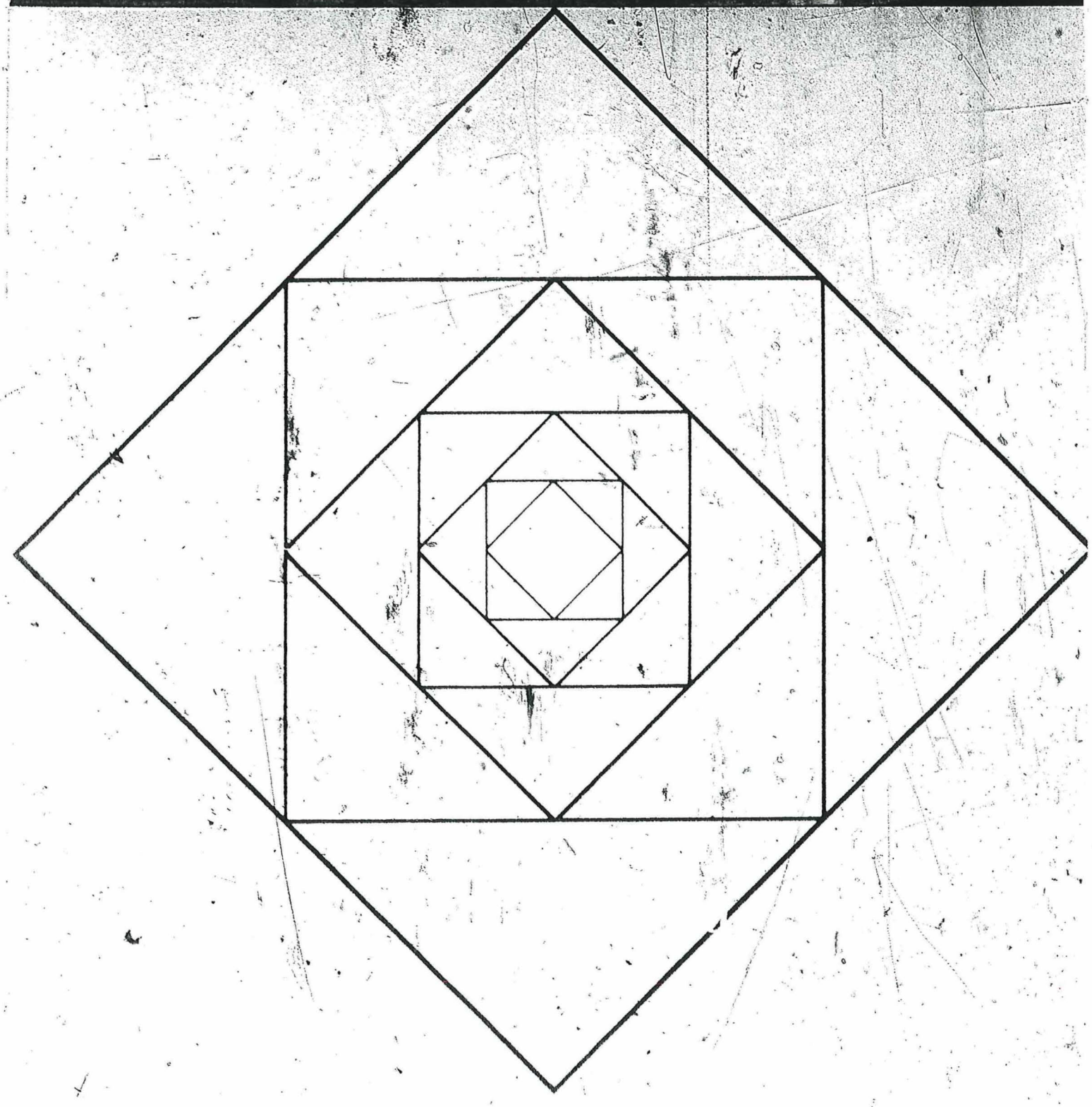
Cambridge Scientific Center

36.010
May 1966

IBM
Data Processing Division

A Virtual Machine System for the 360/40.



