

# System Programming

## Second Class

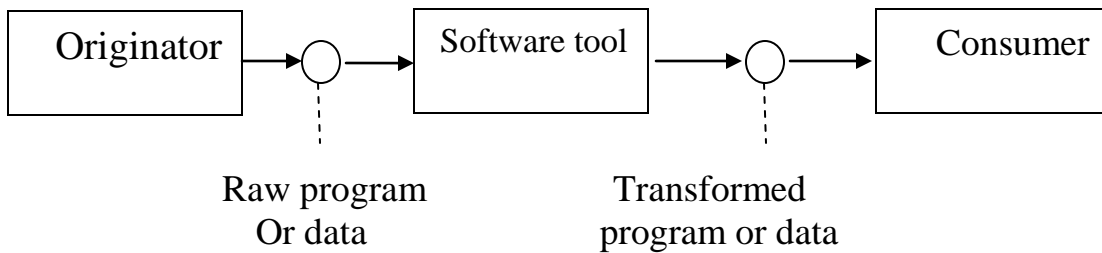
مدرس مساعد: منال مطلوب

### CHAPTER SEVEN (System Software Tools)

#### Introduction:

A software tool is a system program which

- a) Interfaces a program with the entity generating its input data or
- b) Interfaces the result of a program with the entity consuming them. Figure below shows the schematic of a software tool.



(Software tool)

#### Software Tools for Program Development:

Step in the program developments are

- 1) Program design, coding and documentation
- 2) Preparation of program in machine readable form.
- 3) Program translation linking and loading
- 4) Program testing debugging
- 5) performance tuning
- 6) Reformatting the data or results of program to suit other programs.

#### 1. Program Design and Coding

Tools used for program design and coding are

- a) Program generators
- b) Programming environments.

- A program generator generates a program which performs a set of functions described in its specification.
- A programming environment supports program coding by incorporating awareness of the programming language syntax and semantics in the language editor.

## **2. Program Entry and Editing**

- In this, text editors are used as front ends tool. The editor operates in two mode:

### **Command mode and data mode**

- In the command mode, it accepts user commands specifying the editing function to be performed. In the data mode, the user keys in the text to added to the file.

## **3. Program Testing and Debugging**

- The important parameters are the selection of test data for the program analysis of test result to detect errors and debugging.
- For program testing, the selection of test data for program and analysis of test results to detect error if any is important.
- User uses the software tools for this purpose.
  - 1) Test data generators.
  - 2) Automated test drivers.
  - 3) Debug monitors.
  - 4) Source code control.
- Test data generators help the programmer in selecting test data.
- Regression testing is done through automated test drives. in this, program correctness is verified by subjecting it to a standard set of tests after every modification.
- Obtaining the information about localization of error, debug monitor is used.
- Source code control system help to keep track of modification in the source code.

## **4. Enhancement of Program Performance**

- Two factors affects the program efficiency.
 

These are:

  - a)Efficiency of the algorithm
  - b)Efficiency of its coding
- An optimizing compiler can improve efficiency of the code but it cannot improve efficiency of the algorithm. Only a program designer can improve efficiency of an algorithm by rewriting it.

