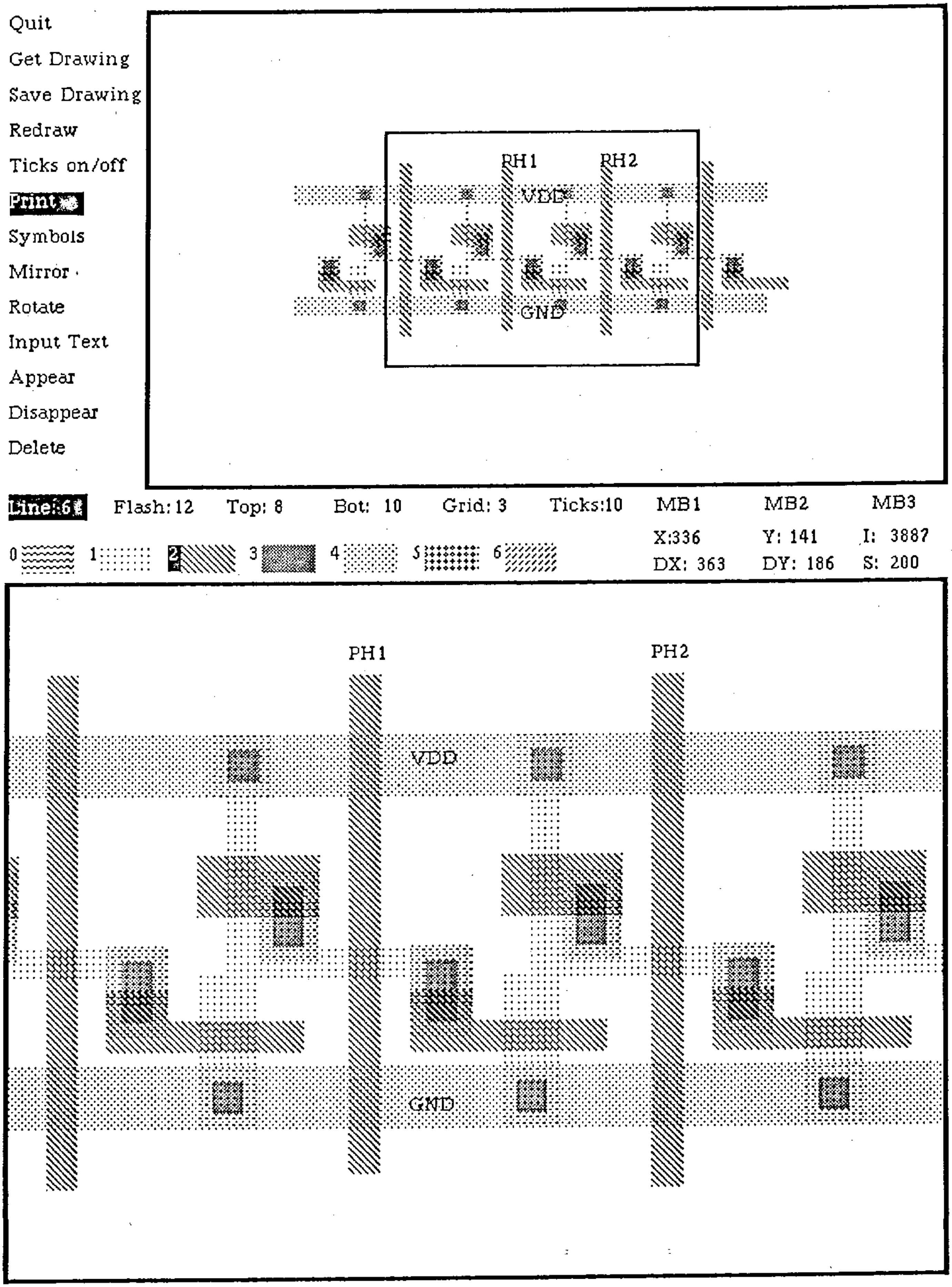

Fig. 17. The ICARUS Display: 2 Views of a Layout in Progress (windows.press)