36.010

A Virtual Machine System for the 360/40

R. J. Adair
R. U. Bayles
L. W. Comeau
R. J. Creasy

IBM Cambridge Scientific Center Report

International Business Machines Corporation
Cambridge Scientific Center
Cambridge, Massachusetts

May, 1966

36.010
May, 1966
Scientific Center Report

A VIRTUAL MACHINE SYSTEM FOR THE 360/40

R. J. Adair
R. U. Bayles
L. W. Comeau
R. J. Creasy

International Business Machines Corporation
Cambridge Scientific Center
Cambridge, Massachusetts

Abstract

A virtual machine system, which provides copies of a 360 computing system for concurrent use by separate operating systems, has been implemented for the IBM 360 Model 40. The user at a terminal interface of a virtual 360 has all of the capability, with minor restrictions, provided by a stand-alone system. The system was designed as a system evaluation tool and as such, CPU efficiency or throughput improvement was not a primary design goal.

Index Terms for the IBM Subject Index

Computers
IBM 0360-40
Computing
Programming
Time-Sharing
07-Computers
21-Programming

TABLE OF CONTENTS

I. INTRODUCTION

Late in 1964, the IBM Scientific Center (formerly Systems Research and Development Center) at Cambridge, Massachusetts, undertook a project with a number of objectives. Among these were:

- the development of means for obtaining data on the operational characteristics of both system applications programs;
- the analysis of this data with a view toward more efficient machine structures and programming techniques, particularly for use in interactive systems;
- the provision of a multiple-console computer system for the Center's computing requirements;
- the investigation of the use of associative memories in the control of multi-user systems.

A system was designed which we thought would satisfy these goals and, in addition, provide other useful features. Efficiency in CPU utilization was not a primary design consideration.

Central to the idea of this system is the concept of the "virtual machine" and, in our case, the "virtual 360"[1]. Because of our desire to be able to measure a broad spectrum of programs, it is important that the imposition of a measuring system results in minimum alteration of the characteristics of the subject program. The "virtual 360" concept effects this minimum while providing the flexibility also required for the multi-user environment. In this system, the subject program interacts with the multi-user controller in the same manner as with the physical machine, and not by specially designed supervisor calls or subroutine calls as in currently implemented

multiprogramming packages. The program does not "see" the software interface between it and the physical hardware.

Within these virtual "360's" (called 360's), programs such as operating systems, which were initially designed to run on a hardware machine, may be run without change. In order to use the available facilities more efficiently, the Control Program supporting these multiple 360's performs the traditional multi-user tasks, such as scheduling, resource allocation, and core management.

We have created, therefore, a multi-user system where each user's virtual machine can run the programming system of his choice. None of these programming systems need consciously make use of multi-tasking facilities to improve machine utilization. Two other advantages accrue from this design - the ability to dynamically alter the virtual machine's configuration (core, size, available input-output units and paths), and the ability to assign more than one virtual 360 to a problem in order to examine the applications of multiprocessing.

We are providing sixteen virtual machines which may address 256K bytes of main storage, a maximum of one multiplexor and two selector channels, and a console typewriter. Some of the virtual machines may have additional typewriters, tape units, and a 2250 display console assigned to them. A user is normally supplied with three disks -- one

read-only disk is to secure for all users access to a library of often-used systems and routines, while providing the protection necessary in a multi-user system. The permanent disk provides continuing storage capability to the user. The temporary disk is available to a virtual machine for the duration of the session only. The user retains complete control over the format and use of his permanent and temporary disk space. Programs may be loaded into the user's virtual machine by name from the read-only disk, or by location from any device attached to the virtual machine.

II. HARDWARE IMPLEMENTATION

To provide these virtual machines, the Center obtained a modified System 360/40[(2)] with a multiplexor and two selector channels, interval timer, storage protection feature, universal instruction set, and 256K bytes of main storage. Its input-output equipment includes a console typewriter, line printer, card reader and punch, 2702 Transmission Control with remote terminals, four 2311 disk drives on two control units, two 2401-III tape drives, and a 2250 display unit with a 4K buffer.

The CPU has been modified to permit dynamic relocation of storage addresses by the addition of a 64 word (one per 4096 byte page of core memory) by 16 bit associative memory (Figure 1). A privileged operation to load and interrogate the memory has been added to the instruction set. (see Figure 1.)

III. PROBLEM MODE OPERATION OF THE ASSOCIATIVE MEMORY

When the CPU is in problem mode, each main storage address presented to memory is mapped by the following method (see Figure 2):

the high order six bits of the eighteen bit memory address plus the user identification number (set by the Control Program) are presented to the associative memory for a match;

if a single match is found, the address of the selected row of the memory replaces the high order six bits on the memory bus, and the memory select takes place;

if a multiple match condition (an error which should never occur) or no match (requested page is not in the memory) occurs, an interruption is generated and the Control Program must take the appropriate steps to resume execution.