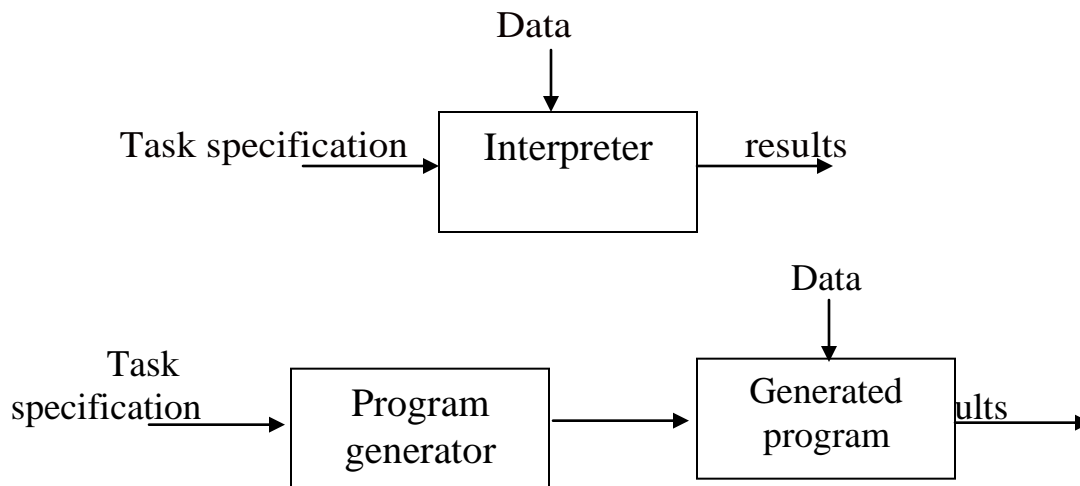
## 5. Program Documentation

Most programming projects suffer from lack of up to date documentation~ Automatic documentation tools are motivated by the desire to overcome this deficiency.

- These tools work on the source program to produce different forms of documentation e.g. flow charts

## 6. Design for Software Tools

- Program preprocessing techniques are used to support static analysis of programs. Program instrumentation implies insertion of statements in a program. The instrumented program is translated using a standard translator.
- Figure below shows the schematic of software tools using the techniques of interpretation and program generation.
- Interpreter based tools suffer from poor efficiency and poor portability, since an interpreter base tool is only as portable as the interpreter it uses. A generate program is more efficient and can be made portable.

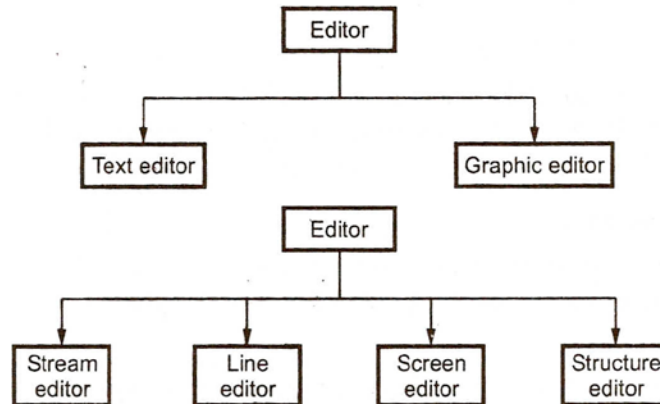


(Software tools)

## Editors:

For any computing environment interactive text editors has an important part.

- Various types of editors are:



(Types of editors)

- Text editor should be the primary interface to the computer for all types of knowledge workers.

Overview of the editing process'