An Interactive Layout System

[Section contributed by Douglas Fairbairn, Xerox PARC, and James Rowson, Caltech]

Computing hardware of sufficient power to support highly interactive graphics has in the past quite expensive, and this has inhibited the widespread application of interactive computing techniques. However, because of expected advances in VLSI technology, we are rapidly approaching the day when many will have access to personal computers with computing power rivaling today's medium to large-scale systems. It will be more difficult to provide effective software for these systems than it will be to build the computers themselves.[3] In this section we describe a highly interactive layout system which runs on a modest personal computer, rather than on an expensive, limited access, centralized system. This system was developed anticipating the work environment of the future, in which most "knowledge" workers will have personal computers as part of their normal office equipment.

ICARUS[1] (Integrated Circuit ARtwork Utility System) is a software system which enables the user to create and modify an integrated system layout directly on a CRT display screen. ICARUS was conceived with the idea that the designer would create and edit a layout at the display, without doing any more than a rough sketch or "stick diagram" before beginning work. Creating and moving items is fast and easy enough so that the designer can truly sketch on the screen. Once the layout is basically correct, the items can be moved or modified to arrive at the most compact layout.

The user is required to remember very little about the available commands or their use because the commands themselves are displayed on the screen and the system prompts the user for additional information as it is needed. The system can format and output check plots to matrix type printers or on raster-scan laser printers. ICARUS design files can be used to create standard pattern generation files from which masks can be made. An overview of design and layout procedures using the system is given in figure 16. It is instructive to compare this with figure 15, which presents equivalent steps for hand layout.

All the software to accomplish these various steps runs on a small experimental minicomputer known as the Alto. This machine was designed by researchers at Xerox PARC as a general purpose personal computer suitable for both text and graphics applications[3]. No additional, special hardware is used by ICARUS. The ICARUS system is programmed in BCPL, an ALGOL-like high level language. There are about 30K words of