This mapping takes place with no degradation of Model 40 cycle time.

Six bits of the associative word are provided to assist the scheduling section of the Control Program in selecting the least costly (i.e. least likely to be brought back) page to roll out when additional main memory space is required.

The used bit is set when a match condition is found for a row of the memory, indicating a reference to the corresponding page, and is reset when all of the pages represented by

the entries in the memory have been either referenced or are locked;

the active bit is set at the same time the used bit is set, but is not, like the used bit, automatically reset;

the changed bit is set when the instruction causing the match condition could result in alteration of the contents of the corresponding page;

the lock bit is set by the control program and is interpreted by it to mean that the page may not be removed from memory;

the transit bit is used by the control program to indicate that the page is currently being brought in or dumped out;

one spare bit is provided for unspecified use.

IV. CONTROL PROGRAM STRUCTURE

When the Control Program code is being executed, the machine is in the supervisor state; at all other times it is in the problem state. Any action of a virtual machine which could cause a change of machine state results in an interrupt. The Control Program, then, is an interrupt driven system whose components reference a set of tables describing the state of a users' virtual machines. For each user, this table (UTABLE) contains a copy of the current Program Status Word (PSW) and the user's general purpose and floating-point registers, the locations of the user's

virtual memory pages (which are either core or disk resident), a description of the input-output equipment and its status, a copy of the user's interrupt region, and other similar information.

There are two basic types of interrupts handled by the Control Program: 1) those which invoke a section of the Control Program to perform some function for the virtual machine, and 2) those which require no special action by the Control Program and are "reflected" to the virtual machine (such as supervisor call and most program interrupts caused by overflow conditions, protection violations, addressing errors, etc.). The reflection of interrupts to the virtual machine is performed by the appropriate swapping of PSW's in UTABLE and setting the proper interruption codes there.

If a virtual machine's current PSW contains the wait bit or if its execution has been delayed due to the temporary unavailability of a necessary resource, it is considered not runnable. At the occurrence of an interrupt which could affect the runnability of a machine, UTABLE is examined for interrupts pending which are enabled. If an enabled pending interrupt is found, the appropriate "reflection" of the interrupt takes place (moving of the current PSW to one of the old PSW's, and of the corresponding new PSW to the current PSW).

A privileged operation interruption (caused by an attempted execution of a privileged operation while in problem mode) results in one of two actions, depending on whether the virtual machine was in

problem or supervisor mode. If the virtual machine was in problem mode, the interrupt is merely reflected to the virtual machine. If the virtual machine was in supervisor mode, the action of the privileged operation, with the exception of input-output operations which are handled separately and discussed in a later section, must be simulated by the Control Program by appropriate changes in UTABLE.

A timer-initiated external interrupt controls CPU scheduling among the virtual machines. Each machine is allotted a quantum of time to run (which may be sliced into smaller intervals for timer simulation purposes) and, at the completion of that interval, a round-robin scan is made of the users to ascertain if another virtual machine is runnable.

An interrupt from the associative memory, caused by an attempted reference to a page not core resident, invokes the core management and scheduling routines. The missing page is indicated by the interruption code and the paging routines must schedule a page to be rolled out (according to algorithms which will be a chief point of study), and the appropriate page retrieved from disk. For the duration of the "page turning", the virtual machine is placed in not runnable condition.

V. INPUT-OUTPUT OPERATIONS

The input-output equipment generally falls into two classes: high data rate devices on the selector channel, and low data rate devices on the multiplexor channel. These characteristics, together with the need to share unit record facilities, the expected programming mode of the

typewriter devices, and the differences in the logical structure of the channels, make it desirable to handle selector channel and multiplexor channel input-output operations separately.

Much of the selector channel input-output is not conveniently interruptable; therefore the channel programs are prescanned and all referenced pages are brought into core and held there for the duration of the operation. During this scan, virtual data addresses are converted to real core addresses, eliminating the need for translation hardware associated with the channels. Similarly, direct access storage addresses (bin, cylinder, and head numbers) spec d within the channel program are modified to provide partitioning of these devices, thus sharing the units among the virtual machines. The modified copies of the channel programs thus produced are used to directly control the selector channel devices. Under this scheme much of the validity checking and interruption sequencing can be performed by the hardware. Tables are provided to map device addresses, detect path conflicts, and stack interruptions for the virtual machines. An input-output scheduler provides request queueing and facilities scheduling at the hardware interface.