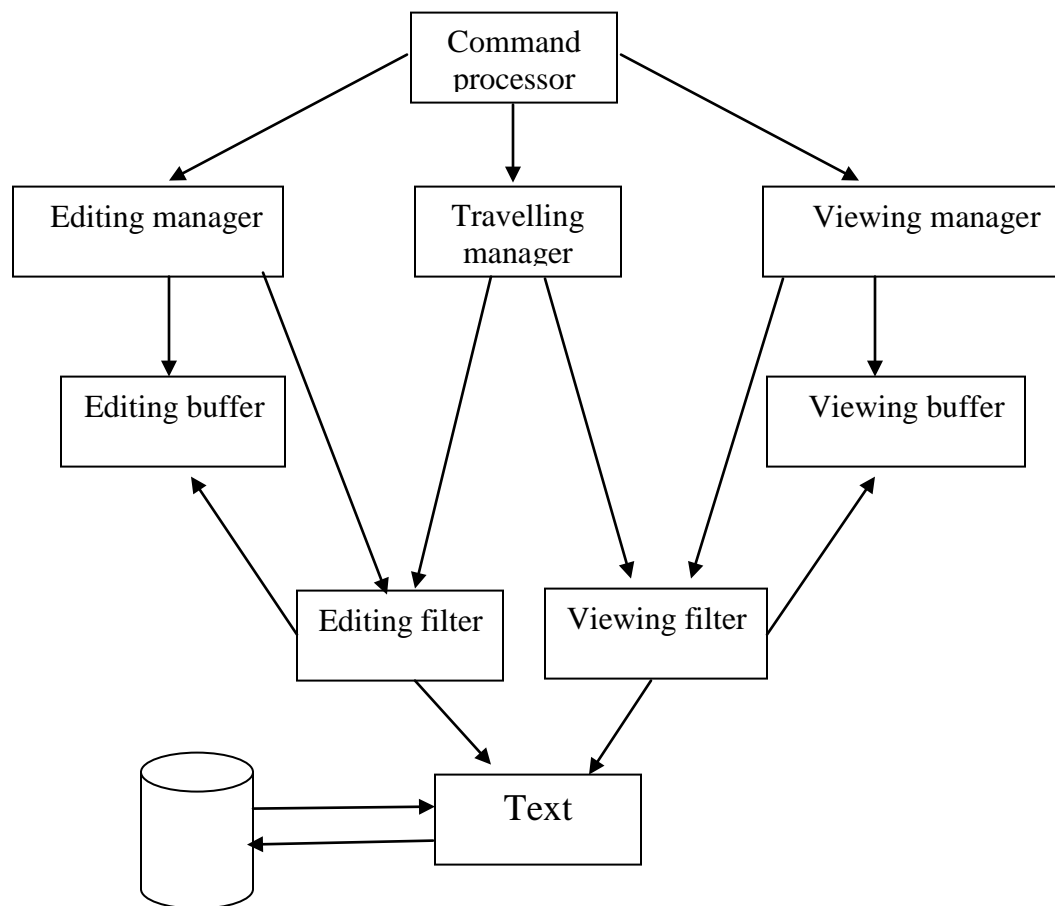
- An interactive editor is a computer program that allows a user to create or revise a target document.
- The document means an object such as computer programs, text, equations tables, diagrams, photographs etc.
- The document editing process in an interactive user computer dialogue designed to accomplish four tasks:
  - 1) Select the part of the target document to be viewed & manipulated.
  - 2) Determine how to format this view online and how to display it.
  - 3) Specify and execute operations that modify the target document.
  - 4) Update the view appropriately.
- Selection of the part of the document to be viewed and edited involves first travelling through the document to locate the area of interest.
- Travelling implies movement of the editing context to a new position within the text. The selection of what is to be viewed and manipulated that is controlled by filtering.
- Filtering extracts the relevant subset of the target document at the point of interest, such as the next screenful of text or the next statement.

- Formatting determines how the result of the filtering will be seen as a visible representation on a display screen or other devices.
- In the actual editing phase, the target document is created or altered with a set of operations such as insert, delete, replace, move and copy.

