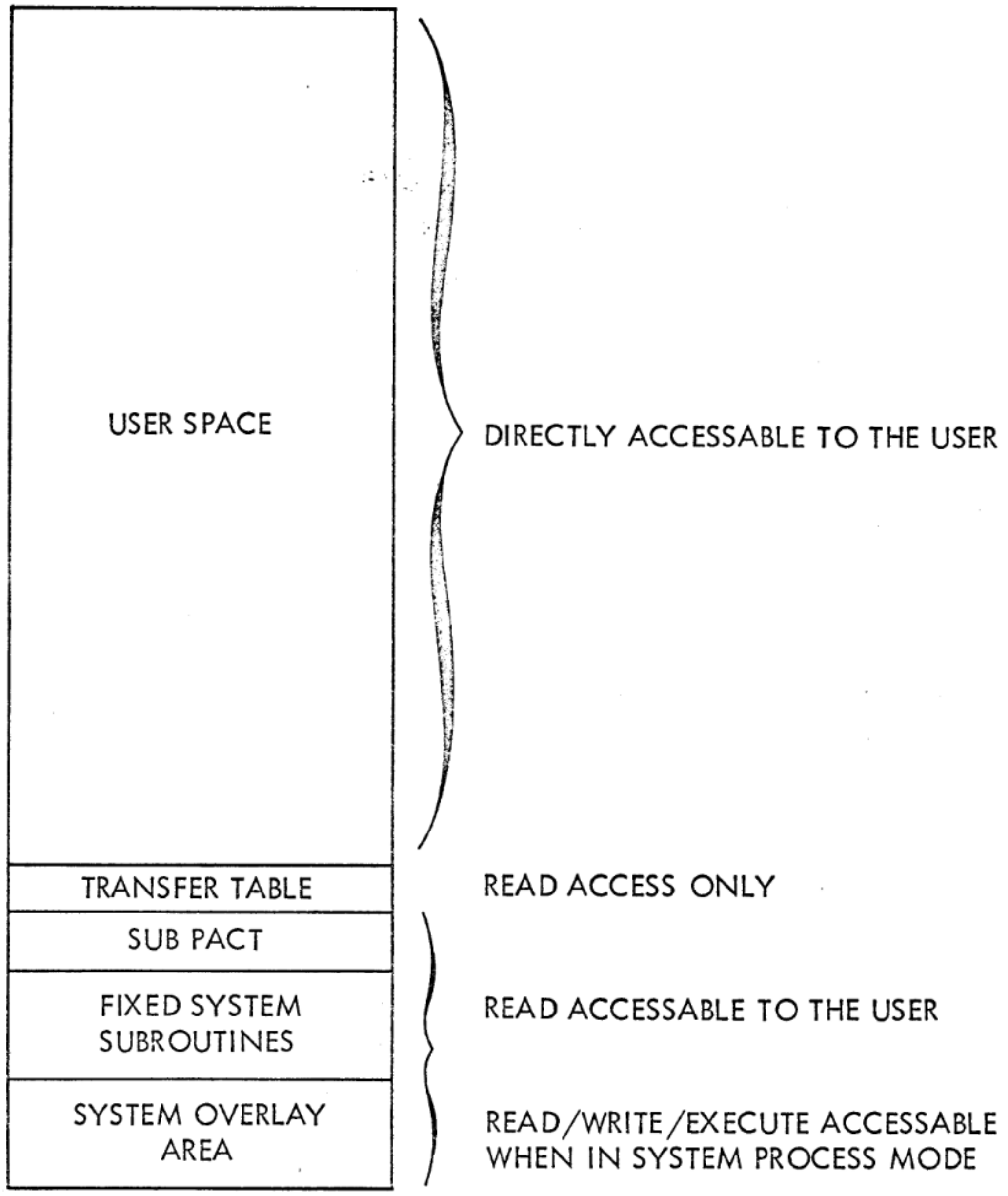
One of our early difficulties was providing the user with asynchronous system facilities. Requests for such services as a file manipulation which required external interaction with a slow speed device simply could not be handled by a resident executive without resorting to a number of kludges such as variable length queues. Our initial solution was to fork a new process (a system process) when necessary. This solution was not desirable because the operation of creating a new process is itself asynchronous and because it became difficult to pass arguments from the requesting user process to the new system process. At this point it was suggested that since normally a user process would suspend execution when it called for a system function it would seem reasonable to somehow combine user and system processes.