The sub-channel programs for the shared unit record equipment (punch, reader, and printer) on the multiplexor channel are run interpretively. All data for these devices are buffered in core and on disk, thus operating in a spool-like mode. All interrupts must be software simulated.

Since multiplexor subchannels servicing typewriter devices are expected to spend most of the time in a read state awaiting input, buffers are provided to reduce the core tied up by these operations from one or two pages to a few hundred bytes. The I/O status at these devices is controlled by the subchannel program; no information is read before being requested by the virtual machine. A mapping program is provided to simulate the on-line typewriter with remote typewriters, when desired.

By depressing the BREAK button (a special feature of our modified 1050 remote terminals which roughly corresponds to the Attention feature of the online 1052 and 2741 remote terminals), the user may break out of his virtual machine and enter conversation activity with the Console Function routines, which provide the simulation of the following hardware console functions:

Address Stop

System Reset

Start

Stop

Load

External Interrupt

Display

Store

VI. LIMITATIONS

Taking the above concepts into consideration, the following limitations were accepted in the initial implementation of the system:

-the result of dynamic alteration of channel control programs, while they are in execution, is generally unpredictable;

-correct operation of input-output timing dependent programs may not be assumed;

-no input-output operation which requires more core than is available for virtual machine page residence is allowed;

-the interval timer will accurately reflect only CPU execution time.

References

(1) "On Virtual Systems", D. Sayre, IBM Watson Research Center

(2) "A Time-Sharing System Using an Associative Memory", A. B. Lindquist and R. R. Seeber, IBM Systems Development Laboratory. Unpublished paper submitted for December, 1966, issue of "Proceedings of the IEEE".

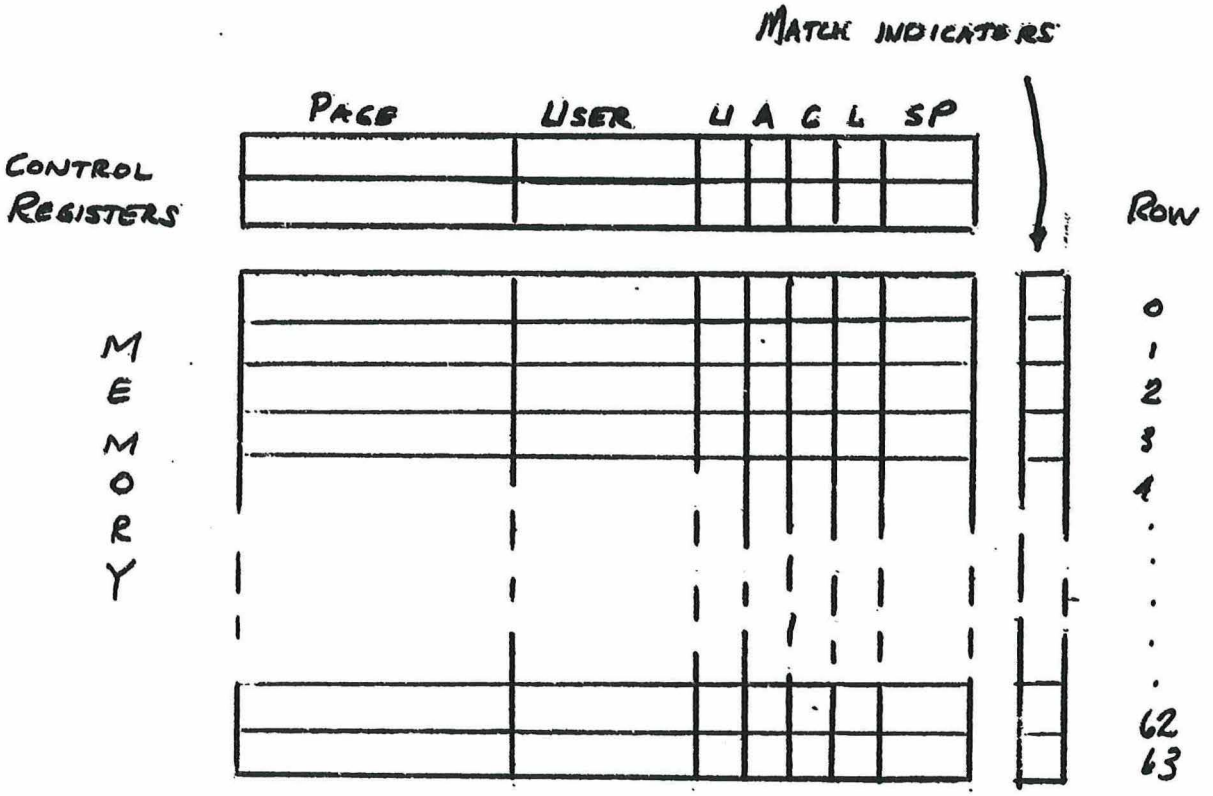


FIGURE 1.