## Design of an Editor:

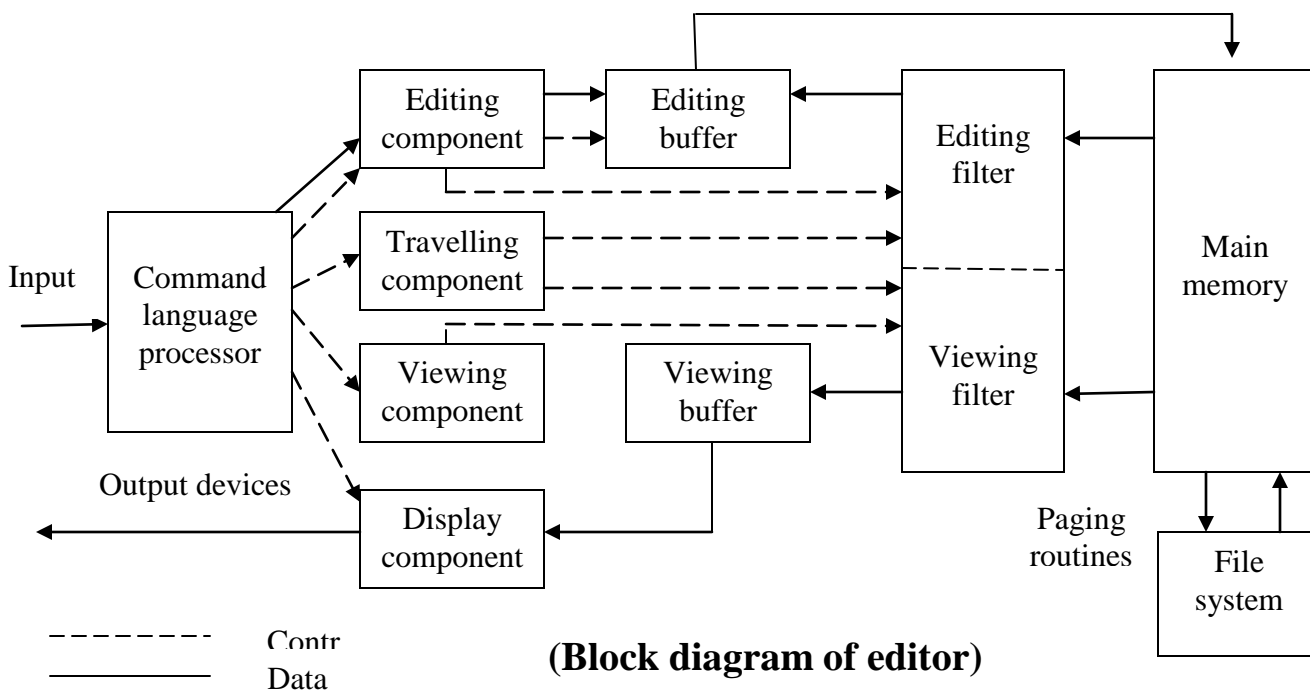
- The fundamental functions in editing are travelling, editing, viewing and display.
- Travelling implies movement of the editing context to a new position within the text.
- Viewing implies formatting the text in a manner desired by the user.
- A simple text editor' may choose *to* combine the viewing and display functions.

Figure below shows the simple editor.

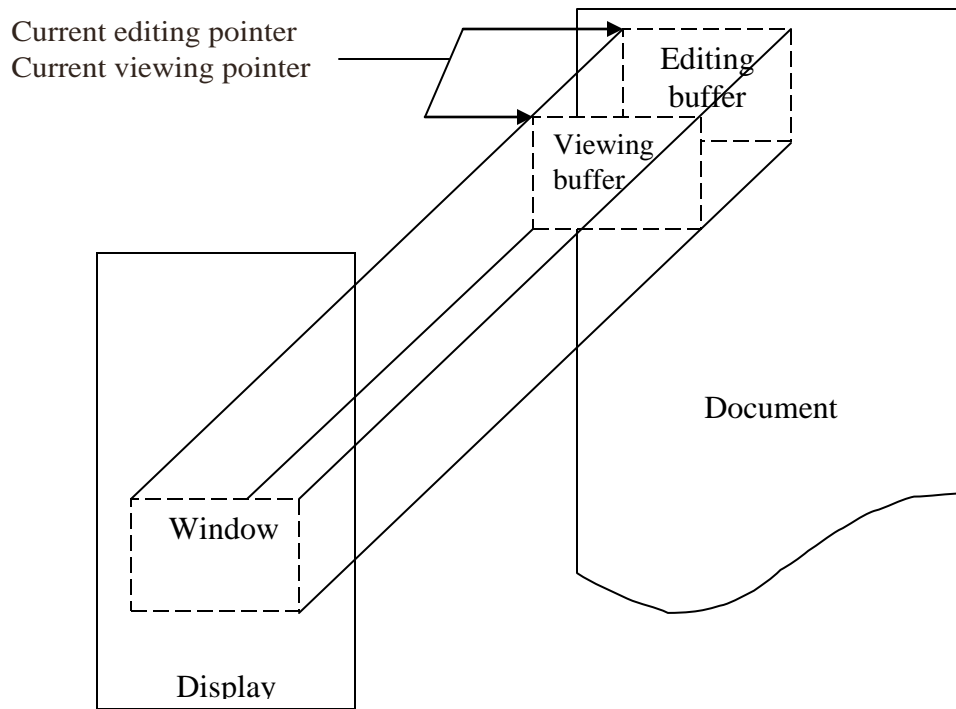


(Editor Structure)