Our method of doing this (figure 2) is to divide the virtual space of the user into two parts. One, the user space (120 k words), is directly accessable to a user process. That is a user may modify the meaning of the space by means of a process called coupling, which we will discuss shortly. The other system process space (8 k words), is normally execute and write protected from the user. This space is composed of:

1. A read only transfer table which allows user controlled access to system process space.
2. A sub-process active table (PACT). This contains the processes state vector and a temporary stack to be used by reentrant system subroutines.
3. A collection of typically used system service subroutines. These will provide the mechanism for most of the user service requests.
4. A system overlay area where large or low use system service subroutines may be overlayed during use.

When a user wishes to use a system subroutine the following sequence takes place.

1. The user makes a normal subroutine call to a location in the transfer table associated with the system routine being called.
2. Since the transfer table has read only access an illegal access trap


USER SPACE
TRANSFER TABLE
SUB PACT
FIXED SYSTEM SUBROUTINES
SYSTEM OVERLAY AREA
DIRECTLY ACCESSABLE TO THE USER
READ ACCESS ONLY
READ ACCESSABLE TO THE USER
READ/WRITE/EXECUTE ACCESSABLE WHEN IN SYSTEM PROCESS MODE

VIRTUAL SPACE AS SEEN BY A USER

Figure 2

exit is made to the executive.

3. The executive determines that execute access of the transfer table was being requested by examining the address of the trapped instruction.
4. The executive sets the access of all of the space above the transfer table to full access and restarts the process at a transfer decoding routine in fixed systems space.
5. The transfer decoding routine determines if the user is trying to execute a legal location in the transfer table. If he is, then the corresponding subroutine is either executed or overlayed in and then executed. If an illegal location is specified then an error is returned to the user.
6. When the system process has finished a return is made through the executive to the user. During this return the access bits of the system space are reset to read only.

This scheme has the following virtues. Since no new process need be created, the time required to access system functions is quite small. System subroutines can be called using normal calling sequences. No special instructions have been preempted from the slave set.

This last point allows one to run slave coding generated for other Σ7 systems under GORDO.

Inclusion of State Vector in Virtual Space

Note the inclusion of the SUBPACT (STATE VECTOR) in the virtual space. This page contains:

1. The master copy of the process memory table (PMT) which maps the pages of virtual memory to the file pages which they represent.