PRELIMINARY SKETCH

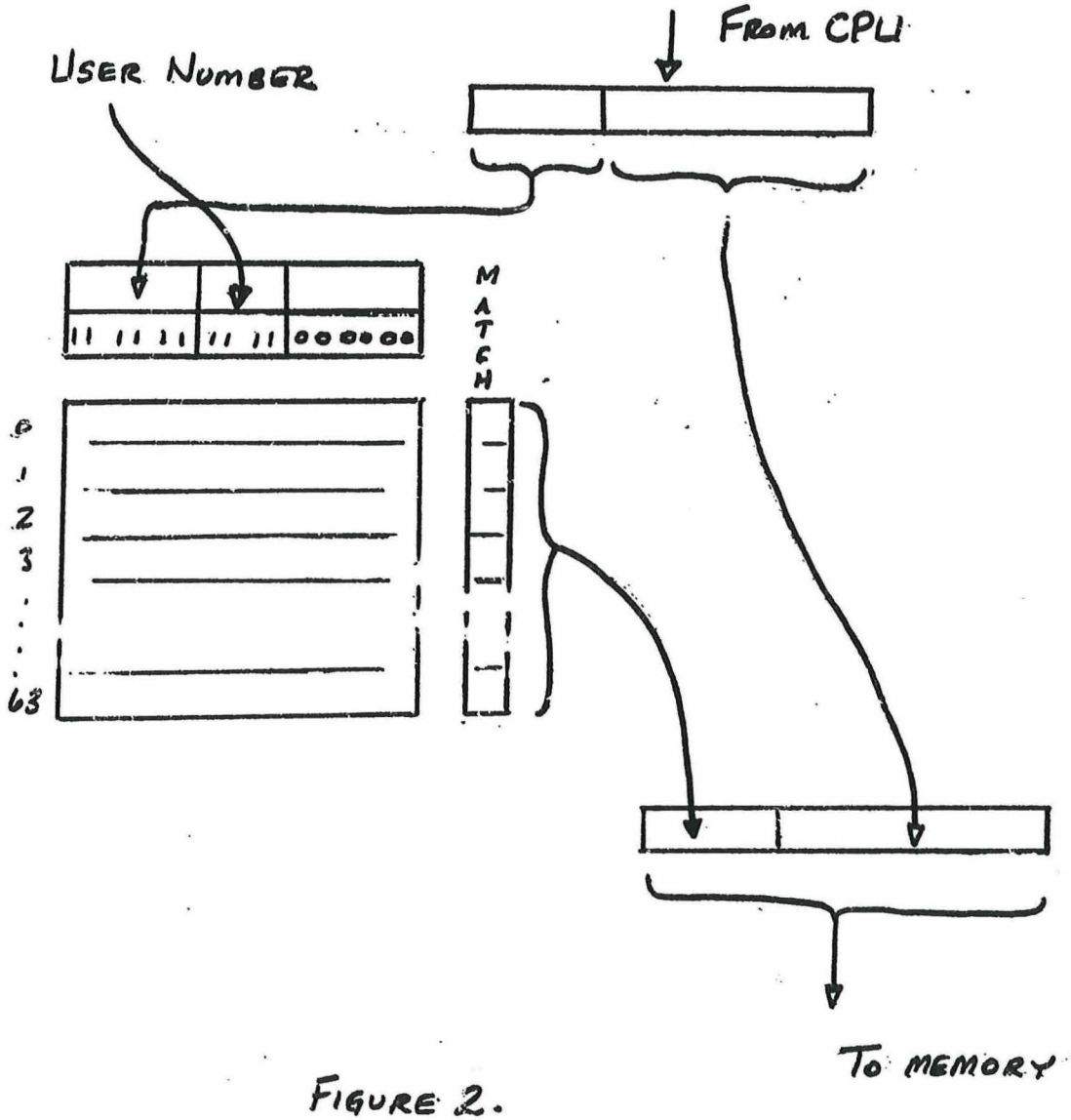


FIGURE 2.

PRELIMINARY SKETCH