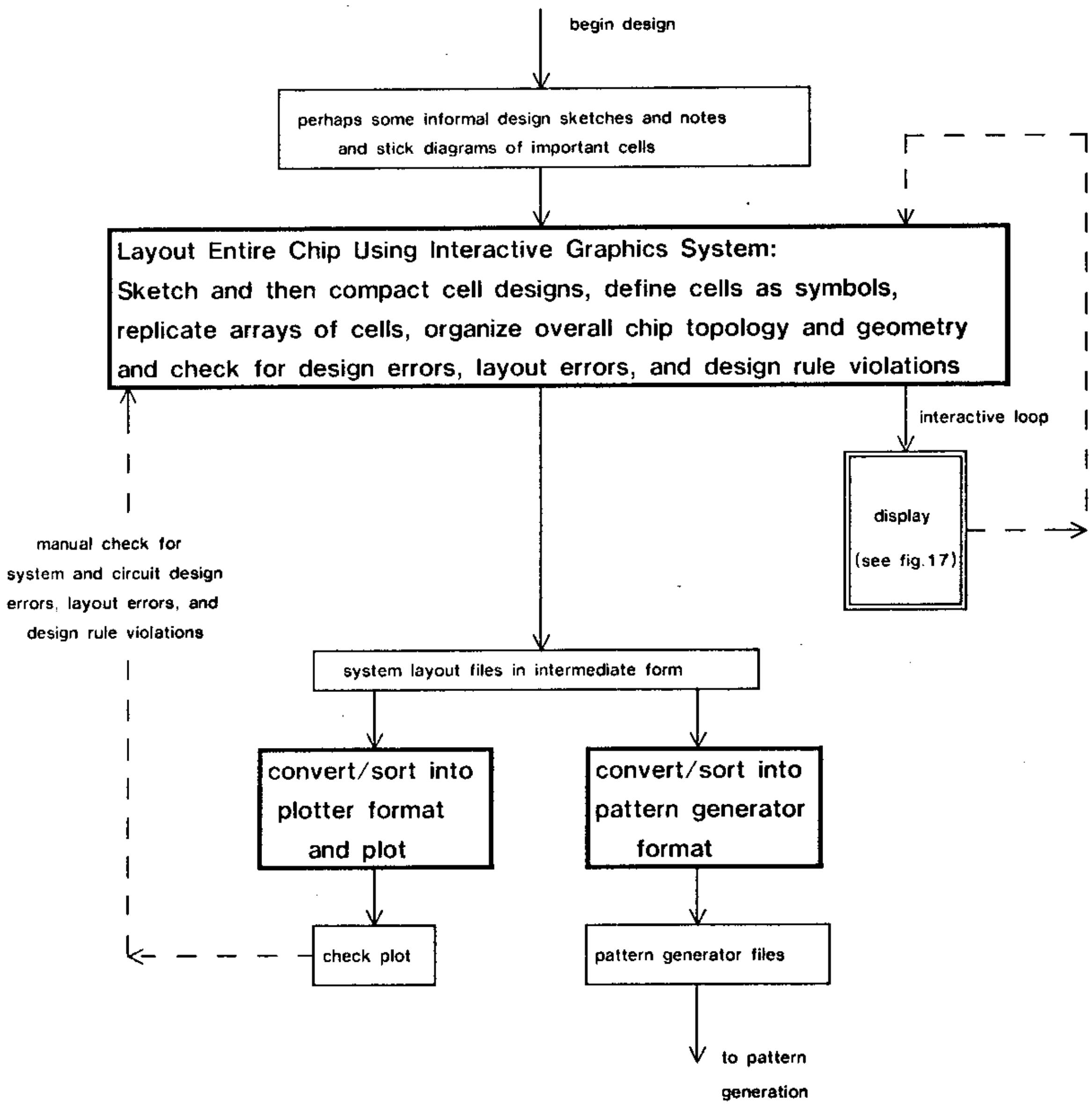

Fig. 16. Design, Layout, Design File Generation, and Design Checking Using an Interactive Graphics Layout System

compiled code in the system of which half is in memory at any given time. At minimum, the Alto memory has 64K 16 bit words. A 2.5 Mbyte cartridge disk drive is an integral part of the system. The user interacts with the system through an unencoded keyboard (software definable keys) and with a pointing device called a mouse (R2, p173). A cursor is controlled on the screen by moving the mouse around on a small area of the user's desk. A bit map display with a resolution of 600x800 dots is used for output, and printers for doing check plots are available through an in-house computer network.

The ICARUS display features two windows which provide a flexible working view of the layout, as shown in figure 17. The upper window is normally used for viewing a large piece of the layout at small magnification, and the lower window used for looking at a smaller section in more detail. The magnifications of the windows may be set independently.

In addition to the two windows there are various menus and status lines presented in the display. The menu on the left is the *command menu.* The menu under the upper window is the *parameter menu.* Under the parameter menu is the *stipple menu*, containing the mask level codes. Rectangles at a given level are stippled with the pattern for that level. The patterns were chosen so that, where necessary, one pattern could be seen through the other to verify that appropriate layers are overlapping properly. Current drawing coordinates and the status of system memory space are displayed to the right of the stipple menu.

The user interface is implemented principally through the display, the mouse and five conveniently located keys on the keyboard. Frequently used commands are given using only one or two simple hand operations, and can be done without glancing away from the display. These characteristics, coupled with rapid display redrawing, enhance the system's interactiveness.

The internal data representation in ICARUS is based on three types of items: rectangles, symbols, and text strings. The organization of these items into memory data structures, and the typical run-time memory space allocation is illustrated in figure 18.