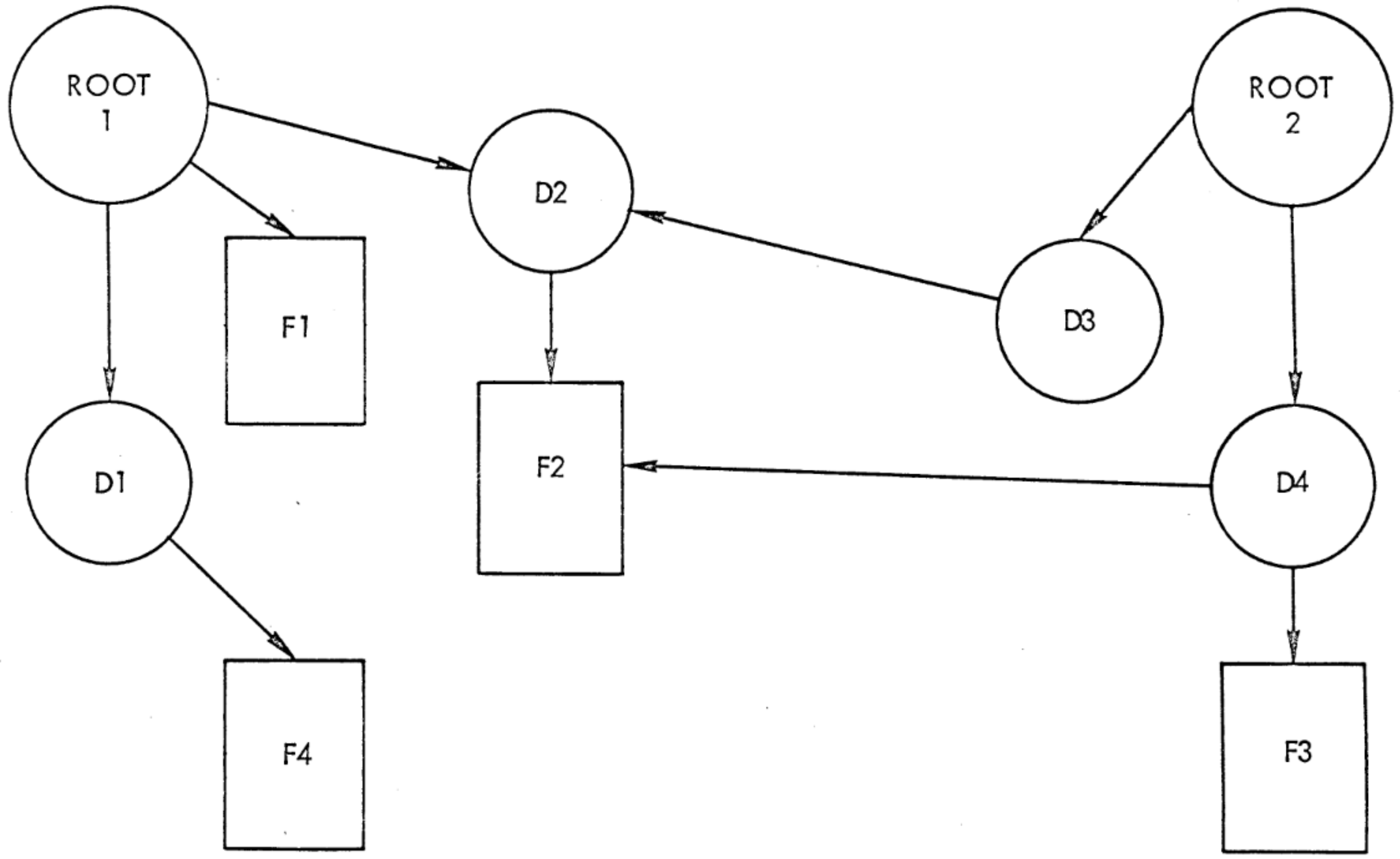
The PMT also contains page access information and sufficient information to identify the files to which the disc pages belong.

2. The program status double word. This contains the instruction counter, mode flags, trap flags and interrupt inhibit flags whenever the process is inactive.
3. High speed register storage when the process is inactive.
4. A number of keywords which contain information about the files and directories which are currently directly accessable by this user (OPEN).
5. A temporary stack to be used by the reentrant system subroutines.

The inclusion of the SUBPACT in virtual space allows portions of it to be manipulated by systems processes directly without resorting to the system executive. Further since the SUBPACT is a disc page and part of a file it is possible to give partial access to it to some other process. This allows for a very powerful method of interprocess control. Also since the SUBPACT must be in core memory while a process is running it provides a convenient place for executive/process communication and for system process temporary storage.

The File Structure

The file system, (figure 3) is composed of files and directories. A file consists of a heading and a number of pages which compose the body of the file. The file heading describes the attributes of the file and contains pointers to each of the pages in the body of the file. A directory consists of a number of named entries. These entries may point to either files or other directories. Files have no intrinsic names and may be accessed symbolically only through the directories. Each user has a root directory through which



DIRECTORY FILE STRUCTURE

Figure 3

he may access all of the files to which he is entitled. Files or directories may be shared with other users by giving them copies of appropriate entries through a common system directory which appears in everyone's root directory. This illustration (figure 3) shows the relationship of four files and directories to two users. Note that D2 and F2 are shared and that user 2 can reach F2 by two different paths.