- In editing document, the start of the area to be edited is determined by the current editing pointer maintained by the editing component.
  - The travelling component of the editor actually performs the setting of the current editing and viewing pointers. It also determines the point at which the viewing and editing filters.
  - The current editing pointer can be set or reset explicitly by the user with travelling commands, such as next paragraph and next screen.
  - When a user issue an editing command the editing component invokes the editing filter.
  - Editing component filters the document to generate a new editing buffer based on the current editing pointer as well as on the editing filter parameters.
  - Filtering may simply consists of the selection of contiguous characters beginning at the current point.
  - Current viewing pointer determined the start of the area to be viewed for viewing a document.
  - Current viewing pointer can be set or reset explicitly by the user with a travelling command or implicitly by the user with a travelling command or implicitly by the system as a result of the previous editing operation.
  - For a given position of the editing context, the editing and viewing filters operate on the internal form of text to prepare the forms suitable for editing viewing.
  - Figure below shows the block diagram of typical editor structure.



- When the cursor position changes, the filters operate on a new portion of text to update the contents of the buffers.
- As editing is performed, the editing filter reflects the changes into the internal form and update the content of the viewing buffer.
- The editing and viewing buffers are independent in some cases. They are identical sometime as shown is Figure below:



### (Editing and viewing buffer)

- The editing and viewing buffer can also partially overlap or one may be completely contained in the other.
- For example, the user might specify a search to the end of the document, starting at a character position in the middle of the screen.
- In the above example, an editing filter creates an editing buffer that contains the document from the selected character of the end of the document.
- The viewing buffer contains the part of the document that is visible on the screen. Only the last part of which is in the editing buffer.

### Screen Editors:

- Screen editor uses the what you see is what you get principle in editor design.
- A editor displays a screenful of text at a time. The user can move the cursor over the screen, position it at the point where user desires to perform some editing and proceed with the editing directly.

- The user has full control over the entire terminal. For example over an type exiting string which user wishes to replace. User can bring the cursor over a character to be deleted and press a delete key.
- It is possible to see the effect of an edit operation on the screen.

### **Line Editor:**

- Line editor is one of the simplest type of editor which uses a buffer to store information:
- It operates in command mode. User give the command to the editor for any operation. Editor will respond this command.
- Buffer is in the main memory. That is set aside to store the information which is entered from the keyboard.

### **Merits**

- 1)Simple for read and write.
- 2)Simple in design and implementation
- 3)Command and responses are interleaved.

### **Demerits**

- 1)Not user friendly
- 2)Context of the text is not displayed.
- 3)Only single line is used for editing.

### **Stream Editor:**

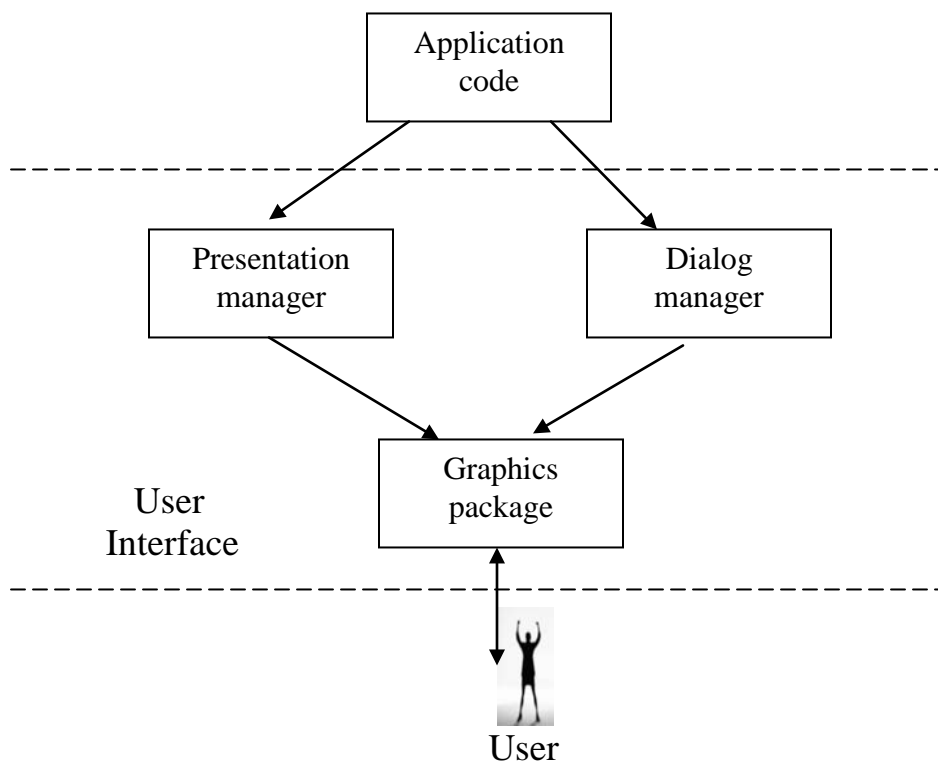
- A stream editor views the entire text as a stream of characters. This permits edit operations to cross line boundaries.
- Stream editor typically support character line and context orient commands.
- In stream editor, the current editing context indicated by the position of text pointer. This pointer can be manipulated using positioning.
- Stream editor maintain multiple representations.