Rectangles are created with the aid of the mouse. They may have angular orientations which are integer multiples of 45°. They can be moved, copied, or deleted using the mouse and one key. As items are created, they are added to an item list in main memory. Each rectangle is stored as 6 words in memory: the first word is the pointer to the next item, the second specifies what layer it is on, what type of item it is, etc. The third through sixth

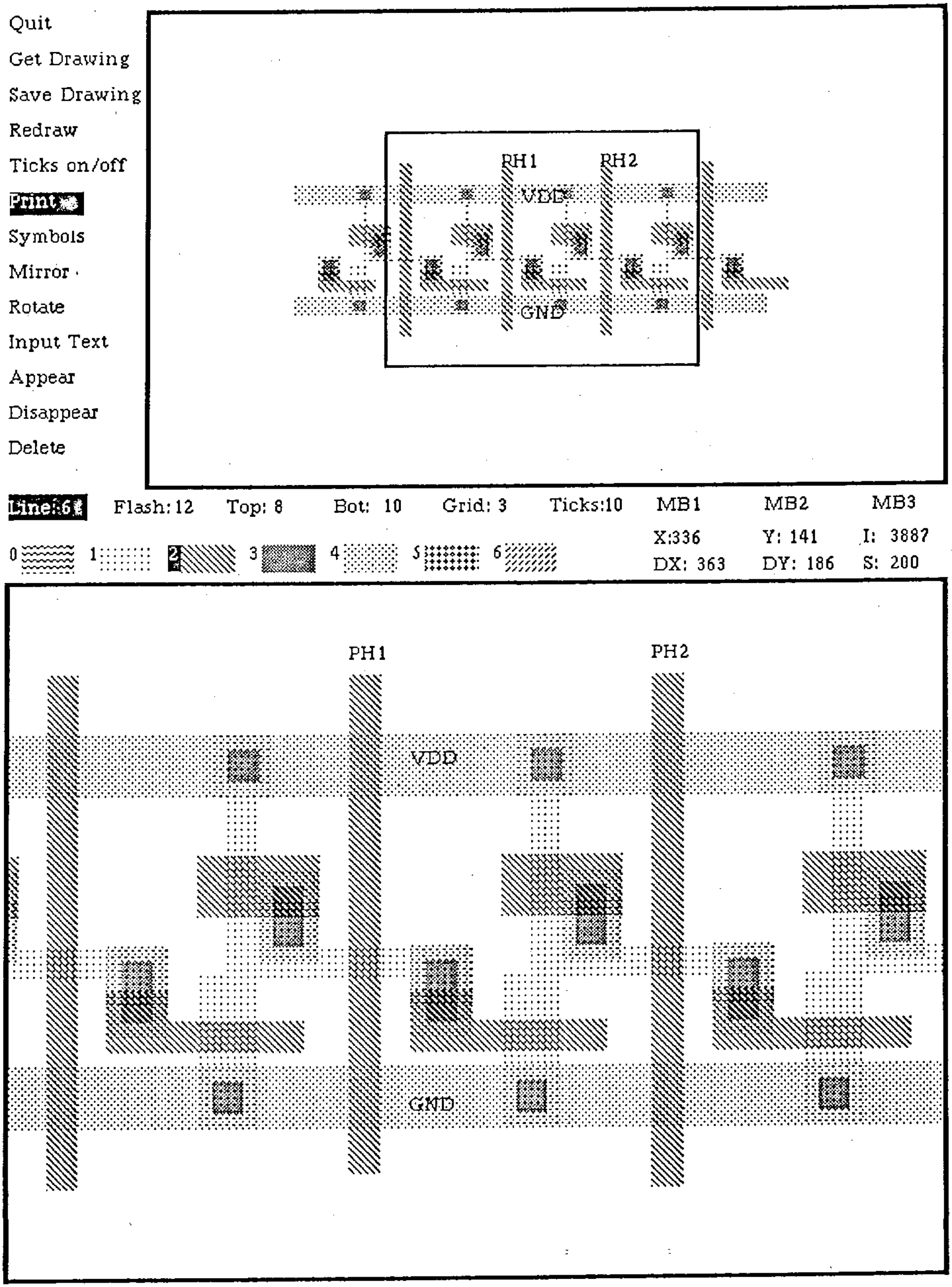

Fig. 17. The ICARUS Display: 2 Views of a Layout in Progress (windows.press)