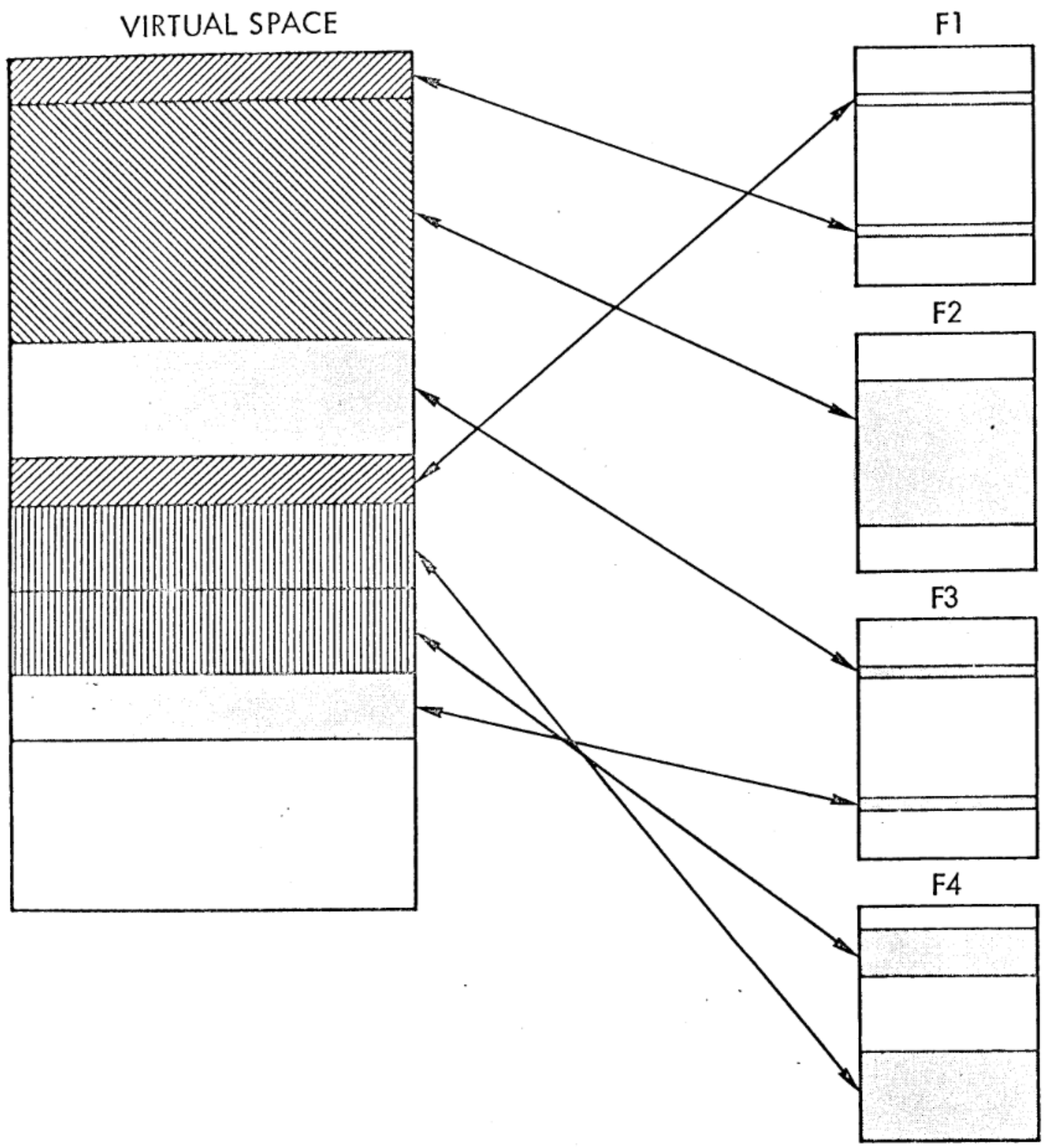
Relationship of Virtual Space and File Space

In GORDO a processes virtual space represents a collection of file pages taken from up to fifteen different files at one time. The mapping of virtual pages to file pages is given by the process memory table (PMT). In the PMT each virtual page is associated with either a disc address of a file page or with a null page. Note that the address space of file pages is much larger than that of virtual space, typically it is as large or larger than the back up store.