### **Word Processors:**

- It is document editors with additional features. It produce well formatted output.
- Features of text
  - a)Merging of text
  - b)Searching and replacement of word
  - c)Moving sections of text

## User Interface:

- User interface simplify the interaction of a user with an application.
- UI consists of two components: Dialog manager and presentation manager.
- Dialog manager manage the conversation between the user and the application.
- Presentation manager displays the data produced by the application in an appropriate manner on the user display.
- Figure below show user interface.



### (User interface)

- Dialog manager is also responsible for error messages and online help function, and for organizing changes in the visual context of the user.
- User Interface Management System (UIMS) automates the generation of user Interface.
- UIMS takes specification of the presentation and dialog semantics to produce the presentation and dialog managers of the UI.

## Commands dialogs

- Command dialogues are used to issue commands.
- Command dialogues are implemented in following ways.
  - a) Command language.
  - b) Command menu.
  - c) Direct manipulation.
- User interface is concerned with the input devices output devices and interaction language of the system.
- Input devices are used to enter elements of the text being edited, to enter commands, and to designate editable elements. These devices as used with editors can be divided into the categories:
  - a) Text devices
  - b) Button devices
  - c) Locator devices
- Text or string devices are typically typewriter like keyboards on which a user presses and releases keys, sending a unique code for each key.
- Button or choice devices generates an interrupt or set a system flag. Such devices typically include a set of special function keys on an alphanumeric keyboard.
- Locator devices are two dimensional analog to digital converters that position a cursor symbol on the screen by observing the user's movement the device. The most common such devices for editing applications are the mouse and the data tablet.
- Text devices with arrow keys can be used to simulate locator devices. Each these keys shows an arow that point up, down, left or right.
- The interaction language of a text editor is generally one of several common types.
  - a) Typing oriented
  - b) Text command oriented
- The user communicates with the editor by typing text strings both for command names and for operands. These strings are sent to the editor and are usually echoed to the output device.
- Typed specification often requires the user to remember the exact form of all commands, or at least their abbreviations. Function key command specification is typically coupled with cursor key movement for specifying operands, which eliminates much typing.
- Typing oriented systems require familiarity with the system and language, as well as some expertise in typing. Function key oriented

systems often have either too few keys, requiring multiple keystroke commands, or have too many unique keys, which result in an unwieldy keyboard.

- The menu oriented user interface solve the problem of typing oriented an function key oriented systems. A menu is a multiple choice set of text strings or icons, which are graphic symbols that represents objects or operations. The user can perform actions by selecting items from the menu.
- Problem with a menu oriented system can arise when there are many possible actions and several choices are required to complete an action.