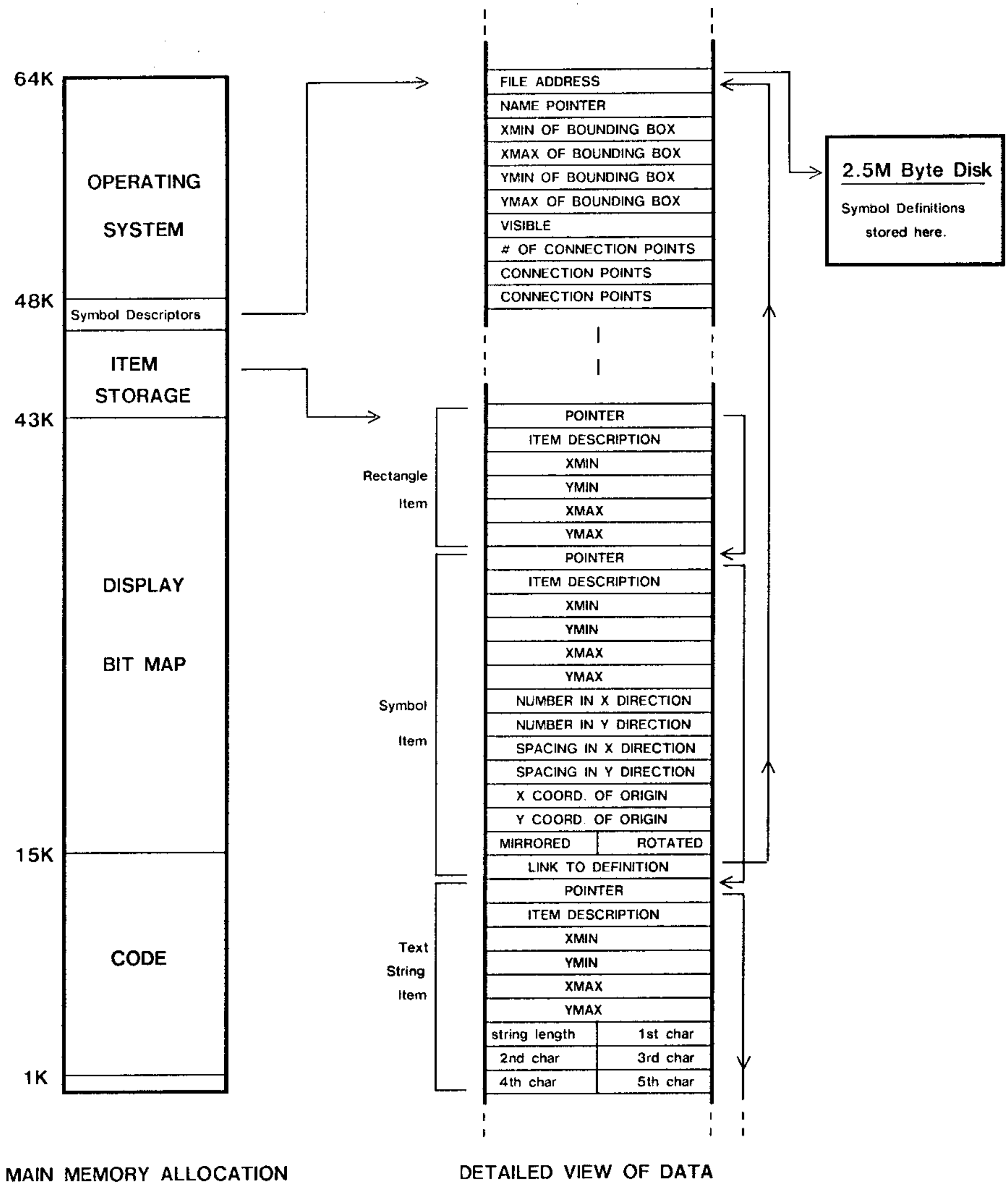

Figure 18. ICARUS Memory Allocation and Data Structure

(memory.sil)