In figure 4 we see an example of the virtual space of a typical user process. As you can see pages which are contiguous in virtual need not be contiguous in file space and vice versa. There is no requirement that all of any file or all of virtual space be used.

GORDO provides a primitive operation to allow the user to dynamically change his PMT (virtual/file space mapping) at any time. This primative is called coupling. The effect of coupling is to provide an easy method of dynamic overlay.

In GORDO there is no direct equivalent of reading and writing secondary storage. Since there is no distinction between swappable information and data files the user simply couples the secondary storage he is interested in to his virtual space and uses it as part of his normal process. This has the further



EXAMPLE OF THE SPACE OF A TYPICAL USER PROCESS.
THE SPACE IS COMPOSED OF PARTS OF FOUR FILES.

Figure 4

advantage that if a secondary page is never actually accessed it will not be transfered into core memory.

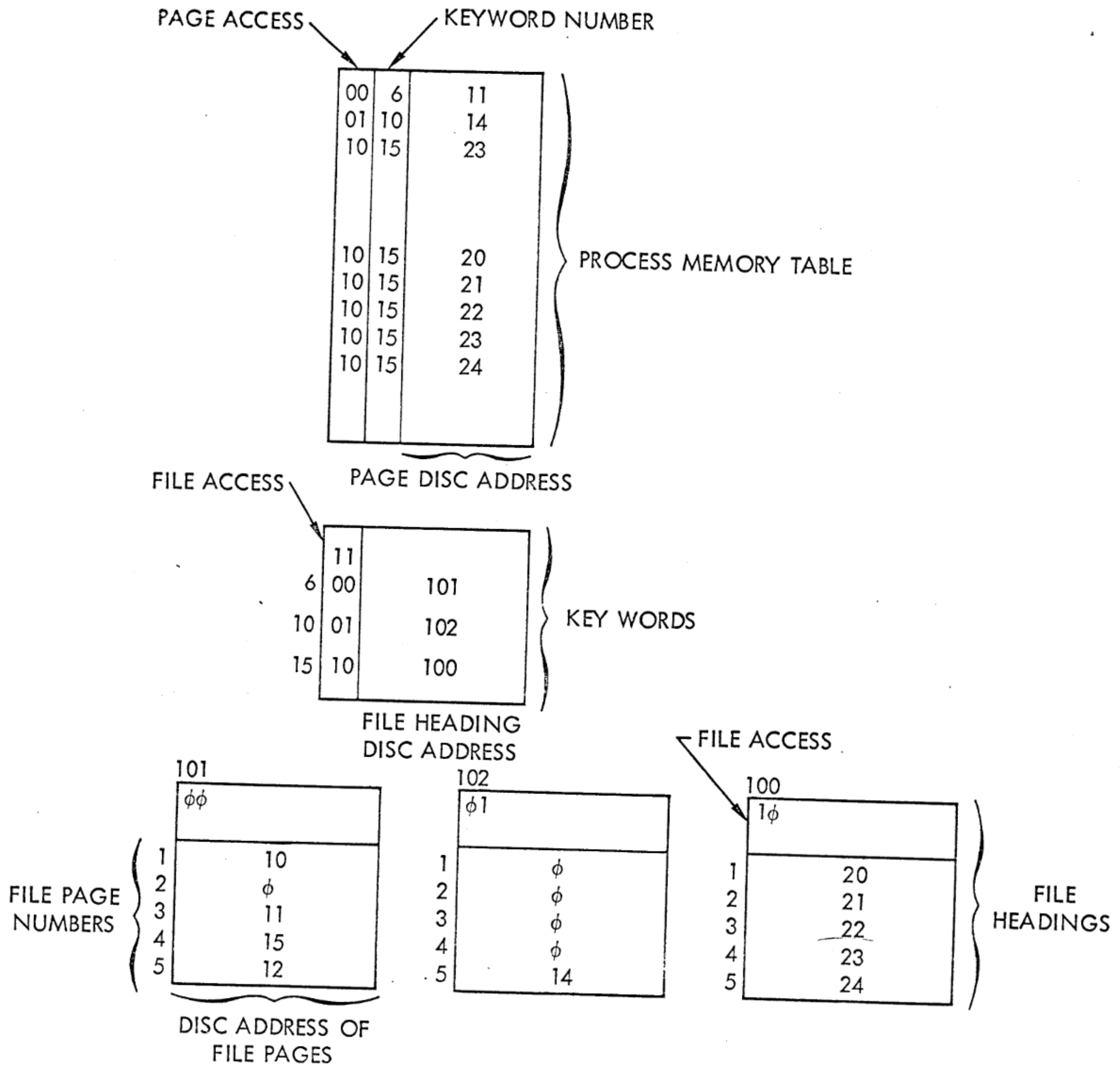
Figure 5 shows some of the details of the virtual/file space mapping. The PMT which is indexed by virtual page number contains the disc page address corresponding to the virtual page, the greatest allowable access to the page and the keyword number of the page. A keyword is a construct which allows the user to retain information about a file, which he has found by a symbolic search of his directories, until he has no further use for it. Finding this information and inserting it into a keyword is performed by an open primative. The inverse operation of destroying the information is performed by close. In this example three files have been opened. A keyword contains the disc address of the file heading represented by the keyword and this users current maximum allowable access to that file. The file headings shown contain some general attributes of each file (in this simplified case only the access is shown) and a list of the disc addresses of the pages which currently compose the file, indexed by the file page numbers of the file.