### **Debugging Function and Capabilities:**

- Debugging system should provide functions such as tracing and traceback.
- Tracing can be used to track the flow of execution logic and data modifications. It can also be based on conditional expressions.
- Traceback can show the path by which the current statement was reached. It can also show which statements have modified a given variable or parameter.
- Debugging system have a good program display capabilities. It must be possible to display the program being debugged, complete with statement numbers.
- The system should save all the debugging specifications across such a recompilation, so the user does not need to reissue all of these debugging commands.
- Debugging system must be sensitive to the specific language being debugged so that procedural, arithmetic and conditional logic can be coded in the syntax of that language.
- The debugger should be able to switch its context when a program written in one language calls a program written in a different language.
- A debugging system must be able to deal with optimized code. Application code used in production environments is usually optimized.

### **Debug Monitors:**

- Dynamic debugging facility is provided by the debug monitors. It includes following activities.
  - 1) Setting breakpoints in the program.

- 2) Initiating a debug conversation when control reaches a breakpoint.
  - 3) Displaying values of variables.
  - 4) Assigning new values to variables.
  - 5) Testing user defined assertions and predicates involving program variables.
- When the user a commands to set a breakpoint the debug monitor instruments the program to introduce a sensing instruction before the start of the statement at breakpoint.

- The debug monitor function can be easily implemented in an interpreter.
- When a user gives a command to set a breakpoint at statement 150, The debug monitor instruments the program to introduce the instruction.

`<SI-instrn><code>`

in place of the no-op instructions preceding no-op 150.

- The debug monitors requires two kind of information regarding the user program.
  - a) Starting address of program statements in the complied code.
  - b) Program variables name and address.

### **Dynamic debugging of a program**

- Following sequence of step involved .in dynamic debugging of a program.
- 1) The user compiles the program under the debug option. The compiler produces two file i.e.
    - a) Complied code file
    - b) Debug information file
  - 2) The user activates the debug monitor and indicates the name of program to be debugged. Debug monitor opens the corresponding two files.
  - 2) User indicates the debug requirements i.e. a list of breakpoints and action to performed at breakpoints. Debug monitor builds a debug table. This table \_contains, <statement number, debug action.>'
  - 3) The instrumented program gets the control and executes up to a breakpoint.
  - 2) When <SCinstrn> is excuted, software interrupt is generated and control is given to the debug monitor. Debug monitor construct the debug table and perform the debug actions associated with it.

- Control now returns to the u program.
- 3) Steps (4) and (5) are repeated until the end of the debug session.

### **Relationship with other parts of the system:**

- Important requirement for an interactive debugger is that it always be available. It must appear to be a part of the run-time environment and an integral part of the system. When an error is discovered, immediate debugging must be possible because it may be difficult or impossible to reproduce the program failure in some other environment or at some other time. So the debugger must communicate and cooperate with other operating system components such as interactive subsystems.
- Debugging is even more important at production time than it is at application development time. When an application fails during a production run, work dependent on that application stops. Since the production environment is often quite different from the test environment, many program failures cannot be repeated outside the production environment.
- The debugger must coordinate its activities with those of existing and future language compilers and interpreters. It is assumed that debugging facilities in existing languages will continue to exist and be maintained.

### **User Interface Criteria:**

- Most common complaint about debugging products is that they are something new to learn. This problem is solved by using simple organization and familiar in its language. The facilities of the debugging system should be organized into a few basic categories of function, which should closely reflect common user tasks. This type of simple organization contributes greatly to ease of training and ease of use.
- The user interaction should make use of full screen displays and windowing system as much as possible. The advantage of this type is that a great deal of information can be displayed and changed easily and quickly. With menus and full screen editors, the user has far less information to enter and remember.
- If the tasks a user needs to perform are reflected in the organization of menus, then the system will feel very natural to use. Menu should have titles that identify the task they help perform. Directions should precede any choices available to the user.

- Use of full screen displays and techniques such as menu is highly desirable. A debugging system should also support interactive users when a full screen terminal device is not present. It also provide complete functional equivalence between commands and menus.
- The command language should have a clear, logical simple syntax. It should also be as similar to the programming language as possible. Use simple command and instead of compound command. Parameters are as few as possible. Parameters should automatically be checked for errors in such attributes as type and range of values. Defaults should be provided for most parameters.
- Command formats should be as flexible as possible. The command language should minimize the use of such punctuation as parentheses, slashes, quotation marks and other special characters.
- Any good interactive system should have an online HELP facility.