words specify the minimum and maximum x and y coordinates. The items are kept in order of increasing values of minimum x coordinate, so that the display may be quickly redrawn.

When a symbol is defined by the user, the items which are contained within it are stored on the disk, while a pointer, the name and the bounding box for the symbol are placed in main memory. Symbols can be nested to any level. Once a *symbol definition* has been created, one is free to define *symbol instances* which are references to that definition. The symbol instance may be a command to draw one copy of the symbol at a certain location, or a whole array. The size of the symbol instance, which resides in main memory, is the same in both cases. The use of symbols wherever possible tends to preserve main memory space. Rather large systems can be designed using ICARUS, if the systems are well structured and make extensive use of symbols. This is true even when using a minimum sized 64K memory, which leaves little space for layout data.

Text is used for identifying data and control lines and is merely a memory aid to the user. There is no attempt to make use of the text or other information in the drawing for connectivity or other types of checking.

Operations more complex than those such as draw and move are implemented through the use of menus as shown in figure 17. The desired command is chosen by pointing at it with the cursor and clicking a mouse button. The selected command is then inverted to white on black video to identify its selection, which the user then confirms with a key on the keyboard. At this point, the system prompts the user with instructions presented in the display area normally holding the stipple menu. The instructions lead the user through the individual steps required, for example, to mirror or rotate a group of items.

Operations on symbols are defined in a secondary menu which can be reached by selecting the command "symbols" on the primary menu. The secondary menu offers commands such as define symbol, draw symbol, list the names of the symbols is the symbol library, or expand symbol. This last command is used to modify a symbol which is already defined, the modified symbol definition immediately updating all symbol instances which point to it.

Various system parameters are displayed in the parameter line directly below the top window. Values such as the default line width for the currently selected layer, the magnification of the top and bottom windows, and the spacing of the tick marks are all displayed. The parameter values can be changed at any time by selecting the desired one and typing the new parameter value on the keyboard. The X,Y layout coordinates of the point last clicked with the mouse are displayed at the right of the screen. The DX,DY distances between the last two clicks are also displayed. This feature provides a convenient "ruler" for measuring distances on the layout.

The construction of an interactive layout system such as ICARUS is a relatively straightforward task for one who is experienced in interactive computer graphics (R2), given a display oriented minicomputer system and effective systems building software. A first version of ICARUS was constructed in 3 man-months, and a mature version produced in an additional 4 man-months.

ICARUS has been used internally in Xerox to lay out many integrated system projects, and to organize a number of multi project chips. Among the users were a number of individuals previously unfamiliar with integrated circuit layout, who nevertheless successfully completed LSI projects with up to 10,000 transistors. We find that the interactive nature of such a system not only aids the experienced designer, but also enhances the learning process for the novice. We believe that such interactive, personal design systems greatly enhance the creative ability of the designer by enabling easy generation and examination of many more design alternatives per unit time than would be the case with centralized, non-interactive design systems.

However, there is more to integrated system design than circuit layout. Design rules must be checked, logic transfer functions tested, and, in certain cases, circuit transfer functions computed to determine delays and predict system performance. We believe that the direction in which to search for further improvements in design tools is in the replacement of the primitive ICARUS type of data structure with one which allows design functions other than just layout to also interactively operate upon the same data base. This is the subject of the later section on fully integrated design systems.