As an example of coupling suppose the user wishes to replace the virtual page which is associated with file 100 page 3 with page 5 of file 102. Couple would simply replace the virtual pages disc address with 14, its access with 01 and its keyword number with 10. Note that a file page may appear in more than one PMT entry at a given instant. When a keyword (file) is closed all entries with that keyword are zeroed out.

The net effect of coupling is that a user can easily and efficiently make any part of any file accessable to him part of his virtual space directly.

Association List

Since the process memory table relates virtual space to file page disc



RELATIONSHIP OF VIRTUAL SPACE TO THE KEYWORDS AND FILES

Figure 5

addresses there must be some mechanism which will relate disc page addresses to core pages. GORDO uses a software association list for this purpose. This list associates the disc address of copies of disc pages in core with their core locations. Thus when a processes pages are loaded the association list is searched to see if any of the appropriate disc pages are in core. Those that are not are transfered in from the disc and their presence is noted on the association list. There are several advantages to this association list method:

1. The current copy of a file page is used by a process no matter where it is located.
2. There is only one structure(the association list) that need be modified when a page changes location.
3. Keeping track of the location of pages shared by processes is completly automatic and requires no further mechanism.
4. All executive routines and system processes may refer to pages by means of their disc addresses. The only time that it is necessary to find their real location is when the hardware memory map is loaded during the first actual access by the running process.

The use of the association list eliminated a great amount of complexity and a number of real core tables from the executive.

Conclusion

GORDO was to be small and simple. The use of the techniques just discussed appears to have achieved this. In most of the cases where we expected problems to appear the above mechanisms were general enough so that new ones were not required. They have proved sufficiently general that we are still investigating new techniques for their use. Examples are user supervised interprocess control, simplified user overlay and the simulation of master made instructions by the user. All of these which have traditionally system functions now appear to be swappable user functions.

A word about system status and size. GORDO currently can sustain a number of users which have been hand placed in file space. The installation of our consoles is not complete so we have no method of interaction. The total system size excepting the display buffers and their drivers is approximately 3k of resident executive with another 3k of nonresident systems processes. We expect the display buffer size to vary from between 1-7k depending upon the number of users.

Finally the techniques presented are not necessarily restricted to a mapped machine. They may also be used in a contiguous environment so long as it is viewed as paged.